SESSION

NOVEL APPLICATIONS + ALGORITHMS + SUPPORTING SYSTEMS

Chair(s)

TBA

Detecting Eating Using a Wrist Mounted Device During Normal Daily Activities

Yujie Dong[1], **Adam Hoover**[1], **Jenna Scisco**[2], **Eric Muth**[2]
[1]Department of Electrical and Computer Engineering, Clemson University, Clemson, SC, USA
[2]Department of Psychology, Clemson University, Clemson, SC, USA

Abstract— *The prevalence of obesity is a growing, worldwide health concern. Self-monitoring of eating consumption is widely recognized as a necessity for weight loss. In this paper we describe a novel method for automated monitoring of eating. Our method uses a single sensor that is worn on the wrist, similar in form to a watch. Wrist orientation was captured at a rate of 60 Hz for an entire day while four subjects conducted their natural daily routine. In our first experiment, we manually segmented the wrist motion data according to task logs kept by the subjects, and developed an algorithm to classify the tasks, achieving an accuracy of 91%. In our second experiment, we automatically segmented the wrist motion data in order to detect eating sessions, achieving a detection accuracy of 82%. Our methods will enable new opportunities in the study of dietetics, weight loss and management, nutrition, and health monitoring.*

Keywords: activity recognition, sensor, motion tracking, eating

1. Introduction

This work is motivated by the growing prevalence of obesity in the world. In 2007-2008, the National Health and Nutrition Examination Survey showed that 68.3% of Americans were overweight and 33.9% of Americans were obese [1]. The World Health Organization reported that 1.5 billion adults (age 20+) were overweight and 500 million adults were obese worldwide [2]. Obesity is strongly associated with several major health risk factors, such as diabetes, heart disease, high blood pressure, stroke and higher rates of certain cancers [3]. In the United States, the annual medical expense of obesity has been estimated at $147 billion in 2008 compared to $78.5 billion in 1998 [4].

Weight control can be assisted by self-monitoring of intake consumption, which has been consistently related to successful weight loss [5]. The most well known tool for monitoring food intake is an eating diary; however, this tool places the burden on the user to manually record all foods eaten. In addition, people have a tendency to forget or underreport the calorie consumption [6] [7]. Some researchers have investigated using a scale embedded in a dining table [8] [9]. However, this method can only monitor consumption when people eat at the instrumented table. Another method is to use a PDA or a cell phone to take photos before and after the eating and use image processing to estimate the amount of food intake [10] [11]. However, because foods must be carefully separated and positioned for imaging, these methods have not yet been studied in natural daily living. Combinations of neck, ear, arm, and back worn sensors have been investigated for recognizing eating activities [12]. While these configurations may find applications in a laboratory or clinic, they are not suitable for day-to-day living. In summary, none of the existing methods automates the process of self-monitoring of eating consumption in an easy-to-use manner.

Our group has previously described methods using a micro-electro-mechanical system (MEMS) sensor to track wrist motion in order to measure the number of bites eaten during a meal [13]. We have discovered that while eating, the wrist motion of a person undergoes a characteristic rolling motion that is indicative of the person taking a bite of food [14]. However, our device requires the user to press a button to turn the device on before eating and turn the device off after eating. In this paper, we explore methods to overcome this limitation by differentiating eating sessions from other activities using the same MEMS sensor.

With their low power and small size, MEMS sensors can be comfortably worn on the human body and operated for hours at a time. Researchers have investigated their use for recognizing common daily activities such as walking, running, sitting and resting [15] [16] [17] [18], accidental falls [19], sports activities [20], assembly tasks [21], and tremors associated with Parkinson's disease [22]. Sensors can be placed on different parts of the body, such as the chest [15], shoulder [19], waist [18] [19], thigh [19], ankle [20], hip [17] and wrist [17]. The sensor type varies as well. The most common type is accelerometers [15] [18] [19] [20] [21] [23], while ECGs [16] [17], light sensors [19], microphones [19] [21] and temperature sensors [17] have also been used.

None of these works has considered the problem of detecting eating activities during normal daily life. To our knowledge, the methods we describe herein are the first to look at this difficult problem. In addition, many of the previous works on activity recognition require a large set of sensors [17] [19] [23], that together with the wiring, are difficult to wear outside the laboratory. Experiments are typically performed in a laboratory setting where subjects are

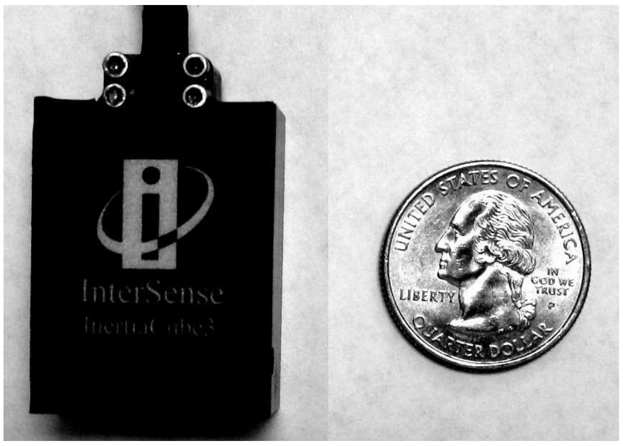


Fig. 1: InertiaCube3 prototype

asked to repeat activities of interest, interspersed with other motions [15] [16] [17] [20]. In contrast, we instrumented our subjects with a single sensor and instructed them to conduct normal activities for an entire day. While the results presented in this paper are preliminary and on a limited number of subjects, we believe our methods will ultimately enable new opportunities for weight management and weight loss paradigms.

The rest of the paper is organized as follows: In section 2 we describe our approach of classifying eating activity on pre-segmented motion data and detecting eating sessions in real time. In section 3 we present experimental results to validate our proposed algorithms. Finally, we conclude our paper and discuss future work in section 4.

2. Methods

2.1 Hardware and prototype

A wired InertiaCube3 sensor produced by InterSense Corporation (InterSense, Inc., 36 Crosby Drive, Suite 150, Bedford, MA 01730, www.isense.com) was used to record the wrist motion data. It is composed of an accelerometer, a gyroscope and a magnetometer on each of the three axes which provide an orientation heading in each of these three orientations: roll, pitch, and yaw. Figure 1 shows the wired Inertiacube3 sensor and its size compared to a US quarter. The sensor was connected to an external 9V battery as a power source and a laptop with a running program to store collected data through an RS232 interface. Both the external battery and the laptop were carried by the subject in a backpack. The adjustable wire connecting the two parts was long enough to make sure the subject's normal behaviors were not being affected.

2.2 Data collection

Subjects were asked to wear the sensor and carry the backpack to record their wrist motion data when they got

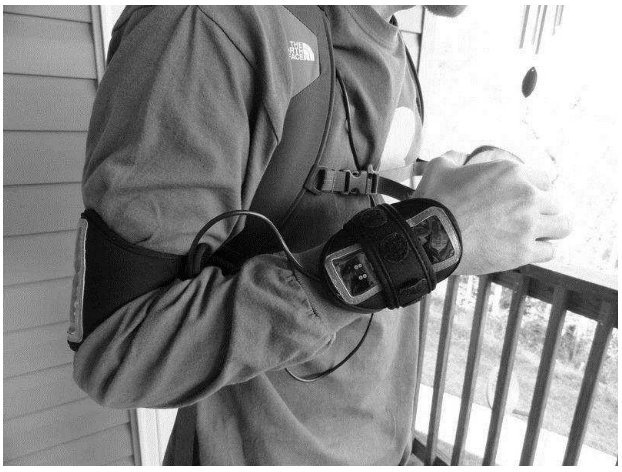


Fig. 2: Data collection using a single orientation sensor on the wrist

Fig. 3: Data collection using a single orientation sensor on the wrist

up in the morning, and to stop recording the data when they went to bed at night. As shown in Figure 2, the subject placed the sensor on the dominant eating hand, and then wrapped the band tightly around the forearm to ensure it would not slide around the arm. The program running on the laptop in the backpack (Figure 3) was set up to collect the orientation data from the sensor in real time.

Using the recording program on the laptop was straightforward. Double clicking the program icon on the desktop would automatically start it to record the pitch, yaw and roll orientation at a rate of 60Hz. Due to the fact that the

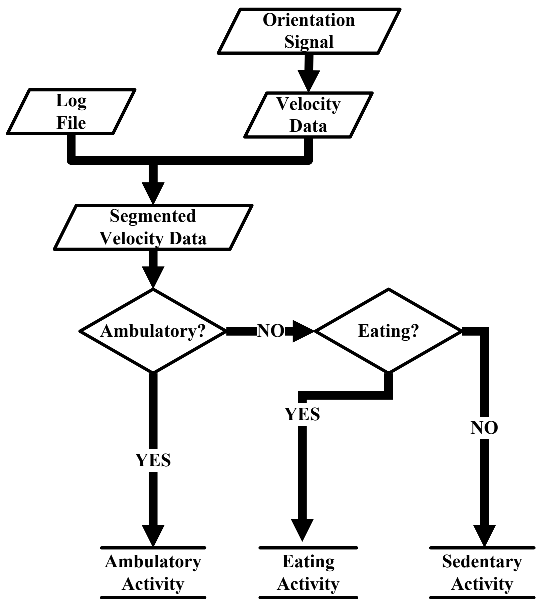


Fig. 4: Diagram of offline detection

battery in the laptop could only last for about four hours, the program generated continuous beeping for 3 minutes when the battery level of the laptop dropped to 10%. The subject was asked to close the program and replace the battery (an extra was provided in the backpack) when he or she heard the beeping reminder. He or she was asked to restart the program afterward to continue recording.

During recording, subjects were asked to conduct daily activities as naturally as possible. The subject was asked to remove the device when engaging in activities which would damage the device, such as taking a shower or playing contact sports. The subject was asked to record activity behaviors in a written log book. The subject was asked to record the start time and the name of the activity for each new task. For example, 08:02:04 eating; 13:24:58 walking. A task was defined as a piece of work or activity to be finished. The log information written by the subject was used for segmenting the ground truth tasks from the wrist motion data later.

A total of 4 subjects participated in this experiment. Two were male and two were female. All the data was collected in a completely free living condition, with no supervision.

2.3 Offline eating classification

For our first experiment, we consider the problem of classifying the eating activity using both the collected motion data and time information in the log book. An outline of the process for the offline eating classification is shown in Figure 4.

Since different subjects might wear the sensor at different angles, it is difficult to define the task if we use the absolute value of the orientation data. Therefore, we calculate the derivative data, which is comparable. Since we have recorded the data at 60Hz, the simplest way to calculate the derivative data is in Equation 1 where d_t is the derivative data at time t and o_t is the orientation data at time t.

$$d_t = (o_t - o_{t-1}) \times 60 \qquad (1)$$

The second step is to segment the derivative data into tasks based on the log file. In the log file, subjects recorded the start time for each new task. We use the start time of current task as one boundary and the start time of the next task as the other boundary for the current task to segment the derivative data. For each segmented data, we categorize it based on the content in the log file into one of 23 categories, as shown in Table 1.

Table 1: activity category

Eating activity	**Sedentary activity**	**Ambulatory activity**
Eating	Using computer	Cooking
	Using phone	Walking
	Reading	Driving
	Writing	Washing dishes
	Napping	Cleaning
	Talking	Doing laundry
	Watching TV	Packing
	Changing laptop battery	Brushing
	Filing nail	Shopping
	Playing card game	
	Going to restroom	
	Being passenger in car	
	Playing video game	

Although we were able to map most user defined tasks into Table 1, a few tasks were difficult to categorize. First, two categories may happen at the same time, such as eating apples and working on a computer. Second, different people can make notes on the same activity in different ways. For instance, some subjects may categorize “walk to car, stop to talk to a friend” as one log entry, but some other subjects put it into two categories. Because we are interested in eating activities, any note with eating is categorized as “eating”. Any notes without eating mentioned were categorized to the best of our ability.

Since eating is the most important activity to us, we do not need to classify all these 23 tasks. We cluster these 23 categories into three clusters:

1) Eating activity: eating activity is a task which related to eating food or drinking liquid.
2) Sedentary activity: sedentary activity is a task (except eating) which involves sitting down, not moving or not exercising. All tasks in the middle column of Table 1 belong to this category.
3) Ambulatory activity: ambulatory activity is a task which is related to walking, moving or exercising. All

tasks in the right column of Table 1 belong to this category.

These clusters were chosen because it is typically easier to distinguish sedentary and ambulatory activities. Once these have been separated, eating activities can be recognized as a subset of sedentary activities.

To classify the segmented tasks, we calculate five features for each task:

1) Variance of yaw velocity (Y_VAR)
2) Variance of pitch velocity (P_VAR)
3) Variance of roll velocity (R_VAR)
4) Bites per minute (BPM) using bite detection method. The method to detect bite counts using the derivative data is described in our previous work [13].
5) Occurrences when the bite detection method does not detect a bite over a span of at least one minute (NOT_EAT).

Using these features, each task is classified as follows:

1) A task is classified as an ambulatory activity if any of the following conditions are met:
   a) Y_VAR + P_VAR + R_VAR $>$ T1
   b) Y_VAR $>$ T2
   c) P_VAR $>$ T3
   d) R_VAR $>$ T4
2) A task is classified as an eating activity if all of the following conditions are met:
   a) Y_VAR $<$ T5 and Y_VAR $>$ T6
   b) P_VAR $<$ T7 and P_VAR $>$ T8
   c) R_VAR $<$ T9 and R_VAR $>$ T10
   d) BPM $>$ T11
   e) NOT_EAT $<$ T12
3) Otherwise a task is classified as a sedentary activity

Here, {T1, T2, ... T12} is a set of thresholds. In our default setting, these values are {8500, 5000, 1000, 5000, 3000, 200, 900, 150, 5000, 600, 2, 3}. If the variance of the task's velocity data is large, it is considered as an ambulatory task. If this criterion is not met, the task is considered as either an eating task or a sedentary task. The eating activity has the following characteristics: the variance of pitch, yaw and roll should be within a certain range. In addition, the eating activity should have reached certain bite counts per minute and should not include a long period where no bite is detected. These characteristics are used to separate eating tasks from sedentary tasks.

2.4 Real time eating detection

Our second experiment considers the problem of detecting eating activity without knowing the start time of each task in the log file. This method has the potential to detect the eating activity in real time as we collect the data. The outline of our method is shown in Figure 5. In our algorithm, we only use the roll orientation data.

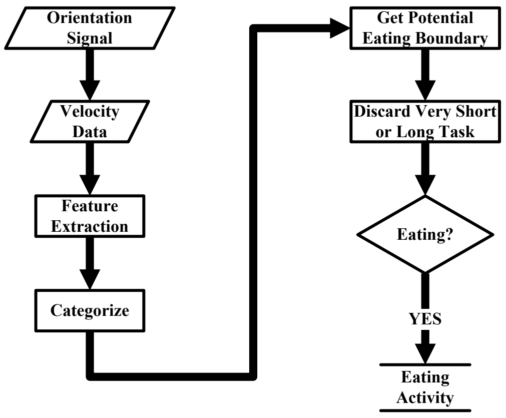


Fig. 5: Diagram of real time detection

We calculate the roll velocity data from the orientation data, same as in section 2.3. To identify eating activity in real time, we use a sliding window to extract the motion feature. The window size is set to 10 minutes and we update the motion feature every 1 minute. For each 10 minute window, the data is segmented into 2 parts, one from $t-10$ to $t-5$ and one from $t-5$ to t. Each of these two parts is classified using the methods outlined in section 2.3. Based on these classifications, the point at time $t-5$ is categorized as one of 4 categories:

1) category 1: this point may be a start boundary for an eating task.
2) category 2: this point may be an end boundary for an eating task.
3) category 3: this point cannot be inside an eating task.
4) category 4: this point may be inside or outside an eating task.

Figure 6 illustrates a state machine that shows our method. Initially we are in the state “not eating”. After that, we update the transition condition (category of the time stamp) every 1 minute. If the transition condition is category 1, the state transits to “possibly eating”, at the same time, we update the potential start time of an eating session. While in the state “possibly-eating”, if the transition condition is category 1, we update the start time; if the transition condition is category 3, we go back to state “not eating”; if the transition condition is category 2, we have detected a potential eating session. We output the start time and the end time of the potential eating session and go back to state “not eating”.

For every potential eating session, we examine the duration. If the duration is too short or too long, it is not to be considered as an eating activity.

We also extract the features and run the same algorithm illustrated in section 2.3 to classify the potential eating

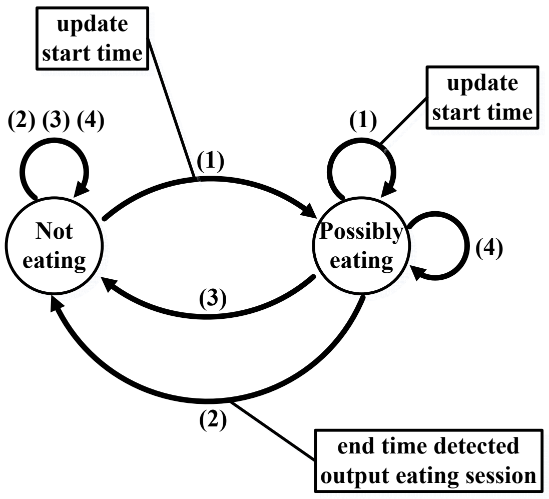


Fig. 6: State machine of potential eating session detection

session. If all criteria are met, an eating session is detected.

3. Results

As described in Section 2.2, a total of 4 subjects were recorded, each for an entire day. We had no restriction on how subjects should do their activities and how long they should do each task. For any eating task, subjects could eat their own food and liquid, and use any utensils they preferred (hand, spoon, fork, or chopsticks). Each recording session was completely unsupervised.

Table 2 shows some statistics of all the tasks for these subjects. The total time recorded for these four subjects ranged from 9.4 hours to 13.4 hours. The total number of tasks for each subject was between 23 and 39. Table 2 also shows the shortest task duration, longest task duration, average task duration, and standard deviation of task duration for each subject. In addition, the most frequent task for different subjects varied.

We also include the statistics of eating tasks for these 4 subjects in Table 3. The total eating time of each subject was from 0.7 hour to 1 hour. This was consistent with the “American Time Use Survey” from the United States Bureau of Labor Statistics [24] which reported an average of 1.18 hours on eating and drinking per weekday. The total number of eating tasks was within a range from 4 to 6 times. Table 3 also shows the shortest eating session, longest eating session, average eating session, and standard deviation of eating session for each subject.

Table 4 shows the results of task classification using the information on the log file. There were 125 total tasks across all 4 subjects; 16% of the tasks were eating activity, 43% of the tasks were sedentary activity, and the rest were ambulatory activity. The classification accuracy is calculated using Equation 2. In our experiment, the classifcation accuracy was 91%.

$$accuracy = \frac{sum\ of\ correct\ classifications}{total\ number\ of\ classifications} \tag{2}$$

Table 5 shows the results of real time eating activity recognition without knowing any information in the log file. In the table, the second and the third column show the ground truth time of each eating task. The second column shows the start time of the eating task and the third column shows the end time of the corresponding eating task. The fourth column and the fifth column show the computer detected boundary for each eating task. The fourth column shows the detected start time of each eating task and the fifth column shows the detected end time of the corresponding eating task. All of these numbers are in minutes. A row without any number in the fourth and fifth column indicates that there is an undetected eating task. A row without any number in the second column and third column indicates that there is a false detection of an eating task. A row with numbers in all columns indicates that this is a detected eating task. The sensitivity is calculated using Equation 3 and the positive predictive value (PPV) is calculated using Equation 4.

$$sensitivity = \frac{true\ detected}{true\ detected + undetected} \tag{3}$$

$$PPV = \frac{true\ detected}{true\ detected + false\ detected} \tag{4}$$

Although there were a total of 20 eating sessions recorded by the subjects, 3 of them lasted for less than 3 minutes so they were not included in Table 5. We excluded these tasks because they were so short that our feature set did not adequately describe them. For the remaining 17 eating tasks, 3 of them were not detected. There were 6 false detections. Thus the sensitivity was 82% and the positive predictive value was 70%. In addition, for the 14 eating sessions detected, 10 of them were detected with start and end boundaries which match the log file within 2 minutes. For the other 4 sessions, the boundary errors are (0, 5), (1, 3), (4, 9), (3, 1) minutes respectively. We hypothesize that these boundary errors are likely due to timing misalignments between the user logs and wrist motion data, as well as judgment calls by the subjects as to when they actually started and stopped eating.

4. Conclusions

The prevalence of obesity is a growing, worldwide health concern. Self-monitoring of eating consumption is widely recognized as a necessity for weight loss. However, there are currently no automated methods for monitoring eating consumption in natural daily living. In this paper we have

Table 2: statistics of all the tasks for these subjects

	Subject 1	Subject 2	Subject 3	Subject 4
total time of all tasks (h)	13.4	9.8	10.1	9.4
total number of tasks	39	36	23	27
shortest task (min)	3	1.2	3.3	3.5
longest task (min)	93.3	90.7	97.5	78.7
average task (min)	20.6	16.3	26.3	20.9
standard deviation of task (min)	20.6	19.1	27.4	21.7
most frequent task	Using computer	Driving	Eating	Using computer

Table 3: statistics of eating tasks for these subjects

	Subject 1	Subject 2	Subject 3	Subject 4
total time of eating (h)	0.8	0.7	1	0.8
total number of eating sessions	5	5	6	4
shortest session (min)	5.8	1.2	3.3	7
longest session (min)	18.6	15.3	12.3	21.7
average session (min)	10.2	7.9	10	12.6
standard deviation of session (min)	5.2	6	3.6	6.5

Table 4: offline classification result

	Classify: Eating	Classify: Sedentary	Classify: Ambulatory
GT: Eating	17	2	1
GT: Sedentary	4	49	1
GT: Ambulatory	2	1	48

Table 5: real time classification result (minutes)

	Ground Truth		PC Detect	
Subject	Start time	End time	Start time	End time
S1	11	17	9	17
S1	195	205	194	205
S1	393	400	393	399
S1	537	547	537	549
S1	654	673	653	674
S1			685	700
S2	78	91	78	96
S2	523	538		
S2	565	573		
S2			100	112
S2			192	201
S2			538	548
S3	85	94	85	94
S3	166	178	166	179
S3	257	269	258	272
S3	412	424	412	426
S3	603	615	601	615
S3			362	370
S4	14	27		
S4	270	277	270	276
S4	462	484	466	475
S4	518	527	515	528

described preliminary experiments that use a single wrist-worn sensor to track wrist motion throughout the day, in order to detect eating sessions. Four subjects were recorded for an average period of 11 hours, performing an average of 31 self-classified tasks, of which an average of 5 were eating. In our first experiment, we segmented the wrist motion data according to the subjects' logs, and demonstrated a 91% accuracy in classifying the tasks. In our second experiment, we automatically segmented the wrist motion data and demonstrated an 82% accuracy in detecting eating sessions. While the number of subjects tested was small, this is the first work to examine the problem of automatically monitoring eating during daily living. In the future we plan to continue this experiment on a much larger number of subjects. We also intend to simplify the apparatus to something that can be worn completely on the wrist. For these first experiments, we used a laptop in a backpack in order to record the large amount of data generated during an entire day. For our next experiments we intend to use a "smart phone".

References

[1] K. Flegal, M. Carroll, C. Ogden and L. Curtin, "Prevalence and trends in obesity among US adults, 1999-2008", in *Journal of the American Medical Association*, vol. 303, 2010, pp. 235-241.

[2] World Health Organization Media Center, ”Obesity and overweight”, http://www.who.int/mediacentre/factsheets/fs311/en/index.html, retrieved March 5, 2011.

[3] N. Wellman and B. Friedberg, “Causes and consequences of adult obesity: health, social and economic impacts in the United States”, in *Asia Pacific Journal of Clinical Nutrition*, vol. 11, no. 667-751, 2002, pp. 705-709.

[4] E. Finkelstein, J. Trogdon, J. Cohen and W. Dietz, “Annual medical spending attributable to obesity: Payer- and service-specific estimates”, in *Health Affairs*, vol. 28, 2009, pp. w822-w831.

[5] L. Burke, J. Wang and M. Sevick, “Self-monitoring in weight loss: A systematic review of the literature”, in *Journal of the American Dietetic Association* vol. 111, no. 1, 2011, pp. 92-102.

[6] C. Champagne, G. Bray, A. Kurtz, J. Monteiro, E. Tucker, J. Volaufova and J. Delany, ”Energy intake and energy expenditure: A controlled study comparing dieticians and non-dieticians”, in *Journal of the American Dietetic Association*, vol. 102(10), 2002, pp. 1428-1432.

[7] K. Glanz, J. Brug and P. van Assema, “Are awareness of dietary food intake and actual fat consumption associated? A Dutch-American Comparison", in *Euro. J. of Clinical Nutrition*, vol. 51, 1997, pp. 542-547.

[8] H. Kissileff, G. Klingsberg and T. Van Itallie, “Universal eating monitor for continuous recording of solid or liquid consumption in man”, in *Amer. J. of Physiology*, vol. 238 no. 1, 1980, pp. R14-R22.

[9] K. Chang, S. Liu, H. Chu, J. Hsu, C. Chen, T. Lin, C. Chen and P. Huang, “The diet-aware dining table: Observing dietary behaviors over a tabletop surface,” in the proc. of *4th Int’l Conf. on Pervasive Computing*, LNCS vol. 3968, 2006, pp. 366-382.

[10] F. Zhu, A. Mariappan, C. Boushey, D. Kerr, K. Lutes, D. Ebert and E. Delp, “Technology-Assisted Dietary Assessment”, in *SPIE: Computational Imaging VI*, vol. 6814, 2008, pp. 1-10.

[11] R. Weiss, P. Stumbo and A. Divakaran, “Automatic Food Documentation and Volume Computation using Digital Imaging and Electronic Transmission”, in *J. of American Dietetics Assoc.*, vol. 110 no. 1, 2010, pp. 42-44.

[12] O. Amft and G. Troster, “On-Body Sensing Solutions for Automatic Dietary Monitoring”, in *Pervasive Computing*, vol. 8(2), 2009, pp. 62-70.

[13] Y. Dong, A. Hoover, and E. Muth, "A Device for Detecting and Counting Bites of Food Taken by a Person During Eating", in the proc. of *IEEE Conf. on Bioinformatics and Biomedicine*, 2009, pp. 265-268.

[14] Y. Dong, "A Device for Detecting and Counting Bites of Food Taken by a Person During Eating", master’s thesis, Electrical & Computer Engineering Dept., Clemson University, 2009.

[15] A. Khan, Y. Lee, S. Lee and T. Kim, “A Triaxial Accelerometer-Based Physical-Activity Recognition via Augmented-Signal Features and a Hierarchical Recognizer”, in *IEEE Trans. on Information Technology in Biomedicine*, vol. 14, no. 5, Sept 2010, pp. 1166-1172.

[16] T. Pawar, S. Chaudhuri and S. Duttagupta, “Body Movement Activity Recognition for Ambulatory Cardiac Monitoring”, in *IEEE Trans. on Biomedical Engineering*, vol. 54, no. 5, May 2007, pp. 874-882.

[17] M. Ermes, J. Parkka, J. Mantyjarvi and I. Korhonen, “Detection of Daily Activities and Sports With Wearable Sensors in Controlled and Uncontrolled Conditions”, in *IEEE Trans. on Information Technology in Biomedicine*, vol. 12, no. 1, Jan 2008, pp. 20-26.

[18] D. Karantonis, M. Narayanan, M. Mathie, N. Lowell and B. Celler, “Implementation of a Real-Time Human Movement Classifier Using a Triaxial Accelerometer for Ambulatory Monitoring”, in *IEEE Trans. on Information Technology in Biomedicine*, vol. 10, no. 1, Jan 2006, pp. 156-167.

[19] J. Yin, Q. Yang and J. Pan, “Sensor-Based Abnormal Human-Activity Detection”, in *IEEE Trans. on Knowledge and Data Engineering*, vol. 20, no. 8, Aug 2008, pp. 1082-1090.

[20] J. Parkka, L. Cluitmans and M. Ermes, “Personalization Algorithm for Real-Time Activity Recognition Using PDA, Wireless Motion Bands, and Binary Decision Tree”, in *IEEE Trans. on Information Technology in Biomedicine*, vol. 14, no. 5, Sept 2010, pp. 1211-1215.

[21] J. Ward, P. Lukowicz, G. Troster and T. Starner, “Activity Recognition of Assembly Tasks Using Body-Worn Microphones and Accelerometers”, in *IEEE Trans. on Pattern Analysis & Machine Intelligence*, vol. 28, no. 10, Oct 2006, pp. 1553-1567.

[22] A. Salarian, H. russmann, C. Wider, P. Burkhard, F. Vingerhoets and K. Aminian, “Quantification of Tremor and Bradykinesia in Parkinson’s Disease Using a Novel Ambulatory Monitoring System”, in *IEEE Trans. on Biomedical Engineering*, vol. 54, no. 2, Feb 2007, pp. 313-322.

[23] L. Atallah, B. Lo, R. Ali, R. King and G. Yang, “Real-Time Activity Classification Using Ambient and Wearable Sensors”, in *IEEE Trans. on Information Technology in Biomedicine*, vol. 13, no. 6, Nov 2009, pp. 1031-1039.

[24] United States Bureau of Labor Statistics, American Time Use Survey, "Eating and Drinking", http://www.bls.gov/tus/current/eating.htm retrieved March 5, 2011.

A Novel Sphygmogram Sampling and Self-Adjusting Scheme for e-Home Healthcare

Wei Xuan Fang[1], **Ming Chui Dong**[1], **Wai Kei Lei**[2], **Xiang Yang Hu**[3]
[1]Faculty of Science and Technology, University of Macau, Macau SAR, China
[2]Institute of Systems and Computer Engineering, Macau SAR, China
[3]The 5th Affiliated Hospital of Sun Yat-sen University, ZhuHai, China

Abstract - *Pulse wave transmit time method has been used to estimate blood pressure by simultaneously measuring electrocardiogram & pulse signals* [1]. *Most researchers use photo reflective sensor to capture photoplethysmograph (PPG) signal by attaching sensor on finger tip. However, such a way would interfere hands in operating further the e-home healthcare system, plus PPG has flatter morphological shape* [2] *which is not adequate for searching feature points. Thus the piezoelectric ceramics is selected to acquire sphygmogram (SPG) signal with sharp morphological shape from wrist. Using such a way can free fingers to operate the e-home healthcare system. A SPG sampling scheme with signal conditioning circuit and relevant software for realizing signal amplitude and baseline-shift self-adjustment are proposed in this paper. A close-loop control is constructed between computer and micro control unit (MCU) such that to acquire the self-adjusted stable SPG signal. The testing results show out superior features of this scheme.*

Keywords: e-Home Healthcare, SPG, Signal Conditioning Circuit, Close-loop Control, Amplitude and Baseline-shift Self-adjustment

1 Introduction

Pulse signal can be sampled from different positions on human body, such as finger tip, wrist, chest, leg or any place with underneath carotid and radial artery. Since photo reflective sensor has been developed in recent years, most researchers choose finger tip as measurement position [3-6], such as ring-type PPG signal measurement device proposed by Chinese Univ. of Hong Kong. However, such a way is uncomfortable due to the space between fingers is limited for a ring-type device, which contains signal conditioning and wireless communication circuits and battery. Moreover, PPG has flatter morphological shape which is not adequate for searching feature points. Selecting other positions, such as chest and leg cannot get stronger signal, plus sticking the sensor on skin is even more uncomfortable. Alternatively, position on wrist has strong pulse signal which can be easily found out by most people, plus the SPG has sharp morphological shape which is good for searching feature points accurately, better still entire measurement device can be miniaturized as a watch-type, thus has extensive application foreground in e-home healthcare. Consequently, location of radial artery is selected for pulse acquisition in this paper.

The hospital used medical instruments having SPG acquisition function from wrist generally is large and the price is too expensive for home user. In addition, they need professional to adjust system parameters and record signal, eliminate existing external disturbance during SPG acquisition. To tackle such a problem, a home used SPG sampling scheme with self-adjusting technology is proposed in this paper, which can record a stable SPG waveform and transmit it to computer through universal serial bus (USB). Following content firstly introduces scheme structure and depicts each module; then explains the designed signal conditioning circuit and shows software flowchart for signal control & transmission; finally presents signal amplitude and baseline-shift self-adjustment method for SPG acquisition.

2 Sampling and self-adjusting scheme

A filmy passive piezoelectric transducer with 3.5cm diameter and 0.5mm thick is constructed as SPG acquisition sensor which transfers mechanical oscillation to electrical signal through piezoelectric effect [9]. His allowed pressure range is -500~5000mmHg with sensitivity of 2000μV/mmHg. Elastic band is used to attach transducer on wrist.

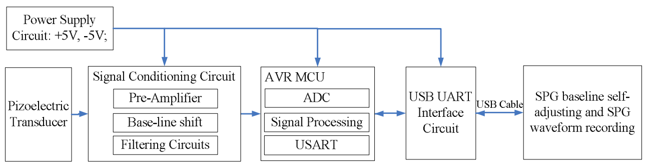
As shown in Fig. 1, the scheme is consists of six functional modules. The piezoelectric transducer transfers pulse signal to electrical waveform. Through signal conditioning circuit the amplitude of this SPG signal is processed as one within analog-to-digital (ADC) required range 0~5V. The signal conditioning circuit includes pre-amplifier, baseline-shift and filtering circuits. The refractory missions of this module are that greatly reduce signal phase delay to less than 40 degree, and keep the signal to noise ratio (SNR) being larger than 10dB. After that, the analog SPG is digitized in MCU module by using ATMEGA88V, which contains six 10bit successive-approximation-type ADC input channels. MCU module is also designed as signal processing and transmission unit since it supports simple math

This work was supported by Science and Technology Development Fund (FDCT) of Macau S.A.R with project ref. No. 018/2009/A1: "Embedded-Link Monitoring Platform for Cardiovascular Healthcare"

calculation and has two programmable USARTs (universal synchronous asynchronous receiver transmitter) [10].

Output signal of MCU is sent to USB interface module through USART, where the latest device FT232R is selected. Software is constructed to graph and analyze the digitized SPG waveform on computer, in the meanwhile it sends information of waveform amplitude and baseline back to MCU to adjust digitizing SPG signal. This close-loop feedback endows scheme with amplitude and baseline-shift self-adjusting capability which helps to stabilize SPG waveform.

The onset point of SPG signal is lower than 0V, which indicates that the operational amplifier needs ±5V power supply, plus the ADC's reference voltage is 5V, using Max1680 and through USB port, computer provides such required powers.



Power Supply Circuit: +5V, -5V;
Pizoelectric Transducer
Signal Conditioning Circuit
Pre-Amplifier
Base-line shift
Filtering Circuits
AVR MCU
ADC
Signal Processing
USART
USB UART Interface Circuit
USB Cable
SPG baseline self-adjusting and SPG waveform recording

Fig. 1 Structure of Sampling and Self-Adjusting Scheme

3 Signal conditioning circuit

The transducer sampled SPG signal is -400mV~0.9V, in which often exist high-frequency interference noises and the baseline-shift affected by tightness of elastic band. The signal conditioning circuit processes SPG signal as one within ADC required input range and filters out noises. Multisim 8 is used to analyze circuit performance which offers bode plot and distortion analysis.

As shown in Fig. 2 and 3, a 1st order high pass filter (HPF) with 0.0008Hz cut-off frequency and large loading impedance (20MΩ) is designed to reduce signal's DC offset. The buffer circuit offers high input impedance and low output impedance. Then a pre-amplifier is added to increase SPG signal amplitude and SNR. Subsequently, a summing circuit is designed to shift signal minimum points to above 0V. Due to noise amplitudes are also increased after using pre-amplifier and summing circuit, a low pass filter (LPF) with 40.8Hz cut-off frequency is designed to reduce noises. Finally, ADC buffer is used to provide low output impedance.

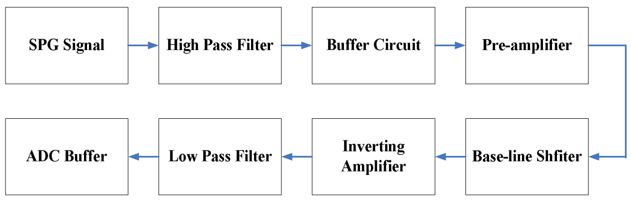


Fig. 2 Functional Diagram of Signal Conditioning Circuit

The frequency of SPG signal varies from 0.03Hz to 40Hz [8], thus a 10Hz sinusoidal signal with 0.6V offset and 1V peak-to-peak amplitude is used for simulation in Multisim. Eq. (1) determines HPF resistive and capacitive values. In Fig. 3, buffer circuit adds equivalent resistors to inverting and non-inverting nodes which compensate voltage drop caused by bias current and reduce total harmonic distortion (THD) by 0.017%.

$$f_c = \frac{1}{2\pi RC} \tag{1}$$

where f_c is cut off frequency, R and C are corresponding resistor and capacitor in HPF circuit.

The maximum peak-to-peak output voltage of operation amplifier (TL064) is 8V and the amplitude of SPG signal is about 1V. To satisfy ADC required range 0~5V and increase SNR, SPG signal is amplified by gain 3.35 so that let its amplitude be close to 4V. Eq. (2) determines resistors values in pre-amplifier circuit.

$$A_o = \frac{R_{38} + R_{39}}{R_{34}} \tag{2}$$

where A_o is gain, R_{38}, R_{39} & R_{34} are corresponding resistors in pre-amplifier circuit.

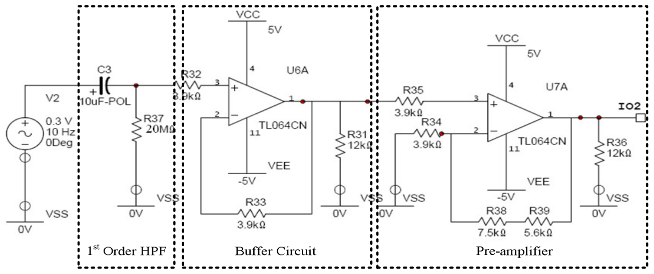


Fig. 3 HPF, Buffer and Pre-amplifier Circuits

After pre-amplifier, the negative amplitude of SPG signal becomes -1.34V with baseline located at 0V and noise amplitude is increased from 180mV to 460mV. Thus a summing circuit and an inverting amplifier are integrated together as shown on Fig. 4 to shift up baseline about 1.54V. Fig. 5 shows LPF and ADC buffer circuits. Through a 1st order LPF, noises are further reduced. All determined parameter values, that defined through simulation first and further adjusted by hardware experiment later are clearly marked in Fig. 3~5.

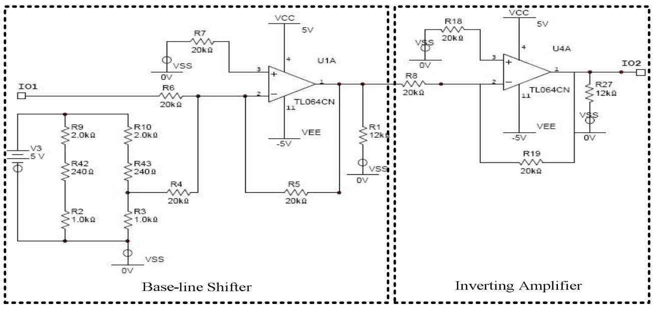


Fig. 4 Summing Circuit and Inverting Amplifier Circuits

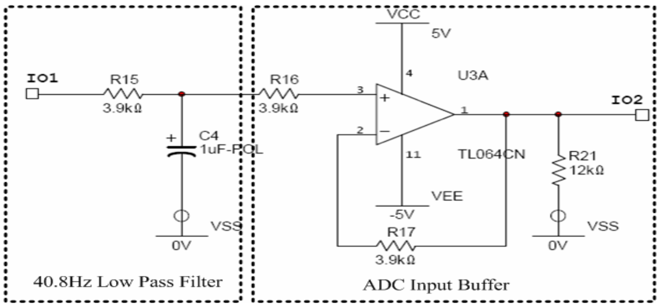


Fig. 5 LPF and ADC Buffer Circuits

The bode plot shown in Fig. 6 indicates effect of conditioning circuit on sampled SPG signal. When spectrum of SPG signal varies from 0.001Hz to above 40Hz, its amplitude decays quickly after 40Hz. The phase shift is zero degree within 0.03-1Hz and starts to increase after 1Hz, reaches -31degree at 40Hz. Eq. (3) transfers phase shift at 40Hz to delay time as 2.08ms which is acceptable in this scheme. Fig. 7 shows out the comparison between original sampled SPG signal and processed SPG signal after signal conditioning. The amplitude of processed SPG is within ADC required range (0~5V) with baseline located at about 1.5V. Moreover, its SNR is increased from 7.48dB to 12.7dB and satisfies design requirement (>10dB).

$$T_{pd} = \frac{\phi}{360}\frac{1}{f_x} \qquad (3)$$

where ϕ is phase delay in degree, T_{pd} and f_x are corresponding propagation delay time and frequency [7].

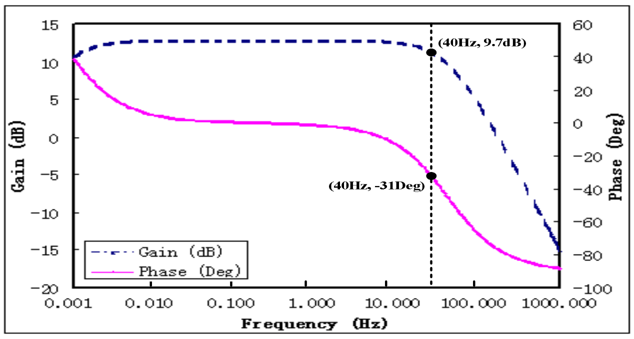


Fig. 6 Bode Plot of Signal Conditioning Circuit

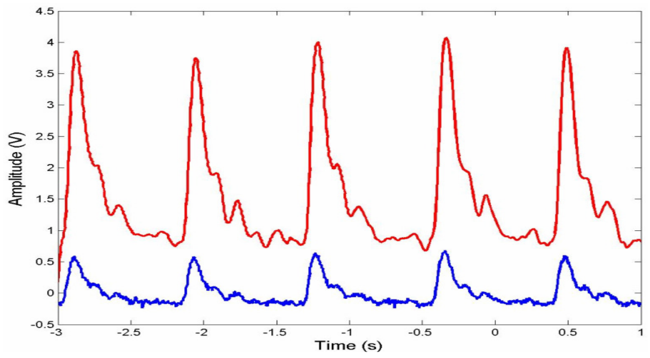


Fig. 7 Comparison between Original Sampled SPG (blue) and Processed SPG (red) Signals

4 Close-loop MCU control

Microcontroller ATMEGA88V supports C language in-system programming, thus it is programmed to control ADC, signal conditioning, timing and USB data transmission. Fig. 8 shows flowchart of MCU Control Program.

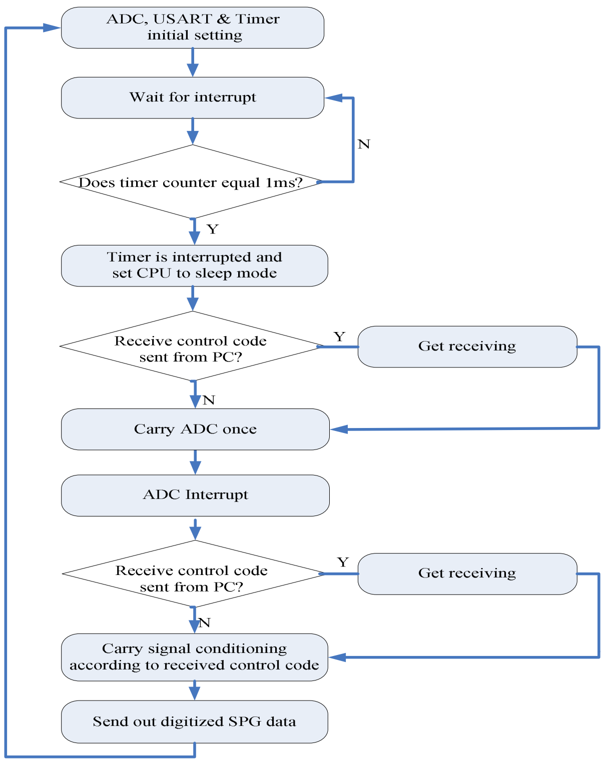

ADC, USART & Timer initial setting
Wait for interrupt
Does timer counter equal 1ms?
N
Y
Timer is interrupted and set CPU to sleep mode
Receive control code sent from PC?
Y
Get receiving
N
Carry ADC once
ADC Interrupt
Receive control code sent from PC?
Y
Get receiving
N
Carry signal conditioning according to received control code
Send out digitized SPG data

Fig. 8 Flowchart of MCU Control Program

In initial setting of MCU, the sampling frequency of ADC is set to 1000Hz and baud rate to 38400/s. Timer and the interrupt receiver are all enabled. Then MCU control program waits for interrupt signal. Once timer counter equals 1ms, timer is interrupted and set CPU to sleep mode. MCU carries ADC once with less power consumption and obtains smaller noise from I/O periphery equipment due to CPU is in sleep mode. Subsequently, ADC interrupt wakes up MCU and stores digitized SPG data. Finally, MCU carries signal conditioning according to received control code and send out finalized digitized SPG data using USART. During whole procedure, once get the receiving interrupt MCU stores control code sent from PC.

5 Amplitude and baseline-shift self-adjusting method

Ideally SPG baseline can be stably located at 0V after signal conditioning, but the home-user's improper operation in SPG measurement might shift SPG waveform to saturation or cutoff area and cause distortion. To tackle such an uncertainty and imprecision problem, an amplitude and baseline-shift self-adjusting method is proposed in this paper, which adjusts SPG baseline & amplitude, minimizes SPG distortion and lets SPG waveform totally satisfy the sampling criteria, then starts recording SPG automatically.

C++ program is constructed to realize this self-adjustment function and automatic SPG recording. Once receives SPG waveform from MCU, it compares its amplitude with sampling criterion every second. The highest point of SPG waveform is required to be larger than 4V but smaller than 5V; the lowest point must be lower than 1V but larger than 0V. When this sampling criterion is satisfied stably and continuously for 10 seconds, software system will start recording SPG waveform occurred in these 10 seconds. If the sampling criterion cannot be satisfied, software system starts to analyze amplitude and baseline of SPG waveform, and feedbacks control code to MCU. This hardware and software integrated, analysis and feedback loop between computer and MCU form a close-loop control which speeds up the sampling and guarantees the quality of sampled SPG waveform. Obviously it is a prominent brightness in such a novel scheme.

Actually, the highest and lowest points of SPG waveform are separately used to estimate amplification and baseline adjusting degree. Three ranks of amplification degree A1, A2, A3 and three ranks of baseline-shift degree B1, B2, B3 are defined. As shown in Fig. 9, software system analyzes the amplitude of input SPG waveform, classify its highest and lowest points according to above defined ranks. This analysis follows two rules: (1) if the highest point is larger than 5V, the amplification degree decreases one rank. (2) if the lowest point is lower or equals to 0V, the baseline-shift degree increases one rank. Therefore, the software system determines amplification and baseline-shift adjusting rank and feedbacks the control code to hardware MCU to adjust its digitizing SPG signal accordingly. This self-adjusting happens every second until the SPG waveform satisfies the sampling criterion. If the highest and lowest points of SPG waveform are not located at defined ranges, which is caused by the elastic band is too tight or too loose, or the measurement position is wrong, then the software system will show out a message to notice user to re-tie elastic band or adjust the measurement position on wrist. Fig. 10 shows the results of prototyping system adapted this novel SPG sampling and self-adjusting scheme, which had been successfully used in our developed e-Home Healthcare system.

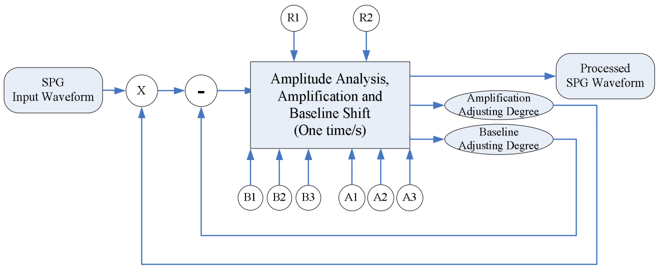

R1
R2
SPG Input Waveform
X
-
Amplitude Analysis, Amplification and Baseline Shift (One time/s)
Processed SPG Waveform
Amplification Adjusting Degree
Baseline Adjusting Degree
B1
B2
B3
A1
A2
A3

Fig. 9 SPG Amplitude and Baseline-shift Self-adjusting Method

Notation:

A1: the SPG highest point is located in 5V~4V, use amplification gain 1;

A2: the SPG highest point is located in 3.5V~4V, use amplification gain 1.2;

A3: the SPG highest point is located in 3.5V~3V, use amplification gain 1.4;

B1: the SPG lowest point is located in 2V~1.5V, shift baseline down 1.5V;

B2: the SPG lowest point is located in 1.5V~1V, shift baseline down 1V;

B3: the SPG lowest point is located in 1V ~0V, do not shift baseline.

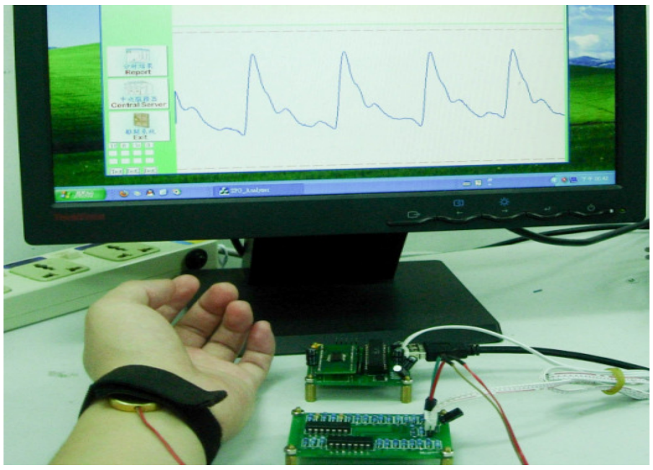


Fig. 10 Result of Prototyping System Adapted This SPG Sampling and Self-Adjusting Scheme

6 Conclusions

Demanded by pressure PWT method to estimate blood pressure, hands free SPG fast and stable sampling is a new challenge to all researchers working on e-Home Healthcare field. Towards to this mission, a SPG sampling scheme using piezoelectric transducer with signal conditioning circuit, close-loop control and relevant software for realizing signal amplitude and baseline-shift self-adjustment are elaborated and constructed. The test results show that this novel SPG sampling and self-adjusting scheme makes significant improvement in fast sampling SPG signal with tiny distortion and larger SNR.

By combining Bluetooth communication technology with this sampling scheme and designing watch-type measurement device instead of using elastic band to attach piezoelectric transducer to wrist, this scheme can offer better solution to cardiovascular monitoring and diagnosis system in e-home healthcare.

7 References

[1] Weixuan Fang, Mingchui Dong, Weikei Lei. "Novel System Sampling Multi Vital Signs for e-Home Healthcare"; 7th Int. Conf. on Information, Communications and Signal Processing, pp 1-5, Dec. 2009.

[2] Z. C. Luo, S. Zhang, Z. M. Yang. "Engineering Analysis for Pulse Wave and its Application in Clinical Practice"; The College of Life Science and Bio-engineering, pp. 19, 2006.

[3] D. C. Zheng, Y. T. Zhang. "A Ring-type Device for the Noninvasive Measurement of Aterial Blood Pressure"; Proceedings of the 25th Annual Int. Con. of IEEE EMBS, pp. 17-21, Sep. 2003.

[4] P. A. Shaltis, A. Reisner, H. H. Asada. "Wearable, Cuff-less PPG-Based Blood Pressure Monitor with Novel Height Sensor"; Proceeding of the 28th Annual Int. Con. of IEEE EMBS, pp. 908-911, Sep. 2006.

[5] J. M. Zhang, P. F. Wei, Y. Li, "A LabVIEW Based Measure System for Pulse Wave Transit Time"; Proceeding of the 5th Int. Con. on Information Technology and Application in Biomedicine, pp. 477-480, May 2008.

[6] P. Roncagliolo, L. Arredondo, A. Gonzalez. "Biomedical Signal Acquisition, Processing and Transmission Using Smartphone"; 16th Argenine Bioengineering Congress and the 5th Conference of Clinical Engineering, 2007.

[7] A. B. Williams, F. J. Taylor. "Selecting the Response Characteristic"; Electronic Filter Design Handbook, pp. 8-25, 2008.

[8] L. Y. Wei, P. Chow. "Frequency Distribution of Human Pulse Spectra"; IEEE Transactions on Biomedical Engineering, Vol.BME-32, No.3, Mar. 1985.

[9] X. Chen. "Pulse Signal Acquisition Based on LabView Software"; IEEE International Seminar on Future BioMedical Information Engineering, pp. 336-339, Jun. 2009.

[10] ATMEL. "8-bit AVR Microcontroller with 8K Bytes Programmable Flash"; ATMEL Corporation, Apr. 2007.

Wireless Sensor Nodes with IR Room Location Capability

Scott A. Henry[1], **Ramzi Ammari**[2], **and Jack Jean**[2]
[1]Dept. of Electrical Engineering, Wright State University, Dayton, Oh, USA
[2]Dept. of Computer Science and Engineering, Wright State University, Dayton, Oh, USA

Abstract— *This paper describes the design approach of two wireless sensor nodes for health care applications. Both nodes are wireless and have the capability to identify their location through an infrared signal assigned to individual rooms. This gives a baseline for comparison in the following aspects: performance of Zigbee vs. Bluetooth, overall power concern, and development boards vs. custom prototyped printed circuit boards.*

Keywords: medical device; wireless sensor nodes; Zigbee; Bluetooth, temperature, fall detection

1. Introduction

This study is part of an effort to develop two wireless sensor nodes that will provide a path towards better health care monitoring and more independent living for the elderly. In many situations healthcare requires a cyclic measurement of patient vitals, but because of either a large number of patients or small staff sizes, these measurements are only taken at longer intervals or are not monitored at all. In assisted living situations for example a nurse may visit the residents only a few times, but could provide better patient diagnostics or even emergency care in critical situations if they could provide more frequent monitoring.

To improve the diagnostic ability and emergence response the need for wireless sensor nodes for healthcare monitoring is extremely important. There are many vitals parameters and situational events that can aid in better healthcare monitoring. Temperature (a vitality parameter) and fall detection (a situational event) will be develop on separate platforms since both parameters have different specifications for coding as well as recording frequency. Being able to compare both sensors' issues and constraints gives a baseline set of data that can be applied to other vitality statistics that can be measured as technology grows and the number of sensor types increase.

Another important issue in healthcare monitoring is the location of patients. Infrared (IR) transmitters or beacons placed in strategic locations can be programmed to broadcast their location ID; with the use of an IR receivers these signals can be decoded and the position of the patient can be determined with minimum processor resources and a low impact on the cost of a resulting product.

The project was divided into two parallel elements. One approach used Zigbee protocal as the wireless communication method, temperature monitoring as its sensor component and a custom pcb design. While the second approach used Bluetooth, an accelerometer as its sensor to detect falls and a modified development board. Both platforms include an IR sensor with two approaches for decoding the room location ID from a commercial product A750R Room Locater by RF Code.

2. Zigbee Node Platform

The Zigbee sensor platform was designed to be a small device that will monitor human body temperature on a regular interval and run on a small battery. The approach for the Zigbee Sensor design will be documented here in detail, including the components chosen and why.

2.1 Xbee RF Module

The wireless transmission method was chosen based on a few specifications. The device had to be able to transmit a reasonable range so only one wireless base station would be required to receive data for a normal size house. Next, the device had to have low power consumption so that a small battery can last for a resonable amount of time. Finally, the device had to have a flexible network structure for data transmission, and a built-in encryption engine would be beneficial later in production.

A few wireless transmission appraoches met these requirements; Bluetooth, the 802.11(Wi-Fi) standard and the Zigbee protocol. Out of those three, two were chosen to proceed with, mainly based on cost and ease of use. Both of Zigbee and Bluetooth had commerical of the shelf devices

Table 1: Wireless standards.

	Zigbee	802.11(Wi-Fi)	Bluetooth
Data Rates	20, 40, and 250kbits/s	11/54 Mbits/s	1Mbits/s
Range	10-100m	50-100m	10m
Network Topology	Ad-hoc, peer to peer, star or mesh	Point to hub	Ad-hoc, and very small network
Complexity	Low	High	High
Power Consumption	Very low (sleep in uA range)	High	Medium
Join Time	Join existing network in under 30ms	Requires 3-5 seconds	Requires up to 10 seconds

Table 2: Maximum Error Temperature Range

Temperature	Max. Error
Less than 35.8 °C	±0.3 °C
35.8 °C to less than 37.0 °C	±0.2 °C
37.0 °C to 39.0 °C	±0.1 °C
Greater than 39.0 °C to 41.0 °C	±0.2 °C
Greater than 41.0 °C	±0.3 °C

that consume low power and can be easily implemented into a design. In Table 1 a brief overview of the features and specifications are given [10]. After further research, Zigbee was the protocol of chosen for one of the platforms and Bluetooth was the choice for the second platform.

Within the Zigbee protocol there are many different companies that make a Zigbee compliant product. Microchip® and Digi® were the two companies initially looked at for component selection. Microchip® was chosen first for the cheap price, but later was changed to Digi® because of the time constraint for developing the device stack on the micro-controller. The Digi® Xbee Module was a “no configuration needed out-of-the-box RF communication” [12]. But also had the option for a more complicated communication setup as the platform matured. Later revision of the platform could revert back to the Microchip’s MRF24J40MA module, because the power consumption and the physical size are slightly smaller. Another point is that the Microchip’s MRF24J40(the RF integrated circuit that handles the physical level of the protocol on the MRF24J40MA module) could be used to further reduce the size by designing the antenna and other circuity on the same PCB as the rest of the senosr. This would then require an FCC certification and testing, which is an additional step in manufacturing commercially marketable device quickly.

The Xbee Module has a few versions in its product line. In our current platform the Series 1 modules are being used, but can be interchanged with the Series 2 giving more networking options later if the need presents itself. Also within the Series 1 family there are two types, standard and PRO. The standard is used here for its low current draw. Although the PRO module does have extended range, it comes with the cost of higher current needs.

The actual function of the Xbee module is simply receiving a stream of UART data from the micro-controller and sending it to the receiving module which is connected to a PC for data logging, then the module is put to sleep to conserve power while not needed. This feature had a few options, but based on the fact that an external MCU would control the overall system, the pin sleep mode was chosen to lower the current draw to ~10μA for the long duration of inactivity. While transmitting and receiving, current draw peaked around 55mA but only for approximately of 150ms, giving great results for battery life at a very low duty cycle.

2.2 Temperature

Temperature measurement devices range from thermistors to silicon packaged IC’s. All have their pros and cons from accuracy and resolution to linear or non-linear outputs[2]. For the Zigbee Sensor Node platform, a silicon IC was used from Microchip, the TC77. This sensor meets the standards set forth by the ASTM specification for Thermometers in Table 2[11].

The TC77 sensor outputs a 13-bit resolution (0.0625 °C/Bit) two’s compliment digital word via an SPI interface. Figure 1 shows the IC surface mounted on the PCB platform. Careful attention to the specifications in the data sheet for the TC77 temperature sensor was made . The website states that the sensor has a ±1.0 °C accuracy, but a closer look at Figure 2-1 of the data sheet shows that it does meet the ±0.1 °C accuracy required by the ASTM standard[11][8].

Another reason for the choice of the TC77 is its low power usage. According to the data-sheet the current for Continuous Conversion Mode is 250μA (typ.) this has been verified by the performance test that has been done on this platform. The goal though is to have further control of the device and reach its Shutdown Mode current at 0.1μA (typ.). This goal would be achieved in future work. Further research has been done on temperature devices and the uses of thermistors seem to be a common practice. This could be because of cost and size, and will likely be implemented to further reducing the current draw as well as overall size of the sensor platform. With the current PCB layout the temperature sensor is on the bottom level of the board and creates unused space that could be used for the battery or other devices in the future. Thus, with a thermistor, it can

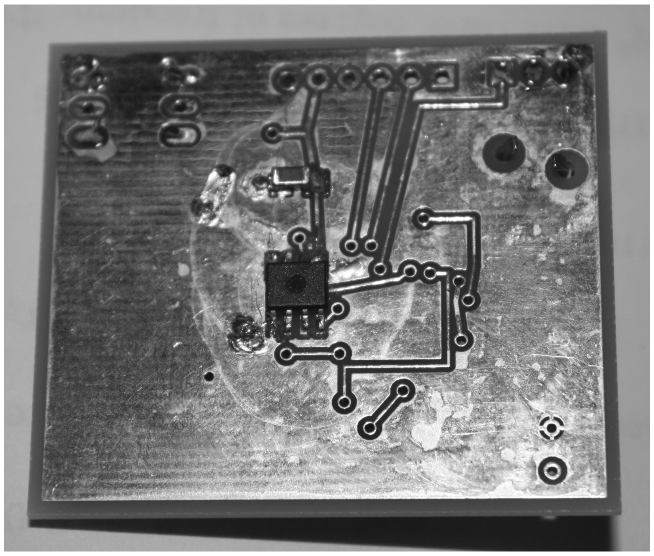


Fig. 1: TC77 Surface Mounted to PCB Platform.

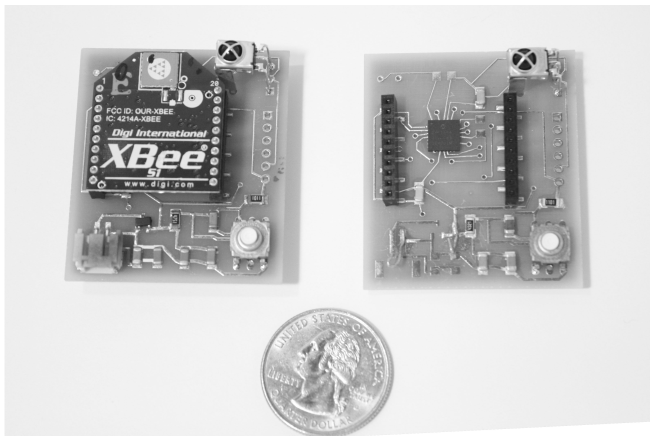


Fig. 2: First Revision of the Zigbee Sensor Platform.

be attached by two leads and the actual sensor itself can be placed on the shell of the enclosure. This would elminate the need for the bottom level to be clear of parts.

2.3 PIC24F16KA102 MCU

The Microchip® PIC24F family has a wide range of low power MCU's with many features and peripherals. The micro-controller that was chosen was based on previous experience and also its ultra-low power capabilities. Another reason the PIC24F16KA102 was used is because it has a sleep mode that can run a RTCC (Real Time Clock and Calendar) around 500nA [9]. This is ideal for a long term battery operated sensor platform as well as its wide range of voltages (1.8 to 3.6V).

2.4 Power Supply

The power supply was and is the major challenge for this platform to reach its goal of 30 days or more of run time. Thus size and capacity are the key specification for this platform.

Initially a Polymer Lithium Ion battery was considered as a possible solution, but as the development progressed the need for a rechargeable battery was proven to be less important than focus on footprint size. At this point two options were researched, a single AA battery and a CR2032 Button Cell battery.

The AA battery with a boost converter would reach the needed 3.0-3.3V of the Xbee module as well as the peak current draw of 55mA. The capacity of the AA battery (2200mAh) would give us a theoretical life of ~45 days based on current measurement of the boost converter's current draw of approximately 2mA[5]. This solution worked well for the power consumption and did reduce the footprint size of a comparable Polymer Lithium Ion battery by about one-half, but this still did not meet the intended market for the size.

The CR2032 Lithium Coin battery is the ideal size, but this battery presents a few issues. The typical capacity of a CR2032 is around 240mA, with a max pulse current of 15mA[4]. The latter is the major issue since the 15mA pulse current does not meet the peak current draw needs of the Xbee module during transmission, which is close to 55mA. At this point a secondary component had to be added to the power supply section to get the extra current needed by the Xbee module for the short 55mA pulse. This can be accomplished by using a capacitor in parallel with the battery. And thus during the low current draw sleep mode (~450μA), the capacitor will charge and then during the needed high peak current phase the capacitor will source the needed 55mA. Further research is needed to find an optimum capacitor size, but currently a 40mF supercapacitor has worked in this fashion. Another note about this configuration is that a start-up circuit will be needed for this platform to allow the capacitor to charge, or a longer delay before the first transmission is done than currently programmed.

2.5 IR Room Location

This part of the platform is what makes this device unique compared to other wireless sensor platforms. Using an infrared receiver the platform decodes the signal transmitted by the RF code's A750 Room Locater via a finite state machine(FSM) style algorithm. The FSM uses the PIC24 input capture feature to capture the timing of the incoming signal and decodes it. The signal is a bi-phase binary stream that repeats the room location 3 to 4 times back to back. This bit stream is transmitted every 3 second from the A750 Room Locater. After the room location is converted from the binary bi-phased stream, it is then framed in an octal format and sent to the UART output of the PIC24 in ASCII format.

2.6 Further Development

At this point there are two areas of improvement needed to reach a commercially marketable device. The first area is reaching the goal of over 30 days of run time on the button cell battery or similar size. This can be done by taking advantage of the other device's sleep modes and being able to control the sensor power whether it is on or off. Second, reducing the physical footprint of the platform. This will require a change in the temperature sensor location or type of sensor used. One thought is to use a thermistor and the PIC24's ADC to measure the temperature, thus giving us the entire bottom of the PCB to put components on. This will take some minor adjusting with the time thresholds and a calibration method for getting an accutate temperature.

3. Bluetooth Node Platform

For elderly people, unexpected falls can cause significant injuries. Fall related injuries are the leading cause of injury-related deaths among adults 65 years old and older; these injuries are also the number one reason for emergency room visits [3]. For that reason previous research has approached this problem to come up with a mechanism to detect falls.

Common fall detection methods include a detection feature and a trigger to issue an emergency call to provide medical care. Fall detection is built on algorithms that use sensor values such as accelerometers, gyroscopes, and air pressure sensors. These sensors cannot fully differentiate between falls and ADL (Activity of Daily living) which leads to a high percentage of false alarms; the main reason for the failure of these systems and the fact that it is mainly rejected for commercial use by monitoring services is the high percentage of false alarms [6]. In our design we added the room location capability to identify the location of the fall, send this information to software that triggers a wireless camera to stream live video feed from that room to identify the fall and send medical care to the patient first. This is an optimal solution for an assisted living situation where you have a large number of elderly people as well as large number of rooms.

An independent living for the elderly is also an important aspect of this paper. The high cost of these systems for personal use made it difficult to be commercially successful. The choice of Bluetooth vs. Zigbee has its pros and cons, for the fall detection device we chose to use Bluetooth so that we cut the cost of having to use a base unit to initiate the emergency call, and the need to have a dispatcher to send the help needed.

The idea is that most people use cell phones that have Bluetooth capability. By the close of 2005; wireless subscriptions hit nearly 2 billion on a worldwide basis, with cellular mobile dominating the wireless technology field, according to a trends study from Deloitte Research. By establishing a connection between the Bluetooth modules in the device and the cell phone once a fall is detected, we can initiate an emergency call directly from the phone to 911 or to family members.

3.1 Hardware Architecture

The initial goal for this device was to design a prototype with sensor interfacing and wireless capability. So we used the Ardunio Uno board which is the latest revision of the basic Arduino USB board. Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. Future work will include the design of printed circuit board with only needed components on the board which will ultimately lower power consumption and make the size much smaller.

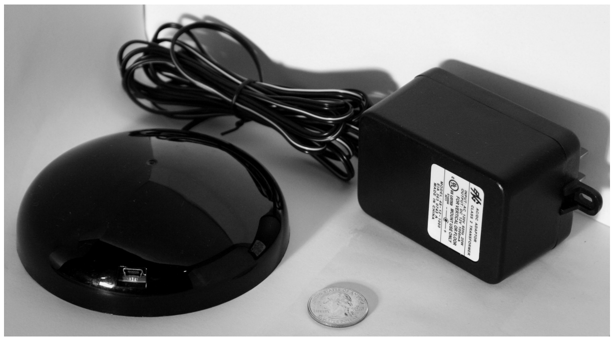


Fig. 3: RF Code A750 Room Locator.

The Arduino Uno is a micro-controller board based on the Atmega328. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the micro-controller [1].

The accelerometer used is the ADXL335 which is a small, thin, low power complete 3-axis acceleration measurement system. The ADXL335 has a measurement range of $\pm 3g$ minimum. It contains a poly-silicon surface-micro machined sensor and signal conditioning circuitry to implement open-loop acceleration measurement architecture. The output signals are analog voltages that are proportional to acceleration. Figure 5 shows the numeric values for the accelerometer, the x,y,z graphs of the accelerometer as well as the room location.

The Bluetooth module used is the Bluesmirf gold Bluetooth which is an FCC Approved Class 1 Bluetooth Radio Modem delivers up to 3 Mbps data rate for distances up to 100m. This module has Low power consumption (30mA connected, and less than 10mA sniff mode).

3.2 36kHz Infrared (IR) Receiver Module

Infra-Red is a normal light with a particular color. Humans cannot see this color because its wavelength of 950nm is below the visible spectrum. An IR remote works by turning the LED on and off in a particular pattern. However, to prevent interference from IR sources such as sunlight or lights, the LED is not turned on steadily, but is turned on and off at a modulation frequency (typically 36, 38, or 40KHz). Modulation is the answer to make our signal stand out above the noise. With modulation we make the IR light source blink in a particular frequency. The IR receiver will be tuned to that frequency, so it can ignore everything else. You can think of this blinking as attracting the receiver's attention. Humans also notice the blinking of yellow lights at construction sites instantly, even in bright daylight.

In serial communication we usually speak of 'marks' and 'spaces', which are decoded into 1-s and 0-s depending on the protocol that's being used. If you know the encoding algorithm, you can determine the code value. For example, televisions decode the infrared signal received from the remote control into sequence of 1-s and 0-s; for each button pressed you get a unique code that can be used to trigger a specific function such as turning the volume up or down.

The A750R Room Locator decoded signal is more complex than the ordinary infrared signal. The signal transmitted

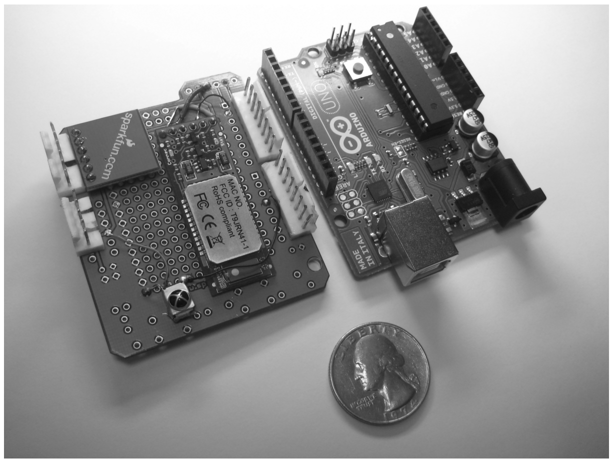


Fig. 4: Arduino UNO with Bluetooth and Accelerometer

carries information about the tag ID, room location, and privileges for whom can access certain rooms. This complex signal is hard to decode with any known protocol. So our approach is to uniquely distinguish each signal transmitted. If we turn each unique sequence of pulses into a unique value, then this value will indicate the room location.

A simple routine developed by Ken Shirriff was modified to decode the infrared signal [7]. The code looks at the sequence of the space signals and compares them according to their length. The code then assign a 0,1, or 2 if the length is equal, longer, or shorter. Same way with the mark signals. The resulting 0-s, 1-s and 2-s are hashed into a 32-bit value. This gives a unique code for each room. The device searches a look up table to match the code to the room number.

4. Comparison and Discussion

The design of these two platforms gives a good starting point for future research and a good starting point for further reduction in size and power consumption. Both platforms have a viable use in further research that is needed in healthcare monitoring for both assisted living and clinical situations. In comparing the two devices on a performance basis we found it somewhat difficult, mainly because one was an out of the box development board and the other was a design from the ground up. The ground up Zigbee platform did not have to worry about extra current running LEDs or other devices. On the other hand the development using Bluetooth Arduino board was a very simple to use platform, that was flexible for changes later in the development. So from that aspect of comparison of platform to platform, both devices had their advantages and disadvantages.

The wireless communication is another point of discussion. Here the Zigbee outperformed the Bluetooth device for its intended application, in both ease of use as well as power consumption. The ease of use came from the fact that the Xbee module could take a UART signal and pass the signal onto the base station without any configuration. Also it was flexible enough to be able to give an assigned ID to the node through a simple GUI interface on the PC base station. And finally the sleep modes were far easier to access: in the Zigbee case a sleep/wake pin could be chosen for putting the module to sleep. This in turn dropped the current draw to ~50μA or less, saving a very large draw of current. The Bluetooth had similar power saving features that use 30mA when connected, and less than 10mA in sniff mode. Future research and development should be able to improve the power consumption of both platforms as well as any other problems that might arise.

As far as the IR Room Location decoding approaches, we proposed two methods. The first approach is a capture and compare to known patterns (basically a look up table of sorts) and the second is a state machine algorithm, two trade-offs were apparent. The weakness of the first being the time it takes to learn the patterns for each Room ID and then the physical memory it takes to store them. This approach worked very well, but limits the Room ID's to the available space on the MCUa and the time it takes to gather and input the patterns acquired. Since the protocol is proprietary and is currently inaccessible to our research group, the look-up table works and still fits into the space currently available on the MCU. But the second approach uses far less resources. By deciphering the protocol with a software serial decoder algorithm, a better result can be achieved since any number of Room ID's can be read without having to learn or pre-program the MCU with the appropriate code. The down side is the timing requirement if the FSM misses the transmitted or reads an incorrect input capture then is must start over and look for a start bit again, until it can decode the data correctly. Currently a more robust algorithm is being adjusted to decode the signal faster and still maintain accuracy.

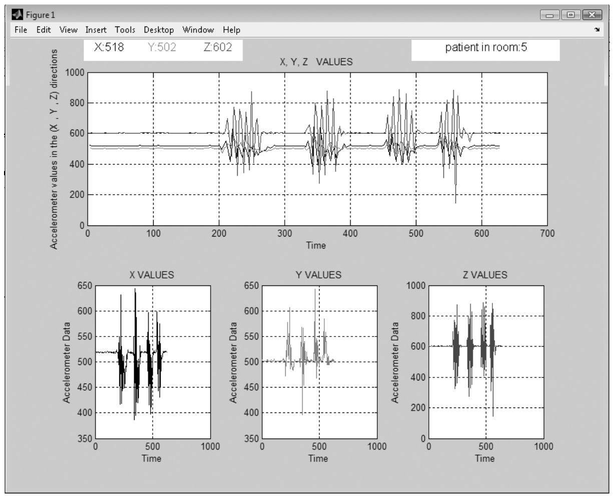


Fig. 5: Accelerometer Data Resukts

5. Conclusion

The healthcare industry is in need of better ways to improve the care of both clinical and assisted living patients. These platforms can be good starting point for a commercially developed product that will solve vital monitoring needs as well as provide data for medical researchers to find new treatments and solutions to aid and monitor the growing population.

Developing these two platforms in parallel was very useful to compare different technologies and learn the limitations and advantages of these platforms. Further research is still needed and will be continued in this area to improve these platforms to meet the standards of consumer quality and use.

References

[1] "Arduino–ArduinoBoardUno." [Online]. Available: http://arduino.cc/en/Main/ArduinoBoardUno.

[2] Bonnie Baker. (1999) "AN679 Temperature Sensing Technologies." Microchip Technology Inc. [Online]. Available: http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1824&appnote=en011702.

[3] Department of Health and Human Services. "Fatalities and Injuries from Falls Among Older Adults–United States, 1993-2003 and 2001-2005." [Online]. Available: http://www.cdc.gov/mmwr/preview/mmwrhtml/mm5545a1.htm. November 2006.

[4] Energizer, "Energizer CR2032." CR2032 data sheet. 2010.

[5] Energizer, "Energizer E91." AA E91 data sheet. 2010.

[6] G. Pérolle, P Fraisse, M Mavros, and I Etxeberria. "Automatic fall detection and activity monitoring for elderly." Sponsored by the CRAFT project.

[7] "IR remote control theory." [Online]. Available: http://www.sbprojects.com/knowledge/ir/ir.htm. [September 23, 2009].

[8] Microchip Technology. "TC77." TC77 data sheet. [Jan. 2011].

[9] Microchip Technology. "PIC24F16KA102." PIC24F data sheet. [Feb. 2011].

[10] Software Technologies Group, Inc. "Zigbee vs versus other networking standards." [Online]. Available: http://www.stg.com/wireless/ZigBee_comp.html. 2009.

[11] Standard Specification for Electronic Thermometer for Intermittent Determination of Patient Temperature, ASTM International Standard E1112-00, 2006.

[12] "XBee-PRO®802.15.4 OEM RF Modules –Digi International." [Online]. Available: http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/point-multipoint-rfmodules/xbee-series1-module.jsp#overview. [Feb. 2011].

Developing a Remote Digital Wildlife Camera Triggered by Spatially Deployed Infrared Sensors

William Collins[1], Daniel Sanchez[1], Zachary Sharp[2], Scott C. Smith (contact author)[3], and Jingxian Wu[3]
Department of Engineering Science, Sonoma State University, Rohnert Park, CA[1]
Department of Electrical Engineering, Arkansas Tech University, Russellville, AR[2]
Department of Electrical Engineering, University of Arkansas, Fayetteville, AR[3]
collinswf@gmail.com, danjsanch@gmail.com, zacharylsharp@gmail.com, smithsco@uark.edu, and wuj@uark.edu

Abstract—**This paper describes the design, construction, and operation of a system devised to capture images of wildlife with a full 360 degrees of view. The system uses infrared detection in a circular configuration to detect movement of heat signatures and then relays that information to a base station via short distance radios. The base station uses a microcontroller to take this information and turn a camera using a stepper motor. When the camera is in the correct position it is triggered to take a picture. Images are stored on an internal SD Card, and are then transferred to a remote location via long distance radios where a user can view captured images without manually retrieving them.**

Keywords- Wireless Sensor Network, Stepper Motor, 8051 Microprocessor, ePIR Sensor, XBee Radio

I. INTRODUCTION

The main motivation driving the work done throughout the Electrical Engineering Summer Research Experiences for Undergraduates (REU) was to improve the current technology available to wildlife researchers and hunters through the enhancement of wireless embedded system instrumentation. This was achieved by the design and fabrication of a system capable of using infrared sensors arranged along a perimeter that wirelessly communicate with a base station that uses a stepper motor to turn a camera towards a detected moving heat signature and then take a picture.

Section II describes previous work done in the field of automatic wildlife imagery; Section III provides an overview of the system and details all of its components; Section IV describes how the system operates; and Section V discusses our results and conclusion with future work.

II. PREVIOUS WORK

There are several companies that manufacture and market cameras designed for wildlife research and scouting. One major problem with these digital wildlife cameras is the limited range of their sensing mechanism and corresponding viewing area. Most game cameras use a single differential passive infrared (PIR) sensor configuration mounted in a fixed direction to detect the heat signatures of passing game, which triggers the camera to take its picture. The problem with this sensing mechanism is that it is singular in its point of focus and covers a narrow area defined by the camera’s field of view and the acceptable sensitivity range of the embedded sensor. A typical configuration would capture activity on a known trail within a wedge shaped area of approximately 440 square feet. This is highly inadequate for monitoring most fields, which are normally much larger than 30 feet across, such that much of the wildlife passing through may not trigger the sensors, and therefore the camera will not take their picture.

III. OVERVIEW OF SYSTEM

This project developed a remote access digital wildlife camera. To achieve the objective, multiple wireless nodes equipped with PIR sensors were deployed in a circular configuration, thus forming a monitored perimeter with a digital camera at the center. The game camera was mounted on a microprocessor controlled stepper motor. When one of the sensors detects the heat signature of passing wildlife, the digital camera is directed to turn toward the sensor and record a still image. This configuration expands the monitored area by creating multiple image zones, which are determined by the maximum sensitivity (range) of the sensors, image collection strategy (i.e., the camera lens field of view), and number of images taken in each zone. In order to simplify the required camera control interface a fixed focal length lens was utilized. Since multiple sensor nodes were deployed, the radius and therefore the total coverage area of the monitored zones was constrained only by the resolution of the camera and the effective distance of the flash for low light image acquisition.

A. Base Station

The base station is comprised of several elements including: a camera, stepper motor, 8051 microprocessor, 2.4GHz and 900MHz XBee Radios, and a power supply sourced by a 12 volt battery.

i. Camera

While reviewing cameras, several different makes and models were considered, from commercial point and shoot, digital SLR, and specialized industrial cameras. The final decision was to buy and modify a refurbished commercial trail camera. The Moultrie Game Spy I65 Digital Trail Camera was chosen based on its features and price. Besides having four picture resolution settings and a built-in infrared lens and flash, this camera came with many additional features that are well suited to its intended purpose. The wide viewing angle of the camera helps ensure that the system captures pictures of animals under a broad set of circumstances including multiple animals passing simultaneously, animals moving laterally across the field of view and directly toward the camera, and animals that have moved past the trigger point while the camera is positioned and prepared for image capture. The lens of the camera is fixed but provides an adequate focus distance. A major advantage of this camera is that it automatically switches to an infrared mode in night time shooting conditions and includes an integrated IR LED flash that was tested as effective up to 50 feet in night time conditions. The camera is also easily accessible, meaning that it is very simple to open and manipulate, and is housed in a waterproof case, as shown in Fig. 1, which makes it ideal for its intended use. The camera is capable of operating from a 12 volt input that allows us to use a standard deep cycle 12 volt lead acid battery as its power source, which can also be used to drive the stepper motor with adequate torque. A final feature is the camera's information bar, which imprints each picture with time, date, moon phase, temperature, and barometric pressure. All of this information could be viable in studying the habits of wild game.

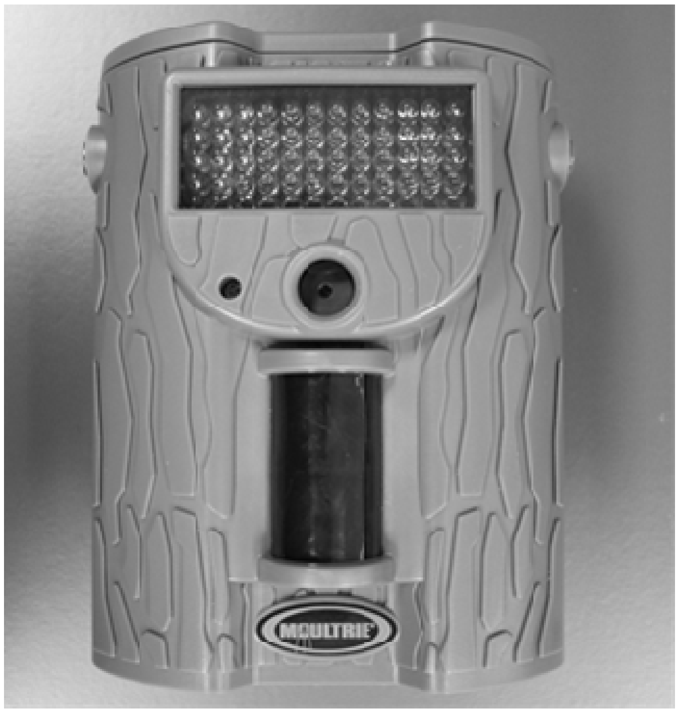


Figure 1: Camera in Weatherproof Enclosure

ii. Stepper Motor

One of the main components of the base station is the stepping motor. By utilizing the stepping motor's full range of rotation, the viewing field of the camera was increased from 52 degrees to 360 degrees. The stepping motor is an Airpax brand, 6 volt, 5 ohm, 1.8 degrees per step device. While it is rated for 6 volts, it was driven by the 12 volts coming from the power supply. The stepping motor is a 6 wire motor and is therefore capable of bipolar operation, but for ease of use the wire leads coming from the nodes connecting each of the two half stators together have been left floating so that the motor operates as a unipolar stepping motor. Unipolar stepping motors are easier to use considering that their pulse timing is not as critical since their configuration does not lend to shorting current over multiple transistors. The two windings within the stepping motor are driven by their own individual UC3770B Stepper Motor Drive Circuits. These circuits control and drive the current of their respective windings.

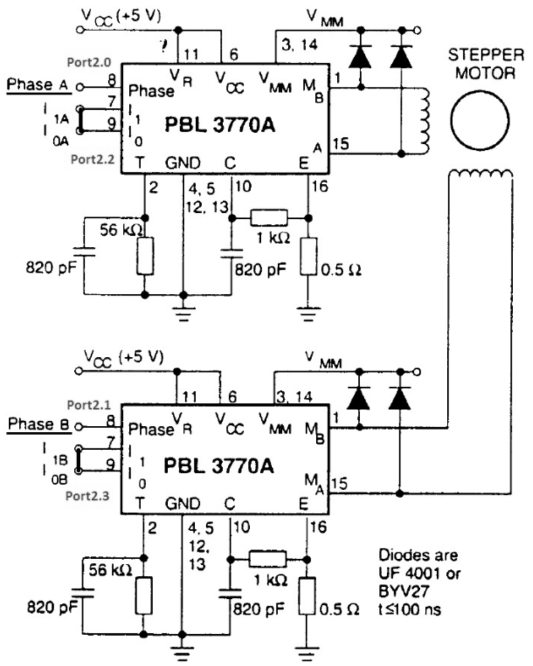


Figure 2: Stepper Motor Control Circuit

The Phase, I_0, and I_1 pins control the direction and speed of the stepper motor as shown in Fig. 2. Each whole motor step increment requires an 8 step sequence diagramed in Fig. 3.

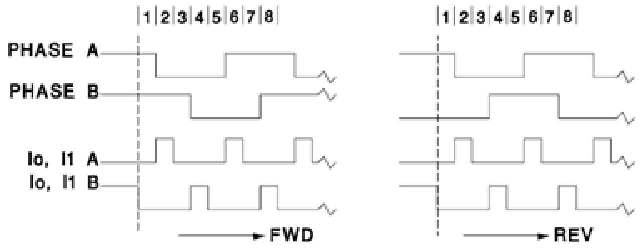


Figure 3: Stepper Control Logic

The stepping motor described above is part of a gear system that was salvaged from a decommissioned custom lab instrument. The gears were rearranged from a 10:1 ratio to produce a 2.5:1 ratio. That is 2 ½ motor rotations will make one full rotation of the turntable. The turntable is where the camera is mounted, so this gear ratio allows for more accurate positioning of the camera. On the shaft just under the turntable is a collar with one small slit. Around this collar a Photologic OPB972 slotted optical switch is positioned in order to help calibrate the camera. The switch outputs a logic high unless the turntable is in the location such that the slit is positioned where the infrared passes through the slit and is received by the collector. Using this switch configuration allows the user to know exactly where the camera is positioned.

iii. Base Station Circuitry

The base station serves as the hub of the system both physically and logically. Its primary purpose is to: coordinate, prioritize, and respond to the signals from the sensor nodes and the remote management node; control the digital camera and manage the image acquisition process by controlling the digital camera shutter release and stepper motor positioning circuits; and provide a method for a remote file retrieval service for the remote management node.

The base station was designed as an embedded system with a Dallas Semiconductor DS89C450 [1], which is an Intel 8051 microcontroller variant that provides up to 64K of program space, one instruction per clock cycle at 11.0592MHz, and 4 eight pin I/O ports including two UARTs for serial I/O and on chip programming. The communication sub-system consists of a short distance (up to 300’ line of sight) 2.4 GHz, 1mW, IEEE 802.15.4 radio [2] for interacting with the sensor nodes and a long distance (up to 1.8 mi. line of sight) 900 MHz, 50mW radio [3] for communication with the remote management node.

The camera subsystem provides a weatherproof enclosure for the camera, integrated IR flash, base station circuit board, both radios, and all interfaces to the stepper motor and external power supplies.

As shown in Fig. 4, the layout of the main base station circuit board was determined by the relative location of interfaces to both on and off-board components and subsystems. In order to minimize the demand of limited on-board real estate the 3.3V, 5V, and 12V power supplies are located in an external enclosure. In addition, the DB9 serial programming port is to be brought to an external location while the RS232 to TTL level shifter will remain on the main circuit board. If desired the 8051 could be programmed by physically removing the chip and placing it in an external programming platform which could reduce wiring congestion and free up space by eliminating the in-circuit programming capability. This would reduce the space requirement by approximately 1.5 square inches by removing two LM3904 transistors and several bias resistors along with the MAX202 level shifter and charge pump capacitors. The camera enclosure provides enough space for a second smaller circuit board that will host a Vinculum USB host controller chipset [4] to be used as an interface to the camera SD card storage subsystem, as shown in Fig. 5.

iv. XBee Radios

The communication sub-system will consist of a short distance (up to 300’ line of sight) 2.4 GHz, 1mW, IEEE 802.15.4 radio [2] for interacting with the sensor nodes and a long distance (up to 1.8 mi. line of sight) 900 MHz, 50mW radio [3] for communication with the remote management node. These radios are mounted to the back covering of the camera with leads running to the microprocessor board inside of the camera enclosure.

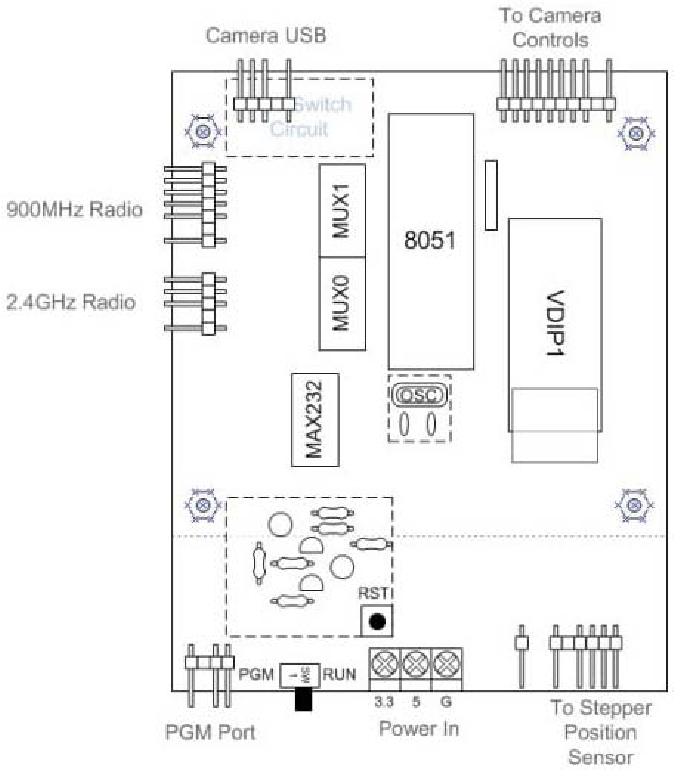


Figure 4: Base Station Block Diagram – Bottom Tier

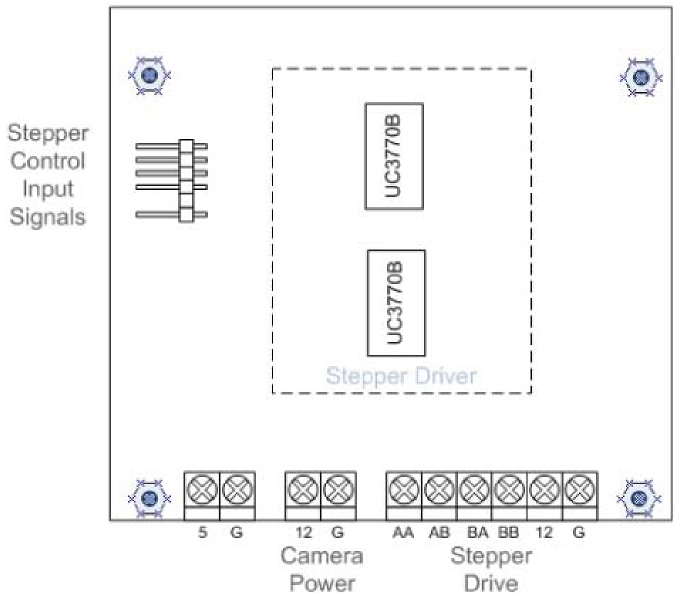


Figure 5: Base Station Block Diagram - Top Tier

v. Base Station Power Supply

To extend the service life of the entire system, a 12 volt battery was implemented into the design, as shown in Fig. 6. The power source consists of Step Down (Buck) converters that use pulse width modulation to bring the 12 volt source down to 3.3 volts and 5 volts, both rated for 3 amps. The power supply has in line fuses and diodes to prevent damage to any circuitry if polarity on the battery is accidentally reversed or if too much current is being drawn. Lastly, the power supply has a simple single pull single throw switch for easily turning the power to the base station on and off. LM2576 switching regulators rated for 5 volts and 3.3 volts were implemented. Also on the power supply box is a single pull single throw switch tied to a pull up resistor through a 5 volt source that acts as our calibration switch.

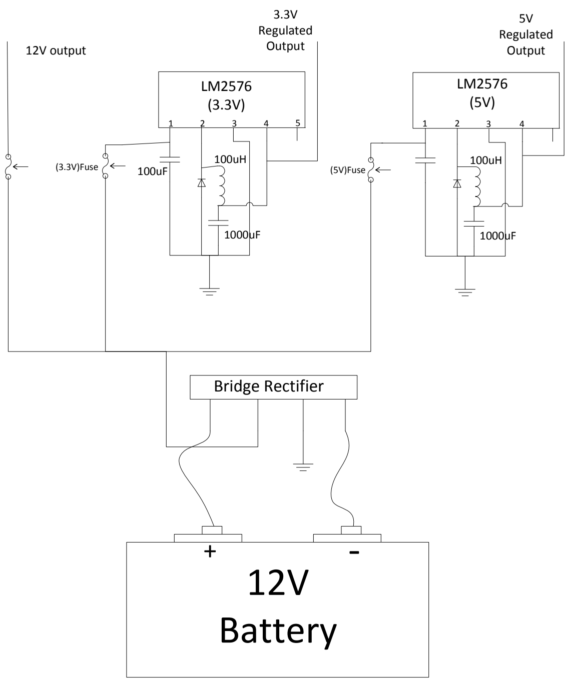


Figure 6: Base Station Power Supply Schematic

vi. Base Station Construction

Since our final product was to be field operational, many precautions were taken to ensure that the system could endure a variety of weather conditions typical of our location. All circuitry chosen was rated to withstand temperatures well within the norm of North West Arkansas. However, our main concern was making the system water-proof. The power supply was built inside of a project box, and then placed with the battery inside of a battery box. The stepping motor was mounted inside of a section of 6 inch PVC tubing with only the mounting bracket for the camera exposed at the top. The camera itself is waterproof, and the whole system sets on top of a surveyor's tripod, which is capable of being adjusted for leveling and height purposes.

B. Remote Sensing Nodes

The main components of the remote sensing nodes include: ePIR sensors, power supply, and the 2.4GHz XBee radio.

i. ePIR Sensors

The product chosen for the motion detection was Zilog's ePIR Motion Detection Zdots Single Board Computer (SBC). This product is a complete motion detection system that has a PIR sensor (passive infrared) and a Fresnel lens and comes pre-programmed with motion detection software. Its sensitivity parameters are controlled via simple hardware configuration. The important factor for choosing a sensor was the detection pattern. The area covered by the ePIR, or range, is maximized by using it in Extended Range Mode. This actual distance depends on the sensitivity setting but it ranges from 3 meters by 3 meters to 5 meters by 5 meters with a 60 degree angle. Preliminary tests indicated that the range of detection was indeed up to 5 meters but held most consistent up to 4 meters. It has a digital output so that when motion is detected the output is 0V, and 5V when no motion is detected. The detection is most reactive to motion perpendicular to its pattern which is what determined the optimum configuration of the sensors around the base station.

With a few options for configuring the sensors, the objective was to minimize the number of sensors required to ensure full coverage (minimize cost) as well as maximize the efficiency of the range. The first sensor formation that was considered was an array of sensors facing one direction where each node would be placed at the exact range distance with no overlap. The lack of overlap was to minimize the number of sensor nodes required. It was decided that if the detection range for some reason wasn't as consistent as its specs specify there would be a possibility of misdetection, which is why the second formation was considered. This formation was just a slight variance of the first being that there would be overlapping. The tradeoff for this formation was the higher accuracy over the expense of adding more sensor nodes to our system. The final configuration decided was to place two sensors on each sensor node at an angle of X to meet the requirement of covering a radius of 40ft from the base station. With this formation, the number of radios necessary was cut in half and the overlapping range of coverage by each ePIR was addressed. As shown in Fig. 7, each sensor node covers a 40 degree zone from the camera in the center, requiring 9 sensor nodes to cover all 360 degrees.

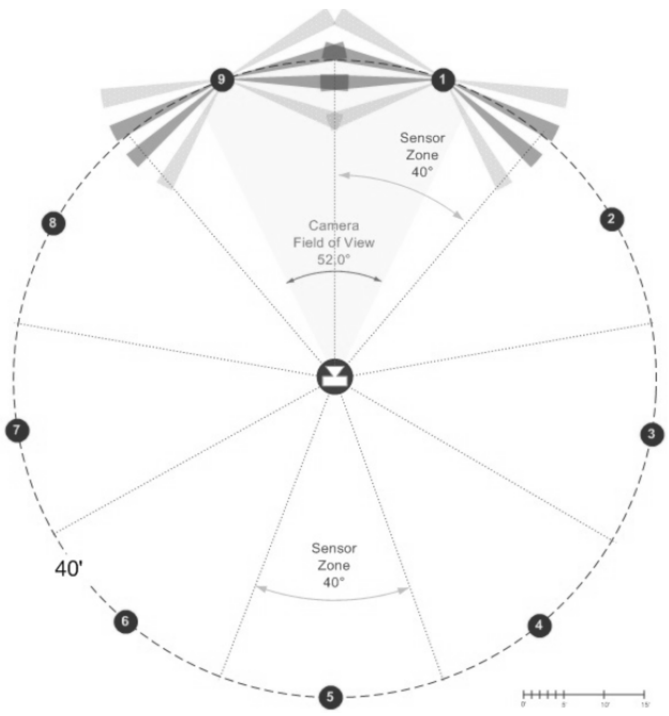


Figure 7: Sensor Node Coverage Pattern

ii. Node Power Supply

Size and longevity were both considered when choosing a power source for the nodes. After weighing the advantages and disadvantages of several different sources, it was decided to use packs of two C-Cell batteries per node. These C batteries were believed to give the best power output for their size and were easily adapted into node construction. As shown in Fig. 8, Max756 chips were used to step the voltage output of the batteries up to a regulated 3.3V. Stability was a main factor when choosing the Max756 for the radio's source voltage, since when this varies within a few hundred mV, the packets become distorted and inconsistent. This converter accepts all the way down to .7V input and converts it to a stable 3.3V or 5V depending on its configuration, with 87% efficiency at 200mA.

iii. Node Construction

The nodes were placed in a PVC enclosure comprised of a 4 inch DWV Female Adapter, a 4 inch cleanout plug, and a 4 inch cap. On the bottom side of the cleanout plug a 1 inch plug was attached and tapped for a ¼ 20 bolt. This allowed us to fasten the nodes to 3 foot iron stakes. The PVC enclosure was drilled for the two ePIR sensors to point out of. We then used clear silicon to secure the ePIR sensors into place and to prevent water from entering the node. The power supply and 2.4GHz XBee radio were also put into the enclosure along with an on off switch and a calibration button.

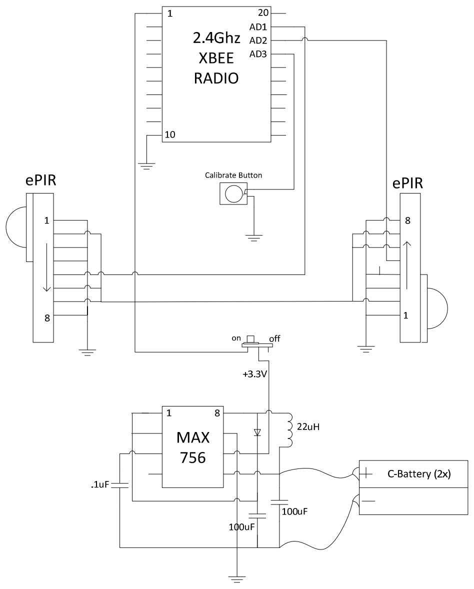


Figure 8: Sensing Node Circuitry

C. Remote Management Node

A remote management node was constructed consisting of an XBee Pro 900MHz, 50mW radio and an FTDI USB to UART interface. This unit can be tethered to a PC. Drivers are readily available for most common operating systems and provide a virtual com port that can be addressed with a simple terminal program to facilitate the bi-directional communication with the Base node. The radio is configured for operation at 115,400 baud, 8 data bits, no parity and 1 stop bit. The remote node hardware is powered from the USB port and has an external RP-SMA connector for a 900MHz quarter wave antenna.

IV. Software Design

The functional logic of the base station is implemented as a simple finite state machine. There are three primary operational states, shown in Fig. 9: Idle, Sense, and Manage.

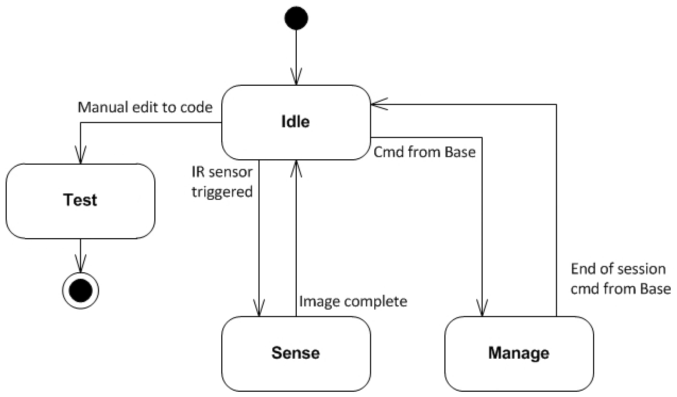


Figure 9: Finite State Machine

In addition a fourth test state can be utilized for future system development and verification by manually altering the code but this state is not active in normal operation.

Once powered up, the base station sits in the Idle state waiting for a byte stream from one of the two base radios. The arrival of a byte from one of these radios will invoke an interrupt service routine associated with one of the two 8051 UARTs. The ISR will set the software defined state variable to the appropriate state value, Sense or Manage, and perform minimal processing before surrendering control to the finite state machine code segment.

A received packet from one of the sensor nodes via the 2.4GHz radio on serial port 1 will cause the finite state machine to transition to the Sense state. It will remain in this state and perform stepper calibration, camera position, and image acquisition sub-states or actions until a complete image cycle has occurred as shown in Fig. 10. Upon completing an image acquisition cycle, it will transition back to the Idle state.

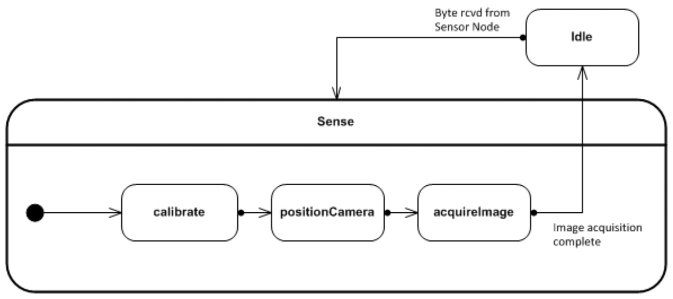


Figure 10: Sense State

When a command byte is received from the remote management node via the 900 MHz radio on serial port 0 the system will transition to the Manage state. The primary

Manage state envelops sub-states and actions necessary to respond to a remote management node as shown in Fig. 11.

Each sub-state corresponds to or supports a command request from the remote management node and additionally provides the initialization routines required to communicate with the camera's SD card storage system via the Vinculum VDIP USB host controller. Once the initialization routines have established a communication link with the camera file system, individual files may be retrieved by the remote node one at a time by sending a four byte command from the remote management node to the base station.

A simple command protocol was created as shown in Fig. 12. Each command consists of a four byte sequence that begins with a command flag byte (0xE7), followed by a command ID byte (0xC0-0xCF or 0xA0-0xAF), a parameter byte for command arguments, and a command termination flag (0xE8). For example, the command to request file number 10 from the camera would be formulated as the four bytes: **`0xE7 0xC2 0x0A 0xE8`**. Each byte is sent as a hex value with no carriage return or line feed within or at the end of the string.

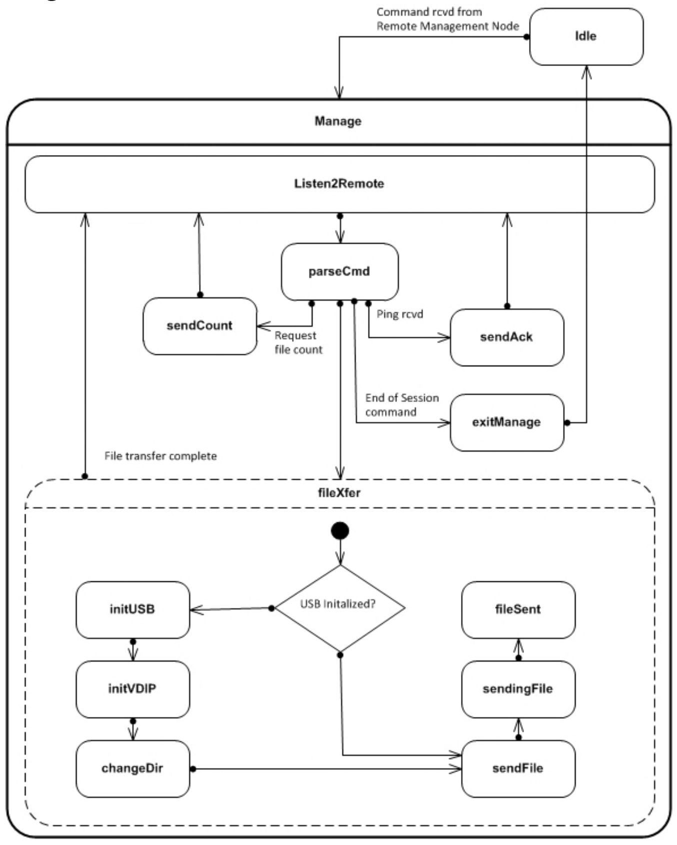


Figure 11: Manage State

The 16 available command IDs from 0xC0 to 0xCF are reserved for commands from the remote management node to the base. Command IDs from 0xA0 to 0xAf are responses sent from the base to the remote management node.

The camera images are recorded in medium resolution JPEG EXIF format. The base station processes the file request command by converting the file number parameter into a properly formatted file name for the camera file system such as PICT0nnn.JPG where nnn is replaced by the decimal converted hex value sent as a parameter. All bytes are sent as a continuous stream until the base station detects the end of the file. The base station will send an A9 command string to the remote node to signal that the file transfer is complete.

The Manage state must be terminated by the remote node via an End of Session command C9. Upon receipt of this command the base station will disconnect from the camera file system by removing power from the USB interface and return to the base station Idle state.

Command ID	Description	Parameter
C0	Ping remote	none
C1	Get file count	none
C2	Request file	Hex byte
C9	End Of Session	none
A0	Remote ping response	0x01
A1	File count	Hex byte
A9	End of file transfer	none

Figure 12: Command Protocol

V. SYSTEM OPERATION

To operate the system, the first thing required is to set up the base station and sensor nodes. The base station that sits on top of the surveyor’s tripod should be leveled and approximately 4 feet off the ground. Once this is in place the system should be turned on and switched into calibrate mode. While in this mode pressing the calibrate button on any of the sensing nodes causes the camera to turn and face the direction where the node should be placed. To help with alignment the camera itself is switched into aim mode which lights an LED on the face of the camera. Users can utilize this light along with a measuring device of 40 feet to locate the exact position of every sensor node. The node is also marked so that the enclosure should face the camera correctly and align the ePIR sensors properly.

It is important to note that before the camera turns towards the correct location, it returns to its home position, which is known by the use of the Photologic OPB972 slotted optical switch. This is to prevent miss positioning the nodes due to the camera being manually turned or a gear slipping. Another reason for implementing this switch is to keep the camera from turning more than 360 degrees. If the motor is turned more than 360 degrees without the OPB972 switch being triggered, the system is shut down. This is to prevent the camera from constantly turning in one direction which would twist and break the wires running through the shaft of the motor hardware.

Once the system is set up, the calibrate switch is flipped to sense. This turns the camera on so that it is ready to snap a picture, and also has the microprocessor looking for radio packets that tell it where to point the camera. By default the camera is at its home position until it receives an order to turn a certain direction.

There are two identifiers when decoding a packet received from a tripped motion sensor: one is the radio nodes (1-9) and the other is which side of the node it is, left or right. The decoding process of these 14 byte packets includes entering an interrupt service routine where it processes each individual byte. For these packets byte 5 defines which radio is being triggered and byte 12 defines which side is being triggered. The radio node and side are called in a function of code that moves the stepper motor to point in the appropriate direction.

Once the camera is turned towards the heat signature that has tripped the ePIR sensor, it is then pulsed to take a picture. The camera has a sleep function that takes effect after 100 seconds of inactivity. If in sleep mode, the camera takes approximately 12 seconds to wake up. The code is set to recognize if the camera is in sleep mode and enter a subroutine to wake it up and wait an appropriate amount of time before capturing an image.

Another feature of the system is its nighttime operation. The ePIR sensors function well in lowlight conditions and the camera has a light sensor which switches between daytime and nighttime pictures. The camera has an infrared flash and lens which are automatically deployed when low light levels are detected.

Files can be retrieved remotely using the remote management radio node. A terminal communication program such as Hyperterm or RealTerm is required to send commands to the base station and receive the image files from the base station. Each command from the remote node consists of four bytes as described in Section IV. When a command is sent to the base station from the remote management node, the base station will respond with the appropriate data. If a file is being retrieved, the remote management node terminal software must be ready to capture all of the bytes and save them to a file as there is effectively no delay between receipt of the command and the initiation of the data stream. The base station will send the four byte sequence of 0xE7 0xA9 0x00 0xE8 to indicate that the file transmission is complete. The remote management session must be terminated by sending the End of Session command to the base station.

VI. Results, Conclusions, and Future Work

The final design operated as intended. The calibration mode made setting up the sensing nodes simple. Switching into sense mode allowed the camera and motor to operate simultaneously with correct timing to obtain high quality still images when a packet was transferred and received by the 2.4GHz XBee radios.

There was some difficulty with receiving false triggers from the ePIR sensors. The exact source of the problem is unknown but some probable causes could be the high ambient temperature observed while testing, movement of vehicles and trees, and the reflection of the sun off of surrounding buildings. A possible solution to reduce these false triggers would be to attach a CDS photo cell to the light gates of the ePIR sensors. This would give a voltage to the light gate proportional to that of the ambient temperature.

One other design problem encountered was an excessive amount of noise in the voltage lines which seemed to be produced by the stepping motor circuitry while the motor was still with holding torque. This was a critical issue as the excess noise would not allow for the base station radio to distinguish packets coming from the node radios. A solution to this problem was to run a separate ground from the power source straight to the stepping motor circuitry. This seemed to reduce the noise and allow the base station 2.4GHz radio to receive packets from its node counterparts.

In conclusion, the design and implementation of the project was a success as all of the subsystems worked together to achieve the image capture of moving heat signatures. Our final system increased the viewing angle of the camera from 52 degrees to a full 360 degrees at a 40 foot radius, increasing the image capture range from 726 ft^2 to 5026 ft^2. This much broader scope should capture images of wildlife that would not be photographed by today's leading wildlife cameras.

Future work could include enhancements to the remote management functionality by providing custom software to provide more remote control features such as on demand image acquisition, automated multiple file transfer, and error and flow control options. This software could also provide the ability to view and catalog images dynamically. A more elaborate front end would include a command driven interface to provide camera file management tasks such as rename or delete, and camera image parameter controls such as resolution. This feature set would require additional hardware modifications to the base station to provide access to additional camera controls.

References

[1] "DS89C430/DS89C450 Ultra-High-Speed Flash Microcontrollers", MAXIM Semiconductor, April 05, 2007, [Online], Available: http://www.maxim-ic.com/datasheet/index.mvp/id/4078

[2] "XBee®/XBee-PRO®RFModules", Digi International,September 23, 2009, [Online], Available: http://ftp1.digi.com/support/documentation/90000982_B.pdf

[3] "XBee-PRO® 900/DigiMesh™ 900 RF Modules", Digi International, September 25, 2009, [Online], Available: http://ftp1.digi.com/support/documentation/90000903_C.pdf

[4] "Vinculum VNC1L Module Datasheet", Vinculum, March 03, 2010, [Online], Available: http://www.vinculum.com/documents/datasheets/DS_VDIP2.pdf

[5] "Instructions for I65 Game Camera", Moultrie, September 08, 2009, [Online], Available: http://images.ebsco.com/pob/moultrie/catalog/game-spy-i65-user-manual.pdf

Synthesis of Embedded Test System for Process Plants: A Proposed Architecture

Akin Cellatoglu and Karuppanan Balasubramanian
Department of Computer Engineering, European University of Lefke
Turkish Republic of Northern Cyprus, Mersin 10, TURKEY
(Email: acellatoglu@eul.edu.tr , kbala@eul.edu.tr)

Abstract - *An embedded test system to assess the dynamic performance of a process plant is proposed. A simple Process rate controller plant and a simple chemical plant are taken as models and the test systems are synthesized. In each system a virtual plant performing all tasks of real time plant is emulated and an analysis package to assess the performance is incorporated. The virtual plant is synthesized to provide anticipated ideal behavior and outcome of the plant. The responses obtained at selected output terminals of the real time plant and virtual plant are compared and analyzed in the analysis package of the test system. The test results assess the performance of the real time plant and suggest possible modifications to its electronic hardware and programs as to improve the dynamic performance of the real time plant.*

Keywords: Dynamic performance, process plant, process rate controller, test system , virtual controller.

1 Introduction

Process plants are designed to meet the production of materials and products meeting the required specifications. In real time process plants there are numerous varieties of parameters involved and many of them would be interactive in nature[1]-[3]. In some plants only few parameters are being monitored and controlled and in some cases even just one parameter is being controlled and monitored. This project envisages establishing a test system for systematic assessment of the process plants so as to exploit their full capacities. In order to explain the approach a temperature rate controller employed in several process plants is selected and implemented. This is based on the technique used for assessing the dynamics of a simple temperature controller reported[4]. Both approaches on ON-OFF control and continuos control were attempted.

The real time temperature rate controller keeps the reference temperature changing with time and follows a predetermined profile. The tempearture rate controller is emulated in the proposed test system and the responses of real time rate controller and emulated rate controller are analyzed. Also, a test system for a model chemical plant involving more parameters is synthesized. This study would enable us to optimize the performance of the respective real time plants.

2 System for Temperature Rate Controller Plant

Temperature rate controllers are widely being used in several production plants involved in producing materials needed for several industrial and commercial applications. While in operation they keep the process chamber to follow a definite preset temperature-time profile.

2.1 Profile of Temperature Rate Controller

A PC based programmable process rate controller was reported in the past[5]. Subsequently, on similar lines a microcontroller based process rate controller was employed in a project on remote control[6]. In principle the temperature time profile used for the temperature rate controller is having a structure as shown in Fig.1.

When the production is started the temperature *T* has to be maintained at different time rates shown. Four nodes of Temperature-time pattern are shown {*T1-t1, T2-t2, T3-t3 and T4-t4*} wherein *T1-t1* is the starting node and *T4-t4* is the terminating node. When the process is set ON it has to start with the temperature T1and the starting time is denoted as *t1*.

2.2 Real Time Temperature Rate Controller

At any instant of time of its functioning, the temperature is maintained at the set value in profile by employing ON/OFF control performed through a relay switching the electric heater. Fig.2 shows the simplified hardware schematic of the Temperature rate controller which controls the temperature rate to adhere to the profile.

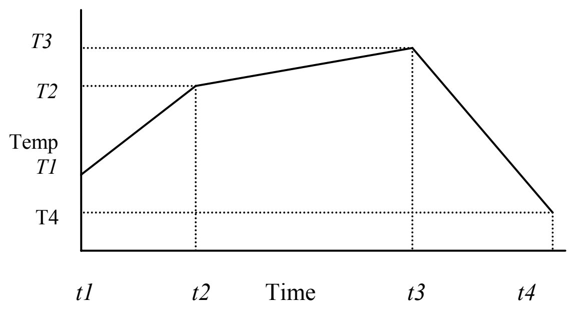


Fig. 1.Temperature Time Profile

A microcontroller is established with the processor 80286 to have standard RAM, EPROM, Keyboard, Display, Timer and Interrupt Interface.

2.2.1 Control Activity

The temperature of the process chamber is sensed in a sensitive Thermistor sensor and the signal is conditioned as in standard procedure and implementations. The sensed temperature signal is digitized with an 8-bit flash ADC (Analogue to Digital Converter) built under improved architecture [7]-[8] and read to the microprocessor.

Two interrupts are used for the purpose of ON/OFF control as to maintain the temperature-time profile. The Timer involved in the microcontroller hardware is programmed to give periodic pulse train at a set frequency. This pulse train is frequency divided at two levels such that 1Hz frequency and 1/20Hz frequency output are obtained. They are given as hardware interrupts to the microprocessor. While INTR1 is occurring once in a second the interrupt INTR2 is occurring once in 20 seconds. INTR1 reads the temperature of the processor chamber, checks it with the current temperature in the profile and makes the relay ON or OFF accordingly. It repeats its operation 20 times in 20 seconds before it takes up the next temperature from the profile. The main instructions used in the Interrupt Service procedures are shown in Fig.3.

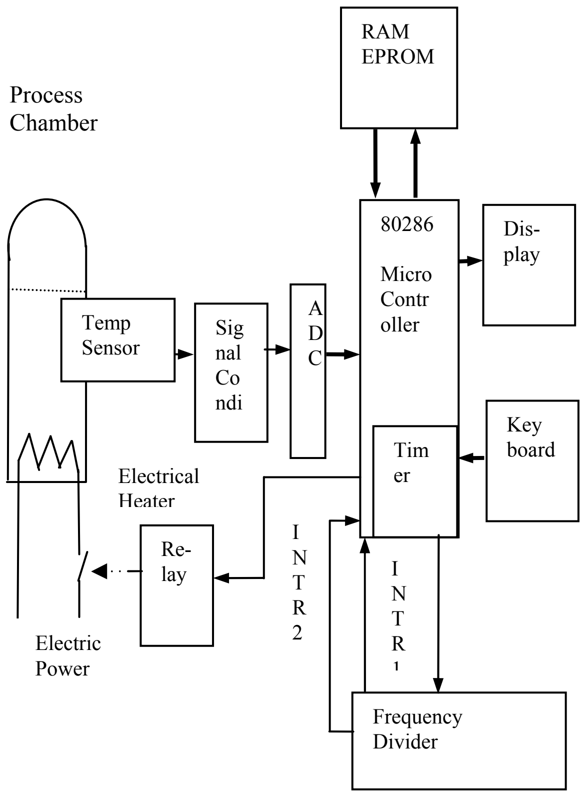


Fig.2. Simplified Schematic of the Temperature Rate Control Plant

The reference temperature is loaded to DH register from the lookup table where the temperature-time profile is stored. To avoid oscillations Hysterisis effect is utilized in switching. INTR2 routine reads the DH register from the table and increments the address in the table to take the next data. In the main routine the present time and latched temperature are displayed alternately in a period of one second each .

2.3 Test System for Assessing the Performance of Rate Controller Plant

There are three main functional units involved in the design of the test setup and analysis system; *i.* real time temperature rate controller *ii*. virtual instrument emulating the process rate controller and *iii*. analytical unit performing the assessment of the dynamics of the rate controller. Both the virtual instrument and the analytical unit are realized in a PC supported by its peripherals. The real time temperature rate controller performs the display routine and the maintenance of temperature rate is performed through interrupts. The test system for temperature rate controller is shown in Fig.4.

The real time temperature rate controller senses the current temperature of the chamber compares it with the reference temperature in the profile and switches ON or OFF the heater as to keep the chamber temperature same as that of the reference temperature.

```
;In the main routine
    MOV BX, OFFSET TABLE; Initialize
                  ;lookup table
    MOV  TIME, 0 ; Set the Running
                       ;Time as zero
  L1: CALL Display

 ;INTR1 service procedure
  INC TIME
  IN AL, 22H ; read temp t from plant
  INC DH; Keep hysterisis in switching,
             ; t: tr+Δt
  CMP AL,DH
  JNC  OFFRelay ;    t> tr+Δt
  DEC DH ;
  DEC  DH;  Keep lower level tr-Δt
  CMP AL,DH ;  Compare t with tr-Δt
   JC ONRelay ;    t < tr-Δt
OFFRelay:     MOV AL,0
     JMP L2
ONRELAY:  MOV AL,1
 L2:  OUT 24H, AL; control relay
 L3:    IRET

 ; INTR2 service procedure
   MOV DH, [BX]
   INC BX
   IRET
```

Fig. 3.Main Instructions in Service Procedures

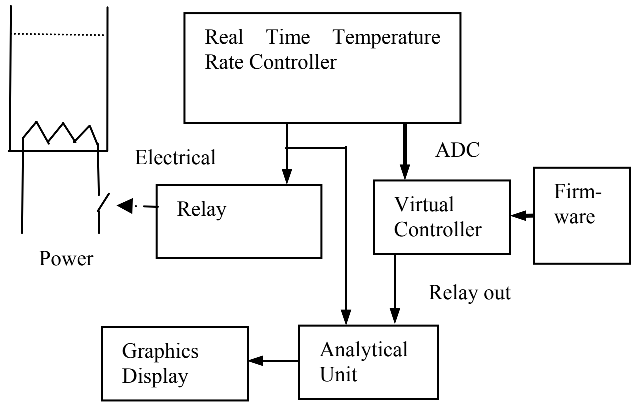


Fig.4. Test System for Temperature Rate Controller Plant

The digitized value of the sensed temperature is fed to the virtual controller to do the comparison with the reference temperature. The reference temperature is made available in a simulated manner from a firmware. The temperature-time profile is programmed in the firmware and it is read periodically matching the time span of the profile.

In the virtual controller, when relay is ON the temperature increases and the increase in process temperature with time is programmed in it. It calculates the next temperature in accordance with the current temperature level and the status of the relay. Also, if the relay is OFF the decay rate of the temperature is programmed accordingly. Wherever possible in the software lookup table approaches are implemented. As time progresses the change in status of the relay with time is provided by virtual controller and fed to the analysis package. Moreover, in the analysis package the pattern of the relay status of virtual controller and the real time controller are compared in time domain and results are drawn.

2.3.1 Virtual Package

Work bench software[9] and the software of Mathlab[10] installed in PC enables us to simulate the real time temperature rate controller as to form the virtual instrument. During analytical observation, both real time controller and virtual controller would be running simultaneously. The final control bit obtained from digital comparator of real time unit and that from virtual unit are taken for comparison and analysis. The virtual instrument is so programmed with the expectations of the controller yielding optimal results and solutions.

In the time domain part the relay status variation on time is put into correlation analysis and matching factor is determined. The matching factor would be unity when the relay status of real controller is exactly the same as that of the virtual controller. This factor is determined on various temperature levels and results are saved for estimating the needs for changing the hardware components or the software

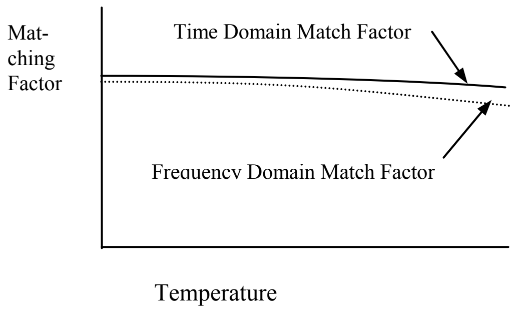


Fig. 5. Matching Factor Variation with Temperature

parameters of the real time temperature rate controller. On frequency domain analysis part the spectral contents of the pattern of the relay status of the real time controller and the virtual controller are evaluated and spectral match is determined. Here again the matching factor at various temperature levels are determined and saved. Fig.5 shows sample results.

2.4. Additional Tests Performed

The setup enables us to perform static and dynamic tests required to assess the performance of the real time temperature rate controller for suggesting possible changes needed to reach optimal performance.

2.4.1 Effect of Word Length

The ADC provides the binary word of current temperature in 8-b word length and the controller program also works with data of 8-bits word length. In order to perform the test on word length we need to tap the analog signal from temperature sensor and feed it to the programmable word length flash ADC[8] and get the binary word in different word lengths of 16-bits, 24 bits and 32 bits. In the virtual unit separate programs are made to deal the data in respective word lengths and the control bit driving the relay is obtained for each case. Also, processing is made in different word lengths for keeping the temperature profile. Then these results are compared with the results of the real time temperature controller working with a word length of 8-bits. The path in which the current temperature progresses in time is compared with the demanded temperature time profile and the magnitude of the error is computed.

$$Er = \sum_{n=1}^{N} \surd\, (1/Ts).(Ts\text{-}Tc)^2 \qquad (1)$$

where N is total number of samples, *Ts*= set temperature in the profile and *Tc*=current temperature trying to reach the *Ts*. A typical characteristic of error response with word length is shown in Fig.6. The error gets reduced with increase in word length and gets saturated after some length and therefore there

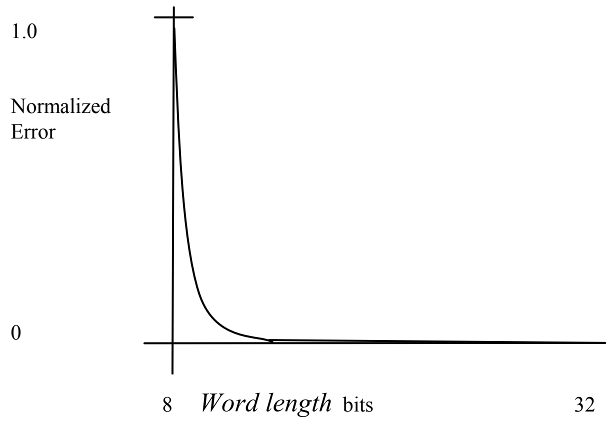


Fig.6 Error Response with Word Length

would be no further effect in error on increasing the word length.

3 Test System for Chemical Plant

The real chemical plant involving in producing chemical products such as pharmaceuticals have several inlets of raw materials to process chamber, performs their thermal and chemical treatments and takes the finished products through appropriate outlets[11]. In order to visualize the synthesis of test system for such real chemical plant, it involves in considering large number of parameters and their control actions. However, the principle involved in the synthesis of test system for a chemical plant can be presented by considering a simple chemical plant involving only few parameters for control. Fig.7 shows the schematic of such a simple model for the chemical process chamber. The synthesis of the test system for this chemical plant would furnish enough information for extending it to the synthesis of large sized real chemical plant.

There are two inlet pipes admitting the raw chemical fluids into the process chamber through valves V1 and V2 controlled from microcontroller. There is an outlet pipe taking out the processed chemical fluid whose flow is controlled by the valve V3. The chemical process demands the temperature of the fluid to be maintained at a reference level. This is achieved by sensing the temperature from a Wheatstone bridge having thermistor sensor and reading the voltage to microcontroller after digitizing the same. In the microcontroller it is compared with the reference temperature and a bit is issued to switch ON or OFF of the electric heater. In other words ON-OFF control with Hysterisis is used for maintaining temperature.

Likewise, the pressure signal and the level signal obtained from the respective transducers are processed based on set rules and stepper motors are driven to operate the control valves as to allow the inlet fluid to the process chamber and take out the output flow. The plant is running itself with the built-in program available in the firmware of the microcontroller. Therefore, the microcontroller senses the three parameters in real time and controls the inlet flow of raw chemicals and outlet flow of finished chemical fluid. The schematic of the microcontroller in simplest form is shown in Fig.8.

The analogue signals obtained from the respective transducers are given to a MUX-ADC such that any one signal could be selected to the ADC by the microprocessor. The control word for selecting a channel is issued from microprocessor through one of its output port. The selected signal from the three inputs is digitized by the ADC and read into microprocessor through an input port.

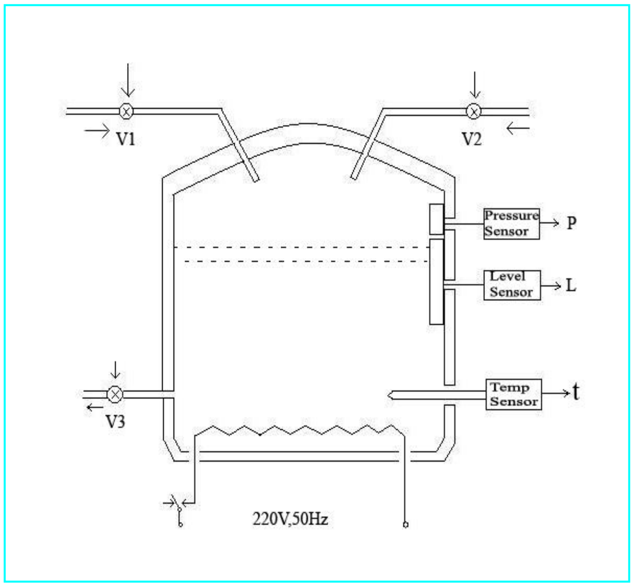


Fig.7 Schematic of Proposed Chemical Process Plant

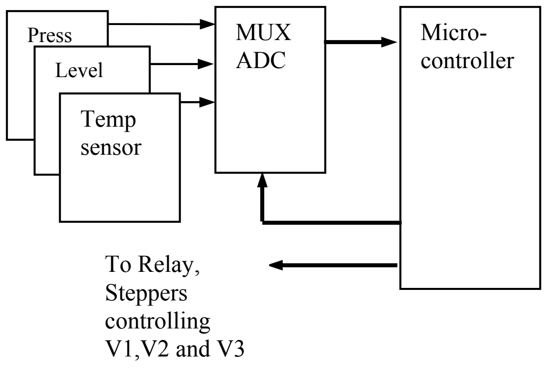


Fig.8 Schematic of Microcontroller Peripherals

```
ST:  MOV  AL,0
      OUT   40H,AL   ;    select temp t
       IN AL,42H        ;    Read temp
      CMP AL,[SI+1]  ;    Compare t with tr+Δt
      JNC L1              ;    t > tr+Δt
      CMP AL,[SI-1]  ;   Compare t with tr-Δt
      JNC L2              ;    t  with in Hysterisis band
      MOV AL,01       ;   set heater ON
      JMP L3              ;

L1  :  MOV AL,0        ;   set heater OFF
L3  :  OUT 4AH,AL    ;
L2  :  MOV AL,01       ;   set selecting Level
       OUT 40H,AL    ;   select Level
       IN AL,42H        ;   Read L
      MOV BX,Table1 ; set look up table for V1 ,V2
      XLAT                 ; Get data for V1,V2
      OUT 44H,AL      ; Control V1
      OUT 46H,AL      ; Control V2
      MOV AL,02       ;  Set selecting p
      OUT 40H,AL      ;  select p
      IN AL,42H         ;  Read p
      MOV BX,Table2; set look up table for V3
      XLAT
      OUT 48H,AL     ;        Control V3
      JMP ST;
```

Fig.9 Main Instrctions in Control Program

The microprocessor also provides data to control the stepper motors associated with the valves V1, V2 and V3. The processing of signals is following a sequence and it repeats periodically and cyclically.

While starting a cycle the temperature is processed first for ON-OFF control with Hysterisis and then the level signal is processed to control the inlet valves V1 and V2. Then pressure signal is processed to control the outlet valve V3. The control word for controlling valves is derived through lookup table approach. The control program in its simple form is shown in Fig.9.

3.1 Functionalities of the Test System

The test system for chemical plant takes the relay control bit and the status of stepper control words for controlling stepper motors from the real time plant and from the virtual plant. The running time and sequence are evaluated and the results extracted from the output are displayed in the CRT monitor. Its simple schematic is shown in Fig.10. The virtual microcontroller is synthesized based on the expected ideal performance of the plant and from the real performance of the best chemical plant run in the past. All mathematical models available for chemical reaction are applied and the final results are put in the form of lookup tables involving the parameters that could be accessed and controlled. It produces the relay bit status based on the present temperature and the reference temperature.

Also, keeping the information of level and pressure the chemical reaction results are applied to derive the word of control bits for stepper motors. It continues to produce the stepper control drive pattern sequence for all three valves by

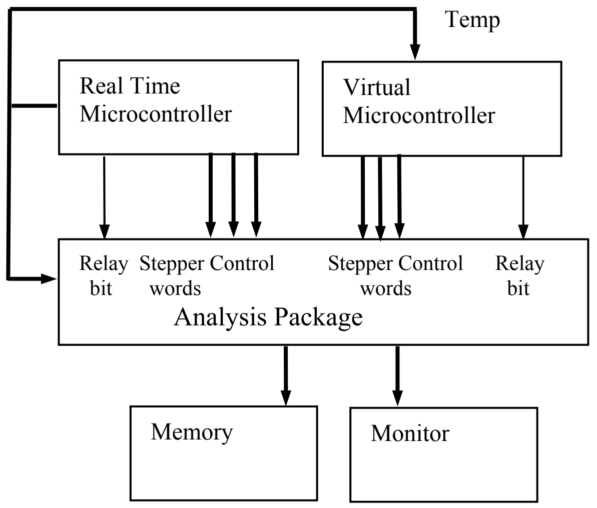


Fig.10. Schematic of the Test System for Chemical Plant

using the current values of temperature, level and pressure and lookup tables are employed for all these activities. The matching factors estimated in time domain and frequency domain are displayed and also saved for further analysis.

4 Discussions and Conclusions

Two models of process plants have been considered here and the test systems are established for assessing the dynamic performance of the plants. One is process rate controller plant and the other is chemical production plant. Although only few parameters have been considered for the synthesis of the test system the underlying priciple is good enough to provide required information for synthesis procedures of test systems of plants involving in large number of parameters. Therfore any process plant with large number of interactive parameters could easily be attempted to synthesize its test system.

The test setups enable us to achieve quality in the system implementation of the real time process plants as full knowledge of the dynamic performance of them could be obtained. The performance of the real time plants are compared with the virtual plants yielding ideal behavior and compensatory measures could be undertaken as to reach close to the ideal performance with the real time plants. This can be in the form of changes in the operating ranges of the hardware electronic components or in the control programs used. The test system also helps in performing quick fault diagnosis of the plants for counteracting unwarrented occurrence of faults, if any.

5 References

[1] S.Rangan, G.R.Sarma and V.S.V.Mani, Instrumentation Devices and Systems, Tata McGraw Hill, NewDelhi, 2003.

[2] Kocaarslan, I., Cam, E. and Tiryaki, H, "An Investigation of Cleanness in Boilers of Thermal Power Plants with Fuzzy Logic Controller", 2nd International Conference on TPE, TPE2004, September 2004.

[3] Ronald W. Breault, "National Energy Technology Laboratory's Advanced Gasification Program and FBC Update", Proceedings of Turkish American Clean Energy Conference, Istanbul, Jan 2008. http://www.turkey-now.org/db/docs/Clean_Energy_Conference_2008/

[4] A.Cellatoglu and K.Balasubramanian, "Assessment of the Dynamic Performance of Process Control Systems: A Case Study with Temperature Controller", Proceedings of 3rd International Conference on Electronics Computer Technology (ICECT 2011), Kanyakumari, India, April 2011, ppV4-19-V4-23.

[5] Z.G.Altun, K.Balasubramanian and K.Guven, "PC-based programmable process rate controller", Proceedings of the 1994 IEEE Instrumentation and Measurement Technology Conference, Hamamtsu, Japan, May 1994, pp 845-848.

[6] K.Balasubramanian and A. Cellatoglu, "Remote Control Techniques for Selected Applications Performed through Internet", Proceedings of the International Conference on Control, Automation, Communication and Energy Conservation, Perundurai, India, June 2009, pp 419-425.

[7] K.Balasubramanian, "On the design and development of flash ADCs", Journal of AMSE-Modelling Measurement and Control, A-series, France, 2003, Vol 76, No. 5, pp 31-43.

[8] K.Balasubramanian, "A High Performance Flash ADC with Programmable Word-Length", Proceedings of the 2006 International Conference on Real-Time Computing Systems and Applications, Las Vegas, USA, June 26-29, 2006, pp 1021-1027.

[9] http://www.electronicsworkbench.com/ Feb2011.

[10] http://www.mathworks.com/products/matlab/ Feb 2011.

[11] Paul Hass, An Introduction to the Chemistry of Plant Products-Vol-II: Metabolic Processes, Obscure Press, 2007.

An Embedded Architecture for Smart Wheelchair Navigation via Wireless Network

Y. Touati, A. Ali-Cherif, H. Aoudia and M. Demri
Computer Science and Artificial Intelligence Lab. LIASD dept. MIME
University of Paris 8 at Saint-Denis
2, rue de la Liberté, 93526 Saint-Denis Cedex, France
touati@ai.univ-paris8.fr

***Abstract*—One of the main trends in smart wheelchairs design is how to ensure a reliable remote tele-operation task with obstacle avoidance in a constrained environment. In this direction, various methods based on impedance control, potential field and edge detection have been investigated. These methods have advantage of making the fast motion planning for nearby obstacles, but with a shortcoming of getting into a local minimum where the attractive and repulsive forces are equal. To overcome the local minimum, a virtual impedance method is proposed, where a free vector is added to the repulsive force. The principle is to generate a virtual force between mobile system and environment using exteroceptive localization tools. In this paper, we are interested by this kind of approach to monitor a smart wheelchair through a wireless network based on 802.11 standards. Virtual forces are translated to a human operator through a joystick as tactile information. To illustrate the efficiency of the proposed approach, experimentation on a smart wheelchair developed in our Lab. called *LIASD-WheelChair* has been performed.**

Keywords-*Smart Wheelchairs; Virtual impedance; Wireless Network.*

I. INTRODUCTION

Several projects have been investigated in the last decade to conceive and to develop new hardware architectures for disabled and handicapped people's assistance [1]. TAO-2, is a project concerning autonomous wheelchair conception intended for people with various types and degrees of handicap [2]. The project aims to establish a methodology to design, implement, and test an effective add-on autonomy management system for use in conjunction with most common commercially available power wheelchairs. The Intelligent Assistive Technology and System Lab of University of Toronto is developing an assistive technology that is adaptive, flexible and intelligent, enabling people who have cognitive diseases to participate fully in their daily lives [3][4]. In this project, an anti-collision and navigation system for smart wheelchairs has been proposed in order to provide a safe and intuitive means of mobility. In [5], an intelligent assistance-system for an electrically powered wheelchair has been developed in order to facilitate driving a wheelchair for severely disabled and elderly people with heavily reduced physical and/or mental abilities. The approach is to make use of a cognitive model of the user to considerably reduce the amount of necessary user commands in order to enhance the dependability of the control of the wheelchair. This makes driving the wheelchair safe and reliable at a high degree. An UT-Intelligent Wheelchair Project has been investigated at University of Texas at Austin [6]. A notion of graceful motion is introduced for a robotic wheelchair motion that is safe, smooth, fast, and intuitive. In same manner, the wheelchair is experimented in complex navigational situations. Thus, for a given task, conventional joystick and proportional head-joystick can be used to assist the operator in his manoeuvring. Altering the translational and rotational velocities in situations where an obstacle blocks the user-commanded way, the driving assistance module significantly improves driver-performance by preventing all collisions along the way [7]. In the same requirements, low level approaches based control methods have been prospected in order to increase performances in terms of accessibility and autonomy.

Actually, one of the main trends in smart wheelchairs development is how to ensure a reliable remote tele-operation task with obstacle avoidance in a constrained environment taking into account system's interactive behaviour. For this purpose, various methods have been investigated such as edge detection [8], potential field [9] and impedance control [11]. In edge detection methods, the objective is to determine the vertical edges of the obstacle and consequently attempts to steer the mobile system around an edge. The main drawback is the proximity between the mobile system and the obstacles which is not suitable for the real time computation. Potential field methods allow a collision avoidance motion planning by generating a virtual force against obstacles and toward the goal. An attractive force between the mobile system and the target, and a repulsive force against an obstacle are generated. As it's mentioned in [10], all these methods have advantage of making the fast motion planning for nearby obstacles, but with a shortcoming of getting into a local minimum where the attractive and repulsive forces are equal. To overcome the local minimum, an extended virtual force field method is proposed, where a free vector is added to the repulsive force. This force becomes larger for larger obstacles since repulsive forces are coming from all obstacle-detecting sensors, which makes this method unsuitable for certain applications. Impedance methods which are based on the concept of active control constitute a considerable contribution to deal with issues cited above. Both free and constrained motions can be handled by controlling the relation between input commands and robot-environment contact forces. In the same manner, impedance methods based on virtual aspects [12] are used to determine a repulsive force for mobile robot navigation avoiding obstacles. The principle is to generate a virtual force between mobile system and environment using exteroceptive localization tools such as US sensors, LIDAR and so on. In the most cases, these generated virtual forces are transferred and translated to a human operator through a joystick as tactile information. In this paper, we are interested by this kind of approach to monitor a smart wheelchair through a wireless network based on 802.11 standards.

The remainder of this paper is organized as follows: In section 2, we describe the system overview describing hardware/software architectures of the developed LIASD-Wheelchair. Section 3 presents the proposed Virtual Impedance (VI) approach leading to control and to monitor LIASD-Wheelchair remotely. This kind of approach allows a reliable remote manoeuvring with obstacle avoidance in a constrained environment taking into account system's interactive behaviour. Simulations and experimentation results of our automated wheelchair are outlined in Section 4. Finally, a brief conclusion and future research are summarized in Section 5.

II. System Overview

A. Embedded System

As it illustrated on Fig.1, LIASD-Wheelchair is an adjustable adults' powered wheelchair [13][14]. It is suitable for indoor or outdoor use, has a range of 30km, and has the capacity to climb 35% slopes and 15cm kerbs. The wheelchair includes some standard features: a non-powered seating platform, adjustable removable armrests with embedded joystick, adjustable swinging leg-rests and a head-rest incorporating an embedded camera. The wheelchair is fitted with four ultrasonic (US) sensors to identify obstacles from distance of 3cm to 6m, with 3 to 4cm of resolution. Our system includes two optical incremental encoders called Line Drivers Encoders HEDL-5540#A06 with resolution of 500 Counts per Revolution.

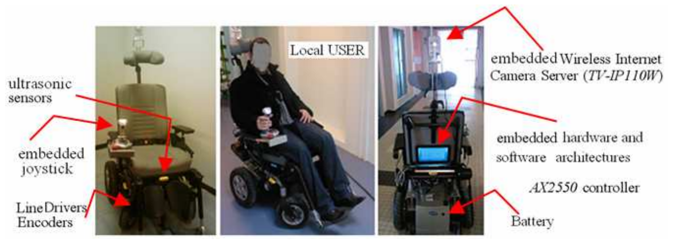

ultrasonic sensors
embedded joystick
Line Drivers Encoders
Local USER
embedded Wireless Internet Camera Server (TV-IP110W)
embedded hardware and software architectures
AX2550 controller
Battery

Figure 1. LIASD-WheelChair structure

Fig.2 shows different modules interconnection (remote computer, embedded laptop, different sensors, .etc.) for a better data flow using both wired and wireless networks. The wireless network is based on IEEE 802.11 standard with a router as an access point. Thus, the wheelchair can be controlled remotly using an external force-feedback joystick. An embedded laptop mounted at the rear of the wheelchair allows managing US sensors via Fiveco card and controls Roboteq card using respectively an Ethernet and serial RS232 links.

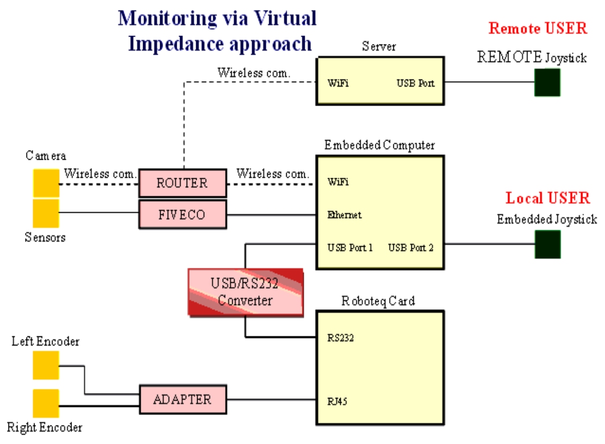


Figure 2. Interaction devices and communication scheme

In order to ensure navigation and anti-collision objectives a Wireless Internet Camera Server (*TV-IP110W*) is mounted on the wheelchair headrest. Images from the camera can be used to refine the odometry by computing the distance of objects from the wheelchair and then determine the safest routes to the desired location taking into account collision avoidance strategy. The wheelchair embeds also a USB joystick which is a little bit more sophisticated than traditional one. The joystick which is a *SpeedLink SL-6612* has many configurable buttons that makes the navigation easier.

B. Communication network architecture

As it's illustrated on Fig.3, an I2C protocol is used to manage US sensors module via a Fiveco card, and implement USB and RS232 communication bus for embedded laptop (*Asus-EeePC-1002H*) and *AX2550* controller interconnection. Via an adapter, incremental encoders are connected to Roboteq card trough an RJ45 link.

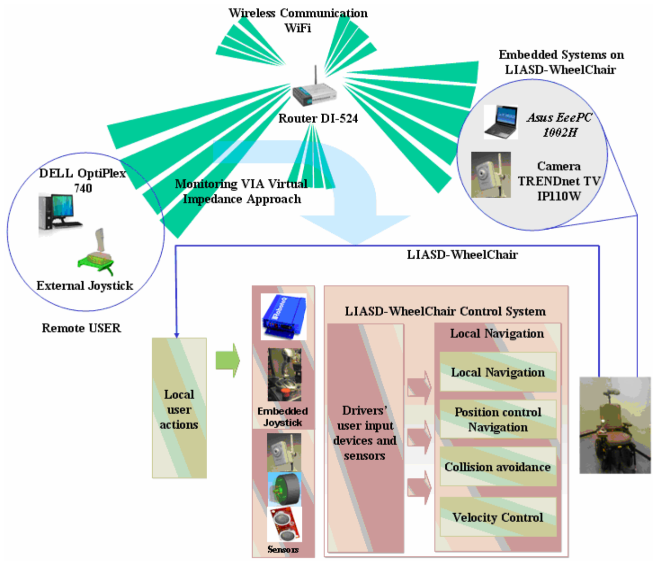

Wireless Communication WiFi
Embedded Systems on LIASD-WheelChair
Router DI-524
Asus EeePC 1002H
Camera TRENDnet TV IP110W
DELL OptiPlex 740
Monitoring VIA Virtual Impedance Approach
External Joystick
LIASD-WheelChair
Remote USER
LIASD-WheelChair Control System
Local Navigation
Local Navigation
Position control Navigation
Collision avoidance
Velocity Control
Local user actions
Embedded Joystick
Drivers' user input devices and sensors
Sensors

Figure 3. Communication network architecture

An embedded *Asus-EeePC-1002H* laptop use to control the whole architecture of the wheelchair, includes some features allowing a better information handling of different embedded modules (FiveCo, RoboTek, *TRENDnet-TV* camera,...) as well as those remotely located (server, routers ,...) via a wireless network.

In order to insure the safety of assisted people, a remote monitoring system using wireless network based on the IEEE 802.11 standard is implemented. Thus, an external joystick is requested to ensure the wheelchair monitoring and control continuously. This kind of control transition can happen in emergency cases particularly, when the wheelchair doesn't respond to some desired control actions. One can note that an approach based on virtual impedance method is implemented in this direction. Moreover, the use of a Wireless Internet Camera Server, which is mounted on the wheelchair headrest, enables the user to refine the odometry by computing the distance of objects from the wheelchair and then determine the safest routes to the desired location by avoiding collisions.

In this paper, we are interested by this kind of communication network.

III. VIRTUAL IMPEDANCE APPROACH

A. *Principles of the proposed approach*

Since the works of Hogan in [15], impedance control approach is considered as one of the most frameworks to control the interactions between system and environment. This kind of approach ensures a smooth implementation of desired tasks by taking into account some real impedance parameters properties such as inertia, viscosity and stiffness, so that:

$$F_e = M_e \cdot d\ddot{X}_e + B_e \cdot d\dot{X}_e + K_e \cdot dX_e \,. \qquad (1)$$

Where F_e describes the interaction force between system and environment and M_e, B_e and K_e represent respectively the desired inertia, viscosity and stiffness of the considered system.

The displacement between the current position of the system X_e and the desired one X_d, can be defined as follows:

$$dX = X_e - X_d \,. \qquad (2)$$

The principle of classical impedance is clearly illustrated bellow (Fig.4).

However, since no effort is exerted until making contact between system and environment, this approach is not useful [16]. Recent researches have been directed towards the development of new concepts based on VI principles using visual information [17][18]. Based on these works, and particularly on those developed in [11], we propose in this paper a non-contact impedance control approach for smart wheelchair monitoring through wireless communication based on 802.11 standard.

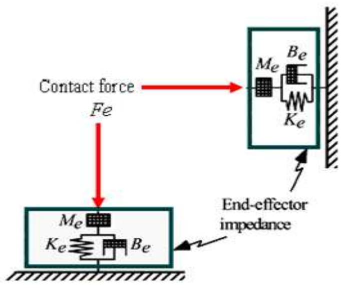


Figure 4. Impedance interpretation

As it's shown on Fig.5, VI is represented as spherical configuration between a mobile system and environment obstacles. Thus, when the mobile comes inside the virtual sphere, a virtual force is then generated before any contact.

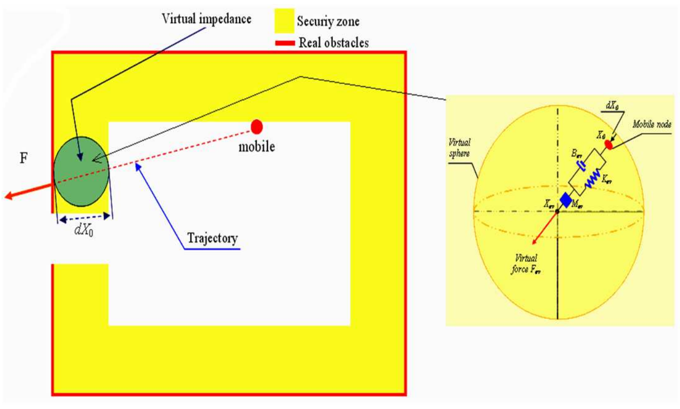


Figure 5. Virtual impedance representation

Let us consider now, the case when the mobile approaches a security zone, and set a virtual sphere with radius *r* at the center of the mobile.

When the mobile comes into the interior of the virtual sphere, the normal vector from the surface of the sphere to the mobile dX_0 can be written as:

$$dX_0 = X_r - r \cdot n \,. \qquad (3)$$

Where $X_r = X_0 - X_{ev}$ represents the displacement vector of the mobile from position X_0 to the center of the sphere X_{ev}.

The vector $n \in \Re^l$ is given by:

$$n = \begin{cases} \dfrac{X_r}{|X_r|} & \text{si } \left(|X_r| \neq 0\right) \\ 0 & \text{si } \left(|X_r| = 0\right) \end{cases} . \qquad (4)$$

Where, M_{ev}, B_{ev} and K_{ev} represent respectively the virtual inertia, viscosity and stiffness. One can note that the virtual force F_{ev} becomes zero when the mobile is not in the virtual sphere or is at the center of that sphere.

Furthermore, it should be mentioned that a Proportional, Integral and Differential (PID) controller using an AX2550 module which is embedded in the rear of the wheelchair is implemented also. Thus, a closed loop speed mode is performed using a full featured PID algorithm. The ultimate goal in well tuned PID parameters is to allow reaching the desired speed quickly without overshoot or oscillation. In that way, every 16 milliseconds, the controller can measure the actual motor speed and subtracts it from the desired position to compute the speed error. The resulting error value is then multiplied by a user selectable Proportional Gain. The effect of this part of the algorithm is to apply power to the motor which is proportional with the difference between the current and desired speed: when far apart, high power is applied, with the power being gradually reduced as the motor moves to the desired speed.

B. Software aspects

This approach is implemented according to some considerations concerning force-feedback joystick abilities to monitor remotely LIASD-WheelChair taking into account environment obstacles. As it is illustrated on Fig.6, the developed algorithm for our application shows that according to US sensors data, the perceived efforts as tactile information on the joystick allow the users to take right decisions to monitor the wheelchair.

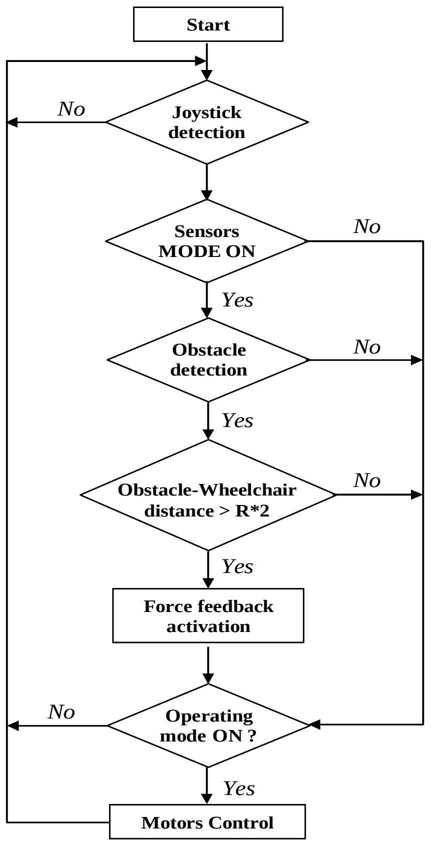


Figure 6. Implementation phases

In this context, as it's shown on Fig.7, we have developed a software architecture including a set of classes taking into account the application entirely. Thus, *G_SDL_Jostick* and *joy_impl* represent two classes developed to handle the force-feedback joystick module allowing the user to monitor remotely the wheelchair.

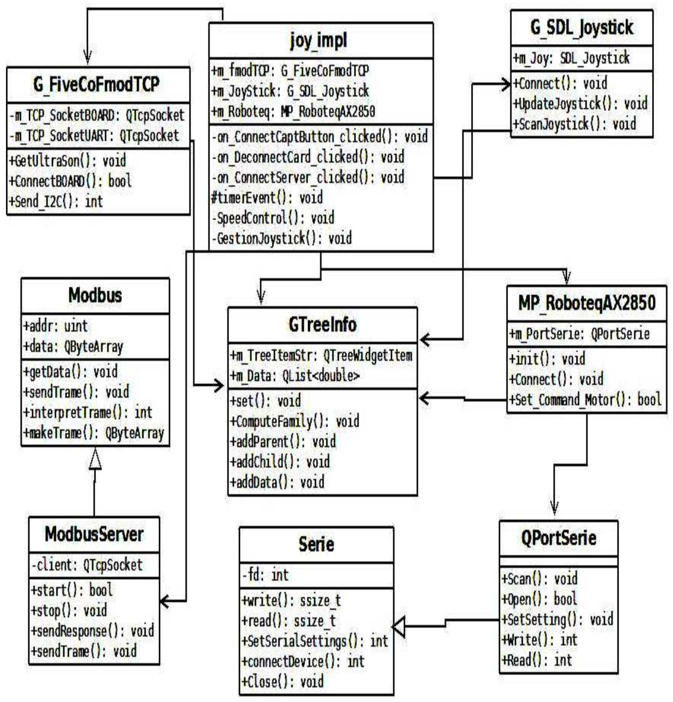


Figure 7. Software architecture

We note that these classes are developed with Qt and implemented under *directInput* tool.

IV. EXPERIMENTATL AND SIMULATION RESULTS

The objective is to perform some tests in order to analyze the behaviour of the LIASD-WheelChair interacting with environment. Thus, based on generated virtual forces that are transferred and translated through a joystick as tactile information, the human operator can perform a set of movements to maneuver the wheelchair. In this context, as a model of environment, we have proposed a circular configuration with a real behaviour model of LIASD-WheelChair. One can note that different models of environment can be chosen such as square, ellipse, etc.

First, VI parameters K_{ev}, B_{ev} and M_{ev} are set arbitrarily and respectively to the following values 50 [N/m], 10 [Ns/m] and 2 [kg]. Notice that the stiffness of the object is much higher that the positional stiffness of the impedance, so that the environment can be considered rigid. Moreover, for Sample-axis, each second corresponds to 100 Samples.

Thus, according to the obtained results, Fig.8 illustrates the exerted efforts by the joystick in the environment (Environment 1). Indeed, from sample 250, we note an increasing of the perceived efforts on the joystick corresponding to the first contact between the wheelchair and the virtual sphere. At this time, the reached value is 1100 and oscillates around 1000 and 1250. After sample 2000, the perceived force decreases to a minimum value of 400. Despite this, the intensity remains very weak.

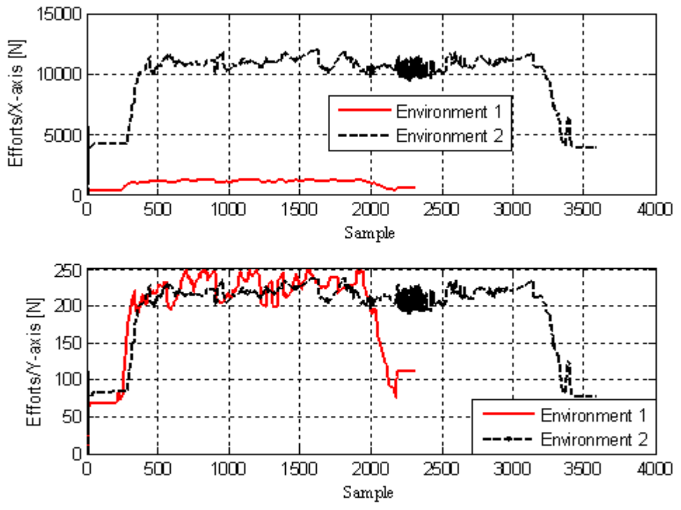


Figure 8. Exerted efforts on the environment

From Fig.9, one can see that the real trajectory of wheelchair (Environment 1) doesn't reflect the desired one, but it also goes beyond the specified environment. This can be explained by the fact that the virtual stiffness has a small value which leads to a weak force felt on the joystick.

To show the interest of VI method, we have simulated another implementation with different value of K_{ev}. Thus, we have integrated a value of 250 [N/m]. Thus, as we can see from Fig.8 and 9, the joystick set a real trajectory which doesn't go through the desired one (Environment 2). The fact that the stiffness value is significant, the perceived efforts are also significant (Environment 2) leading thus, to make the wheelchair follow the trajectory without any direct contact with real environment.

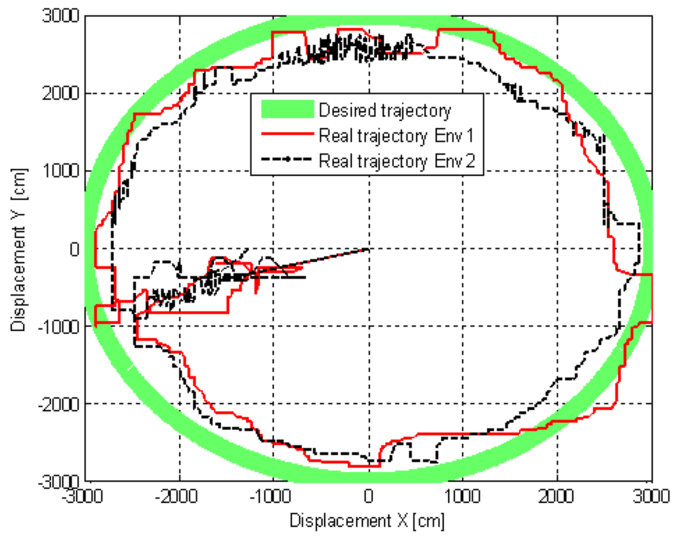


Figure 9. Desired and real trajectories

These results show that the increasing of the perceived efforts, from sample 300, to the value of 1000 corresponds to an elementary displacement dX_0 of 20 [cm]. This illustrates the first contact between the wheelchair and the virtual sphere. These efforts tend to oscillate respectively between 1200 and 25 according to X and Y-axis.

Fig.10 and 11 show respectively the wheelchair trajectories according to X-axis and Y-axis for different values of stiffness K_{ev}= 50 [N/m] and 250 [N/m].

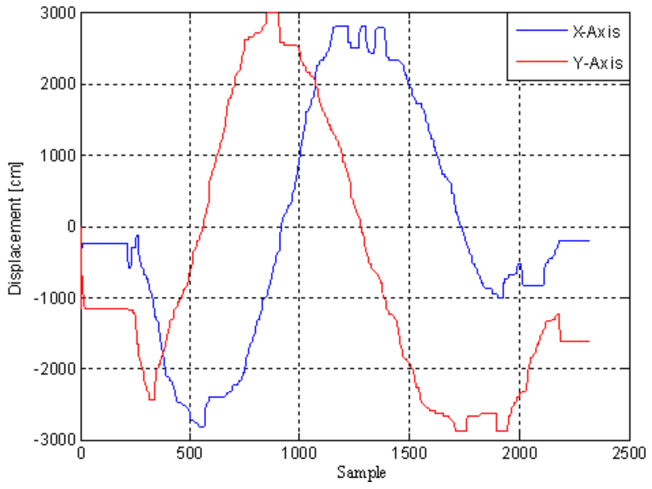


Figure 10. Real trajectories for K_{ev} = 50 [N/m]

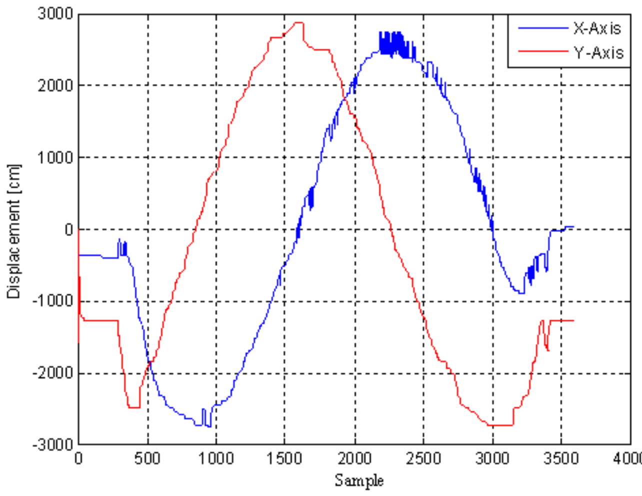


Figure 11. Real trajectories for K_{ev} = 250 [N/m]

To illustrate the usefulness of the proposed VI approach for LIASD-WheelChair remote monitoring experimentally, we have performed some tests. For this purpose, information coming from various sources, i.e., US sensors and wireless camera, through wireless network allow the human operator to ensure a right positioning a sweet maneuverability of the wheelchair.

Thus, Fig.12 illustrates different postures of the wheelchair. Here, the monitoring is ensured using both an external and embedded joysticks with identical characteristics. Thus, to ensure a safe control of the wheelchair according to some tele-operated movements of the joysticks, a specific PID parameters related to speed controller have been determined, so that, Proportional, Differential and Integral components values are respectively $K_p = 2$, $K_d = 2$ and $K_i = 2$.

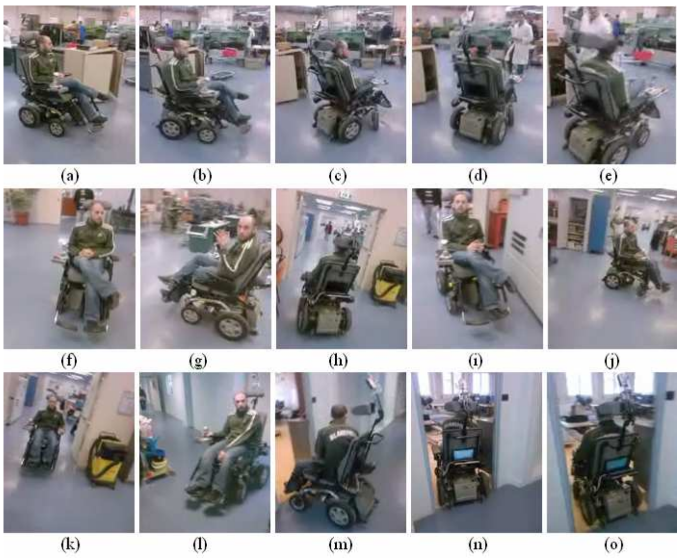


Figure 12. LIASD-WheelChair moving in constrained environment

Using Wi-Fi communication mode (Fig.12 (a)-(j)), the wheelchair performs a good trajectory in constrained environment. In this case, an external joystick is requested to ensure the wheelchair monitoring continuously. This kind of control transition can happen in emergency cases particularly, when the wheelchair doesn't respond to some desired control actions. Moreover, the use of a Wireless Internet Camera Server, which is mounted on the wheelchair headrest, enables the user to refine the odometry by computing the distance of objects from the wheelchair and then determine the safest routes to the desired location by avoiding collisions. In Fig.12 (k)-(m), the wheelchair is locally monitored by the user. Here, as in tele-operated mode, via a force feedback joystick the user perceives virtual forces leading him to have a better monitoring.

V. CONCLUSION

In this paper we have proposed an approach dedicated for improving handicapped people's assistance and particularly those concerning smart wheelchairs. We have presented an approach based on virtual impedance method leading to control and to monitor LIASD-Wheelchair remotely through wireless communication network which is based on 802.11 standards. This kind of approach allows a reliable remote maneuvering with obstacle avoidance in a constrained environment taking into account system's interactive behaviour. In order to test the effectiveness of the proposed approach, simulations and experimentation have been designed in this respect. Thus, according to the obtained results, some future researches and perspectives involving new techniques such as neural networks and fuzzy logic should be undertaken in order to increase performances in terms of accessibility and autonomy.

REFERENCES

[1] C. Gao, I. Hoffman, T. Panzarella, and J. Spletzer, ATRS - A Technology-based solution to automobility for wheelchair users, *6th Int. Conf. on Field and Service Robotics*, vol. 42, 2007.

[2] G. Takashi, The TAO Project: Intelligent wheelchairs for the handicapped, AAAI Technical Report FS-96-05, 1996, pp.28-37.

[3] A. Mihailidis, P. Elinas, J. Boger, and J. Hoey, An intelligent powered wheelchair to enable mobility of cognitively impaired older adults: An anti-collision System, *IEEE Transactions on Neural Systems & Rehabilitation Engineering, 15*(1), 2007, pp.136-14.

[4] P. Viswanathan, J. Boger, J. Hoey, P. Elinas and A. Mihailidis, The future of wheelchairs: Intelligent collision avoidance and navigation assistance, *Geriatrics and Aging Magazine, 10*(4), 2007, pp.253-256.

[5] M. Jipp, A. Wagner and F. Badredine, Individual ability-based system design of dependable human-technology interaction, *the 17th IFAC World Congress*, 2008, Seoul, South Korea, pp.15779-14784.

[6] S. Gulati and B. Kuipers, High performance control for graceful motion of an intelligent wheelchair, *IEEE Int. Conf. on Robotics and Automation*, 2008, Pasadena, California, pp.3932-3938.

[7] T. Röfer, C. Mandel & T. Laue, Controlling an automated wheelchair via joystick/head-joystick supported by smart driving assistance, *IEEE Int. Conf. on Rehabilitation Robotics*, ICORR'09, 2009, Kyoto, Japan, pp.743-748.

[8] J. Borenstein and Y. Koren, Obstacle avoidance with ultrasonic sensors, *IEEE Journal of Robotics and Automation*, vol.RA-4, n°.2, 1988, pp.213-218.

[9] J. Borenstein and Y. Koren, Real-time obstacle avoidance for fact mobile robots, *IEEE Transaction on System, Man and Cybernetics*, vol.19, n°.5, 1989, pp.1179-1187.

[10] L. Tao, Z. Haibing and H. Huosheng, An embedded control system for intelligent wheelchair, *Proc. of 27th Annual Int. Conf. of the IEEE Engineering in Medecine & Biology Society*, 2007, Shanghai, China, pp.5036-5039

[11] N. Hogan, Impedance control: An approach to manipulation: Part I, part II, part III, *ASME J. Dynam. Systems, Measurements and Control,* vol.107, n°.1, 1985, pp.1-24.

[12] J. Ota, T. Arai, E. Yoshida, D. Kurabayashi and T. Mori, Real time planning method for multiple mobile robots, *Proc. of the IEEE Int. Symp. On Assembly and Task Planning*, 1995, pp.406-411.

[13] Y. Touati and A. Ali-Chérif, Smart Wheelchair design and monitoring via Wired and Wireless Networks, *IEEE International Symposium on Industrial Electronics and Applications*, 2009, Kuala Lumpur, Malaysia, pp.920-925.

[14] Y. Touati, H. Aoudia, and A. Ali-Chérif, Intelligent Wheelchair localization in wireless sensor network environment: A fuzzy logic approach, *5th IEEE International Conference on Intelligent Systems*, 2010, London, UK , pp.408-413.

[15] N. Hogan, Stable Execution of Contact Tasks using Impedance Control, *IEEE Int. Conf. on Robotics and Automation*, 1987, Raleigh, North Carolina, USA, pp.1047-1054.

[16] Y. Tanaka, M. Terauchi, T. Tsuji and M. Kaneko, Online learning of virtual impedance parameters in non-contact impedance control using neural networks, *Int. Symp. on Flexible Automation*, 2002, Hiroshima, Japan, pp.807-812.

[17] T. Tsuji and M. Kaneko, Non-contact impedance control for redundant manipulator, *IEEE Transaction on Systems, Man and Cybernetics*, Part-A, vol.29, n°.2, 1999, pp.184-193.

[18] Y. Nakabo, I. Ishii, and M. Ishikawa, Robot control using visual impedance, *in Proceedings of the JSME Annual Conference on Robotics and Mechatronics,* JSME'96, vol.B, 1996, pp.999-1002.

Sub-Interval and Feed forward Techniques to Improve Signal Quality.

R. Dixit[1], **and H. Singh**[1]
[1]Department of Electrical and Computer Engineering, Wayne State University, Detroit, MI, USA

Abstract - *Noise on the signal line has been commonplace, and many researchers have been addressing the techniques to remove like-spectra noise from control signal. Degraded signal to noise can affect control system performance. Application of such schemes is wide ranging, including improving recognition accuracy of limited vocabulary speech interactive systems, improving speech based password access, and reducing noise pollution in many sensor and tracking applications – like radar. The problem is significant, as the characteristics of the noise sources and of the environment are often time varying. The frequency content, amplitude, phase and velocity of the undesired noise are non-stationary, and of similar spectra as the desired signal. In this work, we extend these techniques using two novel approaches, sub-interval spectral subtraction techniques for momentary-stationary noise and, feed-forward techniques for non-stationary noise. We show that it is possible to improve degraded signal to noise by up to 10dB, and thereby improve control system performance (in our example for key-word recognition) from 40 – 50% to better than 90%.*

Keywords: control noise, adaptive filtering, convergence algorithms.

1 Introduction – Active Noise Control

Additive noise in the control channel is a very commonplace occurrence. Most plant control systems utilize various strategies for characterizing and controlling noise. When the noise is out-of-band from the control signal, it is fairly trivial to filter out the un-wanted signals. But when the noise is in-band, and of similar spectra as the desired control signals, noise suppression is challenging. Typically, noise can be broadband, or narrowband, and can be stationary or non-stationary. The problem is very generic, and can affect any system, where the control signal is corrupted – and includes such applications as password recognition, range & track, and generally systems with low tolerance for false positives/ false negatives. As an illustrative example, in this research, control signals are assumed to be voice, and interfering noise – is assumed to be also acoustical. The 'plant' used for analysis is a limited-vocabulary key-word, speech recognition system. Many researchers have been looking at various means of noise suppression [1]. This particular investigation was directed towards active noise control. That is, for stationary noise, it is possible to develop anti-noise and thereby improve s/n, but for non-stationary noise, adaptive filtering is required.

In the context of signal processing, the term filtering refers to the linear process designed to alter the spectral content of an input signal in a specified manner. Filters, whose magnitude and phase responses satisfy certain specifications in the frequency domain, accomplish this task. Conventional filters are linear and time-invariant. They perform a constant set of linear operations on the data sequence $\mathbf{x}(n)$ to provide an output based on the coefficient values. In the case of adaptive filters, this restriction of time-invariance is removed. Adaptive filtering means that the filter parameters such as bandwidth and resonant frequency change with time. This is done, by allowing the coefficients of the adaptive filter to vary with time and to be adjusted automatically by an adaptive algorithm [2]. By allowing the coefficients of an adaptive filter to vary with time, it is possible to change bandwidth and resonant frequency.

2 The key-word, limited-dictionary Speech Recognition algorithms

For purpose of this paper, we pick the task of key-word recognition, using a speaker-dependent, pattern matching approach. This can be applied many varied classes of problems, including password verification systems as well as to radar return detection of arbitrary waveforms. The task is training a system, then, using that to recognize key words spoken by that, or other speakers. The challenges are that the training and use of the key-word may have time-warping issues. ie: the trained word may not be not time-identical to the usage word. For example, in radar returns, there are frequency shift issues. The input pattern consists of a vector sequence, derived from the speech input by some form of preprocessing, i.e. usually frequency spectrum analysis using Fast Fourier Transform (FFT). The comparison is done by calculating the Euclidean distance between the incoming speech pattern and all the reference patterns, after the time normalization of the patterns to a standard duration [3]. As developed by Baum-Walch [4], the hidden Markov model approach, is best suited for this pattern matching.

After filtering, ie cleaning the input signal, the task is straight forward:

- to Detect word-edge boundaries….this packetizes the input to region of interest

- to Code the input, time warp the input, into the stored-word timeline, and,

- to use Modeling to perform recognition.

MATLAB™ provides extensive tools for word edg detection, coding and for time warping, and using their HM and Fuzzy logic tool kits, it is easy to perform the speec recognition function. The flowchart shown in Figure 1 is suc a MATLAB evaluation model.

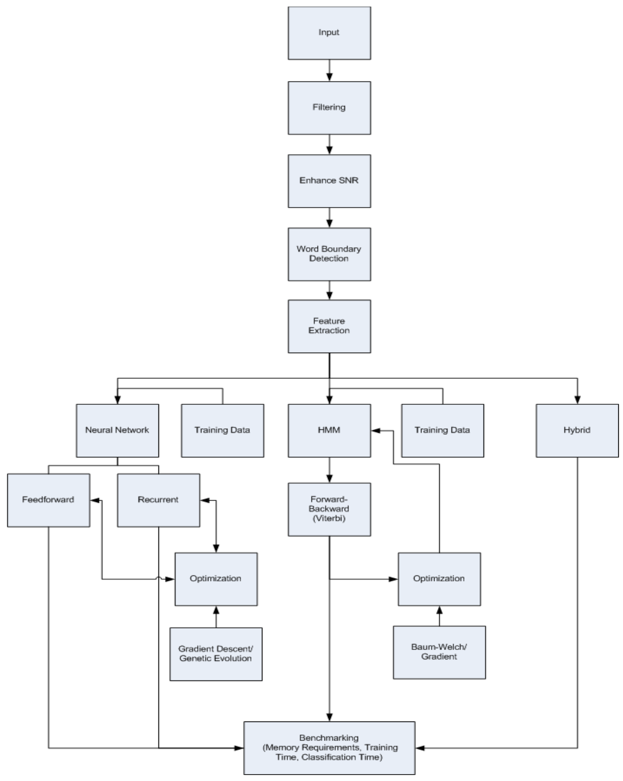


Figure 1. MATLAB algorithm for analysis.

Most error starts with inaccurate word edge boundary detection. Typical segmentation algorithms [5] look at the energy, zero crossing rate, and duration of the signal. Dynamic Time Warping and Hidden Markov models are applied to automatic word boundary detection, and Viterbi algorithm is used to segment signals. The algorithm, as implemented, is based on signal energy and adaptive thresholds.

The speech signals are essentially non-stationary signals over long period of time. However, speech can be considered as a quasi-stationary process over short periods of time (on the order of 20ms). If we take a 20ms segment of a speech signal, we can see that it shows a high degree of predictability due to this quasi-stationary property. Speech coders try to exploit this predictability to reduce the amount of data required and gain efficiency in speech recognition. The HMM techniques exploit this, 20 ms window segmentation approach to create the speech vectors. Figure (2) illustrates this.

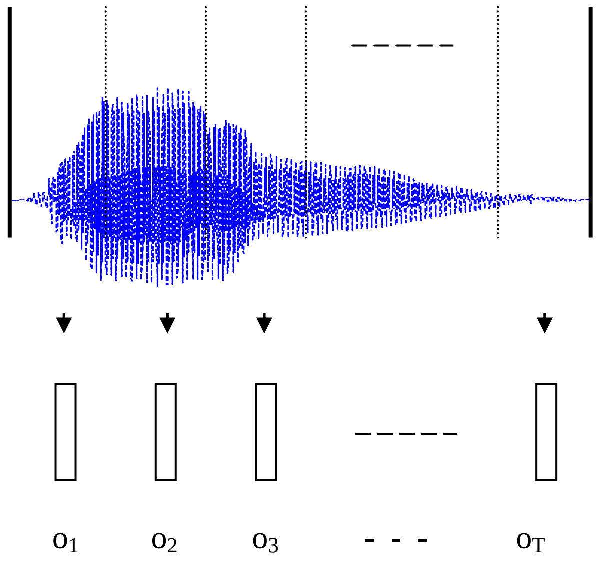


Figure 2 – Speech Coding

Many researchers [6] have adapted Linear predictive coding (LPC) is to code the speech signals, especially since during a short period of time; the speech signal shows a high degree of predictability. This predictability arises from the periodic repetition of the wave shapes. Linear predictors try to predict the value of the next sample of a speech signal, represented as a linear combination of the previous samples.

After extraction and coding, the next task in designing a speech recognizer is modeling and recognition of these patterns. A detailed explanation of Hidden Markov models is provided in [7].

Basically, the observations are probabilistic functions of the states rather than the states themselves. Unlike the states of a Markov model, the states of an HMM do not represent a physical event. The underlying stochastic process, the state sequence, is not observable, and can only be estimated through another set of stochastic processes that produce the sequence of observations. This work uses a left-to-right hidden Markov model.

The following steps are taken in building an HMM for a word:

1) Segmentation: The speech signal that contains the word is segmented using the algorithm above.
2) Coding: Segmented speech data is coded using linear predictive coding, which gives a set of speech patterns, O = $o_1, o_2, \ldots, o_T$.
3) HMM Modeling: An HMM is initialized by k-means segmentation into the states. The initialized HMM is trained using the Baum-Welch algorithm. After training, the probability of the speech patterns O increases. The training procedure is repeated until the probability of O is unchanged within a certain tolerance.

3 Putting it together - A Voice Activated System, with noise reduction, and results.

The speech signals were between 300 – 3500 Hz, sampled at 8 kHz. Over an arbitrary window of 128 ms. Some of the noise sources were time varying. Short time frequency transform (STFT) was used to characterize the temporal aspect of the signal. STFT of the word '*dial'* is shown in the three-dimensional plot of Figure 3. This section presents the 3-steps, cleaning the signal, coding the signal, and recognition.

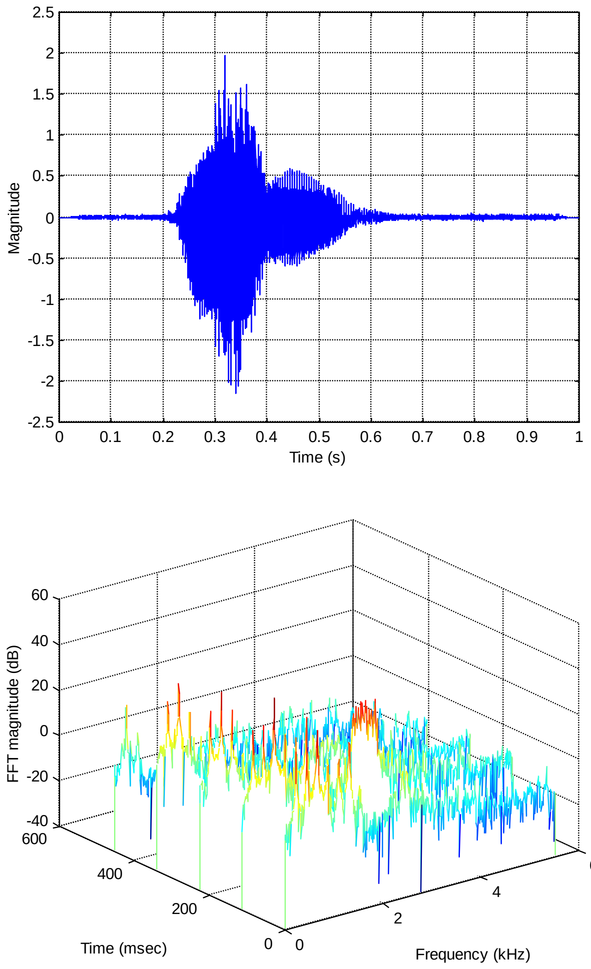


Figure 3. Short time frequency transform (STFT) of the word '*dial'*.

Both HMM and DTW based speech recognizers are speaker dependent. The training and the test data were the digits (0 to 9). Two recognizer models were used, including w and w/o DTW, and tested against four speakers under varying s/n. Figure 4 shows the results.

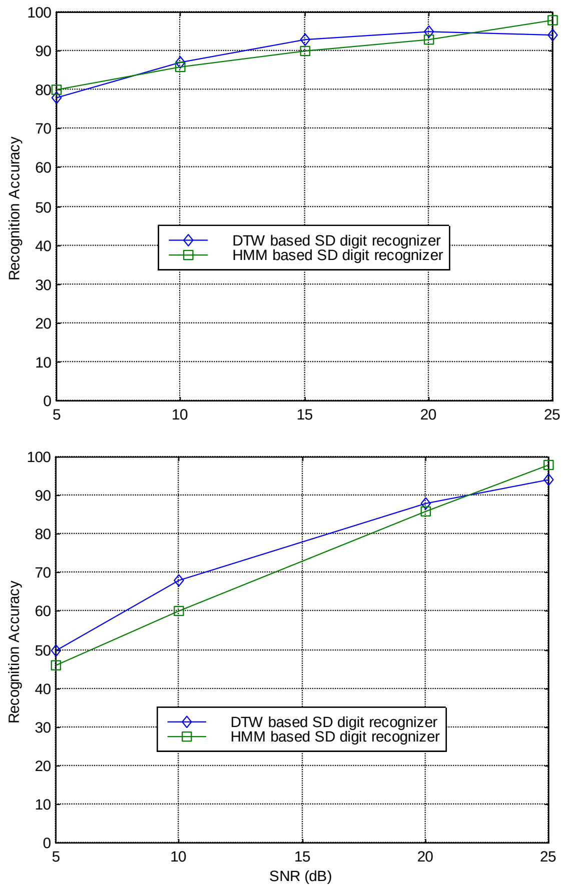


Figure 4 – Recognition Accuracy of DTW and HMM based SD Recognition Systems for test a and test b

In our example, low s/n can degrade key-word recognition accuracy to as low as 46%. In case of rapidly changing non-stationary noise, the situation will worsen. This paper discussed novel feed-forward implementation for pre-processing algorithms as shown in Figure 5.

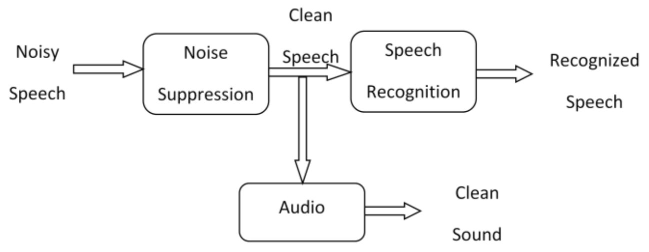


Figure 5 – Front-end Noise Suppression for Speech Recognition

Additionally, if we assume a windowed noise signal n(k) has been added to a windowed speech signal s(k), with their sum denoted by x(k). Then [8],

$$x(k) = s(k) + n(k) \qquad (1)$$

Taking the Fourier Transform of both sides gives

$$X(e^{jw}) = S(e^{jw}) + N(e^{jw}) \qquad (2)$$

The spectral subtraction estimate of *S(e ^{jw})* is

$$\hat{S}(e^{jw}) = [|\,X(e^{jw})\,| - \mu(e^{jw})]e^{j\theta}), where\theta = \theta_X(e^{j}$$

or

$$\hat{S}(e^{jw}) = H(e^{jw})X(e^{jw})$$

with

$$H(e^{jw}) = 1 - \frac{\mu(e^{jw})}{|X(e^{jw})|}$$

$$\mu(e^{jw}) = E(|\,N(e^{jw})\,|)$$

And, a novel spectral subtraction estimate of the speech signal *s(k)* is calculated by taking the inverse Fourier transform of the estimated spectrum of the signal,

$$\hat{s}(k) = \frac{1}{2\pi}\int_{-\pi}^{\pi}\hat{S}(e^{jw})e^{jwk}dw \qquad (4)$$

Spectral subtraction based noise suppression does not require additional microphones for noise spectrum estimation. Instead, noise spectrum is estimated from the signal during non-speech activity. As summarized by the above equations, during the speech activity the spectrum of noise is subtracted from the spectrum of the noisy speech to obtain clean speech. Here, it is assumed that the noise remains stationary during the speech activity and when the noise changes to a new stationary state, there exists enough time (on the order of 300ms) to obtain a new estimate of the noise spectral magnitude before speech activity starts again. Our results show that spectral subtraction techniques can improve s/n by up to 10dB. If the noise cannot be assumed to be stationary during the sampling window, then adaptive (multi-microphone) approaches can be employed. Other results show that this can further improve s/n [9]. The accuracy of a speech recognizer can be measured in different ways. One possible measurement can be done as follows:

$$\text{Re}cognition\ Accuracy = 100x\frac{\#\ of\ words\ recognized - \#\ of\ rejection\ errors/2}{total\,\#of\ words}$$

Recognition accuracy depends on various factors. SNR is one of the most important factors that affect the accuracy. Figure 6, shows that, the higher the SNR is the better will be the accuracy.

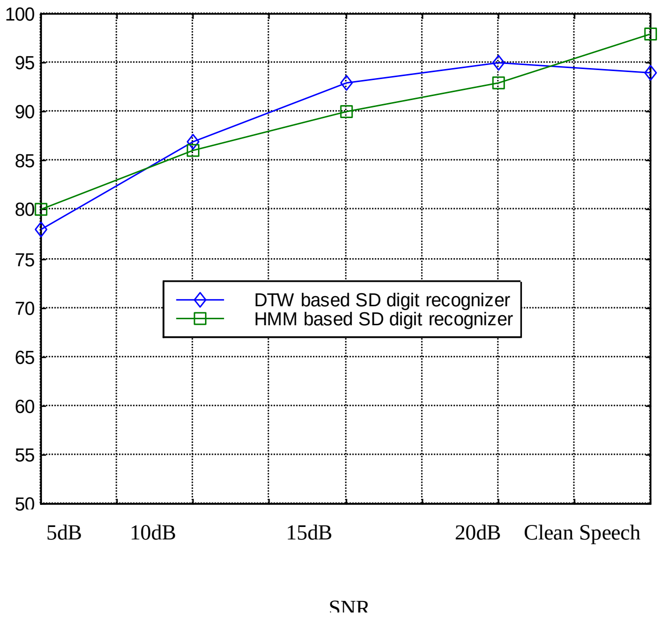


Figure 6 – Recognition Accuracy of DTW and HMM based SD Recognition Systems under several SNR ratios

4 Results

It was observed that the performance of the designed speech recognizer depends on the words that were chosen as commands. The speech recognizer was tested using ten words (the numbers *one* thru *ten*) The words with two or more phonemes were easier than recognizing short words. In addition, accurate recognition of short words is very sensitive to the boundaries found by the automatic boundary detection algorithm. Missing the actual boundaries by a couple of frames decreases the recognition accuracy. However, accurate recognition of long words is less sensitive to proper identification of the word boundaries. Missing the boundaries of long words by a couple of frames does not affect the recognition accuracy as much as it does with short words. The training data were ten words (the numbers *one* thru *ten*) , four utterances each. Testing consisted of any of these ten numbers. The effect of additive non-stationary noise and the recognizer performance was investigated [9]. The window length was 20ms, and 10 linear prediction coefficients were used to represent each window. Two state (N=2) left-to-right hidden Markov models were built for each word. It was observed that models from 2 to 6 states were appropriate. However, increasing the number of states does not necessarily increase the recognition accuracy. In general, it is appropriate to select the number of states to be equal to the number of sounds within the word. This is reasonable because speech vectors derived from a word are distributed among the states of an HMM, and the states of the HMM approximately correspond to different sounds in the word. Because most of the words used to test the recognizer had two phonemes, two state HMMs were used to model these words. The probability distribution of the observations accounted with each state of the HMMs were modeled using five Gaussian distributions (M=5).

The recognizer was tested in two ways:

1-) training and the test data were manually segmented,

2-) training and the test data were automatically segmented using the algorithm developed earlier. The recognizer is also tested under different levels of white noise conditions.

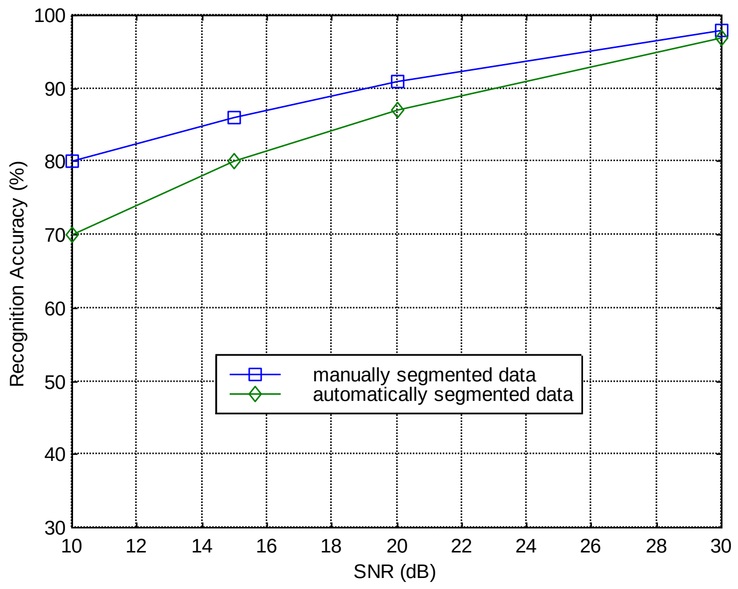


Figure 7 – Recognition Accuracy

Figure 7 shows the accuracy of the speech recognizer as a function of signal to noise ratio (SNR). The noise free data (clean speech) was recorded in a lab environment where there was actually a fair amount of noise. The accuracy of the recognizer with manually segmented data was 98%. When the noise free data was automatically segmented, the recognition accuracy of the recognizer was 97%. This accuracy is, for all practical considerations, almost equal to that of the manually segmented noise free data.

5 Conclusions

The issue about noise in the signal line is a significant control problem. Two techniques were investigated and modified to suit in-line speech recognition engines. These are the feed-forward, and in-situ spectral subtraction active noise control schemes. Dynamic time warping and hidden Markov model toolkits from MATLAB ™ implemented the speech recognition algorithms. We conclude that it is indeed possible to improve s/n and thus to improve word-recognition accuracy by greater than 90%. The recognition accuracy for manually segmented data was 98%. In order to automate this, we investigated a new segmentation algorithm to separate speech from noise and background noise. The recognition accuracy for the automatically segmented data was 97%, which is almost same as for the manually segmented data. This was the case when the data was noise free. When we added various noise to the speech signals, the recognition accuracy for the automatically segmented data was very close to that of the manually segmented data. Thus, we conclude that the new segmentation algorithm performs well, even in the presence of some noise. All the words were modeled using hidden Markov models with same number of states, N, and mixture coefficients, M. The testing was limited to speaker-dependent, isolated word recognizers. In fact, this can be easily adapted to a wide variety of signal processing applications.

REFERENCES

1. T. H. V. Pelt, R. Venugopal and D.S. Bernstein, “Experimental comparison of adaptive cancellation algorithms for active noise control”. Proceedings of the 1997 IEEE International Conference on Control Applications. Hartford CT. October 5-7 1997. pp. 559-564.

2. B. Widrow, M. G. Larimore and C. R. Johnson, “Stationary characteristics of SG adaptive filter”. Proceedings of the IEEE, 70. August 1996, pp. 1151-1162.

3. K. C. Zangi, “A new two-sensor active noise cancellation algorithm”. Proceedings of ICASSP Vol. II, 1993, pp 351-354.

4. Y. C. Sung, H. T. Chang, H. L. Yang and S. C. Lin, “Adaptive noise cancellation combined tree-structured sub-band filtering and cross talk adaptive filtering: A two microphone approach”. ISCPAT Conference proceedings, Toronto 1998, pp. 27-31.

5. S. J. Young and J. A. N. Flores, “Continuous speech recognition in noise using spectral subtraction and HMM adaptation”. Journal of Speech recognition, Vol 1, pp 409. 1999.

6. R. Lahouari; B. Abdelkader; M. Larbi, “Application of hidden Markov model and neural network approach for radar target detection**”.** 2005 ICSC Congress on Computational Intelligence Methods and Applications, 07 August 2006

7. Guoshen Yu, Stéphane Mallat, and Emmanuel Bacry, “Audio De-noising by Time-Frequency Block Thresholding”. IEEE TRANSACTIONS ON SIGNAL PROCESSING, VOL. 56, NO. 5, MAY 2008

8. Hai Huyen Dam, Sven Nordholm,, Siow Yong Low, and Antonio Cantoni, “Blind Signal Separation Using Steepest Descent Method”. IEEE TRANSACTIONS ON SIGNAL PROCESSING, VOL. 55, NO. 8, AUGUST 2007

9. L. R. Rabiner, M. R. Sambur, “An Algorithm for Determining the Endpoints of Isolated Utterances”. Bell Systems technical Journal, Vol 54, No. 2, pp. 297 – 315, February 1975

Health Monitoring Nano -Wear System for Astronauts

[1]Jennifer Rajkumari K, [2]T.Arun Prasad, [3]L. Nirmala Rani
[1] IEEE Member, Karunya University, Coimbatore, Tamil Nadu, INDIA
[2] IEEE Member, Silicon Technologies, Coimbatore, Tamil Nadu, INDIA
[3] Dept of Mathematics, Karunya University, Coimbatore, Tamil Nadu, INDIA

Abstract- *This paper discusses the novel sensor technology which aims at monitoring health status to improve comfort and efficiency of astronauts as well as to eliminate catastrophic failure to the individual and mission. Continuous ambulatory monitoring of vital signs will enable proactive personal health management and better treatment of astronauts. It proposes to develop a low-weight, non-invasive, fully-interconnected Nano system to be worn underneath the spacesuit without the complexity of multi-wire and Multi-locations for mapping crewmembers' health status. It can sense more than 7 vital parameters such as heart rate, electrocardiogram (ECG), Blood Pressure (BP), Respiration, Sp02, radiation, Phonocardiogram (PCG), Body Temperature for both extra and intra-vehicular activities with the help of nuclear battery. In this system, all parameters' signals can be processed by MSP430F2274 and displayed through TMS320 DSP controller with Novel architecture to meet space requirements.*

Keywords: Multi-locations, 7 parameters, sensor, Nuclear Battery, Nano, Microcontroller.

1 Introduction

Life of the flesh is in the blood [1]. Definitely the abnormalities & the disease will be reflected (will have variations) in the flow of blood. The pain and abnormalities in the body is due to the obstruction of blood in the blood vessels in the affected area that are sensed by the brain through our nervous system.

1. 1 Existing Health Monitoring Devices

There exists a large variety of situations in which noninvasive and continuous monitoring of physiologic and related parameters are extremely useful in a remote setting. Remote telemedicine [2], in-home care [3], patient transport (ambulance, aircraft [4]), military use, emergency worker monitoring (first responders) [5], as well as in- and out-of-hospital clinical monitoring [6-9], cardiac monitoring, sleep studies, clinical trials of medications) are but a few. For space applications [10], these include extravehicular activities (EVA), launch and deorbit, exercise in microgravity, physiologic research, and unexpected medical events.

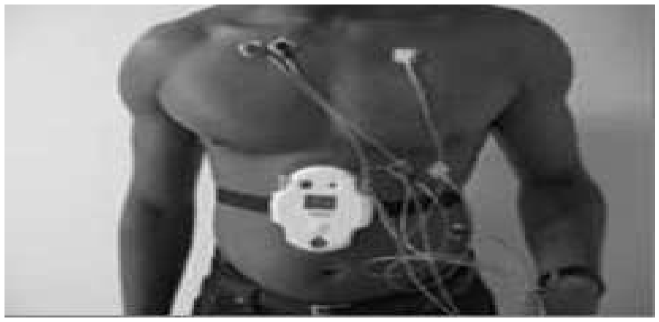


Fig. 1.1 Existing multisensory placements

Many of these applications require a rugged device, capable of daily use in an extreme environment due to pressure (hyperbaric, hypobaric), vibration (shuttle launch), radiation (on-orbit), temperature and humidity (emergency workers) or other environmental factors.

This automatic reflecting diagnostic system is bulky, expensive and it takes a professional medical doctor fluent with meridian points to interpret the results. Therefore, in the late 1990s, a lot of research was done in Taiwan to come up with a testing system that incorporated Dr. Volla's technology and the Traditional Chinese Meridian points and acupuncture, in a small enough packages that are affordable for the average person and also the results are interpretable to the average person. The bioelectric current, which is released by the meridian and the acupoint in the human body, can be read by the sensor connected to the computer and be statistically analyzed and correlated with the millions of clinical data records of the Data Center and the resulting report will be sent back to the computer. With the data being accumulated and revised continuously, the resulting report will be more and more precise and accurate. We give below some of the physiological parameter specifications that are in the existing health monitoring systems.

Fig1.2 Automatic reflecting diagnostic system

Table 1.1 Sensor Specifications

Parameter	**Sensor/Device**	**Range**	**Accuracy**
ECG (Lead II and V5)	Button electrodes	-	12bit
Respiration (Body-Impedance)	Button electrodes	-	12bit
Temperature (Skin)	FM50	-40C … +125C	+/-0.5C
Activity 2 x (2-axis Acceleration)	ADXL210E	+/-10g	2mg
SpO2 (Pulse Oximeter)	Nonin Xpod 3011	70…100%	+/-4 digits
Pulse Rate (Pulse Oximeter)	Nonin Xpod 3011	18… 300 pulses per min.	+/-3 digits
Blood Pressure (Cuff, Auscultatory)	Accutracker II	-	1mmHg

1.2 Disadvantages of existing systems

All these systems are bulky and during diagnosis we need an assisstant for measurements and some experts to interpret the results. More over in these systems, the person has to be in the lying position for ECG and for BP measuers the hand of the person should be at rest.

2. Proposed Novel System

It is a simple, pocket size, inexpensive, and a tiny battery-powered instrument. It consists of Wireless network system with *a single location* for the sensors to include 8 parameters that can process all the data and transmit them either simultaneously or individually to a display system worn on the wrist of the astronauts or in the ground control. *Our system is an excellent* one as it needs *zero preparation* for measurements during diagnosis.

2.1 Significant Location & Sensor Module

The *significant organ* of the blood flow is the *unique heart* in our human body. We have the smooth flat surface of the chest (*upper sternum*) as the significant location for the sensors, which is more suitable even in *life threatening situations* when blood circulation is limited to torso and head without any time delay of minimum 15seconds as usual in the existing health monitoring systems for the blood to reach diagnosis locations namely finger & wrist [11]

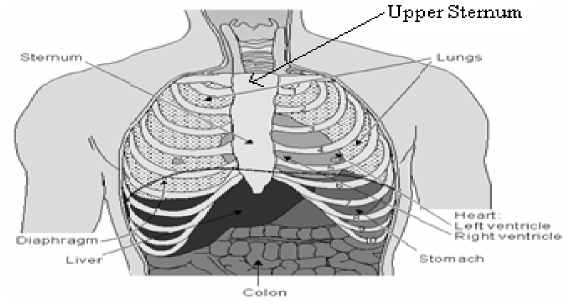


Fig 2.1.1 Upper Sternum for sensor location

In our novel system all the sensors are placed on *a single circular disc with angle specifications for each sensor*. Each sensor has a protective lid on its top so that, during the usage of an individual sensor , the mixing of other sensors' signals can be avoided.The ECG copper fill sensors are provided with adjustable extension holders with scale measurement. Under this circular disc all the signal processing circular discs are kept one below the other with microcontroller for signal processing. There is a button system for the projection of each individual sensors during its usage

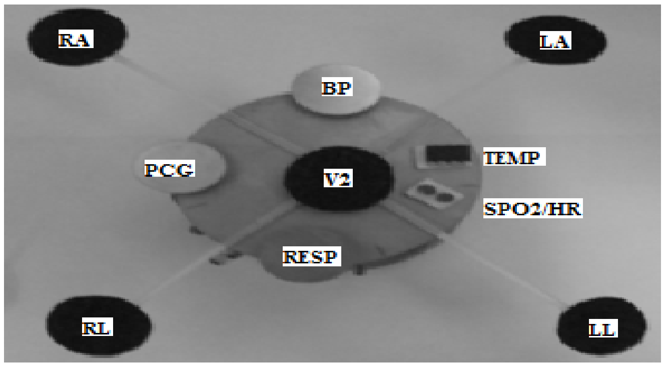


Fig 2.1.2 Top View with Sensors

The sensor system can be firmly and compactly fixed on the chest at all times throughout the journey both in & out of the space shuttle if needed. The sensor operation for a specific diagnosis can be triggered and controlled by remote system of either the individual astronauts in the shuttle or ground control station. The description of each parameter and its function in our novel system is dealt below.

2.2 ELECTROCARDIOGRAM

An electrocardiogram is an instrument that measures the *electro-mechanical activity* of the heartbeat. With each beat, an electrical impulse (or wave) travels through the heart. This wave causes the muscle to squeeze and pump blood from the heart. The ECG (Electrocardiogram) sensor measures *cardiac electric potential waveforms* (voltages produced during contractions of the heart).

2.2.1 Problem Identification:

Normally in the surface of the heart, muscle *action potential* is 20mV. It withstands up to 150-300ms at a frequency of 0.05-100Hz. As, the *action potential in the surface of the chest is only 1mV,* we need a *highly sensitive single located sensor instead of 5 electrodes that are located* at *different parts* of the body namely chest, hands and legs to pickup the signals of RA, LA, LL, RL and V2.

The sensor in our proposed system is a set of non-contact *bio potential copper fill capacitive type electrodes* [11] placed at the top circular disc with angle specifications to pick up signals of leads I, II, III, aVR, aVL, aVF and V_2. *In our novel system the person need not lie down.*

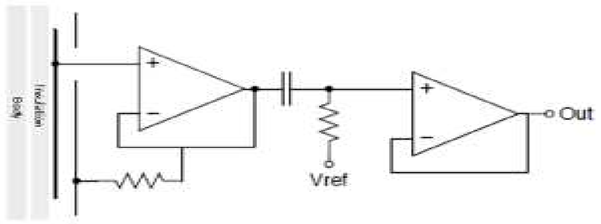


Fig 2.2.1 Copper fill capacitive type electrodes with amplifier unit

2.3 RESPIRATION

Respiratory rate is the *number of breaths* a human being takes *per minute*. It is usually measured when a person is at rest by counting the number of times the chest rises per minute. The Ultra-Piezo Sensor generates a small voltage signal from the *normal expansion and contraction* of the chest or abdominal wall. This voltage is immediately passed through an electronic filter before the respiratory signal is applied onto a physiological monitoring or recording system.

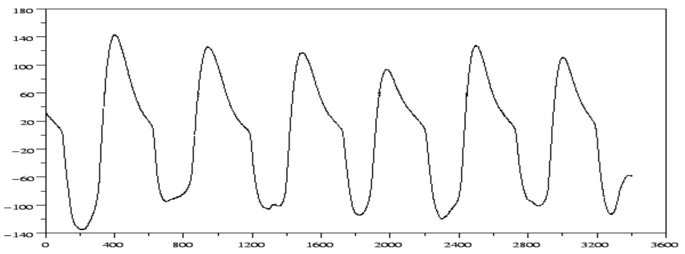


Fig 2.3.1 Continuous Respiration signal

Since the sensor *generates a voltage* when stressed by breathing, *no battery* is required for operation. This system is convenient, non-constraining, comfortable and durable for sleep disorder testing also.

Fig 2.3.1 Ultra Piezo Sensor

2.4 BLOOD PRESSURE

Blood pressure (BP) is the *pressure exerted* by circulating blood, upon the walls of blood vessels, and is one of the principal vital signs. During each heartbeat, BP varies between a *maximum* (systolic) and a *minimum* (diastolic) pressure.

Arterial hypertension may have adverse effects. Persistent hypertension is one of the risk factors for strokes, heart attacks, heart failure and arterial aneurysms and is the leading cause for chronic renal failure. Even moderate elevation of arterial pressure leads to shortened life expectancy.

Our proposed system is a *chest based continuous and non-invasive* method of monitoring and measuring blood pressure with display of *continuous graph of systolic and diastolic readings every second*. The design of continuous method provides a better monitoring of rise and falls in the blood pressure values. This device can be used so that the risk of higher and lower blood pressure can be eliminated and suitable precautionary measures can be adopted before the condition gets critical.

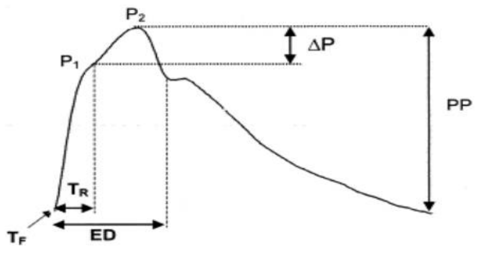


Fig 2.4.1 Analysis of Blood Pressure Curve

2.5 HEART RATE

Heart rate is the number of heartbeats per unit of time, typically expressed as *beats per minute* (bpm).

The *R wave to R wave interval* (*RR interval*) is the inverse of the heart rate.Heart rate is measured from the chest (apex of heart), which can be felt with one's hand or fingers.

In our method the heart rate (HR) is readily calculated from the ECG as follows:

HR = 1,500/RR interval in millimeters,

HR = 60/RR interval in seconds,

or HR = 300/number of large squares between successive R waves.

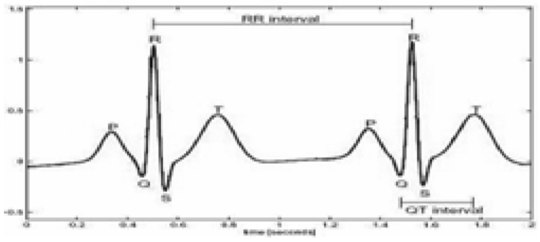


Fig 2.5.1 Heart Rate Analysis

2.6 TEMPERATURE

As a precision CMOS temperature sensor, the *FM50* is cost effective for *accurate, low power;* temperature monitoring applications. Output voltage versus temperature is extremely linear. With no load, the supply current is typically 130µA. For normal operation, the load on V_{OUT} should be 100KΩ or less.

In a typical application, a remotely mounted FM50 is monitored by a microcontroller unit (MCU) with an analog A/D converter input. Alternatively, the FM50 can drive a comparator with a high-impedance input.

Accuracy is typically $\pm 0.5^oC$ at room temperature, and better than $\pm 2^oC$ from 0 to 75^oC. FM50 is available in a 3-pin SOT-23 package.

2.7 Sao2

This paper presents a prototyped novel *chest-based* Pulse Oximetry system. It reports on test results from comparative trials with a commercially available finger-based Pulse Oximetry system using several human subjects. In our chest based novel system a *reflective sensor* is used where the LEDs and photodiode are mounted *beside each other at the center of the sensor circular disc*. Initially, a simple sensor arrangement is realized by mounting a bi-colour LED (SMT660/910) and a blue enhanced PIN silicon photodiode (PDV-C173SM) onto a PCB board. This enabled the optimal gap size between both components to be experimentally determined during later testing. The main circuit consists of two amplifier stages (figure 2.7.1). In the trans-impedance amplifier stage the PIN photodiode is placed across both inputs. In the second stage a DC offset removes a large DC component of the photodiode signal before further amplification. Two readings are taken by a 12-bit ADC module when one LED is switched on. The ADC readings are distinguished by the MCU *into two signal pairs* (red and infrared).

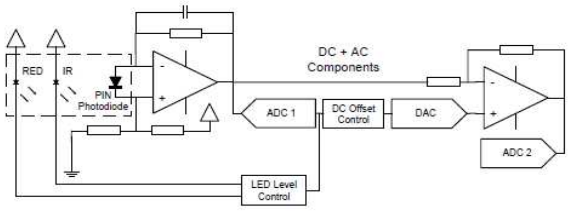


Fig 2.7.1 Input front end circuit and LED control of the single-chip Pulse Oximeter design using the MSP430.

Each *signal pai*r is processed by the MCU to calculate the current for the individual LED and the required DC offset. The *LED current* is used to control the *illumination of tissue* with the aim to obtain the largest possible peak-to-peak amplitude in the amplified photodiode signal. ADC readings of the second amplifier output produce a red and infrared *peripheral pulse graph* (*PPG*). The baseline of both raw PPG signals varies and sudden signal distortions and baseline shifts are mainly induced by motion artefacts (breathing, movements of the sensor on the chest, etc.). The noise in both PPG signals differs from subject to subject and also depends on the environment (ambient light, electromagnetic interferences (EMI)). The DC residuals and noise in both raw signals are reduced through filtering with a high and low pass digital filter.

2.8 Phonocardiograph

A Phonocardiogram or PCG is a *plot of high fidelity recording of the sounds and murmurs* made by the heart with the help of the machine called phonocardiograph, or "*Recording of the sounds made by the heart during a cardiac cycle*." The *sounds* are thought to result from *vibrations created by closure of the heart valves*. There are at least two: (i) when the atrio ventricular valves close at the beginning of systole (ii) when the aortic valve closes at the end of systole. It allows the detection of sub audible sounds and murmurs and makes a permanent record of these events. In contrast, the ordinary stethoscope cannot detect such sounds or murmurs, and provides no record of their occurrence.

A *piezo electric heart sound transducer* is used to pick up the *vibrations of the four heart valves*. The PCG is then interfaced to the display unit for processing after which the *frequency spectrum and time period analysis* is done. The PCG along with its data is provided on the screen for detecting the disorders associated with the heart Valves. The technique of phonocardiography has evolved continuously to grab an important role in the proper and accurate diagnosis of the defects of the heart. This technique, though seemingly quite reliable, is quite difficult to master. As with the stethoscope, it requires highly educated professionals to read the PCG signal. [12]

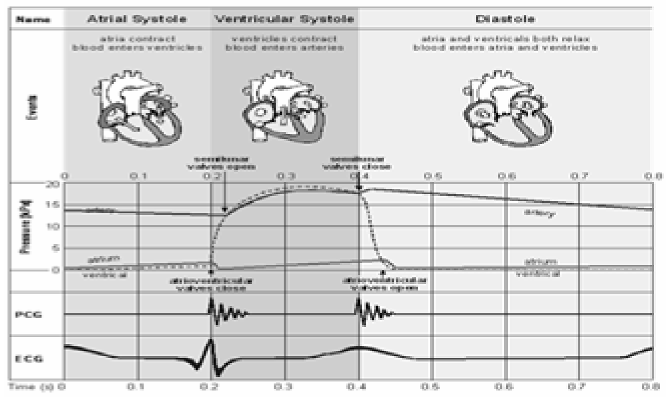


Fig.2.8.1 Cardiovascular signals comparison

3. Signal Processing and Transmission

A small *base unit* powers the entire system by *nuclear battery* and signal processing is done using MSP430F2274 and with the help of *wireless transmitter* like *zigbee* the data is sent to either *TMS320 based receiver* or other external device for display.

Each parameters' sensor is connected to a signal processing circular disc (with on / off button) kept one below the other sensors disc. Data in each signal processing disc is transmitted to the display unit worn on the wrist or to remote ground control unit

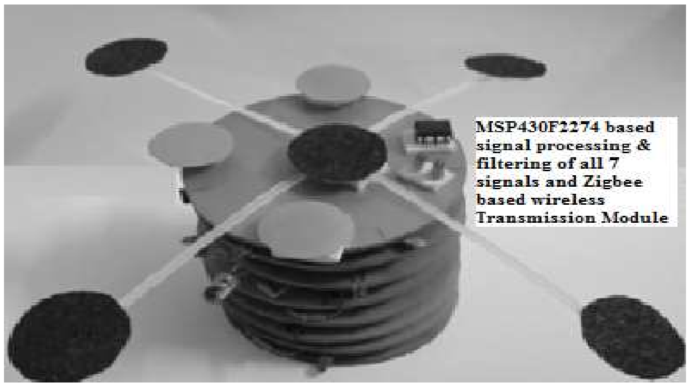

MSP430F2274 based signal processing & filtering of all 7 signals and Zigbee based wireless Transmission Module

Fig 3.1 Lateral View of Signal Processing Unit

3.1 Nuclear Battery

In our novel system we use *3V nuclear battery* for many reasons.In space applications; nuclear power units offer *advantages over solar cells, fuel cells and ordinary batteries.*

Fig 3.1.1 Nuclear Battery

When the satellite orbits pass through *radiation belts* such as the Van- Allen belts around the Earth that could *destroy the solar cells*. Operations on the moon or Mars where long periods of *darkness require heavy batteries* to supply power when solar cells would not have access to sun light.

Space missions in opaque atmospheres such as Venus, where *solar cells would be useless* because of lack of light. At distances far from the Sun, for long duration missions where fuel *cells, batteries and solar arrays would be too large and heavy*. Heating the electronics and storage batteries in the deep cold of space at –245°C is a necessity. So in the future it is ensured that these *nuclear batteries will replace all the existing power supplies due to its incredible* advantages over the other. It is quite sure that the future will be of '*Nuclear Batteries'* because of the applications which require a *high power, a high life time, a compact design over the density, an atmospheric conditions-independent power supply*. NASA is on the hot pursuit of harnessing this technology in space applications.

3.2 Mixed Signal Microcontroller MSP430F2274

The Texas Instruments MSP430 family of ultra low power microcontrollers consists of several devices featuring different sets of peripherals targeted for various applications. The architecture combined with *five low power modes* is optimized to achieve extended battery life in portable measurement applications. The device features a powerful 16-bit RISC CPU, 16-bit registers and constant generators that attribute to maximum code efficiency. The digitally controlled oscillator (DCO) allows wake-up from low-power modes to active mode in less than 1 ms.

The MSP430F2274M series is an ultra low-power mixed signal microcontroller with two built-in 16-bit timers, a universal serial communication interface, 10-bit A/D converter with integrated reference and data transfer controller (DTC), two general-purpose operational amplifiers in the MSP430F2274M devices and 32 I/O pins. Typical applications include sensor systems that capture analog signals, convert them to digital values and then process the data for display or for transmission to a host system. Stand-alone RF sensor front end is another area of application. *Available in Military Temperature (–55°C/125°C) Range.*

3.3 Zig bee

Zig Bee is a specification for a suite of *high level communication protocols* using small, low-power digital radios based on the IEEE 802.15.4-2003 standard for Low-Rate Wireless Personal Area Networks (LR-WPANs), such as wireless light switches with lamps, electrical meters with in-home-displays, consumer electronics equipment via short-range radio needing low rates of data transfer. The technology defined by the Zig Bee specification is intended to be *simpler and less expensive* than other WPANs, such as Bluetooth.

Zig Bee is targeted at radio-frequency (RF) applications that require a *low data rate, long battery life, and secure networking.* Zig Bee is a low-cost, low-power, wireless mesh networking standard. The *low cost* allows the technology to be widely deployed in wireless control and monitoring applications. The *low power-usage* allows longer life with *smaller batteries*. The *mesh networking* provides high reliability and more extensive range.

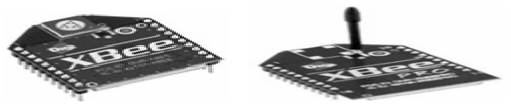


Fig 3.3.1 Zig Bee Module

The software is designed to be easy to develop on small, inexpensive microprocessors. The radio design used by Zig Bee has been carefully *optimized for low cost* in large scale production. It has few analog stages and uses digital circuits wherever possible.

3.4 TMS320 DSP controller

The TMS320C672x is the next generation of Texas Instruments' C67x generation of high-performance 32-/64-bit floating-point digital signal processors. The TMS320C672x includes the TMS320C6727, TMS320C6726, and TMS320C6722 devices. *C67x+ is an enhanced* version of the C67x CPU used on the C671x DSPs. It is compatible with the C67x CPU but offers significant improvements in speed, code density and floating-point performance per clock cycle. *The Efficient Memory System* maps the large on-chip 256K-byte RAM and 384K-byte ROM as unified program/data memory. Development is simplified since there is no fixed division between program and data memory size as on some other devices. *Universal Host-Port Interface (UHPI)* is a parallel interface through which an external host CPU can access memories on the DSP. Multichannel Audio Serial Ports (McASP0, McASP1, McASP2) - Up to 16 Stereo Channels I2S.

The flexibility of this line of processors has led to it being used not merely as a co-processor for digital signal processing but also as a main CPU. Newer implementations support standard IEEE JTAG control for boundary scan and/or in-circuit debugging.

3.5 Receiver and display Unit

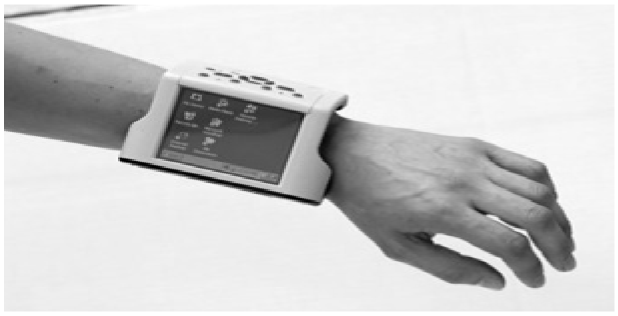


Fig 3.5.1 Novel TMS320 DSP based Multipara System

Using TMS320C6727 the display of the ECG curve, continuous BP curve, Sao2, oxygen level, PCG, body temperature, heart rate, respiration radiation effect of space can be mounted on the wrist of the individual astronauts & the ground control center as well.

4. Conclusion

The proposed novel integrated wireless single location sensor system is without the complexity of multi-wire and multi-locations. It can acquire more than 7 vital parameters simultaneously or single parameter of our choice. The signal processing and simultaneous mapping is done using TMS320C6727 DSP controller with a novel wrist watch type display.

5. References

[1] Bible: Leviticus: Chapter 17, Verse 11- Kings James Version

[2] Satava, R., Angood, P.B., Harnett, B.,Macedonia., and Merrell, R., “The Physiologic Cipher at Altitude: Telemedicine and Real-Time Monitoring of Climbers on Mount Everest, ” Telemedicine Journal and e-health,vol. 6 No. 3, 2000, pp. 303 – 313.

[3] Korhonen, I., Pärkkä, J., and van Gils, M., “Health Monitoring in the Home of the Future,” IEEE Engineering in Medicine and Biology Magazine, vol. 22, no. 3, May/June 2003, pp. 66 – 73.

[4] Gandsas A; Montgomery, K; Altrudi, R; and McKenas, D; "In-Flight continuous vital sign telemetry via the Internet," Journal of Aviation, Space, and Environmental Medicine, v71(1), January 2000.

[5] Park, S., and Jayaraman, S., "Enhancing the Quality of Life Through Wearable Technology," IEEE Engineering in Medicine and Biology Magazine, vol. 22, no. 3, May/June 2003, pp. 41 – 48.

[6] Moy, M. L., Mentzer, S. J., and Reilly, J. J., "Ambulatory Monitoring of Cumulative Free-Living Activity," IEEE Engineering in Medicine and Biology Magazine, vol. 22, no. 3, May/June 2003, pp. 89 – 95.

[7] Asada, H. H., Shaltis, P.,Reisner, A., Rhee, S, and Hutchinson, R. C., "Mobile Monitoring with Wearable Photoplethysmographic Biosensors," IEEE Engineering in Medicine and Biology Magazine, vol. 22, no. 3, May-June 2003, pp. 28 – 40.

[8] Jovanov, E., O'Donnell Lords, A., Raskovic, D., Cox,P.G., Adhami, R., and Andrasik, F., "Stress Monitoring Using a Distributed Wireless Intelligent Sensor System," IEEE Engineering in Medicine and Biology Magazine, vol. 22, no. 3, May-June 2003. pp. 49 – 55.

[9] Waterhouse, E., "New Horizons in Ambulatory Electroencephalography," IEEE Engineering in Medicine and Biology Magazine, vol. 22, no. 3, May/June 2003, pp. 74-80.

[10] Kramer,C.D., and Kalla, E. M., "The Challenge of Designing Biomedical Equipment During Human Research for Long Duration Low-Gravity Missions," Proceedings of the Sixteenth Southern Biomedical Engineering Conference, 4-6 April 1997, pp. 30 – 37.

[11] Yu M. Chi and Gert Cauwenberghs, "Non-contact EEG/ECG Electrodes for Body Sensor Networks." University of California, San Diego La Jolla, CA 92093

[12] A. Mahabuba, J. Vijay Ramnath and G. Anil Analysis of heart sounds and cardiac murmurs for detecting ardiacdisorders using phonocardiography Department of Electrical & Electronics Engineering, B.S.A. Crescent Engineering College, Chennai, Jl. of Instrum. Soc. of India Vol. 39 No. 1 March 2009

[13] Collin Schreiner, Philip Catherwood, John Anderson and James McLaughlin, Blood Oxygen Level Measurement with a chest-based Pulse Oximetry Prototype System NIBEC, University of Ulster, Newtownabbey, Northern Ireland Intelesens Ltd, Belfast, Northern Ireland

[14] E.S. Valchinov and N.E. Pallikarakis. An active electrode for bio potential recording from small localized biosources. Biomedical engineering online, 3, July 2004.

SESSION

COMPILERS + OS + SOFTWARE TOOLS AND ENVIRONMENTS, DEVELOPMENT ISSUES + LIBRARIES

Chair(s)

TBA

Bi-Endian Compiler: A Robust and High Performance Approach for Migrating Byte Order Sensitive Applications

M. Domeika[1]**, M. Loenko**[2]**, P. Ozhdikhin**[2]**, and E. Brevnov**[2]
[1]Intel Compiler and Languages, Intel Corporation, Hillsboro, Oregon, U.S.A.
[2]Intel Compiler and Languages, ZAO Intel A/O, Novosibirsk, Russia

Abstract - *In this paper, we describe the implementation and performance evaluation of a bi-endian capable compiler. Software migration of legacy applications from one endian architecture to another is oftentimes hampered by byte order dependencies hidden in the source code. These dependencies can be expensive to find and address in a systematic fashion.*

We introduce the Bi-endian Compiler (BEC) capable of generating code that executes with the opposite byte order semantics as the underlying architecture. The programmer designates the byte order required of data in memory. During translation, the compiler inserts additional instructions to transform, where necessary, data into the native byte order before it's operated upon in the processor registers. This paper details the language extensions, compiler design, and optimization of the implementation. We include a performance characterization comparing against native-endian SPEC 2000 and EEMBC benchmarks which show the performance overhead of enforcing opposite endian semantics to be in the range of 5.6% to 13.8%. The BEC has been successfully employed in production-quality software applications comprising millions of lines of source code.

Keywords: Byte order, big-endian, little-endian, endian-neutral, endian conversions, software migration

1 Introduction

Programming in standardized high level languages has the benefit of being readily portable across architectures. One would think that portability exists as long as programmers constrain themselves to features specified by the language standard and employ compilers that are dutiful in implementing to the standard. Unfortunately, this is not always the case. Migration of software between architectures can become problematic when byte order dependent code [7] exists in the source base only to be discovered when runtime problems surface. In large, legacy code bases consisting of millions of lines of code, it is very difficult to find and transform all of the byte order dependencies using known techniques [1] into endian-neutral code.

The Bi-Endian Compiler (BEC) enables applications to execute with the byte order semantics as which they were designed. For example, the BEC implementation discussed in this paper enables applications to execute with big-endian semantics on a little-endian processor. Employing BEC requires the programmer to designate the byte order of all data. During compilation, BEC inserts code sequences, where necessary, to load data into processor registers such that the data is in native endianness before operations are performed. Subsequently, code sequences are inserted that transform the results in native byte order into the resulting data's declared byte order before storing to memory.

This paper first provides background on the subject by reviewing byte order dependencies and current techniques to mitigate issues involving them. The BEC is then introduced, discussing the language features necessary to communicate byte order and the underlying compiler implementation. Feature enhancements made as a result of interaction with customers are then discussed; these features include enhanced diagnostics and special accomodations for pointers. Performance optimization and evaluation is then detailed showing techniques to improve the performance of the implementation. The conclusion includes discussion on future enhancements planned for the technology.

2 Background

Endianness, or byte order, is the format of how multi-byte data is stored in memory [3][9]. It specifies the location of the most significant and least significant bytes that comprise a multi-byte type such as a 32-bit integer. The two types of endian architectures are termed Big-Endian and Little-Endian. Discussions on the advantages and disadvantages of each has been characterized to being akin to a religious war [4][6]. Regardless, both big-endian and little-endian architectures exist and this can cause problems when migrating between architectures due to byte order dependent code. Figure 1 shows a code snippet and sample output which is different depending upon the byte order of the processor architecture. On a big-endian processor, where the most significant byte is stored in the lowest memory address, the pointer *ap* points to 12. On a little-endian processor, where the least significant byte is stored in the lowest memory byte address, the pointer *ap* points to 78. Legacy code bases built up over several years by many different programmers can be littered with such snippets of code motivated in many cases by optimization; assuming the location of a smaller subset of bytes in a multibyte element saves in terms of memory transactions.

```
#include <stdio.h>
int a = 0x12345678;
char *ap = (char *)&a;
printf("%2x %x\n", *ap, a);

Output on a big endian processor:
12 12345678
Output on a little endian processor:
78 12345678
```

Figure 1. Byte Order Dependent Code Example

Techniques of transforming byte order dependent code into endian-neutral code are well understood [1][7]. In the previous example, macros could be defined whose implementation would be platform dependent, but would agree upon which byte of a larger component is considered first, second, and so on. The techniques require the programmer to first identify byte order dependent code and make manual code changes to enforce endian-neutrality. In comparison, BEC does not require the programmer to find the specific byte order dependent code, but to only identify the byte order of the data. The compiler enforces that the correct byte order semantics are being executed. In the example from Figure 1, if the code was written to assume big-endian, the programmer only specifies that the variable *a* is big-endian and the compiler ensures that the expectation is met.

A second approach to migrating byte order dependent code is encapsulated by binary translation techniques [8]. These techniques encompass more in that they enable execution of one processor's instruction set architecture (ISA) on a processor with a different ISA by intelligently and efficiently translating between the two. This approach is attractive due to its relative ease of use for the customer; Apple employed its Rosetta technology to help migrate from the PowerPC architecture to Intel architecture. Compared to BEC, this approach typically incurs greater overhead as the application is translated during runtime without the benefit of aggressive static compiler optimization techniques.

3 Bi-endian Implementation

Byte order is a type attribute and can be bound to a builtin type, typedef or to a type as part of a variable declaration. The byte order attribute can be bound to pointer types, floating point types, and be part of a type chain consisting of multiple pointer indirections, integral, and floating point types. The language constructs and compiler implementation comprising the technology is numerous. High level components are detailed here and provide a basic understanding.

3.1 Language Features

The primary function of the language extensions is to enable the programmer to communicate the byte order of translation units, code sections, and individual declarations to the compiler. Figure 2 shows a code sample employing each. The source file, file.c, is compiled using the *–little-endian* option which specifies all data declarations in the translation unit are little-endian. This method is termed an implicit endianness declaration. The variable, *a*, would be stored in little-endian byte order. In the file, *#pragma byte_order (push, bigendian)*, specifies that declarations following the pragma are big endian. The variable, *b*, would be stored in big-endian byte order. In addition, the optional parameter *push* specifies that a stack of byte orders is maintained which enable byte order declarations spanning nested include files. This declaration method is also implicit and overrides the byte order specified at the command line. A section of code that has an implicit declaration bound to it is termed a big- or little-endian section (depending on the specified byte order). At the finest granularity an explicit declaration occurs via a byte order attribute. The variable, *c*, would be stored in big-endian byte order. The byte order attribute overrides both implicit methods.

```
icc –little-endian file.c
/* file.c */
int a = 0x12345678; /* little-endian */
#pragma byte_order (push, bigendian)
int b = 0x12345678; /* big-endian */
#pragma byte_order(pop)
int __attribute__((bigendian) c=0x12345678; /*big-
endian */
```

Figure 2. BEC Byte Order Declarations

3.2 Compilation Phases

Similar to many modern compilers, the BEC is multiphase, transforming one representation of the code to another beginning with the source code and concluding with the executable. Figure 3 illustrates the compilation phases.

The front-end phase parses the source code and transforms it into an abstract syntax tree (AST). During this phase, byte order attributes are associated with the program types represented in the AST and are dependent upon the byte order context at the point of declaration and as discussed in the previous section.

The BEC employs a proprietary Intermediate Language (IL) to represent the program under compilation. The IL translation phase converts the AST representation into this IL representation. Translated variables may have byte order conversion operations (BOCOs) placed both before and after the variable in cases when its type has a byte order opposite of the underlying target.

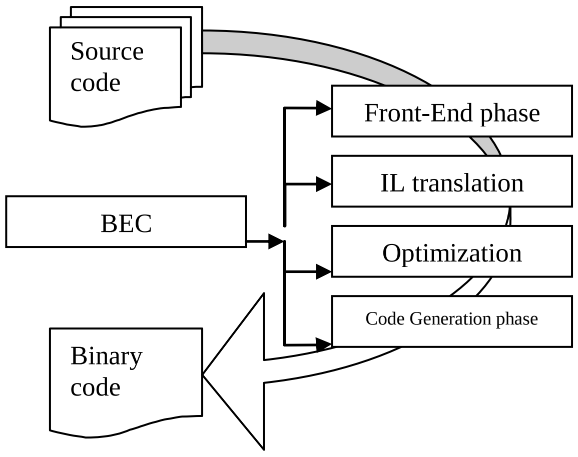


Figure 3. BEC Compilation Phases

The optimization phase operates on the IL representation to make execution of the code on the target platform more efficient. Since the BOCOs are represented in the IL just as any other instructions, standard compiler optimization can be applied. These optimizations and a description of each include:

- common subexpression elimination – removes redundant BOCOs for unused data
- code motion - moves BOCOs up to the function entry which reduces the number of BOCOs
- constant propagation - determines if a constant that requires a BOCO has already been loaded (and converted) which eliminates unnecessary BOCOs.

In addition, an optimization solely designed to remove redundant BOCOs is invoked. This optimization, termed the bswap elimination optimization, is described in a later section.

The code generator phase converts the IL representation into the binary code specific to the target platform. BOCOs are implemented using either hardware shift instructions or BSWAP instructions. Hardware BSWAP instructions provide an efficient method of converting between byte orders.

3.3 Data Initialization

In the BEC pointer data types can also be attributed with a byte order. As a result there may be situations where a pointer has the opposite byte order as the target platform. Pointer variables possessing opposite byte order types must be byte swapped upon initialization. This presents a challenge when pointers are initialized by link time constants because these constants are unknown at compile time and are resolved later at the linking stage.

In order to perform the necessary byte swap operations for pointers, the compiler generates and places special initialization data in a section of the object file, the *.initdata* section. This information is used in a three step data initialization process detailed as follows:

1. At the *static data initialization step*, a post link tool is employed which initializes data that can be initialized statically such as data that does not have relocations associated with it.

2. The *dynamic loader initialization step* is optional and requires a modified operating system loader.

3. The *dynamic runtime initialization step*, employs a run-time routine to initialize data stored in the opposite byte order as the underlying platform. This routine is automatically invoked prior to passing control to the main routine.

4 Advanced Features

4.1 Diagnostics for Code Endianness Issues

With the bi-endian technology it is possible to execute code that possesses mixed endian semantics; some of the data is little-endian and some is big-endian. In combination with the fact that C and C++ are not type safe, this introduces special considerations. For example, conversion between pointers to values of different sizes (for example between *char* * and *int* *), while safe in code that employs little-endian types, may result in an incorrect pointed–to value in code that employs big-endian types executing on a little-endian architecture. Standard compiler diagnostics are extended to account for pointer casts of different sizes.

The compiler also implements byte order specific warnings. For instance, when a function is declared implicitly the compiler would assume its parameters have the byte order that is implicitly declared at the function call. In the general case, the implementation of the function may assume a different byte order. The compiler emits a diagnostic in these cases to discourage such programming practice.

The compiler also emits a diagnostic when it detects adjacent bit fields of opposite byte order. Big-endian and little-endian bit fields are allocated differently in their containers. As a result, bit fields of different byte order would overlap if they were allowed in the same container.

The void type allows a byte order to be attributed to it. This alleviates potential issues casting through the void type where the original and final casted type is of opposite byte order.

4.2 Handling Parameters Passed by Pointer

When data of a primitive type is passed by value to a function that expects data of the opposite byte order, the BEC ensures correctness by automatically converting the byte order. However, if a pointer to big-endian data is passed to a function expecting a pointer to little-endian data, or vice versa, the compiler cannot automatically perform the conversion. For example, when an array is passed and the size of the array is not statically known, the compiler cannot determine how many data elements to swap.

There are cases where the compiler manages to modify the code to solve this problem and preserve the intended semantics. The most typical case is a big-endian local variable in the user code whose address is passed to a system function expecting a pointer to little-endian data. The BEC may convert the local variable to little-endian byte order if it can prove that such a change is safe. In a more complex scenario, a local variable may be used in both big- and little-endian contexts. In this case, the application of a copy-in/copy-out approach is employed if proven to be safe. Finally, if a global variable is passed as a parameter in big- and little-endian functions then endianess of the formal parameter of the method may be changed. This may require function specialization techniques.

Finally, if the compiler does not manage to automatically convert the specific mismatch it issues a diagnostic. The programmer would need to manually convert between byte orders for correct operation.

5 Performance Evaluation and Optimization

In addition to implementing traditional compiler optimizations, the BEC implements several byte order-specific optimizations to reduce the overhead of byte order swap operations added at the boundaries of big- and little-endian code.

5.1 Bswap Elimination Optimization

The Bswap elimination optimization is based on the concept of *swap-tolerant expressions*, which is defined as an expression that can be replaced with another expression operating on (some or all of) arguments of different byte order and producing a valid result of the same or a different byte order.

For example, a comparison to a constant (e.g. x == 0x12345678) is swap-tolerant since it has a counterpart (y == 0x78563412) which if given a swapped argument SWAP(x) would produce the same result as the original expression.

A bit-wise AND is swap-tolerant since there is an operation (the same AND) such that taking the swapped arguments it would produce the correct but of different byte order result.

Arithmetic (+, -, *, /) operations are not swap-tolerant since they strictly require data of specific byte order to produce correct results.

Domain is defined as a set of expressions of the code under compilation.

Domain entry is an expression outside the domain, result of which is taken as an argument by an expression belonging to the domain.

Domain exit is an expression outside the domain that takes a result of an expression belonging to the domain as an argument.

A *Swap-tolerant domain* is defined as a set of swap-tolerant expressions that can be replaced by their counterpart expressions such that if some or all of the domain arguments are replaced with data of different byte order then all the domain results would be valid results of the same or a different byte order.

Swap of the domain is a code transformation involving the following two actions:

1. Changes byte order of some or all of the domain entries and exits by placing or removing byte swap operations at necessary domain entries and exits.

2. Substituting all expressions in the domain with their counterparts operating on different byte order, so that code semantics are preserved. Byte swap operations are removed if the entry or exit expression is a byte swap.

Example:

```
T1 = SWAP(A)

T2 = SWAP(B)

RES = T1 == T2
```

The expression “T1 == T2” comprises a swap-tolerant domain, expressions “SWAP(A)” and “SWAP(B)” are domain entries, assignment “RES = ...” is a domain exit. A domain swap would be:

```
T1 = A   //byte swap is removed

T2 = B   //byte swap is removed

RES = T1 == T2  //byte order of result is the same
```

Domain swap benefit is an estimate of the performance benefit from the swap and is computed by factoring in the amount of code removed minus the amount of code added (taking into account execution cycle counts of specific instructions placed and removed).

To build a swap-tolerant domain one should start with any swap-tolerant expression or from a byte swap desired to be removed and extend the domain with connected swap-tolerant expressions. If further domain extension is either impossible or performance negative, convert the current domain and move to the next expression.

5.2 Data Byte Order Choice

The BEC discussed in this paper is primarily used to compile legacy code to execute on modern little-endian architectures. One typical usage model is where the programmer marks all of the legacy code as big-endian rather than determining and employing explicit byte order declarations on the minimal set of data. In this use case, the compiler adds BOCOs after loads and before stores of the data even when its byte order is not really sensitive. A second usage model is in a mixed endian environment where the user application requires big endian semantics and the underlying operating system is a commercial offering for the target requiring little endian semantics. This usage model also has a fair number of additional, but semantically unnecessary BOCOs.

Another optimization that improves performance is related to the choice of the actual data byte order. At times, the compiler can prove that the programmer specified byte order has no impact on the program semantics. In cases where data with opposite byte order would perform better, the compiler will convert the data to employ that byte order.

The byte order of the data (variables, data structures, heap data, function arguments, etc) is not visible to programmer if all of the stores and loads of the data are of the same size.

Figure 4 is a flow chart that represents the algorithm used to prove that all stores and loads of the data are of the same size. For example, byte order of a top level static variable, address of which is never taken is not visible to the programmer.

Data byte order visibility is calculated conservatively, i.e. treat byte order as visible if not proved otherwise. To define byte order visibility of function arguments the compiler additionally ensures that all the calls of the function are known (including indirect calls).

An implementation works on two compilation passes. On the first pass it accumulates information about data usage and also calculates byte order preference from the performance perspective. At the second pass, the data usage is adjusted according to the selected byte order for each specific piece of data. To reduce compile time the BEC may heuristically break analysis and make conservative decisions.

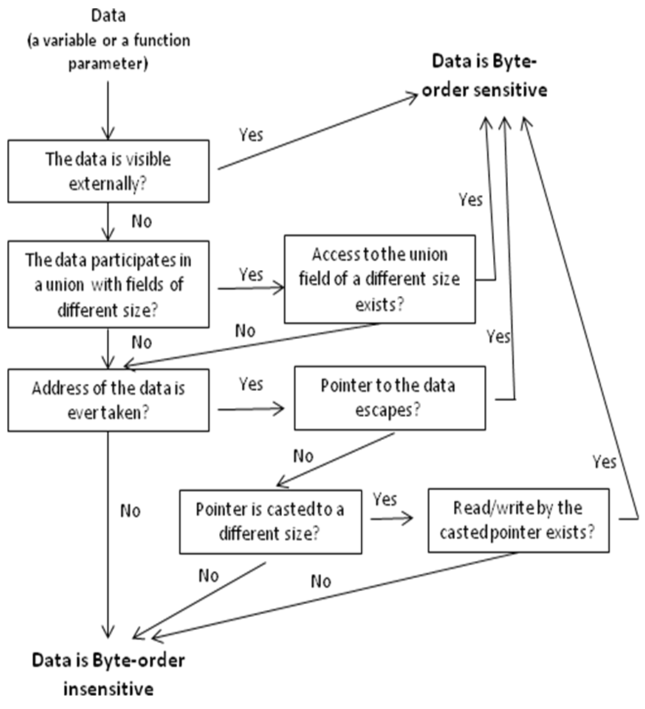

Data
(a variable or a function parameter)
Data is Byte-order sensitive
The data is visible externally?
Yes
No
The data participates in a union with fields of different size?
Yes
Access to the union field of a different size exists?
Yes
No
No
Yes
Address of the data is ever taken?
Yes
Pointer to the data escapes?
No
Yes
No
Pointer is casted to a different size?
Yes
Read/write by the casted pointer exists?
No
No
Data is Byte-order insensitive

Figure 4. Determining Byte Order Sensitivity

5.3 Performance

To estimate the penalty for the BOCOs a modified version of little-endian benchmarks was used: the benchmarks were adapted and compiled to execute with big-endian semantics. The execution times were compared with the original benchmarks compiled as a regular little-endian code.

Table 1 shows the performance impact of the benchmarks compiled as big-endian (system libraries compiled as little-endian) compared to the same benchmarks compiled entirely as little-endian. The benchmarks[i] comprise the C benchmarks in SPEC2000 and EEMBC 1.1. The benchmarks were executed on systems based upon the Intel® Atom^{TM} processor and the Intel® Core^{TM} i7 processor under both 32 and 64-bit modes as detailed in the table. The Intel® Atom^{TM} processor-based system executed at 1.66 GHz, included 2GB RAM, and ran Ubuntu Linux 10.04. The Intel® Core^{TM} i7 processor-based system executed at 3.33 GHz, included 6GB RAM, and ran Red Hat Enterprise Linux5.U2 x86. The Intel® C++ Compiler Standard Edition for Linux* OS with Bi-Endian Technology, version 11.2, was employed in all cases.

Table 1. Penalty for the Byte Swaps

CPU	**Mode**	**Overhead compared to little-endian**	
		Benchmarks	**Geomean**
Intel® Atom™ Processor	32	1-13%	5.6%
Intel® Core™ i7 Processor	32	1-14%	8.5%
Intel® Core™ i7 Processor	64	5-30%	13.8%

There are two reasons for the overhead. The first is that byte swap operations take time. The overhead depends on the efficiency of byte swap elimination optimizations. Performance impact of the bswap operations also depends on the efficiency of byte swaps on underlying computer architecture. For that reason the gap on the Intel® Atom™ architecture having the MOVBE instruction is lower.

The second reason is that the presence of the byte swaps might be an obstacle for optimizations to generate good code. For example, byte swaps in a loop may prevent it from being vectorized. Byte swap eliminations applied at an early compilation phase usually helps to minimize this issue.

The overall impact of byte swap elimination on geomean score for the selected benchmarks is about 10-15%. There are tests (e.g. in the EEMBC suite) which improve their performance in times when these optimizations are applied.

6 Conclusions

The BEC enables migration of large legacy code bases containing byte order dependencies. The BEC does not find the specific byte order depedencies, but instead enables the application to execute with the same byte order semantics as which it was produced. This paper detailed the design of the bi-endian technology, extensions to aid in its use, and performance optimizations. A performance assessment shows the overhead of employing the BEC as ranging from 5.6% to 13.8% on a range of benchmark programs. Future research directions include enabling full use of C++ features such as operator overloading with bi-endian types.

7 Acknowledgements

Our thanks go to Hugh Wilkinson, Matt Adiletta, Bob Kushlis and Jack Johnson for early work on the Bi-Endian Compiler. We thank Azita Refah, Peter Horn, Michael Rice, Kittur Ganesh, Xinan Tang, Damion Desai, Greg Anderson, Joe Wolf, and Kevin J. Smith for their contributions to this project spanning multiple years.

8 References

[1] Adiga, H. Writing Endian-Independent Code in C. Retrieved April 24, 2007 from, http://www.ibm.com/developerworks/aix/library/au-endianc/index.html?ca=drs-.

[2] Adiletta, M., Wilkinson, H., and Kushlis R., "Method and apparatus for implementing a bi-endian capable compiler," U.S. Patent 7 552 427, June 22, 2006.

[3] Blanc, B. and Marraoui, B. Endianness or Where is Byte 0. Retrieved Dec. 21, 2008 from, http://3bc.bertrand-blanc.com/endianness05.pdf.

[4] Cohen, D. 1981. On Holy Wars and a Plea for Peace. *IEEE Computer* 14, 10 (Oct. 1981), 48-54. DOI= http://dx.doi.org/10.1109/C-M.1981.220208.

[5] Domeika, M. 2008. *Software Development for Embedded Multi-core Systems.* Elsevier Inc., Burlington, MA, 93-99.

[6] James, D. 1990. Multiplexed Buses: the Endian Wars Continue. *IEEE Micro* 10, 3 (Jun. 1990), 9-21. DOI= http://dx.doi.org/10.1109/40.56322.

[7] Matassa, L., Endianness Whitepaper, Retrieved May 13, 2008 from, Intel Software Network: http://software.intel.com/en-us/articles/endianness-whitepaper/.

[8] Souloglou, J., Rawsthorne, A., "Program code conversion," U.S. Patent 7 421 686, Sept. 2, 2008.

[9] Understanding Big and Little Endian Byte Order. Retrieved Sept. 19, 2006 from, Better Explained: http://betterexplained.com/articles/understanding-big-and-little-endian-byte-order/.

[i] The benchmark results are not official results because the source code was modified to execute with big endian semantics.

Refer to:
http://software.intel.com/en-us/articles/optimization-notice
for more information regarding performance and optimization choices in Intel software products.

Model transformation and scheduling analysis of an AUTOSAR system

Ahmed Daghsen, Khaled Chaaban, Sébastien Saudrais
ESTACA campus ouest
Embedded systems laboratory
Laval, 53000, France
ahmed.daghsen@estaca.fr

Mohamed Shawky
Université de Technologie de Compiègne
Heudiasyc laboratory
Compiègne, 60200, France
mohamed.shawky@hds.utc.fr

Abstract— **AUTOSAR standard provides a common framework for software development in the automotive domain. It enables to manage the growing of the automotive architecture complexity by facilitating the integration and reuse of software components. However, additional work is needed to enable scheduling analysis and to handle with more timing properties in the system. In this paper, we propose an approach to enable a model transformation of the AUTOSAR timing model to a classical scheduling one. This allows to apply directly fundamentals scheduling theories for timing analysis. Then, we apply our approach through a steering-by-wire case study. Finally, we analyze the results given by the holistic algorithm and those given by a compositional one.**

I. Introduction

Many automotive applications are considered as time-critical or at least time-dependent. Thus, precise timing and prioritization of functions are essential for both safety and comfort of in-vehicle applications.The AUTomotive Open System ARchitecture (AUTOSAR) standard [1] is introduced to define a standard layered software architecture and interfaces. Many improvements and extensions to the current AUTOSAR system model have been developed recently to handle all timing-related information during the development process. Thus, complexity and development cost cycle are reduced significantly while reliability is improved. AUTOSAR allows an easy integration of timing information, however, few works use these timing properties and constraints to make a global timing analysis of the system. The local timing analysis addresses tasks scheduling regarding an Electronic Control Unit (ECU), and global scheduling considers the global distributed system where communication bus and gateways must be analyzed together with ECUs tasks. We can distinguish between local and global timing analysis. Where local timing analysis addresses tasks scheduling regarding a processor or an ECU, global scheduling considers the global distributed system where communication bus and gateways must be analyzed together with ECUs tasks. There are several problems related to such distributed system that must be addressed, such as task synchronizations and communication dependencies between processes. In this paper we propose an approach to enable a transformation of AUTOSAR timing properties and constraints into a complete scheduling model. By using this model, we can apply directly existing scheduling theories to the AUTOSAR application. As shown in figure 1 the approach consists of two main steps. The first step consists in applying a model transformation to convert AUTOSAR timing model to a scheduling model. The second step permits to apply the scheduling techniques for a global timing analysis of the AUTOSAR system. We show that this analysis allows to take into account further timing properties like task synchronizations, process communications modeled as offsets, jitter, constrained deadlines, process preemption and blocking overheads. The proposed method is applied to a steering-by-wire case study and we analyze scheduling results given by both a holistic and compositional scheduling approaches.

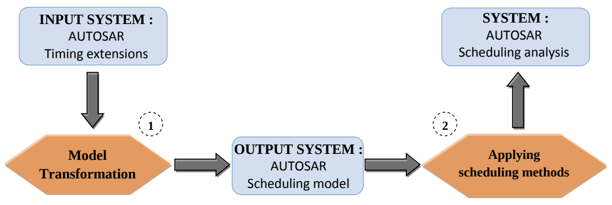

INPUT SYSTEM : AUTOSAR Timing extensions
1
Model Transformation
OUTPUT SYSTEM : AUTOSAR Scheduling model
2
Applying scheduling methods
SYSTEM : AUTOSAR Scheduling analysis

Fig. 1. AUTOSAR model transformation

II. AUTOSAR Methodology

In this section, we present a brief description of AUTOSAR methodology. Figure 2 describes the development process structure of an AUTOSAR software.

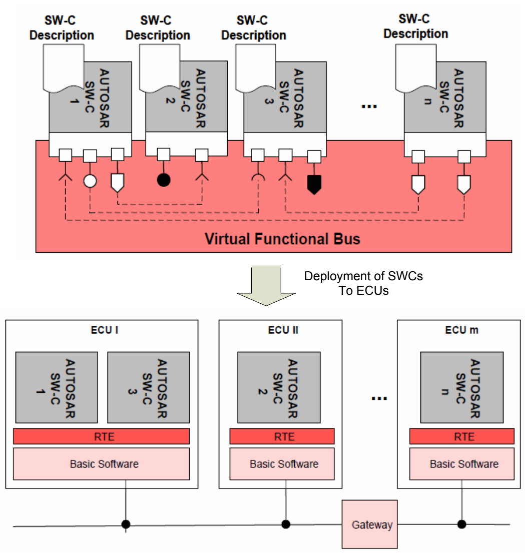


Fig. 2. AUTOSAR methodology

The first step consists on the definition of software components (SWCs) constituting the user software applications. SWCs communicate using ports through their interfaces. A SWC may be one of the three types: sensor/actuator, application or calibration type. Each SWC contains runnable entities which represents the C code that will be executed on the ECU. A runnable is triggered using an event which may be of timing or data type. In second step, at the Virtual Functional Bus (VFB) level, SWCs are defined without consideration of the underlying hardware on which these SWCs will run on later. So, two software components might run on the same ECU or on different ECUs and this is completely transparent to software developers. The communication between the components is then either an intra-ECU communication or an inter-ECU communication and is routed via the VFB bus which allows a virtual integration of the system independently of underlying software and hardware. Next, the mapping of the SWCs to available ECUs is performed. This phase requires some information about system and ECU constraints such as the input/output hardware connection. Finally, we can proceed by the development and integration of each ECU. The software architecture of an ECU is composed of three main layers: the SWCs, the Run Time Environment (RTE) and then the Basic Software (BSW) layer. The SWCs contain the application's functional code. RTE represents an instance of the VFB bus per ECU. It provides standardized interfaces to communicate with the BSW layer and to communicate between SWCs themselves. Data exchange between SWCs themselves and between SWCs and the underlying BSW layer is performed exclusively via RTE. Depending on SWCs locations, data exchange is performed either directly via a shared memory or by sending messages via a network bus. BSW layer makes the link between RTE layer and all hardware features of the ECU.

III. System Scheduling Analysis

In this section, we present related works dealing with timing analysis in AUTOSAR system and the existing formal approaches for system level performance analysis.

a) Scheduling in AUTOSAR: few works are dealing with the exploitation of AUTOSAR timing extensions for timing analysis. In the scope of TIMMO project, [6] gives a general framework for the relation between AUTOSAR concepts and timing constraints. They have proposed also an extension of AUTOSAR standard towards the possibility to specify the system's timing constraints. Thus, a scheduling analysis of an AUTOSAR application can be performed at the low-level. But the resulting task timing reveals hardly any direct timing-relation with high-level software components to which timing information shall finally be attached.

b) Compositional & Holistic scheduling: compositional performance analysis enables performance analysis for complex heterogeneous embedded architectures and supports subsystem integration [4]. This approach consists of integrating either offline or online scheduling analysis techniques into a system-level analysis. On the other hand, holistic analysis [8] introduced by Tindel, refers to a consistent end-to-end response time analysis approach for multi-processor real-time systems, where processors communicate over a bus and offline scheduling methods could be applied for timing analysis. The holistic technique also captures the timing using system-level equations. However, flexibility, subsystem integration and scalability are major weaknesses of holistic techniques. Recent works of Turja and Nolin [5] presented a method for calculating tighter (i.e. lower) response-times. Their method, under certain conditions, calculates the exact worst-case response time with offset. In practice the holistic approach is used in system configurations having low dependency complexity such as deterministic TDMA network. [2]

IV. Transformation of the AUTOSAR model

Our approach aims to perform a transformation of AUTOSAR timing properties and constraints into a complete scheduling model. By using this model, we can apply existing scheduling theories to the AUTOSAR application.

A. *AUTOSAR input model*

We consider an AUTOSAR system model as an input of the transformation process. The main timing related concepts of this model (AUTOSAR release 4.0) are:

- SWC: encapsulates a part of the functionality of the application.
- Event timing chain: Temporal correlation between two observable events.
- Period: Time interval between two consecutive event occurrences.
- Jitter: The maximum variation of timing event period
- Latency: The time duration between the occurrence of the stimulus and the occurrence of the response.
- Runnable: Is a part of an atomic software component which can be executed and scheduled independently.

In this model, a task is called an end-to-end chain which consists of a set of subchains and has an end-to-end deadline. Each subchain is assigned a proper priority and its worst-case response time can be bounded.

B. *Model transformation*

Table I illustrates some relationship between scheduling system model and the AUTOSAR timing concepts one. The output model consists of an end-to-end system model which is used as the basis of this work.

AUTOSAR model	**System model**
Subchain	Subtask
ECU	Processor
Communication bus	Link processor
Latency	Release time
Runnable	Subtask

TABLE I
Relationship between scheduling system model and AUTOSAR 4.0 concepts

In real-time system, an end-to-end system consists of more than one processor and a set of end-to-end tasks. In our model, the workload on a multiprocessor P_i system consists of a set T_i of end-bend tasks, each of which is a periodic task with period p_i, phase f_i, execution time t_i, and relative deadline D_i. In this paper, we assume that the relative deadline of a task is less than or equal to its period, i.e., $D_i \leq p_i$. The release time of the first instance of $T_{i,1}$ is the phase f_i of task T_i. An instance of $T_{i,j}$ cannot start to execute before the complete execution of $T_{i,j-1}$. A task T_i is a chain of subtasks $T_{i,j}$. Each subtask $T_{i,j}$ is one continuous execution thread of T_i on one processor and has a maximum execution time $Y_{i,j}$ and a fixed priority $prio_{i,j}$. Subtasks are statically assigned to processors. Subtask $T_{i,j}$ is a predecessor (successor) of subtask $T_{i,k}$ if $j < k$ $(j > k)$, and $T_{i,j}$ is the immediate predecessor (successor) of $T_{i,k}$ if they are also adjacent ($| j-k |= 1$). The system model imposes strong restrictions on tasks properties. We consider both preemptive and non-preemptive tasks and subtasks. We also assume a common time base for all processors and we consider the jitter and the offset of periodic tasks.

V. Case study

The main objective of this section is to apply both the holistic and compositional analysis to a steering-by-wire case study. We begin by presenting the steering-by-wire system, which is developed in our laboratory. Then, we apply our approach using a holistic scheduling algorithm: Per Task Time Demand function (PTTDF). Finally, we simulate the system using a compositional scheduling tool and we analyse the results of each scheduling approach.

A. *Steering-by-wire system*

As depicted in Figure 3, a basic steering-by-wire system is composed of three main blocks: the hand wheel (i.e. steering), controllers and the road wheels. When the driver operates the hand wheel to turn the vehicle, a steering angle signal will be sent to the controller. Two kinds of sensors are necessary to acquire the steer angle and the torque applied by the driver. The controllers will process all acquiring signals and also perform

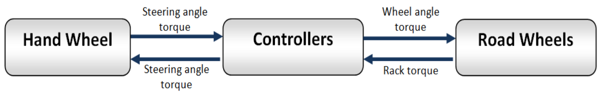


Fig. 3. Steering-by-wire system

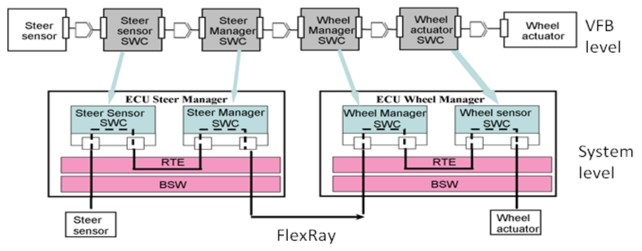


Fig. 4. Implementation of the rack torque function in AUTOSAR

some control functions associated with the vehicle's steering function and output an actuator angle for the road wheels that in turn will turn the wheels through an actuator. The feedback signals (actuator feedback and wheel feedback) involve some kind of force or torque sensors and are necessary so that the driver get the feeling of turning a traditional steering wheel and feel the effect of turning the wheels on a certain type of road. The steering-by-wire system may be composed of two main functions: the feedback torque function and the rack torque function. The feedback torque function permits to compute the feedback force applied to the steering wheel, so that the driver feels the effect of tuning the wheels on a certain type of road. The rack torque function is the main system function that permits to control the front axle actuator. The distributed steering-by-wire architecture involves several components: ECUs, Flexray bus for the communication lines and appropriate sensors and actuators. Flexray technology provides higher bandwidth, fast communication, fault tolerant and deterministic latency, enabling the development of innovative automotive system. Figure 4 illustrates the implementation of the rack torque function according to AUTOSAR approach. At the VFB level, the signal path involves four components. The "Steer Sensor" component acquires the sensor physical data and passes it to the application software component "Steer Manager". Afterwards the signal is sent to the application software component "Wheel Manager" for order computation until it is finally send to the actuator via the "Wheel Actuator" component. At the system level, we map SWCs to available ECUs and then we configure RTE and BSW modules. In our case study we have only two ECUs: the steer ECU and the wheel ECU.

B. Applying scheduling algorithm to the AUTOSAR transformed model

We consider the system model obtained after the transformation process. Note that in AUTOSAR, and end-to-end task passes by three stages: from hardware to software represented by the transformation of data from the physical sensor to the sensor SWC (e.g. steer sensor or wheel sensor SWC), the second stage is all the actions that pass between the sensor SWC till the software control represented by the actuator SWC. The last stage is the interface that is done between the actuator SWC and the physical actuator (as shown in Figure 5). Each function, feedback torque function and rack torque function is represented as an end-to-end task. Then, the steering-by-wire system is represented by two end-to-end tasks. Let's note T_1 as the rack torque end-to-end task and T_2 as the feedback torque end-to-end task:
T_1: has 17 subtasks: $T_{1,1}$, $T_{1,2}$, $T_{1,3}$, $T_{1,17}$.
T_2: has 17 subtasks: $T_{2,1}$, $T_{2,2}$, $T_{2,3}$, $T_{2,17}$.
Each end-to-end timing chain segment in AUTOSAR model corresponds to a subtask. Subtasks of each function are executed on a specific ECU. As noted above, we have two processors: P1 at the steer side and P2 at the wheel side. The communication bus represents a link processor P3.

We remind that each process can be specified as a constrained deadline periodic task T=(O; J; p; t; D), where O is offset, J is jitter, p is period, t is the execution time, and $D(\leq T)$ is deadline. Priorities are fixed with respect of precedence constraint.

After having established the AUTOSAR scheduling model, we can now apply the PTTDF holistic algorithm. This algorithm presented in [7] allows to compute the tighter upper bounds of the response times of the end-to-end tasks in static systems. After computing the different equations of the algorithm, we obtain:

- The upper bound of task T_1: $C_1 = 133 > 50$ then T_1 is not schedulable.
- The upper bound of task T_2: $C_2 = 203.91 >> 50$ then T_2 is not schedulable.

Finally, according to the proposed algorithm and the obtained results the system is not schedulable.

C. Compositional analysis

In order to make a compositional analysis of the steering by wire system we use the SymTA/S tool [3]. It is a formal system-level performance and timing analysis tool of distributed systems. The approach is based on the compositional analysis. It permits to couple local scheduling analysis algorithms using event streams. Event streams describe the possible input/output timing of tasks.

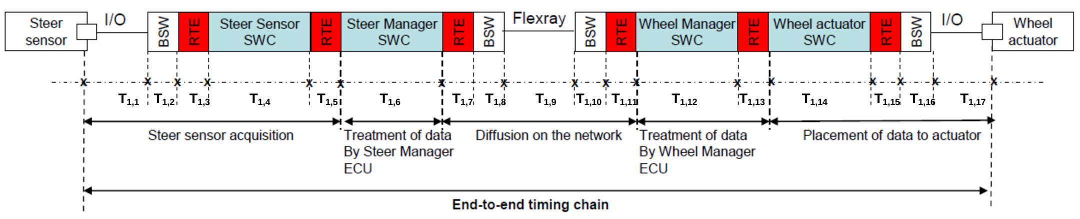


Fig. 5. End-to-end timing representation of AUTOSAR methodology

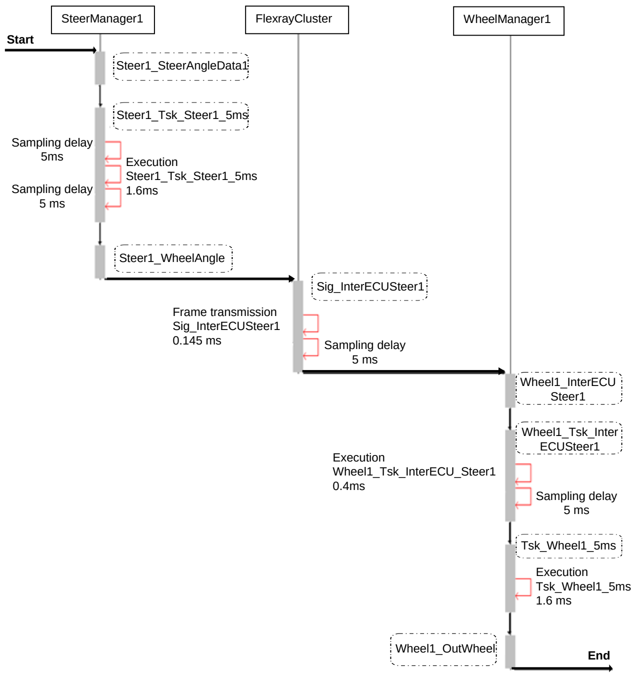


Fig. 6. End to end path of the rack torque function

By using SymTA/S analysis tool, we can calculate the local best-case and worst-case response times for tasks and frames, end-to-end best-case and worst-case response times for critical paths, deadline violations, utilization of resources, load contribution of individual tasks and individual frames.

Figure 6 illustrates the path of the rack torque function.

SymTA/S was used to model the steering-by-wire system with the timing properties like the task execution time, task periodicity, task execution type (preemptive, non-preemptive etc) etc. These timing models could define the system on design level and implementation level. The feasibility of the system was found out by using timing analysis feature available in the tool. Figure 7 shows the response time of the end-to-end path taking into consideration the synchronization between calculators, the offset and the jitter constraints.

D. Results and discussion

It has been shown that the holistic approach leads to pessimistic bounds. Since the used algorithm does not take into account the precedence constraints of subtasks when computing PTTD function which is the sum of all subtasks' execution time. This function assumes that the subtasks of each end-to-end task are independent, but the actual time demand may be less than the sum because of the precedence constraints among subtasks. Then the PTTD function can be considered as the maximum of the sum of all subtasks. Moreover, by considering the jitter and offset values, the obtained bound are higher. The holistic approach is more accurate by applying analytic equations. This technique is more suitable for system configurations with simplified equations such as deterministic Flexray and TDMA networks. The compositional analysis is a good candidate for more complicated heterogeneous system. The compositional model are well structured and uses event stream representation to allow component wise local analysis and also facilitates the subsystem integration.

VI. Conclusion

Performance analysis and timing requirements in AUTOSAR have received a wide attention recently. The main goal is to perform an early verification and analysis of the system performance at the design level and before implementation. There are few formal approaches to heterogeneous systems. In this paper, we have proposed an approach to transform an AUTOSAR timing model to a scheduling model. By this transformation, we can apply fundamentals scheduling techniques to the AUTOSAR system. Although, the proposed transformation is sufficiently generic to be integrated to other AUTOSAR architecture and communication paradigms like CAN or LIN, our approach makes several simplifications on the system model. However, a more accurate system model must be considered in order to take into consideration the architecture complexity, tasks interdependency and the low level layer interaction. Also, as a future work, we plan

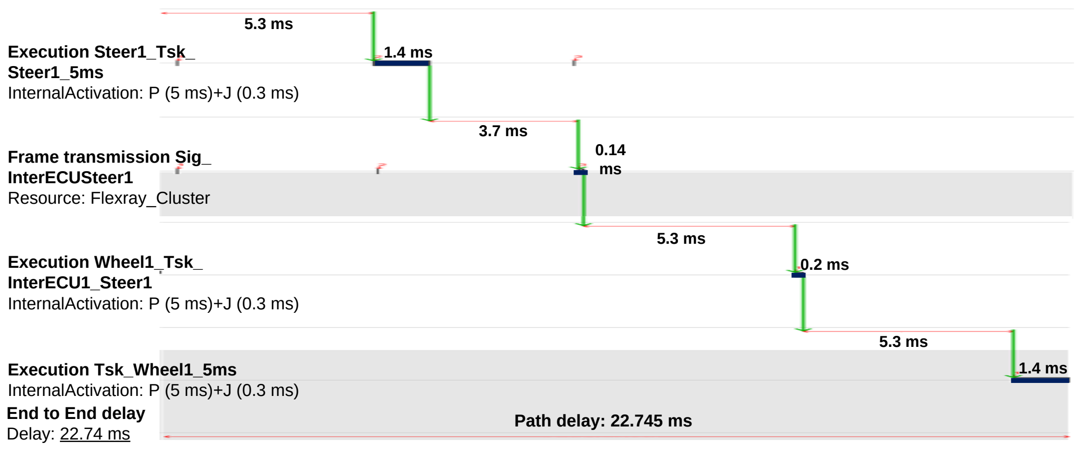


Fig. 7. Process with synchronization, offset = 0.2 and jitter = 0.3

to extend our system model by taking into consideration CPU resource sharing such as shared memory, multi-cores architecture and hardware constraints such as pipelining and caching.

References

[1] Simon Fürst et al. Autosar - a worldwide standard is on the road. In *14th International VDI Congress Electronic Systems for Vehicles, Baden-Baden*, 2009.

[2] Rajesh K. Gupta and Giovanni De Micheli. *System-level Synthesis using Re-programmable Components*. PhD thesis, 1992.

[3] Rafik Henia, Arne Hamann, Marek Jersak, Razvan Racu, Kai Richter, and Rolf Ernst. System level performance analysis - the symta/s approach. 2005.

[4] Marek Jersak. *Compositional Performance Analysis for Complex Embedded Applications*. PhD thesis, Technical University of Braunschweig, 2004.

[5] Jukka Mki-turja and Mikael Nolin. Tighter response-times for tasks with offsets. 2004.

[6] M. Rudorfer O. Scheickl. Automotive real time development using a timing-augmented autosar specification. *BMW, Munich*.

[7] Jun Sun, Riccardo Bettati, and Jane W. s. Liu. An end-to-end approach to schedule tasks with shared resources in multiprocessor systems. 1994.

[8] Ken Tindell and John Clark. Holistic schedulability analysis for distributed hard real-time systems. *Microprocess. Microprogram.*, 40(2-3):117–134, 1994.

Army Vehicle Software Complexity Prediction Metric-Five Factors

M. S. Dattathreya[1], and H. Singh[2]
[1]Tank Automotive Research, Development and Engineering Center, Warren, MI, USA
[2]Electrical and Computer Engineering Department, Wayne State University, Detroit, MI, USA

Abstract - *Army vehicle software interacts with complex electronics from multiple vendors. The software structure complexity is influenced by many factors prior to software development. Understanding, predicting and resolving complexity of vehicle software prior to its development is a necessity for Army mission success. Current complexity metrics focus on software and its technical structure with no consideration of its influencing factors. Non-technical metrics related to software complexity are required to address diverse skill set including the management. In this paper, the authors propose five non-technical factor metrics based on the current software development process to predict future Army vehicle software complexity. Fuzzy logic techniques are used for developing, modeling and analyzing the software complexity prediction metric.*

Keywords: Metrics, Process Metrics, Reliability, and maintainability.

1 Introduction[3]

ARMY has to perform several critical complex functions to defeat enemy forces. To carry out a mission, the Army uses vehicles including armored fighting, medical, and mortar carrying vehicles. These vehicles have multiple electronic devices and computing resources. The vehicle on-board devices are from multiple vendors and have unique software interfaces. These devices interact with each other using a complex network. The Army battle success depends on effective interoperable communications between these devices and its software.

The Army vehicles' devices are real-time and are controlled by embedded operating systems. The Army vehicles have software related to radios, navigation, diagnostics, etc. For the Army, the vehicle software is crucial and any complexity in it hinders the mission success. The Army software is developed under intense tight requirements using multiple vendors. The software has millions of lines of code. They are developed using many different programming languages and require efficient integration to minimize possible software complexity. The Army has to focus more on its vehicle software complexity than any other commercial software, because, the Army environment is dynamic and its requirements are changing frequently to meet mission needs. The complex software in an Army vehicle introduces many defects and makes it difficult to understand the software and correct the defects in a relatively faster pace. If the software cannot accommodate frequent changes in a faster way, the vehicles cannot perform its intended function and it is not acceptable to the Army. There is a bigger need for identifying metrics to predict the Army vehicle software complexity in very early stages of its development cycle. Many researchers have postmortem historical software and identified the reasons why a given software structure is complex, but, to our knowledge, no body has made an attempt to identify the Army vehicle software complexity contributing factors by inspecting the current software development process.

We define true software complexity as a combination of three main elements i.e. reliability, availability, and maintainability (RAM). Reliability (R) is the probability of performing a required function under stated conditions for a specified period of time [1]. Availability (A) is a measure of the degree to which software is in an operable state and can be committed at the start of a mission when called for at an unknown (random) point in time. Availability as measured by the user is a function of how often failures occur and corrective maintenance is required, how often preventative maintenance is performed, how quickly indicated failures can be isolated and repaired, and how quickly preventive maintenance tasks can be performed [1]. Maintainability (M) is the ability of software to be retained in, or restored to, a specified condition when maintenance is performed by personnel having specified skill levels, using prescribed procedures and resources, at each prescribed level of maintenance and repair [1]. If the software is complex, its reliability is hard to achieve, due to this, the software may suffer availability issues. If the software is hard to understand,

[3] This document is *UNCLASSIFIED: Dist A. Approved for public release.*

hard to fix defects, then the maintainability suffers. The bottom-line is, the RAM controls the entire software dependability, and without it the Army cannot perform its intended functions. These arguments confirm that the software complexity is really consisting of RAM elements.

The complexity is influenced by many factors prior to software development. Understanding, predicting and resolving complexity of vehicle software prior to its development is a necessity for Army mission success. In this paper we describe the proposed software complexity prediction metric for Army vehicle software. We describe the proposed software complexity prediction metric in the subsequent sections.

2 Related Work

The McCabe's [2] software complexity introduces the concept of Cyclomatic Complexity, where the number of flow graph edges, nodes and predicate nodes are combined to represent the complexity. The Cyclomatic Complexity of a source code is the linearly independent paths count through the source code.

The Halstead [3] software complexity measures the complexity by counting number of operators and operands in software. It measures the software's ability to understand and estimates the effort required to develop a software algorithm. It also indicates the amount of time to implement an algorithm. Halstead metrics are difficult to calculate and it is very hard to count the distinct and total operators and operands in a software program.

The Henry and Kafura [4] provide the measure of couplings between modules in terms of number of parameters, global variables and function calls. It measures given software's procedure, module and interfaces.

The Entropy software complexity measure [5] is based on the average information content of each operator in a software program's source code.

The Cognitive weights [6] from Jingqiu Shao and Yingxu Wang models the software complexity based on the cognitive functional size of the software.

The Relative complexity metrics [7] represents a single, unified measure on the structure of a software program. It serves to classify a set of software programs in order of their increasing complexity in relation to each other.

The [2], [3], [4], [5], [6], and [7] metrics are too technical and focus only on technical structure of a software program. These data are hard to compute and requires many skilled resources to understand and implement solutions.

We need metrics captured from the current software development process documents rather than the software itself. These metrics must be easily understood by both the technical and non- technical resources.

3 The Software Complexity Algorithm

We propose the following to predict software complexity.

1) Technical readiness level (TRL)
2) Number of open requirements (OR)
3) Number of planned technical reviews (TR)
4) Number of planned documentation tasks (DOC)
5) Number of planned configuration management tasks (CM)

Subsequent paragraphs describe the proposed metric elements and its association with R, A, and M. For predicting the Army vehicle software complexity, all the five factors must be considered because the combination of factors predicts R, A, and M component of the software complexity as shown below.

1) TRL, TR, and OR factors predict reliability.
2) TRL and TR factors predict availability.
3) DOC and CM factors to predict maintainability

The data can be captured from the software development project plan, development strategy, test strategy, technology strategy, and requirements analysis documents.

3.1 Technology readiness level (TRL)

The TRL measures evolving technologies maturity prior to its implementation. The readiness is indicated by 1 to 9 levels.

TRL1) Basic principles observed and reported
TRL2) Technology concept and/or application formulated
TRL3) Analytical and experimental critical function and/or characteristic proof of concept
TRL4) Breadboard validation in laboratory environment
TRL5) Breadboard validation in relevant environment
TRL6) Model or prototype demonstration in a relevant environment
TRL7) Prototype demonstration in an operational environment
TRL8) Actual system completed and 'flight qualified' through test and demonstration
TRL9) Actual system 'flight proven'

A technology with a lower TRL contributes to frequent failures when the software is developed. This creates lower mean time between failures (MTBF), increased downtime, lower meantime between repairs (MTBR), etc. When the TRL level is more than six it is considered mature enough to provide good reliable software. Higher MTBF indicates more reliable software and decreased downtime. Lower MTBR reduces the availability of software to perform intended functions. Careful analysis must be performed before a given technology is chosen. The TRL is a very good indicator of future R & A of given software.

3.2 Number of open requirements (OR)

Open requirements have issues & unanswered questions. Unknown clarity on the requirements contributes to misunderstood requirements, increased redesigns, missed schedules, un-maintainable complex modules susceptible to higher failures and defects. These characteristics jeopardize the reliability of future Army vehicle software.

3.3 Number of planned technical reviews (TR)

Technical reviews are performed during software development phases to find problems soon. If the Army vehicle software development has planned for relatively fewer required technical reviews, it will be hard to find the problems. Fewer planned technical reviews increases rework, redesign, bad coding practices, defects, etc. The technical reviews consist of code, design, architecture, and integration reviews. Lack of these planned reviews is a very good indicator of future R & A of Army vehicle software.

3.4 Number of planned documentation (DOC)

Tasks for creating technical documents are a must requirement for software development. The higher the number of documentation tasks scheduled for complex functionality the lower the data integrity, interoperability, testing, and rework issues. Documentation includes code, design, architecture, test cases, and requirements in order to reduce future unknowns and maintenance issues. This is a very good indicator of future M of Army vehicle software.

3.5 Configuration management (CM)

CM allows all parts and versions of software to be properly integrated and documented. A greater number of configuration management tasks scheduled for complex functionality to reduce interoperability, maintenance, and testing issues. CM tasks such as source and documentation control, release schedules, etc contributes to a reduced logistics and maintenance footprint. This is a very good indicator of future M of Army vehicle software.

Fuzzy logic solution offers great advantages to solve complex problems using a number of inputs. Fuzzy logic [8] proposed by L. A. Zadeh has been used where uncertainty and no mathematical relations exists. Fuzzy logic provides rule based approaches to solve a given problem using simple steps. We propose the following Army vehicle software complexity prediction algorithm using fuzzy logic

```
Step1: Read inputs S = {Software 1…. Software N} for predicting software complexity;
Step2: for i=1 to N (for each software)
N (1) = collect TRL number from the technology
            Strategy document for software (i);
N (2) = calculate number of planned technical reviews
            (TR) from the project plan for software (i);
N (3) = calculate number of open requirements (OR)
            From the requirements analysis document for
        software (i);
N (4) = calculate number of planned documentation
            tasks (DOC) from the project plan for soft
            ware (i);
N (5) = calculate number of planned configuration management tasks (CM) from the project plan for software (i);
M (i) = N; (Store N array for software (i) in M array).
     end for
Step3: Read integer inputs array from M array;
Step4: Store Fuzzy rules in array X = {rule1.... rule15};
Step5: for i = 1 to N // Loop for computing software complexity for each software
  W = M(i)  //get the ith element from W array
  for j=1 to 5
          Y(i) = fuzzify (W(i));
       end for
Step6:   for i = 1 to 15
          if i <=11 then
            Z(i) = apply fuzzyrule(X(i)) on Y(1) & Y(2) & Y(3);
          else
             Z(i) = apply fuzzyrule(X(i)) on Y(4) & Y(5);
            end if
       end for
Step7: Compute Reliability, R = centroid De-fuzzufication of
        Z(1) to Z(11);
       Compute Availability, A = centroid De-fuzzufication of
       Z(1) to Z(10);
      Compute Maintainability, M = centroid De-fuzzufication
       of Z(12) to Z(15);
Step8: Predict Software complexity from R, A, and M values
end for
```

4 The Software Complexity Prediction Model

We propose a fuzzy logic based Army vehicle software complexity prediction model using the proposed five factors metrics. The software fuzzy logic toolbox was used to develop the model. The prediction model has three components i.e. fuzzification, rule-based fuzzy inference engine, and de-fuzzification. The model consists of five inputs, three outputs and 15 rules. According to the predefined rules, the model predicts the appropriate software complexity in terms of

RAM. Both the fuzzy inputs and outputs are modeled using the trapezoidal membership functions. The membership grades for TRL are described by LOW, MEDIUM, and HIGH membership functions (Fig. 1) and all other the fuzzy inputs are described by NOTHING, SOME and FULL membership functions (Fig. 2). The fuzzy output's membership grades are described by RED, YELLOW, and GREEN membership functions (Fig. 3).

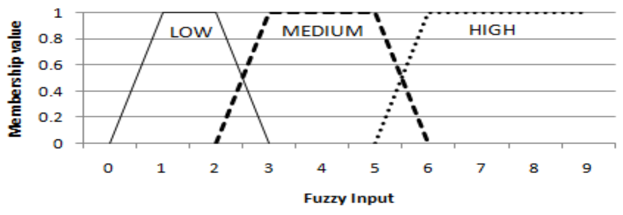


Fig.1 TR Fuzzy input membership

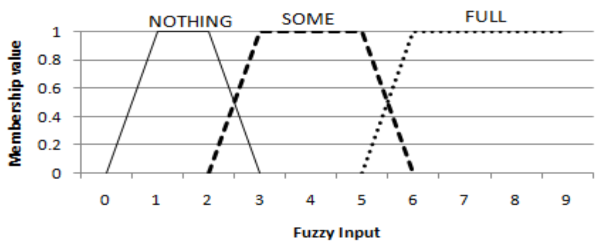


Fig.2 TR, OR, CM, and DOC Fuzzy input membership

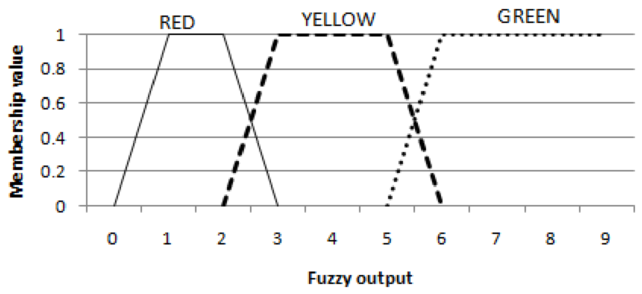


Fig.3 R, A, and M Fuzzy output membership

The fuzzy logic toolbox provides a rule-based model as a software prototype to analyze all the inputs and compute the output. The de-fuzzification rules are based on the proposed five factors metrics. The list below shows the 15 fuzzy rules to predict the software complexity using five inputs and three outputs.

1) If TRL= LOW & TR = NOTHING & OR =NOTHING then R = RED & A = RED
2) If TRL= LOW & TR = SOME & OR = NOTHING then R = RED & A = RED
3) If TRL = LOW & TR=FULL & OR=NOTHING then R=RED & A=YELLOW
4) If TRL=MEDIUM & TR=NOTHING & OR=NOTHING then R=RED & A=YELLOW
5) If TRL=MEDIUM & TR=SOME & OR=NOTHING then R=RED & A=YELLOW
6) If TRL=MEDIUM & TR=FULL & OR=NOTHING then R=YELLOW & A=YELLOW
7) If TRL=HIGH & TR=NOTHING & OR=NOTHING then R=RED & A=YELLOW
8) If TRL=HIGH & TR=SOME & OR=NOTHING then R=YELLOW & A=YELLOW
9) If TRL=HIGH & TR=FULL & OR=NOTHING then R=GREEN & A=GREEN
10) If OR=SOME then R=YELLOW
11) If OR=FULL then R=RED
12) If DOC=NOTHING || CM=NOTHING then M=RED
13) If DOC=SOME & CM=SOME then M=YELLOW
14) If DOC=SOME & CM=FULL then M=YELLOW
15) If DOC=FULL & CM=FULL then M=GREEN

R in the RED membership grades indicates that the reliability component of the Army vehicle software complexity is in trouble and needs significant improvements in "TRL" or "TR" or "OR". A in the RED membership grades indicates that the availability component of the Army vehicle software complexity is in trouble and needs significant improvements in "TRL" or "TR" factors. Similarly, M in the RED membership grades indicates that the maintainability component of the Army vehicle software complexity is in trouble and needs improvements in "CM" or "DOC" factors. The output values of YELLOW indicate that some improvements are needed for the associated factors. The output values of GREEN indicate no improvements needed for the associated factors. The Fig. 4 describes the complexity prediction model elements.

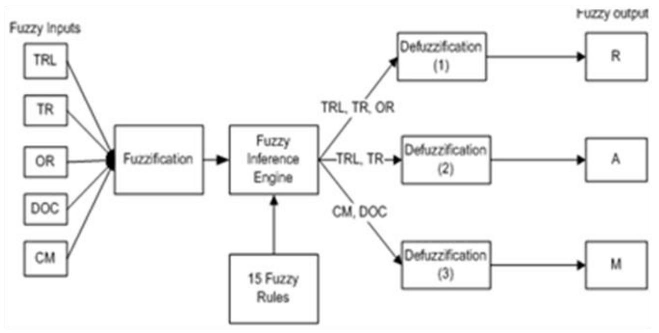


Fig.4 Software complexity model

The fuzzification model element fuzzifies the model inputs via a max function evaluation based on the membership functions defined to determine its appropriate membership grades.

In this model, when the inputs and outputs are fuzzified, the max function is applied between the membership functions. For example, the crisp value of 2.8 falls into two membership functions i.e. LOW (0.4) and MEDIUM (0.8), but when max function is applied between them, the membership grade falls into MEDIUM membership function. The fuzzified value is 0.8 for an input of 2.8. In this model, when the results are de-fuzzied, they use the centroid method of de-fuzzification. This process returns the center area of the curve

5 Example

Let's explain the Army vehicle software complexity prediction model using a simple example. The following are the example data: TRL = 5, TR=4, OR=1, DOC=3 and CM=4. Per the software complexity prediction algorithm, all the above five inputs are fuzzified using the max function. The fuzzified input values are TRL = 1 (MEDIUM), TR = 1 (SOME), OR = 1 (NOTHING), DOC = 1 (SOME), CM = 1 (SOME).

Now the above fuzzified inputs are tested by 15 fuzzy rules. Per rule#5, If TRL=MEDIUM & TR=SOME & OR=NOTHING then R=RED & A=YELLOW. Per rule#13, If DOC=SOME & CM=SOME then M=YELLOW From the two fuzzy rules #5 & #13, software complexity can be predicted. The results indicate that the reliability part of the software complexity as RED (crisp values between 0 & 3). Availability part of the software complexity is YELLOW (crisp values between 2 & 6). The maintainability part of the software complexity is YELLOW (crisp values between 2 & 6). The software fuzzy logic toolbox can be used to simulate five factors metric input to determine fuzzy outputs and the appropriate de-fuzzified values. Depending on the output membership grades appropriate fuzzy value for the output can be determined from the fuzzy logic toolbox output.

6 Conclusions

Historical software data based focus on fixing the symptoms rather than the problem. The proposed software complexity provides non-technical variables (factors) to predict the Army vehicle software complexity. The proposed metric elements are known to all parties involved in a software development and the metric data collection is simple.

When complexity is expressed in terms of RAM, the user can visualize which part of the complexity is having problems e.g. if R = YELLOW, A = GREEN, and M = RED then we can say the reliability and maintainability part of the software complexity is in trouble.

7 Acknowledgment

The authors wish to thank wishes to thank Pame Watts Dean of TARDEC University for her support

8 Disclaimer

9 References

[1] T.J. McCabe, “Complexity Measure”, IEEE Trans. Soft Eng., vol. SE-2, no. 4, pp. 308-320, Dec. 1976.

[2] Halstead, and H. Maurice, “Elements of Software Science”, Elsevier North-Holland, New York, 1977.

[3] Henry and Kafura,”Software Structure Metrics Based on Information Flow”, IEEE Trans. Soft Eng., vol. SE-7, no. 5, pp. 510-518, Sep. 1981.

[4] Warren Harrison,”An Entropy Based Software Complexity Measure”, IEEE Trans. Soft Eng., vol. SE-18, no. 11, pp. 1025-1029, Nov. 1992.

[5] Jingqiu Shao and Yingxu Wang, “A New Measure of Software Complexity based on Cognitive Weights”, Canadian

[6] Journal of Electrical and Computer Eng., vol.28, no.2, pp. 1333-1338, Apr. 2003.

[7] Khoshgoftaar and Munson, "Applications of a Relative Complexity Metric for Software Project Management ", Journal of Systems Software, vol. 12, no. 3, pp. 283-293, Jul. 1990.

[8] L. Zadeh “Fuzzy Sets,” Inform. Contr., vol. 8, no. 3, pp. 338-353, 1965.

JoinFS: a Semantic File System for Embedded Systems

Matthew Harlan Gabriel Parmer

Computer Science Department
The George Washington University
Washington, DC
{mharlan,gparmer}@gwu.edu

Abstract

Hierarchical file systems are the de-facto abstraction for storing information on modern computing systems. Though they are useful for providing structure for categorical data, their failings are most pronounced on consumer embedded devices. The limited interface for organizing a directory hierarchy, and abundant structured, network-accessible data complicate the management of the hierarchy.

JoinFS *attempts to enhance the tradition hierarchical file system abstraction by marrying it with an efficient query system. Special* dynamic directories *are populated not by a rigid hierarchy, but by an active matching of query terms to meta-data associated with the files in the system. Hierarchical dynamic directories provide the novel concept of categorizing semantics, rather than the data itself. These dynamic directories are exposed as normal filesystem directories, thus enabling applications and scripts to harness the querying power of* JoinFS *with negligible effort.* JoinFS *maintains file-system performance on normal files and directories, has a small footprint, and is implemented as a modular addition to traditional file-systems.*

1 Introduction

File systems provide a hierarchical name-space enabling the organization of system data into separate directories. Though pervasive, this abstraction is increasingly inappropriate with the steady increases in hard-drive/SSD size, amount of data stored, and the constrained interfaces used to access and organize embedded systems and consumer electronics. All of these trends make it more difficult to organize larger amounts of data. For example, cell phones and other personal embedded systems often do not provide a natural file-system interface for users to use to organize their data, yet the amount of data available to such systems is vast. Current mobile systems rely on individual applications to manually manage their own data. The static organization of directories seems unable to scale to vast amounts of complex data a system is exposed to, even on devices as small a cell-phones. The web is a collection of data with little (global) coordination and classification, and is naturally organized by *search*. Fundamentally, complex data often has no single natural category, and instead has many attributes, any one of which might be useful as a discriminator for a user or program.

Others have claimed that "hierarchical file systems are dead"[6], and that search-based data-access should be the dominant form of organization. Indeed, commodity operating systems have integrated search functionality in the form of Windows search [9] and Apple Spotlight [7]. Such concepts are not new: twenty years ago, the semantic filesystem [3] was proposed in which directories are organized by attributes, and in '93 the Inversion FS implemented a file system on top of a data-base [4] while exposing querying functionality. JoinFS is motivated by this shared goal of enabling semantically-aware access (i.e. that is aware of the meaning of the contents of files) to vast amounts of data. However, we also focus on the implementation of a practical system with low resource requirements, that is fully backwards compatible, and study its performance impact on modern hardware.

To provide a semantic, search-based interface to traditional FSs, JoinFS provides *dynamic directories*. Dynamic directories appear as file system directories, but are populated with either files or directories that semantically match some a predicate associated with the directory. Each file has a set of meta-data associated with it, and dynamic directory predicates are used to match this meta-data, thus presenting only those files relevant to the search. Importantly, we maintain the hierarchical structure even in dynamic directories: JoinFS enables dynamic directories to nest inside other dynamic directories, effectively returning the intersection of the result set for the parent and child dynamic directories. Thus, the normal manual organization possible in the hierarchical name-space can also be used in dynamic directories.

JoinFS design goals. We designed JoinFS with a number of goals in mind. These include:

• *Dynamic directories.* Fundamental to JoinFS is the concept of dynamic directories that are populated dynamically with content with meta-data that matches a predicate associated with the directory. This enables search at the

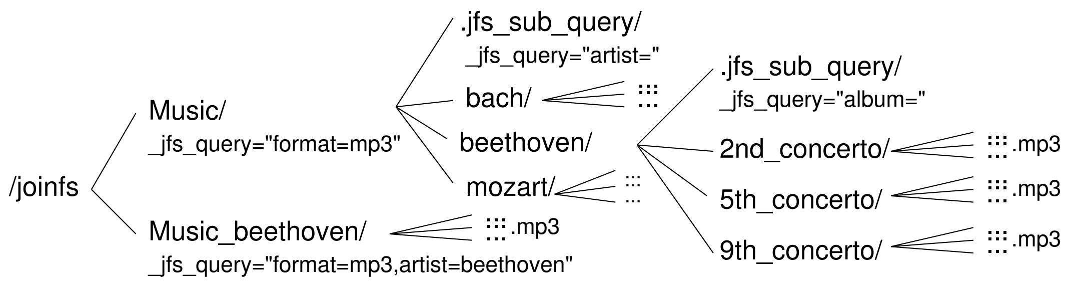


Figure 1. Example dynamic directory hierarchies created by specific search terms to demonstrate JOINFS usage. Though we use music in this example, dynamic directory hierarchies can be applied to any files and directories with any type of meta-data.

file-system level. Additionally, we marry this search capability with the hierarchical model by enabling *hierarchical dynamic directories*, thus enabling the manual categorization of the desired semantics of data, rather than the data itself.

• *Transparent Application Enhancement.* As JOINFS is implemented at the actual file-system level, and not in the shell or as a separate indexing program, any application can benefit from its functionality. For example, an image viewing program can get a succinct list of all images in the system by simply creating (or using an existing) an appropriately configured dynamic directory. Indeed this increases the power of even lowly shell scripts.

• *Backwards compatibility.* We strive for not only functional compatibility with traditional hierarchical systems, but also architect JOINFS as an *extension* on a normal file system. This means that a traditional FS is still used, and we maintain performance compatibility with previous systems.

• *Low resource requirements.* As appropriate for embedded devices including consumer electronics such as cell phones, we focus on minimal resource utilization including both CPU and memory. JOINFS uses only technologies and libraries that are appropriate and commonly found in these environments, and does not require – through also does not preclude – a kernel-level implementation.

Contributions. This paper makes the following contributions: 1) it introduces the concept of dynamic directories that marry semantic search and hierarchical categorization to enable *hierarchical categorization on semantics, not data*, 2) it details a practical user-level implementation of JOINFS that builds on commonly available technologies, and 3) empirically evaluates the system to measure its performance characteristics. JOINFS provides a practical and powerful solution to the data management problem and is tailored for use not only on conventional desktops and server, but also embedded and mobile systems.

The rest of this paper is organized as follows: Section 2 discusses how JOINFS is harnessed by applications and users. Section 3 outlines JOINFS's implementation, and Section 4 discusses its experimental validation. Section 5 covers the related work while Section 6 concludes.

2 JOINFS Interface and Use

Much of the novelty of JOINFS is in the interface of the semantic search with the file system, and in hierarchical dynamic directories. JOINFS must associate queries with directories to populate dynamic directories. For this purpose, and to store the meta-data associated with each file, we use POSIX extended attributes. Many modern file-systems enable the storage of meta-data in the form of untyped strings in extended attributes associated with a file.

Dynamic directories. The simplest example usage scenario for JOINFS use is to create a dynamic directory holding file matching a search. We use the `setfattr` command to associate a named attribute with its value. For this paper, we will describe setting attribute `n` to value `v` on file `f` as `(f,n,v)`. Given a file system with a number of file types, we can search for all of the `mp3` files by creating a directory `Music`, and setting a fixed attribute name `_jfs_query` to a list of `key=value` pairs: `(Music,"format=mp3")`. The directory will be populated with all mp3 music files. Queries can be more specific, for example, `(Music_beethoven,"format=mp3, artist=beethoven")`, in `Music_beethoven` will contain all `mp3`s with music by Beethoven. Dynamic directories is dynamically updated with changes in the file system. If new `mp3`s are added to the file system, or existing ones are removed, they will appear or disappear from the dynamic directory, respectively.

Hierarchical dynamic directories. JOINFS not only adds semantic search to the file system interface; it also maintains a hierarchical file system interface. This enables the simple categorization of search data. To enable hierarchical dynamic directories, each dynamic directory can optionally include a `.jfs_sub_query` directory. In this case, the parent dynamic directory's query will result in *directory* results, rather than files. The `.jfs_sub_query` directory can have queries associated with it similar to the parent dynamic directory: `(Music/.jfs_sub_query,"artist=")`. Intuitively, this says that the dynamic directory `Music` should be populated with a number of directories, each with the name

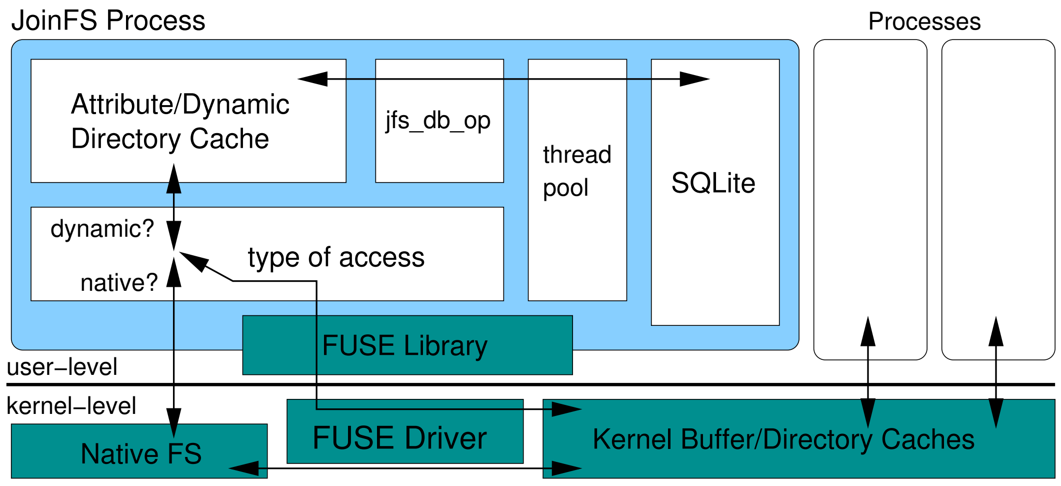


Figure 2. The high-level JOINFS design. The kernel buffers file data, and JOINFS passes file requests on to the native file system. Dynamic directories are handled by the logic and `SQLite` backend.

of one of the artists (i.e. the directories names are the unspecified values associated with given key). Each of these directories includes that artist's `mp3` files. Adding `Music/.jfs_sub_query/.jfs_sub_query` and the attribute `(Music/.jfs_sub_query/ .jfs_sub_query,"album=")`, will now create an organization where within `Music`, there is a directory for each artist, in each of those directories, there is a subdirectory for each album for that artist, and in that directory are the actual `mp3` files by that artist *and* in that album.

This example demonstrates how JOINFS *enables the categorization using hierarchical structures of not the data itself, but of the semantics of interest*. Dynamic directories are created for semantic concepts, not data, and JOINFS automatically populates the directory with the appropriate data.

2.1 Metadata Generation

Though the previous example involved `mp3` files, JOINFS is generic in that any file can be annotated with metadata specific to its semantics. This annotation is beyond the scope of JOINFS. We currently use shell scripts to parse `id3` tags for music files, but similar programs could insert semantically relevant metadata for other forms of data such as images, video, or source code.

3 JOINFS Design and Implementation

JOINFS uses a number of existing technologies to avoid recreating the wheel. We implement JOINFS on Linux, and use `FUSE` [1] to integrate with the file system namespace. `FUSE` is "File systems in USEr-space", and enables the implementation of file systems as separate user-level processes. Though `FUSE` has some performance disadvantages [5] compared to native, kernel file systems, it is appropriate for prototyping advanced file system functionality. With `FUSE`, file system operations and requests are translated into inter-process communication (IPC) with the file system process.

JOINFS makes use of a data-base backend to track file system files and meta-data. For this, we use the `SQLite` [8] database. Though `SQLite` does not typically provide competitive write performance compared to more complex databases, it is widely accepted on embedded platforms and is deployed, for instance, in the Android mobile platform. We find in our evaluation (Section 4) that its performance is not prohibitive, and its memory consumption is appropriate for a somewhat capable embedded platform.

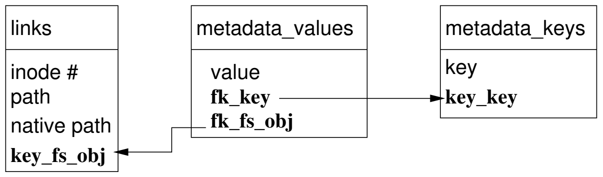


Figure 3. The database schema for JOINFS. Bold items are primary keys; arrows are foreign keys.

The database has a simple schema, and includes only three tables: `links`, `metadata_values`, and `metadata_keys`. The relations between these tables are depicted in Figure 3. The `path` is the complete path through the file system namespace exported by JOINFS to the actual file identified by `filename`.

3.1 Normal File System Operations

A key design decision in JOINFS is to minimize the involvement of the database in normal file operations. Due to this, we do *not* maintain all file data in the database, and instead use the native file system for holding file data. This means that critical read and write performance are not substantively effected by JOINFS, thus taking JOINFS off of a critical path. In fact, as much functionality as possible is handed off by JOINFS to the native file system. This includes permission checking, real path lookup, hard-link reference counting, and file I/O.

When a request to open, or read a directory is made, JOINFS must determine if the access is made to a dynamic directory and it should be handled via data-base accesses, or to a native file object and it is passed to the native file system. The JOINFS file system is mirrored on the native file system with the exception of the contents of dynamic directories. To determine if a file system request is for dynamic contents, JOINFS simply executes a `stat` on the object in the native file system. This does impose the overhead of an additional system call on each file access that reaches JOINFS (i.e. requests that aren't satisfied by the Linux buffer cache). This is a relic of our implementation on `FUSE` and this overhead would be diminished if JOINFS were implemented in the kernel.

JOINFS does impose overhead on some normal file operations. Namely, creating, deleting, and adding attributes to file system objects. File system object creation and deletion do entail some overhead as the file must be inserted into or removed from the `links` table, and attribute operations have overheads as they are stored in the `metadata` tables. We study this effect in Section 4.

3.2 Dynamic Directories and SQL Generation

When a request is made to read the contents of a directory (*e.g.* via `readdir`), and JOINFS finds that it does exist in the native file system, it must determine if it is a dynamic directory. To do this, JOINFS checks if the attribute `_jfs_query` exists for that directory. If the attribute has a value, the directory has dynamically generated contents. Attributes are cached, so this check can often be performed while avoiding data-base interaction. To return the contents of the dynamic directory based on the query, JOINFS implements a lightweight cache of pathnames used to save the resulting files generated from a data-base query. This technique enables the results of expensive SQL query synthesis, data-base queries, and parsing of the return values, to be saved for future access.

Our experiments demonstrate that the memory overhead for these caches is minimal. These caches serve the important functions of avoiding database accesses when possible.

If the results of listing a dynamic directory are not cached, the SQL query is formed from the key value pairs of all nested hierarchical dynamic directories. They intuitively equate to the intersection (or conjunction) of each of the terms. For each level of dynamic directories, an SQL sub-statement is constructed that matches the current metadata value of the parent directory (e.g. if in the "Beethoven" directory, it matches `key=artist` and `value=Beethoven`), and returns the set of file identifiers (database keys). The intersection of these keys is used to compose the queries of multiple levels of dynamic directories. When the final level of dynamic directories is found, and it is time to generate files, thus a SQL statement takes the file identifiers generated through this process, and returns file information that JOINFS uses as the return values for the `readdir` call.

3.3 Database Interaction

`SQLite` provides a fairly standard database interface. However, for performance, we added a thread pool, and an extra caching layer. All interactions with the database are done using a `jfs_db_op` structure. This API provides functions for creating queries, synchronizing processes with the multithreaded `SQLite` query engine that actually executes the database operations, and cleans up database operations once they finish. The API essentially acts as a layer between JOINFS and the database, enabling the migration away from `SQLite` in the future if desired. JOINFS uses `SQLite` as it has a low footprint, has acceptable performance, and is widely available for many embedded and mobile platforms.

Thread pool for increased concurrency. In order to enable fast concurrent database reads, JOINFS makes use of a thread pool. This thread pool initially starts with the same number of threads as `FUSE`, but can expand and shrink as necessary. The SQL interface adds all database read operations to a job queue. The read pool grabs jobs from this queue, executes them, and returns the results. The thread pool then wakes up the thread that added the job so that they can continue processing. JOINFS prevents write operations from taking place by restricting all database connections to read-only for all reader threads.

The thread pool has the additional benefit of enabling database connection caching with `SQLite`. Without the thread pool, a database connection would have to be made for each database operation which significantly degrades performance.

JOINFS handles writes separately because `SQLite` does not support concurrent write operations. JOINFS instead uses a single writer thread with its own job queue to perform all database writes. JOINFS also supports executing multiple writes at once as a transaction. This enables more complex inserts without having to repeatedly attain an exclusive lock on the database file.

Caching to avoid database access. Even though databases often cache their results in memory, it is desirable in certain situations to have an additional caching layer

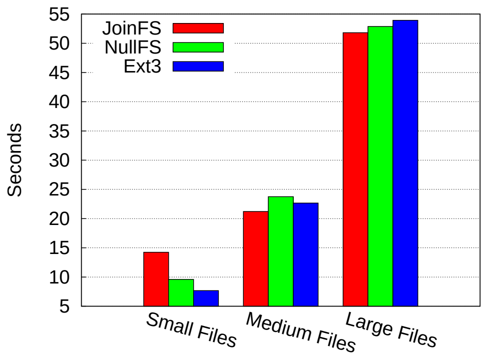


Figure 4. Overheads of transferring a directory hierarchy into JOINFS.

in JOINFS to avoid data-base interaction all-together. Thus, JOINFS caches pathnames and metadata for the contents of dynamic directories to avoid having to query the data-base again.

3.4 Hybrid Directories

As JOINFS marriages file systems with database concepts, there are interesting interplays between the two abstractions. For example, a natural question is what happens when a file is created *inside* a dynamic directory? In such cases, JOINFS stores the file or directories created in the dynamic directory's native backing store [1]. Any files created in these directories will appear in the contents of the dynamic directory. In the future, we will support automatically generating metadata for files placed into dynamic directories. This metadata will be determined by the metadata search terms of the dynamic directory.

4 Experimental Results

To evaluate the performance impact of JOINFS, we use a machine with a 3.3GHz Core i7-920, Intel processor with 6GB of RAM, and a 7200 rpm Western Digital Caviar Black 1 TB SATA drive. We use Ubuntu 10.04 LTS configured to use the ext3 file system and the Ubuntu version of Linux, `kernel-2.6.32-30-generic`.

4.1 Data Transfer

In this section, we wish to investigate the overheads of JOINFS for normal file operations (not to dynamic directories) such as creation, and population of file data. JOINFS is not designed to make these operations faster, and is instead meant to minimize the overheads to these operations.

[1]Each dynamic directory and `.jfs_sub_query` are stored on the native file system. The query results that populate those directories dynamically are not.

Figure 4 plots the amount of time it takes to transfer files from an `ext3` file system mount point into a mount point for 1) JOINFS using a normal directory, 2) NULLFS the `FUSE` file system that simply passes all file system operations on to the Linux file system (in this case `ext3`), and 3) `ext3`, the default Linux file system. We include the results for `ext3` only as a lower-bound on performance. `FUSE` imposes some overhead, especially on transfers for small files [5], so we compare more directly to NULLFS. We transfer three different classes of files: 1) *small files* – the Linux source code directory which includes 37,998 comparably small files totaling 400.3 MB (10.5 KB each on average), 2) *medium files* – a set of 209 music files totaling 1.4 GB, and 3) *large file* – 1 movie file totaling 1.4 GB. In Figure 4, we plot the average of 5 runs for each of the file systems.

For small files, we see that there is some overhead both for NULLFS, and JOINFS, and that JOINFS has an overhead over NULLFS of 48% (14.24 seconds vs 9.59 seconds). Though this overhead is not insignificant, as the file operations become dominated by reads and writes instead of file creation (for medium and large files), we see that all techniques are roughly equivalent (this echos results from [5]). For devices such as mobile phones and embedded systems where the focus is on data consumption rather than creation, we believe these overheads are acceptable.

Filesystem	Memory consumption
NULLFS	6.9 MB
JOINFS	9.8 MB
JOINFS Overhead	75 B per file

Table 1. Memory consumption of JOINFS after creation of 38,208 files.

Table 1 plots the memory usage of JOINFS and NULLFS. Memory consumption of the system is an important factor in analyzing its acceptability in many consumer electronics and embedded applications. We find that JOINFS imposes some overhead due to the data held in the `links` table of 75 bytes per file. Even in memory-constrained devices, we deem this to be acceptable.

4.2 Dynamic Directory Performance

In this section, we investigate the performance of the dynamic directory implementation in JOINFS. We wish to study the basic costs of doing a `readdir` in dynamic directories that queries the meta-data for all files, and produces the resulting directory contents. We study the effects of changing the number of files returned by the query, and the depth in the hierarchy of dynamic directories of the results.

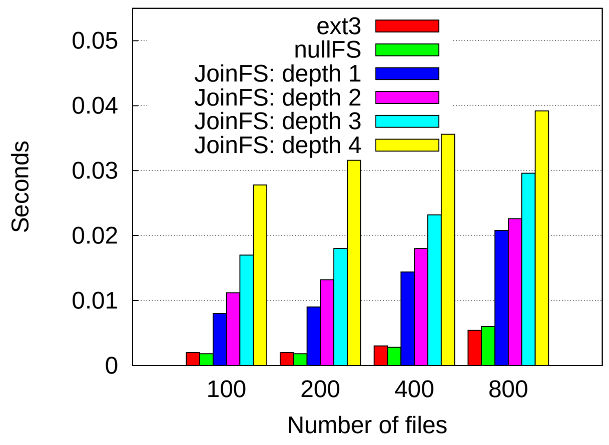


Figure 5. Latency for listing a variable number of files from a directory hierarchy. For JOINFS, we vary the depth of the hierarchy of dynamic directories, and measure query latency.

Figure 5 plots the latency for completing an `ls` command in a directory for 1) JOINFS, 2) NULLFS, and 3) `ext3`. For JOINFS, we plot the latency in a sub-directory of a variable depth into the hierarchy of dynamic directories. The other file systems do not use nested directories. For all file systems, we plot the latency for a varying number of files in the directory. Again, we plot the average of 5 runs.

JOINFS for a dynamic directory at a depth of 1 is within a factor of 5 of the latency for all numbers of query results to NULLFS. As we have nested dynamic directories, the overheads grow as complicated queries are synthesized along the path of directories, and the SQL queries generated increase in complexity. We don't believe these overheads are significant enough to discourage the use of nested dynamic directories for a categorical organization of semantic results.

Filesystem	Memory consumption
NULLFS	0.42 MB
JOINFS	2.8 MB
JOINFS Overhead	396 B per file

Table 2. Memory consumption of JOINFS for 6000 files in dynamic directories.

Table 2 plots the memory overhead after the queries (i.e. `readdirs`) have completed for all number of files (1500 in total), for all dynamic directory depths (thus, 6000 files listed in total). The overhead *above* NULLFS per file queried is 396 bytes per file. This includes index overheads in `SQLite`, and the overhead of JOINFS data-structures. We believe that an overhead of 2.8 MB is not insignificant, but is acceptable for queries that return 6K files.

5 Related Work

Multiple operating systems have investigated incorporating relational concepts into the file system [2, 10]. We are unaware of any previous system that has aggressively mixed a relational data-base with a file system while enhancing both categorical hierarchical structures with search *and* search with categorical, hierarchical structure. *Dynamic directories* provide not only a file-system representation of search, but nested dynamic directories enable the categorical, hierarchical organization of not data, but semantic information.

Semantic search and database file systems. [6] proclaims that hierarchical file systems are dead. JOINFS takes a more nuanced approach in which we still support hierarchical file systems, but integrate in a novel way dynamic directories into a hierarchical structure. [3] provides an implementation of semantic search in a file system, but does not provide nested semantic searches. As JOINFS is implemented on a modern OS, we provide valuable insights into its applicability in embedded and mobile systems. [4] implements a file system interface on top of a data-base. Unlike this approach, JOINFS focuses on backwards compatibility where possible by relying on a traditional file system for data-storage and hierarchical organization while providing dynamic directories as an extension.

Data indexing on top of the file system. Many modern systems use search applications on top of an existing file system to index data and enable search [9, 7]. Even the lowly `locate` command is implemented in this fashion.

6 Conclusions

This paper presents the design and implementation of JOINFS. JOINFS integrates semantic search on top of a hierarchical file system structure in a manner that enables the continued use of optimized, existing file system, while adding *dynamic directories* that are populated by a search term. In integrating the hierarchical namespace with semantic search, JOINFS provides the novel concept of nested dynamic directories. Each sub-directory enables the refinement of the parent directory's search. In such a way, the results of a search can include not only files, but dynamic directories themselves. This enables the normal hierarchical techniques for categorization to be applied not to the data itself, but to the semantic searches.

In integrating search in a natural way with hierarchical file-systems, the power of semantic search is made available to any application without any additional code. A mobile application for displaying `pdf` files can easily access all `pdfs` in the system by listing a dynamic directory. Even shell scripting is enhanced.

We have shown that the processing overheads imposed by JOINFS are acceptable for a great many applications, and that the memory overheads are generally acceptable for a variety of embedded and consumer electronics applications.

The JOINFS source is located at `http://github.com/mharlan/joinFS`.

References

[1] Filesystem in user space: http://http://fuse.sourceforge.net/, retrieved 4/1/11.

[2] D. Giampaolo. *Practical File System Design with the Be File System*. Morgan Kaufmann, 1999.

[3] D. K. Gifford, P. Jouvelot, M. A. Sheldon, and J. W. O'Toole, Jr. Semantic file systems. In *Proceedings of the thirteenth ACM symposium on Operating systems principles*, pages 16–25, 1991.

[4] M. Olson. The design and implementation of the inversion file system. In *Proceedings of the USENIX Winter 1993 Technical Conference*, 1993.

[5] A. Rajgarhia and A. Gehani. Performance and extension of user space file systems. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, SAC '10, 2010.

[6] M. Seltzer and N. Murphy. Hierarchical file systems are dead. In *Proceedings of the 12th conference on Hot topics in operating systems*, 2009.

[7] Apple spotlight: http://www.apple.com/macosx/what-is-macosx/spotlight.html, retrieved 4/1/11.

[8] SQLite: http://www.sqlite.org/, retrieved 4/1/11.

[9] Windows search: http://www.microsoft.com/windows/products/winfamily/desktopsearch/default.mspx.

[10] WinFS: http://blogs.msdn.com/b/winfs/, retrieved 4/1/11.

Specification of Embedded Systems Environment Behaviour with Alvis Modelling Language

M. Szpyrka[1], **L. Kotulski**[1], **and P. Matyasik**[1]
[1]Department of Automatics, AGH University of Science and Technology, Kraków, Poland

Abstract—*Alvis, a modelling language for concurrent systems, combines the advantages of formal methods and practical modelling languages. Even though Alvis has its origins in process algebras, instead of algebraic equations, it uses a high level programming language for the description of agents behaviour and a graphical language for the modelling of interconnections among agents. An Alvis model can be translated into an LTS graph (a labelled transition system) that is used for formal verification purposes. Alvis has been designed especially for embedded systems and one of its main advantages is a possibility of a flexible specification of a behaviour of an embedded system's environment. Instead of describing an environment as a part of an Alvis model, we can specify signals/values sent or received by the environment. Moreover, such an environment border can be freely modified while developing the corresponding embedded system. The paper describes the Alvis method of specification of an embedded system's environment. This is illustrated with an Automatic Train Stop system example.*

Keywords: Alvis, embedded system, environment specification

1. Introduction

An embedded system is one that is a part of a larger one. It is surrounded by other parts of the larger system that constitute the embedded system environment. Such an embedded system collects inputs that come from its environment (from sensors) and provide outputs that go to the environment (to controllers). To verify an embedded system formally we cannot separate it from its environment. Thus, if a formal language is used e.g. Petri nets [1], [2], [3], time automata [4], process algebra [5] etc., an embedded system model must include both the system and its environment. As a result of such a situation a model is often significantly more complex and the state explosion problem makes a formal verification difficult or even impossible.

For example, let us consider a simple Automatic Train Stop system (ATS system for short). This is a kind of an Automatic Train Protection system used to guarantee a train safety even if a driver is not capable of controlling the train. In the ATS system, a light signal is turned on every 60 seconds to check whether a driver controls the train. If the driver fails to acknowledge the signal within 6 seconds, a sound signal is turned on. Then, if the driver does not disactivate the signals within 3 seconds, using the acknowledge button, the emergency brakes are applied to stop the train automatically. An RTCP-net (Petri net) model of such a system is presented in [3]. To verify the Petri net model of the ATS system, it was necessary to add places and transitions that simulate the driver behaviour.

The paper presents the Alvis approach to modelling a behaviour of an embedded system's environment. Alvis [6], [7] is a novel modelling language designed for real-time systems, especially for embedded ones. The main goal of the Alvis project was to strike a happy medium between formal and practical modelling languages. We use the language to design an embedded system software, but from an Alvis source code we can designate its formal description in the form of a Labelled Transition System automatically.

The key concept of Alvis is an *agent* that denotes any distinguished part of the system under consideration with a defined identity persisting in time. An Alvis model is a system of agents that usually run concurrently, communicate one with another, compete for shared resources etc. Agents are divided into three groups: *active agents* can be treated as processing nodes, *passive agents* represent shared resources and *hierarchical agents* represent submodels. A model consists of three layers. The *code layer* uses Alvis statements supported by the Haskell functional programming language [8] to define a behaviour of individual agents. The *graphical layer* is used to define communication channels among agents. The layer takes the form of a hierarchical graph, that allows designers to combine sets of agents into modules that are represented as hierarchical agents. The third *system layer* is used for the simulation and verification purposes. It is the predefined one and is strictly connected with the considered system hardware environment.

One of the main Alvis advantages is the modelling of the considered embedded system's environment. Instead of designing the environment as a part of the model it is possible to specify signals generated or collected by the environment in a very simple way. This approach is also very useful from the analysis point of view. We can freely move the system border. In other words, we can start with modelling a very small part of the considered system in the first stage, moving all other parts to the environment. Then we can add more agents in the next stages. A state of a model is a sequence of agents' states. Moving some agents to the embedded system environment can reduce the model states space significantly.

The paper is organised as follows. Section 2 provides a short presentation of the Alvis language. Section 3 deals with the communication among agents. The method of the specification of an environment is discussed in Section 4. Two models of the considered ATS system are describe in Section 5. The paper is summarised in the final section.

2. Alvis modelling language

Alvis has its origins in the CCS process algebra [9], [5] and the XCCS language [10]. The main result of the fact is the communication model used in Alvis that is similar to the one used in CCS and the rendes-vous mechanism used in Ada [11]. However, Alvis uses a simplified rendez-vous mechanism with equal agents without distinguishing servers and clients. In contrast to synchronous programming languages like Esterel [12] or SCADE [13], Alvis does not use the broadcast communication mechanism. Only agents connected with a communication channel can communicate one with another. A few constructs in Ada were an inspiration while developing Alvis. For example, protected objects have been used to define passive agents and the Ada *select statement* has been used to define the Alvis *select statement*. An Alvis model composed of few agents that work concurrently is similar to an Ada distributed system. Active agents can be treated as processing nodes, while passive agents as storage ones. Alvis has many features in common with E-LOTOS – an extension of the LOTOS modelling language [14]. Alvis, like E-LOTOS, was intended to allow formal modelling and verification of distributed real-time systems. In contrast to E-LOTOS, Alvis provides graphical modelling language. Moreover, Alvis Toolkit supports an LTS graph generation, which significantly simplifies a formal verification of models. Alvis has also many features in common with System Modelling Language (SysML)[15] – a general purpose modelling language for systems engineering applications. It contains concept similar to SysML ports, property blocks, communication among the blocks and hierarchical models. Unlike SysML, Alvis combines structure diagrams (block diagrams) and behaviour (activity diagrams) into a single diagram. In addition, Alvis defines formal semantics for the various artifacts, which is not the case in SysML. Due to the use of Ada origins, VHDL [16] and Alvis have a similar syntax for communication and parallel processing. The concept of an agent in Alvis is also similar to a design entity in VHDL and both languages use ports for communication among system components. It should be noted, however, that Alvis is closely linked with a graphical model layer. Graphical composition allows for easier identification of the system hierarchy and components.

From programmers point of view, it is necessary to design two layers of an Alvis model – graphical and code ones. The graphical layer is called a *communication diagram*. It contains active agents drawn as rounded boxes, passive agents drawn as rectangles and hierarchical agents indicated by black triangles. *Ports* are drawn as circles placed at the edges of rounded boxes or rectangles. Alvis agents can communicate one with another directly using communication channels. A *communication channel* is defined explicitly between two agents and connects two ports. Communication channels are drawn as lines (or broken lines). An arrowhead points out the input port for the particular connection. Communication channels without arrowheads represent pairs of connections with opposite directions.

Alvis communication diagrams enable distributing parts of a diagram across multiple subdiagrams called *pages*. Pages are combined using the so-called *substitution mechanism*. A hierarchical agent from one level can be replaced by a page on the lower level, which usually gives a more precise and detailed description of the activity represented by the agent.

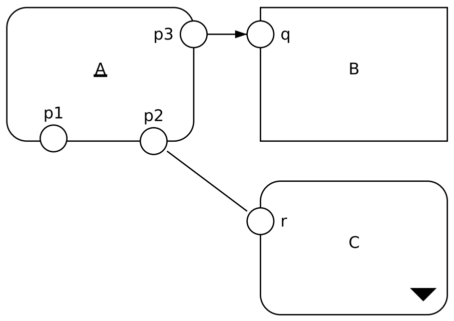


Fig. 1
ELEMENTS OF ALVIS COMMUNICATION DIAGRAMS

The code layer is used to define data types used in the model under consideration, functions for data manipulation and a behaviour of individual agents. The layer uses the Haskell functional language (e.g. the Haskell type system) and original Alvis statements. The set of Alvis statements is given in Table 1. To simplify the syntax, the following symbols have been used. `A` stands for an agent name, `p` stands for a port name, `x` stands for a parameter, `g`, `g1`, `g2`,... stand for guards (Boolean conditions), `e` stands for an expression and `ms` stands for miliseconds. Each non-hierarchical agent placed in a communication diagram must be defined in the corresponding code layer and vice-versa. For more details see [6], [7].

3. Communication between agents

Before discussing the communication with an environment let us recall some information about the communication among agents in Alvis. Alvis uses two statements for the communication. The *in* statement for collecting data and *out* for sending. Each of them takes a port name as its first argument and optionally a parameter name as the second. A communication between two active agents can

Table 1
ALVIS STATEMENTS

Statement	Description
cli	Turns off the interrupts handlers.
critical {...}	Define a set of statements that must be executed as a single one.
delay ms	Delays an agent execution for a given number of miliseconds.
exec x = e	Evaluates the expression and assign the result to the parameter; the **exec** keyword can be omitted.
exit	Terminates the agent that performs the statement.

if (g1) {...}
elseif (g2) {...}
...
else {...}
	Conditional statement.

in p **in** p x	Collects a signal via the port p. Collects a value via the port p and assigns it to the parameter x.
jump label	Transfers the control to the line of code identified with the label.
jump far A	Transfers the control to the agent A.

loop (g) {...}
 	
loop (**every** ms)
{...}
loop {...}
	Repeats execution of the contents while the guard if satisfied..
	Repeats execution every
	ms miliseconds.
	Infinite loop.

null	Empty statement.
out p **out** p x	Sends a signal via the port p. Sends a value of the parameter x via the port p; a literal value can be used instead of a parameter.
proc (g) p {...}	Defines the procedure for the port p of a passive agent. The guard is optional.

select {
  alt (g1) {...}
  alt (g2) {...}
  ... }
	Selects one of the alternative choices.

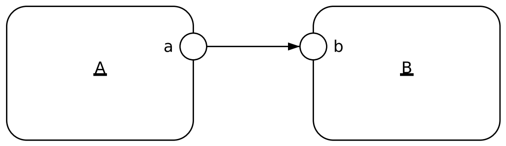
start A	Starts the agent A if it is in the *Init* state, otherwise do nothing.
sti	Turns on the interrupts handlers.

be initialised by any of them. The agent that initialises it, performs the *out* statement to provide some information and waits for the second agent to take it, or performs the *in* statement to express its readiness to collect some information and waits until the second agent provides it.

Let us consider the example shown in Fig. 2. Both agents performs only one statement connected with a communication between them. To describe the current state of an agent, we need a tuple with four pieces of information [6]:

- agent mode (am);
- program counter (pc);
- context information list (ci);
- parameters values tuple (pv).

The mode is used to indicate whether an agent is for example *running* or *waiting* for an event. The program counter points out the current or the next step to be executed. The context information list contains additional information about the



```
agent A {
  out a;
}
agent B {
  in b;
}
```

Fig. 2
COMMUNICATION BETWEEN ACTIVE AGENTS

current agent's state e.g. the name of the port used for the current communication. The parameters values tuple contains values of agent's parameters.

Thus, the initial state of the considered model can be described as follows:

```
A: (running,1,[],())
B: (running,1,[],())
```

The agents can perform their steps concurrently that moves the model to the final state:

```
A: (finished,0,[],())
B: (finished,0,[],())
```

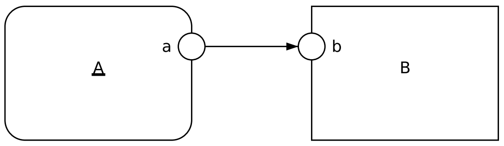
Assume that only the A agent executed its step. In such a case the system state is as follows:

```
A: (waiting,1,[out(a)],())
B: (running,1,[],())
```

It means that A is waiting for another agent to collect a signal from the port a. To complete the example discussion, it should be underlined that the following state is also possible.

```
A: (running,1,[],())
B: (waiting,1,[in(b)],())
```

A communication between an active and a passive agent can be initialised only by the former. Any procedure in Alvis uses only one either input or output parameter (or signal in case of parameterless communication). In case of an input procedure, an active agent calls the procedure using the *out* statement (and provides the parameter, if any, at the same time). If the corresponding passive agent is in the *waiting* mode and the procedure is accessible, the agent starts it in the active agent context. The passive agent collects the signal/parameter using the *in* statement, but it is not necessary to put the statement as the first procedure step. Similarly, in case of an output procedure, an active agent calls the procedure using the *in* statement. The passive agent returns the result using the *out* statement, but it is not necessary to put the statement as the last procedure step. A procedure is finished after executing its last step.



```
agent A {
  out a;
}
agent B {
  counter ::Int = 0;
  proc b {
    in b;
    counter = 1; }
}
```

Fig. 3
COMMUNICATION BETWEEN AN ACTIVE AND A PASSIVE AGENT

Let us consider the example shown in Fig. 3. The initial state is presented below:

```
A: (running,1,[],())
B: (waiting,0,[in(b)],(0))
```

A passive agent is always in one of two modes: *waiting* or *taken*. The former one means that the agent is inactive and waits for another agent to call one of its accessible procedures. In such a situation *pc* is equal to zero and *ci* contains names of accessible procedures. The *taken* mode means that one of the passive agent procedures has been called and the agent executes it. In such a case, *ci* contains the name of the called procedure (i.e. the name of the port used for the current communication).

The next states for the model under consideration are as follows:

```
A: (running,1,[proc(B.a)],())
B: (taken,1,[in(b)],(0))

A: (running,1,[proc(B.a)],())
B: (taken,2,[in(b)],(0))

A: (finished,0,[],())
B: (waiting,0,[in(b)],(1))
```

4. Communication with environment

Alvis agents may contain ports that are not used in any connection. Such ports are called *border ports* and are used for a communication with the considered system environment. Border ports can be used both for collecting or sending some information to the embedded system environment. Properties of border ports are specified in the code layer preamble with the use of the *environment* statement. Each border port used as an input one is described with at least one *in* clause. Similarly, each border port used as an output one is described with at least one *out* clause. Each clause inside the *environment* statement contains the following pieces of information:

- *in* or *out* key word,
- the border port name,
- a type name or a list of permissible values to be sent through the port,
- a list of time points, when the port is accessible,
- optionally some modifiers: *durable, queue, signal.*

```
in a [1..4] [];
in b [1..4] (map (100*) [1..]);
in c [1..4] (map (100*) [1..]) signal;
in d [1..4] (map (100*) [1..]) durable;
in e [1..4] (map (100*) [1..]) queue;
in f [1..4] (map (100*) [1..]) signal durable;
in g [1..4] (map (100*) [1..]) signal queue;
```

Fig. 4
BORDER PORTS' SPECIFICATION

Let us focus on the description of border ports presented in Fig. 4. Signals directions are considered from an embedded system point of view, thus all considered ports are used to send information from an environment to the corresponding embedded system. In each case, one of the values $1, 2, 3, 4$ (at random) can be collected through a port. However, the ports differ about the time points when values are accessible.

a	A value from the port can be collected at any time point. An agent that performs the *in* statement receives the value immediately (never waits for it). Such border ports are useful for a modelling of input sensors whose values can be read at any time.
b	Every 100 ms a value is provided by the environment via the port. If none agent waits for it (*waiting* mode), the value is lost.
c	The port behaves similar to the b one, but the signal may not be provided.
d	Every 100 ms a value is provided by the environment via the port. The value is accessible for the corresponding embedded system until an agent collects it. If while waiting for a collecting the value, another one is sent via the port, the previous one is overwritten.
e	The port behaves similar to the d one, but if while waiting for a collecting the value, another one is sent via the port, it is put into a FIFO queue.
f	The port behaves similar to the c one, but the value is accessible for the corresponding embedded system until an agent collects it or it is overwritten.
g	The port behaves similar to the f one, but the values are put into a FIFO queue.

The specifications of the ports $b, \ldots, g$ ports use the Haskell `map` function and an infinite list. For more details (if necessary) see [8].

If a border port is used for a parameterless communication, then the first list is empty. If a border port is used both as an input and an output one, then it must be described both with the *in* and *out* clauses. If different kinds of signals can be sent through a border port, then more than one *in* or *out* clause can be used.

It should be underlined that only the *signal* modifier should be used in the final model of an embedded system. Other modifiers are defined mainly for the verification purposes, if reduced models are considered.

Border ports must have unique names in a model. The same name of a border port used twice means that two agents use the same border port.

5. ATS system models

Trains could not run safely without signalling devices. Some automatic systems are used to transfer signals directly to a driver cab. A driver must always obey such a signal, but the possibility of human error can cause serious accidents. As it was already said, Automatic Train Protection (ATP) systems are used to guarantee a train safety even if a driver is not capable of controlling the train. Furthermore, computer systems can drive a train without a human support. The Automatic Train Stop considered here is used to check whether a driver controls the train.

The default time unit in Alvis is 1 millisecond. However, due to the specific features of the system under consideration, we will use 1 second as the basic time unit. As a result of this assumption, we will omit durations of steps execution. A single step in this example takes about 1 or 2 milliseconds, so they do not influence the system properties.

5.1 First approach

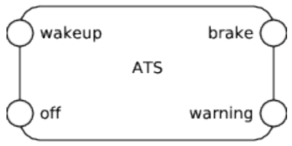
We start with a model that contains only one agent called ATS. Other elements: the cab console, timer, brake etc. are elements of our system environment. The model is shown in Fig. 5. Comments contain the numbers of steps.

In the considered example all ports are border ones. However, the model is already suitable to verify properties of the ATS agent. The *off* signal can be generated any one second, but it does not influence the system behaviour, if it is generated before the system is waked up. It is possible to specify the *off* port behaviour in a more sophisticated way using Haskell functions:

```
offlist p0 p1 p2 n k
  | k <= n     = (p1 + k * p2)
               : offlist p0 p1 p2 n (k + 1)
  | otherwise = offlist p0 (p0 + p1) p2 n 1
offlist' = offlist 60000 61000 1000 10 1
```

In the presented example, the *off* signal may be generated only for 10 s after the $wakeup$ signal.

A formal representation of an Alvis model is an LTS graph (Labelled Transition System) that contains all reachable states of a considered system and all transitions among



```
environment {
  in wakeup   [] (map (60*) [0..]) durable;
  in off      [] [1..] signal;
  out warning [0,1,2] [];
  out brake   [] [];
}

agent ATS {
  loop {                                -- 1
    in wakeup;                          -- 2
    out warning 1;                      -- 3
    select {                            -- 4
      alt (ready [in(off)]) {
        in off;                         -- 5
        out warning 0; }                -- 6
      alt(delay 6) {
        out warning 2;                  -- 7
        select {                        -- 8
          alt (ready [in(off)]) {
            in off;                     -- 9
            out warning 0; }            -- 10
          alt (delay 3) {
            out brake;                  -- 11
            exit; }                     -- 12
        } } } } }
```

Fig. 5
ATS SYSTEM – MODEL 1

them. A part of such an LTS graph for the considered model is shown in Fig. 6. Let us note that an LTS graph is generated for an Alvis model automatically. The complete graph contains 118 nodes (states). Most of them are results of the *off* border port specification. In the considered example the signal can be generated (label `in(*.off)`) every one second but most of these signals are ignored. The `in(!*.off)` label denotes lack of the *off* signal at a moment it could be generated. The included part of the LTS graph contains, among other things, the system time out 6 s after the *wakeup* signal.

5.2 Second approach

In the second approach, we decided to treat the driver *console* as a part of the embedded system. Moreover, the system contains a timer that wakes it up every 60 s. The model is shown in Fig. 7.

The second model contains only two border ports. In spite of the fact that two new agents have been included into the model, the definition of the ATS agent is still the same. The

[0]The complete LTS-graph can be found at Alvis website `http://fm.ia.agh.edu.pl`.

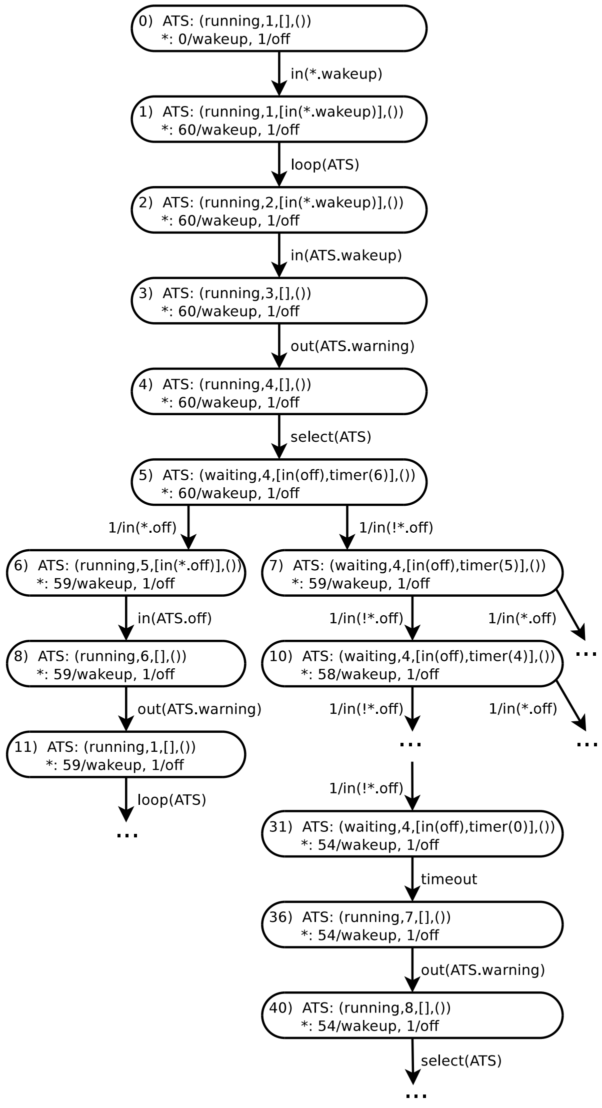
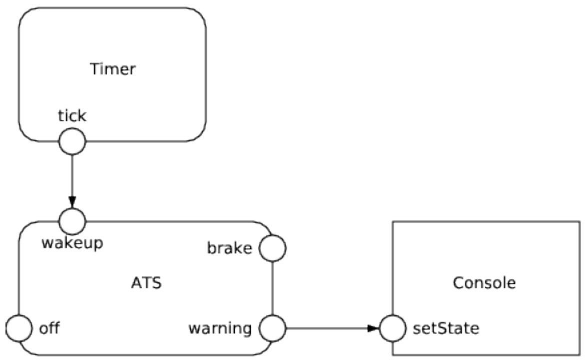


Fig. 6

PART OF THE LTS GRAPH FOR THE ALVIS MODEL SHOWN IN FIG. 5

driver console is represented by the *Console* passive agent with a single procedure used to set the console state. The *Timer* agent is an active one with a *loop every* statement. The agent sends the *tick* signal every 60 s.

A part of the LTS graph for the model is shown if Fig. 8. The graph contains 134 nodes. It is easy to find out similarities between these two LTS graphs. Both models have similar properties e.g. one deadlock (after turning on the brake), and similar paths in the corresponding LTS graphs. Frankly speaking there is not a big difference between sizes of these two graphs – only 16 nodes. On the other hand, the first LTS graph contains 49 nodes that are the result of generating the *off* signals that have been omitted by the ATS system, while the second graph contains 50 such nodes.



```
environment {
  in off [] signal;
  out brake [] []; }

agent ATS {
  -- ...
}
agent Timer {
  loop (every 6000) {
    out tick; }
}
agent Console {
  state ::Int = 0;
  proc setState { in setState state ; }
}
```

Fig. 7

ATS SYSTEM – MODEL 2

These nodes can be eliminated from the graphs if a more sophisticated specification of the *off* port is used. From this point of view, we have two graphs with 69 and 84 nodes respectively, i.e. the difference is 18%.

6. Conclusions

The presented examples of ATS systems allow one to find out the usefulness of the Alvis language for the design of embedded systems. The possibility of moving borders of an embedded system environment allows designers to develop a system increasing the number of its details in subsequent stages. Thus, we can design a small part of the target system and test its behaviour or verify it in a formal way using the LTS graph. Then, a more detailed version of such a system can be designed and we check whether the new version is compatible with the old one.

On the other hand, the presented approach can be useful for verification of more complex models with a very big state space. We can divide such a model into a set of subsystems and verify each of them separately. For each subsystem, we treat other parts of the model as the subsystem environment. The sum of sizes of LTS graphs generated for such subsystems is usually less than the size of the LTS graph generated for the whole system.

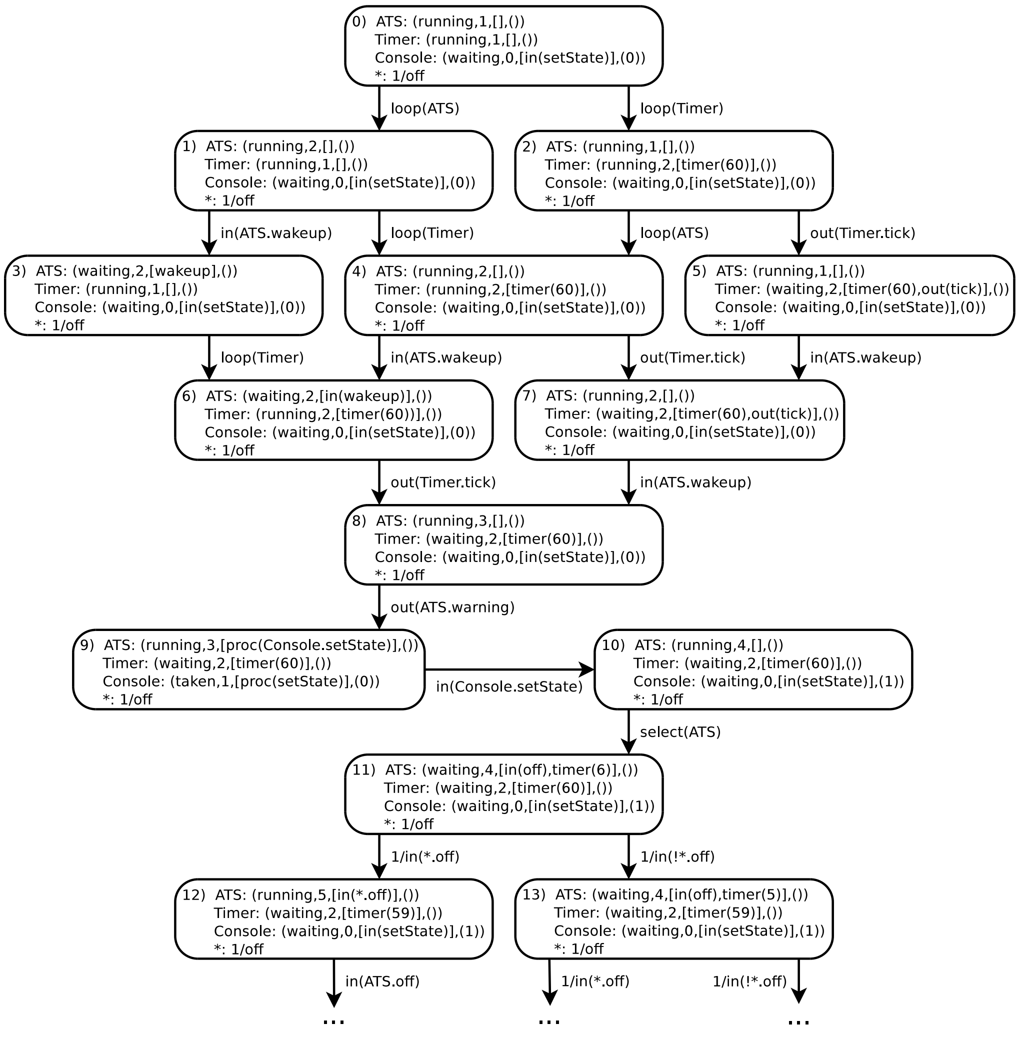

0) ATS: (running,1,[],())
Timer: (running,1,[],())
Console: (waiting,0,[in(setState)],(0))
*: 1/off
loop(ATS)
loop(Timer)
1) ATS: (running,2,[],())
Timer: (running,1,[],())
Console: (waiting,0,[in(setState)],(0))
*: 1/off
2) ATS: (running,1,[],())
Timer: (running,2,[timer(60)],())
Console: (waiting,0,[in(setState)],(0))
*: 1/off
in(ATS.wakeup)
loop(Timer)
loop(ATS)
out(Timer.tick)
3) ATS: (waiting,2,[wakeup],())
Timer: (running,1,[],())
Console: (waiting,0,[in(setState)],(0))
*: 1/off
4) ATS: (running,2,[],())
Timer: (running,2,[timer(60)],())
Console: (waiting,0,[in(setState)],(0))
*: 1/off
5) ATS: (running,1,[],())
Timer: (waiting,2,[timer(60),out(tick)],())
Console: (waiting,0,[in(setState)],(0))
*: 1/off
loop(Timer)
in(ATS.wakeup)
out(Timer.tick)
in(ATS.wakeup)
6) ATS: (waiting,2,[in(wakeup)],())
Timer: (running,2,[timer(60))],())
Console: (waiting,0,[in(setState)],(0))
*: 1/off
7) ATS: (running,2,[],())
Timer: (waiting,2,[timer(60),out(tick)],())
Console: (waiting,0,[in(setState)],(0))
*: 1/off
out(Timer.tick)
in(ATS.wakeup)
8) ATS: (running,3,[],())
Timer: (waiting,2,[timer(60)],())
Console: (waiting,0,[in(setState)],(0))
*: 1/off
out(ATS.warning)
9) ATS: (running,3,[proc(Console.setState)],())
Timer: (waiting,2,[timer(60)],())
Console: (taken,1,[proc(setState)],(0))
*: 1/off
in(Console.setState)
10) ATS: (running,4,[],())
Timer: (waiting,2,[timer(60)],())
Console: (waiting,0,[in(setState)],(1))
*: 1/off
select(ATS)
11) ATS: (waiting,4,[in(off),timer(6)],())
Timer: (waiting,2,[timer(60)],())
Console: (waiting,0,[in(setState)],(1))
*: 1/off
1/in(*.off)
1/in(!*.off)
12) ATS: (running,5,[in(*.off)],())
Timer: (waiting,2,[timer(59)],())
Console: (waiting,0,[in(setState)],(1))
*: 1/off
13) ATS: (waiting,4,[in(off),timer(5)],())
Timer: (waiting,2,[timer(59)],())
Console: (waiting,0,[in(setState)],(1))
*: 1/off
in(ATS.off)
1/in(*.off)
1/in(!*.off)
...
...
...

Fig. 8
PART OF THE LTS GRAPH FOR THE ALVIS MODEL SHOWN IN FIG. 7

Acknowledgement

The paper is supported by the Alvis Project funded from 2009-2010 resources for science as a research project.

References

[1] T. Murata, "Petri nets: Properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, 1989.
[2] K. Jensen and L. Kristensen, *Coloured Petri nets. Modelling and Validation of Concurrent Systems*. Heidelberg: Springer, 2009.
[3] M. Szpyrka and T. Szmuc, "Verification of automatic train protection systems with RTCP-nets," in *Computer Safety, Reliability and Security*, ser. LNCS, J. Górski, Ed. Springer-Verlag, 2006, vol. 4166, pp. 344–357.
[4] J. A. Bergstra, A. Ponse, and S. A. Smolka, Eds., *Handbook of Process Algebra*. Upper Saddle River, NJ, USA: Elsevier Science, 2001.
[5] L. Aceto, A. Ingófsdóttir, K. Larsen, and J. Srba, *Reactive Systems: Modelling, Specification and Verification*. Cambridge, UK: Cambridge University Press, 2007.
[6] M. Szpyrka, P. Matyasik, and R. Mrówka, "Alvis – modelling language for concurrent systems," in *Intelligent Decision Systems in Large-Scale Distributed Environments*, ser. Studies in Computational Intelligence, P. Bouvry, H. Gonzalez-Velez, and J. Kołodziej, Eds. Springer-Verlag, 2011, ch. 15.
[7] M. Szpyrka, *Alvis On-line Manual*, AGH Univ. of Science and Tech., 2011. [Online]. Available: http://fm.ia.agh.edu.pl/alvis:manual
[8] B. O'Sullivan, J. Goerzen, and D. Stewart, *Real World Haskell*. Sebastopol, CA, USA: O'Reilly Media, 2008.
[9] R. Milner, *Communication and Concurrency*. Prentice-Hall, 1989.
[10] K. Balicki and M. Szpyrka, "Formal definition of XCCS modelling language," *Fundamenta Informaticae*, vol. 93, no. 1-3, pp. 1–15, 2009.
[11] J. Barnes, *Programming in Ada 2005*. Addison Wesley, 2006.
[12] G. Berry, *The Esterel v5 Language Primer Version v5 91*, Centre de Mathématiques Appliquées Ecole des Mines and INRIA, 2000.
[13] *Welcome to SCADE 6.0*, Esterel Technologies SA, 2007.
[14] ISO, "Information processing systems, open systems interconnection LOTOS, Tech. Rep. ISO 8807, 1989.
[15] *OMG Systems Modeling Language (OMG SysML)*, Object Management Group, 2008.
[16] P. Ashenden, *The Designer's Guide to VHDL*, 3rd ed. Morgan Kaufmann, 2008, vol. 3.

DEVELOPMENT OF A LIBRARY FOR TEACHING AND IMPLEMENTING RESOURCE-LIMITED EMBEDDED SYSTEMS

William A. Stapleton
Ingram School of Engineering, Texas State University – San Marcos
601 University Drive, 5202 Roy F. Mitte Building, San Marcos, TX 78666 USA
wstapleton@txstate.edu, Voice: (512) 245-8746, FAX: (512) 245-7771
The 2011 International Conference on Embedded Systems and Applications

ABSTRACT

Texas State University – San Marcos began offering its Computer Engineering curriculum in the Fall 2010 semester. A key element of this curriculum is an emphasis on Embedded Systems design as a mechanism for addressing the key educational objectives of the IEEE/ACM Model Curriculum. One of the key differences between Embedded Systems design and general system design is the need to address a restrictive resource-limited environment. This paper describes the development of an embedded systems library suited for the dual purposes of teaching resource-limited Embedded Systems design and for implementation of practical Embedded Systems.

Keywords: embedded systems, engineering education

1. INTRODUCTION

In 2008, Texas State University-San Marcos (TSUSM) created the Ingram School of Engineering by moving the Manufacturing Engineering and Industrial Engineering programs from the Department of Technology and creating a new Electrical Engineering (EE) program [1]. Beginning in Fall 2010, The Ingram School of Engineering began offering a Computer Engineering (CompE) program as an option in the EE program. This CompE curriculum was developed using the IEEE/ACM Model Computer Engineering curriculum, implementation C [2-3]. By partnering with the Computer Science program, the EE program was able to offer all of the necessary coursework to create the CompE program only two years after offering EE courses for the first time at TSUSM.

In developing the CompE program, the TSUSM faculty considered how they would incorporate as many of the items from the Computer Engineering Body of Knowledge [3]. Building on a model developed at The University of Alabama, a curricular focus utilizing the theme of Embedded Systems was developed that covers a wide range of these knowledge objectives [4-9]. In brief, there are considerations in embedded systems design which translate well into general system design which do not translate as well in the other direction. In particular, many embedded systems labor under the constraints of very limited resources, e.g. small memory, slow processing speed, limited electrical power, and limited I/O bandwidth. Techniques that work with a MHz microprocessor with kibibytes of RAM will work with a GHz microprocessor with gibibytes of RAM but the reverse is often not true. Many modern general purpose system design tools such as object-oriented compilers and graphical user interfaces simply overwhelm the capabilities of embedded systems.

In order to address all of these concerns, the embedded systems design techniques are incorporated across multiple courses in the curriculum along with more general-purpose design techniques. During the incorporation of these techniques into the curriculum, a library of fundamental routines was developed. These routines are useful as a teaching tool as showing the students how they were developed is instructive of how to interact at a low level with the microprocessor hardware. These same routines are sufficiently versatile to serve as a set of building blocks for complex projects including capstone design and beyond.

2. SELECTING AN ARCHITECTURE

The IEEE/ACM Model Curriculum lists a number of Embedded Systems topics appropriate to a Computer Engineering Curriculum [3]. These include: history and overview; embedded microcontrollers; embedded programs; real-time operating systems; low-power computing; reliable system design; design methodologies; tool support; embedded multiprocessors; networked embedded systems; and interfacing and mixed-signal systems.

While developing the TSUSM CompE curriculum, the concept of incorporating these topics beginning as early as the Digital Logic and Microprocessors courses and continuing through the Capstone Design courses via the development and use of a core set of programming libraries came to fruition. The importance of and techniques for incorporating these topics has been previously studied [4-6,8-9].

In order to build a library, a microprocessor architecture needed to be selected. A prior-existing course, CS 2318: Assembly Language, uses the RISC-style MIPS architecture for development solely through a software emulation environment. A founding tenet of the Ingram School of Engineering is to provide strong “hands on” engineering whenever possible without compromising theoretical rigor [4,10]. Therefore, the decision was made to incorporate a physical architecture in EE 3420: Microprocessors and subsequent courses. Several were considered and the salient points for making a decision included a desire for a simple enough architecture that students could become fluent in its use in one semester, a readily-available and inexpensive hardware platform that students could easily purchase for themselves if desired, a readily-available toolset for both low-level assembly programming and high-level languages such as C/C++, and common usage in at least one industrial context.

The selected architecture was the FreeScale MC9S12 (formerly Motorola 68HC12) family [11-13]. This is an older 16-bit CISC-style architecture which is commonly used in automotive industry applications. The toolset for this family of microprocessors includes a number of freeware/shareware tools such as AsmIDE, MiniIDE, and GCC for 68HC11/68HC12 [14-16]. The CodeWarrior development platform has also been made available for MC9S12 [17]. Evaluation board architectures using the MC9S12 microprocessors are available from a number of vendors [11,13,18-19]. The TSUSM EE/CompE laboratories are equipped with Dragon12, MiniDragon, and DragonFly12 boards from Wytec but the libraries are suited for use with any MC9S12 products and have been tested with boards from Technological Arts and Axiom Manufacturing [13,18-19]. The libraries are similarly compatible with either the older D-BUG12 monitor or newer Serial Monitor (via uBug12 or CodeWarrior) on any of the MC9S12 boards.

3. LIBRARY CONTENTS AND EMBEDDED SYSTEMS TOPICS

The library incorporates code examples and subroutines intended to both address the Model Curriculum topics and to serve as useful building blocks for student projects. When designing for general-purpose computers, having hardware and operating system support for a common set of input/output (I/O) devices, *e.g.* keyboard, mouse, VGA screen, printers, etc. Embedded systems designers cannot expect the same standardized hardware.

The first routines included in the library are used to establish input from a 4x4 grid of switches used as a touch-tone telephone style keypad and interfaced via 4 row pins and 4 column pins. The library routines allow for polling of the keypad or, if the selected I/O pins allow, for interrupt-driven key action. The keypad routines allow for custom mapping of ASCII characters to each key. Similar routines allow individual switches to be attached to any I/O pin and monitored.

The next routines included in the library are used to allow output to either simple indicator LEDs or alphanumeric LED displays. Because using multiple digit displays (even simple 7-segment displays) may quickly allocate all I/O pins on a microprocessor, the routines incorporate time-multiplexing of multiple 7-segment displays. Two versions of the routine exist differing by triggering the digit multiplexing may by either an active delay loop or via a hardware-timed interrupt.

The next routines allow the use of standard character-based LCD panels up to 4 rows of 40 characters each. The panel may be interfaced using either a byte-wide data interface or a 4-bit wide interface directly connected from the microprocessor I/O pins to the panel.

In addition to supporting direct attachment of devices to the microprocessor I/O pins, the library contains routines for support of standard interface types. The first interface supported is the RS-232 serial standard (called Serial Communication Interface or SCI by FreeScale). This is an asynchronous, peer-to-peer standard with no addressing. The subroutine support for SCI includes transmission and receipt of single bytes similar in form to the *putc*( ) and *getc*( ) routines in C. Extended routines allowing for null-terminated strings of bytes similar to the C *puts*( ) and *gets*( ) via SCI are also included. Because SCI I/O speeds are relatively slow, support routines allow for non-blocking tests of the transmit and receive buffers to allow background I/O processing.

The most common interface for development of programs involves using a PC as a terminal for the microprocessor with data transmitted over RS-232. The most common interface model used as a terminal emulator on PCs is the VT-100 terminal. In addition to input and output of simple text, the VT-100 terminal model allows control of the terminal properties such as cursor position, text size/font, text color, and type variants such as boldface, italics, or blinking text.

A second standard interface type is Serial Peripheral Interconnect (SPI). SPI is a synchronous master-slave standard with a single select bit per slave for addressing. With SPI transmit and receive operations are always simultaneous so the library includes an SPI byte-exchange routine.

The basic SPI routines are also utilized to form software interfaces for several peripherals. One such interface in the library is for the LCD panels on the Axiom Manufacturing boards which physically access the LCD panel via a 74HC595 serial-to-parallel interface chip connected to the SPI port. Another device supported by the library is the MAXIM Integrated Circuits MAX3110E SPI to RS-232 UART bridge which provides an additional RS-232 UART to the microprocessor system [20]. The library supports this chip as one of multiple chips on the SPI bus for extending the I/O capabilities of a system.

Both the SCI and SPI interfaces are typically used for transfer of characters. Internally, all microprocessors treat data as binary numbers. The library contains routines for conversion between ASCII character data and binary formats. These are similar to the C *atoi*( ) and *itoa*( ) routines. The inherent data types on the MC9S12 family are 8-bit and 16-bit integers. The library routines add support for basic arithmetic for extended-length (32-bit, 64-bit, and arbitrary-length) integers and fixed-length fraction formats.

While the MC9S12 does not include floating-point data formats natively, the library offers support for simple floating point I/O and basic arithmetic via software. More comprehensive floating-point support is also provided in the library using the MicroMega µM-FPU floating-point unit [21]. The µM-FPU is a stand-alone IEEE 754 standard floating-point arithmetic chip interfaced via SPI. The library incorporates routines which use the aforementioned SPI functions to provide access to all of the floating-point arithmetic capability of the µM-FPU. The libraries also include a full set of macros which allow FPU access to appear as extensions to the existing assembly language syntax.

In addition to the simple point-to-point serial protocols, SCI and SPI, all of the MC9S12 variants used in the laboratories support two additional communications protocols used for multi-point communications, Inter-Integrated Circuit (IIC) and

Controller Area Networking (CAN). The library includes support for addressing and simple message passing using each of these protocols.

The IIC bus allows for multiple devices, up to 128, to share a single bus and was originally designed as the control bus for chip-to-chip communications on a single backplane for multimedia equipment. The IIC protocol allows any bus device to become the bus master then send or request data from any other device. The IIC-based components of the library allow for simple messages to be sent or received via IIC. The Dragon12 board contains a Dallas Semiconductor DS1302 Real Time Clock (RTC) interfaced via IIC and library utilizes the basic IIC routines to provide an interface for reading and writing the RTC.

The CAN bus is an automotive industry standard for noise-tolerant communications. This is a complete networking standard with message routing and other communications protocols components. The library provides the simplest case support for specifying a target address and sending a message or filtering incoming message to receive only those locally addressed. Complex CAN bus support is not included in the current version of the library.

The various members of the MC9S12 family have both simple EEPROM and FLASH EEPROM. Support for reading and writing these memory stores are are integrated into the library. Typically, when students first learn to program using wither D-BUG12 or the Serial Monitor, many of the system initialization tasks are completed by the monitor before a student’s program begins so that their programs always begin from a known, sane state. Students building programs to take control of a board from power-up for the first time rarely remember to include all of the setup details in their programs. The library includes setup routines for most of the common functional units in the MC9S12 to allow the students to quickly choose their initial setup easily.

One of the common microprocessor interfacing tasks is creating pulse-width modulated (PWM) signals for control of various devices. The MC9S12 chips contain hardware PWM units. The library includes functions to streamline setting the PWM parameters for applications such as servo motor control and tone generation. Specifically for tone generation, the library includes a table of the PWM periods for musical notes covering eight octaves.

The MC9S12 hardware includes several independent variable-length timers. The library incorporates routines for streamlining control of setting the times for desired periods. Accompanying this support in the library is support for pulse accumulation for applications such as tachometer speed feedback for motor control.

The MC9S12 chips contain integrated analog-to-digital (A/D) conversion. The library includes routines for setup and control of the A/D units and scaling of the digital samples back to a voltage or other desired quantity. The library contains support for using several common thermistors for reading ambient temperature. The MC9S12 chips do not include internal digital-to-analog (D/A) conversion. The library includes support for interfacing the Linear Technology LTC1661 dual D/A convertor via the SPI bus. These chips are available on the Dragon12 boards in the laboratories.

The final group of library support routines is for control of motors. There are library routines which allow for easy control of servo motors which follow the 50Hz, 5%-10% duty-cycle format common in radio-controlled hobby applications. There are library routines for control of reversible DC motors via H-bridge with PWM speed control in either clockwise or counter-clockwise rotation. There are routines for control of unidirectional DC motors with integrated tachometers (e.g. PC case fans) allowing for speed control. There are routines for control of both 4-phase unipolar stepper motors and 2-phase bipolar stepper motors in both single and half step modes.

4. INCORPORATING THE LIBRARY INTO STUDENT LEARNING EXPERIENCES

In EE 3420: Microprocessors, students are introduced to the MC9S12 architecture and the accompanying assembly language as well as an associated C compiler. As this microprocessor architecture is taught as one exemplar of a complete architecture complete with design compromises, the students are shown the libraries piecemeal after each portion of the library is recreated as class exercises. The accompanying laboratory assignments begin with relatively simple tasks such as creating a serial link over infrared between two boards and continue to include such things as motor control with feedback to control a fan speed using PWM as measured by a tachometer pulse in response to temperature measured by a thermocouple. The culminating experience in the class involves the students building simple robots which must follow lines on a track searching for objects to either collect or reject.

The other courses which have shown the most widespread usage of the library for applications are the two Capstone Design courses, EE4390 and EE4391. While the EE program is young enough to only have its third group of students enter the Capstone Design sequence, the library has already proved useful as a building block for several projects. The first Capstone Design class had three groups who used the library. The first group used the library on a MC9S12 microprocessor working in conjunction with a DSP board to create a powerline networking protocol capable of passing signals through multiple levels transformers between communicating nodes. The second group used the library on a MC9S12 board to build a controller for a press and oven for producing semiconductor ceramics. The third group using the library built a digital goniometer for the Physical Therapy program in the School of Nursing at TSUSM. The goniometer is used to make measurements of joint position for patient flexibility, record those measurements, and transmit them into the patient database The second Capstone Design class included several groups using the library. One group is building a camera mount which automatically tracks movement. A second group is building an energy harvesting monitor to report the condition of battery charge at a combined solar/wind generation station. A third group is building a self-organizing mesh communications network for the public transit system on campus. In the most recent Capstone Design class, one group is using the library to create a controller for an automated telescope mount. Another group is building an RFID-based asset inventory system. A third group is using the library to build a game controller for the Wii incorporated into a treadmill for wheelchairs which provides resistance and flexibility training for wheelchair physical therapy.

In addition to use in required coursework, two teams of students involved in the IEEE Student Branch at TSUSM have used the library for design of the control electronics in the Student Autonomous Vehicle competition sponsored by the IEEE Region 5. In the 2010 competition, which was the first time that TSUSM fielded a team, the participating students placed sixth in a field of twenty-seven.

5. FUTURE EXPANSION

The library currently supports the most common members of the FreeScale MC9S12 family found in evaluation boards. Part of the ongoing library creation and maintenance program involves adding support for additional features present in subsets of the MC9S12 family. A few members of the MC9S12 family include on-board Ethernet capability. Adding Ethernet support is one of the targeted additions to the library in the near future.

FreeScale has also released the MC9S12X microprocessor family which contains a secondary programmable logic core called the XGATE in addition to the standard MC9S12 core. Programming the XGATE coprocessor requires a very different programming model to the traditional MC9S12 core. Support for XGATE functionality is a long-term possibility for the library.

FreeScale has other microprocessor families for which similar libraries could be built. The Freescale MC9S12 family grew from the Motorola 6800 family. Similarly the FreeScale ColdFire family grew from the Motorola 68000 family. Not surprisingly, there are some strong parallels in the architectures and many of the techniques used to program one family should be readily adaptable to the other.

6. CONCLUSIONS

The Ingram School of Engineering at Texas State University – San Marcos has been in existence for three years and has offered its full complement of programs for two years. The Electrical Engineering program, including the Computer Engineering option, has grown from non-existent to over 270 students in just over two years. A key element in the coursework design for the EE and CompE curricula has been the incorporation of a strong Embedded Systems theme. Central to a cohesive Embedded Systems theme is a common library of tools used for both teaching of concepts and as an underpinning basis for student projects. This library has allowed students to attempt more extensive projects than they might otherwise have been able without such a resource.

7. REFERENCES

[1] Stapleton, William, “Zero to Two Hundred in Two Years: Launching a New Program”, The 2010 International Conference on Frontiers in Education: Computer Science and Computer Engineering, Las Vegas, NV, July 12-15, 2010.

[2] URL: http://www.computer.org/portal/web/education

[3] URL: http://www.eng.auburn.edu/ece/CCCE/CCCE-FinalReport-2004Dec12.pdf

[4] Stapleton, William A., “Embedded Assessment of Microcomputer Fundamentals for Embedded Systems Education”, The 2008 International Conference on Frontiers in Education: Computer Science and Computer Engineering, Las Vegas, NV, July 2008.

[5] Ricks, K.G., Jackson, D.J., Stapleton, W.A., “An Embedded Systems Curriculum Based on the IEEE/ACM Model Curriculum,” IEEE Transactions on Education, Vol. 51, Issue 2, pp. 2262-2270, (2008) ***– selected IEEE Education Society 2009 Best Transactions Paper***.

[6] Ricks, Kenneth G., David J Jackson, William A. Stapleton, “Incorporating Embedded Programming Skills within an ECE Curriculum”, SIGBED Review, Volume 4, No. 1, pp. 17-26 (2007).

[7] Kenneth G. Ricks, David J. Jackson, William A. Stapleton, “An Evaluation of the VME Architecture for Use in Embedded Systems Education,” SIGBED Review, Vol 2. No. 4, pp. 63-69, (2005).

[8] Ricks, K. G., Stapleton, W. A., Jackson, D. J., “A Focused Curriculum for Embedded Systems”, Proceedings of the 32nd Annual International Symposium on Computer Architecture (ISCA), Madison, Wisconsin, June 2005.

[9] Stapleton, William, Kenneth Ricks, Jeff Jackson, “Implementation of an Embedded Systems Curriculum”, Proceedings of the ISCA 20th International Conference on Computers and Their Applications (CATA’05), New Orleans, LA, March 2005.

[10] Stapleton, William, Stern, Harold, Asiabanpour, Bahram, Gourgey, Hannah, “A Novel Engineering Outreach to High School Education”, 39th Annual Frontiers in Education Conference (FIE 2009), San Antonio, TX, October 18-21, 2009.

[11] URL: http://www.freescale.com/webapp/sps/site/homepage.jsp?code=UNIVPROGRAM_HOME_CAT

[12] URL: http://www.freescale.com/webapp/sps/site/overview.jsp?code=UNIVPROGRAM_SLKITS_CAT

[13] URL: http://www.evbplus.com/

[14] URL: http://directory.fsf.org/project/asmide/

[15] URL: http://www.mgtek.com/miniide/

[16] URL: http://m68hc11.serveftp.org/

[17]
URL: http://cache.freescale.com/lgfiles/devsuites/HC12/CW_HC12_v5.1_SPECIAL.exe?WT_TYPE=IDE%20-%20Debug,%20Compile%20and%20Build%20Tools&WT_VENDOR=FREESCALE&WT_FILE_FORMAT=exe&WT_ASSET=Downloads

[18] URL: http://axman.com/

[19] URL: http://technologicalarts.com/

[20]
URL: http://datasheets.maxim-ic.com/en/ds/MAX3110E-MAX3111E.pdf

[21] URL: http://micromegacorp.com/umfpu-v3.html

Generating Hardware from Java Using Self-Propagating Flowpaths

Darrin M. Hanna, Bryant Jones, Lincoln Lorenz, and Mark Bowers

Abstract—**The performance of software executed on a microprocessor is adversely affected by the basic fetch-execute cycle. A further performance penalty results from the load-execute-store paradigm associated with the use of local variables in most high-level languages. Implementing a software algorithm directly in hardware such as on an FPGA can alleviate these performance penalties. Such implementations are normally developed in a hardware description language such as VHDL or Verilog. Previous work has been completed to create a compiler for converting high-level stack-based languages to VHDL for use on an FPGA or ASIC. This allowed for special-purpose processors to be generated efficiently from high-level algorithms with minimal design time. Introduced in this paper is a significant optimization to the original flowpaths – we have completely eliminated the controller and modified all operations to control themselves. These new self-propagating flowpaths execute faster and are less resource intensive. Comparisons to previous examples show that the new design exhibits, on average, a decrease in execution time of 32%, operating frequencies of 1.6 times higher, and a 33% decrease in power consumption. These flowpaths can be generated from languages with a stack-based intermediate representation including Java, C++, C#, and VB.**

Index Terms—**Field Programmable Gate Arrays, Program Compilers, Embedded Systems, Object-oriented Design.**

I. Introduction

Over the past ten years, field-programmable gate arrays (FPGAs) have become increasingly popular in the area of embedded systems. Due to lower costs and an increase in the resources available with lower-end models, FPGAs can be used in a wide range of applications. FPGAs have shown to be optimal for use in high-performance systems while reducing power consumption.

A special-purpose processor (SPP) or custom digital circuit implemented on an FPGA is an ideal replacement for a microcontroller. Custom hardware such as SPPs can realize an algorithm more efficiently than a general-purpose microcontroller with load-execute-store overhead. However, SPPs increase in size, requiring more logic for larger algorithms, while a microcontroller can execute as large an algorithm as the program memory can hold using a fixed amount of logic. For others, a SPP on an FPGA can be used as a coprocessor to a microcontroller to help speed up particular functions or sub-procedures.

Manuscript received March 31, 2011.

Darrin M. Hanna, Bryant Jones, Lincoln Lorenz, and Mark Bowers are with the School of Engineering and Computer Science at Oakland University in Rochester, MI (dmhanna@oakland.edu).

Designing a SPP is much more difficult than writing software in a high-level programming language to execute on a microcontroller. In order to design a SPP for a particular algorithm, a designer must learn how to develop hardware using a hardware description language such as VHDL or Verilog. Further, design of a SPP for a lengthy algorithm can be time consuming and requires a skilled computer engineer to do so efficiently.

One common idea to decrease development time is to use a high-level language to develop hardware. Several techniques have been introduced that use this concept. Techniques for generating SPPs such as Handel-C, often require learning a new or significantly altered language, and have the bottleneck of often being register based as described in [1].

Using the method introduced in [1, 3] SPPs can be generated from algorithms written in high-level stack-based intermediate representations (IR). This has the advantage of being generated from an unmodified high-level language. This is also more efficient than previous methods that use registers for each variable. The SPPs generated using this technique are called flowpaths. An embedded system can be designed and implemented rapidly using a high-level programming language.

Our previous method generated flowpath SPPs with two basic components, a datapath and a state controller. An optimization of this architecture is to distribute the controller into each low-level operation to allow for smaller, more efficient designs that can operate at higher frequencies.

In this paper, an optimization of the flowpaths architecture is introduced using a stateless self-propagating method that results in improvements for both speed and chip utilization. Outlined in Section 2 is the new stateless self-propagating architecture. Section 3 shows results using several benchmarking algorithms, comparing efficiency in an embedded system. Sections 4 and 5 describe additional benefits of flowpaths. The paper closes by providing concluding remarks and future work.

II. Self-Propagating Flowpaths

Software programs written in a stack-based language can be converted directly to circuits called flowpaths [3]. Stack-based programming languages inherently minimize the use of local variables. This is in contrast to other methods that have been developed for converting register-based code into circuits by converting each variable into a register and each assignment and access into a sequential operation. Those methods suffer from fan-out and routing issues and therefore operate at lower clock rates [1]. Several software-programming languages compile to an intermediate representation (IR) that is stack-based. Examples include the stack-based Java bytecode which is compiled from Java, and the Common Interface Language (CIL) that languages compatible with the .NET framework compile to, such as C++, C#, VB, and J#. Any of these languages could be represented similarly in hardware. The flowpaths compiler described here currently uses Java. The Java Virtual Machine (JVM) is a stack machine that runs on a microprocessor and executes Java bytecodes. Therefore, instead of executing bytecodes on a JVM, flowpaths completely eliminate the JVM by creating custom hardware that implements the program. Java bytecodes are translated to hardware operations, also known as OPs. A function-call within a Java program translates to a datapath which contains a series of connected OPs. Since flowpaths is IR-based, the generated hardware is represented in a human-readable way, unlike similar tools which often generate hardware that is obscure and difficult to modify.

Self-propagating (SP) flowpaths uses a system of cascading enables to avoid the need for an overall state controller. The nature of the hardware generated is such that algorithms or parts of algorithms execute in sequence where one operation after another is active. In this scheme, no overall controller is necessary; no single, overarching entity requires knowledge of every operation in the datapath. Rather, this knowledge is intrinsic to the individual OPs, and is therefore distributed.

A. Control Signals

Each OP is triggered by an "enable" signal, and its completion is conveyed with a "done" signal. Self-propagation is achieved by wiring the done signal from a given OP to the enable signal of the successive OP. An initial enable pulse to the system starts the cascading enables. The changing execution stack and the locals stack flow alongside this cascading status. Both combinational and sequential operations adhere to this format. The overall architecture including the new control signals is explained in Fig 1 using an example of a greatest common divisor (GCD) algorithm. A simplified flowpath to compute the GCD requires three OPs: an equality detector (OPEq), a magnitude comparator (OPLt), and a subtraction OP (OPMinus). A path using multiplexers and branches connects these OPs. The top of Fig 1 shows the GCD with the original flowpath including the state controller, and the bottom of Fig 1 shows the SP flowpath with cascading enables.

Conditional branching was previously controlled using the main state controller. Two boolean results were received from the conditional OP to notify the controller which OP should be enabled next. In this new architecture, conditional OPs simply produce two done signals, representing two paths the flow could take. When the two paths converge again, a multiplexer

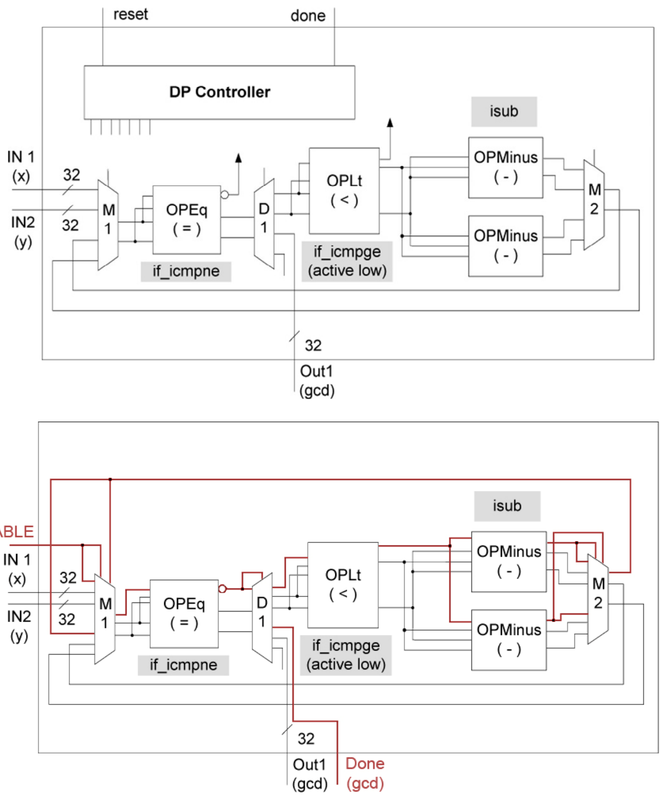


Fig 1. Comparison of the original controller (top) and SP (bottom) flowpath architectures

is used to select the appropriate flow to propagate onward. A one-hot select line is used for the multiplexer, which is driven by the done control signal output by the last OP in the active path. This is demonstrated in Fig 2.

Software loops, such as those generated by the *while* and *for* statements in Java, are very similar to conditional branches, the difference being that they contain an unconditional branch at the bottom of the statement. Unconditional branches are simply represented as connections between two OPs. If there is a conditional check within a loop there is a possibility that three paths will flow to a multiplexer: an initial entrance path, a loop condition path, and the conditional OP path.

B. Memory

Memory operations simply assume control of the memory when activated. Currently, since only one OP in a given

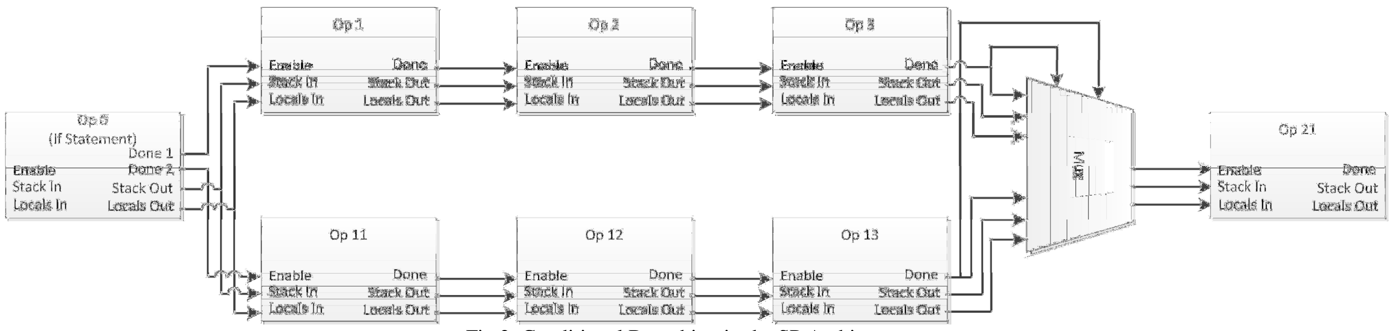


Fig 2. Conditional Branching in the SP Architecture

datapath is active at any one time, no memory arbitration scheme is necessary. Every memory OP within a given datapath is multiplexed to the memory controller. If every memory OP were directly multiplexed into the memory controller, problems would occur with routing as the design increased in complexity. In the case of a method call, OPs are multiplexed within the datapath itself and a single set of memory control signals are routed to the datapath that calls it. The top method of the hierarchy is wired directly to the memory controller. Multiplexing within a single datapath occurs through a sub-multiplexing routine where operations are multiplexed into groups before the final multiplexing stage to the single output. Relative to a datapath, a sub-method call with memory appears as a normal OP with memory. In order to support multithreaded designs with parallel flowpaths, a memory arbiter is needed.

III. Results

Several examples varying in difficulty were tested to verify functionality and performance relative to both the original flowpaths architecture and a microcontroller-based embedded system. Euclid's greatest common divisor (GCD), a quicksort, the Sieve of Eratosthenes, a complex FFT, Linpack, and the Mandelbrot fractal were tested. The GCD is a small algorithm with relatively simple constructs, such as branching, subtracting, comparing, and method calls. The Sieve of Eratosthenes and quicksort both require the use of memory, with quicksort requiring the most. FFT, Linpack, and the Mandelbrot fractal use fixed-point arithmetic.

The flowpaths produced by the compiler have been experimentally verified by simulation in Xilinx ISE version 12.3. Since all of the algorithms were written in standard Java, it is easy to verify functionality. Additionally, since the process does not alter the Java language, the exact same source code is compiled on every platform. This adds to the relevancy of performance comparisons and it aids in debugging.

Euclid's GCD was compared in both architectures of the flowpath compiler and the jStamp j-80 [4], a custom architecture that natively executes Java bytecodes at 73.7 MHz. The GCD of the values 12,365,400 and 906 was calculated. Table I displays the results of the implementation using a Xilinx Spartan 6 XC6SLX75. The new architecture showed a large decrease in the number of clock cycles necessary, along with a significant increase in the maximum execution frequency. In the original design, the algorithm required 112 slices, and the SP design only required 77 slices of the device.

Method	*Data Bus*	*Clock Cycles*	*Time(ms) @ Max Freq*	*Max Freq (MHz)*	*Energy (mW·ms) @Max*
Original Flowpath	32 bit	95,648	0.637	150	148
SP Flowpath	32 bit	54,652	0.273	200	54
jStamp	32 bit	2,690,000	36.5	73.7	6862

Table I. Relative performance comparison of GCD

The Sieve of Eratosthenes algorithm for finding all of the prime numbers less than 2048 was executed using several different methods to provide a relative performance comparison. The algorithm was compiled to a flowpath using both the original and SP architectures, and the jStamp. Results comparing the architectures are provided in Table II. Both flowpaths targeted a Xilinx Spartan6 XC6SLX75. The original flowpath generated hardware that requires 496 slices, occupying 4% of the chip. The SP flowpath, however, only requires 268 slices at 2% consumption. This space savings is expected as a result of the removal of the state controller.

Method	*Data Bus*	*Clock Cycles*	*Time(ms) @ Max Freq*	*Max Freq (MHz)*	*Energy (mW·ms) @Max*
Original Flowpath	16 bit	22,116	0.211	105	29
SP Flowpath	16 bit	16,023	0.1282	125	12
jStamp	32 bit	943,000	12.8	73.7	2406

Table II. Relative performance comparison of Sieve of Eratosthenes

A comparison of the two compilers was also done for quicksort using an identical series of 4000 random data values. The algorithm used was an iterative version, since recursion is not yet supported in flowpaths. Both designs were implemented using the same Xilinx Spartan6 XC6SLX75. Implementation results comparing the architectures are provided in Table III.

Method	*Data Bus*	*Clock Cycles*	*Time(ms) @ Max Freq*	*Max Freq (MHz)*	*Energy (mW·ms) @Max*
Original Flowpath	16 bit	659,671	13.19	50	1279
SP Flowpath	16 bit	486,520	3.892	125	366
jStamp	32 bit	37,520,000	509.1	73.7	95,711

Table III. Relative performance comparison of QSort

To demonstrate the quick design prototyping capabilities of a flowpath, a 1024-point complex FFT was created in Java. Implementing the same algorithm in hardware would take a considerable amount of time and expertise. Using flowpaths, a moderately efficient FFT implementation can be created for use in an embedded system. This was implemented targeting a Xilinx Spartan 6 XC6SLX75T FPGA. The generated hardware required 8,349 slices, utilizing 71% of the chip. The same Java algorithm was implemented on a jStamp embedded processor. The FFT utilized a 32-bit fixed-point notation for computations. Results are summarized in Table IV. In comparison to the jStamp, the flowpath FFT showed superior performance. Since the FFT algorithm can be effectively parallelized, the serial version generated by the compiler is not expected to achieve optimal results.

Method	*Data Bus*	*Clock Cycles*	*Time(ms) @ Max Freq*	*Max Freq (MHz)*	*Energy (mW·ms) @Max*
SP Flowpath	32 bit	268,891	3.841	70	714
jStamp	32 bit	17,500,000	237.6	73.7	44,669

Table IV. Relative performance comparison of 1024-point complex FFT

The classic benchmarking algorithm, Linpack, which computes the solution to a system of linear equations, was generated using the flowpath compiler. Implementing the same hardware in custom VHDL would be considerably expensive in terms of time and expertise. This generated flowpath was implemented targeting a Xilinx Spartan 6 XC6SLX150T FPGA. The hardware required 15,632 slices, utilizing 67% of the chip. The flowpath was compiled using 32-bit fixed-point notation for the computations, and the results are shown in Table V. Times are given for the solution of a linear system of size 100x100. As shown with the FFT algorithm, Linpack showed an extreme performance increase in comparison to the jStamp equivalent.

Method	*Data Bus*	*Clock Cycles*	*Time(ms) @ Max Freq*	*Max Freq (MHz)*	*Energy (mW·ms) @Max*
SP Flowpath	32 bit	4,276,400	122.2	35	34,065
jStamp	32 bit	2.064×10^8	2800.0	73.7	526,400

Table V. Relative performance comparison of Linpack

The Mandelbrot set is a fractal image produced by iterating a quadratic polynomial across points in the complex plane. A Mandelbrot calculation unit was generated using the flowpath compiler. This unit operates upon a point in the Mandelbrot set until the exit conditions for that particular point are reached. The generated flowpath was implemented targeting an Altera Cyclone II EP2C35F672C6 FPGA. The flowpath was compiled using a 32-bit fixed-point number system. This was profiled against both a custom VHDL component written by hand, and the same Java code running in a jStamp.

The generated flowpath consumed 3217 logic elements (9% of the chip), while the custom VHDL component consumed 725 logic elements (2% of the chip). Performance results for calculating a single point in the Mandelbrot set for 255 iterations are summarized in Table VI.

The flowpath version is on the same order-of-magnitude as the custom VHDL version, in terms of speed, power consumption and resource usage. Additionally, since the flowpath was generated directly from Java, the development time was substantially faster. Both hardware versions greatly exceeded the performance of the jStamp processor.

Method	*Data Bus*	*Clock Cycles*	*Time(ms) @ Max Freq*	*Max Freq (MHz)*	*Energy (mW·ms) @Max*
SP Flowpath	32 bit	6,893	0.0656	105	15.55
Custom VHDL	32 bit	1,276	0.0111	115	2.02
jStamp	32 bit	199,000	2.7	73.7	319

Table VI. Relative performance comparison of Mandelbrot calculation

Ratiometric comparisons between original flowpaths and SP flowpaths were calculated and are shown in Table VII. Examples shown are GCD, QSort, and Sieve. Data was unavailable for other examples.

Experiment	*Time*	*Max Freq*	*Energy*
GCD	0.45	1.3	0.36
Sieve	0.61	1.19	0.41
Qsort	0.29	2.5	0.29

Table VII. Ratiometric comparison of SP flowpaths vs. original flowpaths

IV. Options for further space reduction

Occasionally, a hardware designer may approach size limitations with a specific design. Depending on how large the design is, several choices can be considered to work around such an issue. Choices may include: exploring optimization methods, inserting a soft-core processor, dynamic reconfiguration, or partitioning a device over multiple FPGAs. One space optimization is to remove repetitive OPs in a datapath. A series of similar OPs could also be considered redundant in terms of space utilization. Another optimization to save space would be to insert an elastic processor capable of computing the OPs needed, in their place. Elastic cores are also ideal for complex pieces of an algorithm that are executed relatively few times. Using SP flowpaths, a design could be easily partitioned by splitting the flowpath into sections and using the simple interface to every OP as a bus to an adjacent FPGA.

V. Design Flexibility

Flowpaths have the capability to make use of custom hand-crafted VHDL blocks for use into the flowpath. An interface can be created in Java to describe the custom VHDL component. The Java method would be empty and only defines the inputs and output of the block. The compiler will recognize this as a custom block and insert it into the generated flowpath. This can be used to define the interconnection to multiple custom VHDL blocks. This concept is further described in the paper [5].

VI. Future Work

Future work includes defining a metric for determining and minimizing the critical delay path of the system. Improved optimization techniques for enhancing the speed and reducing the size of the flowpaths generated by the compiler are also being explored. These optimizations include hardware component reuse and further reduction in unnecessary clock cycles through optimization of the operations and memory usage. Furthermore, future work includes integrating CIL, another stack-based IR, to the compiler, allowing for a wide range of .NET languages to be compiled to hardware.

VII. Conclusion

This paper shows how standard stack-based programs, such as Java bytecodes, can be compiled directly to flowpaths without a centralized controller. A refined architecture was introduced here that demonstrates improved efficiency in the areas of execution speed, maximum clock frequency, power dissipation, and the amount of logic used. Using this methodology, not only is the performance increased, but also the development time is significantly decreased.

Flowpaths can outperform microprocessors at lower clock frequencies and therefore consume less energy than microprocessors or microprocessor cores. Even in situations where the FPGA requires power on the same order of magnitude as a processor, the energy required to perform a function is significantly less since special-purpose processors, including flowpaths, greatly reduce the execution time and number of clock cycles required. Energy consumption is compared in the last column of each table in Section 3. On average, flowpaths running on FPGAs consumed over 50 times less energy than a Java microcontroller.

The space reduction and performance increase of SP flowpaths makes generation of SPPs for complex algorithms such as FFT or Linpack practical to implement in an embedded hardware system. Highly complex algorithms implemented in flowpaths have shown to be superior to an identical algorithm executed on a jStamp embedded processor. Furthermore, designs can be created easily with minimal design time, and the resulting hardware is easily understandable and modifiable by a hardware designer.

References

1. D. M. Hanna and R. E. Haskell, “Flowpaths: Compiling Stack-Based IR to Hardware,” Microprocessors and Microsystems, vol. 30, pp. 125 - 136, 2006.
2. D. M. Hanna and M. Duchene, "Executing Large Algorithms on Low-Capacity FPGAs using Algorithm Partitioning and Runtime Reconfiguration," Journal of Microprocessors and Microsystems, vol 31/5 pp 302-312, August 1, 2007.
3. D. M. Hanna, M. Duchene, L. Kennedy, and B. Carpenter, "A Compiler to Generate Hardware from Java Byte Codes for High Performance, Low Energy Embedded Systems," The 2007 International Conference on Engineering of Reconfigurable Systems and Algorithms, Las Vegas, NV, June 25 - 28, 2007.
4. Systronix, “JStamp: Real-time Native Java Module,” 2003.
5. D. M. Hanna, B. Jones, L. Lorenz, and M. Bowers, “Flexible Embedded System Design Using Flowpaths,” Submitted to the International Conference on Embedded Systems and Applications, 2011.

Using Real Time Java for Evaluating the Performance of DDS Platforms

Rojdi Rekik[1] **and Salem Hasnaoui**[2]

[1] Sys-Com Laboratory, ENIT University, Tunis, TUNISIA

[2] Sys-Com Laboratory, ENIT University, Tunis, TUNISIA

Abstract- *It is necessary to develop QoS for DDS platforms that enable participants to communicate by publishing the information they have and subscribing to the information they need in a timely manner. Since there is little existing evaluation of the ability of these platforms to meet the performance needs of DRE information management, this paper provides two contributions: (1) it describes three metrics choose for Evaluating the performance of DDS platforms (2) it describes the exploitation of Real Time Java to inject theses metrics in life cycle of DDS component to be evaluating. Our strategy is available to be developing for other Real Time middleware.*

Keywords: Information Management in DRE Systems; QoS-enabled Publish/Subscribe Platforms; Data Distribution Service; RTSJ;

1. Introduction

Data Distribution Service (DDS) are data-centric QoS-enabled publish/subscribe middleware responsible to ensure the right information gets to the right place at the right time by satisfying end-to-end quality of service (QoS) requirements, such as latency, throughput, dependability, and scalability.

Service-Oriented Architecture (SOA) middleware platforms not guarantee theses QoS mechanisms because to their lack of support for data-centric. For example, the Java Messaging Service is a SOA middleware platform that is not adapted for Distributed Real Time Environment (DRE) due to its limited QoS support, lack of real-time operating system integration, and high time/space overhead.

Real-time CORBA [7] is poorly adapted Publish/Subscribe communication due to excessive layering, extra time/space overhead, and inflexible QoS policies.

The Object Management Group (OMG) has adopted the Data Distribution Service (DDS) [5] specification to be applied for DRE systems. DDS is a standard for QoS-enabled pub/sub communication aimed at mission-critical DRE systems. It is designed to provide (1) location independence via anonymous pub/sub protocols that enable communication between collocated or remote publishers and subscribers, (2) scalability by supporting large numbers of topics, data readers, and data writers, and (3) platform portability and interoperability via standard inter faces and transport protocols. Ling unmanned vehicle communication with their ground stations, and in semiconductor fabrication devices.

This paper describe choose and implementing of metrics used to evaluate the performance of DDS middleware.

The remainder of this paper is organized as follows: Section 2 summarizes the DDS specification; Section 3 describes choose of metrics; Section 4 summarizes Real Time characteristic of RTSJ; Section 5 describes use of Real-Time Java to implement these metrics; Section 6 resume our future directions.

2. Overview of DDS

DDS is the Data-Centric Publish-Subscribe (DCPS) model, whose specification defines standard interfaces that enable applications to write/read data to/from a global data space in a DRE system. To sharing information with others, the applications can use global data space to publish data belonging one or more topics of interest to participants. The DCPS model decouples the declaration of information access intent from the information access itself, thereby enabling the DDS middleware to support and optimize QoS-enabled communication [1].

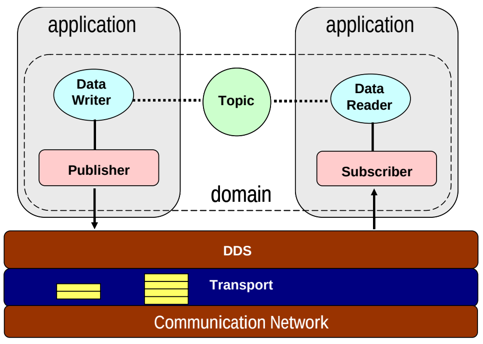


Figure 1: Architecture of DDS

When we create a DCPS DDS application, the following DDS entities are involved, as shown in Figure 1:

- **Domain** – DDS applications send and receive data within a domain, only participants within the same domain can communicate.

- **Domain Participant** - It's the entity responsible to factory, container, and manager the publishers and subscribers in the domain.
- **Data Writer and Publisher** – Data writers publish data values to the global data space of a domain. A publisher is created by a domain participant and used as a factory to create and manage a group of data writers with similar behavior or QoS policies.
- **Subscriber and Data Reader** – Data readers receive data. A subscriber is created by a domain participant and used as a factory to create and manage data readers. A data reader can obtain its subscribed data via two approaches: listener-based, which provides an asynchronous mechanism to obtain data via callbacks in a separate thread that does not block the main application and wait-based, which provides a synchronous mechanism that blocks the application until a designated condition is met.
- **Topic** – A topic connects a data writer with a data reader: communication happens only if the topic published by a data writer matches a topic subscribed to by a data reader. Publishers and subscribers need not be concerned with how topics are created nor who is writing/reading them since the DDS DCPS middleware manages these issues [2].

DDS supports many QoS properties, such as

- The degree and scope of coherency for information updates, i.e., whether a group of updates can be received as a unit and in the order in which they were sent.
- The frequency of information updates, i.e., the rate at which updated values are sent and/or received.
- The maximum latency of data delivery, i.e., a bound on the acceptable interval between the time data is sent and received
- The priority of data delivery, i.e., the priority used by the underlying transport to deliver the data.
- The reliability of data delivery, i.e., whether missed deliveries will be retried.
- How to arbitrate simultaneous modifications to shared data by multiple writers, i.e., to determine which modification to apply.
- Mechanisms to assert and determine liveliness, i.e., whether or not a publish-related entity is active.
- Parameters for filtering by data receivers, i.e., predicates which determine which data values are accepted or rejected.
- The duration of data validity, i.e., the specification of an expiration time for data to avoid delivering "stale" data.
- The depth of the 'history' included in updates, i.e., how many prior updates will be available at any time, e.g., 'only the most recent update,' 'the last n updates,' or 'all prior updates' [3].

These parameters can be configured at various levels of granularity (i.e., topics, publishers, data writers, subscribers, and data readers), thereby allowing application developers to construct customized contracts based on the specific QoS requirements of individual entities. Since the identity of publishers and subscribers are unknown to each other, the DDS DCPS middleware is responsible for determining whether QoS parameters offered by a publisher are compatible with those required by a subscriber, allowing data distribution only when compatibility is satisfied.

DDS enables clients to subscribe to meta-events that they can use to detect dynamic changes in network topology, membership, and QoS levels. This mechanism helps DRE information management systems adapt to environments that are continuously changing [4].

3. Metrics for DDS Platforms Evaluation

This section describes our methodology for evaluating DDS platforms. To be attending our goal, we are chosen three metrics in accordance with real-time and embedded characteristics of DDS middleware;

3.1. Data Age

For each data value, Data Age is the difference between the consumption date of this instance by the Data Reader and the creation date by Data Writer. Data Age must be less than lifespan to ensure consume data by the Data Reader before they expire.

3.2. Delivery Rate

Suppose the publishers have published k Data Objects in the super-period, denoted by d1, d2… dk. For a Data Object di, let the number of Data Readers interested in it be tsi, and the number of Data Readers that receive it before the deadline be dsi. We can define a metric called delivery rate of the system as follows:

$$\frac{\sum_{i=1}^{k} ds_i}{\sum_{i=1}^{k} ts_i}$$

This metric shows consumer satisfaction.

3.3. Completion Time

Completion time represents the difference between the point of time in which the Data Reader's execution cycle takes end and the release time of the Data Reader.

$$CT(DR) - R(DR)$$

Completion time must be less than deadline.

4. Real-Time Java Timing

Time is the essence of real-time systems and the basic requirement of the Real-time-Specification for Java (RTSJ). Time precision (millisecond, nanosecond) may vary with hardware present on the system. Temporal constraints are presented with the release parameters classes that contain relative time and absolute time attributes.

The specification also mandates that time objects must be constructed from other time objects or from time values, and the addition and subtraction of both objects and time values and comparison (using the Comparable-interface) must be supported.

4.1. High Resolution Time

RTSJ specifications define a few classes used to manage the time with very important precision. HighResolutionTime is the base class for AbsoluteTime, RelativeTime.

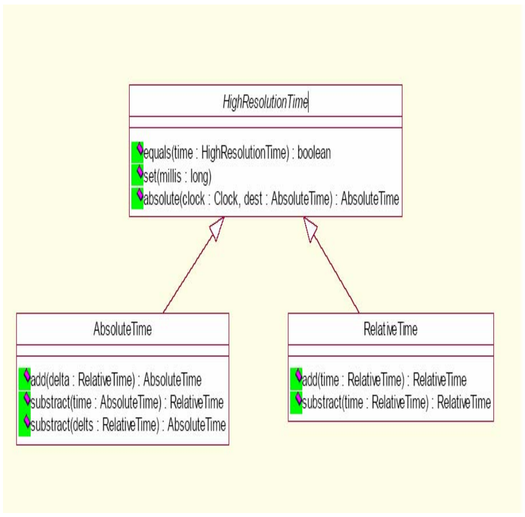

HighResolutionTime
equals(time : HighResolutionTime) : boolean
set(millis : long)
absolute(clock : Clock, dest : AbsoluteTime) : AbsoluteTime
AbsoluteTime
add(delta : RelativeTime) : AbsoluteTime
substract(time : AbsoluteTime) : RelativeTime
substract(delts : RelativeTime) : AbsoluteTime
RelativeTime
add(time : RelativeTime) : RelativeTime
substract(time : RelativeTime) : RelativeTime

Figure 2: RT-JAVA Timing

There is two type of High Resolution Time: Absolute Time and Relative Time; Absolute Time is the time exactly (example 10:44:55), Relative time is a rounded basically or duration time. RTSJ offer a few methods that are permit to add and subtract time.

4.2. Release Parameters

ReleaseParameters is the top-level class for release characteristics of schedulable objects.

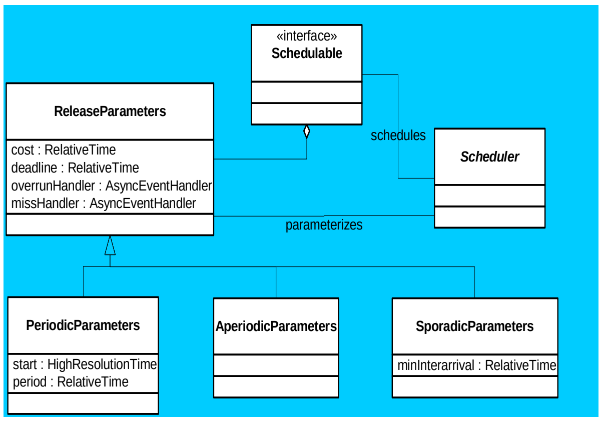


Figure 3: Temporal Parameter

Release parameters use HighResolutionTime values for cost, and deadline. Since the times are expressed as a HighResolutionTime values, these values use accurate timers with nanosecond granularity. The actual resolution available and even the quantity the timers measure depend on the clock associated with each time value.

5. Using RT-JAVA to inject metrics in DDS code

This section describes our chosen methodology to evaluate DDS middleware component. We choose to calculate metrics values with RT-JAVA to enjoy the precision and available algorithms in the language. We implement a real time clock and we inject it in the source code of DDS component. This Real Time Clock assists this DDS component in their life cycle.

```
public class RealTimeClock extends
javax.realtime.RealtimeThread
 {

   private String name;
   private AbsoluteTime startTime;
   private RelativeTime remainingTime;
   public Clock myClock;
   private boolean counting;
   private boolean go;
   private RelativeTime tick;
   public RealTimeClock (AbsoluteTime at,RelativeTime
countDown,int delay) {
super();
startTime = at;
remainingTime = countDown;
 myClock = Clock.getRealtimeClock();
counting = true;
go = false;
tick = new RelativeTime(delay,0);
                              }
…}
```

Figure 4: Attributes and Constructor of RealTime-Clock class

The attributes presented in Figure 4 are necessary to realize our real time counter, the start time, remaining time, System clock, and the delay of each execution period are initialized in the constructor. If counting equals true our counter started, if go equals true the execution time is completed.
A few methods are detailed in figure 5, this methods permit to consult a period, a current time and to reset, stop and restart counter.

```
public RelativeTime getResolution(){
                    return tick;
}
public synchronized AbsoluteTime getCurrentLaunchTime(){
return new
AbsoluteTime(myClock.getTime().add(remainingTime));
                              }
public synchronized void stopCountDown(){
                    counting = false;
          }
public synchronized void restartCountDown(){
                    counting = true;
                    notifyAll();
          }
public synchronized void resetCountDown(RelativeTime to){
remainingTime = to;
}
```

Figure 5: Getters and Setters methods

Run method are developed in Figure 6

```
public void run(){

try   {

synchronized(this) {

while(myClock.getTime().compareTo(startTime) < 0)
{
System.out.println("waiting:!
myClock.getTime().compareTo(startTime) < 0 "+name);
HighResolutionTime.waitForObject(this, startTime);
// wait for next period
  }
 System.out.println("---------");

while(remainingTime.getMilliseconds() > 0) {
System.out.print("waiting!remainingTime.getMillisecond
s() > 0"+name);
System.out.println(remainingTime.getMilliseconds());
////remaining time is shown
 while(!counting){
     wait();
                              }
  HighResolutionTime.waitForObject(this, tick);
 remainingTime.set(remainingTime.getMilliseconds()-
tick.getMilliseconds()); // update remaining time
          }
          go = true;
        notifyAll();
                              }
          }
catch(InterruptedException ie) {
          System.out.println("failer to manage system
time");
          }
          }
```

Figure 6: Run Method

The source code of run method explain that the real time counter defined in real time thread arrive to calculate remaining time, wait next period and update the attributes in class if necessary to conserve a correct functionality.

5.1. Data Age metric

Our real time counter must be injected in a source code of Data entity from DDS to manage life cycle.

```
public class DataImpl extends javax.realtime.RealtimeThread
implements Data{

          SchedulingParameters   sched;
          ReleaseParameters release;
          RealTimeClock   clockData;
          private MemoryCalc memory;
          private AbsoluteTime thisTime;
          private AbsoluteTime endTime;
          private RelativeTime dataAge;;


public DataImpl(….)
{
clockData = new RealTimeClock
(release.getStart(),release.getPeriod(),release.getDeadline());

}


public void run(){
…..
clockData.run();
thisTime=clockData.myClock.getTime();
dataAge= thisTime.subtract(memory.getTimestamp());

}
```

Figure 7: DataImpl class

Through the temporal parameters of RT-Java, we can calculate the Data Age. ClockData is creating in a constructor of DataImpl so they will have a same life cycle. Data Age is a subtract of current time and the creation time of data from Data Writer (return by getTimestamp() method).

5.2. Delivery rate metric

In order to calculate delivery rate, programming intervention will be at in the DataReaderImpl class and SubscriberImpl class.

```
public class DataReaderImpl   extends
javax.realtime.RealtimeThread implements DataReader {

private static int numberOfDataReader=0;
private static int numberOfDataReaderDeliver=0;
private Subscriber SubParent ;
public DataReaderImpl   (….)
{
numberOfDataReader++;
}
Public void incrementNumberOfDataReaderDeliver()
{
numberOfDataReaderDeliver++;
}
public double deliveryRate() { return numberOfDataReaderDeliver/
numberOfDataReader ;}
}
```

Figure 8: DataReaderImpl class

```
public class SubscriberImpl   extends
javax.realtime.RealtimeThread implements Publisher {

private Vector<DataReader> dataReaders ;

public DataReader lookup_datareader(String topicName) {

{ boolean read=false;
 DataReader found=null;

// find DataReader

if(read) found.incrementNumberOfDataReaderDeliver();


return found;

}
```

Figure 9: SubscriberImpl class

If Subsriber arrive to read DataReader the number of data reader deliver was increment. The number of data reader and number of data reader deliver are static to be shared from one instance of data reader.

5.3. Completion Time metric

For calculate completion time we inject our real time clock in DataReaderImpl class.

```
public class DataReaderImpl   extends
javax.realtime.RealtimeThread implements DataReader {
…
 private RealTimeClock clockDataReader;
 private AbsoluteTime thisTime;
 private AbsoluteTime endTime;
 private AbsoluteTime startTime;
 private RelativeTime completionTime;
public void run(){
……………………………………………..
do{
completionTime = thisTime.subtract(this. startTime);
}
While(waitForNextPeriod() )
}
}
```

Figure 10: Completion Time

thisTime and startTime attributes are manager by clockDataReader. CompletionTime is a relative time, the application of network priority mapping algorithm showed an improvement of completion time.

6. Conclusions

This paper present one solution for evaluate a middleware in general and specially DDS middleware. This solution is applicable if the middleware is developed with RT-JAVA language. Use RTSJ to evaluate a performance of DDS middleware give more precision of results and prevents the application of mathematical models. Our research group seek an effective solution be easily integrated in process development. We are developed and evaluate with RT-JAVA a functionally DDS middleware working via Control Area Network (CAN) Bus. Actually we applied a same work on FlexRay bus. We search soft ant hardware real time characteristics.

7. References

[1] Gerardo Pardo-Castellote, Bert Farabaugh, Rick Warren, “An introduction to DDS and Data-Centric Communications,” www.rti.com/resources.html.

[2] Douglas C. Schmidt and Carlos O'Ryan, “Patterns and Performance of Distributed Real-time and Embedded Publisher/Subscriber Architectures,” Journal of Systems and Software, Special Issue on Software Architecture -- Engineering Quality Attributes, edited by Jan Bosch and Lars Lundberg, October 2002.

[3] Chris Gill, Jeanna M. Gossett, David Corman, Joseph P. Loyall, Richard E. Schantz, Michael Atighetchi, and Douglas C. Schmidt, “Integrated Adaptive QoS Management in Middleware: An Empirical Case Study,” Proceedings of the 10th Real-time Technology and Application Symposium, May 25-28, 2004, Toronto, CA.

[4] Gerardo Pardo-Castellote, “DDS Spec Outfits Publish-Subscribe Technology for GIG,” COTS Journal, April 2005.

[5] OMG, “Data Distribution Service for Real-Time Systems Specification,” www.omg.org/docs/formal/04-12-02.pdf.

[6] Ioana Burcea, Milenko Petrovic, Hans-Arno Jacobsen. S-ToPSS: Semantic Toronto Publish/Subscribe System. International Conference on Very Large Databases (VLDB). p. 1101-1104. Berlin, Germany, 2003.

[7] Arvind S. Krishna, Douglas C. Schmidt, Ray Klefstad, and Angelo Corsaro, “Real-time CORBA Middleware,” in Middleware for Communications, edited by Qusay Mahmoud, Wiley and Sons, New York, 2003.

[8] Fox, G., Ho,A., Pallickara, S., Pierce, M., and Wu,W, “Grids for the GiG and Real Time Simulations,” Proceedings of Ninth IEEE International Symposium DS-RT 2005 on Distributed Simulation and Real Time Applications, 2005.

[9] D.S.Rosenblum, A.L.Wolf, “A Design Framework for Internet-Scale Event Observation and Notification,” 6th European Software Engineering Conference. Lecture Notes in Computer Science 1301, Springer, Berlin, 1997, pages 344-360.

SESSION

FPGA + NOC + MULTI-CORE SYSTEMS + MICRO-CONTROLLERS + COMMUNICATION ISSUES

Chair(s)

TBA

Connectivity based Dual V_{dd} Assignment Algorithm for Power Reduction in FPGA

G. Veera Sekhar and Jatindra Kumar Deka
Department of Computer Science and Engineering, Indian Institute of Technology Guwahati, Assam, India

Abstract— *In the past few decades with advancement in VLSI technology, FPGA chip density has increased and FPGA devices now provide a large number of smaller feature size transistors and can support higher clock speeds. While this advancement is beneficial for implementing larger and faster designs within a single chip, it also leads to increased power consumption. With the remarkable growth of FPGA based battery-powered systems, such as personal computing devices, wireless equipment and consumer electronics, low power FPGA design is of increased importance. With this huge continuing increase in FPGA size, its complexity and the new technology emergence has made power estimation and optimization as an ultimate design consideration. In this work, we are supplying dual V_{dd} to the logic blocks, instead of the single V_{dd} used in traditional FPGAs. The assignment of dual V_{dd}(low/high) to a logic block is based on the criticality of the path. We have proposed an alternative algorithm to find out the critical path in the FPGA implementation of the circuit. Depending on the priority of a node in a critical path, we assign low V_{dd} to a node so that the performance of the circuit remains within the specified limit. Due to the application of low V_{dd} to some of the nodes in the FPGA implementation, the power consumption is reduced.*

Keywords: FPGA, Dual Voltage Assignment, Power Reduction, Critical Path

1. Introduction

FPGA, field programmable gate array is the most commonly used programmable device at present. FPGA is an integrated circuit(IC) designed to be configured by the customer or designer hence field programmable. Hardware description language (HDL) is generally used to specify the FPGA configuration similar to that used for ASIC. Using FPGA we can implement any logic function that an ASIC could perform [2]. The main advantages of FPGAs are lower development, lesser time to market and their ability to be reprogrammed several times. Due to these advantages, a lot of research has been done on development environment, architecture and applications for FPGA.

Nowadays usage of mobile devices and portable devices like mobile phones, digital cameras, notebooks, etc. increase rapidly. From end user point of view most often performance, features, size, and weight are the main criteria. This criterion has become the main design constraints in the design process and has an impact on the power consumption [1], [5], [7], [8], [10]. These demand for low power consuming devices and so power becomes most important issue in FPGA architecture design. As the application of FPGA extends to image processing and to other high complexity processes, a multimillion gate FPGAs are become necessary. These consume a lot of energy. Thus in case of mobile devices power optimization or maintenance techniques has become more important to guarantee long battery life. Even in non mobile devices, where power is continuously available the low power design constraint is still important. In many applications we can achieve the desired performance by increasing the operating frequency under given power constraints. It is crucial to implement a power efficient FPGA design without affecting the performance for many FPGA systems. There are a number of techniques proposed already for the reduction of power in ASIC domain. But, they cannot be applied directly to FPGA's, because FPGA and ASIC differ in architecture design. One more disadvantage about consuming high power is that devices generate lot of heat which further affects device performance though we have some techniques to get rid from the heat generated. Low power techniques are much better than in some cooling techniques such as sinks and fans. Thus thermal management and power management has got more importance in the FPGA design architecture [3], [4], [6], [9], [11], [12].

Moreover, programmability of FPGA can be leveraged to develop efficient low power design techniques. There are two possible approaches for the reduction of power consumption by FPGA based systems. (i) Redesign the FPGA device to reduce the static power and its components which contribute to dynamic power such as output gate capacitance and resistance of CMOS gates etc. (ii) take dynamic power into consideration while designing an FPGA circuit. The first option will require modification of FPGA architecture and topological change in its implementation. The second option mainly considers about reducing the dynamic power. Dynamic power consumed by an FPGA is primarily dependent on clock frequency, switching activity, supply voltage and resource utilization. In this work, we consider the reduction of dynamic power by reducing the supply voltage and for that we use two voltage level low V_{DD} and high V_{DD}.

A field-programmable gate array (FPGA) is a user or customer configured integrated circuit, and so is called "field programmable". A HDL (Hardware Description Language) language can be used to specify the configuration of a FPGA, similar to that an ASIC.

Nowadays FPGA's are used in many applications including many portable devices like mobile applications. Considering both high performance and mobile applications, power consumed by these have become a limiting factor for FPGA's wide applications.

Low power design is important from three different reasons, there are mainly, device temperature, life of the battery and overall energy consumption.

The remaining of the paper is organized as follows: In Section 2, we present the basic architecture of FPGA. Section 3 and Section 4 deal with the power sources of FPGA and low power techniques respectively. The dual V_{dd} assignment algorithm is presented in Section 5. The experiment results is shown in Section 6 and finally the conclusion of the paper is given in Section 7.

2. FPGA Architecture

2.1 Logic Block

Logic blocks are the primary elements of FPGA through which any function can be implemented. The capacity of the logic blocks can be increased by increasing the size, e.g., by increasing the number of inputs, one can augment the possible functions which could be implemented by less number of logic blocks. But the research works showed that the area delay product will increase with the increase of the size. However this could be a waste in some applications where not all inputs are utilized. FPGAs use Look Up Tables (LUTs) for implementation of logic functions. With n input LUT one can implement 2^m possible functions, where m is 2^n and each function requires 2^n bits configuration. Previous work has shown that 4 inputs LUT is optimum in terms of area and power [3]. Fig. 1 shows the basic logic which consists of one 4-input LUT, where a combination function is implemented. A flip-flop that will be needed in sequential circuit design. A 2-to-1 MUX is used to switch between registered and unregistered output.

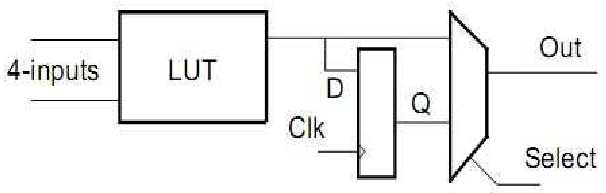


Fig. 1: Basic Logic Block

2.2 The Connection Block

The connection block is responsible of connecting the resources between each other and assures that data can flow to the I/Os. Each connection block consist of a programmable connection block which selects the signals in the given routing channel to be connected to the logic block's terminal, and a programmable switch block that connects between horizontal and vertical routing resources. The structure of Connection Block is shown in Fig. 2.

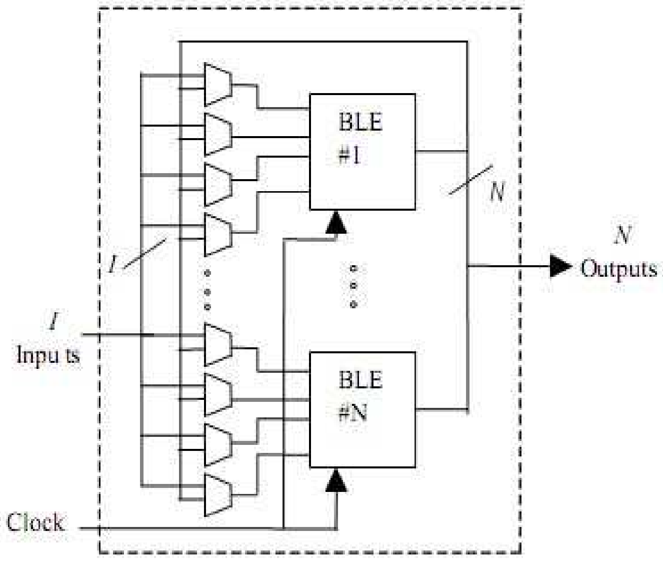


Fig. 2: Clustered Based Logic Block

The programmable interconnect is the core of an FPGA, which connects different LUTs and flip-flops to each other and route signals to and from the input output blocks(IOBs). In addition current generation FPGAs feature a large number of on chip synchronous blocks and a few on chip macro functions such as multiplexers and shift registers. The LUTs in current generation are clustered into a logic element or configurable logic block (CLB).

2.3 Routing Architectures

Since the routing wires consume a major part of the total amount of FPGA area and power, selection of routing architecture is crucial in a FPGA design. Island style routing architecture is used in this work. In Island Style Routing Architecture the logic blocks are connected by a two dimensional, mesh-like interconnect structure with horizontal and vertical routing channels and these are connected by programmable switch boxes. A simplified view of island style routing architecture is shown in Fig. 3. In this routing structure, half of the routing tracks consists of length 1 wires (wire that span for one logic block) and remaining half consists of length 2 wires. Pass transistorize tri-state buffers are used as programmable routing switches. There are also connecting boxes connecting the wire segment to the logic block inputs and outputs.

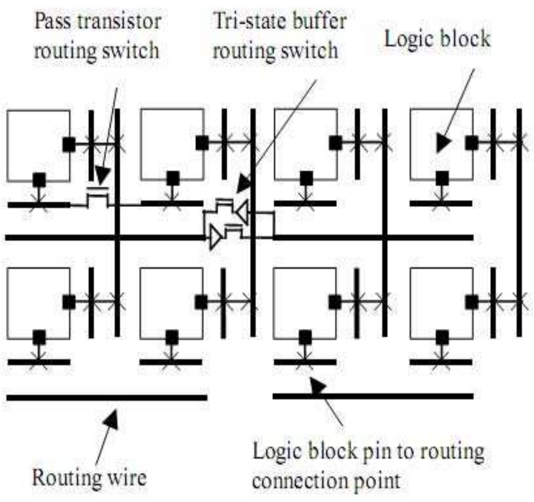

Pass transistor routing switch
Tri-state buffer routing switch
Logic block
Routing wire
Logic block pin to routing connection point

Fig. 3: Island Style Routing Architecture

3. Power Sources in FPGA

In FPGA, power is mainly divided into two categories - static power and dynamic power. Static power is the power consumed by the node when there is no signal transition to that node, that is, the power required to keep it on. Static power dissipation is mainly due to leakage current. Leakage current is caused due to two main reasons - Reverse biased diode leakage current and Sub-threshold conduction of transistors [4]. These leakage currents depend on supply voltage and the technology used. By reducing the supply voltage, the leakage current may be reduced.

Dynamic power is the power consumption caused by dynamic circuit activities which are reflected by dynamic current flowing between two circuit nodes. On the other hand, we can say that dynamic power is proportional to the signal transitions of a design that mapped into an FPGA. This dynamic power is consumed only when there occurs a signal transition at gate output and there are two types of signal transitions.

- *Functional transition:* This is the signal transition necessary to perform the required logic functions between two consecutive ticks.
- *Glitch or spouse transition:* It is an unnecessary signal transition due to the unbalanced path delay to the inputs of a gate. Glitch power consumes most part of the dynamic power. So, a change from supply voltage is transferred to the CMOS gate output capacitance when the output of a CMOS gate changes from level 0 to 1. This transition causes power dissipation in the resistive CMOS network.

The dynamic power consumed by an FPGA is primarily dependent on Clock frequency, Switching activity, Supply voltage and Resource utilization. Unused resources in an FPGA design have a switching activity nil and hence they do not consume any dynamic power. Hence dynamic power consumption in FPGA is proportional to the amount of resource used by a design.

Switching power is the power that is consumed due to the current charging and discharging the capacitance at the gate output. A CMOS inverter is a good example to illustrate this analysis of switching power. When the input to the inverter goes low, the NMOS transistor is cut-off and the PMOS transistor conducts, creating a direct path from the supply voltage to the output capacitance. The equation for switching power is given as,

$$\P_{sw} = f.V_{dd}{}^{2}.\sum_{i=1}^{n} C_i\, E_i \tag{1}$$

Where n is the total number of nodes, f is the clock frequency, V_{dd} is the supply voltage, C_i is the load capacitance for node i and E_i is the transition density for node i.

It is obvious from Equation 1, the switching power can be reduced by reducing the supply voltage. But the reduction of supply voltage may degrade the performance of the circuit. In this work, we are exploring the use of dual voltages in FPGA. We will apply high V_{dd} to the nodes which are in critical paths and effect the performance of the circuit. We will apply low V_{dd} to others nodes.

4. Low Power Techniques

In our work, we use dual V_{dd} architecture where, the V_{dd} of a circuit block is selected between V_{ddh} (high V_{dd}) and V_{ddl} (low V_{dd}) by using two transistors connecting the block to the supply voltages. By using our V_{dd} assignment algorithm, we will set a configuration bit which controls the on/off state of each supply voltage transistor. The configuration bit is set in such a way that the block will either connect to one of the power supply or it will disconnect completely from power supply if the block is not in use. That is, we will switch off the block when it is completely idle or unused. This reduces the static power required to a block when it is idle.

We are using cluster based island style FPGA architecture for our proposed dual V_{dd} architecture and the configuration bits are stored in SRAM cells. This facilitates configurable supply voltage for logic blocks and routing multiplexers. The architecture of the FPGA is shown in Fig. 4. The basic logic element consists of one 4-input LUT and one flip-flop. A CLB is formed with the clustering of 8 BLE's together. Dual V_{dd} design requires level conversion when a low V_{dd} block drives a block operating at high V_{dd} and vice versa. In our dual V_{dd} architecture, level conversion only takes place at CLB pins. So, we attach the level converters to the CLB pins.

This placement of level converters (LCs) at the CLB pins reduces the complexity. We have two different architectures for placing the level converters at CLB pins [16]. One at CLB input pins and another CLB output pins. Fig. 4 shows the second architecture, where LC's are placed at CLB output pins.

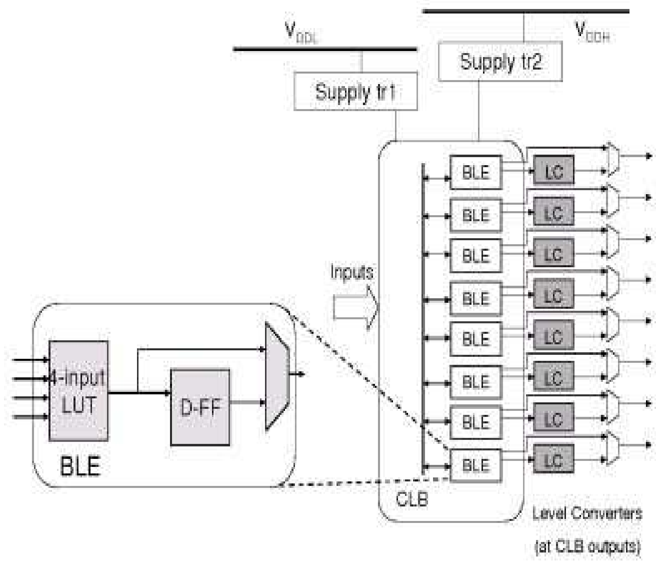


Fig. 4: CLB used in Dual v_{dd} architecture

A routing multiplexer is used in dual V_{dd} architecture as shown in Fig. 4. As per our requirements, the output of previous stage is directly given to the next stage or it may be provided after level conversion. Level converters are used to change the level of a signal voltage from low voltage to high or high to low.

5. Connectivity Based Dual V_{dd} Assignment

This algorithm takes all the nodes and critical paths into consideration. It arranges all the nodes of the critical paths based on their connectivity factor. Connectivity of a node defines the number of paths connected to the node. If a node is connected to more number of the paths than the other nodes then it has the highest priority to get assigned high V_{dd} than the other. Before assigning the high V_{dd} to this nodes, we check whether all the paths passing through this node meet the delay requirement. If some of the paths do not meet the delay specification, we assign high V_{dd} to this selected node. After assigning high V_{dd} to this node, if the performance does not improve or the delay requirement goes below the specified limit, we backtrack and reassign the low V_{dd} to that node. We repeat this procedure for other nodes till we achieve the required performance.

```
Input: Critical Path nodes and delay factors
Output: Critical path nodes with high $V_{dd}$
Assign $V_{ddh}$ to all CLBs and routing MUXes;
$P \leftarrow$ List of all paths in the design;
$T \leftarrow$ Longest delay path when all circuit blocks
operate at $V_{ddh}$;
$T_d \leftarrow x{*}T$, where $x >= 1$ is a user defined
performance metric;
$V_{ddl}$Delay($P_i$) $\leftarrow$ Delay($P_i$) when all blocks in $P_i$ are
at $V_{ddl}$;
Path List($P$) $\leftarrow$ {All paths || Criticality($p_i$) $>$
Criticality ($P_{i+1}$)};
Critical Path $\leftarrow \{P_i \in P \|\ V_{ddl}$ Delay($P_i$) $> T_d\}$;
N $\leftarrow$ All CLBs in the circuit sorted in descending
order based on their priority given to them based on
their Connectivity;
while N is Not Empty do
    Assign $V_{ddh}$ to $N_i$ and to all the routing MUXes
    driven by $N_i$;
    Remove $N_i$ from the list;
    if Delay of any PathList($P_i$) $< T_d$ then
        Reset last action;
    else
        Update delay of all paths passing through $N_i$;
    end
end
```

Algorithm 1: Priority Based Dual V_{dd} Assignment Algorithm

In this work, we are proposing a low-high priority based algorithm. We first assign V_{ddl} to all CLBs and routing muxes and calculates the delays of each path. After the calculation of delays, we sort the paths such that longest delay path should come first (descending order). Now, define a performance metric T_d, T_d=x∗t, where x >=1 (t is the longest delay) as a threshold voltage applicable to circuit. This should be different for different circuits. Now, consider the CLBs and calculate the number of paths to which each CLB is connected. Assign the number to priority variable. Now, sort them (in descending order) based on their priority value. Now, we consider each path one by one and assign the high V_{ddh} to the first occurring block of that path. Now, check if any path violates this allocation. That is if any path delay that involves this block exceeds than its threshold delay. If there is no path like that, we proceed to next node, otherwise we will reset the block voltage and proceeds further.

This algorithm gives a relatively better V_{ddl} allocation for the nodes in a circuit. We are using low-to-high assignment of V_{dd}. This is preferable than high-low, because in high-low all the blocks will be assigned high voltage at the beginning and will be assigned low voltage afterward. In this each CLB consumes high V_{ddh} at startup and sometimes this may cause the paths delay to be reduced more than what is necessary in assigning low V_{ddl}.

6. Experimental Results

In this work, we are trying to reduce the power consumed by the FPGA circuits. We proposed one algorithm, which is used to assign the voltages to the nodes such that the total power consumed by the FPGA circuit will be reduced without reducing the specified performance.

We try to decrease the dynamic power consumed by the FPGA circuit. The power consumed by the FPGA circuit is the total power consumed by its each and every individual node. The circuit includes LUTs, switch boxes, input and output pads. So to decrease the power consumed by the complete FPGA circuit means, we have to decrease the voltage (power depends on the voltage applied) applied to the individual node. There are some critical paths in the circuit and we must ensure that the total delay of the critical path shouldn't vary as it is pre programmed to get the expected result. Though we can change the delay metrics of individual nodes of the critical path until we ensure the total path delay is not altered.

To reduce the power consumed by the FPGA, we will assign low voltage (we divide the voltage into two operative voltages depending on the technology used) to the nodes of the circuit that are not on the critical path. We will assign high voltage to some of the nodes on the critical path based on our proposed algorithm. In the process, we must ensure that the circuit performance should not degrade while trying to reduce the power.

In our experimental set up, we have used the ISCAS89 bench mark circuits to show the benefit of our method. The detail flow of our experiment is:

- Logic synthesis and optimization
- LUT mapping (cluster based)
- Packing using t-v pack
- Placement and routing
- V_{dd} assignment
- Power estimation

For our simulation purpose, we have taken the ISCAS89 benchmark circuit in .bench format. We have used the ABC synthesizer for logic synthesis and logic optimization. Next we use the RASP tool for technology mapping, which maps the circuit to LUT based FPGA. It also does the post processing for area reduction. After that we use the placement and routing tool VPR to place and route the FPGA circuit.

We have implemented the algorithm for the assignment of voltage to the different nodes of the FPGA. According to the algorithm, it assign either low voltage or high voltage to a node.

We have used the power evaluation tool FPGAEVA-LP to estimate the power consumed by an FPGA implementation of the benchmark circuits. This tool also estimates the delay of the circuit. For comparison, we have estimated the power consumption by the existing method [14] and by our proposed method.

Circuit	Gates	LUTs	Cri.Nodes	V_{ddh} Nodes(pro)	V_{ddh} Nodes(Exi)
S27	8	10	21	11	16
S400	106	162	32	19	22
S444	119	181	36	19	21
S838a	288	446	50	6	8
S953	311	418	48	26	31
S208	61	96	42	16	21
S298	75	119	30	18	26
S382	99	158	32	20	29
S510	179	211	34	21	26

Table 1: Comparing Both The Solutions

The experiment result is presented in Table 1. The first column of the table indicate the circuit number that we have used. Second and third column provide the number of gates of the circuit and number of LUTs used during FPGA implementation respectively. Column four gives the number of critical nodes of the circuit. Column five and column six indicate the number of nodes to which high V_{dd} have been assigned by our proposed method and the existing algorithm respectively. It has been observed that our proposed method has assigned high V_{dd} to less number of nodes in all test circuits in comparison to the existing method.

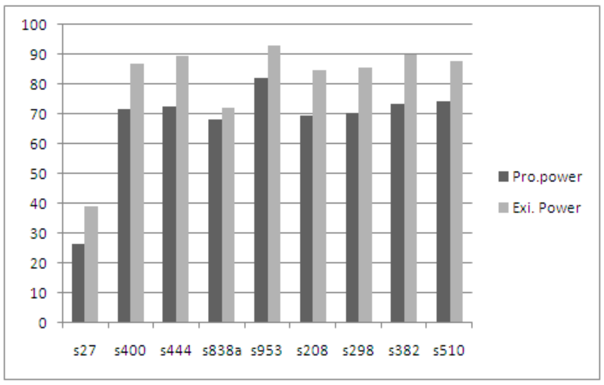


Fig. 5: Graphical Representation of Power consumed by two algorithms

The power consumed by each circuit in both the algorithms are represented graphically in Fig. 5. It is observed that in our proposed method the power consumption is always less. The delay experienced by each circuit in both the algorithms are represented graphically in Fig. 6. Obviously the delay incurred by a circuit in our proposed method is more than the delay incurred by the existing algorithm. It is quite obvious that in our case we are applying high V_{dd} to less number of nodes. But in both the cases, the delay incurred by each circuit is always less than the specified delay of the circuit.

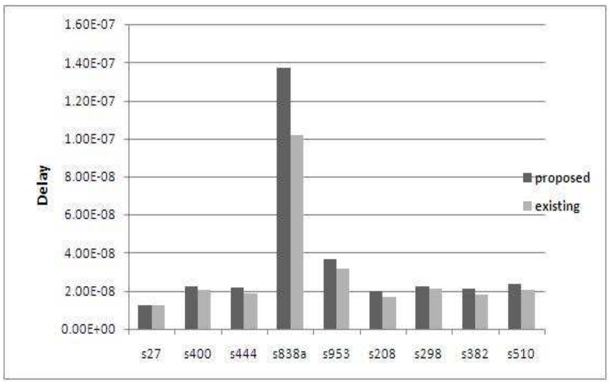


Fig. 6: Graphical Representation of delay by two algorithms

Circuit	PowerSaved(%)
S27	32.436919
S400	17.419271
S444	18.656502
S838a	12.443598
S953	12.013707
S208	17.867090
S298	17.704979
S382	18.536043
S510	15.448448

Table 2: Total Power Saved

The percentage of power saved by our method is shown in Table 2.

7. Conclusion

We have presented a Dual V_{dd} assignment algorithm for low-power FPGA based on priority. The priority of the node is determined by the critical path. If a node is present in several critical paths, then it contributes more for the degradation of performance. In such situation, we apply high V_{dd} to this node. After applying the high V_{dd} to such nodes, we again check for the performance of the circuit. In experiment result, we have observed that our proposed method able to reduce the power consumption without affecting the performance of the circuit.

References

[1] A. P. Chandrakasan and R. W. Brodersen, "Low Power Digital CMOS Design Norwell," *MA,USA: Kluwer Academic Publishers*, 1995.
[2] "Introduction of FPGA," Wikipedia.org.
[3] H. Hassan, "Design Methodologies and CAD Tools for Leakage Power Optimization in FPGAs,"University of Waterlo., 2008.
[4] J.M. Chang and M. Pedram, "Register allocation and binding for low power," in DAC '95: Proceedings of the 32nd annual ACM/IEEE Design Automation Conference. New York, NY, USA: ACM, 1995, pp. 29-35.
[5] N.Weste and K. Eshraghian, "Principles of CMOS VLSI design," AddisonWesley VLSI system series, 2008.
[6] H. Veendrick, "Short-circuit dissipation of static cmos circuitry and its impact on the design of buÂőer circuits, Solid-State Circuits," IEEE Journal of, vol. 19, no. 4, pp. 468-473, Aug 1984.
[7] A. Raghunathan, N. K. Jha, and S. Dey, "High-Level Power Analysis and Optimization," Norwell, MA, USA: Kluwer Academic Publishers, 1998.
[8] Chandrakasan.P and Brodersen.R, "Low Power Digital CMOS Design," Kluwer Academic Publishers, 1995.
[9] J. P. Knight and R. S. Martin, Power-proÂŕler: "Optimizing asics power consumption at the behavioral level," Design Automation Conference, pp. 42-47, 1995.
[10] S. Malik and S. Devadas, "A survey of optimization techniques targeting low power vlsi circuits," Design Automation Conference, pp. 242-247, 1995.
[11] M. E. Rabaey, Jan M.; Pedram, "Low Power Design Methodologies," Kluwer Academic Publishers, 1995.
[12] F. Li, D. Chen, L. He, and J. Cong, "Architecture evaluation for power-efficient FPGAs," in FPGA '03: Proceedings of the 2003 ACM/SIGDA eleventh international symposium on Field programmable gate arrays. New York, NY, USA: ACM, 2003, pp. 175-184.
[13] D. Chen, J. Cong, F. Li, and L. He, "Low-power technology mapping for fpga architectures with dual supply voltages," in FPGA '04: Proceedings of the 2004 ACM/SIGDA 12th international symposium on Field programmable gate arrays. New York, NY, USA: ACM, 2004,pp. 109-117.
[14] Fei Lei, Yan Lin, Lei He, "FPGA Power Reduction using Configurable Dual V_{dd}," Elcetric Engineering Dept, UCLA,2006
[15] Fei Li and Lei He, "Power Modeling and Characteristics of Field Programmable Gate Arrays", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.
[16] Fei Lei, Yan Lin, Lei He, " Low power FPGA using pre-defined Dual-V_{dd} Dual V_T fabrics", Electrical engineering department, University of California, Losangels.

Parametrizable NoC Emulation Framework for Performance Evaluations

Jaya Suseela and Venkatesan Muthukumar
Dept. of Electrical and Computer Engineering, University of Nevada Las Vegas, Las Vegas, NV-USA

Abstract: Specific parameters for Network on Chips (NoCs), such as topology, switching method, and packet sizes, have a huge impact on performance of NoCs. Cycle and bit accurate simulation and emulation are necessary to evaluate and validate the performance of the NoC system. The goal of this work is to develop an open platform, synthesizable NoC framework that would evaluate such performance metrics as area, power, latency, and congestion for various design explorations. The NoC framework developed is completely parameterizable, where the designer can evaluate various design space explorations like topology, PE architecture, switching and routing algorithms, packet size, and error correction, by modifying the configuration file. The proposed NoC framework has been evaluated for various congestion scenarios, and the results are discussed.

Keywords: NoC, Router, Implementation, Cycle and bit accurate, Openrisc.

1 Introduction

Networks on Chips (NoCs) have been proposed as a promising solution to complex on-chip communication problems. However, many challenging research problems remain unsolved at all levels of design abstraction, such as design exploration of NoC architecture for applications; scheduling and mapping algorithms; evaluation of switching, topology or routing algorithms for efficient execution of applications; and optimizing communication costs, area, energy, and so forth. A solution to solving the above problems calls for the development of a synthesizable, parameterizable NoC framework that would evaluate and implement these problems and algorithms with minimum ease and flexibility.

The main contribution of our work is the implementation of a parameterizable cycle accurate NoC framework. The framework helps us to: (1) explore the architectural design space faster (2) evaluate and choose efficient NoC architecture from a range of switching techniques and topologies, with regard to latency, area and power; and (3) evaluate different architecture, topologies, switching, and routing algorithms extensively with various regular traffic patterns and application-oriented traffic. Moreover, the design is fully synthesizable and has been implemented in a field-programmable gate array (FPGA).

A brief summary of existing NoC simulators and emulators are presented here. Orion [7] and LUNA [8], two NoC simulators especially developed for power simulation of on-chip interconnection networks, do not consider computational cores. FAST [6] is a functionally accurate NoC simulator limited to IBM's proprietary Cyclops-64 architecture. SICOSYS [13] is a general-purpose interconnection network simulator that captures essential details of low-level simulation. RSIM simulates shared-memory multiprocessors and uniprocessors built from processors that aggressively exploit instruction-level parallelism (ILP). RSIM, which is execution-driven, models state-of-the-art ILP processors, an aggressive memory system, and a multiprocessor coherence protocol and interconnect, including contention at all resources. NoC simulators such as NNSE [9], Noxim [10], and NIRGAM [11] have flexibilities in configuring parameters of on-chip networks and are capable of obtaining performance metrics; however, these simulators are based on SystemC and are not synthesizable. XPIPES [19] consists of parameterizable network building blocks that can be composed at instantiation time; the parameterizable factors are the network interface, switches, and links.

The ability of a network to efficiently disseminate information depends largely on the topology. Mesh and Torus are the most commonly used topologies. The WK-recursive networks [16] are a class of recursively scalable networks that offer a high degree of regularity, scalability, and symmetry.

The NoC framework with WK-recursive topology is shown in Figure 1.

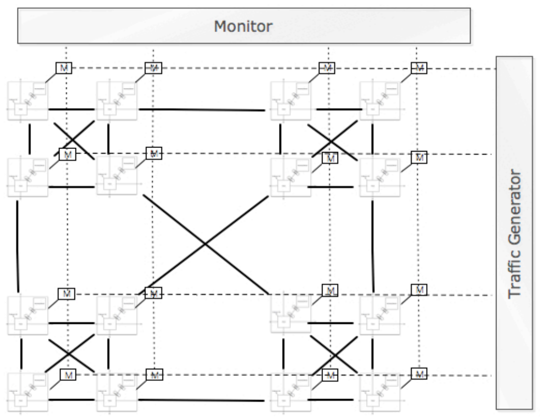


Figure 1: NoC Framework with WK Topology

In this paper, Section 2 explains briefly the implementation details, and Section 3 details the communication flow in the NoC framework. Performance analysis of switching networks and topologies from simulation and synthesis results also are presented in Section 3. Section 4 provides the results and conclusions of this work.

2 NoC Framework Implementation

The proposed NoC framework consists of five main modules: i) the Processing Architecture, ii) the Communication Infrastructure, iii) a Communication Paradigm, iv) the Monitor, and v) the Traffic Generation Module.

The Processing Architecture module consists of a Processing Element (PE) and Network Adapter (Core Network Interface) module. The Communication Infrastructure consists of network topology and a routing node. The Communication Paradigm describes the switching techniques and routing algorithms employed in the NoC Communication Infrastructure. The Monitor module includes two sub modules: a) a Node monitor, which monitors the activities in a routing node, and b) an NoC monitor, which monitors the communication within the framework.

2.1 Processing Architecture

The processing element (PE) in the framework can be a master PE or a slave PE. Only master PEs can initiate a message transfer. Slave PEs respond to the requests from the master PE either by sending back the requested signals/data or by saving the received information. In our framework, UART, TIMER, Instruction/Data Memory and slave processors are considered as slave PEs, and the master PEs and slave processors are capable of performing computational operations.

Each master PE or slave processor consists of one OpenRisc 1000 (OR1K) [1] processor that communicates to an Instruction memory (IMEM) through a Wishbone bus. OR1K is a 32-bit load and stores an ARM9-based RISC embedded processor with 5 state pipelines; it has a maximum clock frequency of 250MHz. The OR1K processor shows better performance per clock cycle than MicroBlaze in the Stanford benchmark [4], and therefore is considered a more efficient architecture than the MicroBlaze architecture [3]. The architecture defines several features that are quite useful for networking and embedded computer environments. Most notable are the 32/64-bit architecture, the Programmable Interrupt Controller, several instruction extensions, a 2/3 Level Cache, a configurable number of general-purpose registers, a configurable cache and TLB sizes, dynamic power management support, and space for user-provided instructions. IMEM is a Block Ram with an 8-bit data bus and a 32-bit address bus. The instructions to be executed by the core are loaded in IMEM. The Wishbone clock frequency can be equal to an OR1K or an OR1K/2 clock frequency.

The Network Adapter (NA) interfaces the PEs with the network. Its main function is to generate and process packets to and from the PEs. The NA component on the master side is called the Core Interface (CI); the Network Adapter on the slave side is called a slave network interface (NI).

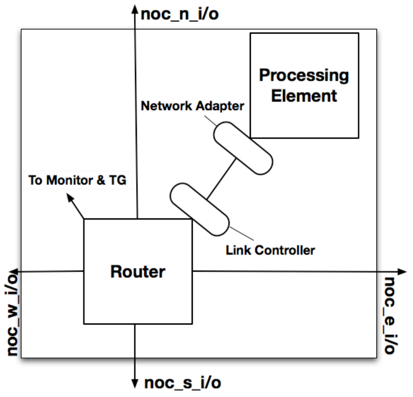


Figure 2: Routing node

2.2 Communication Infrastructure

The communication infrastructure consists of a routing node and network topology. The routing node (shown in Figure 2) consists of a link controller and a router. The link controller (LC) provides an interface between the NA and the NoC. Its main function is to match the NA clock rate with that of the network topology. Routing nodes run at four times the frequency of PEs. Synchronization registers are used to match clock rates between the slow PE and fast routing nodes. First-in first-out (FIFO) buffers are also added in the LC to store data packets from the network before transmitting to adjacent PEs.

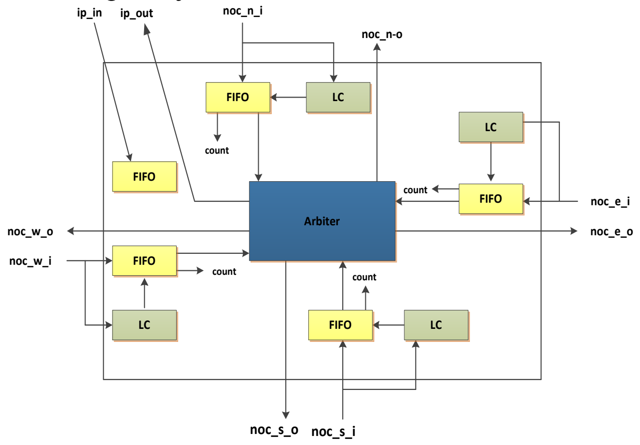


Figure 3: Router Architecture

The Network Router is responsible for the transfer of packets between nodes. Each router consists of two main components: input buffers and an arbiter. Each router (shown in Figure 3) has five input and five output ports. There exist four inputs/outputs from/to the four cardinal directions (North, East, South and West), and one from/to the PE. To prevent deadlocks, the input buffer implements the virtual channel concept.. The Virtual Channel (VC) identifier module determines which VC should be used based on occupancy of the input buffer. VC identifier polls the buffer count of each VC and directs the incoming packet to the least occupied VC. The Switch Identifier module chooses a packet from each VC in a round-robin manner and sends it to the output port based on the routing signals obtained from the arbiter. The arbiter implemented is FSM-based, and consists of the routing table with the shortest path to the destination PE.

2.3 Communication Paradigm

In order to forward the message/packet, the implemented NoC framework can choose either the Store and Forward (SF) switching technique or the Wormhole (WH) switching technique. In SF switching, the message can be sent either as packets or in the form of flits. Each flit is contains 25 bits. When the message is transmitted as flits, each routing node will wait until the entire message is received before processing the HEADER. The end of the message/packet is determined by the TAIL flit. In Wormhole routing, the message is transmitted as soon as the HEADER is available. The path is determined from the HEADER as it moves through the network. The remaining flits follow the same path. The path is disconnected when the TAIL flit is received. For the Torus and Mesh topologies, the implementation uses an XY routing algorithm with store and forward switching. For WK-recursive topology the framework uses the adaptive routing algorithm with wormhole switching.[18].

2.4 Monitor Module

Every routing node in the NoC is connected to a “Node monitor,” which connects to a top-level monitor called the “NoC monitor.” The main function of the NoC monitor is to collect information from individual Node monitors regarding the traffic. The Node monitors generate control information based on the buffer conditions of that router node. The Node monitor uses a few ON/OFF signals, such as FAIL, FULL and ALMOST FULL, to communicate with the NoC monitor.

2.5 Traffic Generator Module

The Traffic Generator (TG) module is responsible for generating different traffic distribution in the network. The TG can generate mainly three different types of traffic: 1) uniform traffic, where packets are send at equal intervals of time; 2) hotspot traffic [2], where the cores either receive packets at a higher rate than the rate they can process or else generate packets at a higher rate than the destination can process; and 3) sporadic traffic, where each core generates a burst of packets.

3 NoC Framework Communication Flow

This section explains the control and data flow sequence followed during the transfer of packets between PEs. Each packet can contain a variable number of flits. Every packet consists of a header and a tail flit. Data and address flits are optional. The framework allows packing of a variable number of data flits into a single packet, which further reduces latency during burst mode data transfers. The packets are classified as request packets, which have an optional address field; and response packets, which have a data field. The address field defines the local memory address in the destination PE. This allows the slave PE to have its own unique memory address space. The packet format is shown in Figure 4.

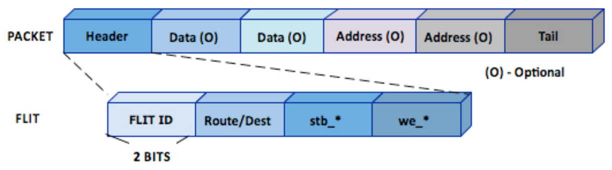


Figure 4: Packet Format

The header flit can consist of either only the source and destination address or the entire route. Two bits in each flit are used as a packet ID, which determines the type of packet (00-Header, 01-Data, 10-Address, 11-Tail). In the data and address flits, the first three bits next to the packet ID determines the order of the data/address flits. Every flit contains a control bit (stb_*), which determines the validity of packet. The we_* bit signifies if the packet has to perform a read or write operation. Typically, each data or address flit contains 16 bits of data or address information. The tail flit contains parity bits for every data or address flit. A parity bit is a bit that is added to ensure that the number of bits with the value of “one” in a set of bits is even or odd. Parity bits are used as the simplest form of error detecting code (ECC). The type of ECC (odd or even parity) used in tail flit is parameterizable.

Every channel/link in the network is full duplex, i.e., two messages can travel simultaneously on the link in opposite directions. A channel/link is said to be congested if its associated router buffer is completely full or partially full (parameterizable). A READY and SEND signal are used to communicate between adjacent routers. Whenever the channel buffer is partially full, the monitor informs every adjacent router about a possibility of congestion.. When the channel buffer of the router is full, the monitor flags the congestion by setting the READY signal to low.

The communication flow in the proposed NoC framework, as shown in Figure 5, can be summarized as follows. Let us assume a scenario where the master PE wants to write data to a specific address location on a slave PE (D-MEM). The master PE initiates the transfer by activating the Network Adapter (NA), which validates the memory address and assembles the packet. In source routing, the NA uses a routing table to determine the route. Once the packet is ready, the Link Controller stores the packet in the input buffer of the router. The router arbiter, based on the route information and the availability of input buffers in the adjacent routers, determines the output port and forwards the packet.

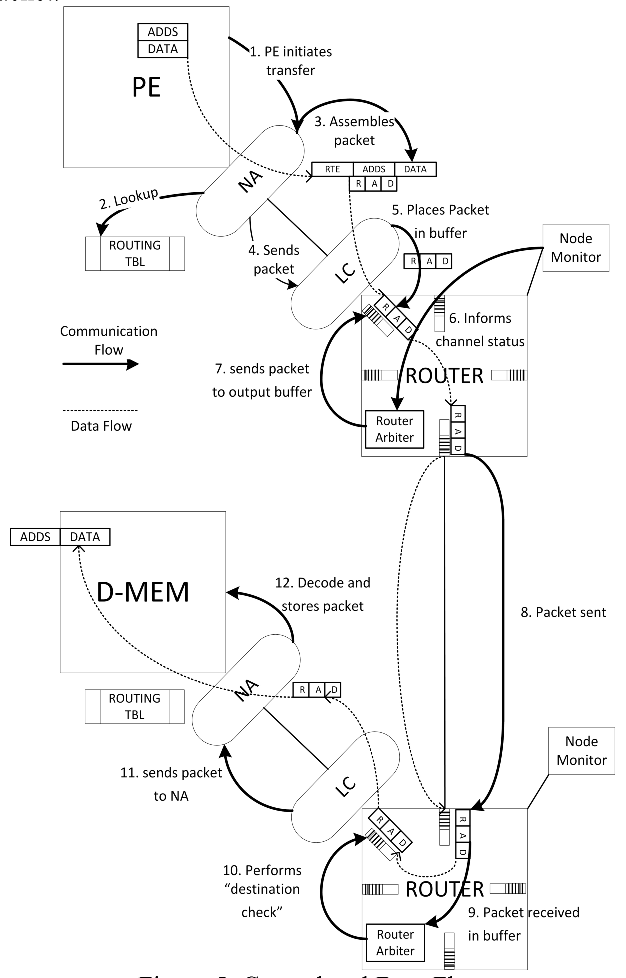


Figure 5: Control and Data Flow

When the packet reaches the input buffer of the adjacent router, the router arbiter performs a "destination check" on the packet to determine if the packet has reached its destination. If the packet had reached its destination, the router arbiter sends the packet to the network adapter. The network adapter decodes the packet into the D-MEM address and the data, and performs the write operation. If the packet has not reached its destination, the router arbiter forwards the packet to the next router in a similar manner as the master router arbiter. In case of channel congestion, the router stores the packet in its buffer until the congestion is eliminated.

4 Evaluations and Results

The parameterizable NoC framework that was developed was implemented in Verilog HDL. The input to the design is a configuration file that determines the topology (torus, mesh, or WK-recursive), switching (Store and Forward or Wormhole), size, depth and the number of virtual channel buffers, size of packet, and type of ECC (odd or even parity). The parameters are static, which means the configurations cannot be modified during simulation or after synthesis. Also, different architectural components of the framework can operate at different frequencies to mimic real-world applications and real-time traffic. These characteristics can be modeled by modifying the NoC configuration file. Table 1 summarizes the most important features that can be parameterized in the framework.

Table 1: Parameterizable features in the NoC framework

NETWORK TOPOLOGIES	LINEAR, MESH, TORUS, WK-RECURSIVE
Channel Width/Packet size	22-80 bits
Flow Control	ON/OFF
Switching mode	SF and Wormhole
Routing Algorithm	XY for Mesh, Torus, Simplex Routing Algorithm for WK-recursive
Buffering	Input Buffering, Single or Multiple Virtual Channels
Network Synchronization	Synchronous
Virtual Channels	Depth, size and number of virtual channels are parameterizable
Traffic Patterns	Uniform, Sporadic, Hotspot

A packet flow/traffic can be initiated either by the Traffic Generator (TG) or by the master PE. OR1k has a GNU tool chain, including the GCC compiler and the GNU debugger. The application software can be loaded in the external instruction memory of the corresponding master PE. The processor uses on-chip RAM to execute a bootloader, in which the cache, stack, and MMU are enabled and initialized. After initialization, the master PE interacts with other PEs in framework, thereby executing the application. The complete emulation framework is presented in Figure 6.

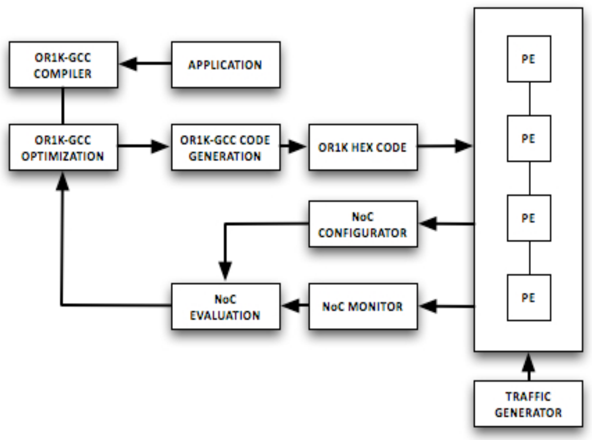


Figure 6: The complete NoC Emulation Framework

The NoC network clock (Router node) is operated at 1GHz. Framework consists of several master and slave PEs. Master PEs (OR1K+IMEM) operates at 250MHz. The slave PEs, which includes: TIMER, UART, and normal data memory unit operate at 125MHz, a D-MEM operates at 250MHz (Slow Mode) or 500MHz (Fast Mode). Congestion scenarios (hot-spot) are simulated when a faster PE sends a series of packets to a slower PE. Regular traffic scenarios are simulated when a faster/slower PE sends a series of packets to a faster PE. Also traffic scenarios for single and multiple hop transfers are modeled. Figure 7, shows the different traffic scenarios modeled in the proposed NoC framework.

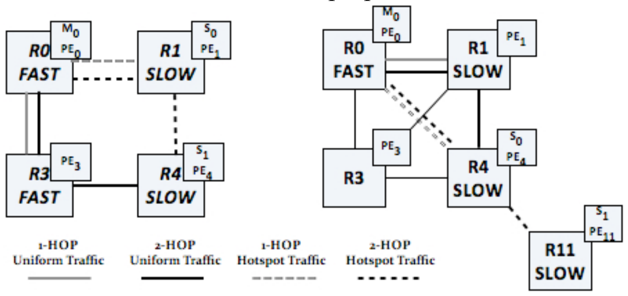


Figure 7: Congestion Traffic Models

In order to evaluate the flexibility and diversity of the proposed NoC framework, the following design variations are considered. First, the 3x3 Torus topology with SF switching and XY routing algorithm is evaluated. Second, the WK(2,4) topology with Wormhole(WH) switching and minimum routing algorithm is evaluated. Evaluation of WK topology includes support for burst data transfer. A master sending multiple data to same destination can be packed into a single packet in WH, hence further reducing latency.

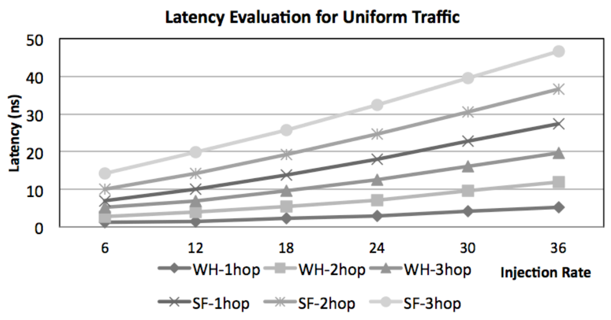


Figure 8: Latency Evaluation of Torus Topology

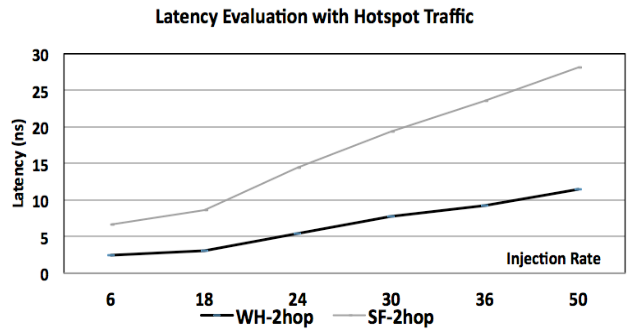


Figure 9: Latency Evaluation of WK Topology

The performance metrics of these design variations include: latency vs injection rate (packets/clock cycle), logic area (number of slices) and power (in watts). The designs were also evaluated for performance metrics under normal and hotspot traffic conditions. Figure 8 shows the latency for Torus and WK topologies for uniform traffic, and Figure 9 shows the latency for hotspot traffic for different hops. The design was synthesized for Virtex4, using Xilinx ISE 11.1. Table 2 shows the area and static power comparisons of different NoC architecture components for the Torus and WK topologies.

Table 2: Area and Power of Torus and WK

	Area (Slices)		Power (W)	
Block/Module	WH	SF	WH	SF
PE-OR1K	5992	5992	0.38216	0.38216
Router	1670	1778	0.17106	0.19048
Core Interface	65	543	0.17948	0.16633
Network Interface	37	69	0.16183	0.16893

5 Conclusion

A synthesizable NoC framework was developed using Verilog HDL. The framework is parameterizable, and has been used as a tool for design space exploration of various topology (Torus and WK), switching techniques (SF and WH), and network traffic (uniform and hot-spot). The performance metrics of the design space exploration include latency, area, and power. By using the proposed framework, researchers will be able to evaluate and compare various novel NoC architectures and algorithms with accurate performance, power, and area parameters, and hence facilitate the determination of system-level performance.

6 References

[1] OpenRisc 1000 architecture Manual from <opencores.com>

[2] A. Kumar et al; “Toward Ideal On-Chip Communication Using Express Virtual Channels”. IEEE Micro Vol 28, Issue 1, January- February 2008, pp 191-202.

[3] D.Mattsson and M. Christensson , “Evaluation of Synthesizable CPU Cores” master’s thesis 2004.

[4] https://benchmark.stanford.edu

[5] Krishnan Srinivasan, Karam S. Chatha, and Goran Konjevod “Linear-Programming-Based Techniques for Synthesis of Network-on-Chip Architectures”. In *Proceedings of the IEEE International Conference on Computer Design* (ICCD '04). IEEE Computer Society, Washington, DC, USA, 422-429.

[6] Juan del Cuvillo et al.: FAST: A Functionally Accurate Simulation Toolset for the Cyclops64 Cellular Architecture. MoBS’05 Workshop in conjunction with ISCA’05, 2005.

[7] Hangsheng Wang et al.: Orion: A Power-Performance Simulator for Interconnection Networks. In Proceedings of MICRO 35, 2002.

[8] Zhonghai Lu, Rikard Thid, et al.: NNSE: Nostrum network-on-chip simulation environment. Design, Automation and Test in Europe Conference, 2005.

[9] Noxim. http://sourceforge.net/projects/noxim, 2008.

[10] Lavina Jain et al.: NIRGAM: A Simulator for NoC Interconnect Routing and Application Modeling. Design, Automation and Test in Europe Conference, 2007.

[11] CellSim. http://pcsostres.ac.upc.edu/cellsim, 2007.

[12] V. S. Pai et al., “RSIM: Rice Simulator for ILP Multiprocessors,” SIGARCH Comput. Archit. News, vol. 25, no. 5, p. 1, 1997.

[13] V. Puente et al., “Sicosys: An integrated framework for studying interconnection network performance in multiprocessor systems,” Parallel, Distributed, and Network-Based Processing, Euromicro Conference on, vol. 0, p. 0015, 2002.

[14] Y. Hu, H.Chen, Y.Zhu, A. A. Chien and C. Cheng, "Physical Synthesis of Energy-Efficient Network-on-Chip Through Topology Exploration and Wire Style Optimizations," Design (ICCD), pp.111-118, 2005.

[15] K. Srinivasan and K.S. Chatha, “A technique for low energy mapping and routing in network on chip architectures”. In Proceedings of the 2005 international symposium on Low power electronics and design (ISLPED '05). ACM, New York, NY, USA, 387-392.

[16] G. D. Vecchia and C. Sanges, “A recursively scalable network VLSI implementation,” Future Generation Computer Systems, 4(3) 235-243, 1988.

[17] D.Rahmati, A.Kiasari, S.Hessabi, H.Sarbazi-Azad, "A Performance and Power Analysis of WK-Recursive and Mesh Networks for Network-on-Chips", IEEE International Conference on Computer Design (ICCD 2006), San Jose, CA, USA, Oct. 2006.

[18] Della Vecchia, G., Sanges, C.: A Recursively Scalable Network VLSI Implementation. Future Generation Computer Systems 4(3), 235–243 (1988).

[19] M. Dallosso et al., “Pipes: A Latency Insensitive Parameterized Network-on-chip Architecture for Multi-Processor SoCs,” pp. 536-539, Proc. Int’l Conf. Computer Design, 2003.

Demand Based Routing in Network-on-Chip(NoC)

Kullai Reddy Meka and Jatindra Kumar Deka
Department of Computer Science and Engineering, Indian Institute of Technology Guwahati, Guwahati, India

Abstract—*In this paper, we present new routing technique called "Demand based routing" for mesh Network-on-Chip(NoC) which improves throughput in terms of tasks. We are proposing this because usually different nodes/tiles on NoC perform different tasks which intern may depend themselves. Various existing routing algorithms for Network-on-Chip(NoC) are improving throughput in terms of packets, global delay and energy consumption etc. In all these algorithms sender node is the initiator of packet transmission. Our proposal of "Demand based routing in NoC" aims to improve throughput in terms of tasks, where packet transmission can be initiated by receiver node along with sender node based on situation. So in this schema receiver can request a packet(which we termed as "demand") based on situation. This routing technique improves task completion time of a node through demanding the required packet.*

Keywords: Demand Based Routing, Network-on-Chip, NoC Routing, System-on-Chip, NoC, SoC.

1. Introduction

The regular tile-based NoC architecture was proposed as a solution to the complex communication problems of System-on-Chip[1]. NoC concept nicely separates the concerns of computing and communication. It is hoped that NoC will provide solutions to increased system complexity and declining system productivity. Several researchers have suggested that a 2-D mesh[2], [3] architecture for NOC will be more efficient in terms of latency, power consumption and ease of implementation, as compared to other topologies. Most common NoC topologies are mesh and torus which constitute over 60% of cases[2]. Due to these two reasons the platform under consideration is composed of a $n \times n$ array of tiles which are inter-connected by a 2D mesh network.

NoC Routing algorithms can be generally classified into oblivious and adaptive[4]. In oblivious routing algorithms the path to route a packet/flit is determined by the source and the destination address, but in case of adaptive routing the path to route a particular packet with source and a destination address depends on dynamic network conditions (e.g.congested links due to traffic variability).

The main advantage of using deterministic routing is its simplicity of the routers design. Because of the simplified logic, the deterministic routing provides low latency when the network is not congested. However, as the packet injection rate increases, deterministic routers are likely to suffer from throughput degradation as they can not dynamically respond to network congestion. In contrast, adaptive routers avoid congested links by using alternative routing paths, this leads to higher throughput. However, due to the extra logic needed to decide on a good routing path, adaptive routing has a higher latency at low levels of network congestion.

Jingcao Hu and Radu Marculescu has proposed a routing algorithm called Dynamic Adaptive Deterministic routing(DyAD) for NoC which combines the advantages of both deterministic and adaptive routing schemes[5]. This DyAD algorithm consists both deterministic and adaptive algorithms, switches between these two based on the network's congestion conditions.

Nodes/tiles of NoC may generate huge amount of data for communication as complexity of applications is increasing day by day. Due to the increase of the data, the packet size will increase and to handle the bigger packet, the complexity of the router will also increase. The amount of data generated by a node will be treated as message and the message will be divided into several packets and packets will be transmitted flit by flit. In this scenario, it may happen that one node of NoC receives few packets of a message and waiting for other packets of the same message to arrive. To reduce the waiting time by a node, the concept of demand based routing is introduced in this paper. When a node is waiting for some packets of a message, it can raise the demand for those packets. Once the router of a node receives a demand flit for a packet, and if the packet is available in the buffer of that router, then priority will be given to that packet and it will forwarded accordingly.

The major advantage of the Demand based routing is to improve delivery time of messages which intern improves throughput in terms of tasks as tasks depends on these messages. We proposed this Demand based routing for NoC by adding demanding a packet/flit and supplying a packet/flit to DyAD routing.

Rest of the paper is organized as follows. In next section we review the related work. In section 3, we present the Demand based routing algorithm and its router architecture. Analysis is done in section 4. Finally, we conclude in the last section.

2. Related Work

There have been various routing algorithms proposed for NoC that are based on wormhole routing in the literature. Most of the algorithms given in this section for NoC mesh topology. Majority of these routing protocols are classified into two classes they are Oblivious and Adaptive routing algorithms[4]. These algorithms need to handle deadlock, live lock and starvation.

2.1 Oblivious Routing Algorithms

Oblivious routing algorithms have no information about conditions of the network, like traffic amounts or congestion. A router makes routing decisions on the grounds of some algorithm or for example randomly. There are three types of Oblivious Routing Algorithms, they are Dimension order routing, Turn Models and Deterministic Routing Algorithms.

Dimension order routing (DOR) is a typical minimal turn algorithm. The algorithm determines to what direction packets are routed during every stage of the routing[6]. XY-Routing[7], Pseudo Adaptive XY Routing[7] and Surrounding XY Routing[4] comes under this category.

Turn model algorithms determine a turn or turns which are not allowed while routing packets through a network such that network is livelock free[8][4]. West-first Routing, North-last Routing and Negative-first Routing[4][9] comes under this category.

Deterministic routing algorithms route packets every time from a particular source to particular destination along a fixed path. A router makes routing decisions on the grounds of pre-collected data and some algorithm. There are various deterministic routing algorithms they are Distance vector, Link state, Source routing, destination tag routing and Stochastic Routing algorithms. Deterministic algorithms are used in both regular and irregular networks, in congestion free networks deterministic algorithms are reliable and have low latency[4]. They suit well on real time systems because packets always reach the destination in correct order and so a reordering is not necessary. In the simplest case each router has a routing table that includes routes to all other routers in the network. When network structure changes, every router has to be updated. Deterministic routing is not reliable in congested networks because all flits from particular source to particular destination will follow same path which creates unbalanced load in network.

2.2 Adaptive Routing Algorithms

These algorithms consider information about conditions of the network, like traffic amounts or congestion while routing. A router makes routing decisions on the grounds of network condition and some algorithm. Adaptive Routing Algorithms can be classified into two categories, they are Minimal Adaptive Routing and Fully Adaptive Routing.

Minimal adaptive routing algorithm always routes packets along the shortest path. The algorithm is effective when more than one minimal or as short as possible, routes between sender and receiver exist. The algorithm uses route which is least congested[6]

Fully adaptive routing algorithm uses always a route which is not congested. The algorithm does not care although the route is not the shortest path between sender and receiver. Typically an adaptive routing algorithm sets alternative congestion free routes to order of superiority. The shortest route is the best one[6]. Various Fully adaptive algorithms are Odd-even routing, Q-routing and New Dynamic Routing, Turnaround Routing, IVAL, 2TURN, slack-time aware and Hot-Potato Routing Algorithms [4].

Deadlock in wormhole routing[13] is caused by packets waiting on each other in a cycle[13]. Previous methods, such as the turn model and the XY algorithm, avoid deadlock by prohibiting certain turns. Instead, the odd-even turn model is based on restricting the locations at which certain turns can be taken so that a circular wait can never occur. Odd-even routing is a deadlock free turn model where circular wait never as it prohibits turns from east to north and from east to south at tiles located in even columns and turns from north to west and south to west at tiles located in odd columns. In odd-even routing adaptiveness is more when compared with XY-routing or turn model algorithms. It will work well in the congested networks[10].

Q-routing algorithm is based on the network traffic statistics. The algorithm collects information about latencies and congestion, and maintains statistics about network traffic. The Q-routing algorithm does the routing decisions based on these statistics[11]. Collecting these statistics create more overhead in NoC, so because of overhead it is using very less in NoC[11], [4].

New Dynamic Routing algorithm proposed in the year 2007[12]. It combines two partial adaptive algorithms, North-Last and South-Last routing algorithms, to achieve better results. The algorithm says that if more than one packet arrive at the same time and require the same destination, one packet passes while others wait till the destination is free, then they obtain the link in the priority. Thus they are preventing starvation[12].

Turnaround Routing, IVAL, 2TURN, slack-time aware and Hot-Potato Routing Algorithms are not using much as these are having overheads in implementation on on-Chip[4].

2.3 Other Routing Algorithms

DyAD: Dynamic Adaptive Deterministic switching(DyAD) uses both Deterministic and Adaptive algorithms. It uses XY and Odd-even routing. Depends on the Network condition it switches between these two algorithms. It improves performance but have little overhead, so depends on application requirement we can choose[5]. In this paper, we are adding Demanding a packet/flit and supplying a packet/flit of demand to DyAD as now a days we can have good amount of buffer space(Improvements in hardware is very high).

3. Proposed Methodology

Proposed work for demand based routing for NoC comprises of two modules, they are Transmitting module and Receiving module. Each router in the NoC should have these modules in order to have the demand based routing. Some preliminaries to Demand based routing are given in the next subsection, before going to details of these modules.

3.1 Preliminaries

Packets and Flits: In NoC application level datagrams are known as messages, node level datagrams are known as packets. These packets are transmitted in units of flits. Fig. 1 shows message which divided into packets, which intern divided into flits. So flits are smallest units in which packets are transmitted between two nodes.

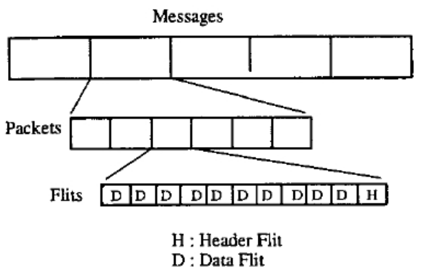


Fig. 1: Message format

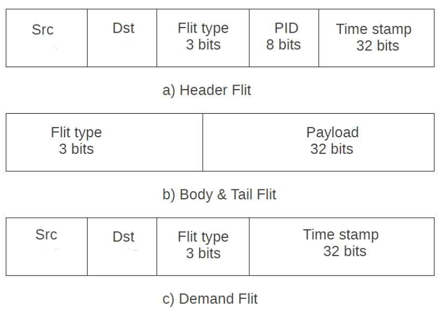


Fig. 2: Flit formats

Usually a packet is divided into Header flit, Body flits and Tail flit. Only header flit consists of routing information and it indicates starting of packet as it uses NoC wormhole switching[13]. Tail flit indicates end of packet. We are introducing new flit for our work which is known as Demand flit. This Demand flit is used to send request, when a node requires a packet(s) of particular message from particular node. Formats of these flits are given in the Fig. 2.

Flit source and destination fields sizes depends on NoC mesh size, for a 8×8 mesh NoC 6 bits are required to represent a node. Flit type which is a 3 bit field and router considers this before going to take routing action. Various flit types are given in the Table 1.

Demand Matrix: Router uses a Demand Matrix of size $Number\_of\_nodes\_of\_mesh \times Number\_of\_nodes\_of\_mesh$ in order to maintain demands raised so far, where each entry of Demand Matrix is either 0 or 1 to indicate whether a demand is raised or not. So the memory required for this matrix is $Number\_of\_nodes\_of\_mesh \times Number\_of\_nodes\_of\_mesh$ bits. Initially all the entries of this matrix are 0. These are updated to 1 whenever Demand ON flit is raised and 0 whenever Demand OFF flit is raised. It can be checked in unit time whether a demand between particular source and destination is raised or not.

Table 1: Various Flit Types

FLIT TYPE VALUE	MEANING
000	Header Flit
001	Body Flit
010	Tail Flit of Packet and saying still Packets are there
011	Tail Flit and No more Packets of Message
100	Demand ON Flit
101	Demand OFF Flit

Packet Matrix: Router uses a Packet Matrix of size $Number\_of\_nodes\_of\_mesh \times Number\_of\_nodes\_of\_mesh$ in order to maintain what are the packets present in router buffers so far, where each entry of Packet Matrix is either 0 or 1 to indicate whether a packet from a particular source to destination is present or not. So the memory required for this matrix is $Number\_of\_nodes\_of\_mesh \times Number\_of\_nodes\_of\_mesh$ bits. Initially all the entries of this matrix are 0. These are updated to 1 whenever packet header flit is kept in router buffers and 0 whenever packet header flit is moved from router buffers to router output ports. It is used to check whether a packet from a particular source and destination is there or not(in router buffers) in constant time, which intern helps in taking the decision of whether a demand is need to forward further or not(when it raised).

Router Architecture: Fig. 3 illustrates the router architecture of Demand based routing. DyAD router[5] is reference router for our Demand Based router. Each cardinal buffer is collection of three buffers, general buffer, Demanded_packets_buffer and Demand_flits_buffer. Demanded_packets_buffer is only for demanded packet's flits and Demand_flits_buffer is only for demand ON and OFF flits.

Whenever a flit is coming into router the port controller decides its buffer by considering its flit type, Demand Matrix and Packet Matrix. Matrices controller updates matrices based on incoming flits and it also controls matrices accession by port controllers.

The Crossbar Arbiter maintains the status of the current crossbar connection and determines whether to grant connection permission to the port controller. When there are multiple input port controllers requests for the same available output port, the Crossbar Arbiter uses the first-come-first-served policy to decide which input port to grant the access, such that the starvation at a particular port can be avoided.

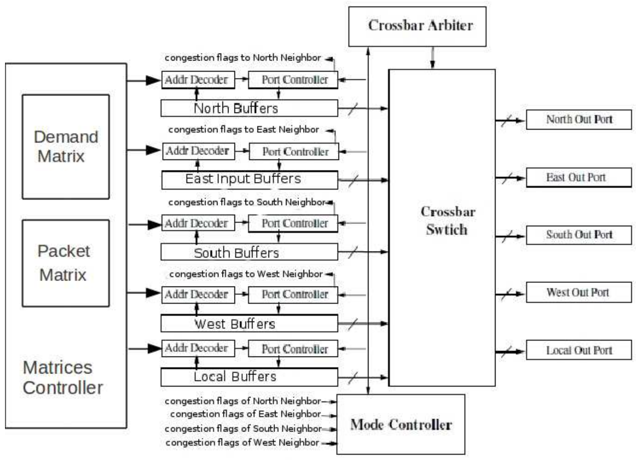

Crossbar Arbiter
Demand Matrix
Packet Matrix
Matrices Controller
congestion flags to North Neighbor
Addr Decoder
Port Controller
North Buffers
congestion flags to East Neighbor
East Input Buffers
congestion flags to South Neighbor
South Buffers
congestion flags to West Neighbor
West Buffers
Local Buffers
Crossbar Swtich
North Out Port
East Out Port
South Out Port
West Out Port
Local Out Port
congestion flags of North Neighbor
congestion flags of East Neighbor
congestion flags of South Neighbor
congestion flags of West Neighbor
Mode Controller

Fig. 3: Router Architecture

Each port controller monitors buffer ratios and if ratio reaches threshold of that buffer then it set corresponding congestion flag to indicate that buffer has reached threshold.

The Mode Controller continuously monitors its neighboring congestion to determine either deterministic or the adaptive routing mode of DyAD(our Demand based routing intern uses DyAD). If any congestion flag from its neighboring routers are asserted, then the Mode Controller commands all the input port controllers to work in the adaptive mode. Otherwise, it switches the port controllers to the deterministic mode.

Now Demand based routing is simple, whenever a node is waiting for packets from a particular source it will raise demand by doing Restricted broadcasting of demand ON flit. When a node receives a demand ON flit, it will set the corresponding position of Demand Matrix to 1 and broadcast the Demand ON flit using restricted broadcasting algorithm if the node is not the destination one. While receiving a packet by a node, it checks against Demand Matrix. If there is demand then put the packet into corresponding demanded_packet_buffers else put it into corresponding buffer and update Packet Matrix.

While transmitting, first transmit Demanded Packets if any and then normal packets. Once a node receives required packets then it will do Restricted broadcasting of demand OFF flit in order to tell that received required packets. In general broadcasting is costly but here we proposed a Restricted broadcasting technique for mesh topology which is enough to Demand based routing. This Restricted broadcasting is given in Transmitting Module subsection of proposed work.

To achieve this(Demand based routing) each router need to have two modules, Receiving Module and Transmitting Module. These two modules are described in next subsections.

3.2 Receiving Module

Every router should have receiving process, which handles all the incoming flits. In this algorithm when a flit coming into router, depends on its type the Receiving Process algorithm will update all corresponding Demand Matrix entries, Packet Matrix entries and buffers. This reduces searching time of transmitting process for any raised Demand and Packet.

Receiving Process Algorithm

```
Input: Incoming flit
Output:Updated Demand Matrix, Packet Matrix
and Buffers
Begin if Header Flit then
    if corresponding Demand Matrix entry then
        Put flit in corresponding Demanded_packets_buffer
        & Update buffer level.
    else
        Update related Packet Matrix entry to 1. Put flit in
        corresponding buffer(based on in direction) &
        Update buffer level.
    end
    if corresponding buffer is full then
        Set corresponding congestion_flag.
    end
else
    if Demand Flit then
        Update related Demand Matrix entry to 1. if
        Demand_dest_id != local_id then
            Put Demand flit into corresponding
            Demand_flit_buffer(Based in-port) & Update its
            buffer level.
        end
    else
        Put flit into corresponding buffer(wormhole
        switching) & Update buffer level. if corresponding
        buffer is full then
            Set corresponding congestion_flag.
        end
    end
end
End
```

According to the Receiving Process algorithm, if incoming flit is header flit of a packet then check against Demand Matrix and if there is demand then put it into Demanded_packets_buffer of that direction, else put the flit into corresponding buffer(based on incoming direction), update Packet Matrix and buffer levels. If incoming flit is demand flit, update corresponding Demand Matrix entry to 1 and put demand flit into corresponding Demand_flit_buffer(based on in-port). Suppose the incoming flit is demand flit and its destination Id is equal to local Id then just update corresponding Demand Matrix entry to 1.

Finally if flit is either body flit or tail flit, simply put it into corresponding buffers based on directions and update the buffer level.

3.3 Transmitting Module

Every router should have Transmitting process in order to control all the outgoing packets. Transmitting module first handles Demand flits if any, then Demanded packets if any and then normal packets. This algorithm intern uses two functions, these are Broadcast algorithm and DyAD routing algorithm[5]. Broadcast algorithm to broadcast demand flit and DyAD routing algorithm for getting the path(route) or direction for forwarding data flits of packets. DyAD routing algorithm applies either adaptive or deterministic routing algorithm to determine route or direction to forward flit by considering network congestion status[5].

Transmitting Process Algorithm

```
Input:Buffers, Demand and Packet Matrices
Output:route to forward(Direction)

Begin if current router want to generate Demand flit then
    Generate Demand flit and put into NORTH or SOUTH
    Demand_flit_buffer.
end
if Demand_flit_buffers has Demand flits then
    Restricted Broadcast of those flits
end
if If last received flit is tail flit of any demanded packet then
    Restricted Broadcast of Demand OFF flit of that packet
end
if last transmitted flit == tail flit then
    if Demanded_packets_buffers has demanded packets then
        Get route by applying DyAD to header flit. Send
        remaining flit of that packet in the same route by
        wormhole switching.
    end
else
    if current flit == tail flit then
        Send tail flit by wormhole switching. Update related
        entry of Packet Matrix to 0
    else
        Send body flit by wormhole switching.
    end
end
End
```

The transmitting module handles flit in order- first Demand flits, then Demanded Packets and then General Packets. It also update corresponding buffer level which is obvious thing done by transmitting module after transmission of any flit.

Restricted Broadcasting Algorithm:

In Demand Based Routing, broadcasting is needed only to broadcast demand ON and OFF flits. Demand flits need to broadcast as we don't know where the required packets are resided in the mesh because routing path is decided dynamically. In Computer Networks broadcasting is costly, but in NoC we are making it not much costly as we already know the topology of it.

Here restricted means we are restricting broadcasting to particular area based on relative position of destination with source of demand flit as the probability of availability of required packet in that are is high. An example of Restricted Broadcasting and general way of Broadcasting from node(3,3) to node(1,1) is given in the Fig. 4 and Fig 5, which clearly shows the differences between Restricted Broadcasting algorithm and general way of broadcasting. Restricted Broadcasting Algorithm is very simple, we are just finding where the destination lies(means Upper or lower and left or right) from current node by comparing their coordinates. Once we got position then based on the incoming port of that demand flit we are deciding outgoing ports for broadcasting.

Restricted Broadcast Algorithm

```
Input:Demand flit for Restricted broadcast
Output:Restricted Broadcast

Begin if (Current id == Destination id) or related entry of
Packet Matrix is 1 then
    Stop forwarding Demand flit. Remove it from its
    Demand_flit_buffer.
else
    if Demand_flit_buffer is EAST or WEST then
        if Destination.x<Current.x then
            Make a duplicate of Demand flit and it towards
            WEST.
        else
            if Destination.x>Current.x then
                Make a duplicate of Demand flit and send it
                towards EAST.
            end
        end
        Remove it from it's Demand_flit_buffer.
    else
        if Destination.x<Current.x then
            Make a duplicate of Demand flit and send it
            towards WEST.
        else
            if Destination.x>Current.x then
                Make a duplicate of Demand flit and send it
                towards EAST.
            end
        end
        if Destination.y<Current.y then
            Make a duplicate of Demand flit and send it
            towards NORTH.
        else
            if Destination.y>Current.y then
                Make a duplicate of Demand flit and send it
                towards SOUTH.
            end
        end
        Remove it from it's Demand_flit_buffer.
    end
end
End
```

As we considered static mesh topology for NoC, we designed this Restricted broadcasting algorithm. According to Restricted broadcasting algorithm, if the demand flit destination Id is current/local Id (i.e current Id is demanded packet's origin) or required packet is in local buffers (by checking Packet Matrix) then stop broadcasting (further forwarding) of the demand flit. Otherwise, based on the input port of the demand flit into router and the position of the destination, the outgoing port is decided for further broadcasting. Fig. 4 shows an example of this algorithm, broadcasting of demand flit generated by node(3,3) to node(1,1).

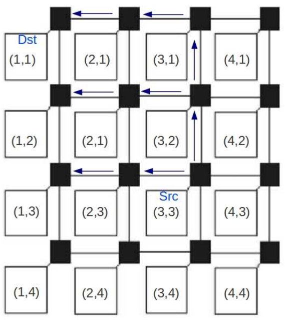


Fig. 4: Restricted Broadcasting from node(3,3) to node(1,1)

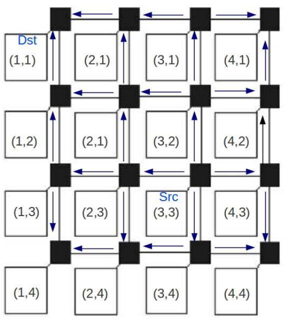


Fig. 5: General way of Broadcasting from node(3,3) to node(1,1)

4. Analysis

In this section we analyze the complexities and performance of our proposed work.

Each router need to maintain Demand Matrix and Packet Matrix. Each of this matrices need $Number\_of\_nodes\_of\_mesh \times Number\_of\_nodes\_of\_mesh$ bits of memory, example is 8×8 mesh needed 64×64 bits which is 512 bytes. Routers need to have Demanded_Packet_buffers and Demand_flit_buffers along with general buffers of routers, so these extra buffers is 25% of general buffers as per our assumptions of our router architecture. So here we can reduce the general buffers by 25% and use it for extra buffer space needed or if we are able to have extra 25% of general buffers then we can put this extra buffers in router. If any application needed this Demand based routing then we should be able to handle this memory.

Packet Matrix is used to decide whether a demand ON flit needs to forward further or not. This can be done in unit time by using Packet Matrix as Packet Matrix maintains the information of any packet about its source and destination. If router doesn't have enough memory we can remove Packet Matrix and it costs some more traffic by forwarding demand ON flit further. The extra traffic without Packet Matrix is less as we are using Restricted broadcasting which is not costly. So depends on memory availability we can decide whether to use Packet Matrix or not and accordingly it will effect the performance.

Router has to update Demand Matrix entry whenever it receives a demand flit, it can be done in unit time by checking addresses in demand flit. Some times router need to update Packet Matrix based on flit type and Demand matrix, it also can be done in unit time. Selection of buffer for incoming flits from general buffer, Demanded_packets_buffer and Demand_flit_buffer(based on flit type and Demand Matrix and Packet matrix) can be done in constant time by flit type and matrices. So overall time overhead is constant.

In our algorithm, a node broadcasts Demand ON and OFF flits for each packet which needs to be demanded. Each demand flit consumes space in Demand_flit_buffer and Demanded packet consumes some space in Demanded_packets_buffer and creates some traffic. So some restriction may be imposed on the number of nodes that can demand a packet and the number of packets that can be demanded by each node. This restriction may vary according to the application run on a NoC.

5. Conclusion

In this paper we have presented a routing technique for NoC where the task of bigger size can be handled and the amount of data transfer between nodes of the NoC is more. For the proper management of data in a router, we require some extra memory for book keeping operation. Since we are handling the data packet wise, the buffer size of router need not to increase more. Since the waiting time has been reduced by introducing demand flit, so the throughput will be improved. It may have some extra overhead due to the broadcast of demand flit, but we have used restricted broadcast to overcome this problem. There is a possibility to reduce the broadcast packet by introducing priority to nodes and only high priority node can demand for a packet. Sometimes this restriction depends on application that is executed in the NoC.

References

[1] W.J. Dally and B. Towles, *Route Packets, Not Wires: On-Chip Interconnection Networks*, Design Automation Conf. (DAC), pp. 683-689, 2001.

[2] Erno Salminen, Ari Kulmala, and Timo, *Survey of Network-on-chip Proposals*, White Paper, OCP-IP, March 2008 2005.

[3] P. Pratim Pande, C. Grecu, M. Jones, A. Ivanov, and R. Saleh, *Performance evaluation and design trade-offs for network-on-chip interconnect architectures*, IEEE Transactions on Computers, vol. 54, no. 8, pp. 1025-1040, 2005.

[4] Ville Rantala, Teijo Lehtonen, Juha Plosila, *Network on Chip Routing Algorithms*, TUCS Technical Report No 779, August 2006.

[5] Jingcao Hu, Radu Marculescu, *DyAD-Smart Routing for Network-sonchip*, ACM Journal, 2004.
[6] J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*, Morgan Kaufmann, 2004.
[7] M. Dehyadgari, M. Nickray, A. Afzali-kusha, Z. Navabi, *Evaluation of Pseudo Adaptive XY Routing Using an Object OrientedModel for NOC*, 17th International Conference on Microelectronics, 13-15 December 2005.
[8] Christopher J. Glass, Lionel M.Ni, *The Turn Model for Adaptive Routing*, ACM Journal, vol 41, 1994.
[9] H. Kariniemi, J. Nurmi, *Arbitration and Routing Schemes for On-chip Packet Networks. Interconnect-Centric Design for Advanced SoC and NoC*, Kluwer Academic Publishers, 2004, pp. 253-282.
[10] Ge-Ming Chiu, *The Odd-Even Turn Model for Adaptive Routing* , IEEE Transactions on Parallel and Distributed Systems, Vol. 11, July 2000.
[11] M.Majer, C. Bobda, A. Ahmadinia, J. Teich, *Packet Routing in Dynamically Changing Networks on Chip* , 19th IEEE International Parallel and Distributed Processing Symposium, 4âĂŞ8 April 2005, pp. 154b.
[12] Mohamed M. Sabry, M. Watheq El-Kharashi, Hassan Shehata Bedor, *A New Dynamic Routing Algorithm for Networks-on-Chips*, Communications, Computers and Signal Processing, 2007. PacRim 2007. IEEE Pacific Rim Conference.
[13] Mohapatra, *Wormhole routing techniques for directly connected multicomputer systems*, ACM Computing Surveys, 30(3), pp. 374-410, September 1998.

Performance Analysis of WK-Recursive and Torus Routing Algorithms for NoCs

Jaya Suseela and Venkatesan Muthukumar
Dept. of Electrical and Computer Engineering, University of Nevada Las Vegas, Las Vegas, NV-USA

Abstract: *Network on Chip (NoC), specific parameters such as topology, switching methods and routing algorithms, have a huge impact on performance and the cost of the NoCs. In this work, we propose a deterministic mirror routing algorithm for Torus topology and an adaptive routing for WK-recursive topology. To evaluate the routing algorithms, a complete NoC synthesizable framework was implemented in Verilog HDL. Congestion scenarios in the NoC framework have been simulated, and the routing algorithms are evaluated for latency, area, and power. Evaluation shows that the performances of the proposed algorithms are superior compared to classical routing algorithms.*

Keywords: NoC, topology, WK-recursive, torus, adaptive routing.

1 Introduction

Networks-on-Chip (NoC) have been proposed as a promising solution to multi-processor on-chip communication problems. To catalyze the deployment of the NoC paradigm for many high performance computational applications, many challenging research problems of NoC design abstractions need to be addressed at all levels. The active problems in the field of NoC design include: design space exploration of NoC architecture for applications, application scheduling and mapping algorithms, evaluation of switching, topology or routing algorithm for efficient execution of application, and optimization of communication cost, area, and power.

The design of the NoC system is divided into four procedural levels of abstraction, or models: i) the Application model, which includes traffic generation and monitoring; ii) the NoC framework architecture model and its components; iii) the Communication Flow model, which models communication among different NoC components; and iv) Algorithm models, which model switching and routing algorithms in the NoC architecture. Figure 1 shows the NoC framework model developed for this evaluation.

The Application model encompasses three main components: i) application mapping and scheduling, ii) a traffic controller, and iii) a monitor. The Architecture model includes: i) the NoC Framework, which includes such processing elements (PEs) as OR1K processors, TIMERs, UARTs, Instruction (IMEM) and data memories (DMEM), and a Network Adapter; ii) the router, and iii) network topology. The Communication Flow model defines the control and data flow in the NoC at the system level, the network level, and the data link level. The system level defines the flow between master to slave PEs, slave to master PEs and between master PEs. The control flow within the router is defined in network layer. The lowest level is the data link level. This level deals with encoding, decoding, and synchronizing packets or flits. The Algorithmic model defines the various switching and routing algorithms used in data and control flow.

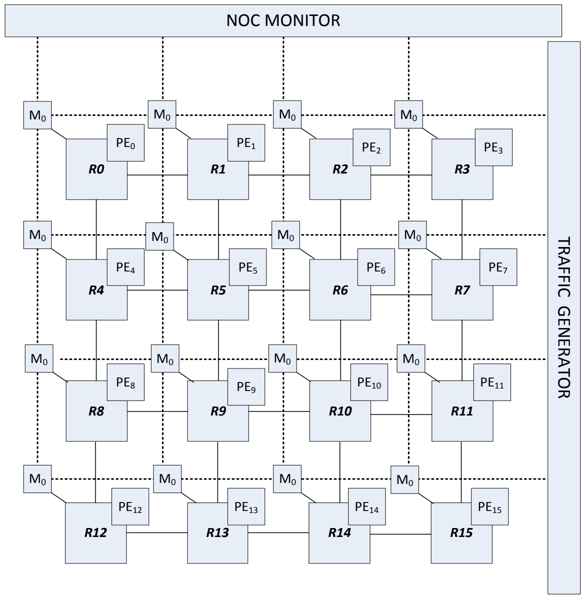


Figure 1: NoC Framework

Latency and power consumption overheads of the NoC system are determined by the communication architecture (router design and topology) and the algorithmic model that is adopted.

In this work, we propose, implement, and evaluate efficient routing algorithms -- both static and adaptive routing algorithms -- for Torus and WK-recursive topologies, respectively. The performances (latency, power, and area) of the proposed routing algorithms are evaluated and summarized.

2 Literature Review

In this section, a brief overview of different topologies and routing algorithms adopted in our NoC framework are presented. The ability of the network to efficiently disseminate information depends largely on the topology. Also, application mapping and routing protocol are dependent mainly on the topology. Linear, Mesh, and Torus [2] are the most widely used homogenous topologies;

however, Spidergon [12] and WK-recursive topologies [6,11] are gaining greater importance.

The WK-recursive networks [6] are a class of recursively scalable networks with many desirable properties. They offer a high degree of regularity, scalability, and symmetry, which conforms very well to a modular design and to the implementation of distributed systems involving a large number of computing elements. WK(d,L) represents a WK topology with node degree d and expansion level L. For this family of topologies, one can start from a WK(d,1) structure and recursively expand and arrive to level L (WK(d,L)). Characteristics of the WK-recursive, Torus, Mesh, and Hypercube topologies are shown in Table 1.

Table 1: WK, Torus, Mesh and Hypercube Topology Characteristics

Network	# N	degree	Diameter	Bisection Width
WK(d,L)	d^L	d	2^L-1	d
Torus	k^2	4	k-1	2k
Mesh	k^2	2-4	2*Sqrt(k)	Sqrt(k)
Hypercube	2*n	n	n	$2^{(n-1)}$

Routing Algorithms

Routing algorithms define the path taken by a packet between source and target switches. They must prevent deadlock, livelock, and starvation [8,9] situations.

A great deal of research has been conducted in creating various NoC routing algorithms. The most common and efficient routing algorithm proposed is the XY routing algorithm. The XY routing algorithm is based on the turn model, which restricts packets moving from Y to X direction and hence avoids a deadlock [7]. West First (WF), North Last (NL), and Negative First (NF) are various partially adaptive routing algorithms based on turn models. For medium to large NoCs, Mello, A. et al. [14] proved that the deterministic XY routing algorithm outperforms partially adaptive North last, Negative first, and West first algorithms. A comparative study done by W. J. Dally [15] proves that full or partial adaptive algorithms are not necessarily beneficial for wormhole routing. Their simulation results indicate that the partially adaptive NL algorithm performs worse than the non-adaptive e-cube routing algorithm for all three traffic patterns (hotspot, uniform, and local). M. H. Ghadiry et al. [13] compared the node load caused by several routing algorithms in presence of a blocking or a fault in a node; they concluded that the e-cube routing algorithm has better load balancing than other algorithms. Compressionless routing prevents deadlock by using fine-grained flow control and back-pressure of wormhole routing.

For WK-recursive mesh topologies, Della Vecchina et. al. [18] proposed a simple deterministic routing algorithm, called the self-routing algorithm. Mostafa Rezazad et al. investigated a deadlock-free shortest path routing algorithm for WK-recursive mesh networks [1]. This algorithm uses the self-routing algorithm [18] and also a second order routing scheme to route the packet. The Multi Path Graph algorithm, proposed by Fernandes et al. [10], is able to tolerate up to d-2 faults; however, it also has increased network traffic due to the d-1 copies send for every message, which is a drawback.

This work proposes: i) a low overhead, mirror routing algorithm for torus topology, classified as source routing, lookup-table-based, deterministic routing algorithm; and ii) an adaptive routing algorithm for WK topology, which is classified as a MUX control based, adaptive routing algorithm. In order to evaluate the proposed routing algorithms, the below described NoC framework was developed.

3 NoC Framework

The NoC framework/system consists of five main modules: i) the processing architecture, ii) the communication infrastructure, iii) the communication paradigm, iv) the monitor module, and v) the traffic generator module.

The processing architecture module consists of a processing lement (PE) and a network adapter module. The communication infrastructure consists of the network topology and the routing node. The communication paradigm describes the switching techniques and routing algorithms employed in the NoC communication infrastructure. The monitor module includes two sub-modules: a node monitor, which monitors the activities in a routing node, and a NoC monitor, which monitors the communication in the framework. The traffic generator (TG) module injects packets (traffic) into the network. It can initiate either a request or a start of transmission from the top level. The TG also determines the type of traffic (uniform, hotspot, sporadic) as well as the source and destination nodes for traffic flow. Different congestion scenarios and node failures can also be created through the TG. The design consists of a node monitor at each router and an NoC monitor at the network level (see Figure 1). The transaction monitor at each router contains information about the buffer count in each virtual channel and sends this information to the top NoC monitor and traffic controller. It also keeps track of PE status.

The NoC processing architecture consists of several master/slave processing elements (PEs) that are connected to the communication infrastructure by means of a network adapter. The PEs can be a master PE or slave PE, depending on whether it can initiate a message transfer or only respond to a request. Only master PEs can initiate a message transfer. Slave PEs respond to the requests from master PE either by sending back the requested signals/data or by saving the received information. UART, TIMER, and Instruction/Data Memory all are considered slave PEs, whereas the master PEs used in the design are capable of performing arithmetic and logical operations. The network adapter receives signals from the PEs and generates packets to be sent to the communication infrastructure. Hence, the main function of the adapter module is to transform the data to and from the format required by underlying infrastructure.

The data/message is communicated as packets. The entire message can be either generated as a single packet or the packets can be divided into flits before actually transmitted. The packet format is shown in Figure 2.

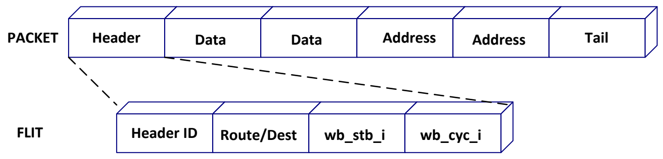


Figure 2: Packet Format

Routing Node

Routing nodes connect each PE in the desired topology. Routing logic consists of three components: i) a link cntroller ii) channels/links, and iii) a router arbiter. Routing nodes route the message/packets/flits according to the adopted switching technique and routing algorithm. Each router in Torus and Mesh topology has five input and five output ports: four input ports from the four cardinal directions (North, East, South, and West) and one (ip_*) from the local processing wlement (PE). In WK topology, router nodes have three to four input/output ports, depending on the position of router in network. The ports are named as North (noc_n*), South (noc_s*), Across (noc_a*), Over (noc_o*), East (noc_e*), and West (noc_w*). The ports to South-West or South-East are called Across ports and those to North-West and North-East directions are known as Over ports.
Figure 3, shows the network router module with virtual channels and a link controller.

4 Routing Algorithms

This work proposes two novel, low-overhead routing algorithms: 1) mirror routing for torus topology, based on turn model routing; and 2) an adaptive routing algorithm for WK-recursive mesh topology, based on the minimal routing algorithm.

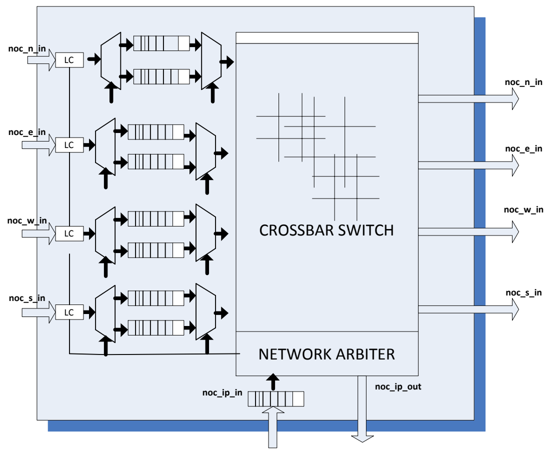


Figure 3: Network Router and Channels

XY Routing

This is the most common and efficient routing algorithm. The framework implements the XY routing for torus topology. Consider the scenario (Figure 4), in which the processing element PE3 wishes to send packets to PE7 by the XY routing algorithm; packets will flow from R3 to R4 (R3–R4) and then to R7 (R4-R7). In cases where either of the links R3-R4 or R4-R7 is congested, or if PE R4 fails, then the packets will stall until resources become available.

Mirror Routing

This section explains a deterministic routing algorithm for torus topology called “Mirror Routing” (MR). It exploits the path diversity property of the torus structure and hence many routing possibilities. The mirror routing implemented is an incremental extension of the XY routing algorithm. The MR algorithm (based on Store and Forward switching) is implemented to compare the performance of the adaptive routing algorithm (based on wormhole switching) for WK topology.
When the torus network is path diversified (or path multiplied, where several parallel lanes exist in each row or column), the address encapsulation mechanism can be used to take advantage of the path diversity while preserving the simplicity and obliviousness of dimension order routing [16]. In mirror routing, switching is implemented based on SF switching mechanism; lookup table based routing implementation and source routing for simplicity. Hence, the source node has a routing table that includes the entire path the packet transverses before it reaches the destination node. If a link is congested or failed, the node monitor informs the NOC monitor through READY and FULL signals. The monitor can even inform the onset of congestion by considering the present buffer occupancy. When the network arbiter (NA) of any routing node receives congestion/failure information from an adjacent node or from the NOC monitor, it chooses an alternative route that is opposite in direction to the normal route. This is done by including an alternative exit path in the routing table in the routing node. The area overhead incurred due to the inclusion of this alternative exit path is negligible.
Consider the scenario where PE3 has to send packets to PE7, and link R3-R4 is congested. Mirror routing will route the packet through R3-R5 and R4-R7, shown by arrows in Figure 5. Similarly, if PE4 has to send packets to PE2 and link/channel R4-R1 is congested, the packets will get redirected through R7.
Since the Mirror Routing follows the XY turn model, mirror routing is also deadlock-free. The 3D Torus network offers a high degree of path diversity, and therefore is relevant in such topologies.

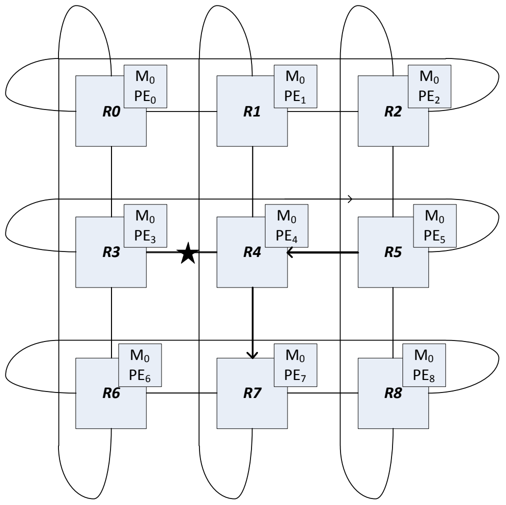


Figure 4: Mirror Routing

Minimal Routing for WK Topology

A WK(d,L) network contains exactly $d^{(L-1)}$ different WK(d,1) sub-networks. Each WK(d,1) is connected by a complete graph, and the link between any two sub-networks is referred to as a flipping link. The flipping links are shown by dotted lines in Figure 5. Every Extern nodes has 3 ports. All nodes except Extern nodes have 4 links/ports.

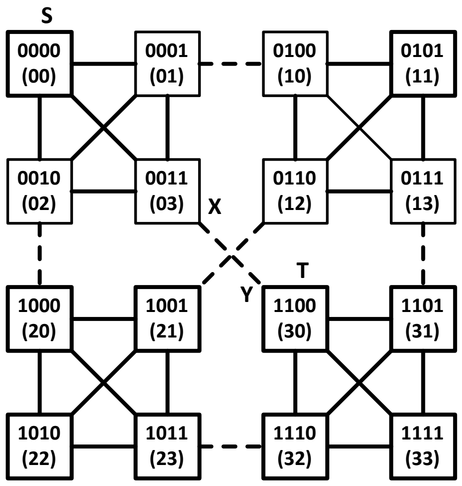


Figure 5. Node Addressing in WK Topology

In WK topology, each node is addressed, as shown in Figure 5. In a deterministic minimal routing algorithm [11], only the destination address is required to deliver a message. Suppose the node S (00) has to send data to node T (30). A routing path between them can be constructed as follows:

- Step 1: Compare the MSB of the node addresses of S and T. If they are the same, the destination node is in same sub-network. In this case, the routing becomes elementary, as each node in the sub-network is interconnected to every other node. Here they are 0 and 3. Hence, the nodes are not in the same sub-network. If they are in a different sub-network, proceed to Step 2.
- Step 2: Determine the flipping edge. The flipping edge (X, Y) is the bridge between the two sub-networks. In our example, the flipping edge is the edge between (03) and (30).
- Step 3: Determine the routing path from S to X and the routing path from X to T. The routing path from S to T is the concatenation of the routing path from S to X, the flipping edge (X, Y), and the routing path from Y to T. Hence the path is (00),(03),(30).

Adaptive Routing (AR) for WK Topology

A new adaptive routing algorithm is proposed for WK topology, based on priority routing. Depending on the destination, different routing paths are chosen based on availability and priority, as shown in Table 2.

Table 2: Priority Routing Table

Source	Des/1st P	2nd P	3rd P
IP/O/N/E/W	A	S	E/W
S	A	W	E
IP/O/A/N	S	E	A
E/W	S	A	W/E
IP/O/A/N	E	S	A
S	E	A	N

A = Across, N= North, S= South, E= East, W= West, O= Over.

Consider the scenario in which the source node (00) has to send data to (30) and the link (00)-(03) is congested. According to first row in the priority table, when the packet from PE (IP) has to go to Across (A) and if Across is congested, then the second priority of South (S) is selected. Hence, the packet from (00) is routed to (02). Thereafter, it follows the shortest path. Therefore, the new path that the packet will follow is given as (00),(02),(03),(30).
The adaptive routing algorithm requires virtual channels to provide deadlock-free routing. To provide deadlock free routing, a proposed policy for pHOP virtual channel selection [11] is adopted in this work.. Each node is assumed to know the status of its neighbors.

5 Synthesis Results

The NoC framework was implemented in Verilog HDL. The framework is completely parameterizable. The user can parameterize the topology, switching, depth and size of the channels, packet structure, and router buffer size. The framework also includes a synthesizable PE. Different architectural components of the framework can operate at different frequencies to mimic real-world applications and real-time traffic. These characteristics can be modeled by modifying the NoC configuration file.

A packet flow/traffic can be initiated either by traffic generator (TG) or by the master PE. The NoC network clock (router node) is operated at 1GHz. Master PE operates at 250MHz. The slave PEs, which include TIMER, UART, and an instruction memory unit(I-MEM), operate at 125MHz; a D-MEM operates at 250MHz (Slow Mode) or 500MHz (Fast Mode). Congestion scenarios (hot-spots) are simulated when a faster PE sends a series of packets to a slower PE. Regular traffic scenarios are simulated when a faster/slower PE sends a series of packets to a faster PE. Also, traffic scenarios for single and multiple hop transfers have been modeled.

In order to evaluate the performances of the proposed routing algorithms, the following design and synthesis variations are considered: 1) the evaluation of the 3x3 and 4x4 Torus topologies with SF switching to the XY routing algorithm and the Mirror Routing algorithm and 2) the evaluation of the WK(4,2) topology with Wormhole (WH) switching, minimum routing and an adaptive routing algorithm. Evaluation of WK topology includes support for burst data transfer. A master sending multiple data to the same destination can be packed into a single packet in WH, hence further reducing latency.

The performance metrics of these design variations include: determining latency for different congestion scenarios, logic area (number of slices) and power (in watts). In addition, the design is evaluated for performance metrics under normal and hotspot traffic conditions. The design is synthesized for Virtex4 using Xilinx ISE 11.1. Table 3 shows comparisons of the area, static power, and total power (static and dynamic) of different NoC architecture components for the Torus and WK topologies.

Mirror Routing Simulation.

To evaluate the mirror routing, we considered the scenario where a faster master PE, PE0, sends information to Slave PEs, PE1 and PE4. The processing elements PE0 and PE4 have faster clocks of 500MHz, whereas PE1 processes data at a much lower frequency, 250MHz. PE0 sends the bulk of data to PE1 and then to PE4; this eventually results in congestion in link R0-R1.

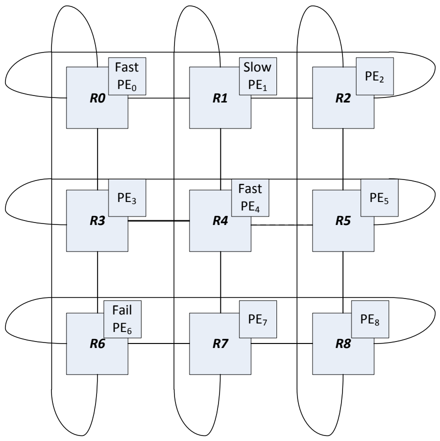


Figure 7. Torus structure with a congested link and failed node.

In normal XY routing, the node R0 will wait until the channel R0-R1 is available; whereas in mirror routing, the NA at R0 chooses an alternative route for the messages through R0-R2, R1-R4. A similar scenario is shown in Figure 7. Latency of packets using SF switching with XY-routing and mirror routing for 3x3 and 4x4 torus topologies are shown in the graph in Figure 8. The latency increases with size of topology, but it also ensures correct data transfer in presence of a single fault. The power and area performance metrics are shown in Table 3.

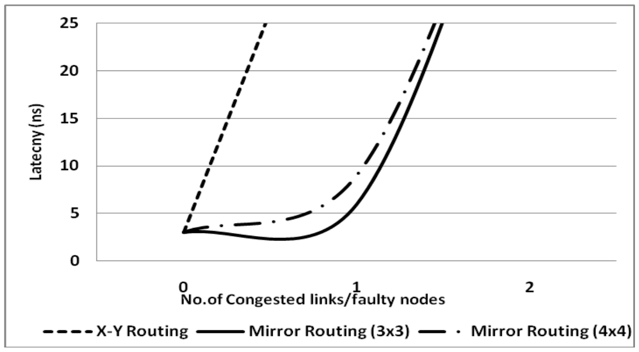


Figure 8: Mirror routing algorithm comparison: Latency

Adaptive Routing (AR) Simulation.

The adaptive routing algorithm for WK(4,2) topology is evaluated for various hops and congestion scenarios. Figure 9 shows the latency performance of the minimal routing, adaptive routing, and an exhaustive routing with respective to the ratio of congested links to the number of hops. The power and area performance metrics are shown in Table 3.

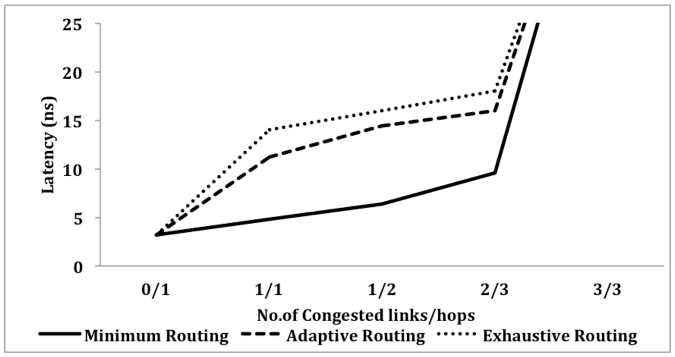


Figure 9: Adaptive routing algorithm comparison: Latency.

6 Conclusion

This work proposes two routing algorithms, one for Torus and one for WK-recursive topology. The routing algorithms are evaluated for congestion scenarios on respective topologies, and their performance for latency is determined. From Figure 8, it is clear that for congestion scenarios, the mirror routing algorithm performance is similar to the classical XY routing algorithm. However, the performance of the mirror algorithm degrades as the network size increases. The difference in area and static power overhead between the mirror and XY routing algorithms is insignificant for small size topologies. From Figure 9, we conclude that the latency of the adaptive routing algorithm is less than that of the exhaustive routing algorithm, where all minimal and alternative routes are stored in the routing table and the first uncongested route is selected for transmitting the packets.

Also, note that if no congestion or fault exists in the links or PEs, then the adaptive routing algorithm performance matches that of the minimal routing algorithm.. Similarly, since the control logic for physical and virtual channel selections are implemented using multiplexers, the area and static power overhead of the adaptive routing algorithm is negligible compared to the minimal routing algorithm.

Table 3. Synthesis Results

	Area (Slices)		**Static Power (W)**		**Total (S+D) Power (W)**	
Routing	AR	MR	AR	MR	AR	MR
PE-OR1K	2605	2605	0.382	0.382	-	-
Router	3330	933	0.202	0.190	1.472	1.023
Core Interface	32	40	0.197	0.205	0.729	0.752
Network Interface	34	34	0.167	0.167	0.767	0.762
Switching	WH	SF	WH	SF	WH	SF

In general, the area overhead of the WH switching (with 4 VCs for WK topology) is 72% more, compared to the SF switching. The static power overhead of WH switching is 31% more and latency is 18% less, compared to SF switching. The total power (static + dynamic power) of implementing the mirror routing on torus topology and the adaptive routing on WK topology are dependent on the number of additional hops encountered during congestion or fault. The total power overhead for each additional hop can be calculated by considering the total power, as shown in Table 3.

7 References

[1] Mostafa Rezazad, M. Hoseiny Farahabady, and Hamid Sarbazi-Azad "A deadlock free shortest path routing algorithm for WK-recursive meshes ICDCN'08 Proceedings of 9th international conference on Distributed computing and networking,2008

[2] J. Duato, et. al. "Interconnection Networks: An Engineering Approach". Morgan Kaufmann, 2002 ..

[3] H. T. Salminen E., Kulmala A., "Survey on Network on chip Proposals" White Paper OCP-IP, March 2008.

[4] C. Seitz, "Let's Route Packets Instead of Wires", Advanced Research in VLSI: Proceedings of the Sixth MIT Conference, pp. 133-138, 1990.

[5] Ni, L. M.; McKinley, P. K. A Survey of Wormhole Routing Techniques in Direct Networks. IEEE Computer Magazine, v.26(2), February, 1993, pp. 62-76.

[6] Della Vecchia, G., Sanges, C. "A Recursively Scalable Network VLSI Implementation.", Future Generation Computer Systems 4(3), 235–243 (1988)

[7] C. J. Glass, L. M. Ni. "The turn model for adaptive routing". In Proc. ISCA, 1992.

[8] S. Toueg and K. Steiglitz, "Some Complexity Results in the Design of Deadlock-Free Packet Switching Networks," SIAM Journal on Computing, Vol. 10, No. 4, pp. 702-712, November 1981.

[9] Liang, J.; Swaminathan, S.; Tessier, R. aSOC: "A Scalable, Single-Chip communications Architecture. In: IEEE International Conference on Parallel Architectures and Compilation Techniques", Oct. 2000, pp. 37-46.

[10] R. Fernandes, D.K. Friesen, and A. Kanevsky. "Efficient Routing and Broadcasting in Recursive" Interconnection Networks. In Proceedings of Int'l Conference on Parallel Processing, 1994

[11] D.Rahmati, A.Kiasari, S.Hessabi, H.Sarbazi-Azad, "A Performance and Power Analysis of WK-Recursive and Mesh Networks for Network-on-Chips", Computer Design, 2006. ICCD 2006. International Conference on Oct. 2007, pp 142 – 147

[12] Mahmoud Moadeli, Ali Shahrabi, Wim Vanderbauwhede, Mohamed Ould-Khaoua "An Analytical Performance Model for the Spidergon NOC", Advanced Information Networking and Applications, 2007. AINA '07. 21st International Conference, May 2007, pp 1014 - 1021

[13] M.H. Ghadiry, M.Nadi, M.T. Manzuri-Shalmani, D.Rahmati, "Performance and Power analysis of Routing Algorithms on NOC" 2008

[14] Mello, A.; Copello Ost, L.; Gehm Moraes, O.; Laert, N.; Calazans, V. "Evaluation of Routing Algorithms on Mesh Based NOCs", Faculty of Informatics, Pontifícia Universidade Católica do Rio Grande do Sul, Brasil, Technical Report Series, No. 040, May 2004.

[15] W.J.Dally "Virutal Channel Flow Control" , IEEE Trans. On Parallel and Distributed Systems, Vol 3, No. 2, March 1992, pp 194.

[16] Assaf Shacham, Keran Bergman, Luca P. Carloni "On the design of a Photonic Network on Chip", Networks-on-Chip, 2007. NOCS 2007. First International Symposium, May 2007, pp 53 - 64

[17] Y. Hu, H.Chen, Y.Zhu, A. A. Chien and C. Cheng, "Physical Synthesis of Energy-Efficient Network-on-Chip Through Topology Exploration and Wire Style Optimizations," Design (ICCD), pp.111-118, 2005.

[18] Verdoscia, L., Vaccaro, R.: "An Adaptive Routing Algorithm for WK-Recursive Topologies", Computing 63(2), 171–184 (1999).

Genetic Algorithm Based Bank Selection for Partitioned Memory Architectures

Liao Ximi[1, 2], He Yanxiang[1, 2], Chen Yong[1, 2], Li Fuyang[1, 2]
[1] School of computer, Wuhan University, Wuhan 430072, China
[2] State Key Laboratory of Software Engineering, Wuhan University, Wuhan 430072, China
Wuhan, China
ximil@whu.edu.cn, yxhe@whu.edu.cn, cyong1000@163.com, dxfly@yahoo.com.cn

Abstract— *For partitioned memory architectures, bank selection is a challenging problem. As we all know, it is a NP-hard problem. To solve this NP-Hard problem, people have studied in this area for many years. But we still have some distance away from the optimal solution. In this paper, I will introduce how to solve the bank selection problem by using Genetic Algorithm [1][2]. Though we still can't give the best solution, the approximate optimal one can be found through this method. And the main attribution of this paper is not only giving an efficient bank selection algorithm, but a new idea of the bank selection for partitioned memory architectures.*

Keywords: partitioned memory architectures, bank selection, Genetic Algorithm.

1 Introduction

For partitioned memory architectures, bank selection is a challenging problem. And it is of vital importance for optimizing compilers. Bank switching is a common technique to increase the size of code and data memory without extending the address buses of CPU. The address space is partitioned into memory banks, and the CPU can only access one bank at a time, which is called the active bank. A bank selection instruction is issued to switch between banks. In this paper, we want to find the right data banks that the variables of a program are placed. We did this because it can affect many aspects, such as runtime, low power, small code size or a combination of these parameters. And we concentrate on the code size. That is to say, we used the genetic algorithm to present an optimization that inserts a minimum number of bank selection instructions in the program to guarantee that banked memory is used completely.

For instance, there is a program which has m variables. They have been represented as P (Vi, value) (0<i<=m). In witch Vi is the identifier of a variable and value represents the size of the variable. And we also have an interference graph G to show the interaction of different variables. We assume there are 2k banks in our architecture. They were signed from 0 to 2k- 1. So our work in this paper is allocating these m variables to the 2k banks and it should satisfy the two rules below:

1) All the variables should be allocated and for every bank, the total size of the variable allocated in this bank can’t be bigger than it is capacity.

2) We should try the best to minimize switching between the banks so that a minimum number of bank selection instructions are used.

The remainder of this paper is organized as follows. In section two, we introduce the conception and principle of genetic algorithms and analyze why we choose this method to solve the bank selection problem. In section three, we introduce the algorithm used in this paper in detail. And then in section four, we will give a small example to show how this algorithms worked. Some related work will be given in section five. We round off the article with a conclusion and list some further work as well in section six.

2 Background

A genetic algorithm is a search heuristic that mimics the process of natural evolution. It is usually used to generate useful solutions to optimization and search problems [3]. Genetic algorithm is a simulation of Darwinian natural selection of genetic selection and biological evolution process model. In a genetic algorithm, it expresses the solution of a problem as "chromosome" and through the change of the "chromosome" from generation to generation which including reproduction, crossover and mutation, eventually converge to the "best adapted individuals".

We can then see that the principle of genetic algorithms is simple:

1) . Encoding of the problem in a binary string.

2) . Random generation of a population. It includes a genetic pool representing a group of possible solutions.

3) . Reckoning of a fitness value for each subject. It will directly depend on the distance to the optimum.

4) . Selection of the subjects that will mate according to

their share in the population global fitness.

5) . Genomes crossover and mutations.

6) . And then start again from point 3.

The steps were shown in Figure 1.

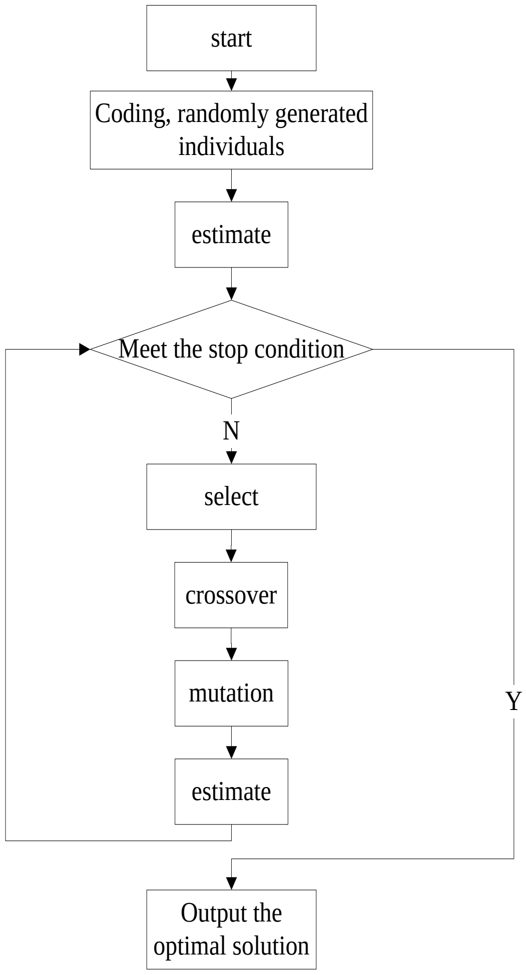


Figure 1: the general steps of genetic algorithm.

1) Initialization: Many individual solutions are randomly generated to form an initial population. The population size depends on the nature of the problem. We encode candidate solutions in binary as strings of 0s and 1s.The different combinations of structural data strings constitute a different individual.

2) Estimate: We use the fitness evaluation function to show the advantages and disadvantages of each of the individual or the solution. The fitness evaluation function is defined in different ways according to different issues.

3) Selection: We have to select good individuals from the current population according to the fitness evaluation to be the "parents". To do this they have the opportunity to breed generations.

4). Crossover： The crossover operation is the most important genetic algorithm operation. Two selected "parent" could generate a new individual by crossover operation. It embodies the idea of information exchange.

5). Mutation： At first, we choose an individual randomly and change a random code of it in a certain probability. The mutation operation provided an opportunity to generate new individual.

6). Stop condition: If the stop condition is met, we choose the individual with the largest fitness in the current population to be the final result and terminate the algorithm.

As we introduced, the genetic algorithm is a good choice to solve optimization problems. The bank selection problem is an optimization problem witch asks for the optimal selection of the bank to make sure that the variables use minimum space of the CPU. Therefore, use the genetic algorithm to solve this problem is a good choice.

3 Genetic algorithms based bank selection

As we introduced in section one, we need to solve the problem of bank selection by genetic algorithm. In the problem, m variables need to be assigned to 2k banks. At the same time, we should try to use the least amount of space and bank selection instructions. Now we will introduce the algorithm in detail.

3.1 The individual coding

Because we have 2k banks, for every variable, there are 2k kinds of placement. We will encode the individual according to the bank of a variable.

For example, if there are 5 variables and 4 banks, we need two bits to represent the selected bank of a variable. That is to say in the total, ten bits will be used to encode and there are 4^5 kinds of encodings. Assuming that a selection as shown in figure 2 below.

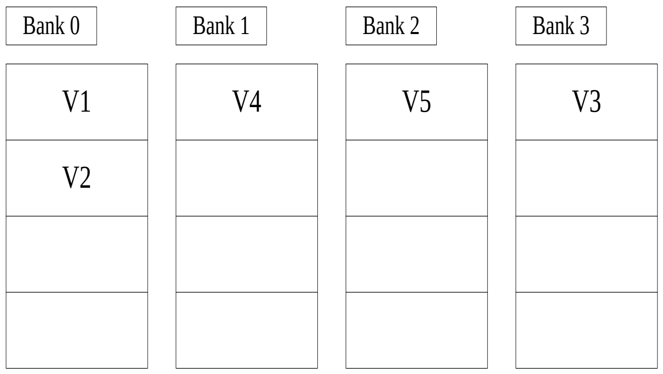


Figure 2:. the selection of an example

The figure shows that V1 and V2 are allocate in bank 1,V4 is in bank 2, V5 is in bank 2 and V3 is in bank 3.So we can encode it as 00 00 11 01 10.

To solve the problem in this paper, we need m*k bits and there will be 2^{mk} kinds of encodings.

3.2 Population initialization

In this paper, we will define the size of the initial population to S = m*2k. We will use an array to store these numbers. Then use the algorithm as shown below randomly generated S individuals of the initial population.

```
1） Initialize array, make array [i] = 0;
2） for ( int i=0; i<=S; i++){
3） randomly generate a m*k bits number s;
4） Make array[i] =s;}
5） exit
```

Figure 3 a simple algorithm for initiation

3.3 Fitness function

The fitness function is used to calculate the fitness of individual. In this paper, the fitness value is the amount of space it will take for a selection. At first, we should calculate the bank selection instructions according to the interference graph G. We use n to records the number of bank selection instructions. At first, we set n=0. And then, we traverse the graph G. if there is an edge from node1 to node2, the two nodes are in deferent banks, n=n+1.Then the total size of the selection is calculated using the formula (1) as following:

$$f(x) = \sum_{i=0}^{m} V_i + n \qquad (1)$$

As shown above, the total size is the number of bank select instructions plus the whole size of the variables.

However, due to the special problems involved in this paper, we should do another thing before we calculate the fitness of individual. We need to first determine the reasonableness of such coding. It is very easy, and we just need to calculate the sum size on each bank. If it bigger than the size of the bank size, the selection must be given up.

3.4 Parent selection

In this paper, roulette wheel selection method [4] was used to select the parents. Roulette wheel selection method simulates gambling game roulette. A roulette wheel is divided into N-sectors. Each sector represents a chromosome of the population. The area of each sector is proportional to the fitness value of the chromosome. In order to select the individuals from the population, we assuming there is a pointer to the wheel, turn the wheel, when the wheel stopped, the pointer points to the selected chromosome. Therefore, the greater the fitness value of a chromosome indicates the larger of the chromosome area and the greater likelihood of it to be chosen. The specific process is as follows:

1). Calculate the sum of the fitness value of all the individuals of the population using formula (2) :

$$Total = \sum_{i=0}^{n-1} f(i) \qquad (2)$$

2). Calculate selection probability for each individual using formula (3):

$$nSel\Pr o(k) = \frac{f(k)}{Total}\left(k = 0,\ldots,n-1\right) \qquad (3)$$

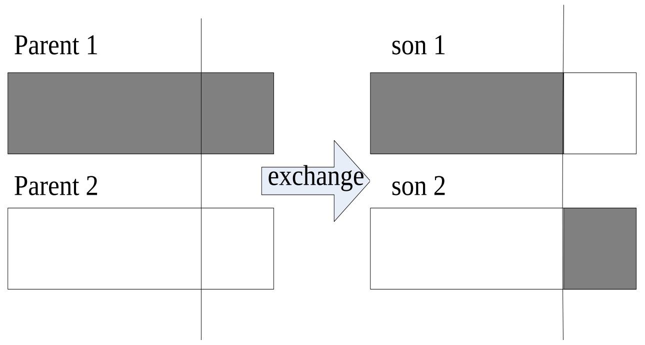
3). Calculate the sum of probability of all the individuals using formula (4):

$$nTo\Pr o(k) = \sum_{i=0}^{k} nSel\Pr o(i)\left(k = 0,\ldots,n-1\right) \qquad (4)$$

4). Turn the wheel N times. We use a random number in [0, 1] to simulate the location pointed when the wheel stopped. If $r \le nTo\Pr o(0)$, it means the pointer pointed the first sector and we choose the first individual. Generally if $nTo\Pr o(k-1) \le r \le nTo\Pr o(k)$, it shows it pointed the k-sector, and then select the k-chromosome.

3.5 Crossover

In this article, we choose Single-point crossover. It is a simple algorithm, but enough for our experiments using. The Single-point crossover algorithm uses two parent bodies to crossover and produce two offspring individuals. Assuming the length of a binary bit string is L. In this paper, it is S. It generates random integer pos to be the crossover point. Then we exchange the substring of two parent bodies which is the right of crossover point and generate two offspring individuals. The schematic diagram of single-point crossover is in figure 4.



F

igure 4 Schematic diagram of single-point crossover

3.6 Mutation

We use a pre-set probability pm to mutate the chromosome genes of population. If a gene was chosen to mutate, we turned 1 to 0 and 0to 1. For every gene of the population, we generate a random number r. If r>pm, then the gene mutation.

3.7 Stop conditional

The user type in the number of iteration Num. we will test whether the number of iteration is bigger than Num. If not, it will go on to do the iteration. After every time of iteration, we should find the best solution of the current population and compare it with the previous one. We will store the better one until we get to Num. Then we quit and return the best solution.

4 Experimental result

To test the effect of the algorithm, we did an experiment on an embedded system. It is HR6P serial micro control unit designed by an integrated circuit company in China, which own 8-bits RISC instruction set. In this paper, we focus on the method of bank selection based on genetic algorithm. The user cases are designed especially for the compiler. We will compare the normal algorithm with the genetic algorithm with the number of bank selection instructions

We test our genetic algorithm on a real system. The experimental environment is shown in Table 1.

TABLE 1 THE EXPERIMENTAL ENVIRONMENT

CPU	Intel Pentium 4 3.06G
operating system	Windows XP sp3
internal memory	DDR2, 1G
develop environment	Visual Studio 2005
Target Chip	RISC
Test case	Cooperation provided
Test board	the actual system

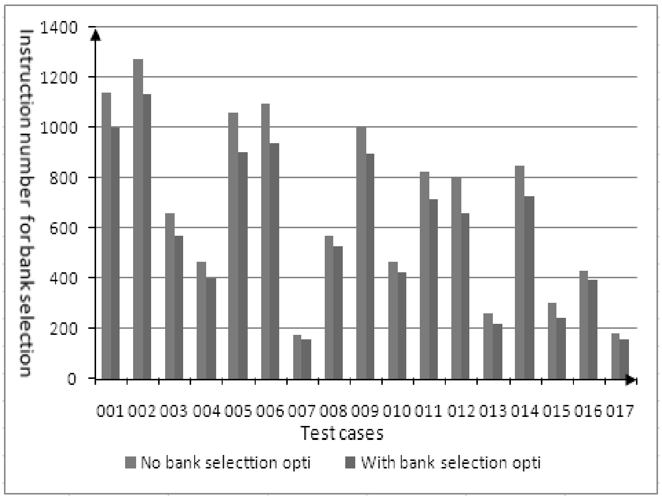


Figure 5. Experimental results

As we shown in Figure 5, we choose 17 cases to evaluate the feasibility and effectiveness of the algorithm. The x-axis present the cases used and the y-axis is the number of instruction used for the bank's selection. From these cases, we can see that it reduces about 12.94% on average and 20.66% for the fifteenth case. So the genetic algorithm is useful to solve the problem of bank selection. After the operation of this algorithm, the number of bank selection instructions significantly reduced.

5 related work

As we all know, many works have been done for optimizing bank selection instructions. Bernhard et al. [5] formulated the problem of optimizing bank selection instructions for multiple goals as a form of Partitioned Boolean Quadratic Programming (PBQP). They assume that the variable had been assigned to specified banks. Liu Tiantian et al.[6] claimed they had integrated variable partitioning into optimizing bank selection instructions. With the analysis of code patterns, they placed the emphasis on the positions for inserting bank selection instructions. Yuan Mengting et al. [7] take the variable partitioning on shared memory. Especially, Li Qingan et al.[8] present an algorithm to reduce the overhead of page switching. To pursue small code size, they place the emphasis on the selection of functions into suitable pages with a heuristic algorithm. Many other works [8][10][11] about variable partitioning focused on DSP processors, where parallelism and energy consumption attracted the main attentions. There is also work to improve the overall throughput for MPSoc architecture by variable partitioning [12].

6 Conclusion and future work

In this paper, we present the genetic algorithm to optimize the bank selection. Our experimental results showed it achieved a great improvement with respect to code size. However, there is still much work to be done to improve this algorithm. In this algorithm, we have not considered a matter of time. And in the future, we will make improvements in this area. Finally, complete content and organizational editing before formatting. Please take note of the following items when proofreading spelling and grammar:

7 References

[1] A.E.Eiben and J.E.Smith. Introduction to Evolutionary Computing. Springer- Verlag Berlin, 2003

[2] K. John, M. Keane, M. Streeter, W. Mydlowec, J. Yu and G. Lanza. Genetic Programming IV: Routine Human-Competitive Machine Intelligence. Kluwer Academic Publishers, 2003

[3] Baragilia R Hidalgo Ji Perego R, A hybrid heuristic for the traveling salesman problem. IEEE Transactions on Evolutionary Computation. 2001(05)

[4] E. E. E. Ail.A proposed genetic algorithm selection method. King Saud University, ccis, 2006.

[5] Bernhard Scholz, Bernd Burgstaller, and Jingling Xue. Minimizing Bank Selection Instructions for Partitioned Memory Architectures. Proceedings of the 2006 International Conference on Compilers, Architecture and Synthesis for Embedded Systems, 2006, pages 201-211.

[6] Liu Tiantian, Minming Li, and Chun Jason Xue. Joint Variable Partitioning and Bank Selection Instruction Optimization on Embedded Systems with Multiple Memory Banks. Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC 2010), 2010.

[7] Yuan Mengting, Wu Guoqing, and Yu Chao. Optimizing Bank Selection Instructions by Using Shared Memory. The 2008 International Conference on Embedded Software Systems (ICESS2008).

[8] Li, Qing'an；He, Yanxiang；Chen, Yong；Wu, Wei；Xu, Wenwen. A Heuristic Algorithm for optimizing Page Selection Instructions. ICSTE 2010 - 2010 2nd International Conference on Software Technology and Engineering, Proceedings,, 2010 v 2, p 143-148.

[9] Zhong Wang, Xiaobo Sharon Hu. Power Aware Variable Partitioning and Instruction Scheduling for Multiple Memory Banks. Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE’04), 2004.

[10] Rainer Leuper, Daniel Kotte. Variable Partitioning For Dual Memory Bank DSPs. ICASSP, 2001.

[11] Meikang Qiu, Lei Zhang, Minyi Guo, Fei Hu, Shaobo Liu and Edwin H. –H. Sha.. Global Variable Partition with Virtually Shared Scratch Pad Memory to Minimize Schedule Length. Proceedings of the 2009 International Conference on Parallel Processing Workshops, 2009.

[12] Qingfeng Zhuge, Edwin Hsing-Mean Sha, Bin Xiao, and Chantana Chantrapornchai. Efficient Variable Partitioning and Scheduling for DSP Processors With Multiple Memory Modules. IEEE Transactions on Signal Processing, Vol. 52, No. 4, 2004.

Dedicated hardware for RC5 cryptography and its implementation

Masaya Yoshikawa, Koichi Sakaue
Department of Information Engineering, Meijo University, Nagoya, Japan

Abstract – *Recently, the number of electronic devices handling confidential information has increased. In these devices, encryption is applied to protect the confidential information. Therefore, technologies to incorporate cryptographic circuits into these cards have become important. This paper proposes a new hardware dedicated to RC5, a typical cryptograph. In the proposed RC5 dedicated hardware, by introducing an architecture suitable for each operation used for the encryption, high-speed processing and area reduction can be realized. Experimental results of which proposed hardware architecture is implemented on FPGA proved the validity of the proposed one.*

Keywords: RC5 cryptography, Dedicated Hardware, FPGA implementation

1 Introduction

Important information such as account numbers is recorded on magnetic tapes on the back of credit and cash cards. Because magnetic tapes can be forged easily, integrated circuit (IC) chips (cryptographic circuits) have been used in recent years. Therefore, technologies to incorporate cryptographic circuits into these cards have become important[1]-[8]. This study proposes hardware dedicated to RC5[9]-[18], a typical cryptograph. RC5 is a common-key block cipher in which the size of the block, the number of rounds, and the size of the key can be changed. RC5 is enciphered by combining three operations of exclusion: exclusive OR, rotation, and residue addition. In the proposed RC5 dedicated hardware, introducing an architecture suitable for each operation used for encryption enables the realization of high-speed processing and area reduction. Moreover, the proposed architecture is incorporated into a field-programmable gate array (FPGA), and we verify its validity.

Mitsuya *et al.*[13] and Nonaka *et al.*[14],[15] conducted typical studies on RC5. Mitsuya *et al.* created an RC5 decryption algorithm. Nonaka *et al.* proposed a known plaintext attack using correlation. These studies were related to software processing. To the author's knowledge, there is no study of RC5-dedicated hardware using the unique architectures investigated in this paper.

In this paper, we describe the outline of the algorithm of RC5 cryptography in the next section. We introduce the proposed architecture in section 3. In section 4, we evaluate the proposed architecture by implementation on FPGA. Finally, section 5 summarizes and concludes this study..

2 RC5 cryptographic algorithm

Please RC5 is a cryptography created by R.L.Rivest[9], which uses the word length w (16/32/64 bits), the number of rounds r (form 0 to 255), and the size of the key b (0-255 bytes) as variables. In addition, one block is composed of two words. Because the variables w = 32, r = 12, and b = 16 are generally used, the present study also uses these values. RC5 uses two secondary keys in each round. Figure 1 shows the method to generate a secondary key.

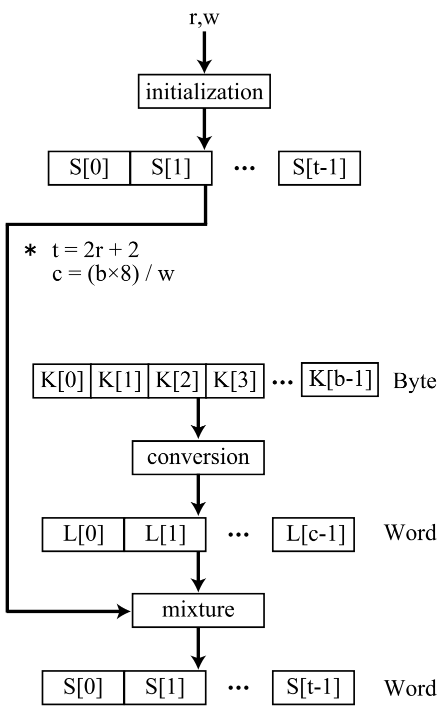

r,w
initialization
S[0] S[1] ··· S[t-1]
∗ t = 2r + 2
c = (b×8) / w
K[0] K[1] K[2] K[3] ··· K[b-1] Byte
conversion
L[0] L[1] ··· L[c-1] Word
mixture
S[0] S[1] ··· S[t-1] Word

Fig.1 The method to generate a secondary key

Moreover, two additional secondary keys are required for the entire encryption. Therefore, when the number of rounds is set at r, 2r + 2 secondary keys are used. As this figure shows, the generation of a secondary key involves three processes: initialization, conversion, and mixture. During the initialization process, a sequence consisting of t words, which is expressed by S[0],..., S[t - 1] and is used for saving a secondary key, is prepared. This sequence is then initialized using random numbers, the input values of which are variables r and w. In the initialization, Pw = Odd[(e - 2)2w] and Qw = Odd[(PHI - 1)2w] are calculated. Here, e represents the base of the natural logarithm and PHI represents the golden ratio (1 + (5)^(1/2))/2. Figure 2 shows the initialization of sequence S using Pw and Sw.

```
S[0] = Pw;
for i = 1 to t-1 do
    S[i] = S[i-1] + Qw;
```

Fig.2 The initialization of sequence S using Pw and Sw

In the conversion process, sequence K[0], K[1],..., K[b - 1] is prepared to develop a key, and the sequence is converted into sequence L[0], L[1],..., L[c - 1] consisting of c words. In addition, c is (b x 8)/w (decimals are raised to the next whole unit). During this processing, sequence L is initialized to 0 and the value of sequence K is directly copied at the memory location of L. Here, if b is not an integral multiple of w, the right end part of L will remain 0.

In the mixture process, the initialized sequence S of the secondary key is mixed with the converted sequence L of the input key to generate sequence S of the final secondary key. Figure 3 illustrates this mixture process.

```
i = j = X = Y = 0
do 3× max(t, c) time :
    S[i] = (S[i] + X + Y) <<< 3;
    X = S[i];
    i = (i + 1) mod (t);
    L[j] = (L[j] + X + Y)<<< (X+Y);
    Y = L[j];
    j = (j + 1) mod (c);
```

Fig.3 Example of mixture process

In this figure, the plus sign ("+") expresses the addition of "mod 2w," x <<< y expresses the leftward shift of word x by y bits, and x >>> y expresses the rightward shift of word x by y bits. The mixture process compares t, which is the size of the secondary key's sequence (S), to c, which is the size of the input key's sequence (L). The larger value is multiplied by three and the calculation is repeated with the obtained number. When the result of the addition exceeds the word length, overflow occurs.

Figure 4 shows encryption of RC5 using secondary keys. In this encryption, plaintexts are stored in 2 w bit registers A and B. When round i is completed, data that were divided into the left and right sides have already been stored in variables LEi and REi, respectively.

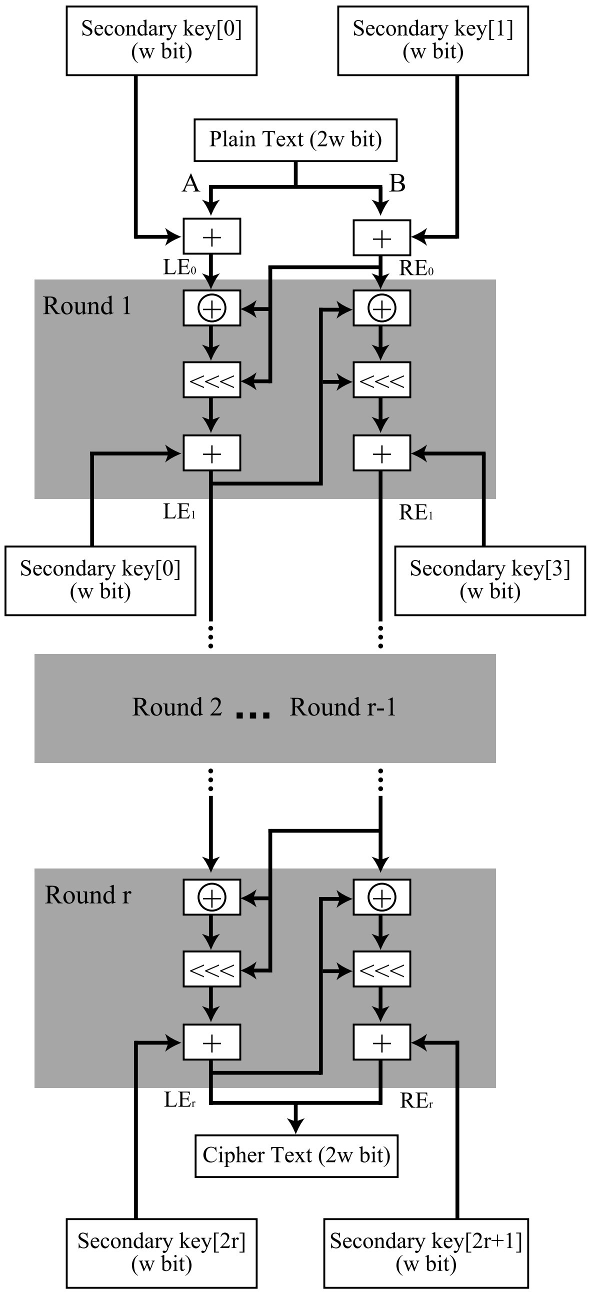


Fig.4 Example of encryption of RC5

The cryptogram that exists after the completion of r rounds is stored in two variables, LEr and REr. Figure 5 shows the processing procedures of LEi and REi. During decryption, 2 w bits of a cryptogram are assigned to LDr and RDr; then the decryption can be performed following the procedure that reverses the encryption. Figure 6 shows the decryption process.

```
LE0 = A +S[0];
RE0 = B + S[1];
for i = 1 to r do
        LEi = ((LEi-1 ? REi-1) <<< REi-1) + S[2 × i];
        REi = ((REi-1 ? LEi) <<< LEi) + S[2 × i+1];
```

Fig.5 Example of the processing procedures of LEi and REi

3 Proposed architecture

In order to realize high-speed processing and area reduction, this study introduces arithmetic processes suitable for hardware during encryption and decryption. First, a shift process used for encryption is replaced by a bit selection process. Because of this substitution, the shift process can be realized in wiring.

The mixture process usually requires t x 3 = 78 calculations. The proposed architecture divides the processing by inserting a register between calculations and introducing a loop process, which reduce the required calculations to 26.

Dividing the processing of each round enables high-speed processing. As Fig. 4 shows, RC5's round processing enciphers a 64-bit plaintext. However, the actual processing is performed every 32 bits. Therefore, the round is divided into the left and right parts to perform left and right processing with two clock signals, which raises the operating frequency and improves latency.

4 Implementation on FPGA

In order to verify the proposed architecture's validity, we incorporated it into an FPGA. The FPGA used in the experiment was XC5VLX30/50 and we used ISE ver.12.3 of Xilinx as the developmental environment. Figure 7 shows the target device for implementation. For comparison, the RC5 algorithm used in Section 2 was described in Verilog HDL. This paper refers to the algorithm used for comparison as the normal method; Table 1 shows that method's circuit scale and Table 2 shows the proposed method's circuit scale.

In each table, "FF" expresses a flip-flop, "LUT" (look-up table) expresses a basic logical element in the FPGA, and "Slice" expresses a block composed of an FF and four LUTs.

As Table 1 shows, because the number of LUTs exceeded the number that the normal method can use, that method cannot be incorporated into an FPGA. As Table 2 shows, although the proposed method had more FFs and Slices than the normal method, the proposed method can be incorporated into an FPGA because it remarkably reduces the number of LUTs. Table 3 shows the operating speed. Because the normal method could not be incorporated into an FPGA, we could not evaluate its operating speed. When we used the proposed method, it operated at 100 MHz.

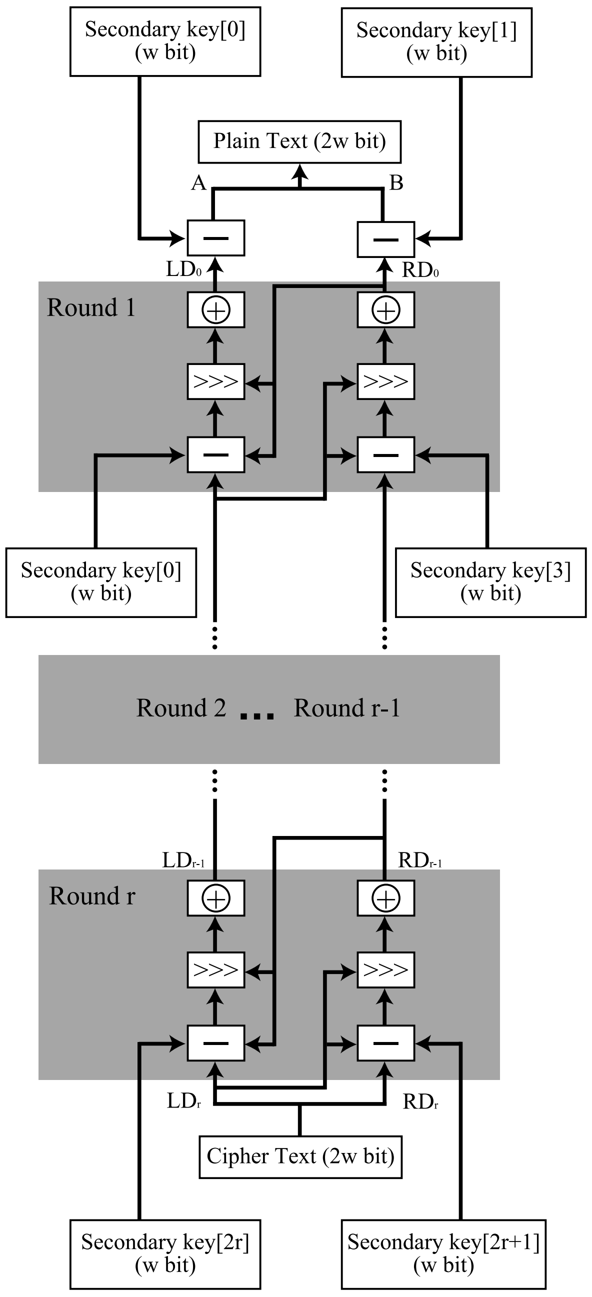


Fig.6 Example of decryption process of RC5

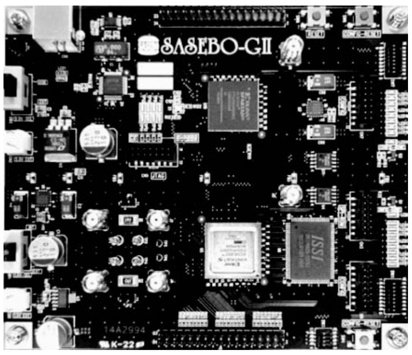


Fig.7 The target device for implementation

Table 1 The normal method's circuit scale

Resources	Used	Available	Utilization
Slices	[N/A]	4,800	[N/A]
LUTs	22,360	19,200	116%
FFs	554	19,200	2%

Table 2 The proposed architecture’s circuit scale

Resources	Used	Available	Utilization
Slices	2,488	4,800	51%
LUTs	8,893	19,200	46%
FFs	2,281	19,200	11%

Table 3 Comparison of the clock frequency

Method	Period	Frequency
Normal	[N/A]	[N/A]
Proposed	9.920 ns	100.806MHz

5 Conclusion

This study proposed hardware dedicated to RC5, which is a common-key block cipher. In the proposed hardware, by introducing a new hardware-oriented algorithm into the shift arithmetic, mixture, and round processes during encryption and decryption, high-speed processing and area reduction were realized. Incorporating it into an FPGA verified the proposed hardware's validity. In the future, we will develop an identification system using the proposed hardware.

6 Acknowledgment

This research was supported by Japan Science and Technology Agency (JST), Core Research for Evolutional Science and Technology (CREST).

7 References

[1] A.Klimm, M.Haas, O.Sander, J.Becker, "A flexible integrated cryptoprocessor for authentication protocols based on hyperelliptic curve cryptography", Proc. of International Symposium on System on Chip (SoC), pp.35-42, 2010.

[2] A.Irwansyah, V.P.Nambiar, M.Khalil-Hani, "An AES Tightly Coupled Hardware Accelerator in an FPGA-based Embedded Processor Core", Proc. of International Conference on Computer Engineering and Technology, Vol.2, pp.521-525, 2009.

[3] B.B.Brumley, K.U.Jarvinen, "Conversion Algorithms and Implementations for Koblitz Curve Cryptography", IEEE Transactions on Computers, Vol.59, No.1, pp.81-92, 2010.

[4] M.Rahman, I.R.Rokon, "Efficient hardware implementation of RSA cryptography", Proc. of International Conference on Anti-counterfeiting, Security, and Identification in Communication, pp.316-319, 2009.

[5] F.A.G.Muzzi, R.B.Chiaramonte, E.D.M.Ordonez, "The Hardware-based PKCS#11 Standard using the RSA Algorithm", IEEE Latin America Transactions, Vol.7, No.2, pp.160-169, 2009.

[6] Zhang Lina, "Research on critical technology of elliptic curve cryptosystem SOC". Proc. of International Conference on Communication Systems, Networks and Applications (ICCSNA), Vol.2, pp.77.80, 2010.

[7] Zhimin Chen, P.Schaumont, "Early feedback on side-channel risks with accelerated toggle-counting", Proc .of IEEE International Workshop on Hardware-Oriented Security and Trust, pp.90-95, 2009.

[8] H.Rahaman, J.Mathew, A.Jabir, D.K.Pradhan, "C-testable S-box implementation for secure advanced

encryption standard", Proc .of IEEE International On-Line Testing Symposium, pp.210-211, 2009.

[9] R.L.Rivest, "The RC5 Encryption Algorithm". Proc. of the Second International Workshop on Fast Software Encryption (FSE), pp.86-96, 1994.

[10] A.Biryukov, "Improved cryptanalysis of RC5", Advances in Cryptology-Proc. of EUROCRYPT'98, Lecture Notes in Computer Science 1403, pp.85-99, 1998.

[11] L.Knudsen, "Improved differential attacks on RC5", Advances in Cryptology-Proc. of CRYPTO'96, Lecture Notes in Computer Science 1109, pp.216-228, 1996.

[12] J.Borst, "Linear Cryptanalysis of RC5 and RC6", Proc. of Fast Software Encryption, Lecture Notes in Computer Science 1636, pp.16-30, 1999.

[13] H.Mitsuya, T.Shimoyama, and S.Tsuji, "Experimental results of cryptanalysis of RC5 (1)", Technical report of IEICE. ISEC 98(426), pp.31-38, 1998.

[14] M.Nonaka, A.Miyaji, Y.Takii, "A Study on Known Plaintext Attack against RC5 by using Correlations", Technical report of IEICE. ISEC 101(311), pp.61-68, 2001.

[15] A.Miyaji, M.Nonaka, Y.Takii, "Known Plaintext Correlation Attack Against RC5", Lecture Notes in Computer Science 2271/2002, pp.115-141, 2002.

[16] A.Ali, L.Aliyar, V.K.Nisha, "RC5 encryption using key derived from fingerprint image", Proc. of IEEE International Conference on Computational Intelligence and Computing Research, pp.1-4, 2010.

[17] K.M.Rudrappa, H.D.Maheshappa, C.Puttamadappa, K.Somashekar, K.S.V.Prasad, "Implementing RC5 protocol for remote control applications", Proc. of International Conference on Control, Automation, Communication and Energy Conservation, pp.1-6, 2009.

[18] Tingyuan Nie, Yansheng Li, Chuanwang Song, "Performance Evaluation for CAST and RC5 Encryption Algorithms", Proc .of International Conference on Computing, Control and Industrial Engineering, Vol.1, pp.106-109, 2010.

Reliability Modeling in Multi-core Embedded Systems: A Fast Approach

E. Cubillos[1], **F. Bolaños**[2], **F. Rivera**[3], **and J. Aedo**[4]
[1,4]Electronics Department, Universidad de Antioquia – ARTICA, Medellin, Colombia
[2]School of Mechatronics, Universidad Nacional de Colombia – ARTICA, Medellin, Colombia
[3]Computer Science Department, Universidad de Antioquia – ARTICA, Medellin, Colombia

Abstract - *Development of embedded systems has been growing significantly in the last years, specially, in systems aimed to multicore platforms. As the complexity of these systems grows, high level reliability models are required in order to guarantee correct operation. Currently, there is a lack of design methodologies that consider reliability as an inherent property of such systems. This research explores a model to evaluate the reliability of a system during the design process. The system is represented by a task graph and redundancy is applied to provide reliability at design time. Several design choices can be explored to allow the designer selects the best according to design requirements. Three case study were used; each case is an embedded system with different complexity. For each one of such systems, assessment for several design choices was obtained using the proposed model, in order to select the most reliable alternative.*

1 Introduction

The technological trend has increased the availability of resources and the complexity of the embedded systems, which has, in turn, boosted its presence on several fields, such as: health, transport, communications, entertainment, and manufacturing process. The embedded systems used in these innovative applications are characterized by their high complexity and the use of architectures with many processing units and an increasing demand on its reliability.

In medical and transport applications, for instance, it is necessary to have a high level of reliability in order to guarantee the correct behavior of the system, even under critical conditions. Because of this, it is necessary to use techniques to increase reliability and to establish models to evaluate it.

Fault tolerance is the most often used technique to achieve reliability. This technique is implemented through the use of redundancy, in which the information flow is replicated to assure a correct result, even if a failure occurs. The redundancy can be applied in two ways: temporal redundancy, in which a specific task is replicated several times in the same resource but in different time interval, and spatial redundancy, in which the task is replicated in other available resources.

This paper presents a model to evaluate the reliability of an embedded system. A given system is represented by a task graph. The proposed model is used to assess several design alternatives, which are generated when different levels of spatial redundancy are implemented for the system's tasks.

This article is organized as follows: Section 2 summarizes some concepts about reliability and related work. The section 3 presents a model to evaluate the reliability. Section 4 presents the experiments to model verification. The conclusions and future work is presented in section 5.

2 Background and Related Work

The dependability can be defined as the capacity of a system to provide a service that can justifiably be trusted on [1, 2]. Such a concept is formed by several attributes: reliability, maintainability, availability, integrity, safety and confidentiality. The reliability can also be defined as the probability of a system to work well in an interval *[to, t]*.

Currently, there are several techniques to increase the reliability in embedded systems as the redundancy, in which the information flow is reproduced to ensure the functioning of system under faults. This technique can be applied of several forms, depending on the number of copies used to replicate a single task (N-module redundancy), Among these approaches, double module redundancy (DMR) [3, 4] and triple module redundancy (TMR) [5, 6] are a common scenario. In TMR, modules are connected to the input of a voting system, aimed to decide which is the correct result for the task. This allows to detect and correct single event upset (SEU) errors, which can be produced by electromagnetic radiation in devices such as microprocessors and memories [7].

On the other hand, it is necessary set models to assess the reliability. These models can be classified in three categories: combinational models, state space models, and simple analytical models.

The fault trees [8, 9, 10, 11], reliability graph [10, 12, 13], and the reliability blocks diagrams [14, 15, 16, 17] belong to combinatorial models. These models focus on a probabilistic evaluation of reliability characteristics, they can isolate system parts that are sensitive to faults and allow individual analysis. However, these models do not either consider the ordered sequence of faults that can arise in the system, nor temporal dependence among the isolated parts, neuther consider they the dependency among components that can be repaired, in which many components share reparation properties. These aspects are very important to total evaluation of reliability system.

The state space models such as the Markov Chains [12, 18, 19, 20, 21, 22] and Petri Nets [23, 24, 25, 26] include the temporal characteristics of system, because they allow modeling deterministic and non-deterministic events similar to real systems, they also can integrate fault rate or system state transitions taking into account the repair state of system components and the temporal dependency of system division. However, these type of models are not easy to be built or studied; they require high level mathematical methods to analyze them. They suffer from a problem of state explosion when they are not properly handled. In complex systems this explosion is not manageable, and the system representation becomes unviable.

Using Petri nets, it is possible to model both fault and reparation states, but its final structure depends on the designer´s experience. There are not formulae to reach a Petri net representation in an univocal way.

Analytical models allow a fast reliability evaluation from the design process, evaluating different design alternatives, in order to select the best according to design requirements. Jhumka *et al*., [27] present a model of this type, described by the equation 1, where C_i is an importance factor of task i, p_i is the fault probability of such task, and f_i is its replication factor. The fault probability is associated with the fault probability of resource where the task is executed. In this model, the design alternatives are represented by several redundancy schemes, i.e. each task in the system may have a different replication factor, which means how many times a given task would be executed simultaneously in different resources.

$$D = \sum_{i=1}^{T} C_i \times (1 - p_i^{f_i}) \tag{1}$$

The Jhumka model has some weaknesses such as:

1. The model calculates the dependability of the system, but only evaluates one of the attributes of dependability: "the reliability". A more complete model should take into account several dependability attributes. The Jhumka´s model is only concerned with reliability

2. The importance factor C_i provides subjectivity to evaluation. This does not represent a good option for the reliability calculation, so it is necessary to define a set of rules for determining the value C_i. Several designers could give completely different values to importance factors on a given design's scenario. Nevertheless, importance factor is a very useful way to give more weight or criticity to a given task in the system, which allows reflecting this importance on the reliability assessment.

3. The expression $(1 – p_i^{fi})$, where f_i is the replication factor and p_i the task fault probability, assumes that the redundancy applied is "temporal redundancy," in which the replicas made are always in the same resource. This type of redundancy increases the reliability of the system, but it is necessary to bear in mind that the fault probability of task is associated to resource in which it is executed if a fault in the task appears, this imply a fault in the resource, and it would not be possible to execute the other replicas.

4. The model ignores the task dependence. That means a drawback, when modeling real designs.

This paper propos a simple analytical model, based on Jhumka's model described in section 2.

3 Proposed Model

3.1 Representation of the System

The system under design can be represented by a set of tasks which are organized by means of a task graph. In such a graph, tasks are represented by nodes and task dependences are represented by means of edges. The figure 1 represents an application with 8 tasks.

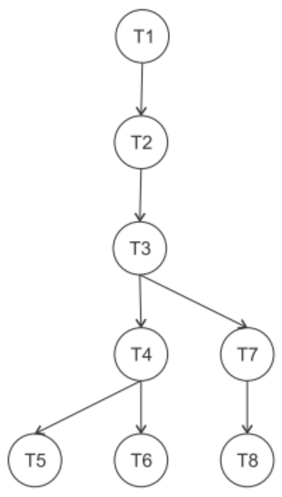


Figure 1. Task graph.

The TGFF tool was used to generate the tasks, in order to test the proposed model [28]. This tool has been widely used in many related scenarios with the digital design [29, 30, 31, 32, 33, 34].

TGFF provides a common base, in order to compare several modeling strategies and provides values for figures of merit such as performance, power, and fault probability.

The design scenario consists on five resources (namely R1 to R5), available for replication. Table 1 provides information about the fault probability of each resource. When a task is about to be replicated, lowest fault probability available resource is chosen in order to optimize the reliability of the system.

Table 1. Fault probability or resource

	Fault Prob.
R_1	0.05
R_2	0.06
R_3	0.14
R_4	0.15
R_5	0.34

There could be up to five resources available for replicating a given task on a given time. The number of actual replicas executed for each task (namely n) will determine the value of reliability of such a task. The system reliability is calculated as the sum of individual tasks reliabilities. Figure 2 shows a given redundancy scheme, for the graph shown on Figure 1. In the figure 2, the task T_1 is not replicated, the task T_2 is replicated three times, the task T_3 is also replicated three times and so on with the rest of the system. As mentioned before, the criterion for assigning resources is the order in which there are organized (lowest to highest fault probability): the task T_1 will be executed in the resource R_1, the task T_2 will be executed in the resources R_1, R_2 and R_3. Besides if a task T_i is replicated 4 times, it will use the resources R_1, R_2, R_3 and R_4, and if it is replicated 5 times, it will use all available resources.

3.2 Model

The proposed model looks for overcoming some drawbacks present in Jhumka's model, given the following considerations:

1. The model is used to calculate the attribute of reliability, which is represented by letter R.

2. The importance factor C_i will be used to give more weight to tasks that have higher index of criticity. In case that the task is not critic, the value will be 1.

3. The expression $(1 - p_i^{fi})$ of Jhumka's model assumes temporal redundancy, since the task is replicated fi times in the same resource but in different time intervals. This means that even if there are idle resources to replicate a given task, the system must execute such replicas using the same resource, degrading the system's performance. This research propose to use the available resources in order to implement redundancy; consequently the expression must be changed for the sake of taking into account several fault probabilities of the resources involved, as shown on equation (2).

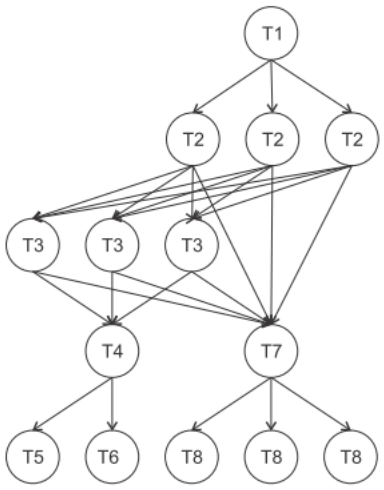


Figure 2. Task´s redundancy.

$$P = 1 - \prod_{k=1}^{R} p_k \qquad (2)$$

4. It is necessary to consider the task dependences when calculating a given task reliability. This is known as conditional reliability. An adjacency matrix to model the tasks dependences is proposed. Table 2 shows such adjacency matrix, for the graph shown on Figure 2. In such a table, 1 placed on row *i* and column *j* means that task *j* has a direct dependence with respect to the task *i*. By definition, every task has dependence with itself, so all entries on diagonal of the matrix are equal to one.

Table 1. Adjacency Matrix

T	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*
1	1	1	0	0	0	0	0	0
2	0	1	1	0	0	0	1	0
3	0	0	1	1	0	0	1	0
4	0	0	0	1	1	1	0	0
5	0	0	0	0	1	0	0	0
6	0	0	0	0	0	1	0	0
7	0	0	0	0	0	0	1	1
8	0	0	0	0	0	0	0	1

Starting from the adjacency matrix, it is necessary to derive an expression in order to assess the conditional reliability for each task. If tasks are numbered properly, T_1, T_2, T_3 etc., fault probability of a single task *i* can be calculated according with equation (3). In such equation,

A_{ji} corresponds to the *(i, j)* entry of the adjacency matrix, and f_j corresponds to the failure probability of resource *j*.

$$P = 1 - \prod_{k=1}^{R} p_k \qquad (3)$$

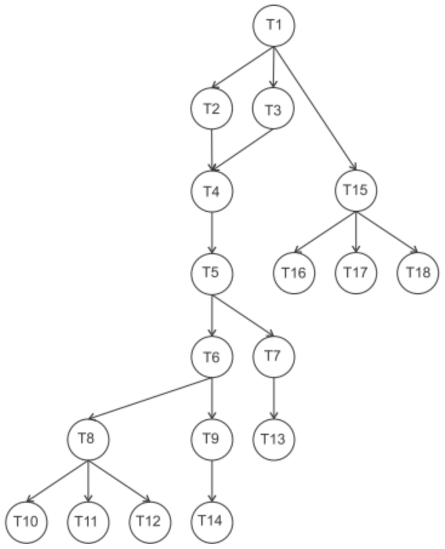


Figure 3. Graph Task N° 2.

The proposed model is shown in equation (4). As can be seen, each term of the sum has two multiplicands: The first is related with the dependence of the task with previous tasks on the graph. The second one is related with the reliability of the individual task given the replication factor assigned to it. The sum of all individual reliabilities is equal to the system reliability.

$$F = \sum_{i=1}^{T} C_i \left\{ \left[\prod_{j=1}^{i-1} f_j^{A_{ji}} \right] \times \left[1 - \prod_{k=1}^{R} p_k \right] \right\} \qquad (4)$$

4 Obtained results

In order to test the proposed model, three experiments were used. As said before, each experiment is represented by a task graph, as shown on figures 1, 3, and 4. Several considerations were taken into account in order to test the model:

- TGFF was used to generate the experimental data.
- All tasks have equal criticity index; consequently the importance factor is equal a 1.
- In the first experiment, 4 design alternatives were used.

1. Without applying redundancy.

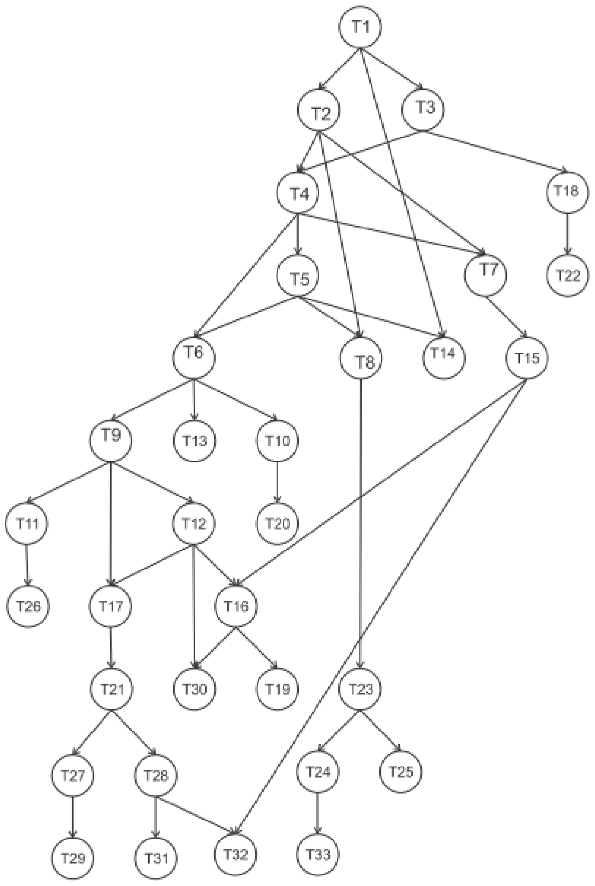


Figure 4. Graph Task N° 3.

2. The task T_2, T_3 and T_8 were replicated three times.
3. The task T_1, T_2, T_4 and T_8 were replicated three times and the task T_7 was replicated four times.
4. All tasks were replicated five times.

- The second and third experiment, the reliability was evaluated for two options: without applying redundancy (minimum redundancy) and applying five replicas (maximum redundancy), to analyze the reliability variation.
- The Jhumka's model was used as reference in three experiments, even if the redundancy scheme used in both approaches was different.

Figures 5, 6 and 7 show the results of models comparison. As can be observed, the biggest difference between the two models occurs when no redundancy is used. This is due to the fact that the proposed model works taking into consideration among tasks, consequently the term related with such dependences, becomes more important in the calculations if there are not replications at all.

Figures 5, 6, and 7 also show that both models provide similar results, if the replication factor for each task grows to its maximum values. This is due to the asymptotic nature of the reliability models, which they themselves saturate to a maximum value.

Execution time for Jhumka's model was about 0.037 ms, while for the proposed model execution time was 4 ms. The difference reflects the additional complexity of the proposed model, but also shows that such model is suitable for design space exploration, since this time does not represent high delays on reliability calculations.

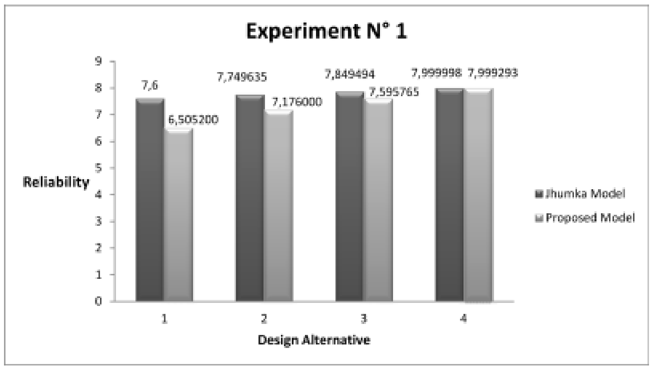


Figure 5. Results experiment N° 1.

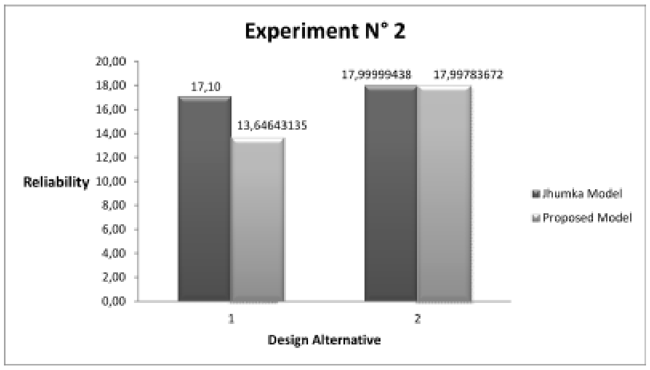


Figure 6. Results experiment N° 2.

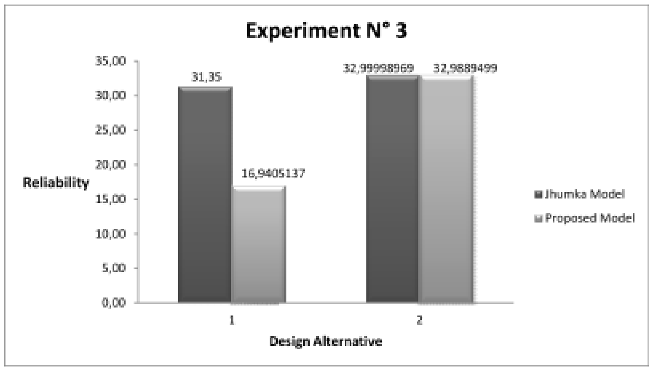


Figure 7. Results experiment N° 3.

To verify the model, an algorithm was implemented in Matlab R2009a, in Windows XP SP3 in a laptop Compaq Presario C708LA with o Intel Dual Core T2310 - 1.47 GHz processor, RAM de 2 GB and HD120 GB.

5 Conclusions and future work

A new model for reliability assessment in design space exploration has been presented, with promising results, when compared with similar analytical models, reported on literature.

The proposed model works in design scenarios in which there are several resources for implementing the system's tasks (multicore). A task graph and the fault probabilities of each available resource are the inputs of the model, in order to assess the reliability of such system.

To effectively apply the model from design process, it is necessary the creation of benchmark or a real database, in which different tasks with their corresponding in a precise resource can be executed.

The main difference between the proposed model and the reference model is that the later uses temporal redundancy, in order to increase system reliability. The proposed model deals with spatial redundancy. As a future work two developments are proposed: 1. Model which takes into account both kinds of redundancy. 2. Model which takes into account other dependability attribute, in order to complete the model proposed here.

6 Acknowledgments

Authors thank to ARTICA, to COLCIENCIAS, to Communications Ministry of Colombia, to Universidad Nacional de Colombia and to Universidad de Antioquia, for their support in this research.

7 References

[1] A. Avizienis, J. Laprie, and B. Randell, “Fundamental concepts of dependability”, *Third Information Survivability Workshop ISW-2000*, 2000.
[2] A. Avizienis, J. Laprie, B. Randell, and C. Landwehr, “Basic concepts and taxonomy of dependable and secure computing”, *IEEE Transactions on Dependable and Secure Computing*, vol. 1, pp. 11-33, 2004.
[3] J.E. Kim, J. Lin, L. McMurchie, C. Sechen, "Mitigation of Single-and Multiple-Cycle-Duration SETs using Double-Mode-Redundancy (DMR) in Time," *2005 IEEE Aerospace Conference*, pp.1-10, 5-12 March 2005..
[4] A. Golander, S. Weiss, and R. Ronen, "Synchronizing Redundant Cores in a Dynamic DMR Multicore Architecture," II: Express Briefs, *IEEE Transactions on*

Circuits and Systems, vol.56, no.6, pp.474-478, June 2009.

[5] O. Ruano, J.A. Maestro, P. Reviriego, "A Methodology for Automatic Insertion of Selective TMR in Digital Circuits Affected by SEUs," *IEEE Transactions on Nuclear Science, vol.56, no.4*, pp.2091-2102, Aug. 2009.

[6] Y. H. Kang, T. Kwon and J. Draper, "Fault-Tolerant Flow Control in On-chip Networks", *2010 Fourth ACM/IEEE International Symposium on Networks-on-Chip (NOCS)*, pp.79-86, 3-6 May 2010.

[7] T. Sato, and T. Funaki, "Dependability, power, and performance trade-off on a multicore processor", *EDA Technofair Design Automation Conference Asia and South Pacific*, pp. 714-719, 2008.

[8] H. Zhu, S. Zhou, J. B. Dugan and K. J. Sullivan, "A benchmark for quantitative fault tree reliability analysis," *Reliability and Maintainability Symposium, 2001*, pp.86-93, 2001.

[9] S.V. Amari and J. B. Akers, "Reliability analysis of large fault trees using the Vesely failure rate," *Reliability and Maintainability, 2004 Annual Symposium - RAMS*, pp. 391- 396, 26-29 January 2004.

[10] S. Distefano and A. Puliafito,"Dependability Modeling and Analysis in Dynamic Systems," *IEEE International Parallel and Distributed Processing Symposium, 2007. IPDPS 2007*. pp. 1-8, 26-30 March 2007.

[11] R. Beresh, J. Ciufo and G. Anders, "Basic Fault Tree Analysis for use in protection reliability," *Power Systems Conference: Advanced Metering, Protection, Control, Communication, and Distributed Resources, 2007*. PSC 2007, pp.1-7, 13-16 March 2007.

[12] K. Trivedi, D. Kim, A. Roy and D. Medhi, "Dependability and Security Models", *7th International Workshop on the Design of Reliable Communication Networks*, 2009.

[13] A. Kusiak and A. Zakarian, "Reliability evaluation of process models," *IEEE Transactions on Components, Packaging, and Manufacturing Technology, Part A, vol.19, no.2*, pp.268-275, Jun 1996.

[14] A. Brall, W. Hagen and H. Tran, "Reliability Block Diagram Modeling - Comparisons of Three Software Packages," *Reliability and Maintainability Symposium, 2007. RAMS '07*, pp.119-124, 22-25 January 2007.

[15] S. Distefano, "Dependability of complex, large, dynamic systems," *8th International Conference on Reliability, Maintainability and Safety, 2009. ICRMS 2009*, pp.27-31, 20-24 July 2009.

[16] O. Boncalo, M. Vladutiu and A. Amaricai, "Accuracy analysis of the parallel composition for the block diagram based reliability assessment of quantum circuits," *2010 International Joint Conference on Computational Cybernetics and Technical Informatics (ICCC-CONTI)*, pp.355-359, 27-29 May 2010.

[17] A. Elamy and B. Far, "Condition-based reliability modeling for systems with partial and standby redundancy," *23rd Canadian Conference on Electrical and Computer Engineering (CCECE)*, pp.1-5, 2-5 May 2010.

[18] M.G. McQuinn, P. Kemper, and W. H. Sanders, "Dependability Analysis with Markov Chains: How Symmetries Improve Symbolic Computations," *Fourth International Conference on the Quantitative Evaluation of Systems, 2007. QEST 2007*, pp.151-160, 17-19 September 2007.

[19] G. Xiao and Z. Li, "Estimation of Dependability Measures and Parameter Sensitivities of a Consecutive- k -out-of- n: F Repairable System With (k-1) -Step Markov Dependence by Simulation," *IEEE Transactions on Reliability, vol.57, no.1*, pp.71-83, March 2008.

[20] H. Boudali, et al., "Architectural dependability evaluation with Arcade," *IEEE International Conference on Dependable Systems and Networks with FTCS and DCC, 2008. DSN 2008*, pp.512-521, 24-27 June 2008.

[21] S. M. Iyer, M. K. Nakayama and A. V. Gerbessiotis, "A Markovian Dependability Model with Cascading Failures," *IEEE Transactions on Computers, vol.58, no.9*, pp.1238-1249, September 2009.

[22] E. Bode, et al., "Compositional Dependability Evaluation for STATEMATE," *IEEE Transactions on Software Engineering, vol.35, no.2*, pp.274-292, March-April 2009.

[23] S. Bernardi, and S. Donatelli, "Building Petri net scenarios for dependable automation systems," *10th International Workshop on Petri Nets and Performance Models*, pp. 72- 81, 2-5 September 2003.

[24] S. Bernardi, and S. Donatelli, "Stochastic Petri nets and inheritance for dependability modelling," *10th IEEE Pacific Rim International Symposium on Dependable Computing*, pp. 363- 372, 3-5 March 2004.

[25] S. Liu, Z. Xu, G. Chen and X. Hu; "Dependability Research of Standby System Based on Stochastic Petri Net," *International Conference on Networks Security, Wireless Communications and Trusted Computing, 2009. NSWCTC '09*, pp.179-183, 25-26 April 2009.

[26] H. Xiaojing, L. Shixi and L. Ma, "Research on dependability of virtual computing system based on Stochastic Petri nets," *International Conference on Computer Application and System Modeling (ICCASM), 2010*, 22-24 October 2010.

[27] A. Jhumka, S. Klaus and A. Huss, "A Dependability-Driven System Level Design Approach for Embedded Systems," *Design, Automation, and Test in Europe, Vol 1*, pp: 372 - 377, 2005.

[28] K. Vallerio, Task graphs for free (TGFF v3.0), http://ziyang.ece.northwestern.edu/tgff/, 2003.

[29] Y. Yu and V.K. Prassana, "Energy-Balanced Task Allocation for Collaborative Processing in Wireless Sensor Networks" *Proceedings of the 2003 ACM SIGPLAN conference on Language, compiler, and tool for embedded systems*, 2003

[30] J. Hu and R. Marculescu, "Energy and Performance aware mapping for regular NoC Architectures", *IEEE*

Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2005.

[31] C. L. Cathey, J. D. Bakos and D. A. Buell, “A Reconfigurable Distributed Computing Fabric Expliting Multilevel Paralelism”, *14th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM’06)*, 2006.

[32] T. Xie, X. Qin and M. Nijim, “Solving Energy-Latency Dilemma: Task Allocation for Parallel Applications. Heterogeneous Embedded Systems” *Proc. 35th Int’l Conf. Parallel Processing*, 2006.

[33] L. Fan, B. Li, Z. Zhuang and Z. Fu, “An Approach for Dynamic Hardware/Software Partitioning Based on DPBIL”, *Third International Conference on Natural Computation (ICNC 2007)*, 2007.

[34] Y. Zhang, et al., “A hardware/software partitioning algorithm based on artificial immune principles”, *Applied Soft Computing - ACM*, 2008.

Developing a Smart Home System

Peter Killeen[1], John Monkus[1], Biz Klessig[1], D Hearn[2], Jingxian Wu[1], and Scott C. Smith (contact author)[1]
Department of Electrical Engineering, University of Arkansas, Fayetteville, AR 72701, USA[1]
Department of Computer Science, Mississippi Valley State University, Itta Bena, MS 38941, USA[2]
pkilleen@uark.edu, jmonkus@uark.edu, eklessig@uark.edu, dantehearn@gmail.com, wuj@uark.edu, and smithsco@uark.edu

***Abstract*—This paper describes the design of a smart home system. This design is able to talk wirelessly with radios attached to each end node and the base station. The end nodes collect data and send it back to the base station where an 8051 microcontroller and an Arduino board interpret the information. The intent is to make a person can feel safer in their home, and provide them with a way to monitor their home from a computer or smart phone while away.**

Keywords- microcontroller; Arduino; wireless sensor network

I. Introduction

This project is about developing a smart home system. A smart home is a system that is set up in someone's house that alerts the home owner if something is not right at their house or can allow a person to change something, such as temperature, in their house while they are away. Its design is for ease of access in one's home. Within this particular design there are multiple radios that talk wirelessly back and forth to each other with help from an 8051 microcontroller [1] and an Arduino board [2], which is then interfaced with a website and text messaging system. All this together helps create a safe environment for the home owner that can allow them to call the police before anything extremely bad happens.

This project was undertaken during a 10-week Research Experiences for Undergraduates (REU) program to gain more knowledge on embedded systems, get work experience as an undergrad, and have something fun and intellectually challenging to do during the summer.

Section II provides an overview of the project, and Section III presents the results, conclusions, and future work.

II. Overview

The smart home monitoring system is composed of several different sensors and nodes. These sensors and nodes, which will be in different destinations of the home, are in essence the subsystems of the smart home. These nodes are designed to communicate between two 8051 microcontroller boards through xBee radios and an Arduino board to perform pre-determined programmed tasks. The nodes are composed of an RFID tag reader, infrared sensors, xBee radios, Arduino, cellular module, web interface, temperature sensor, smoke/ CO detector, buzzer, voice recorder, security camera, and a stepper motor. Using these nodes the home owner will be able to keep track of activity in the house.

The basic look of the house will have an RFID reader at the door that the owner can use to turn on and off the security sensor without typing in a key code. The RFID will also let the base station know that the owner is in the house. This will cause the base station to alter its usual functions between homeowner present and away.

The end nodes will also monitor temperature, humidity, smoke, and CO for comfort and safety. This information will allow the homeowner to monitor his/her house. The information will be updated periodically on a webpage and if certain conditions are met, text messages will be sent to alert the homeowner of possible alarming conditions and the message recorder will sound with a message saying that motion was detected. To further assist the homeowner a camera will have some range of motion that will allow them to further inspect the conditions of the monitored environment.

There will be several infrared sensors throughout the house. The infrared sensors inside the house will be set to detect movement inside the house while the owner is away. The owner reserves the right to activate the security system while he/she is still in the house. If unwanted motion is detected the alarm (a buzzer) will sound and flashing lights will be set off. The buzzer which is connected to the Arduino will sound when the end nodes' safety and environment with infrared sensors on them detect motion. By setting up enough nodes around the house the owner will know of an intruder before the intruder can realize the owner is in the house. This will allow the owner to get to a safer room in the house and alert the police, if necessary. On the other hand, if the homeowner is away anytime motion is detected a text message will inform them of the occurrence.

All of these nodes communicate wirelessly with the base station. This is done using a series of xBee radios. Every node has one radio associated with it. The end node radios transmit data to the coordinator in packages, and the coordinator sends the packages out through UART to a microcontroller. A diagram of a network is shown in Fig. 1.

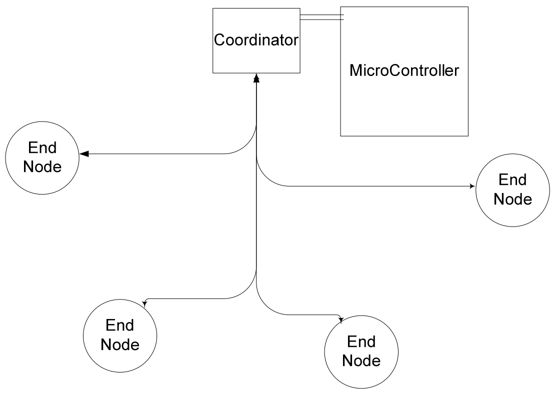


Figure 1. Wireless Network

This is an approach for creating a smart and safe house for an owner to enjoy without the need to run wires throughout the entire house.

A. Base Station

The base station has two main parts, the information retrieval and decoding, and the networking and remote user interface part. The information retrieval and decoding was processed on a DS89C450 8051 microcontroller. The networking and user interface was accomplished on an Arduino main board with an Arduino Ethernet shield and a cellular shield. The base station also has a LCD screen, an xBee radio, a message recorder, and a buzzer for an alarm. The diagram of the base station is shown in Fig. 2.

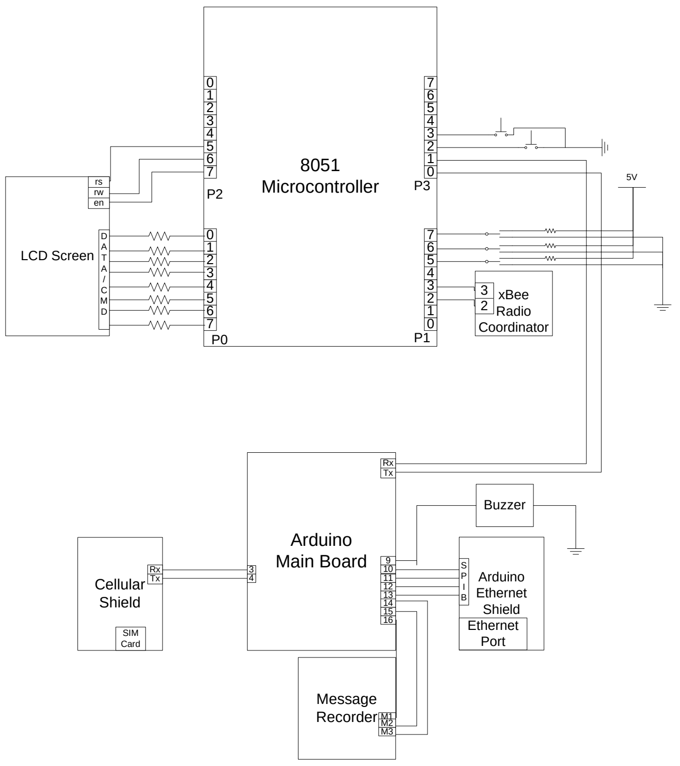


Figure 2. Base Station Block Diagram

1) Information Retrival and Decoding

This subsystem is run by the 8051 microcontroller and is in charge of retrieving RF data from the end nodes and decoding the RF packages as well as allowing some user interface through an LCD screen. The microcontroller receives the information from the wireless network coordinator radio through a UART and processes the information. This information is then decoded by the 8051 and certain functions are preformed depending on the data. The 8051 actions are dependent on several settings that are set either by the user at the base station, with RFID tags at the entrance node, or online. These settings include changing the time at the base station, setting a time dependant alarm, and resetting and clearing the microcontroller's variables.

The microcontroller works almost entirely on interrupts. This allows the base station to receive transmissions at anytime while still operating the LCD screen properly. Using the serial interrupts allows the receiving and sending of data at random when needed, while using the external interrupts for setting and changing features of the LCD screen. The last interrupt is a timer interrupt used to create a clock and also used as a counter for several operations.

The base station receives environment data, motion data, and RFID data. Depending on what information is received, different tasks are performed and different alarms are triggered. It also alternates every five seconds between displaying RFID, and temperature and humidity data, on the LCD screen.

When motion sensor data is received the microcontroller performs two separate tasks. It decodes all the motion sensor data and saves it to memory for further use. This prototype has four motion sensors throughout the nodes. The base station holds one bit of information for each one but it packages it all together in one byte. This allows for easy transmission of the data to the networking section, while also allowing for easy individual changes in memory of each motion sensor. The other function the microcontroller performs is turning on/off lights at the remote camera node. These lights are turned off when the all clear is received from the networking section.

Another simple type of digital data that is received from the end nodes is information on the smoke and CO detector incorporated on the safety node. This information sets off the same alarms in the environment, but the networking section alerts the user in different manners.

When environment data is received the microcontroller performs two operations. First it saves the data to its own memory, and then it passes the data through another UART to the networking and remote user interface section. The environment data consists of humidity and temperature levels in the monitored environment. The microcontroller displays the data on the LCD screen as mentioned before.

When RFID data is received the microcontroller saves it to memory. The RFID tags are used for turning on and off the alarm, working as key passes or fobs for the user. Therefore, when an RFID tag is read the memory changes the state of that card from away to present or vice versa. This allows for the alarm system to be on when no one is home, and turned off when people are present. Using the push buttons and switches on the base station, the user can clear all the RFID tags effectively turning on the alarm. The RFID information is also transmitted to the networking and interface section for the purposes of the web page.

Another way to set the alarm is by using the clock and time dependent alarm. The clock is created using a timer on the microcontroller. Once this was created it allowed for extra time dependant features to be added. One such feature is a time dependant alarm. Using the switches and pushbuttons the correct time can be set and then an alarm on time and alarm off time can be set. The interface of the switches can be observed in Table I. The switches are read by the microcontroller whenever one of the pushbuttons is pressed.

Table I: LCD Interface

Switches			**Push Buttons**	
1	**2**	**3**	**One**	**Two**
1	0	0	Increase Hour	Increase Minutes
0	1	0	Increase Hour of Alarm time Off	Increase Minutes of Alarm time Off
0	1	1	Increase Hour of Alarm time On	Increase Minutes of Alarm time On
1	1	1	no operation	system reset/start up
0	0	0	RFID tags cleared	no operation

The pushbuttons are connected to the external interrupts of the microcontroller which cause the microcontroller to run a certain section of code asynchronously. Once the microcontroller is running one of these code sections, it checks the switch inputs and decides what operation to perform. All these actions are visible on the LCD screen as well.

2) Networking and Remote User Interface

This subsystem is controlled by the Arduino microcontroller which was chosen to be used in conjunction with the 8051 microcontroller for its ability through peripherals to easily host a web server and provide cellular service. Using the Ethernet shield, which contains the Wiznet W5100 Ethernet chip, the Arduino can perform TCP/IP processes. It can easily host a web server with the use of the webduino library, written by Ben Combee, and can generate the GET and POST parameters for a web page. Cellular service is provided by the SM5100B cellular module and SIM card on the cellular shield. The Arduino can communicate with the cellular module by sending AT commands across software serial ports to send and receive text messages and make phone calls. In addition to the Ethernet and cellular shield, the Arduino has a buzzer to signal either intrusion detected or if someone has approached the entrance node and a message recorder to play recorded voice messages for alerting authorities or fire fighters.

The web server generates two web pages: one that asks for a combinational pass code and another that generates the main page with the processed data and user controls if given the correct pass code. The main web page displays the humidity and temperature sensor data, the RFID tags that are present, the security camera along with buttons to pan left and right, and conditional statements with buttons that allow for responses when motion, smoke/CO, or someone at the entrance, is detected.

The cellular module uses simple AT commands transmitted serially from the Arduino to the cellular module. Text message warnings can be sent/received and emergency calls can be made. When an ePIR sensor is tripped the Arduino sends some strings of serial data to the cellular module which uses a SIM card to connect to a cellular network and send a “motion detected” text message. If the home owner is monitoring the web page and a motion sensor goes off then they can decide by pressing a button whether or not to call the police for the detected motion. This call is also made by sending AT commands via the soft serial ports. The cellular module is used to receive text messages as responses to alarms and makes meaningful phone calls with audio recordings via the message recording chip. Also, if smoke/CO are detected the cellular module calls the fire fighters.

B. Android Web Application

The android web application was constructed to allow ease of access for the home owner. With this application the home owner will be able to access the internet from anywhere in the world by use of a cellular phone. The application itself is a web browser that, when opened, will automatically allow the home owner to gain access to their home’s private information. The first page that appears is a log in page for the home owner. This assures that the person accessing the data is an authorized party and not someone trying to gain access to someone’s private information. The application itself is a general view of the web page. It is a general view because the home owner will not be able to view the security camera through the application, at least not at first. If the home owner wants to view the security camera they will have to click the Go button that is at the top of the application. The Go button will cause the application to forward the embedded URL to the phone’s default web browser. To aid in the security, the home owner will have to reenter their security code to view the web page. After they have gained access to the web page they will have to use their log in access code to view the camera. If they do not enter their code they will not be able to view the security camera.

C. End Nodes

The system has four end nodes and a base station. The end nodes have various sensors on them to monitor the environment. This system has an environment, safety, camera, and an entrance end node. Each end node has an xBee Radio that does the necessary transmissions of data to and from the base station. The settings for the xBee Radio are different for each end node; however, as a similarity, each radio has an LED connected to it to indicate its status.

1) Environment Node

The environment end node consists of temperature and humidity sensors to monitor the environment. This end node also has an ePIR motion sensor on it. The purpose of the ePIR is to cover more area of the room/house.

The radio for this end node has several settings different from the rest. First of all, the radio has to read analog inputs from the temperature and humidity sensors, and also monitor the motion sensor. The radio's settings were for reading two analog inputs, and sending the information collected once every minute. It also has digital change detection enabled for the motion sensor. Therefore, this end node sends two types of data packages to the base station. It sends the analog information periodically and the digital information when there is movement detected. The analog to digital conversion takes place in the radios' own microprocessor. However, the analog inputs on the radio have a range from 0 to 1.2V, while our sensors outputted voltages in the range of 0 to 5V. A simple voltage division was used to convert the 0-5V sensor outputs to 0-1.2V. For the motion sensor an inverter was used, therefore logic high is motion, and logic low is the no detection state.

The radio was programmed to go into cyclic sleep and wake up periodically. If something requires action it then proceeds to do it. This behavior can be observed on the associated LED, and all the rest of the radios have this capability as well. This LED can show if it is awake, asleep, or looking for a network to join. The diagram for this node is shown in Fig. 3.

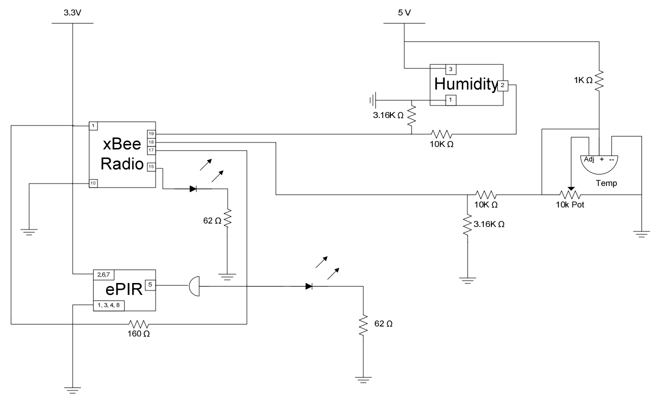


Figure 3. Environmental Node

2) Safety Node

The safety end node consists of a smoke/CO detector and two ePIR motion sensors. The purpose of this node is to keep the home owner safe from fire related occurrences. The smoke/CO detector is a commercial one with an alarm pin. The detector itself comes with an alarm to alert people in the house, but the alarm pin gives the opportunity of connecting it to a bigger system. The ePIR are introduced in this module for the same reason as the last, just to cover more area.

The end node will alert the base station when there is movement or smoke/CO via the radio. This information will then be processed by the base station. The diagram for the node is shown in Fig. 4.

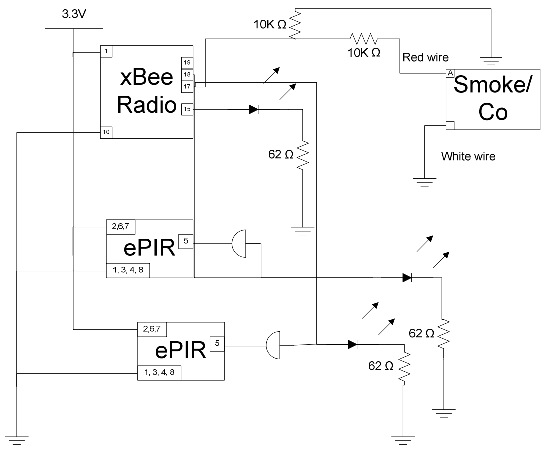


Figure 4. Safety End Node

3) Entrance Node

The entrance end node consists of an RFID reader and a motion sensor. The purpose of this node is to work as an enable/disable for the system when the user exits or enters the house. The motion detector has a dual purpose: one to function as a doorbell to tell the homeowner there is a visitor and/or wake up the radio to transmit RFID data.

This node transmits two types of data packages to the base station. One package is the RFID data, which tells the base station which RFID tag was just read, and the other is the motion sensor data, both of which are edge triggered. The RFID tag is decoded in the base station. Unlike the other radios this one is programmed to have Pin wakeup. This is configured so that the radio wakes up when the motion sensor is tripped to be ready for the RFID read to take place. The radio also does cyclic sleeping just to stay in the network. The diagram for this end node is shown in Fig. 5.

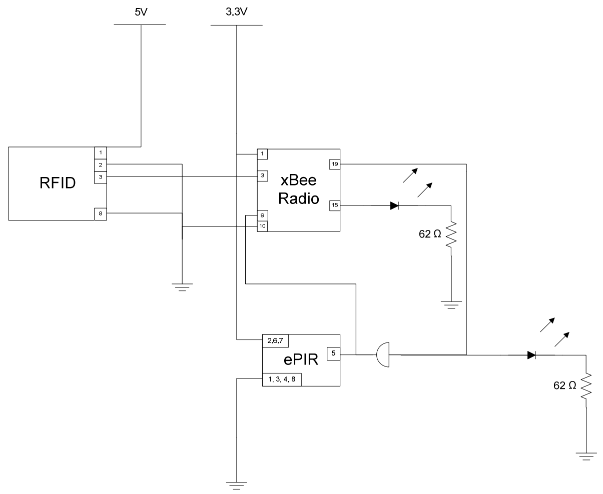


Figure 5. Entrance Node

4) Camera Node

The camera node's purpose is to provide a video feed into the house allowing the home owner to view their house via the internet. For a larger viewing angle a stepper motor was added to the node that allows the camera to pan to different locations on the owners command. For control of the motor there is the Easy Driver that provides easy control for the motor, and an xBee radio to receive remote commands from the base station. However, some decoding to interface between the radio and motor was required so another microcontroller was utilized. This allows for simple left/right control of the motor, and since there is already a full processor on the node, more advanced commands can be implemented in the future.

The radio on this end node is configured to receive data from the base station with instructions on what to do with the camera. Since the radio can be sent data at any time, sleep mode was disabled. This makes sure that no data sent by the base station is lost.

Another difference for this node is that the motor needed a 12V supply, therefore a 12V wall jack was used, and the motor draws its current and voltage directly from it, and not from a voltage regulator like the other components.

The camera, axis 2100 network camera, was chosen for this prototype because of its simplicity. The camera has its own networking capabilities and web server. Therefore, the camera just needs an Ethernet cable and it’s good to go. Not only does it show its live video feed online, but it also gets its power from the Ethernet cable. The live video feed was then embedded in the webserver of the base station and there is only one website that is needed for the system.

The stepper motor is simple and small size but at the same time supplies sufficient torque for the prototype. This motor is also compatible with the easy driver, which allows for very effortless control. The diagram of the camera node is shown in Fig. 6.

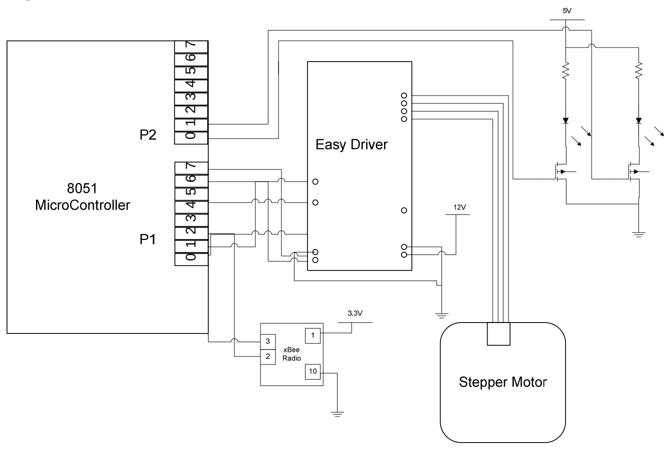


Figure 6. Camera Node

III. Results, Conclusions, and Future Work

The smart home described in Section 2 was successfully designed and a complete working prototype system implemented by the end of the 10-week summer REU program. The smart home security system implements many different sensors to provide a safe environment for home owner and all that may reside with them. As stated earlier in the paper, these sensors include a motion sensor, temperature sensor, security camera, and various other sensors. All of the sensors work together to provide maximum coverage for the home owner. An android app and a security web page have also been constructed to aid in providing protection and enhanced monitoring.

In the future, several improvements can be added to the smart home security system to allow for more comfort and protection. These include but are not limited to: a full climate control interface to provide a comfortable living condition for the home owner, a lighting control system to allow the home owner to turn off/on lights when they are not at home, more security cameras to view more of the house at one time, and a keypad to allow the user to add more RFID tags to the database without reprogramming the base station’s microcontroller. Various other nodes can also be implemented to add to the coverage area.

References

[1] Muhammad Ali Mazidi, Janice Gillispie, and Rolin D. McKinlay. (2006) *The 8051 Microcontroller and Embedded Systems: Using Assembly and C*, 2nd ed., Upper Saddle River, New Jersy: Pearson Education, Inc.

[2] (2010) *Arduino*[online], available: http://www.arduino.cc/ (available April 2011).

[3] Burnette, Ed. (2008) *Hello Andriod*, United States of America: Pragmatic Bookshelf.

A SystemC-TLM Platform for Wireless Sensor Networks Design Exploration

D. Serna[1], S. Villa[1], F. Rivera[2] and J. Aedo[1]

[1]Electronics Department, Universidad de Antioquia – ARTICA, Medellin, Colombia
[2]Computer Science Department, Universidad de Antioquia – ARTICA, Medellin, Colombia

Abstract**—***Wireless Sensor Networks (WSN [1]) have gained worldwide attention in recent years, due to their potential for development of applications in various areas: health, environmental monitoring, control and monitoring processes, etc. Due to the complexity of WSN design, it is necessary to employ design methodologies at a very high level of abstraction which allow the exploration of diverse alternatives of hardware and software architectures. In this research a generic platform using SystemC TLM (Transaction Level Modeling) to support the design of system-level WSN was developed. The platform allows the system's functional evaluation at a very high level of abstraction. A virtual functional model of a WSN could be built using platform components to estimate system performance and power consumption. A model of a WSN using the MSP430 Microcontroller, the CC2420 radio transceiver and a wireless network model were built to show the effectiveness of this approach. The simulation results were highly satisfactory.*

Keywords: *SystemC, Wireless Sensor Networks, modeling platform.*

I. INTRODUCTION

The design of WSN-based systems should consider many factors such as energy consumption, network lifetime, scalability, robustness, simplicity, network connectivity and information security, among others, while ensuring QoS metrics [1]. The optimization of these factors is a non-trivial task, and has a direct impact on the design process complexity, thus directly affecting design times [2]. Therefore, it is essential to have high level design methodologies which simplify the management of the before mentioned complexity, reducing both the design and development times.

Some methodologies that have been proposed in the literature, such as model-based design [3] and platform-based design (PBD) [4] are based on predefined platforms that allow performing early design verification. A platform is defined as a library of communication and computing components, conformed by models that represent their capabilities, and provide an estimate of their physical characteristics. These various components can be instantiated to construct a wide set of architectures [2].

In this research a generic platform using SystemC-TLM is proposed for modeling and simulation of WSN. The platform allows system functional evaluation and validation in early design stages. It is also capable to estimate system performance in terms of instruction count and energy consumption, to emulate the instructions execution, exceptions, interruptions, memory mapping, and peripheral interaction. The platform is presented as a contribution to a WSN-based systems design methodology that is being developed in the ARTICA's embedded systems research group. Such methodology aims at the efficient administration of the complexity in designing a WSN-based system at a high level of abstraction.

The platform modeling is based on SystemC-TLM, which is an IEEE standard language for system-level design [5, 6]. The platform provides an executable model for wireless sensor networks with instruction-level accuracy, power consumption estimation, and hardware details. The executable model is capable of achieve high speeds due to the extensive use of the Transaction Level Modeling (TLM) methodology and standard [7]. TLM is used for communication modeling among components, and allows more flexibility and generality. The model flexibility allows adding, changing or removing different modules without affecting the behavior of others. Thus, the platform components can test different architectures for WSN-based systems in order to choose the best design decision.

The rest of the paper is distributed as follows: Section II describes related work that has been carried out in the field. Section III describes the proposed platform components. Section IV shows the experimental results obtained so far, and Section V presents the conclusions and future work.

[1] WSN: The Wireless Sensor Networks are distributed devices, self-employed, which use sensors to cooperate in a common task.

II. RELATED WORK

Various tools and description languages have been proposed for the design, modeling, and simulation of WSN-based platforms. Languages such as VHDL-AMS [8] and Verilog-AMS [9] have been used, both having some restrictions on simulation performance and interoperability [10]. Matlab has been proposed as a modeling and simulation tool, but the excessive complexity of hardware modeling inside it has made its use ineffective and error prone [11]. Other simulation tools using Java such Avrora [12] or COOJA/MSPsim [13] have been proposed. These tools allow the simulation of a WSN through instruction-level emulators of WSN nodes. One of the main drawbacks is the slow simulation speed due to the use of an interpreted language. Another limitation is their coarse resolution in power estimation [14]. A major concern in the design process is the development of virtual simulation models of WSN in order to test and validate the applications without the need of building real prototypes. These models should provide sufficiently precise estimates of performance and power consumption in order to carry out the exploration of different design alternatives.

Some approaches using SystemC to built virtual WSN prototypes have been reported [15, 16]. In [10, 17] a system level approach to model and simulate WSN was proposed. The main limitation of this work is that it does not consider the power consumption of the entire system. In [18] the modeling and simulation of a WSN using SystemC and SpecC were presented. In this paper, a simple WSN node is developed. The node is constituted by an Analog-Digital Converter (ADC), a microcontroller, and a transmitter. One drawbacks of this work is that the communication process among nodes is modeled using simple FIFO channels. In [19] the simulation of an ultra low power WSN was showed. In this research the power consumption of the whole system was estimated through simulation. The simulation model considers a network with multiple nodes and communication channels among them. This work lacks of a complete modeling of the wireless communication channel, fact that can decrease the accuracy of the power consumption estimation when performing simulation of the entire system.

III. PLATFORM COMPONENTS MODELING

In this research, a generic platform using SystemC-TLM is proposed for modeling and simulation of WSN. The platform allows the construction of functional simulation models of the entire network, in which the application is distributed across all nodes, even considering the use of an operating system (such as TynyOS, Contiki or FreeRTOS). A special strategy was developed to simulate with instruction level accuracy, and to estimate power consumption at both the node and the entire network. Thus the platform enables the exploration of different alternatives to implement the application in an efficient way.

The components of the platform are designed using SystemC-TLM, in which communication among modules is modeled with function calls without considering pin-level detail. The SystemC-TLM provides a great flexibility for the modeling. Thus it is possible to instantiate different components of a node in order to verify various alternatives in the design process, estimating the performance and the power consumption.

As a study case of our modeling approach, it was built a system based on a Shimmer node [20], which is a small WSN system aimed to health monitoring applications. The simulation model structure is show in Figure 1. A functional model for MSP430 processor, for the CC2420 radio transceiver chip, and for the monitors that estimate the cycle's number and power consumption was built. Every instantiated node will communicate with the others via the channel model built in SystemC as well.

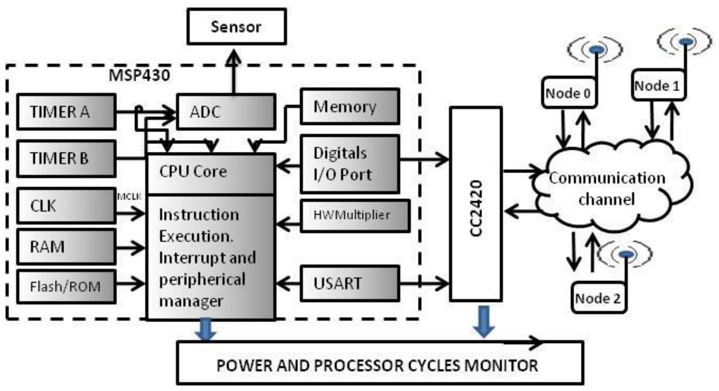


Fig. 1. Architecture of WSN using the platform components.

A. *Platform structure*

As mentioned before, the Shimmer system could be used to record and transmit physiological data in real-time. The application developed to test the models consisted on two nodes to monitor the heart rate and a fall detector, which will be implemented as part of a portable health monitoring device.

The developed model emulates the behavior and functionality of the Shimmer system with instruction accuracy and power consumption estimation. The simulation platform consists on five modules which will be described below:

a) Texas Instruments ultra-low-power MSP430 Microcontroller, which incorporates a 16-bit RISC CPU, peripherals, and a flexible clock system which are interconnect using a von-Neumann common memory address bus (MAB) and memory data bus (MDB). The

MSP430 architecture, combined with five low power modes is optimized to achieve extended battery life in portable measurement applications [21]. Our simulator provides a functional implementation of the MSP430. The processor model is composed of a core and various peripherals. This model is showed in Figure 2. The core model uses an instruction-set simulator (ISS) for emulating the processor behavior on a simulation host machine. The ISS uses interpretive simulation, in which each instruction of the target program is fetched from memory, decoded, and executed. The core of our MSP430 model handles the different modes of operation of the processor and manages interruptions and peripherals. The interruption controller can handle interruption priorities and nested interruptions. The processor model runs the binary code generated by the compiler, including the operating system and the embedded application. Our model uses the *Blocking transport interface* of the OSCI TLM-2 standard for modeling communication between MSP430 core and peripherals, thus allowing more flexibility. The generic payload extensions mechanism for the transaction type was used; these extensions represent a new set of attributes, transported along with the transaction object. Extra attributes for the transaction object were created because the MSP430 architecture has 8-bits and 16-bits peripherals. In the MSP430 model, each peripheral module exports a function *`void b_transport(TRANS &tra)`*. To communicate to some peripherals, the MSP430 core creates a transaction object, by providing an address, a command (either reading or writing), a pointer to data, a data size, and an extension (to indicate 8-bits or 16-bits operation). After this, the core calls the *`b_transport`* function on this object. The bus will forward the transaction to the appropriate peripheral according to the memory map. Eventually, the *`b_transport`* method of the corresponding peripheral module will be executed. Thus the MSP430 is compatible with any peripheral which follows the same communication protocol for transactional modeling.

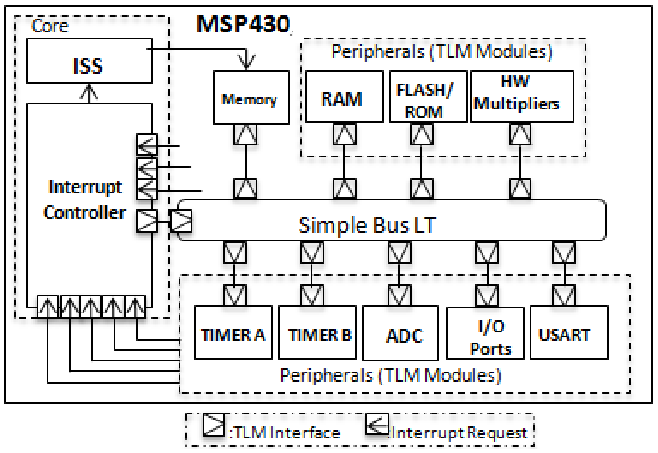


Fig. 2. MSP430 model.

b) CC2420 chipset, which is a radio transceiver that operates at 2.4GHz with the IEEE 802.15.4/ZigBee standard. It is designed for low power consumption and is widely used by most commercial platforms of WSN. The CC2420 provides extensive hardware support for packet handling, data buffering, burst transmissions, data encryption, data authentication, clear channel assessment, link quality indication and packet timing information [22]. This simulator provides a functional implementation of the CC2420, which will include configuration register for status, register for instructions, RAM for transmitting and RAM for receiving packages. The CC2420 architecture model is shown in Figure 3. In Figure 3. A general block diagram of a CC2420 is presented, which is connected to a *Node_proxy* instance through TLM interfaces. *Node_proxy* is a class of the SystemC Network Simulator Library (SCNSL) [23] used to decouple the node implementation of the network simulation. SCNSL allows us to collect accurate statistical data about the network's behavior.

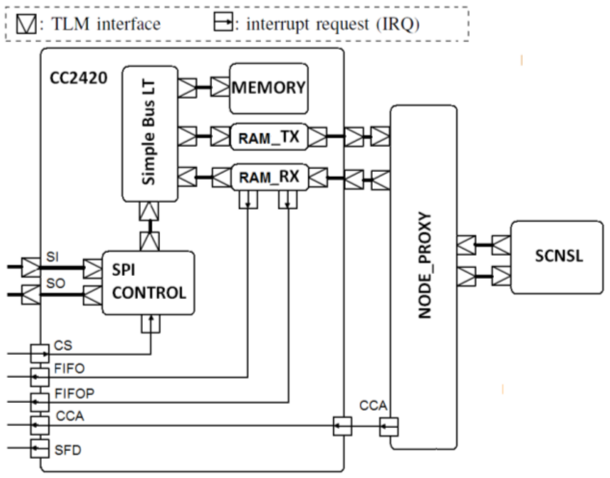


Fig. 3. CC2420 model.

c) The Power Consumption and Processor Cycles Monitor is the module responsible for assessing the performance of the entire network's model. The monitor is not a part of the Shimmer's architecture, but it is modeled to keep track of the node's energy consumption and to record the CPU cycles used in the execution of the different tasks that run inside each node. Each change in the mode of operation or use of a given peripheral updates the power monitor. Each executed instruction updates the CPU cycles counter. So when the execution is completed, the monitor delivers a log of the consumed energy and the used CPU cycles.

d) The sensor module uploads a database of ECG measures taken in the ARTICA's embedded systems laboratory, and passes the data to ADC processor model. The sensor module is annotated with power consumption taken from the sensor manufacturer's data sheets. The integration of a functional model of the ECG sensor with our simulator is still in progress.

e) The network interface responsible for communication between different nodes uses the SystemC Network Simulator Library, which allows us to collect accurate statistical data about the network's behavior. The SCNSL allows modeling network scenarios with different kinds of nodes described at different levels of abstraction. SCNSL was modeled at TLM level and has the same performance as the NS-2 simulator [23]. In our simulator, the devices are modeled in SystemC and their instances are connected to a module that reproduces the behavior of the communication channel, propagation delay, interference, collisions, and packet loss, allowing the design exploration of the entire network space. To use the library, the nodes are modeled by SystemC primitives for the specification of its internal behavior and the SCNSL primitives are used to model communication by sending and receiving packets through the network, using the communication channel model. The SCNSL kernel is responsible for the correct network simulation, executing events in the correct temporal order and taking into account channel physical features such as, propagation delay, signal loss, and so forth.

To build a platform specific model, it is necessary to integrate the components listed above. To interconnect modules, an instance of the desired component is created, and the sockets binding are then carried out. Each output socket of a component model corresponds to an input socket in the other. For proper communication among modules, it is necessary to modify the data sent in each transaction so that the model works according to the desired architecture.

IV. Experimental Results

Component models for a processor with some peripherals and the memory have been developed. The memory model is uploaded with the application's object code generated by the toolchains provided by Texas instruments. The processor model decodes and executes instructions, delivering the CPU cycles spent on each instruction and also reporting the internal status of the CPU. Figure 4 shows a simulation of the processor running a test program that uses all the available addressing modes and the entire instruction set. The instruction decoding and the timing accuracy of the model were compared with Texas Instrument IAR [25] simulator with the same results of CPU cycles used by the test application in both simulators. The model of the platform was compiled in debug mode to print the register internal state, to help the designer to debug programs. Only a fragment of the simulation is shown to save space.

```
Instruction: 4269  op: 4 Instruction MOV: 4  4
RegBank[0  ]:  404e  RegBank[1  ]:  0  RegBank[2  ]:  0  RegBank[3  ]:  0
RegBank[4 ]: 0 RegBank[5 ]: 0 RegBank[6 ]: 0 RegBank[7 ]: 0 RegBank[8
]:  0  RegBank[9  ]:  4  RegBank[a  ]:  0  RegBank[b  ]:  0  RegBank[c  ]:  0
RegBank[d ]: 0 RegBank[e ]: 0 RegBank[f ]: 0
CPU CYCLES: 1
 Instruction: 4364  op: 4 Instruction MOV: 2  2
RegBank[0  ]:  4050  RegBank[1  ]:  0  RegBank[2  ]:  0  RegBank[3  ]:  0
RegBank[4 ]: 2 RegBank[5 ]: 0 RegBank[6 ]: 0 RegBank[7 ]: 0 RegBank[8
]:  0  RegBank[9  ]:  4  RegBank[a  ]:  0  RegBank[b  ]:  0  RegBank[c  ]:  0
RegBank[d ]: 0 RegBank[e ]: 0 RegBank[f ]: 0
CPU CYCLES: 2
 Instruction: 4075  op: 4 Instruction MOV: 3  3
RegBank[0  ]:  4054  RegBank[1  ]:  0  RegBank[2  ]:  0  RegBank[3  ]:  0
RegBank[4 ]: 2 RegBank[5 ]: 3 RegBank[6 ]: 0 RegBank[7 ]: 0 RegBank[8
]:  0  RegBank[9  ]:  4  RegBank[a  ]:  0  RegBank[b  ]:  0  RegBank[c  ]:  0
RegBank[d ]: 0 RegBank[e ]: 0 RegBank[f ]: 0
CPU CYCLES: 4
```

Fig. 4. Instruction decode with cycle accuracy.

Component models were developed at different levels of abstraction in SystemC. The execution times of a test application running from models coded in the 3 different levels of abstraction are shown in Table 1. The Column 2 shows the simulation time of a functional model, which is similar to the simulation time using SystemC-TLM DMI (column 4). For each abstraction level, a simulation speed analysis was performed, and the best simulation performance was obtained when using TLM, and more specifically the Direct Memory Interface (DMI) [24].

Execution Time	SystemC without TLM	SystemC-TLM	SystemC-TLM with DMI
Real	4m39.875s	6m21.123s	4m6.445s
User	0m2.024s	0m1.864s	0m0.940s
System	0m1.148s	0m1.236s	0m0.548s

Table 1. Execution times of models in three different abstraction levels in SystemC.

Preliminary experiments over each node show that it is possible to obtain very accurate measures of performance and power consumption when compared to the actual behavior of both the node and the sensor network. Currently, we are porting the complete WSN application using two nodes.

The platform that is being developed is the base of a platform-based design methodology (PBD) for wireless sensor networks, in which the ARTICA's embedded systems research group is currently working on. This methodology seeks to handle the design complexity of WSN-based systems, allowing the comparison of diverse hardware-software architectures at a high level of abstraction.

CONCLUSIONS

In this research a generic platform using SystemC-TLM to support the system-level design of WSN was developed. The platform allows the system functional evaluation at a very high level of abstraction. A virtual functional model of a WSN could be built using platform components to estimate system performance and power consumption. A model of a WSN using the MSP430 Microcontroller, the CC2420 radio transceiver, and a wireless network model were built to show the effectiveness of this approach. A SystemC-TLM with DMI to improve simulation times while obtaining good speed-accuracy tradeoff was used. Preliminary experiments over each node of the WSN show that it is possible to obtain very accurate measures of performance and power consumption when compared to the actual behavior of both the node and the wireless network.

ACKNOWLEDGMENT

This research has been supported by Colciencias, Colombia ICT Minister and ARTICA (excellence research center) within the project “Design Methodology of Embedded Systems with High Reliability and Performance Focused on Developing of Critical Applications.”

REFERENCES

[1] L. Wang y Y. Xiao, “Energy saving mechanisms in sensor networks,” Broadband Networks, 2005. BroadNets 2005. 2nd International Conference on, 2005, Págs. 724-732 Vol. 1.

[2] A. Sangiovanni-Vincentelli, “Quo Vadis, SLD? Reasoning About the Trends and Challenges of System Level Design,” *Proceedings of the IEEE*, vol. 95, 2007, págs. 467-506.

[3] F. Losilla, C. Vicente-Chicote, B. Álvarez, A. Iborra, P. Sánchez, “Wireless Sensor Network Application Development: An Architecture-Centric MDE Approach”, Springer-Verlag Berlin Heidelberg 2007.

[4] A. Bonivento, L. P. Carloni, A. Sangiovanni, “Platform Based design for wireless sensor network”, Springer-Mobile Netw Appl. 2006.

[5] SystemC website http://www.systemc.org [On line. Cited: 10-02-2010].

[6] T. Grötker, S. Liao, G. Martin, S. Swan. “System Design with SystemC”. © 2002 Springer – Verlag.

[7] D. Black, J. Donovan, B. Bunton, A. Keist. “SystemC: From the Ground Up”. Springer Science+Business Media, LLC. 2010.

[8] J. Ravatin, J. Oudinot, S. Scotti, A. Le-clercq, and J. Lebrun, “Full transceiver circuit simulation using VHDL-AMS,” Microwave Engineering, May 2002.

[9] F. Pecheux, C. Lallement, and A. Vachoux, “VHDL-AMS and Verilog- AMS as Alternative Hardware Description Languages for Efficient Modeling of Multi-Discipline Systems,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems(TCAD), Feb. 2005.

[10] M. Vasilevski, F. Pecheux, N. Beilleau, H. Aboushady, y K. Einwich, “Modeling and Refining Heterogeneous Systems With SystemC-AMS: Application to WSN,” Design, Automation and Test in Europe, 2008. DATE '08, 2008, págs. 134-139.

[11] M. Mostafizur, F.Gregoretti, L. Lavagno. “A Framework for Modeling, Simulation and Automatic Code Generation of Sensor Network Applications”. SECON '08. 5th Annual IEEE Communications Society Conference, 2008.

[12] B. Titzer, D. Lee, y J. Palsberg, “Avrora: scalable sensor network simulation with precise timing,” Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on, 2005, págs. 477-482.

[13] J. Eriksson, F. Osterlind, T. Voigt, N. Finne, S. Raza, N. Tsiftes, y A. Dunkels, “Demo abstract: Accurate power profiling of sensornets with the COOJA/MSPsim simulator,” Mobile Adhoc and Sensor Systems, 2009. MASS '09. IEEE 6th International Conference on, 2009, págs. 1060-1061.

[14] Jun Rao, H. Pirahesh, C. Mohan, y G. Lohman, “Compiled Query Execution Engine using JVM,” in Data Engineering, 2006. ICDE '06. Proceedings of the 22nd International Conference on, pág. 23, 2006.

[15] G. S. Besrra, J. de Medeiros, H. Madureira. “System-level Modeling of a Reconfigurable System on Chip for Wireless Sensor Networks Applications”. InternatIonal ConferenCe on IntellIgent and advanCed SyStemS,2010. Icias 2010.

[16] G. S. Beserra, J.F Eusse, A. Sampaio, J. Pimentel, R.JacobiI, “A VHDL-AMS Co-Simulation of a System on Chip for Wireless Sensor Networks” . IBERCHIP Workshop 2010, IBERCHIP 2010 Proceedings p. 1-5.

[17] M. Vasilevski, N. Beilleau, H. Aboushady, F. Pecheux. “Efficient and Refined Modeling of Wireless Sensor Network Nodes Using SystemC-AMS”. Research in Microelectronics and Electronics, 2008. PRIME 2008. Ph.D. Pags: 81 – 84.

[18] M. Rafiee, M. Ghaznavi-Ghoushchi, S. Kheiri, y B. Seyfe, “Modeling and Simulation of Wireless Sensor Network (WSN) with SpecC and SystemC,” Computer Engineering and Technology, 2009. ICCET '08. International Conference on, 2009, págs. 515-519.

[19] J. Haase, M. Damm, J. Glaser, J. Moreno, C. Grimm. "SystemC-based Power Simulation of Wireless Sensor Networks"Specification & Design Languages, 2009. FDL 2009. Pags. 1 – 4.

[20] Shimmer website http://www.shimmer-research.com/ [On line. Cited:10-02-2010].

[21] MSP430x1xx Family. User's guide.

[22] Chipcon Products from Texas Instrument CC2420 Data Sheet.

[23] F. Fummi, D. Quaglia, F.Stefanni. "A SystemC-based Framework for Modeling and Simulation of Networked Embedded Systems". Forum on Specification, Verification and Design Languages, 2008. FDL 2008. Pags. 49 – 54.

[24] OSCI Tlm-2.0 Language Reference Manual. Website http://www.systemc.org [On line. Cited: 10-02-2009].

[25] IAR User and Reference guide. Website http://www.iar.com/website1/1.0.1.0/225/1/. [On line. Cited: 11-05-2010].

Implementation and Functional Verification of Soft IP Core of USB 3.0 Device MAC Layer

Hasan Baig, Jeong-A Lee
Department of Computer Engineering
Chosun University, South Korea

Abstract – *Universal Serial Bus has supported a wide variety of devices from keyboard, mouse, flash memory, imaging up to high speed broad band devices. In addition, user applications demand a higher performance connection between the PC and other increasingly sophisticated peripherals. USB 3.0 addresses this need by adding even faster transfer rates. It assures a data transfer rate of 4.8 Gbps as compared to its former interface USB 2.0 which has a raw data rate at 480Mbps. This implementation of synthesizable Media Access (MAC) layer of SuperSpeed USB 3.0, with a pipelining concept of processing the packets, is proposed to support high speed transfer rate and high throughputs. Alongside, the use of efficient handshaking signals complies with optimum performance of the overall device. Master controller has also been implemented to command over MAC Layer and other layers that will be implemented in a future research. This implementation meets the required specifications [1].*

Keywords: USB 3.0, MAC Layer, Physical Layer Controller, FPGA.

4 Introduction

The physical layer classifies the PHY portion of a port and the physical connection between a downstream facing port (on a host or hub) and an upstream facing port on a device. The SuperSpeed physical connection is comprised of two differential data pairs, a transmit path and a receive path (Fig. 1).

The physical layer classifies the PHY portion of a port and the physical connection between a downstream facing port (on a host or hub) and an upstream facing port on a device. The SuperSpeed physical connection is comprised of two differential data pairs, a transmit path and a receive path (Fig. 1).

The electrical aspects of each path are characterized as a transmitter, channel, and receiver; these collectively represent a unidirectional differential link. Each differential link is AC-coupled with the capacitors located on the transmitter side of the differential link. The channel includes the electrical characteristics of the cables and connectors [1].

At an electrical level, each differential link is initialized by enabling its receiver termination. The transmitter is responsible for detecting the far end receiver termination as an indicator of a bus connection and informing the link layer so the connect status can be factored into link operation and management.

When receiver termination is present but no signaling is occurring on the differential link, it is considered to be in the electrical idle state.

When in this state, Low Frequency Periodic Signaling (LFPS) is used to signal initialization and power management information. The LFPS is relatively simple to generate and detect and uses very little power.

The USB PHY Layer (PHY Chip depicted in Fig. 1) handles the low level USB protocol and signaling. This includes features such as; data serialization and deserialization, 8b/10b encoding, analog buffers, elastic buffers and receiver detection. The primary focus of this block is to shift the clock domain of the data from the USB rate to one that is compatible with the general logic in the ASIC [1].

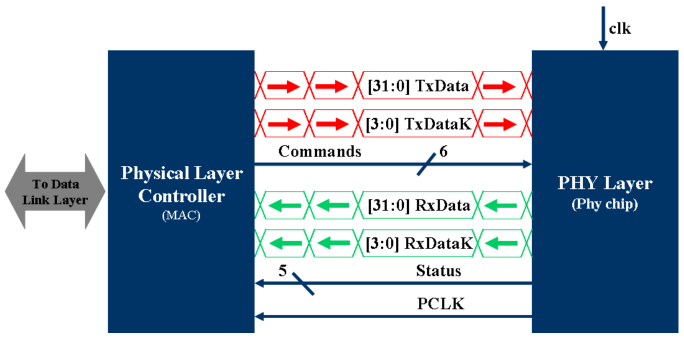


Fig 1: PHY/MAC Interface.

2 MAC Interfaces

Since the PIPE (PHY Interface for the PCI Express) is implemented for USB mode that supports 5.0GT/s, we have chosen 32 bits data paths with PCLK running at 125MHz [2]. The MAC Layer commands the communication of PHY Layer with the Link Layer and LTSSM (Link Training and Status State Machine). PHY layer controller itself is commanded by Master Controller. The top level block diagram of MAC Layer (or Physical Layer Controller)[1] is shown in Fig. 2.

[1] Phy Layer Controller is also called Media Access (MAC) Layer. We will use these terms interchangeably throughout this paper.

It can be observed that the PHY Layer Controller itself comprises of some internal modules that will be described later in the following sub sections.

The MAC layer of USB 3.0 device interacts with Link layer, LTSSM and is commanded by Master controller. LTSSM and Link Layer are beyond the scope of this research paper, so will be described briefly in the next sub sections.

2.1 Link Layer

A Super Speed link is a physical and logical connection between two ports, called link partners. A port has a physical part and a logical part. The link layer identifies the logical portion of a port and the communications between link partners. The main responsibility of link layer is to ensure the successful data transfer with the link partner and to maintain connectivity between them.

2.2 Link Training and Status State Machine

The Link Training and Status State Machine (LTSSM) behaves as a leading workhorse to maintain reliable link, highly optimized power consumptions and efficiently fast and perfect data transfer rate. It also implements various algorithms for link's reliability preservation and is also liable to recover a link when an error arises.

It is LTSSM who manages the power of a device proficiently and greatly reduces the link's power consumption. It also voids the condition that causes the wastage of power.

It co-ordinates and converse with PHY chip, MAC Layer, Link Layer and Master Controller to perform it's duties.

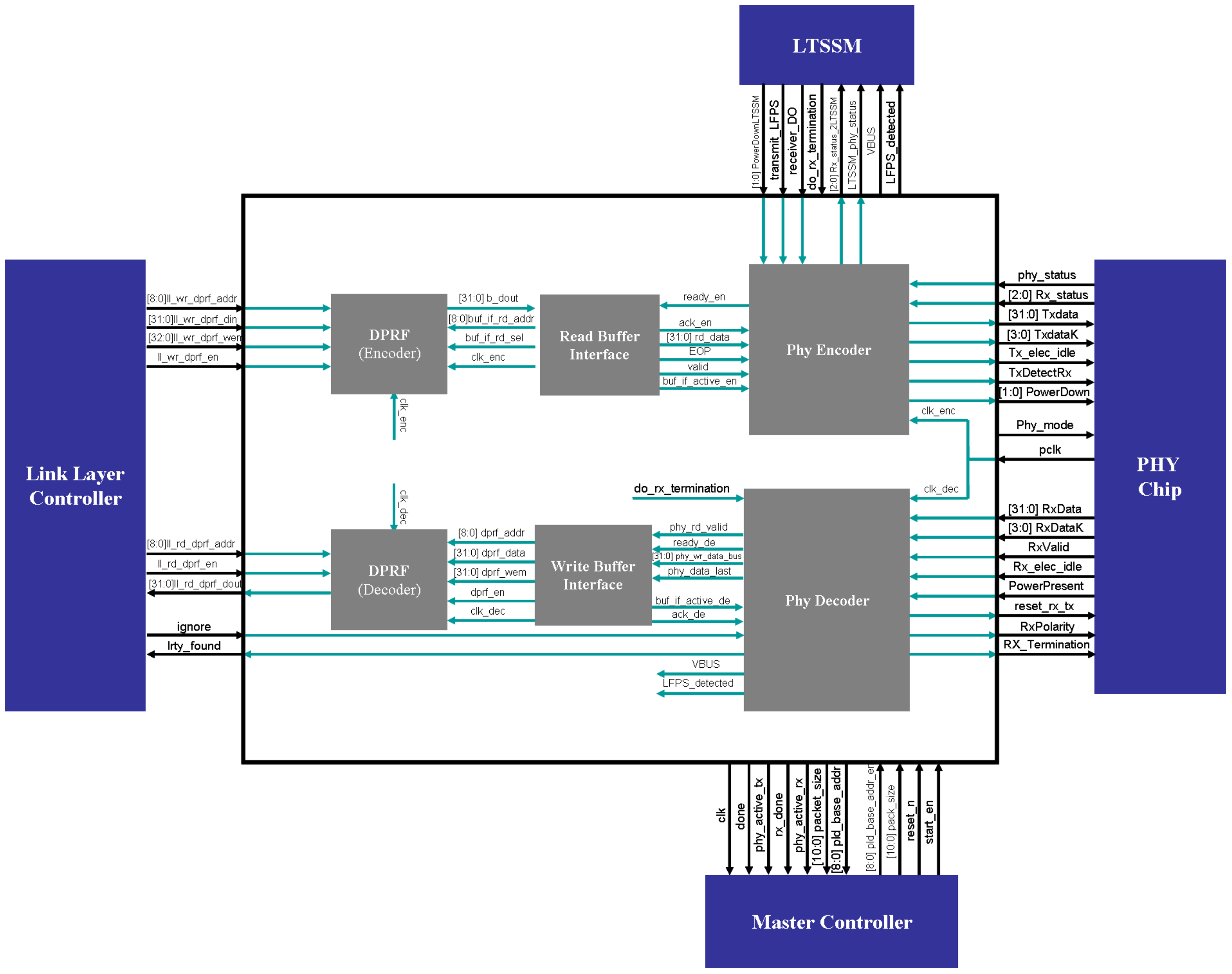


Fig 2: Top Level Block Diagram of PHY Layer Controller.

2.3 Dual Port Reference Memory

USB 3.0 specification [1] provides a complex hardware implication. It has been emphasized to produce such an architecture that can be easily comprehended, incorporated and implemented without an extraordinary knowledge of interfacing other layer in USB 3.0 device.

To achieve this, each layer is kept separated from the other by inserting dual-port-memories (Fig. 2) in between two successive layers. When one of the layers is done with writing data to intermediate memory (dual-port-memory), there is a primary need of notifiying the next concerned layer to begin execution and process the valid memory contents in the intermediate memory area. This need is accomplished by using the Master Controller which schedules the execution of layers in a pre-determined sequence which is described in next section.

2.4 Buffer Interfaces

The primary reason of using buffer interfaces (Fig. 2) is to overcome the need of incorporating a memory controller into a layer's main controller. In case, for example, if MAC layer is instructed to start processing some valid memory contents, there could be two possibilities – First MAC layer fetch the memory contents by issuing address & data to enable ports of the memory with incrementing each time the address for the next valid data and asserting the enable ports.

Second possiblity is that it has a separate module which is notified of the number of bytes to be fetched from memory and which is resposible of incrementing the address each time it gets valid data for memory. In order to simplify the implementation, it is recommended to have separate entities so that hardware can be easily comprehended and debugged or in another words the main controller will remain free from some extra burden while dealing with the memory. Second approach is quite better. Thus buffer interface (or memory controllers) are used in the architecture just beside intermdiate dual-port-memories.

3 Master Controller

Master Controller is developed to command the communication flow between each module. The centralized master controller monitors and controls the decoding and encoding operation separately. It has been designed in such a way that it can easily be integrated later with Link Layer, Protocol Layer and LTSSM, which would be in fact a future enhancement of this research. Top level block diagram of Master Controller is shown in Fig. 3.

3.1 Decoding Path Controller

The decoding process is to take packet from the PHY chip and pass it to link layer controller (decoder) and so forth. Master controller follows the protocols in the sequence mentioned below.

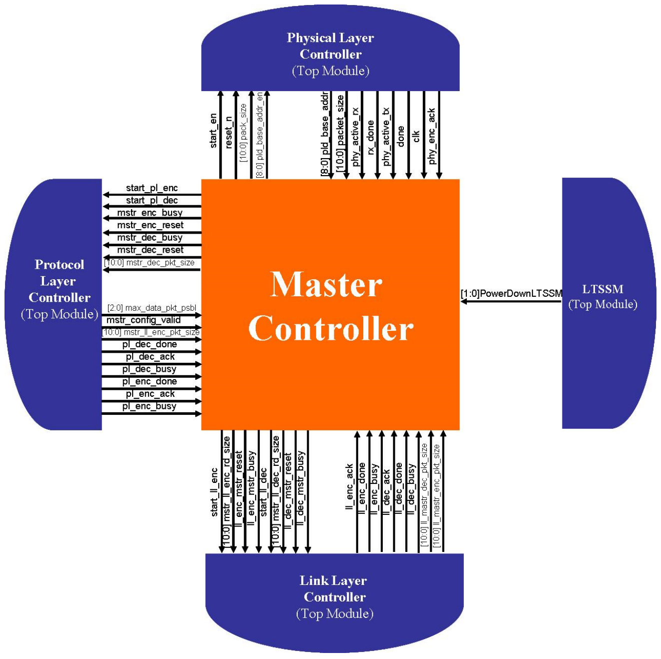


Fig. 3: Top Level Block Diagram of Master Controller - showing IO interface with each layer and LTSSM.

1. When Phy Layer Decoder (Fig. 2) receives the complete packet, it generates an indication signal to master controller which in turn initializes the Link Layer (LL) decoder, provided that LL decoder is not already in a busy state. Meanwhile, master controller also sends the packet size to the LL decoder that it received from the Phy Layer decoder at the complete reception of packet.

2. When the packet is processed by the LL decoder, it generates an indication signal to master controller which in turn initializes the Protocol Layer (PL) decoder, provided that it is not already busy. Link layer decoder de-assembles the packet received and sends the new packet size (packet size changes after passing through the packet de-assembler) to master controller. Master then sends this new packet size to protocol layer decoder at the time of its initialization.

3.2 Encoding Path Controller

The controlling protocols, mentioned below, are followed by the master controller in order to encode the packet

1. Protocol Layer (PL) Encoder is initialized when master-configuration valid signal is received by the Master controller provided that the PL encoder must not already busy. As soon as the complete packet is encoded, PL encoder generates an indicating signal ("pl_enc_done", Fig. 3) to the master informing it the

packet has been transferred into the buffer and ready to be fetched by Link Layer controller. Master controller then generates a signal to initialize the Link Layer (LL) encoder, provided that LL encoder is not already in a busy state. Meanwhile, master also sends the packet size to the Link Layer encoder; it had received from the PL encoder at the complete reception of packet.

2. After processing, assembling and transferring the complete packet in the buffer, LL encoder generates an indication signal (“ll_enc_done”, Fig. 3) to master controller which in turn initializes the Phy Layer encoder, provided that it is not already busy. Link layer encoder also sends the new packet size (packet size changes after passing through the packet assembler) to master controller. Master controller then sends this new packet size to Phy layer encoder at the time of its initialization.

Master controller must deassert the initializing signal of Protocol Layer, Link Layer and Phy Layer Encoders as soon as they acknowledged.

3. Media ACcess (MAC) Layer or Physical Layer Controller

The main object of this research is the implementation of MAC Layer encoder and decoder that runs in parallel and hence ensures the concurrent in-out transaction of USB 3.0 protocols.

Before discussing the developed algorithm of MAC Encoder and Decoder, it is good to have a look at the standard USB 3.0 packet [1] first. It is also portrayed in Fig. 4. Refer [1] for detailed description of packet symbols.

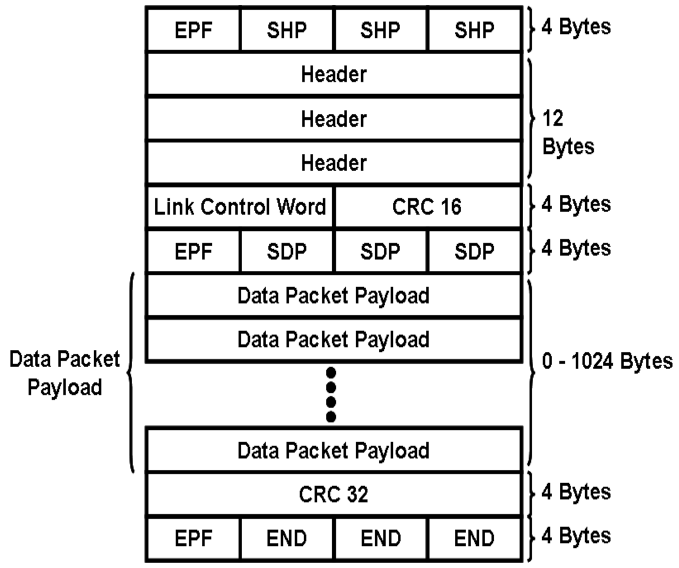


Fig. 4: Standard USB 3.0 packet with maximum of 1024 data bytes.

4.1 MAC Layer Encoder (Phy Encoder)

It is recommended to refer [2] first for PHY Chip encoding signals, in order to understand Phy Encoding algorithm. Algorithmic State Machine Description (ASMD) of PHY Encoder is shown in Fig. 5. When an encoding process is done by Link Layer controller, it asserts “ll_enc_done” (section 3.2), informing master controller that a valid data has been placed in dual-port-memory and must be fetched by Phy Encoder. Master controller then asserts “start_en” (Fig. 3) signal to initialize Phy encoder and waits for being acknowledged by Phy encoder.

LTSSM controls the power state of PHY chip through Phy Encoder. Phy chip remains idle in P1 and P3 power states [1]. In P2 state, encoder waits for the instruction from LTSSM either to force Phy Chip to transmit LFPS [1] or to do receiver detection operation (Fig. 5). When a valid data is present in the buffers, LTSSM instructs Phy Encoder to take Phy chip into P0 state. Encoder starts the process of fetching data, from buffer, only when a positive edge of “transmit” is seen asserted.

When LTSSM asserts “transmit” signal, encoder requests the data and waits for the acknowledgment from Read Buffer Interface. When transaction begins, encoder obtains the data payload size from the packet size (given by master controller, in terms of bytes) and puts into the register, named “data_pld_size”. The purpose of calculating the data payload size is to find out how many number of transactions are required to send the complete packet to Phy chip. Since each transaction can have 4 symbols of transmit data (32-bit bus) [refer 1 for detailed description], therefore a packet size is divided by 4 to obtain the correct number of transactions required. Referring [2], TxDataK bus indicates Control or Data byte in a current transaction.

The RTL of encoder is efficient enough to locate which byte is a control byte or data byte in a current transaction. Fig.4 depicts that there are two such transactions (1st and 6th) which have complete control symbols (bytes) in it. The last transaction should have all control bytes, but it depends on the data payload size. If data payload size is not a multiple of 4, then there must be an ambiguity which symbol is a control or a data byte, in 2nd last transaction. Two least significant bits of “data_pld_size” indicates the position of data byte in 2nd last transaction (Fig. 4).

4.2 MAC Layer Decoder (Phy Decoder)

Decoding process is pretty complicated and a challenging task. It is recommended to refer [2] to grasp the PHY Chip decoding signals. ASMD of Phy Decoder is shown in Fig. 6. “PowerState” of Phy Decoder is again in a control of LTSSM. Phy Decoder remains idle in P1 & P2 states. In P3, LTSSM asserts “receiver_DO” (See Fig. 2) signal when it requires “receiver detection” operation to be performed.

Phy Decoder in-turn asserts “TxDetectRx” signal [2], requesting PHY chip to begin “receiver detection” operation. This signal should remain high until “phy_status” signal [2] from Phy Chip is seen asserted. When the receiver detection operation is completed, PHY chip asserts “phy_status” signal [2]. Phy decoder then deasserts “TxDetectRx”, meanwhile informs LTSSM, the status of receiver through “Rx_status_2LTSSM” bus.

As soon as LTSSM instructs Phy decoder to take PHY Chip into the power state P0, decoder starts looking for “Rx_elec_idle” signal. Phy Decoder informs the LTSSM about LFPS on the basis of “Rx_elec_idle” signal. It then goes into “idle” state until valid data is present at “RxData” bus. When the valid data is present, decoder interrogates the “Write Buffer Interface” (Fig. 2) whether it is ready to accept the incoming data, and jumps to the “ackldg” (acknowledge) state.

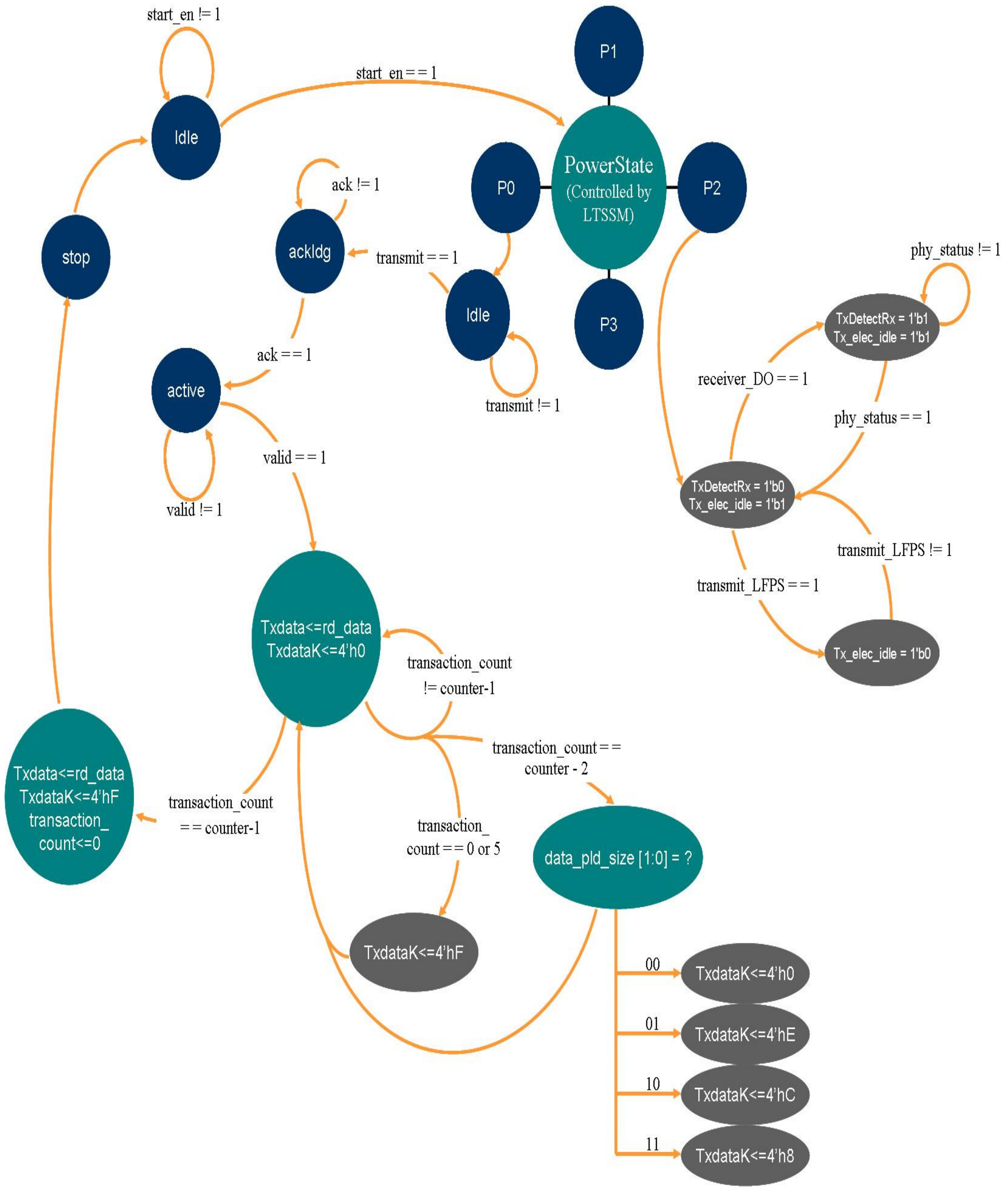


Fig. 5: ASMD of Phy Encoder (see Fig. 2 for block level diagram)

It then waits for the acknowledgment from “Write Buffer Interface”. As soon as the buffer acknowledges, decoder starts fetching and sending data from Phy Chip to Write Buffer Interface respectively (Fig. 2). Phy Decoder keeps on transferring packet from Phy Chip to Write Buffer Interface unless the Link Layer Controller asserts “ignore” signal.

When “ignore” is seen asserted, Phy decoder discards the incoming data from the Phy Chip and starts looking for LRTY [1]. Phy Decoder also calculates the size of packet while transferring data from Phy chip to Write Buffer interface. Fig. 4 depicts that a packet can have a maximum size of up to 1024 bytes (max data payload) + 28 bytes (standard protocol of each packet).

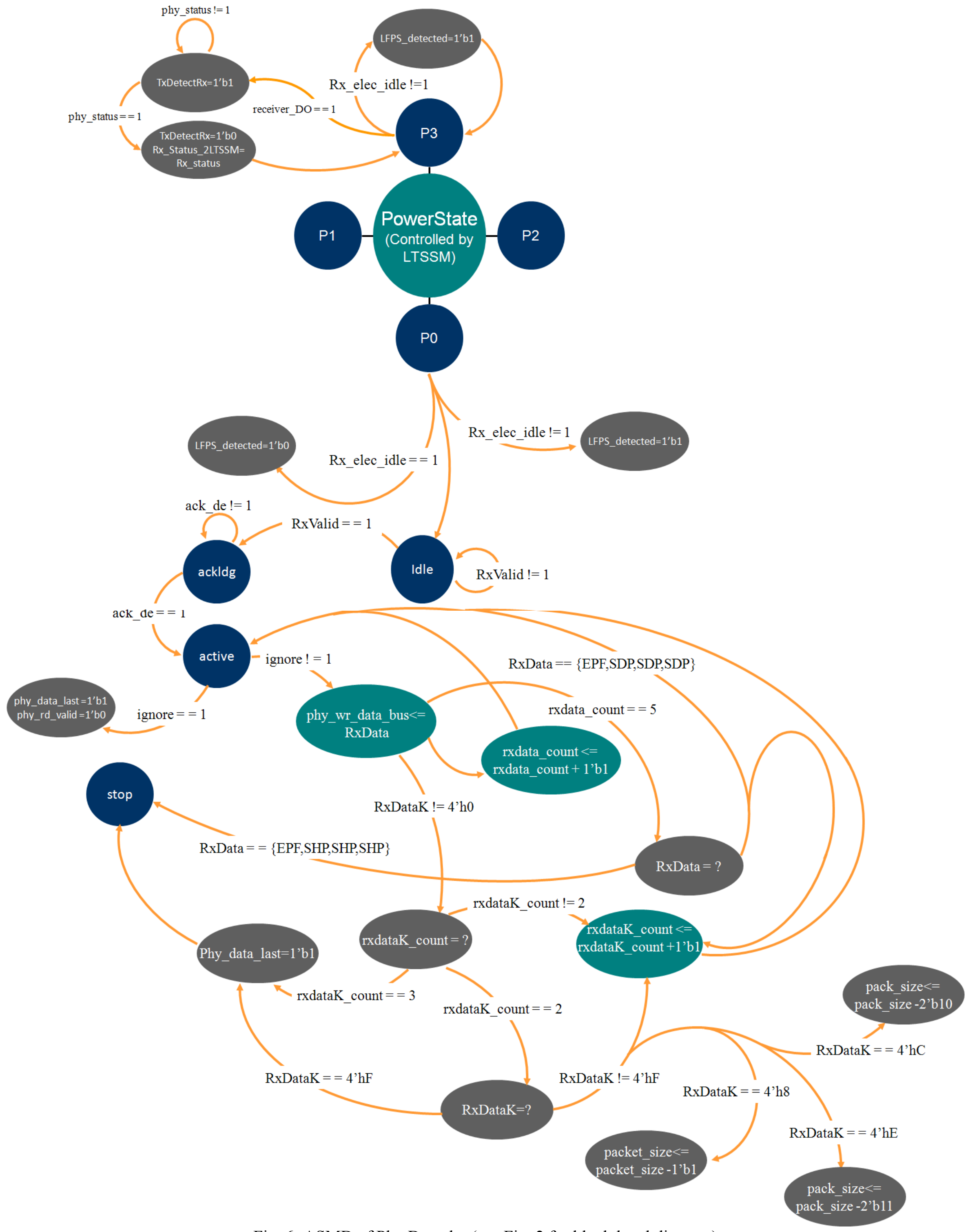


Fig. 6: ASMD of Phy Decoder (see Fig. 2 for block level diagram)

The packet size is calculated in such a way that a counter is incremented each time a transaction occurs. Decoder continuously monitors RxDataK lines. Control byte is indicated by RxDataK bus whenever its value is non-zero. Whenever a non-zero value is present at RxDataK lines, another separated counter is incremented to monitor the number of control byte transactions. Referring to the Fig. 4, it can be observed that there could only be 3 or 4 such transactions which have control bytes in it, i.e. the first transaction, the sixth transaction and the last transaction. There could be fourth control byte transaction when data payload size is not a multiple of 4 (i.e. first three of the last four control bytes can be a part of second last transaction).

Since the first and the sixth transaction is a complete control byte transaction, therefore one doesn't need to care about them. The problem arises after data pay load due to variations in the data payload sizes.

ASMD shown in Fig. 6 depicts that decoder repeatedly checks for "rxdataK_count" to become equal to 2. When "rxdataK_count" become equal to 2, decoder checks the value of RxDataK. RxDataK = 4'hF point towards that all the four bytes are control bytes and a current transaction is End of Packet. RxDataK, other than 4'hF, clearly indicates that the data payload size is not a multiple of 4 and the present transaction contains the data byte(s) along with the control byte(s). Also we would have fourth control byte transaction. If RxDataK = 4'h8 (4'b1000), it shows, there are 3 data bytes and 1 control byte. This one control byte is actually from the four of the last control bytes (shown in Fig. 4). This means that there will be only 3 (remaining) control bytes in the next transaction and the last byte will remain empty, thus a value of 1'b1 is subtracted from the size of packet (shown in Fig. 6). Similar method is implemented for RxDataK = 4'hC and 4'hE.

5 Functional Verification

Although the whole of the USB Device is written in synthesizable RTL code, this entity will be representing the behavior of the Host plus the behavior of the PHY Chip. It is meant only for simulation purposes and can never infer a hardware at all. It can supress the concept of separate layers and can accommodate the behavioral of the host entity and PHY Chip as a single entity which is needed to derive the MAC layer, appearing in the front-line of the upstream facing port (USB Device).

Random data is generated through a testbench and inputs to MAC Layer (assuming that it is coming from Link Layer, See Fig. 1, and Fig. 2) and a pre-defined packet size for each time a simulation runs. This data is processed by Phy Encoder via dual-port-memory and read buffer interface (Fig. 2). Phy Encoder passes this data to PHY Chip (behavioral model) which loop backs that to Phy Decoder.

Phy decoder remains idle unless RxValid signal (from behavioral of PHY, Fig. 2) is seen asserted. As soon as the rising edge of RxValid signal is sensed, decoder requests Write Buffer Interface to received data coming from the Host. Once it acknowledges, the Phy decoders starts fetching the data and place it on the ports facing write buffer interface which in turn place the data into the dual-port-memory (Fig. 2). Meanwhile it also looks for the control bytes (on RxData bus) on the basis of which it could find out the size of packet (See Section 4.2).

6 Conclusion And Future Work

Since SuperSpeed protocols are intended for dual simplex transmission lines, transmitting and receiving transactions in parallel, there is an absolute need of having the architecture which support such protocols.

In order to meet the requirements, separate encode and decode paths work concurrently and independently. Thus encode path is associated with packet assemblers or encoders while the decode path is associated with packet disassembler or decoders. Encode and decode paths are executed by the Master Controller State Machine to fulfill the dual simplex capability of the bus.

This synthesizable implementation of MAC Layer (Physical Layer Controller) follows the latest specification of USB 3.0. It is designed in such a way that other layers can be easily interfaced with it.

Link Layer, Protocol Layer and LTSSM will be developed in future as an independent entity and integrated with this layer. The future objective could be complete USB 3.0 memory device.

7 References

[1]. "Universal Serial Bus 3.0 Specification", *Hewlett-Packard Company, Intel Corporation, Microsoft Corporation, NEC Corporation, ST-NXP Wireless, and Texas Instruments*, *Revision 1.0, November 12, 2008.*

[2]. "PHY Interface for the PCI Express ™ and USB Architectures", *Version 2.90, Intel Corporation, 2007-08.*

[3]. "Universal Serial Bus Specification", *Revision 2.0, April 27, 2000.*

[4]. "On-The-Go Supplement to the USB 2.0 Specification", *Revision 1.3, December 5, 2006.*

[5]. "Inter-Chip USB Supplement to the USB 2.0 Specification", *Revision 1.0, March 13, 2006.*

[6]. "USB System Architecture (USB 2.0)", *MindShare, Inc., Don Anderson.*

[7]. "eXtensible Host Controller Interface for Universal Serial Bus (xHCI)", *Revision 1.0, 2010.*

[8]. Peter J. Ashenden, "*Digital Design: An Embedded System Approach using Verilog", Elsevier, 2008.*

FPGA Based EKF Estimator for DTC Induction Motor Drives

Yadollah Sabri , Virginie Fresse
Hubert Curien Laboratory UMR CNRS 5516
Jean Monnet University- University de Lyon
18 Rue du Professeur Benoît Lauras 42000
Saint Etienne, France
e-mail: yadollah.sabri@gmail.com
virginie.fresse@univ-stetienne.fr

Rachid Beguenane, Francis Okou
Department of Electrical and
Computer Engineering of RMCC
Stn Forces K7K 7B4
Kingston, Canada
Rachid.Beguenane@rmc.ca

Abstract – A Field Programmable Gate Array based Extended Kalman Filter estimator employed in Direct Torque Control system for Induction Motors is presented in this paper. The implemented algorithm of Extended Kalman Filter estimates the required state space variables of Induction Motor for determining the switching pattern of Voltage Scours Inverter. The implementation on FPGA including functional simulations, as well as the hardware in loop tests is presented.

Key words – FPGA; DTC Controller; Sensorless Control, EKF; Induction Motors

I. INTRODUCTION

Over recent years, several researches have been conducted with the aim of proposing alternative solutions to the Field Oriented Control (FOC) of PWM inverter-fed drives for Induction Motors. The main goal of these studies, in essence, was to reducing the complexity while maintaining the accuracy and effectiveness of the control system. Among those solutions, the Direct Torque and Flux (DTFC) control has gained a wide interest satisfying above mentioned conditions as alternative to the (FOC) control systems [1], [2]. Alongside with the increasing interests in the simplified control strategies there is also a growing demand to minimizing the cost of the hardware of control systems highlighting the importance of sensorless controllers. Several sensorless control strategies to estimate the rotor speed (thereby position) of Induction motors eliminating the needs to use their corresponding mechanical sensors have been also developed. Among these methods, the Extended Kalman Filter appears to be an efficient candidate as a robust online estimation towards random noise environment, [2], [3], [4].

It is evident, by considering the successively improving reliability and performance of digital technologies, that even the most complicated control methods are achievable by means of nowadays technologies. High speed Digital Signal Processors (DSP), for instance, because of their software flexibility and ability to perform very complex calculations, have been of the highest interests for those control systems requiring intensive mathematical computations. However, due to the their inflexible architecture, this kind of DSP have been proven to fail to offer sufficiently short execution time which is vital for stability of the controllers dealing with the rapid systems, i.e. the systems with very small time constants. The Field Programmable Gate Arrays (FPGA) technology has, fortunately, appeared to be the solution for overcoming the above mention problem. This technology, providing a flexible architecture, makes it possible to designate the appropriate duties to be shared by hardware and software facilities so as the main goal of minimizing the overall execution time and/or the resource usage to be achieved. The application of FPGA covers a vast area such as signal processing, mathematical computation, control systems, target tracking, navigation and robotics [5], [6], [7]. The implementation of an FPGA, however, faces a major drawbacks which are the complicated and intensive operations such as multiplication and division that demands high computational resources [6], [8].

An FPGA-based EKF has been implemented in this study to be used in DTC controller for Induction Motor Drives. Unfortunately, the whole DTC controller was not implemented due to the lack of laboratory facilities such as Induction Motor-Load, VSI inverter, so that the application was limited to the implementation of EKF estimators using Xilinx ML506 Evaluation Platform. The obtained results from MATLAB complete DTC simulation with those obtained from FPGA were compared to examine the effectiveness of the EKF estimator thus implemented.

Over subsequent sections, the description of the Extended Kalman Filter will be presented including the main features of DTC controller, this section will be followed by description of EKF algorithm refinement and simulations. The development of the FPGA-based EKF algorithm will be then explained providing the results of the Hardware in Loop. The time/ area performance of implemented EKF will be discussed at the final section of this paper.

II . DESCRPTION OF THE EXTENDED KALMAN FILTER (EKF)

A. The state space model of induction motor and description of DTC control system

The space state model of induction motor can be written as:

$$\begin{cases} \dot{x} = f(x,u) + w \\ y = Cx + v \end{cases} \qquad (1)$$

Where,

$$x = [i_{s\alpha}, i_{s\beta}, \varphi_{r\alpha}, \varphi_{r\beta}]$$
$$U = [V_{s\alpha}, V_{s\beta}]^T, \ y = [i_{s\alpha}, i_{s\beta}]^T$$

$$f(x,u) = \begin{bmatrix} (1-T_s\gamma)i_{s\alpha} + T_s \frac{MR_r}{L_r{}^2 k}\varphi_{r\alpha} + T_s \frac{M\omega_r}{L_r k}\varphi_{r\beta} + T_s \frac{1}{k}V_{s\alpha} \\ (1-T_s\gamma)i_{s\beta} - T_s \frac{M\omega_r}{L_r k}\varphi_{r\alpha} + T_s \frac{MR_r}{L_r{}^2 k}\varphi_{r\beta} + T_s \frac{1}{k}V_{s\beta} \\ T_s \frac{M}{T_r} i_{s\alpha} + (1-\frac{T_s}{T_r})\varphi_{r\alpha} - T_s\omega_r\varphi_{r\beta} \\ T_s \frac{M}{T_r} i_{s\beta} + T_s\omega_r\varphi_{r\alpha} + (1-\frac{T_s}{T_r})\varphi_{r\beta} \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}, k = \frac{M}{\sigma L_s L_r}, \gamma = \frac{R_s}{\sigma L_s} + \frac{R_r M^2}{L_r{}^2 \sigma L_s}$$

W and V in Equation (1) are the model and measurement disturbances which are statically described by the zero-mean Gaussian noises and characterized by covariance matrices Q and R as it will be illustrated in more detail through subsequent part B.

The Direct Torque Control (DTC) strategy, in essence, utilizes a specific switching pattern introduced by Takashi and Noguchi [3] with the aim of achieving an effective exploitation of the full capacity of induction machines of producing electromagnetic torque and flux. Unlike the vector control systems, the DTC is the simplest in structure as it eliminates the need of using external current loop, while it attains the same performance as provided by vector controllers [3]. Therefore, the advantages accrued to this control strategy as mentioned above have still maintained it in the top spot of interests for industry sector. The DTC control strategy, however, relies heavily upon the precise measurement or estimation of electromagnetic flux and torque as these two variables determine the switching pattern to be followed by VSI inverter. The importance of the electric flux and torque estimation reveals the necessity of using an identification algorithm to achieve the goal of DTC control strategy. The estimation or identification method employs the Extended Kalman Filter (EKF) in this study, so the subsequent part describes the main features of EKF.

B . Overview of EKF algorithm

The Kalman filter is a well-known recursive algorithm which takes the stochastic state space model of the system with together measured outputs to achieve the optimal state estimation of the system under consideration [2], [3]. The goal of Kalman filter, as it is outlined below, is to obtain the variables which are not measurable, i.e. covariance matrices Q, R and P of the system and measurement noise vectors and state vectors (x), respectively. The filter estimation ($\hat{x}$) is obtained from the predicted values of the states (x) and this is corrected recursively by means of correction factor as the product of the Kalman gain (L) and the deviation of estimate and the actual output vector as (y-C$\hat{x}$).

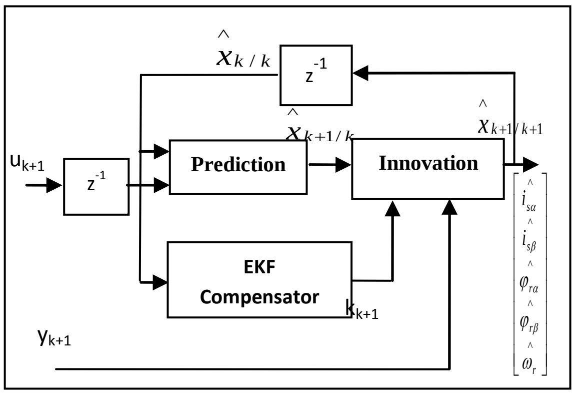


Figure 1, Systolic of the EKF algorithm

The system and measurement noise covariance denoted as Q and R, are $[5 \times 5]$ and $[2 \times 2]$ diagonal matrices, respectively.

As previously stated, the state vector to be estimated is $\hat{x} = [i_{s\alpha}, i_{s\beta}, \varphi_{r\alpha}, \varphi_{r\beta}, \omega_r]$ and the estimation procedure of Kalman filter is performed over following steps as illustrated in block diagram form in figure 1.

1) **Prediction of the state vector:** The

prediction of state vector at sampling time of (k+1) from the input vector U(k) and state vector at previous sampling time (k) is given as:

$$\hat{x}(k+1/k) = f(\hat{x}(k/k), U(k)) \qquad (2)$$

2) **Prediction covariance computation:**

The prediction covariance matrix is updated by:

$$p(k+1/k) = F(k)p(k/k)F(k)^T + Q \qquad (3)$$

Where, Q is the covariance matrix of the system noise with F given as:

$$F(k) = \left.\frac{\partial F}{\partial X}\right|_{x(k)=\hat{x}(k/k)} \qquad (4)$$

3) **Kalman gain computation:**

The Kalman filter gain or correction matrix is computed as:

$$L(k+1) = p(k+1/k)C(k)^T (C(k)P(k+1)C(k)^T + R)^{-1} \qquad (5)$$

The prediction covariance matrix is then updated in terms of Kalman filter gain as:

$$p(k+1/k+1) = p(k+1/k) - L(k+1)C(k)p(k+1/k) \qquad (6)$$

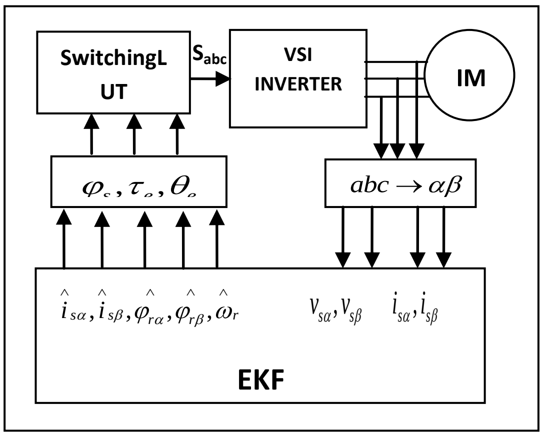
4) State vector estimation or innovation step: The predicted state-vector

$\hat{x}$ (k+1/k) is added to the term referred to as innovation multiplied by Kalman gain to provide the overall state-estimation vector $\hat{x}$ (k+1/k+1) as:

$$\hat{x}(k+1/k+1) = \hat{x}(k+1/k) + L(k+1)(y(k+1) - C\hat{x}(k+1/k)) \qquad (7)$$

III. Algorithm refinement and fixed-point simulation

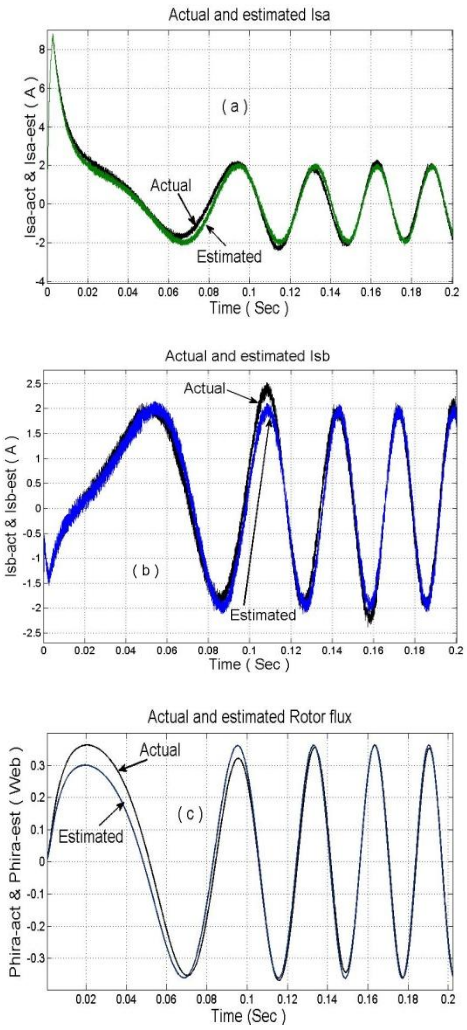
Prior to dealing with implementation of the FPGA-based EKF algorithm the complete DTC controller was first simulated in order to evaluating the functionality of the whole control system.



SwitchingLUT
S_{abc}
VSI INVERTER
IM
$\varphi_e, \tau_e, \theta_e$
$abc \rightarrow \alpha\beta$
$\hat{i}_{s\alpha}, \hat{i}_{s\beta}, \hat{\varphi}_{r\alpha}, \hat{\varphi}_{r\beta}, \hat{\omega}_r$
$v_{s\alpha}, v_{s\beta}$
$i_{s\alpha}, i_{s\beta}$
EKF

Figure 2, Block diagram of DTC control system

The block diagram of DTC controller, as shown in figure 2, is simulated using MATLAB SIMULINK environment thereby the performance of simulated EKF block on estimating the state variables vector $\hat{x} = [i_{s\alpha}, i_{s\beta}, \varphi_{r\alpha}, \varphi_{r\beta}, \omega_r]$ was examined.



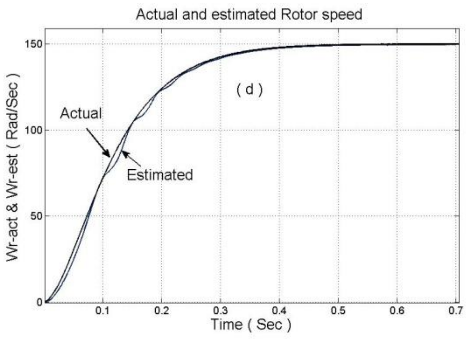


Figure 3, Actual and estimated state space variables: a) Isa, b) Isb, c) Phira, d) Wr

Figure 3 summarizes the comparison between actual and estimated state space vector obtained from simulated Induction Motor and EKF blocks, respectively. By examining figures 3 (a) through (d), it can be seen that the proper convergence between actual and estimated variables was obtained. Moreover, a refinement of the EKF algorithm, to be implemented, was achieved so as the appropriate sampling period of 30 μS satisfying the control system stability, as well as an optimal format of 25-bit fixed-point data is determined.

IV. Development of the FPGA-based EKF algorithm

The EKF as described in section III (B) is an algorithm requiring a huge number of matrix calculations such as multiplication, division and addition. Due to such inevitably intensive matrix treatments, the FPGA architecture needs a special care regarding the optimization of consumed hardware resources without admitting any degradation of timing performances. The FPGA architecture of EKF compensator module is presented in figure 4 where the computation of covariance matrix P(k+1/k+1) and the optimal Kalman gain L(k+1/k+1) required for innovation step is illustrated. The whole matrix multiplication, as shown in figure 4, was achieved using a single $[m \times n \times l]$ pipelined multiplier where m, n and l are the dimensions of two $[m \times n]$ and $[n \times l]$ matrices to be multiplied at each step. The maximum value for m, n and l was 5 corresponding to the step when two F(k) and P(k/k) matrices with dimensions of $[5 \times 5]$ were supposed to be multiplied. It should be mentioned that, due to the dependency of the each step to its precedent operation, unfortunately in case of using one single multiplier, there was no possibility to achieve the concurrent operations. Therefore, the whole EKF algorithm has been fulfilled in a sequential procedure as summarized in section II (B).

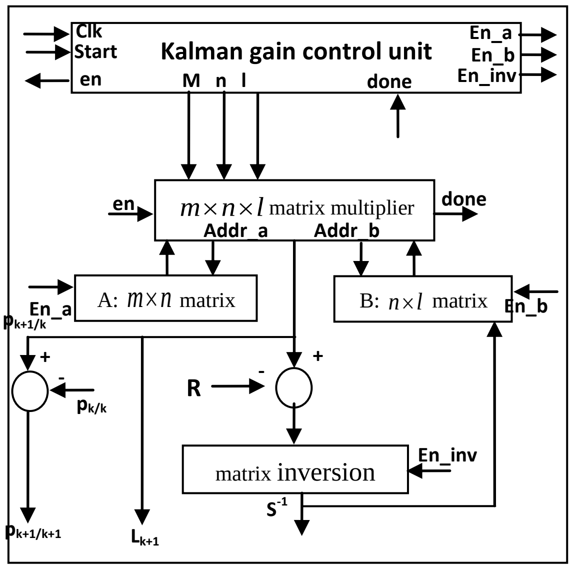


Figure 4, FPGA- based architecture of the EKF compensator

A. Hardware-in-loop (HIL) tests: Figure 5 describes the hardware in loop setup for FPGA-based EKF estimator. The designed and developed architecture was implemented in Xilinx FPGA VIRTEX-5 ML506 Evaluation Platform device using the RTL Precision Synthesis Tool and Xilinx ISE 10.1 [9].

Figure 5, Hardware in Loop

In the proposed setup, the observed or probed signals were the MATLAB output stator voltage and currents, and the output or control signals are the estimated state variables $\hat{x} = [i_{s\alpha}, i_{s\beta}, \varphi_{r\alpha}, \varphi_{r\beta}, \omega_r]$. Figure 6 summarizes the comparisons between the results obtained from FPGA and those from MATLAB simulation model as described in previous sections. By examining the results shown in figure 6 (a) through (d), it is easy to notice that the results obtained from MATLAB SIMULINK and FPGA are in proper convergence as both have used the same fixed-point data format of 25-bit. The maximum error between the corresponding results was less than 2% which was due to the truncation occurred on multiplier operand being treated by 25×18 embedded pipelined multiplier of ML506 Evaluation Platform [9].

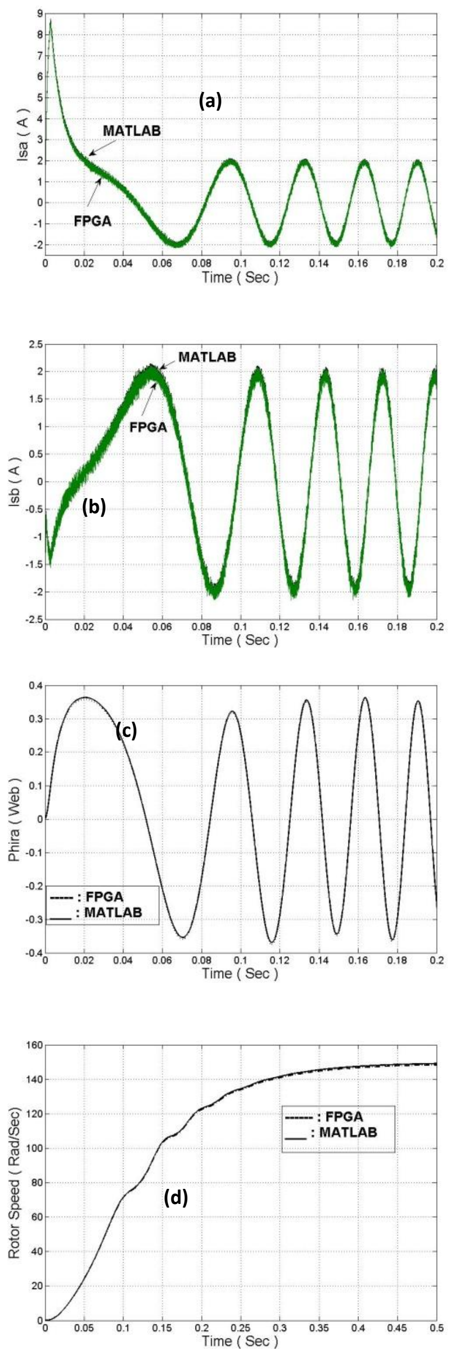


Figure 6, HIL results, a) Isa, b) Isb, c) Phira, d) Wr

B. Timing diagram of the DTC controller and

time/area performances: The complete DTC controller was simulated using MATLAB SIMULINK environment with the sampling period T_s of 30µS. At each sampling moment the MATLAB SIMULINK model updated the electrical quantities of simulated Induction Motor so as the FPGA board started performing the Kalman gain computation process using the latest data provided by MATALB. The EKF architecture was synchronized using 32 MHz clock signal so as the whole EKF algorithm was performed in 18.61 µS. After performing each Kalman gain calculation procedure, the FPGA stopped its operation and waited for the next incoming start signal produced by MATLAB.

Figure 7 describes the EKF timing diagram where the execution time for each step of EKF is shown. The prediction, Kalman gain calculation and innovation times are 1.1 µS, 16.99 µS and 1.61 µS, respectively. The whole EKF execution time is 62% the sampling period of 30 µS.

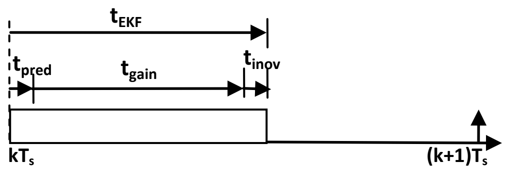


Figure 7, EKF timing diagram

It is worthwhile to mention that the whole EKF algorithm has used only 10% of FPGA resources. Table 1 summarizes the total resources usage of FPGA by EKF in more detail.

Table 1, FPGA resources usage

	Available	Used	Utilization
slice registers	32640	41	1%
Slice LUTs	32640	195	1%
Logics	32640	195	1%
25×18 Hw multipliers	136	9	7%

V. CONCLUSION

The design and implementation of the FPGA-based Extended Kalman Filter estimator was presented in this paper. The implemented EKF algorithm as a part of DTC control system was described focusing on the FPGA architecture. Simulation and hardware in loop (HIL) results were provided in order to validate the performance and effectiveness of the developed design and the time/area analysis was then presented. The later proves that the employment of FPGA as a hardware solution improves considerably the execution time regarding with DSP-based software solutions.

By comparing the dimensions of the P and Q covariance matrices of [5×5] and corresponding total execution time of developed EKF algorithm of 18.61µS with those presented in [5] as [4×4] and 10.25µS, respectively; it is evident that the developed algorithm in this study is as effective as that one developed in [5].

As stated previously, due to the lack of laboratory facilities, the authors have just implemented the EKF estimator where the rest of the whole DTC controller was simulated in MATLAB SIMULINK environment. However, an examination of time/ area performance of FPGA architecture reveals that the whole DTC controller is implementable using one single stand alone Xilinx ML506 FPGA Platform as the total recourses used for implementing EKF algorithm is only 10% of available FPGA hardware. The rest of the resource is quite enough to implement all low speed adders or multipliers required for designing the remaining parts of complete DTC controller.

ACKNOWLEDGEMENTS

This work is granted by the Rhone Alpes Region, SEMBA projet, Cluster ISLE, France.

REFERENCES

[1] G. Bottiglieri, G. Scelba, Sensorless speed estimationin induction motor drives, IEEE. Elec. Mac. IEMDC, pp. 624-630, 2003

[2] M. Barut, O. S. Bogosyan and M. Gokasan, An EKF based reduced order estimator for the sensorless control of IMs, Control Applications. CCA Proceedings of IEEE Conference , pp. 1256-1261, 2003

[3] M. Barut, O. S. Bogosyan and M. Gokasan, EKF based sensorless direct torque control of IMs in the low speed range, IEEE Trans. Indus. Elec. ISIE, pp. 696-974, 2005.

[4] S. Belkacem, F. Naceri and R. Abdessemend, EKF-based speed sensorless direct torque control of Induction Motor drives, Asian Journal of Information Technology 6(2): 185-191, 2007

[5] L. Idkhajine, E. Monmasson, A. Maalouf, FPGA-based Sensorless controller for Synchronous Machine using an Extended Kalman Filter. Power Electronics and Applications. EPE 09. 13th European Conference on, Barcelona, 2009

[6] A. Bigdeli, M. Biglari-Abhari, Z. Salic, Y. T. Lai, A new Pipelined Systolic Array-Based Architecture for Matrix Inversion in FPGA with Kalman Filter Case Study, EURASIP Journal on Applied Signal Processing Vol. 2006

[7] I. Bahri, M-W. Naouar, E. Monmasson, I. Salma-Belkhodja, L. Charaabi, Design of an FPGA-based real-time simulator for electrical system, Power Electronics and Motion Control Conference; EPE-PEMC September 2008

[8] S. Bologani, M. Zigliotto, M. Zordan, EXtended-Range PSM Sensorless Speed Drive Based on Stochatic Filtering, IEEE Trans. On Industrial Electronics, Vol. 16, No 1, January 2001

[9] Xilinx Data Book, [Online], Available on: www.xilinx.com

Design and Implementation of Low Complexity Router for 2D Mesh Topology using FPGA

Maheswari Murali [*] **and Seetharaman Gopalakrishnan** [#]

[*] *Assistant professor, J. J. College of Engineering and Technology, Tiruchirappalli, India*
[#] *Principal, Oxford Engineering College, Tiruchirappalli, India*

Abstract

Modern platform Field Programmable Gate Arrays provide larger gate count with increased performance. This feature allows realization of System On Chip on modern FPGAs. When the number of cores increases, the communication demands between cores also increases in SoCs. Hence, Network On Chip has been proposed, to meet out the challenges between the cores. In this paper, a design of low cost, low complexity router for 2D mesh topology is proposed. The low complexity router is implemented on Altera cyclone II FPGA EP2C35F672C6 device. From the implementation results, the proposed router is operated with higher speed, lower area and lower power dissipation compared with previous designs. We have also tested 2 x 2, 3 x 3 and 4 x 4 mesh for the designed router and implemented on cyclone II FPGA. From the implementation results, the proposed one gives better performance in terms of area, speed and power.

Key words: FPGA, SOPC, SOC, and NOC

1 Introduction

In the last decade there has been an increase in the computation requirement and the number of Intellectuel property (IP) cores for embedded systems. This has fostered the development of high performance embedded platforms that can handle the computational requirements of complex algorithms. With the advancement of semiconductor technology, now-a-days embedded platforms like modern Field Programmable Gate Arrays (FPGAs) have embedded hard and soft core processors, digital signal processors, memories, peripherals, and clock management systems. FPGAs with their Embedded memory and other specialized functionality have become feasible choice to implement a System On Chip (SOC) design in Application Specific Integrated Circuits (ASIC). But for implementation in ASIC, system complexity will increase the requirements for on-chip communication. Hence, the Network On Chip (NOC) has evolved as a solution for addressing this challenge. In addition, FPGA can also take up the NOC model in order to support more complex SOC implementations [1]. For NOC, Router and other logic can be implemented using programmable logic in FPGA, and dedicated NOC elements can lead to better performance and more efficient utilization of on chip FPGA resources [2, 3]. However a certain degree of configurability is required even for embedded NOC support within FPGA. In NOC messages are being routed through routers called switches [4]. IP cores are connected to the router through Network Interface (NI). It is shown in Fig. 1. Connecting IP cores through routers has several advantages than dedicated bus based wiring. It delivers high bandwidth, low latency, and low power [5].

This paper first, describes a reconfigurable router for 2D mesh topology with modified architecture. Second, the design has also been tested for (2 X 2), (3 x 3) and (4 x 4) 2D mesh network. In Altera tool, SOPC builder generates softcore processor NIOS II (32 bit). The reconfigurable router is implemented using NIOS II processor.

The organization of the rest of the paper is as follows: In section 2, the review of previous work related to NOC is discussed. In section 3, the router architecture and design and implementation details are discussed. In section 4, the implementation results and performance analysis are discussed. Finally section 5, summarizes conclusions.

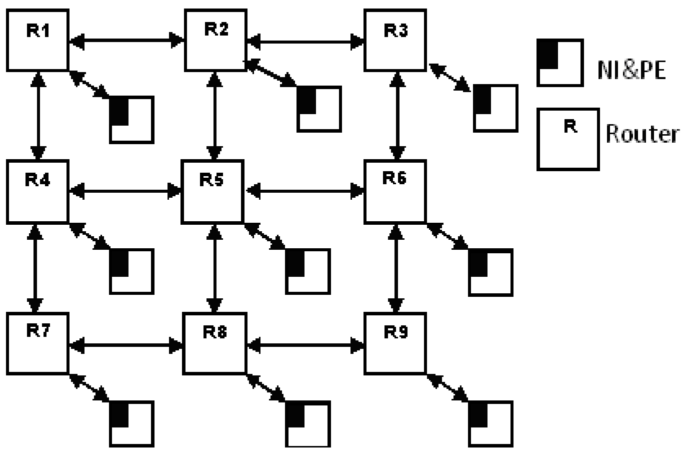


Fig. 1. 3 X 3 mesh NOC

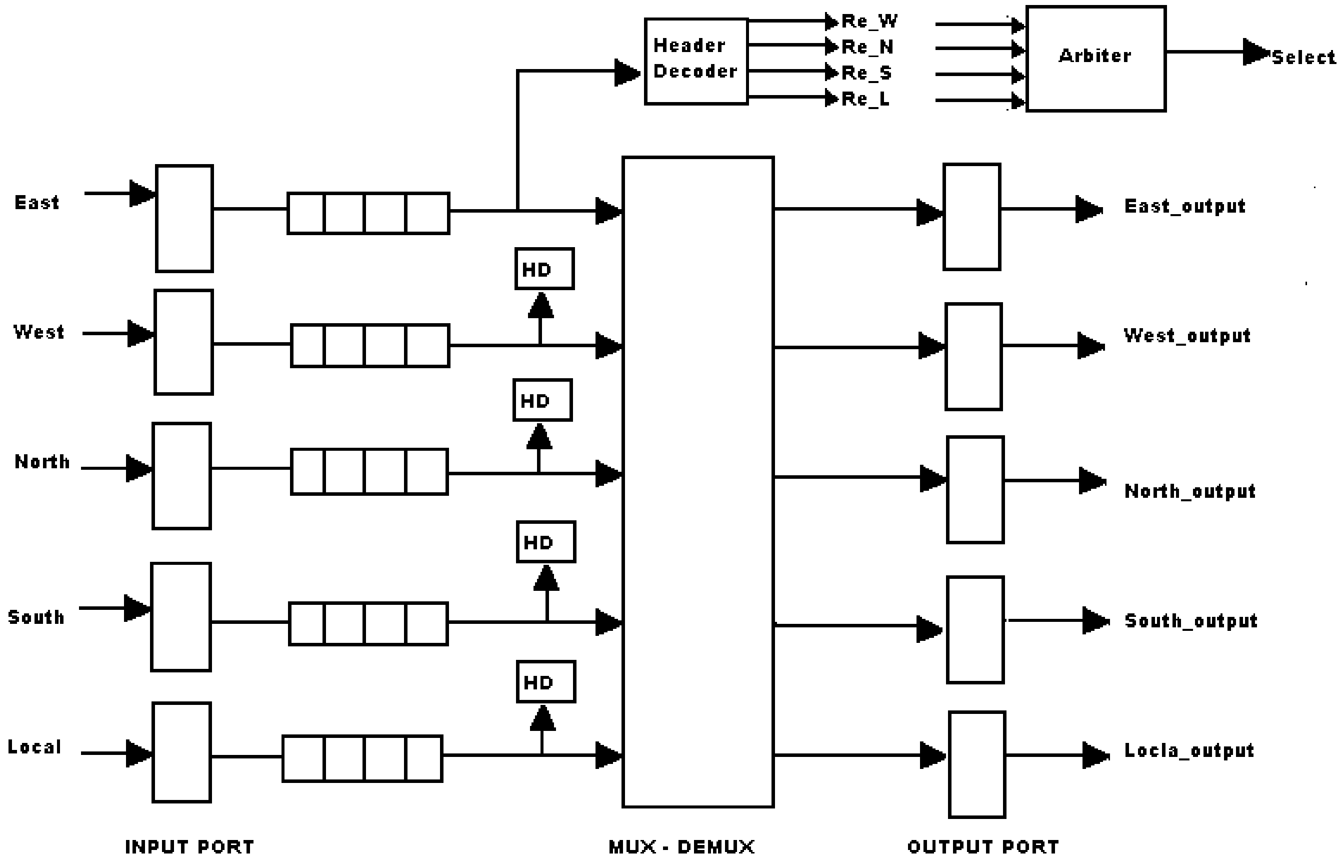


Fig.2. Router Architecture

2 Review of previous work in NOC

Literature of NOC shows that most of the work in Network On Chip have been carried out using ASIC, However a few research works have been carried out using FPGA. Important contributions are made to NOC design in [7] but implementation details are not given. Highly scalable Network On Chip for reconfigurable systems has been made in [8]. In this, the design of NOC with virtual cut through switching which has low latency, and it uses large buffer which occupies more silicon area. In [9], design of a router for NOC has been tested for two applications (FFT & Matrix Multiplication). A scalable packet switch based router for 2D mesh and torus topology has been implemented in both FPGA and ASIC in [10]. R. Gindin et al proposed a design of NOC on FPGA and used a standard mesh topology. In this the reconfigurability is limited to routing schemes only. In [11], a router is designed for 2D mesh topology and it has been synthesized on Xilinx FPGA. In this, the design contains limited routing and arbitration and no applications were tested. Also the design consists of routers that evaluated with logic simulation only.This router design [11] is similar to the proposed work in this paper with minar modifications in the FSM Controller and arbitration. All the above said works have complex design. But, in this paper FSM controller is efficiently designed and it occupies less area, which reduces the complexity of the router compared to other designs discussed above. The proposed router and the mesh topologies are implemented and verified using Altera Nios II processor which first of its kind.

3 Router Architecture

Network On Chip consists of three important components Router, Processing Element, Network Interface. The router architecture is shown in Fig. 2. Routing Node is responsible for forwarding the data packets to the destination node. Each router has associated with unique XY address. In a mesh based NOC, a router has five directional channels. They are EAST, WEST, NORTH, SOUTH and a Local channel to which design core is connected. The directional channels connect the router with its neighboring routers and the local channel is used to connect the Processing Element to the router. Processing Element is used to implement the computing functionality in DSP, Microcontroller or memory block, and input/output device controller. Network Interface is an interface between router and design core. It performs two way communications. First it collects the data from the design core, then pocketize and adds the header and it pushes the packet into the router. Second it receives the packet from the attached router and depacketizes then sends the data to the Processing Element. Fig. 2 shows the router architecture which contains input port, output port, cross bar switch, arbiter and header decoder.The function of these blocks are explained in the following subsections.

3.1 Input Port

Each input port shown in fig. 2 has a FIFO buffer header decoder, and FSM controller. The diagram for input port is shown in Fig. 3(a). FIFO buffer is used to store the data packets. Header decoder is used to decode the destination address. The FSM controller is used to control all the operations. FIFO buffer is implemented using Quartus II Mega wizard function. The size of the FIFO buffer is parameterizable and use 8 bit data width and 32 locations depth to store the packets. When the request comes from neighboring router, if the FIFO buffer is empty the input port sends the acknowledgement (ACK) signal to the neighboring router, after receiving the acknowledgment the neighboring router sends the first flit. This flit is stored in the buffer and a signal is sent to the header decoder to decode the destination address and then first flit is sent to the header decoder. Header decoder after decoding the address send a request to an arbiter of the corresponding output port. If the destination address is matched with router address, then the particular data is for that corresponding PE. So the acknowledgement is sent to the arbiter of the local output channel.

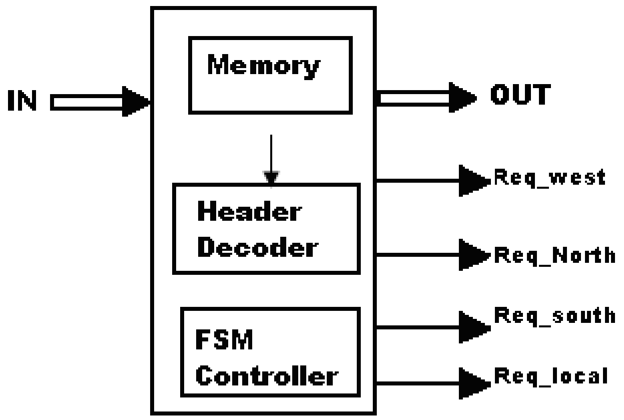


Fig. 3(a). Input Port

3.2 Output Port

The output port has an arbiter, and a FSM controller. The diagram for output port is shown in Fig. 3(b). Each output port has two locations depth buffer to store the flit temporarily untill it gets access from the down stream router. In Fig. 3(b), an arbiter is used to resolve the conflicts when more than one requests come to the same output port. If more than one requests comes to a particular output port, the arbiter gives access to only one input port based on round robin priority. The priority is rotated in round robin fashion. The input port which has been granted access to send the remaining flits, then sends the flits to the output port through the cross bar switch. After receiving the first flit, the output port sends a request to the neighboring router to which it is connected. If ACK comes from neighboring router it sends the flits to next router.

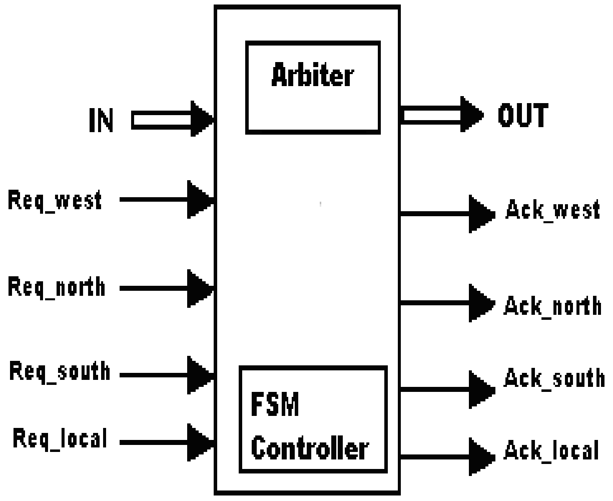


Fig. 3(b). Output Port

3.3 Cross Bar Switch

The cross bar switch is shown fig. 2. In the proposed router, simultaneously five transmissions are possible. Hence each output port has cross bar switch, and it is implemented using Multiplexer and De-multiplexer. All the input ports are connected to the multiplexer inputs and all the De-multiplexer outputs are connected to all the output ports except its own output port . Each output channel configures the multiplexers and De- multiplexer to establish the appropriate input-output connection.

3.4 Arbiter

The arbiter is shown in fig. 2. The arbiter resolves the problem if requests come from more than one input ports to the same output port and gives access to only one input port using round robin priority. This arbiter assigns priority in round robin fashion in the order east, west, north, south, and local. For example if east input port is assigned with the highest priority and then the next highest priority is assigned to west input port. The arbiter gives access to east input port first and then west input port and so on.

3.5 Header Decoder

The Header decoder in the input port is shown in fig.3(a). In this paper, XY routing algorithm is proposed. Each router has its unique coordinate as address (X,Y). When the header decoder receives the destination address (Rx, Ry), it compares local X coordinate with the destination coordinate Rx, and sends the packet through east channel if Rx > X and through west channel if Rx < X. If Rx is equal to the local X coordinate, then Y coordinate of the router is compared with destination coordinate Ry. If Ry > Y coordinate then the packet is routed through North Channel, else the packet is routed through south channel. If Ry =Y, then the packet is sent through the local channel to the Processing Element.

3.6 SOPC approach for NOC

Altera System On Programmable Chip (SOPC) Builder is used to add the design element. Using SOPC builder, the proposed router and all mesh topologies are added to the Nios II processor as the custom block. The program to be executed and verified by writing in C using Nios II IDE [12]. The custom block is invoked as the function in C code. The C program is complied and the configuration bits are downloaded to the FPGA for verification.

4 Implementation Results & Performance Analysis

The proposed router is implemented on Altera cyclone II FPGA. It has been designed by writing HDL. The buffer which is used to store the flits decides the area of the router. Hence, the proposed design uses FIFO buffer which is available in Quartus II mega wizard function. The FIFO which is used in this design is a parameterizable. The data width is 8 bit and the depth of the buffer is chosen as 32 locations for testing purpose. The flow control mechanism used is wormhole method which requires less buffer size to store the data. The proposed design uses XY routing algorithm.

In the proposed design uses random number generator to generate random numbers. First, this random number generator is connected to all the input channels and verified the data from the output channels of a single router. Second, the 2D mesh topology is implemented with the designed router and verified using random number generator for 2 X 2 configurations. The same method has been adopted to test 3 x 3, 4 x 4 configurations of 2D mesh. The designed router has been implemented on Altera cyclone II FPGA EP2C35F672C6 device [6]. We used Quartus II 8.1 to synthesis and simulate the RTL design. The area and the speed of the single router, 2 x 2 and 3 x 3 mesh configurations are compared with the design of [11]. The proposed design occupies 20% less area and 50% higher speed. The synthesis results are shown in figure. 4. The simulation result shown in the figure. 5. The RTL schematic view of the proposed router is shown in figure. 6.

4.1 Synthesis Report

The implementation of the proposed router occupies 592 LUTs which is only 2% of the total Altera EP2C35F672C6 device. The area occupied by the 2 x 2, mesh is 1374 LUTs which is only 4% of the total area 3 x 3 is 2645 which is 8% of total area, and 4 x 4 mesh is 3634 which is 12% of total area.

4.2 Power Analysis

From the implementation results, the power dissipation for single router, 2 x 2 and 3 x 3 are shown in figure 4. The power dissipation is measured using Power play power Analyzer tool available with Altera Quartus 8.1. The power dissipation for the single router design is found to be low compared to [11]. The proposed design consumes only 120mw of thermal power dissipation, 80mw of core static thermal power, and 38mw I/O thermal power dissipation.

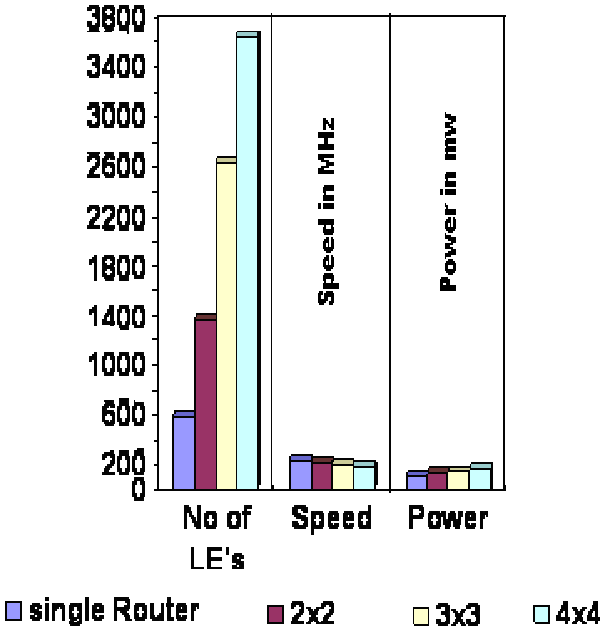


Fig. 4. Implementation Result

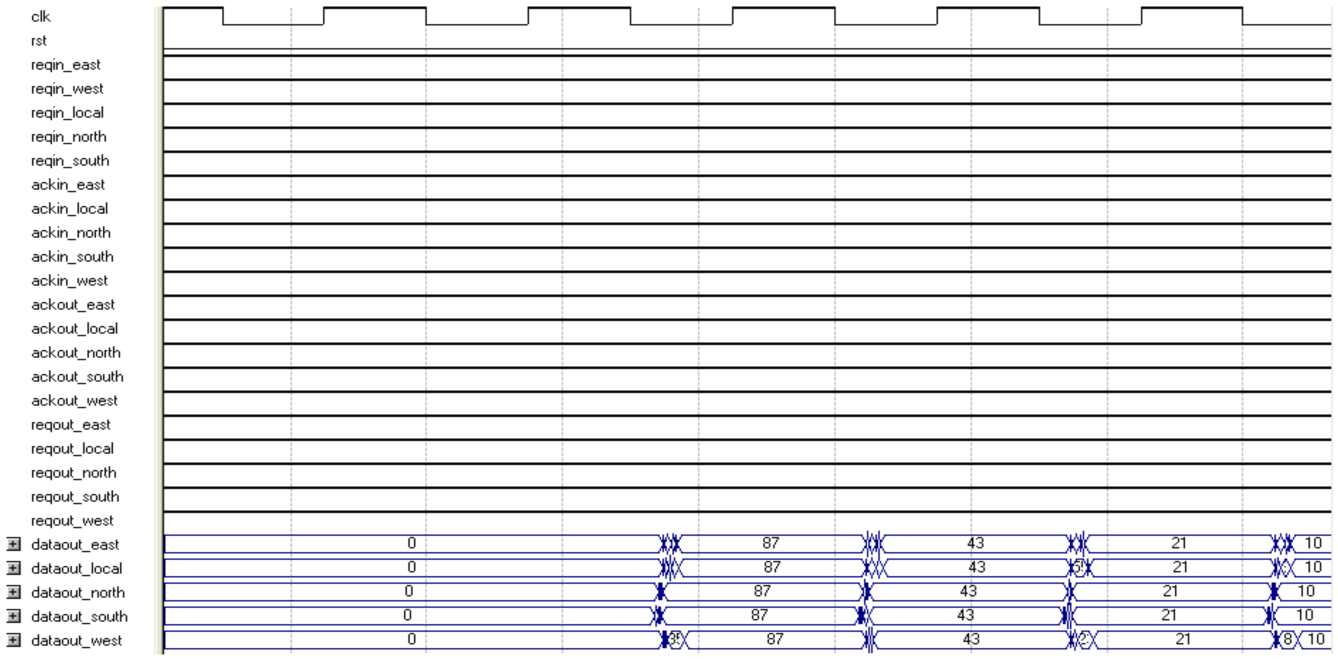


Fig. 5. Simulation result of single router

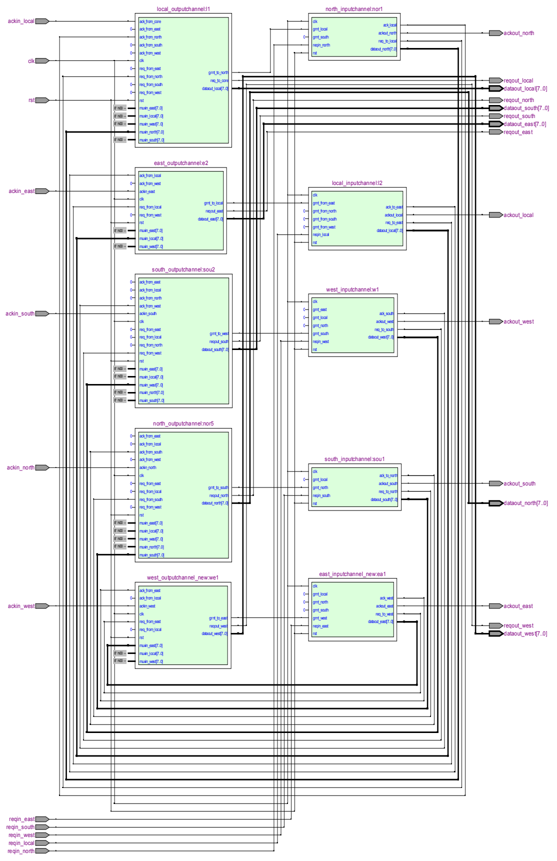


Fig. 6. RTL Schematic view of single route

5 Conclusions

The proposed router has been designed for 2D mesh topology which has low complexity and high speed. The single router and 2 x 2, 3 x 3, 4 x 4 mesh topology have been implemented and tested with SOPC using NIOS II processor. From the implementation results, it is found that the proposed router is operated with higher speed and lower area and lower power dissipation compared with the previous design. Our future work is to test the signal processing application using this proposed router.

6. References

[1]. Ronny Pau and Naraig Manjikian "High level Specification and logic implementation of Single chip Multiprocessor system based on a configurable router" IEEE, pp.1039- 1044, 2008.

[2]. H.Elmiligi et al . "Introducing OperaNP : A Reconfigurable NoC based platform" Proc.2007 IEEE Canadian Con. On Electrical and computer Engineering, pp. 940-943, 2007.

[3]. R. Gindin et al ."NoC based FPGA Architecture and routing" In Proc.

Intet. Sympo. on Network On chip, pp 253-264, 2007.

[4]. W.J.Dolly and B.Towles "Principles and Practices of Interconnect Networks". Morgan Kaufmann Publishers – 2003.

[5]. T.Bjerregaard and S. Mahadevan "A survey of Research Practices of Network On Chip" ACM Computing Survey Vol-38, no-1, pp. 1-51, 2008.

[6]. Altera Inc. http :// www.altera.com.

[7]. L.Benni and G. D Micheli "Network on Chips: A new SOC Paradigm" IEEE Computer, vol. 35 no.1 , pp. 70-78 , 2002.

[8]. A. Batric et al. "Highly Scalable Network On Chip for reconfigurable systems" In proceedings of the International Conference on Systems On Chip 2003, pp. 79-82, NOV. 2003.

[9]. Gaoming Du, Duolo Zhang et al "Scalability Study on mesh based Network On Chip" IEEE PAWCIIA-2008, pp. 681-685, 2008.

[10]. Yahia SALAH, Mohamed ATRI, Rached TOURKI "Design of a 2D Mesh-Torus Router for Network On Chip" IEEE International Symposium on Signal Processing and Information Technology, pp. 626 – 631, 2007.

[11]. Balasubramanian Sethuraman , et al "LiPaR: A Light Weight Parallel Router for FPGA based Network On Chip" GLSVLSI, pp. 452 – 457, 2005.

[12]. G.Seetharaman and B. Venkatramani "SOC implementation of wave pipelined circuits" Proc. of International Conference on Field Programmable Technology, (ICFPT'2007) pp.9-16, Japan, 2007.

SESSION

EMBEDDED SYSTEMS + MICROCONTROLLERS + SMART CARDS + SYSTEM ON CHIP + SENSORS

Chair(s)

Prof. Hamid R. Arabnia

SCADA IMPLEMENTATION USING GSM NETWORK FOR COMMUNICATION

A. Dr. Riaz Ul Hasnain Syed[1], B. Engr. Haider Zaman[2], and C. Engr. M. Hanif[3]
Electronic Engineering Department, University of Engineering and Technology Peshawar, Abbottabad Campus, Pakistan.

Abstract *- Remote and centralized control is the basic need of todays industrial control systems, where a separate microcomputer monitors and controls the process at each remote unit. Collecting and displaying data from remote sites on a single monitoring unit let the operator to supervise all the sites, regardless of the volume of industry.*

Different aspects of Supervisory Control And Data Aquisition (SCADA) system is a vast field for the researchers. This paper presents an implementation of a SCADA system, and particulary focused on the communication scheme between Remote Terminal Unit (RTU) and the Master Terminal Unit (MTU). The developed system consist of a single Remote Unit that is a microcontroller based system with temprature sensor, relays and a DC motor interfaces. A windows application is developed working as the master unit that communicate with the RTU through short messaging services(SMS).The RTU collects data through its interfaces and transmit for MTU using the Global System for Mobile communication (GSM) network.

The proposed system is simulated using Proteus and the positive, accurate and timely response approved the system for implementation.

Keywords: Supervisory Control and Data Aquisition, Remote Terminal Unit (RTU), Master Terminal Unit (MTU), Microcontroller.

1 Introduction

Distributed Control System (DCS) is employed in case of large industrial setup, where the whole system is divided into sections each unit performing a particular task out of the whole process, that is each working as an embedded system. The units communicate to work in collaboration with each other and a Master Station to complete the process cycle.

Many manufacturing companies have developed products to meet the DCS requirments, such that they could control a process and communicate with the other units whenever required.

Varoius researches have been presented regarding control and communication technology in a SCADA system. In communication case the researches are focused on the responsiveness, low cost, security and efficiency of the communication system [1].

Keeping these aspects in mind the developed system utalizes the GSM network. GSM is a well tested and implemented network with a large span advantages including all the above mentioned.

A SCADA system consists of the following components an MTU, RTUs and communication system between MTU and RTUs also between two RTUs.

The block diagram of the system is as shown in Fig.1. In this implementation the complete SCADA system is developed. A microcontroller based RTU and a software package collecting data serving as MTU while the wirless communication is implemented with the help of GSM network.

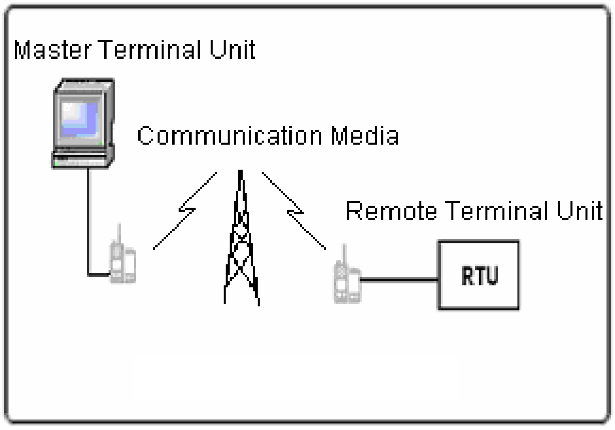


Fig. 1: SCADA System

2 Remote Terminal Unit

The Remote Terminal Unit is usually defined as a communication satellite within the SCADA system and is located at the remote site. The RTU gathers data from field devices stores in memory until the MTU initiates a send command [2].
A microcontroller based system is serving as RTU in this implemention. Although there could be more than one RTU depending on the senerio, however the experiment has been carried out with a single RTU.

The microchip PIC18F452 has been used, that is an 8-bit microcontroller, with 32 K flash, 32 digital I/0 , and 8 analog input channels[3]. Project development with PIC controller is extremly easy. Especialy the programming tools available for these controller are more user friendly.
Fig.2 shows the RTU interfaces. The temperature sensor reads the temperature of the unit and displays on a LCD display. The LM35 has been chosen as it is a low cost and voltage output, semiconductor temprature sensor, that can read temparture from -50 to 150 degree celsius. The output voltage varies for 0 - 2.5 V with a small amount of noise. A non inverting amplifier has been used that redefine the output voltages as 0-5V.

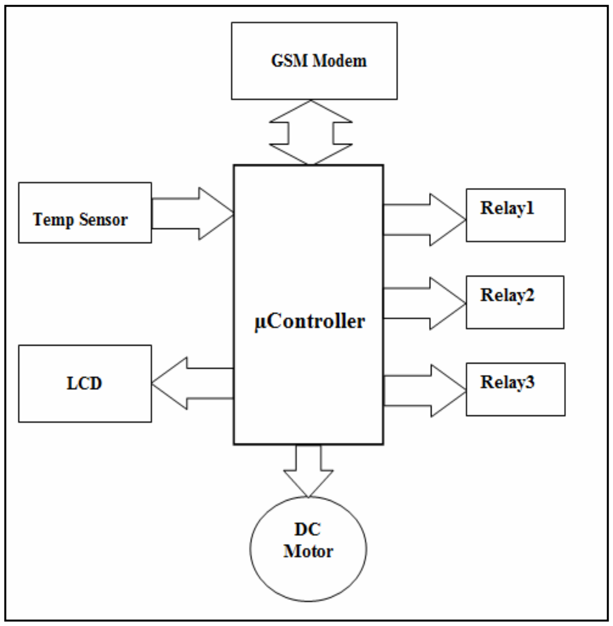


Fig.2: Remote Terminal Unit

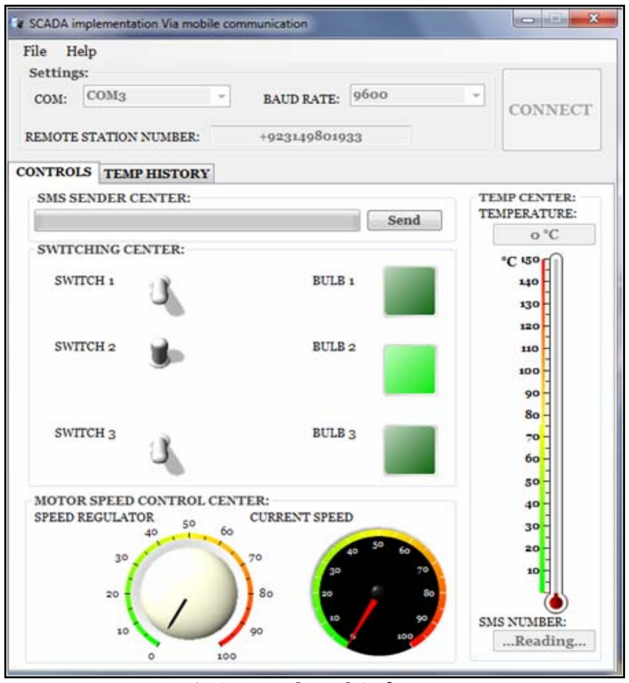
Three Relays are interfaced with microcontroller to control three different ac devices. 100 W bulbs are used for the experimental purpose. Since microcontroller cannot source enough current to excite the windings of a relay so relay driver (ULN2803) are employed to meet the current and voltage rating of relays.

The DC motor has been interfaced such that not only its speed but direction can be changed. Again the problem of current sourcing capability of microcontroller, L293D has been chosen that is an H-Bridge drive with built-in protection diodes.

3 Master Terminal Unit

The master unit is the central control, and is the core part of a SCADA system, that provides interface for operator. The MTU collects data from remote fields, process that data and sends instructions and commands accordingly.

The developed application is programmed such that it sends a data request message after every 3 seconds. In response the RTU verifies the number acting as address of MTU and reply with temperature, duty cycle of the Pulse Width Modulated signal and status of the all the three relays.
The controls provided in the Graphical User Interface (GUI) are according to the RTU interfaces. The RTU's temperature sensor data received is displayed on thermometer scale, the DC motor speed on gauge and state of relays through a bool data-type LED (Light Emitting Diode). Two different colors are assigned to both states of relays. The GUI is refreshed in every 3 seconds by receiving SMS from MTU. The two-way switches can change state of the relays at RTU. GUI is shown in Fig. 3. The developed software can store the temperature history and a graph is of temperature verses time is provided in tab. Temperature history graph is shown in Fig. 4.


SCADA implementation Via mobile communication

Settings:
COM: COM3
BAUD RATE: 9600
REMOTE STATION NUMBER: +923149801933

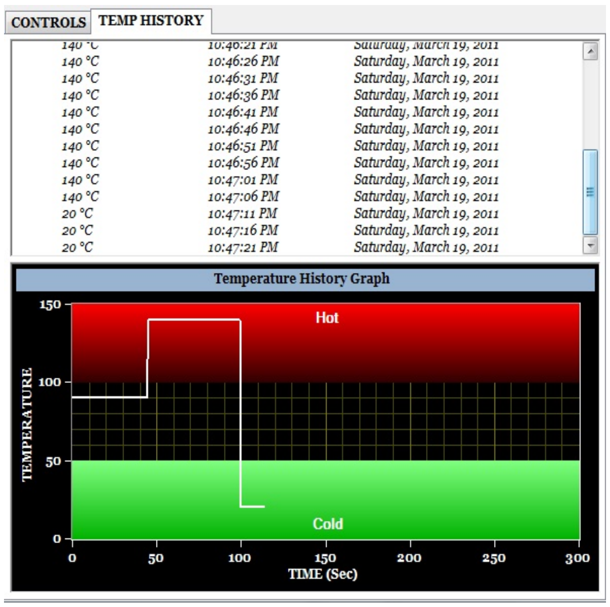
TEMP CENTER: TEMPERATURE: 0 °C

SMS NUMBER: ...Reading...

Fig.3: Developed Software

Operating the software package is very easy all Operater has to do is to enter the destination mobile number, COM port, baud rate (supported by the mobile set being used) for communication.
The interface can also be developed with lab view VIs. The C# is preffered because it results into a standalone package installable on windows system. However the controls of the measurement studio are better tools for designing control interfaces e.g the thermometer, the gauge, LED and switches are the part of measurement studio library.

"TRACE MODE" and "Winlog SCADA" are the software packages used to develop SCADA systems. The communication protocols of PLCs from various manufacturers are defined in these tools.



140 °C	10:46:21 PM	Saturday, March 19, 2011
140 °C	10:46:26 PM	Saturday, March 19, 2011
140 °C	10:46:31 PM	Saturday, March 19, 2011
140 °C	10:46:36 PM	Saturday, March 19, 2011
140 °C	10:46:41 PM	Saturday, March 19, 2011
140 °C	10:46:46 PM	Saturday, March 19, 2011
140 °C	10:46:51 PM	Saturday, March 19, 2011
140 °C	10:46:56 PM	Saturday, March 19, 2011
140 °C	10:47:01 PM	Saturday, March 19, 2011
140 °C	10:47:06 PM	Saturday, March 19, 2011
20 °C	10:47:11 PM	Saturday, March 19, 2011
20 °C	10:47:16 PM	Saturday, March 19, 2011
20 °C	10:47:21 PM	Saturday, March 19, 2011

Fig. 4: Temperature History Graph

4 Communication

Communication infrastructure connects the Remote Terminal Units to supervisory system. Data is transmitted between master unit and remote unit through communication unit. Communication can be a through a wired or wireless media depending on the distance separating the Units. Wired media is preffered in case of small distance while wirless is preffered in long distance case. The main concerns of SCADA networks are its functionality and security.

In this implementation GSM network is employed that is a wireless communication technology; most popular today for transmitting data anywhere in the world. The GSM performance, reliability, flexibility are robust. GSM is more responsive as it provides authentication of messages delivering. GSM services are very economic and are available universally [2].

The GSM modules has been interfaced with RTU and MTU working as transceivers in this implementation. Both units communicate with the GSM modules serially through AT commands sets. In this message-based polling system a unique message format has been implemented, conveying RTU's interfaces status.

Many SCADA systems these days employ internet for communication. However internet represent a significant vulnerability to cyber attacks to the SCADA network. If any location that has a connection to the SCADA network can be a target, especially unguarded remote sites [4]. If Cyber system is not protected at each layer against other systems so the system can be hacked. So, for cyber security additional measures have to be taken for providing security. It is good security engineering practice to avoid connecting SCADA systems to the Internet so the attack surface is reduced. Internet system is at risk of shutting down, if any fault is introduced in the optical fiber e.g if cable is damaged, it is a time consuming and cumbersome job to repair or replace.

GSM infrastructure for communication between MTU and RTU has been chosen as it is more responsive, low cost, secure system and efficient.

In this SCADA implementation unique, proprietary protocols for communications between field devices and master terminal unit has been used.The security of SCADA systems is based on the secrecy of these protocols, which we have defined our own protocols for both transmission and reception**.**

5 Conclusions

The implemented system can be extended for more than one RTU. It can control devices at much higher level in industries. Along with communication between RTU and MTU, inter-RTU communication can also be established. Security of the SCADA system can be ensured for communication between RTU and MTU. GSM is a communication mechanism for concise information and with ability to screen messages. SMS benefits including the delivery of notifications and alerts, guaranteed message delivery, reliable and low-cost. Moreover, as far as the messages are concerned they are in specific codes so cannot be analyzed by intruders.

6 References

[1] Allen Bradley "SCADA System Application guide" Rockwell automation, Publication AG-UM008C-EN-P-February 2005.

[2] Architecture for secure scada and distributed control system networks, comprehensive network-based security for control systems", 2010, juniper networks, inc.

[3] Mohammad Ali Mazidi " The 8051 Microcontroller and Embedded Systems" *Prentice Hal*l, 2nd Edition, 2006.

[4] "21 steps to improve cyber security of SCADA networks", The President's Critical Infrastructure Protection Board, Office of Energy Assurance, U.S. Department of Energy 202/287-1808.

Design of an HVAC Zone Control System

Bassam Shaer and Dana M. Wadsworth
Electrical and Computer Engineering Department, University of West Florida
Shalimar, Florida, United States

Abstract - *This paper deals with the concept of temperature control inside an environmentally controlled structure. In most cases, the temperature inside a home or business is controlled by a centrally located thermostat. The thermostat senses the temperature at the central location and controls the Heating, Ventilating, and Air Conditioning (HVAC) unit based on the centralized temperature. A feasible design has been developed for providing independent temperature for each zone instead of the centrally located thermostat. The design incorporates a temperature sensor circuit and a transmitter for multiple zones. The transmitted data is received and transferred to a microcontroller that invokes an algorithm that controls the HVAC system based on the inputs from the temperature sensors as well as the system thermostat. The microcontroller also provides control for motorized vent registers to vary the airflow in each zone based on the temperature sensor data and provides LCD readout for centralized monitoring.*

Keywords: temperature control, multiple zones, HVAC system, microcontroller

1 Introduction

This paper deals with the concept of temperature control inside an environmentally controlled structure. In most cases, the temperature inside a home or business is controlled by a centrally located thermostat. The thermostat senses the temperature at the central location and controls the Heating, Ventilating, and Air Conditioning (HVAC) unit based on the centralized temperature. This project designed and produced a control system that remotely senses temperatures in multiple areas and uses a microcontroller to maintain a consistent temperature throughout the interior of the structure. This is done by installing remote temperature sensors in different areas of the structure that report temperature back to a central controller and in turn control the HVAC on and off state and close motorized vent registers in the area of interest when the temperature has reached the desired value

2 Problem Definition

All environmentally controlled structures, whether residential or commercial, require some type of control to enable the HVAC unit to maintain a constant temperature throughout the interior of the structure. In most cases this is accomplished by using a centrally located thermostat that senses the temperature and turns the HVAC on and off based on the desired temperature provided by the user. Many variables such as uneven air flow, room size, window placement, window size, and time of day can cause the temperatures of areas within the structure to vary greatly from the central thermostat. Therefore, there was a need to develop a system that will sense temperatures in multiple locations throughout the structure independently and control the HVAC run time accordingly.

This product is intended for use in the situation where temperature variations by area are a concern within an environmentally controlled structure. Both commercial and residential applications would benefit from this design. Users would include people or companies that want all areas within a structure, not just a centrally located area, to maintain a constant temperature.

The objectives of this design are as follows:

The temperature sensing circuit shall:

- Use the LM34 Precision Fahrenheit temperature sensor in a remote sensing circuit designed to detect temperature and produce a DC voltage that is directly proportional to the ambient room temperature.

The Analog to Digital Converter circuit shall:

- Convert the DC Fahrenheit voltage from the LM34 temperature sensing circuit to a digital data stream that can be transferred via wireless connection from the remote sensing circuit to the centralized system controller.

The wireless communications circuit shall:

- Use the XBEE RF transceiver pair operating at 2.4 GHz to transfer digital data via wireless from the remote sensor circuit to the centralized system controller.

The Microcontroller circuit shall:

- Use the Atmel ATmega168 series microcontroller to develop a control system that will receive digital temperature data from multiple remote sensing circuits as well as a centralized thermostat and implement a control algorithm that provides open/close control to motorized HVAC vent registers in multiple areas as well as HVAC on/off control.

The final product is a remote monitoring system that maintains a specified temperature throughout an environmentally controlled structure. This is done with the use of remote temperature sensing circuits and an ATmega microcontroller algorithm to turn on and off the HVAC and motorized HVAC vent registers. The monitoring system detects ambient room temperature in various locations, transfers this temperature data via wireless communications to a centralized control circuit that will use the control algorithm to control the HVAC and operate motorized vent registers in order maintain the specified temperature in multiple areas of a residential or commercial building

3 Details of the System

3.1 Temperature Sensor Circuit

Figure 1 shows how the entire system is implemented. The system consists of at least two subsystems; the temperature sensor subsystem and the HVAC control subsystem. Multiple temperature sensor subsystems will be dependent on the number of areas to be monitored and controlled. The block diagram in Figure 1 shows two subsystems isolated from each other. The two subsystems are shown connected by an RF network. Figures 2 and 3 show each subsystem in greater detail

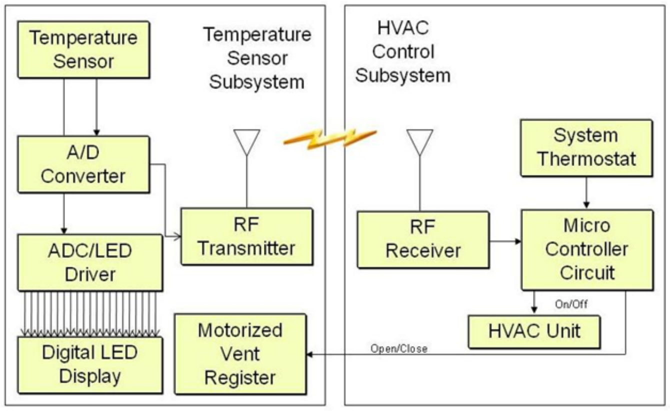


Figure 1. Complete System Block Diagram

The function of the temperature sensing circuit is to sense the ambient room temperature and provide it to the control system and LED display sub circuit. The first component of the sensor circuit is the LM 34. The LM 34 is a precision integrated-circuit temperature sensor, whose output voltage is linearly proportional to the Fahrenheit temperature. The advantage of using the LM 34 over other temperature sensors is the linear to Fahrenheit output voltage. The LM 34 requires no calibration and is accurate at room temperatures to 0.5 degrees. The TO-92 package was chosen to get the sensor away from the other components and as far into the ambient room air as possible. The sensor has a +5V supply voltage and the output is a DC voltage at 10mV/degree F. The typical room temperature of 78 degrees will be processed as 780 mV. The output of the LM 34 provides data to the input of two devices. The first is an ADC 8084 8 bit parallel output Analog to Digital Converter (ADC). The ADC 8084 will convert the output of the LM 34, which is in the mV scale to a digital signal that will be placed on the parallel output pins D0-D7. The negative reference voltage will be ground and the positive will be set by a reference voltage sub circuit. The ADC is powered by a single +5V supply. The reference circuit consists of a LM336 2.5V shunt regulator diode, a 10K potentiometer and an LM358 operational amplifier. With a +5V supply the shunt diode maintains a stable 2.5V reference that is used to adjust the input to the op amp. The op amp is used as a buffer and provides a 1.28V input to the Vref/2 input of the ADC. The 1.28V input sets the ADC reference voltage to 2.56V which provides a digital resolution of 1 bit per degree F.

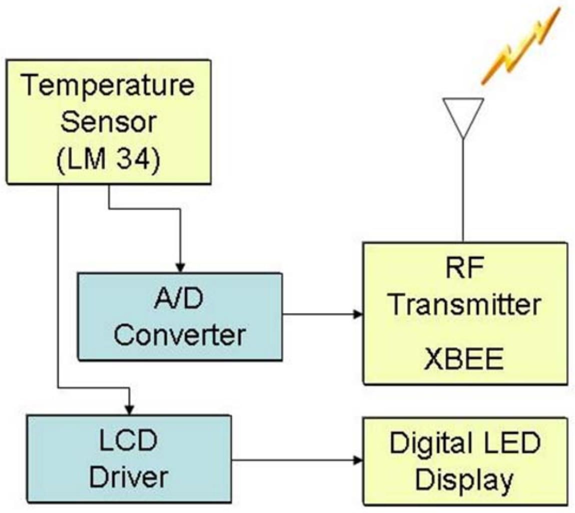


Figure 2. Temperature Sensor Subsystem

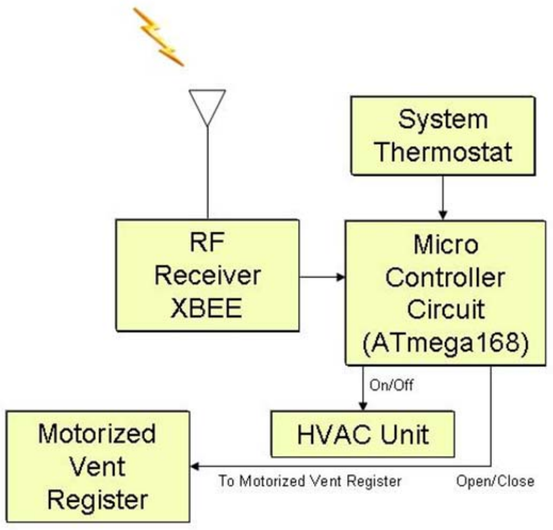


Figure 3. HVAC Control Subsystem

The ADC is operated in a free running/self-clocking mode. No additional circuitry design was required. The data sheet for the ADC0804 provides the wiring and components required for this mode.

The output of the LM 34 will also drive a TC7107A ADC with 7 segment display driver output. The TC7107A eliminates the need for any binary to BCD conversion and BCD to display drivers. The output of the TC7107A will drive 3 - 7 segment LED readouts to provide a temperature monitor at the point of temperature sensing. This will provide the user with information to make decisions as to where the sensor will be located. The actual readout will be in the mV range with the decimal point after the second digit wired for a constant "on" indicating a temperature in degrees and decimal degrees. The TC7107A typically requires a -5V and +5V supply but is capable of operating in certain circumstances with a single +5V supply. The additional circuitry required for an additional reference voltage was not cost effective and using a charge pump DC-DC Voltage Converter was found to be the most efficient. The TC7660 DC-DC Voltage Converter was used to provide the -5V supply.

The TC7107A has a built-in ADC that is driven by the same analog signal driving the ADC0804. The TC7107A requires a 1V reference voltage between the Vref and –Vref. The –Vref is grounded and a voltage reference circuit provides the 1V reference to the Vref pin. Three clocking methods can be used for the TC7107A, an external oscillator connected to pin 40, a crystal between pins 39 and 40 or a self-clocking circuit using an RC circuit between pins 38, 39, and 40. Like the ADC0804, the TC7107A is operated in the self-clocking mode.

The parallel data output of the ADC0804 along with additional RT address bits are loaded into bits of three HEF4021B, 8 Bit shift registers. The 8 bit temperature data along with a 4 bit RT address is embedded into a 24 bit digital data stream. This is done by parallel loading the temperature data into the asynchronous parallel data inputs. The start and stop bits for the temperature data are hard wired to either +5V or ground as is the RT address bits. The serial data output from the third register is wired to the serial data input of the second shift register and the second to the first. The serial data out of the first register will be the 24 bit UART serial data stream.

This data stream is shifted out of the three registers and transferred to the RF transmit circuit using the UART protocol. UART protocol requires a high idle state so the initial bit in the 24 bit stream will be high. The next bit is the "low" start bit for the RT address followed by the RT address and a stop bit. After a 3 bit idle state between the two 8 bit words, the temperature data preceded by a start bit and followed by a high stop bit follows the RT Address word. Figure 4 shows a diagram for the 24 bit data stream.

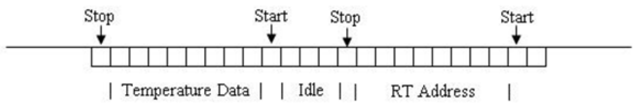


Figure 4. UART Digital Data Stream Diagram

The shift registers' timing is controlled by a NE556N general purpose dual bipolar timer IC. The timer IC controls both the parallel load/Shift functions of the shift register but also the frequency at which the data is shifted.

The first timer controls the load/shift of the registers. The HEF4021B is designed to load data from the asynchronous parallel inputs when the PL pin is high. When PL is low the data is shifted out. A single timer is used that toggles between high and low state alternately loading the register and then shifting the data. The second timer is used to set the clock at which the data is shifted out of the registers. This effectively sets the bit rate or BAUD rate of the digital bit stream. This rate is critical to interfacing to the MCU controlling the HVAC.

Both timers are operated in the astable operation. The load/shift timer components were calculated for various clock rates from 10 seconds to 15 seconds. Each temperature sensor circuit has a unique shift cycle which creates a pseudo TDMA network with each sensor transmitting data at different intervals. . Figure 5 shows the typical diagram used for both timer circuits operating in the astable mode.

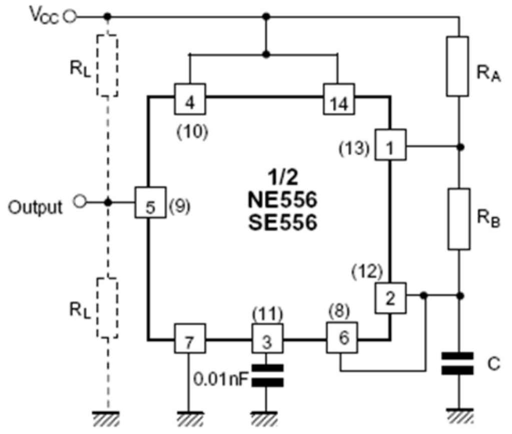


Figure 5. Astable Timer

The clock timer was calculated for 4800 BAUD. A fixed resistance value is used for RA and a potentiometer that gives adequate swing on both sides of the calculated value for RB was used to allow adjustment of the BAUD rate. The BAUD rate was critical due to the interface with the transmission medium. An oscilloscope and an initial test pattern were used to set the BAUD rate. The test pattern was then connected to the UART input of the MCU. RB was then adjusted until the MCU read the proper value without frame error indications.

The output of the shift registers is then be provided to the input of an XBEE ZB RF module, which consists of ZigBee firmware loaded onto an XBee S2 hardware model: XBEE2. The XBEE modules operate within the ISM 2.4 GHz frequency band and are used in transparent mode. When operating in transparent mode, the modules act as a serial line replacement. All UART data received through the DIN pin is queued up for RF transmission. When RF data is received, the data is sent out through the DOUT pin. The module configuration parameters are configured using the AT command mode interface. For this application the XBEE is configured to match the output signal of the sensor circuit in respect to number of bits, no parity, single stop bit and 4800 BAUD rate.

The final circuit senses the local temperature using the LM34 analog temperature sensor; converts the analog signal to a digital signal for local display and remote transmission. The temperature is displayed locally using the TC7107 ADC 7 segment display driver and 7 segment displays. After analog to digital conversion, the digital temperature signal is transferred to the parallel input of the HEF4021B 8 bit shift register along with management bits to form a serial UART signal that is transmitted via the XBEE RF transceiver. Figure 6 shows the complete temperature sensor schematic (less the local display circuit).

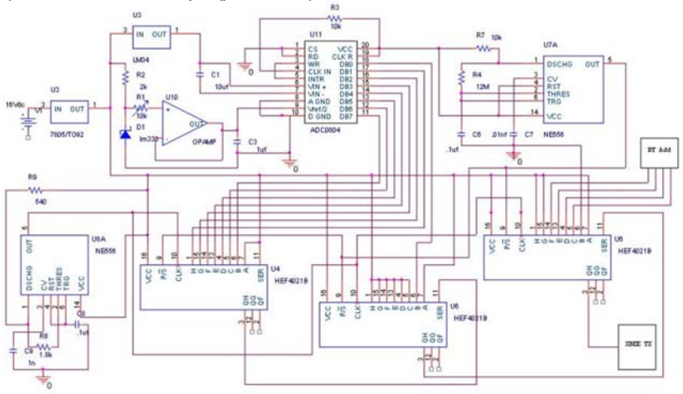


Figure 6. Temperature Sensor Schematic

The finished temperature sensor module was designed to be powered by a typical DC wall transformer. The module draws less than 500mA. A typical 600mA wall transformer was used for the power supply. A 7805 5 volt power regulator was used to provide the TTL logic for the circuitry. Figure 7 shows the prototype sensor circuit.

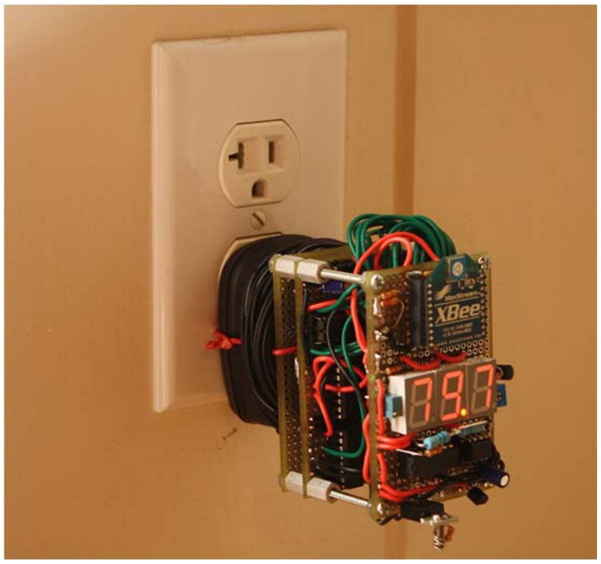


Figure 7. Temperature Sensor Module

3.2 Centralized HVAC Control Circuit

The function of the centralized HVAC control system is to take total control away from the system thermostat. The core component of the control circuit is an ATmega168. The ATMega168 is a High Performance, Low Power Atmel® AVR® 8-Bit Microcontroller Unit (MCU). The MCU acquires digital temperature information from the remote temperature sensor circuits. This data along with control inputs from the system thermostat is sampled by the MCU to make logical decisions for system power on and power off and remote vent register operation.

The Honeywell RTH3100C thermostat provides inputs to an MCU that are normally input directly to the HVAC. The thermostat typically has two 24V inputs that are applied to the actuator of a power relay during the on state of the heat or cool cycle. These inputs are provided to the MCU on pins PB1 and PB2 in the form of TTL logic. The thermostat is connected to +5 V that is in an on state is shorted to an input of the MCU indicating that the system thermostat is in its on state.

The control system turns the system on and off based on inputs by not only the centralized system thermostat, but also the inputs provided by the temperature sensing circuits. Temperature sensor inputs are input on pin PD0 which is set up for UART serial receive. Eight bit digital temperature data is decoded from the UART receive data stream and compared to the desired temperature of operation determined by the user.

The control system has a temperature set input for the user to control the temperature of operation. This was accomplished by using the ADC of the MCU. A simple voltage divider circuit was used to provide a voltage from 0 to 1V at the input of the ADC on pin PC0 which is set up for ADC operation. This analog signal is decoded by the MCU software as a temperature for 0 to 100 degrees.

Heating or cooling operation of the control system is determined by the state of the Heat/Cool switch. A SPDT switch provides a TTL logic input to pin PC5. Logic high places the system in heating mode; logic low places the system in a cooling mode.

The MCU control circuit also provides the user with LCD readout to monitor the system temperature setting and the remote zone temperatures. Figure 8 shows the MCU control circuitry complete with the XBEE transceiver and LCD readout.

After receiving the temperature data from the remote temperature sensors and logic from the system thermostat, the MCU implements a system control algorithm that provides on and off logic as well as control to a motorized vent register circuit. Figure 9 shows control circuit algorithm flowchart for decision making and system control. If the MCU determines that any monitored zone or the local thermostat requires climate control, the MCU turns on the HVAC according to the temperature setting. The MCU then determine which zones if any do not require climate control, and close the appropriate zone vents. Figure 10 shows the integrated thermostat and HVAC Control Module.

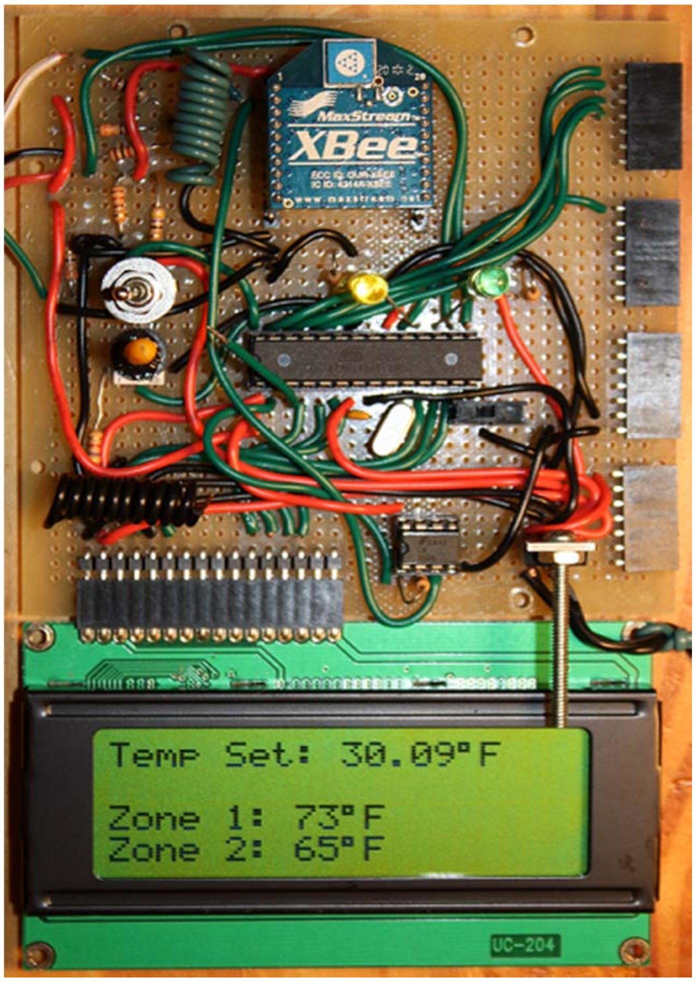


Figure 9. MCU Control Circuit

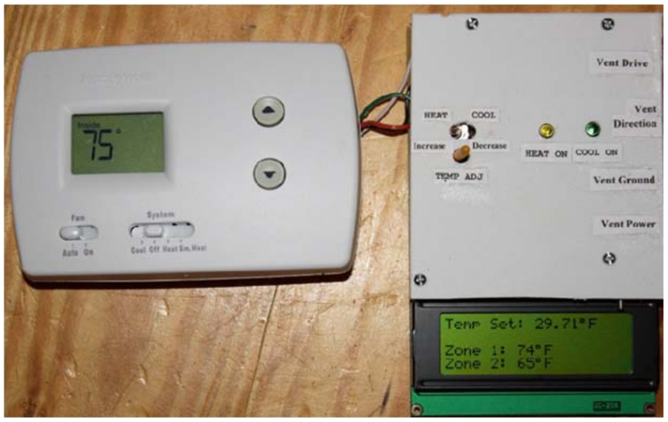


Figure 10. Integrated HVAC Control Module

3.3 Motorized Vent Register Circuit

One major component of the remote climate control is the use of motorized vent registers. The motorized vent register circuit consists of airflow dampeners installed inside the HVAC duct work. The airflow restrictor is controlled by stepper motors driven by the MCU. Figure 11 shows the diagram.

The stepper motors used are small hybrid bipolar stepping motors with a 1.8° step angle (200 steps/revolution). Each phase of the stepper motor draws 600 mA at 3.9 V, allowing for a holding torque of 180 g-cm (2.5 oz-in). The motors have four color-coded wires and a 4mm-diameter output shaft.

The stepper motor is driven by an A4983 stepper motor driver board. The driver board is controlled by two connections from the MCU (step and direction) that will advance the motor by 50 steps open to close. The driver board gets its power from the 16V DC transformer used to power the MCU control circuit and uses a built in power regulator.

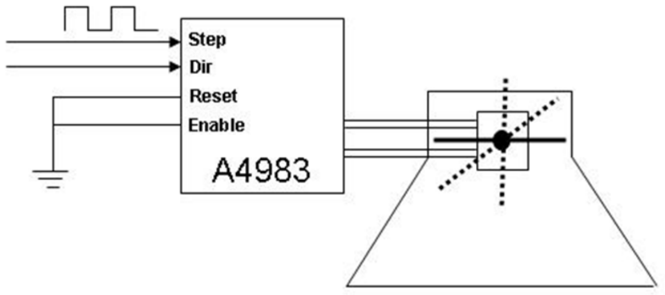


Figure 11. Motorized Vent Register Circuit

3.4 MCU Control Algorithm

The MCU invokes a control algorithm that provides on and off commands based on the thermostat and temperature sensors. The algorithm also makes decisions based on these inputs to set the position of the vent registers and provide the user with centralized temperature monitoring. The software code used to develop the MCU control algorithm relied heavily on sample code from the Nerdkits Guide [9] as a starting point. Example code from Nerdkit projects were integrated into the final software design to provide the MCU ADC, delays, UART, and LCD display functionality. The MCU logic flowchart is shown in Figure 12. The following steps describe the flow of the MCU algorithm.

1. Determine the value of the system temperature setting.
2. Read UART input and determine the RT address and associated temperature data.
3. Determine the absence of any zone reports for a certain set time threshold.
4. Set any zone that exceeds threshold to off.
5. Print system temperature setting and zone temperature to LCD.
6. Determine the Heat/Cool setting.
7. Compare temperature setting to zone temperature and adjust vents accordingly.
8. Determine if thermostat is providing an on command and if so turn system on.
9. Determine if zone temperatures require system to be on and if so turn system on.
10. Repeat steps 1-9

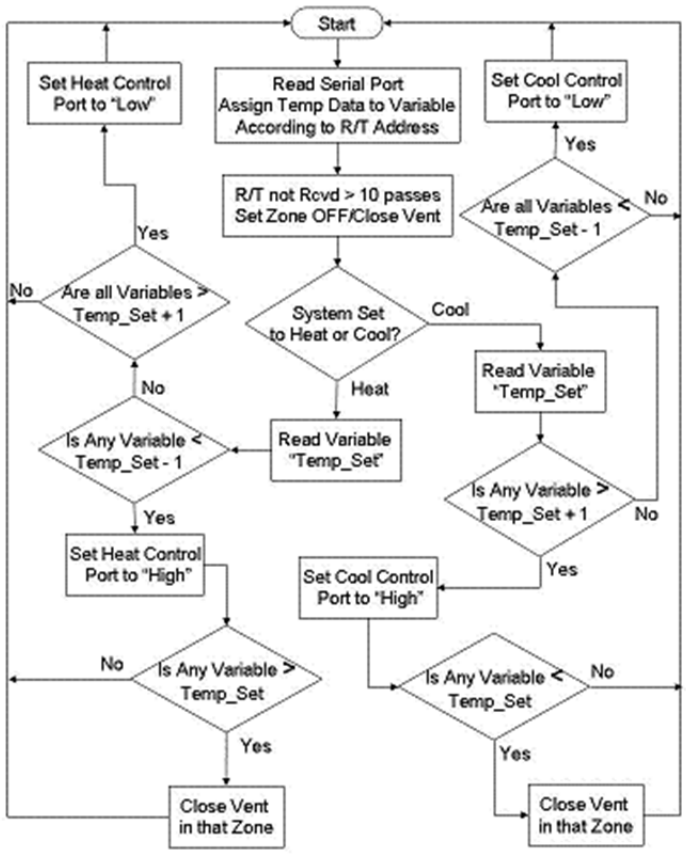

Start
Set Heat Control Port to "Low"
Read Serial Port Assign Temp Data to Variable According to R/T Address
Set Cool Control Port to "Low"
R/T not Rcvd > 10 passes Set Zone OFF/Close Vent
Yes
Are all Variables < Temp_Set - 1
No
Yes
No
Are all Variables > Temp_Set + 1
System Set to Heat or Cool?
Cool
Heat
Read Variable "Temp_Set"
No
Is Any Variable < Temp_Set - 1
Read Variable "Temp_Set"
Is Any Variable > Temp_Set + 1
No
Yes
Set Heat Control Port to "High"
Yes
Set Cool Control Port to "High"
No
Is Any Variable > Temp_Set
Is Any Variable < Temp_Set
No
Yes
Yes
Close Vent in that Zone
Close Vent in that Zone

Figure 12. MCU Logic Flowchart

4 Conclusions

The intended users are customers needing to evenly distribute HVAC airflow throughout multiple zones. A solution was found for the project design. The design incorporated a temperature sensor circuit using an analog temperature sensor that drives two analog to digital converters that in turn drive a 7 segment LED display and an RF transmitter. The digital data that is transmitted contains one of eight possible RT addresses representing a single zone and eight bits of digital temperature data. Each transmitter is timed to transmit out of phase with the other transmitters, thus providing a pseudo TDMA network where and all transmitters can use the same frequency. The transmitted data is received by a compatible receiver and transferred to a microcontroller that invokes an algorithm that turns on and off the HVAC system based on the inputs from the temperature sensors as well as the system thermostat. The microcontroller also provides control for motorized vent registers to vary the airflow in each zone based on the temperature sensor data. A prototype unit was built and tested, and demonstrated. The Zone Control System met all the design objectives and specifications.

During the course of building the HVAC zones control system, there were small improvements that were identified for future designs. The original RF transmission design was abandoned due to the devices inability to reliably transmit a UART signal. The replacement for the RF transmitter was the XBEE which has capabilities that were not used. The XBEE transceiver has a built in ADC that can be taken advantage of in a follow on design, reducing costs by eliminating the ADC and one of the timers.

The XBEE transceivers were used with a built in chip antenna. The antenna specifications stated a 100ft range. In an indoor environment was not realistic, so an XBEE with external antenna would be better suited for this application.

The software code used for the project was not efficient. As an electrical engineer, the level of code required for this project was too complicated. The design works, but a proficient software developer would be able to make significant improvements to the system functionality.

5 References

[1] Gale Reference Team, *Temperature sensors, monitors, controls*, Plastics Technology (Magazine/Journal) November 1, 2008

[2] Waltenegus Dargie and Christian Poellabauer, *Fundamentals of Wireless Sensor Networks: Theory and Practice (Wireless Communications and Mobile Computing)*, Wiley, September 21, 2010

[3] Balraj A., Patvardhan A., Renuka Devi V., Aiswarya R., and Prasen V., *Embedded Temperature Monitoring and Control Unit*, in "Recent Trends in Information, Telecommunication and Computing (ITC), 2010 International Conference, pp 293-297, School of Interdisciplinary Science and Technology, 2010, IEEE 10.1109/ITC.2010.91

[4] Rashid M. H., *Microelectronic Circuits: Analysis & Design: Analysis and Design*, CL-Engineering; 1st edition, April 1998

[5] Mano Morris M., Ciletti Michael D., *Digital Design*, Prentice Hall; 4th edition, December 25, 2006

[6] Atmel Corporation, *8-bit AVR microcontroller users guide*, 2545S–AVR–07/10

[7] National Semiconductor, *Temperature sensor* handbook

[8] Evans Humberto, Robbins Michael F., *The USB Nerdkits Guide*, rev 637-5170nwftf673h88cpzr8, 2009

[9] Digi International, X-CTU Configuration & Test Utility Software User's Guide

REAL-TIME sEMG ACQUISITION AND PROCESSING USING a PIC 32 MICROCONTROLLER

Chandrasekhar Potluri, *Member, IEEE,* Madhavi Anugolu, *Member, IEEE,* Amir Fassih, Yimesker Yihun, Parmod Kumar, Steve Chiu, *Member, IEEE,* and D. Subbaram Naidu, *Fellow, IEEE*

Abstract–**This paper presents a novel approach for the signal acquisition and processing using an embedded platform. A PIC 32 microcontroller is used to acquire Surface Electromyographic (sEMG) signals and the corresponding skeletal muscle force signal for the respective motor point in this case, for the ring finger of the dominant hand of a healthy subject. The data is acquired by MATLAB®/SIMULINK®. Both sEMG and force signals are acquired at a rate of 2000 samples per sec. The acquired sEMG data is filtered using three different types of nonlinear Bayesian filter, Exponential, Poisson, and Half-Gaussian filter and the force signal is filtered using a Chebyshev type II filter. The data acquired from the PIC 32 and embedded test bed are compared with the standard LabVIEW™ data acquisition system using these three filters and Half-Gaussian filter with DE 3.1 electrodes gives better results.**

I. INTRODUCTION

Human body is one of the most complex and intricate system available. In the same way the human body signals such as sEMG are quite complex and challenging to understand. To aid the peoples with upper extremity amputations there has been an active research in the field of upper limb prosthesis. Loss of upper extremity brings a reduction in functions to amputees and also struggle with numerous psychological issues which may further complicate the appropriate control and use of the prosthesis. Research work in [1] and [2] was mainly based on the electromyography (EMG) signals. The EMG signal is a naturally available and can be recorded at the surface of the limb which is known as surface EMG (sEMG).

Chandrasekhar Potluri is with MCERC, School of Engineering, Idaho State University, Pocatello, Idaho 83209, USA (email : potlchan@isu.edu).

Madhavi Anugolu is with MCERC, School of Engineering, Idaho State University, Pocatello, Idaho 83209, USA (email : anugmadh@isu.edu).

Amir Fassih is with Measurement and Control Engineering Research Center (MCERC), School of Engineering, Idaho State University, Pocatello, Idaho 83209, USA (email: fassamir@isu.edu).

Yimesker Yihun is with MCERC, School of Engineering, Idaho State University, Pocatello, Idaho 83209, USA (e-mail: yihuyime@isu.edu).

Parmod Kumar is with Measurement and Control Engineering Research Center (MCERC), School of Engineering, Idaho State University, Pocatello, Idaho 83209, USA (email: kumaparm@isu.edu).

Steve Chiu is with Department of Electrical Engineering and Computer Science, MCERC, Idaho State University, Pocatello, Idaho 83201 USA (email: chiustev@isu.edu).

D. Subbaram Naidu is with Department of Electrical Engineering and Computer Science, MCERC, Idaho State University, Pocatello, Idaho 83209 USA (email: naiduds@isu.edu).

The sEMG is simply an electric voltage ranging between -5 and +5 mV, which is a result of the electrical activity associated with voluntary muscle contraction. This made the sEMG signal of great use for the position and force control of the hand prosthesis [3, 4].

Since the skeletal muscle force and the sEMG signals are related and an increase force production results in the increased sEMG activity. Therefore the latter is used as a control input to realize force and motion control of a prosthetic hand. This makes the precise interpretation of the sEMG signal an essential task.

In today's research environment the embedded systems have become pervasive and as research advances, more and more functions of analog circuits are being realized by microcontrollers, Analog to Digital Converters (ADCs) and Digital to Analog Converters (DACs). In a modern control system, embedded system and control performs most of the data acquisition, processing and control functions. To realize excellent system performance we need to have a well-designed embedded control which can deal with widely varying operating conditions. For robust, fast, precise and consistent high performance, the embedded system should be designed carefully. In case of a prosthetic hand we need to have a real-time embedded control system that performs the desired force and motion control [5-7].

Present work is a step in this direction where the authors explore the (Peripheral Interface Controller) PIC 32 microcontroller as an embedded platform to simultaneously acquire the sEMG and skeletal muscle force. sEMG sensors are placed on the ring finger motor point of the dominant hand of a healthy subject and the subject is asked to squeeze a stress ball which has a force sensing resistor (FSR) attached to it. The data is simultaneously captured using the PIC 32 embedded platform with MATLAB®/SIMULINK® Real-Time Workshop (RTW) and regular NI LabVIEW™ data acquisition. Both sEMG and force data are captured at 2000 Hz. The sEMG signal is filtered using three different types of nonlinear Bayesian filters, Exponential, Poisson, and Half-Gaussian filter and the corresponding skeletal muscle force is filtered by a Chebyshev type-II filter [8]. Among these three different types of filters the Half-Gaussian filter is giving better results [8-14].

The paper is organized as follows: present introductory section followed by the ‘Experimental Set-Up,’ then the ‘Signal Pre-Processing,’ ‘Proposed Design,’ and ‘Results and Discussion,’ are presented. The paper is concluded with the section of ‘Conclusion and Future Work.’

II. EXPERIMENTALSET-UP

Using a muscle stimulator the motor point for the ring finger of the dominant hand of a healthy subjects are marked. Prior to placing the sEMG sensors, the skin surface of the subject was prepared according to International Society of Electrophysiology and Kinesiology (ISEK) protocols. Different sets of experiments are conducted with DE 2.1 and DE 3.1 DELSYS® Bagnoli sEMG sensors. One sensor was placed on top of the motor point location and two sensors were placed next to the motor point. Subject is asked to squeeze the stress ball with the ring finger which has a 0.5 inch force sensing resistor from Interlink™ Electronics. The sEMG and skeletal muscle force signals were acquired using the 16-channel DELSYS® Bagnoli sEMG and NI ELVIS™ respectively. Similar experimental set-up was designed using a PIC 32 embedded platform where the sEMG and the force data is acquired using this platform. Both the data are captured at a sampling frequency of 2000Hz. Fig. 1 and 2 show the two experimental set-ups.

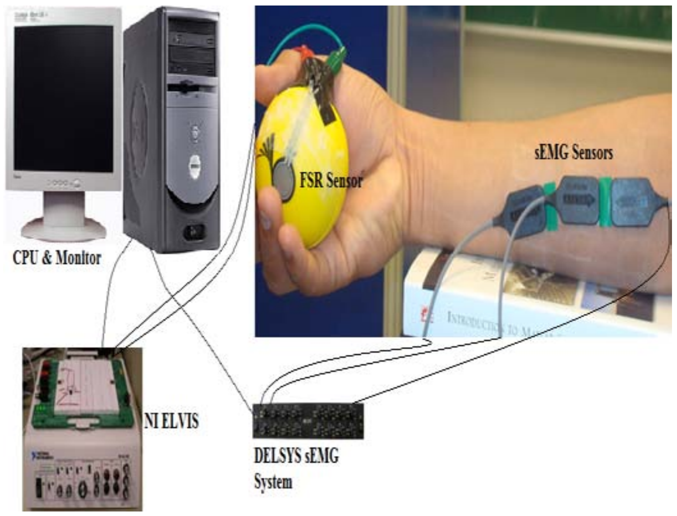


Fig.1. Experimental Set-Up with NI ELVIS and DELSYS® EMG System.

III. SIGNAL PRE-PROCESSING

Previous research work [15] shows that the Bayesian based filtering method yields the most suitable sEMG signals. These nonlinear filters significantly reduce noise and extract a signal that best describes EMG signals and can be effective for prosthetic hand signal processing. The latent driving signal x results in the EMG which can be computed using an instantaneous conditional probability$P(EMG\,|\,x)$,[15]. Research work in [16] describes EMG signal as amplitude-modulated zero mean Gaussian noise sequence. This estimation algorithm uses the model of the conditional probability of the rectified EMG signal $emg = |EMG|$, [15].

Equation (1) gives an “Exponential Measurement Model” for the rectified EMG signal [15].

$$P(emg|x) = \frac{\exp(\frac{-emg}{x})}{x}. \qquad (1)$$

Equation (2) gives a “Poisson Measurement Model” for the rectified EMG signal [15].

$$P(emg|x) \approx x^n \frac{exp^{-x}}{n!}. \qquad (2)$$

In equation (2) n is the number of events. Equation (3) presents the “Half-Gaussian measurement model” for the rectified EMG signal [15].

$$P(emg|x) = \frac{2*\exp(-\frac{emg^2}{2x^2})}{\sqrt{(2\pi x^2)}}. \qquad (3)$$

The model for the conditional probability of the rectified EMG is a filtered random process with random rate and the likelihood function for the rate evolves in time according to a Fokker–Planck partial differential equation [15]. The discrete time Fokker–Planck Equation is given by Equation (4).

$$p(x,t-) \approx \alpha * p(x-\varepsilon, t-1) + (1-2*\alpha) * p(x,t-1) + \alpha * p(x+\varepsilon, t-1) + \beta + (1-\beta) * p(x,t-1) \qquad (4)$$

In the Equation (4) α and β are two free parameters, where $\alpha$the expected rate of gradual is drift and β is the expected rate of sudden shift in the signal [15]. The latent driving signal x is discretized into bins of ε. An elitism based Genetic Algorithm (GA) is used to optimize these free parameters of the non-linear Half-Gaussian filter model. GA is an optimization algorithm which is based on observing nature and its corresponding processes to imitate solving complex problems, most often optimization or estimation problems, see [17-18]. Skeletal muscle force signal from FSR is filtered utilizing a Chebyshev type II low pass filter with a 550 Hz pass band frequency.

IV. PROPOSED DESIGN

In this proposed design the analog input and the UART (universal asynchronous receiver/transmitter**)** modules of the PIC 32 are used for the acquisition and transmission of the sEMG signals. The outputs from the DELSYS® Bagnoli system are connected to the analog input channels of the PIC 32 micro controller. In this present work the signal from the motor unit (middle sensor) is acquired and pre-processed. The sEMG signal and the corresponding skeletal muscle force are acquired at 2000 samples per second. A dsPIC blockset is used to generate the C code for the PIC32 from SIMULINK®. The dsPIC blockset generates a ‘.hex’ file, and this file is imported by MPLAB® to program the PIC32. The analog input module is used for reading the sEMG and the corresponding skeletal muscle force data. The PIC32 has an internal ADC which has a 10-bit resolution so that it can distinguish up to 1024 different voltages, usually in the range of 0 to 3.3 volts, and it yields 3mV resolution.The UART module in the PIC32 is used to transmit the signals from the microcontroller to the PC using serial communication. In this design, a virtual 'com port' is created to feed the data via USB cable to the computer. MATLAB® is used to read the signals from the ports. The acquisition system using the PIC 32 micro controller is shown in the Fig. 2.

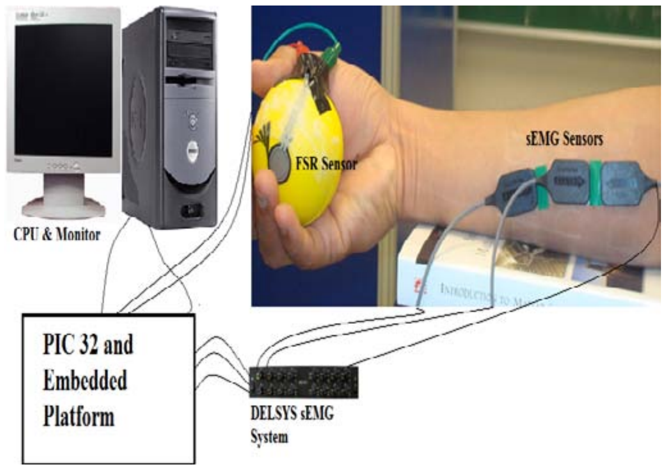


Fig. 2. Experimental Set-Up with PIC 32 Embedded Platform and DELSYS® EMG System.

V. RESULTS AND DISCUSSION

The sEMG and the corresponding skeletal muscle force data are acquired from the microcontroller through UART channel 2 of the PIC32MX360F512L by a virtual com port via USB at 57600 baud rate. The data from the microcontroller is converted into uint16 data before it is transmitted through the UART. The PIC32 microcontroller is running at 80 million instructions per second (MIPS) with its phase lock loop (PLL) activated with a clock frequency of 8MHz with internal scaling enabled. Figs. 3 and 4 show the sEMG signal acquired by the proposed acquisition system using DE 2.1 electrodes. Fig. 5 and 6 show the sEMG signals acquired by the proposed acquisition system using DE 3.1 electrodes. These acquired signals are processed using the Half-Gaussian filter.

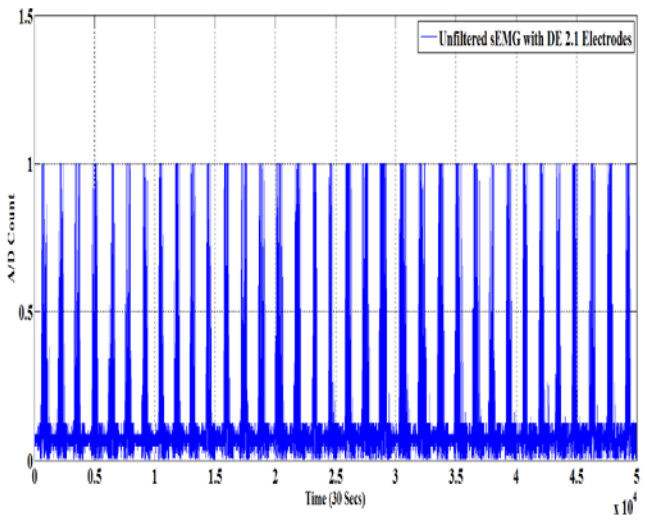


Fig. 3. Unfiltered sEMG Signal from the Proposed Acquisition System Using DE 2.1 Electrodes.

The following experiment is repeated several times to check the consistency and the accuracy of the proposed acquisition system. Figs. 7 and 8 show the validation for the proposed acquisition system for repeated experiments using DE 2.1 and DE 3.1 electrodes. It is evident from the Fig. 6 that the DE 3.1 electrodes are giving good results when compared to DE 2.1 results.

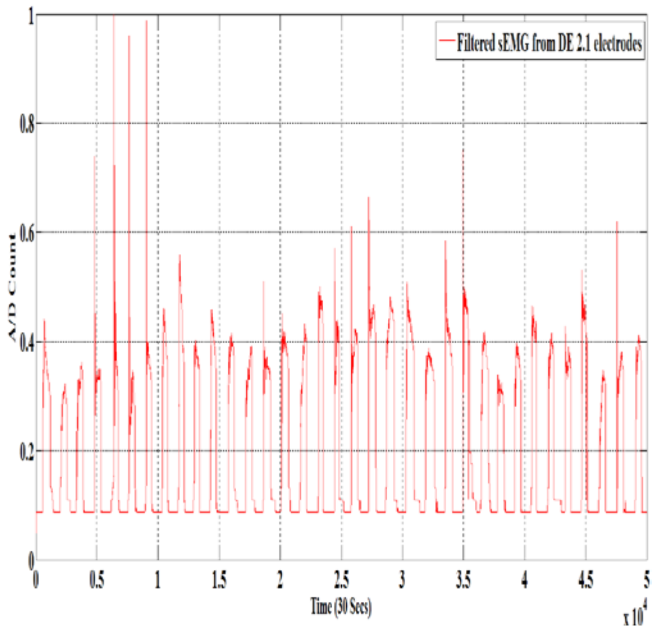


Fig. 4. Filtered sEMG Signal from the Proposed Acquisition System Using DE 2.1 Electrodes.

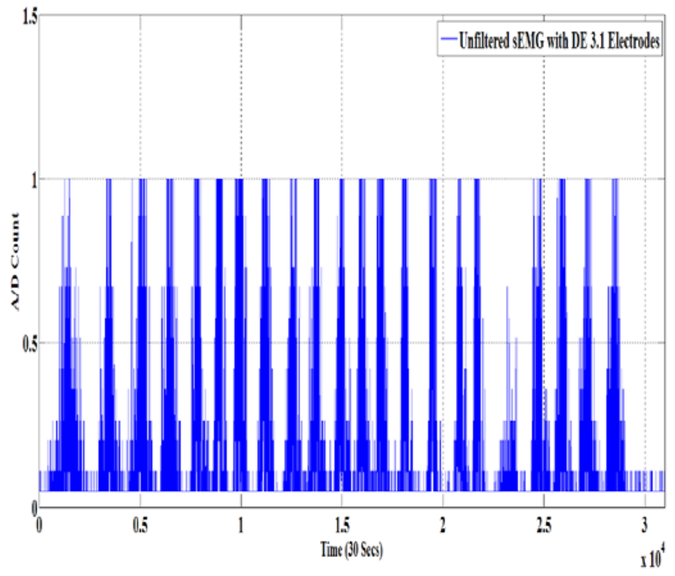


Fig. 5. Unfiltered sEMG Signal from the Proposed Acquisition System Using DE 3.1 Electrodes.

The sEMG signals acquired from the proposed acquisition system using DE 3.1 electrodes are in good correlation with the sEMG signals acquired from the standard acquisition system. The sEMG signals and the corresponding skeletal muscle force acquired from the standard acquisition system are given in Fig. 9 and 10. Since the sEMG is a random signal corrupted with noise it is hard to achieve the same correlation every time.

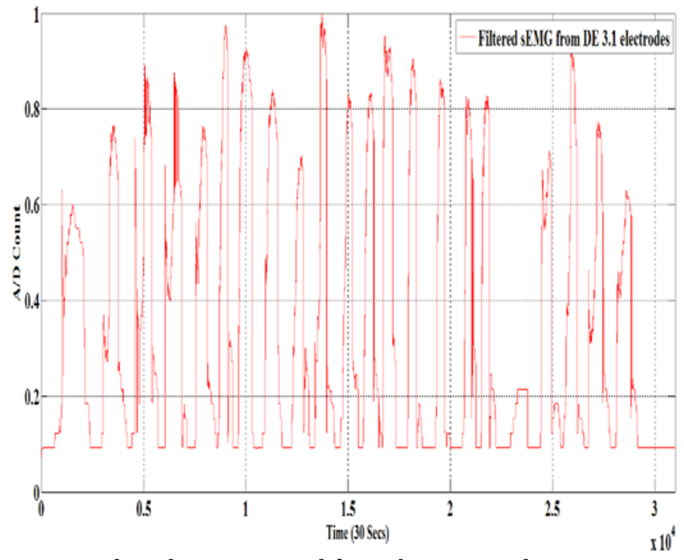


Fig. 6. Filtered sEMG Signal from the Proposed Acquisition System Using DE 3.1 Electrodes.

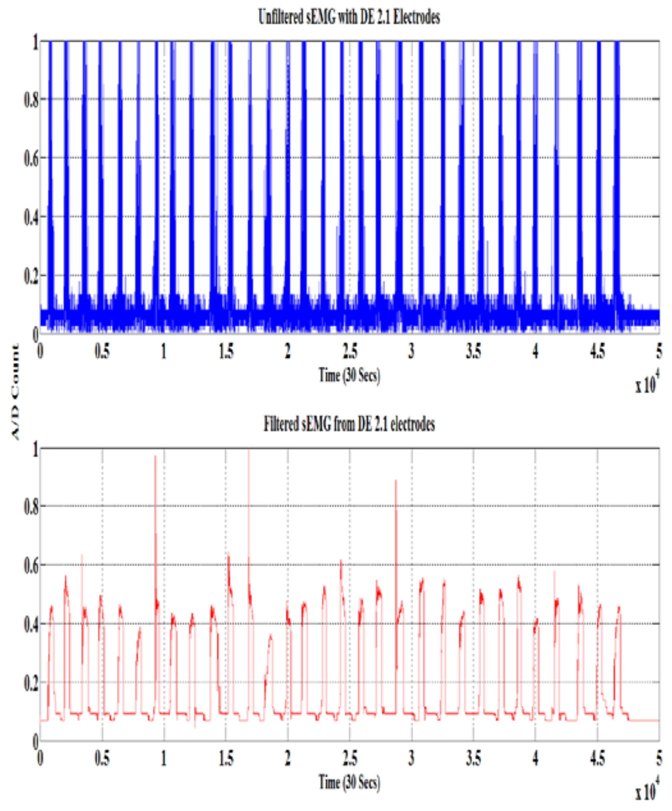


Fig. 7. sEMG Signal from the Proposed Acquisition System Using DE 2.1 Electrodes with Repeated Experiment.

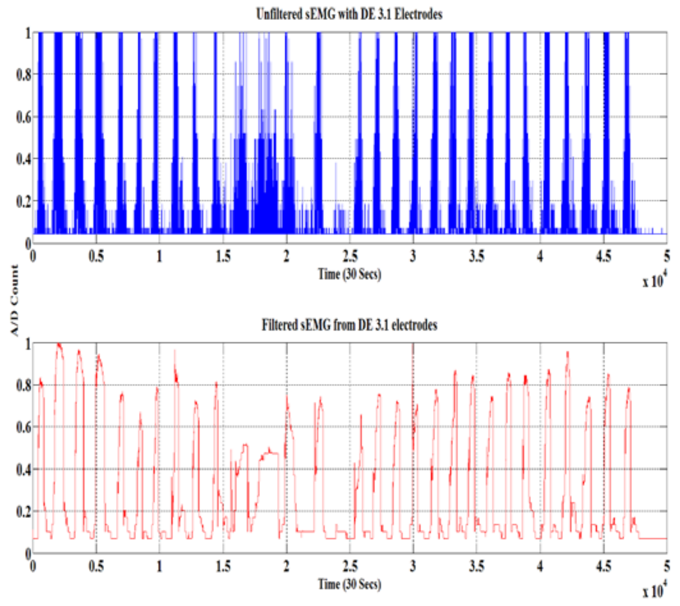


Fig. 8. sEMG Signal from the Proposed Acquisition System Using 3.1 Electrodes with Repeated Experiment.

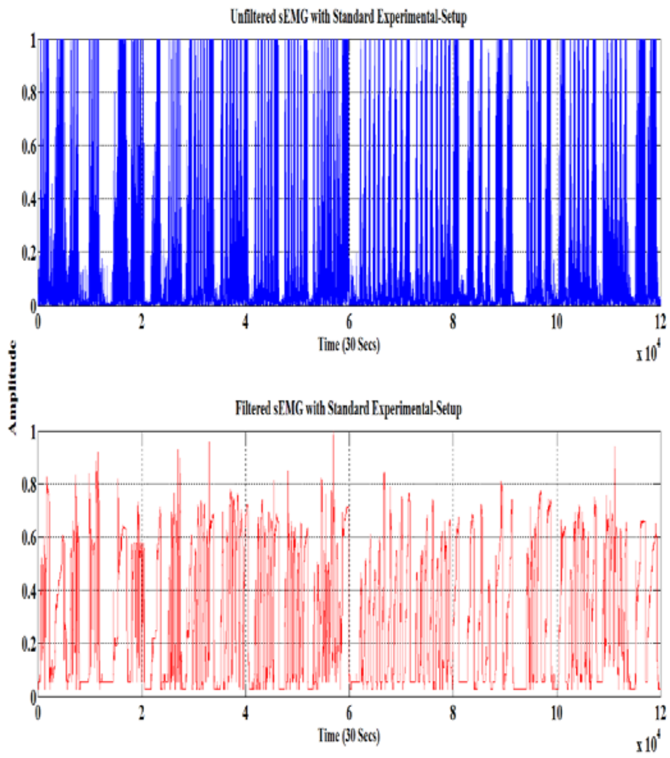


Fig. 9. sEMG Signal from the Standard Acquisition System.

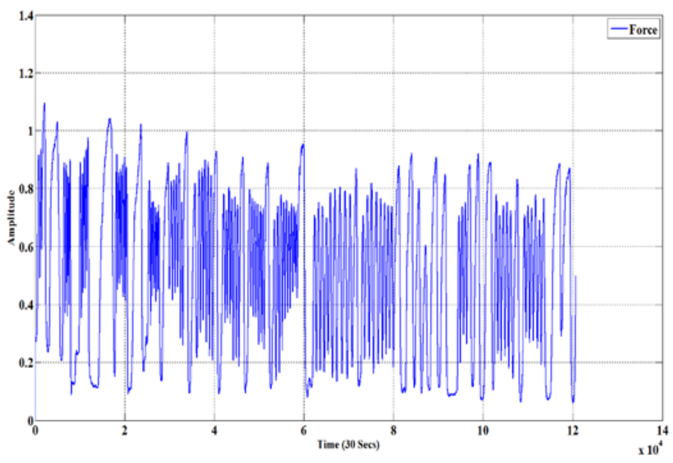


Fig. 10. Filtered Force Signal from the Standard Acquisition System.

VI. CONCLUSION AND FUTUREWORK

A real-time sEMG acquisition system was designed for the control of the robotic hand prototype. The proposed design gives better performance when compared with the standard EMG acquisition system. Comparison between DE 2.1 and the DE 3.1 electrodes is done and the DE 3.1 electrodes are giving better results. This proposed acquisition system facilitates the transmission of the data from the microcontroller to the computer. This enables the user to compare the accuracy and performance of the acquisition system. Investigation was done on various other filters and finally Half-Gaussian filter with DE 3.1 elctrodes is giving better results.

For future work we are planning to implement real time model-based force estimation and along with controller design for position and force control, based on this embedded platform [19, 20]. It will be interesting to acquire signals from the three sensors and then make a comparison between the standard and the proposed acquisition system. Finally, we plan to implement the acquisition and control for a five fingered prototype.

ACKNOWLEDGMENT

This research was sponsored by the US Department of the Army, under the award number W81XWH-10-1-0128 awarded and administered by the U.S. Army Medical Research Acquisition Activity, 820 Chandler Street, Fort Detrick MD 21702-5014. The information does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred. For purposes of this article, information includes news releases, articles, manuscripts, brochures, advertisements, still and motion pictures, speeches, trade association proceedings, etc. Further, the technical help from Dr. Marco P. Schoen is greatly appreciated.

REFERENCES

[1] N. Dechev, W. L. Cleghorn, and S. Naumann,“Multiple finger, passive adaptive grasp prosthetic hand,”*Mechanism and Machine Theory*, 36(2001), pp. 1157-1173.

[2] H. Kawasaki, T. Komatsu, and K. Uchiyama, “Dexterous Anthropomorphic Robot Hand With Distributed Tactile Sensor: Gifu Hand II,”*IEEE/ASME Transactions on Mechatronics*, Vol. 7, No. 3, September 2002, pp. 296-303.

[3] M. Zecca, S. Micera, M. C. Carrozza, and P. Dario, “Control of Multifunctional Prosthetic Hands by Processing the Electromyographic Signal,”*Critical Reviews™ in Biomedical Engineering*, 30(4-6), 2002, pp. 459-485.

[4] C. Castellini and P. van der Smagt, “Surface EMG in advanced hand prosthetics,”*Biological Cybernetics*, (2009) 100, pp. 35-47.

[5] C. Potluri, P. Kumar, J. Moliter, M. Anugolu, A. Jensen, K. Hart, and S. Chiu, “Multi-Level Embedded Motor Control for Prosthesis,” *International Conference on Embedded Systems and Applications*, ESA’2010, Las Vegas, Nevada, USA, July 12-15, 2010.

[6] C. Potluri, P. Kumar, M. Anugolu, S. Chiu, A. Urfer, M. P. Schoen, and D. S. Naidu, “sEMG Based Fuzzy Control Strategy with ANFIS Path Planning For Prosthetic Hand,” *3rd IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics*, Tokyo, Sept 26-30, 2010.

[7] C. Potluri, Y. Yihun, P. Kumar, J. Molitor, S. Chiu, D. S. Naidu, and S.H. Mousavinezhad, “sEMG Based Real-Time Embedded Force Control Strategy for a Prosthetic Hand Prototype” *IEEE International Conference on Electro/Information Technology*, Mankato, Minnesota, USA, May 15-17, 2011.

[8] M. Anugolu, A. Sebastain, P. Kumar, M. P. Schoen, A. Urfer, and D. S. Naidu, “Surface EMG Array Sensor Based Model Fusion Using Bayesian Approaches for Prosthetic Hands,” *2009 ASME Dynamic Systems and Control Conference*, Hollywood, California, USA, Oct. 12-14, 2009.

[9] C. Potluri, P. Kumar, M. Anugolu, A. Urfer, S. Chiu, D. S. Naidu, and M. P. Schoen, "Frequency Domain Surface EMG Sensor Fusion for Estimating Finger Forces," *32nd Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Buenos Aires, Argentina, Aug. 31 - Sept. 4, 2010.

[10] P. Kumar, A. Sebastian, C. Potluri, A. Urfer, D. S. Naidu, and M. P. Schoen, "Towards Smart Prosthetic Hand: Adaptive Probability Based Skeletal Muscle Fatigue Model," *32nd Annual International Conference of the IEEE Engineering in Medicine and Biology Society,* Buenos Aires, Argentina, Aug. 31 – Sept. 4, 2010.

[11] P. Kumar, C. Potluri, A. Sebastian, S. Chiu, A. Urfer, D. S. Naidu, and M. P. Schoen, "An Adaptive Multi Sensor Data Fusion with Hybrid Nonlinear ARX and Wiener-Hammerstein Models for Skeletal Muscle Force Estimation," The 14th World Scientific and Engineering Academy and Society (WSEAS) International Conference on Systems, Corfu Island, Greece, July 22-24, 2010.

[12] P. Kumar, C. Potluri, A. Sebastian, S. Chiu, A. Urfer, D. S. Naidu, and M. P. Schoen, "Adaptive Multi Sensor Based Nonlinear Identification of Skeletal Muscle Force,"*WSEAS Transactions on Systems,* Issue 10, Volume 9, October 2010, pp. 1051-1062, 2010.

[13] P. Kumar, C. Potluri, M. Anugolu, A. Sebastian, J. Creelman, A. Urfer, S. Chiu, D. S. Naidu, and M. P. Schoen, "A Hybrid Adaptive Data Fusion with Linear and Nonlinear Models for Skeletal Muscle Force Estimation," *5th Cairo International Conference on Biomedical Engineering,* Cairo, Egypt, Dec. 16-18, 2010.

[14] P. Kumar, C. Potluri, A. Sebastian, Y. Yihun, A. Ilyas, M. Anugolu, R. Sharma, S. Chiu, J. Creelman, A. Urfer, D. S. Naidu, and M. P. Schoen, "A Hybrid Adaptive Multi Sensor Data Fusion for Estimation of Skeletal Muscle Force for Prosthetic Hand Control," *The 2011 International Conference on Artificial Intelligence, ICAI'11,* Las Vegas, Nevada, USA, July 18-21, 2011.

[15] T. D. Sanger, "Bayesian Filtering of Myoelectric Signals,"*J Neurophysiol,* 97, 2007, pp. 1839–1845.

[16] M. B. I. Reaz, M. S. Hussain and F. Mohd-Yasin, "Techniques of EMG signal analysis: detection, processing, classification and applications,"*Biol. Proced. Online, 2006,* 8(1), pp. 11-35.

[17] E. Kral, L. Vasek, V. Dolinay, P. Varacha, "Usage of PSO Algorithm for Parameter Identification of District Heating Network Simulation Model,"*The 14th World Scientific and Engineering Academy and Society (WSEAS) International Conference on Systems,* Corfu Island, Greece, July 22-24, 2010.

[18] A. Neubaur, "The Intrinsic System Model of the Simple Genetic Algorithm with α-Selection, Uniform Crossover and Bitwise Mutation,"*The 14th World Scientific and Engineering Academy and Society (WSEAS) International Conference on Systems,* Corfu Island, Greece, July 22-24, 2010.

[19] C. Potluri, Y. Yihun, M. Anugolu, P. Kumar, S. Chiu, M. P. Schoen, and D. S. Naidu, "Implementation of sEMG-Based Real-Time Embedded Adaptive Finger Force Control for a Prosthetic Hand", submitted to *IEEE CDC, 2011.*

[20] C. Potluri, M. Anugolu,Y. Yihun,A. Jensen, S. Chiu, M. P. Schoen, and D. S. Naidu, "Optimal Tracking of a sEMG based Force Model for a Prosthetic Hand," submitted to*IEEE EMBS, 2011.*

Electronic Healthcare Model Based on Smart Card For Saudi Medical Centers

Ebtisam Alabdulqader[1], **and H. Fourar-Laidi**[2]
[1] Information Technology Department / [2] Information Systems Department,
College of Computer and Information Sciences, King Saud University, Riyadh, Saudi Arabia

Abstract - *This paper presents a healthcare model based on smart cards. The purpose of the proposed model is to facilitate information exchange and integration across medical organizations. Currently, in Saudi Arabia, all medical centers have their own healthcare system. Each time a patient visits a clinic; a new file is created for him. Clinical information history related to the previous consultations is not available. Confidential information is not protected by the traditional folder. Prescriptions provided by doctors are signed manually and not efficiently authenticated by pharmacies. Claims are transmitted to the insurance companies by fax which makes them subject to falsification. The existing approach of the ministry of health in maintaining information about patients is not efficient, because healthcare institutions are not linked to a central system. In the proposed solution, the smart card will be used to store medical history of a patient that contains all diagnosis and drug prescriptions done in different medical centers. It will be used also to store keys and perform all the cryptographic computations. The medical institutions should adhere to the government healthcare organization system managed by the ministry of health. The healthcare organization should play the trusted third party as certification authority (CA). The information exchanged by the medical institutions should be protected and secured.*

Keywords: Healthcare, Smart card, Cryptography, Medical record, PKI

1 Introduction

Smart cards have been used by many countries to deploy healthcare programs [1, 2]. The smart card issued by healthcare organizations can be read easily with an appropriate smart card reading device and facilitates the secure sharing of patient clinical data amongst multiple healthcare providers. Smart cards can be used by healthcare institutions to considerably reduce the administrative work. Patients will no longer be requested to fill their personal information and describe their medical history each time they visit a healthcare institution. The proposed solution aims to use the smart card as portable storage device for clinical information that can be shared between healthcare institutions. Our approach consist on a integrated solution that use the smart card to perform secure transactions between the medical center, the insurance company, the pharmacy and the trusted healthcare organization.

In the proposed approach, each time a patient visits a medical center, he presents his health card. The medical center will use the card to authenticate the patient and retrieve all needed information and clinical records history. The clinic can also update the card by information related to the current visit. The health card is protected by a Personal Identification Number (PIN) known only by the patient. Retrieving confidential information and updating the card is performed by presenting the card PIN. The information stored in the card is synchronized with a central healthcare system to assure coherence and availability of information in case of lost, or in case the patient visits another medical center. The central healthcare system should be managed by a government organization like the ministry of health. The prescriptions given by doctors are electronically signed and sent to the pharmacy central service, part of the central healthcare system. Doctors cannot deny their prescriptions. Pharmacies can claim their money directly from the insurance company after authorization given by the patient. The pharmacy submits a request to the insurance company. Once the transaction approved, a response is sent to the pharmacy for completion. The transaction between the pharmacy and the insurance company is performed securely by using keys and certificates stored in the smart card.

Due to their cryptographic capability and portability, Smart cards are an ideal secure storage device that can be used to store and secure patient records. Smart card will be used also to secure the information exchange between the clinic access point and the central healthcare system using digital signatures, certificates and keys. The proposed model will be implemented and distributed among the medical centers (clients). In our project, we did not implement the transaction service between the pharmacy and the insurance company. The developed prototype includes the transactions between the patient, medical center and the healthcare central system. The use of smart cards to secure transactions assures more protection for sensitive data. Smart cards reduce the threat of unauthorized access by the use of stolen credentials because the hacker must both steal the smart card and obtain the PIN. The proposed model also offers more flexibility to the current healthcare system by allowing patients to have rapid access to their electronic medical folder in any medical center. Patients do not need any more to redo medical tests each time they are changing clinic. This paper is structured as follows: in section 2, we present some healthcare models based on smart cards. Section 3 describes our proposed smart card healthcare model and its architecture. In section 4, we illustrate the implementation of the healthcare model using MULTOS technology.

2 Healthcare Models

Smart cards offer a new perspective for healthcare applications due to the security level provided for data storage. Smart cards in healthcare applications can be used for storing information including personal data, insurance policy, emergency medical information, hospital admission data and recent medical records [3]. Numerous healthcare systems around the world start using smart cards to improve the quality of healthcare services [4]. Different healthcare models have been proposed either on national level; e.g. in Germany [5] or regional level; e.g. in US. [6].

One of the interesting orientations ongoing is the attempt to further standardize and develop a framework to support Web-based medical-specific smart card applications [7, 9]. Many countries started developing healthcare programs based on smart cards. Here are three health-related examples. In Germany, the Krankenversichertenkarte is used to manage billing to various health-insurance companies for all services received by the public. Ontario plans to use a smart card to reduce fraudulent access to public health services.

“Carte Vitale” is one of the healthcare models issued in France with the aim to automate the paperwork flow between doctors, patients and insurance companies. Another healthcare model was an extending to the national ID card in Malaysia by adding health information as a function on this card which can only be read by medical professional. This health information provides next of kin contact details, detailed information on lifetime health history, hospital admissions and recent treatments [8]. The goal is to streamline treatment, reduce test duplication and automate interactions between patients, healthcare providers and payer organizations.

Chan et al. propose a vertical standard framework that addresses the design requirements specific to the development of medical applications [9]. The concept of the Java Card Web Servlet (JCWS) was developed to provide a seamless access interface between a Web browser and a Java Card-enabled smart card. In essence, the smart card is viewed as a repository of Web-enabled objects comprised of applets, HTML page, data objects, and Java Card applets. The framework supports tight integration of smart card technology with an existing Web infrastructure. Therefore, a hospital with a Java-compliant smart card reader is able to access medical information directly from the card using a standard Web browser via the JCWS. An applet contained within the card can be dynamically loaded into the browser to browse and update medical information. The applet can also provide Web links to Internet databases to facilitate wide area access of further information such as a video of a recent CT scan, high-resolution X-ray image scans, and so forth. The JCWS binds the database to the framework, while providing Web-based browsing and updating services. In this case, the browsing and updating applet can be downloaded from the smart card itself via the browser interface.

Another model was proposed by Song and co. this involved transferring medical prescriptions from the medical center to the pharmacy via the internet based on 2-way double-type smart card terminal. This later is used to control security and privacy of patients and manage drug histories of them [10]. The digital signatures written by the medical professionals (doctors and pharmacists) holding and using their individually master smart cards are applied to all contents of the prescription stored on a patient’s slave smart card at the synchronized status in the 2-way double-type terminal. Finally, Attiaoui and al. describe an approach of using the Web USB smart card service model as a common interface to communicate and access the medical records residing in a smart card that seamlessly integrate to existing Web infrastructure [11].

3 The Smart Card Healthcare Model

Healthcare sector in Saudi Arabia needs simple identity cards for all patients to grant access to certain data in anywhere and at anytime. The proposed smart card healthcare model represents an alternative solution to improve the whole healthcare infrastructure by adopting a comprehensive multifunctional smart card system. The essential idea is based on multifunction smart card that is used as identity for patient authentication and stores medical data along with secure interchange of these data which will be used in various locations, such as hospitals, clinics and ambulances. According to that, the model will support both availability and security of the medical data.

A health smart card will be issued for each patient to replace their paper based medical record that is currently used in the region. The smart card will be utilized to store only the critical medical record information needed for each clinic visit and emergency cases. This limitation is due to the limited capacity of the currently available smart cards. The critical medical record information that will be stored on the smart card includes the personal and emergency contact information, urgent medical data such as allergies and list of current medications if any, as well as the latest patient medical examinations and prescriptions.

3.1 Architecture of the proposed model

The smart card healthcare model consists of three components that consist of the patient’s smart card, the client terminals and the central system located in the ministry of health as shown in Figure.1.

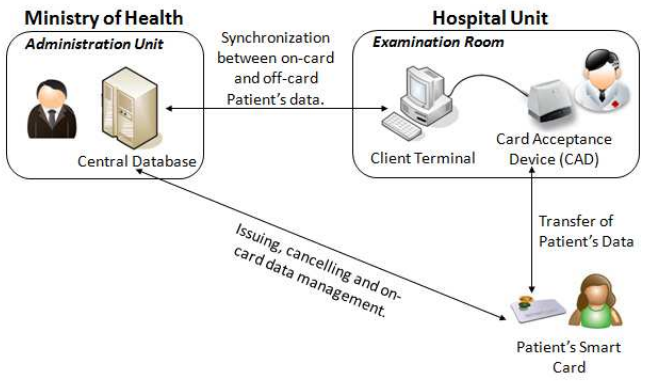

Ministry of Health
Administration Unit
Central Database
Synchronization between on-card and off-card Patient's data.
Hospital Unit
Examination Room
Client Terminal
Card Acceptance Device (CAD)
Transfer of Patient's Data
Issuing, cancelling and on-card data management.
Patient's Smart Card

Fig.1 - Smart Card Healthcare Model Architecture.

The model introduces a hybrid solution in which patient's smart card is used in the system depending on the availability of the network. The patient's smart card will be used as a medical data carrier which can be accessed without any need for network connection to access the central system. It is also used to access additional patient medical data stored in the central system located in the ministry of health if the network connection is available.

In the first option, the data stored in the patient smart card (on-card) can be accessed quickly by the healthcare professional through the client terminal without any need for network connection to access the central system. This data include the latest medical record of the patient with medical examinations and prescriptions, personal information, emergency contact information, allergies and current medications. By this way, the model supports the medical data portability even with off-line client terminals when there is no connection to the central system.

On the other hand, patient's smart card can be used by the healthcare professional through the client terminal to access the healthcare organization system in case the network is available. This allows the healthcare professional to reach the remaining patient medical data stored in the central system and make an update with the information related to the current visit. The central system will store the additional medical data of the patient which is already stored in the medical card. By including a copy of the patient's card data in the central system, the healthcare organization will be able to recover the patient medical data and re-issue another card in case of lost or damage of the patient's smart card.

The second component of the system is the client terminal located in several locations in the medical centers. These terminals will be used by healthcare professionals to access the medical data stored on the patient card using the smart card reader which is connected or embedded to each client terminal [12]. A secure channel is established between the client terminal and the patient smart inserted into the reader. The transmission of patient's medical data will take place between patient card and client terminal.

In on-line communication, the client terminal is connected to the central system and exchanging patient information. The central system use to synchronize via the client terminal with the patient medical card by addition new information related to the last examination and prescription done by the healthcare professional. While if the client terminal is off-line, the synchronization will be done in the next connection of the patient smart card. Each time the client terminal is connected to the central system, a synchronization process is started by the central system. The synchronization process will synchronize the central system with the patient medical card and record the new medical examination and the related prescription stored on smart medical card. Consequently this will insure the availability of the latest medical data in case of lost or damage of the patient medical card.

3.2 Patient smart card communication session

The software running on the client terminal allows the healthcare professional to communicate with the patient smart card and carry out the examination. The communication between the client terminal and the patient's smart card is shown in Figure.2.

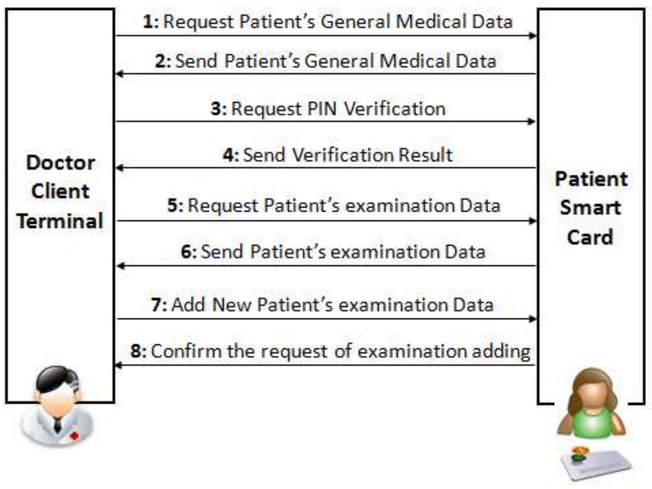

Doctor Client Terminal
Patient Smart Card
1: Request Patient's General Medical Data
2: Send Patient's General Medical Data
3: Request PIN Verification
4: Send Verification Result
5: Request Patient's examination Data
6: Send Patient's examination Data
7: Add New Patient's examination Data
8: Confirm the request of examination adding

Fig.2 - Patient Smart Card Communication Session.

In general, communication can only be opened by a patient smart card. When a patient smart card is inserted in the reader, the healthcare professional can access the personal and emergency information of the patient directly without any verification. In order to access the latest examinations and prescriptions as well as the confidential information, the PIN is requested from the card owner. If the PIN is valid, the client terminal communicates with the patient smart card in order to exchange the latest examination information. The information exchange between the client terminal and the card concerns retrieving information from the card and updating the card with information related to the current visit. Once the examination done, healthcare professionals can add new medical information and prescription on the patient smart card.

In every step of patient smart card communication session, the patient should be located in the same location of the smart card reader. In addition, if any changes have been made during the session or the visit, and the client terminal is in on-line state with the central system, a synchronization request will be automatically issued through the client terminal and sent to the central system. The purpose of this request is to synchronize the central system with the new patient medical card by adding information related to the current visit. In this case, the complete patient medical data will be available in the central system and the two sides (patient medical card and central system) are synchronized. Otherwise, if the client terminal is in off-line state, the changes will be marked to be synchronized in the next session when the connection between the client terminal and the central system will be available. This design will allow the healthcare organization the possibility to make another copy of the patient medical card if this later is lost or damaged.

3.3 The clinic healthcare service (Client)

The clinic healthcare services provided through client terminals will communicate with the patient smart card through the connected smart card reader. This service allows the healthcare professional to carry out examinations and record all related data. Through any client terminal, the healthcare professional can directly access the client personal information, emergency contact, allergies and current medications if any. This service is useful in urgent situation, such as, when the patient is unconscious state.

Moreover, after verification process, which is one of the services as well, the healthcare professional can navigate through the latest medical record of the patient with medical examinations and corresponding prescription and add a new medical examinations and its related prescription information on patient smart card if needed. An additional service should be implemented in the on-line terminal is the synchronization service. The synchronization service synchronizes the patient medical data stored in the smart card to the central system with the new patient medical data already stored in the card.

A ministry unit is responsible for carrying out the system administration functions through their terminals. Mainly, that unit is responsible to manage the central system which holds patients medical records. This unit is also responsible to record the patient's information into the system and to perform the related operations like issuing and cancelling smart card for patients. A synchronization service is integrated in the system that will allow having two data supports close to each other. Due their capacity limitation, we are storing examination information related to only the ten last visits. To be able to see the other visit information, the client terminal should be on line mode with the central system in order to retrieve needed information not available in the patient card. The ministry healthcare organization can recover the patient medical data and re-issue a new patient smart card in case of loss or damage.

3.4 The ministry healthcare service

A ministry unit is responsible for carrying out the system administration functions through their terminals. Mainly, that unit is responsible to manage the central system which holds patients medical records. This unit is also responsible to record the patient's information into the system and to perform the related operations like issuing and cancelling smart card for patients. A synchronization service is integrated in the system that will allow having two data supports close to each other. Due their capacity limitation, we are storing examination information related to only the ten last visits. To be able to see the other visit information, the client terminal should be on line mode with the central system in order to retrieve needed information not available in the patient card. The ministry healthcare organization can recover the patient medical data and re-issue a new patient smart card in case of loss or damage.

4 Implementation of an Electronic Healthcare System

The proposed smart card healthcare model has been implemented to validate our approach [13]. Our motivation in using this technology is the capability of the smart card to store, protect and manipulate medical data in a secure way. MULTOS Application Developer Smart Card has been chosen to implement the patient's smart card. MULTOS is an open multi-applications operating system that is ideal for the security need of our application [14, 15]. The advantage of this system is that many applications can run on the same card. The MULTOS card was used as patient's smart card to store the required amount of patient medical data.

The two applications developed are the application loaded into the smart card and the doctor's client terminal application. The data communication between the patient's card and the doctor's client terminal can start once the card is inserted into the reader. This will allow the doctor's client terminal to have access to the patient medical data and to update the card with the information related to the current visit. The patient's smart card application is responsible to handle the stored medical data on the smart card, while the doctor's client terminal application is an interface connected with a smart card reader. The information exchange between the doctor's client terminal and the central system will be done in a secure way using keys stored in the card. Confidential information is transmitted to the central system using an RSA algorithm. The content of the message is encrypted by the RSA function using the public key of the healthcare organization stored in the card. The same message is decrypted using the RSA function using the healthcare organization private key. Some confidential information is not accessible only if the patient enters the PIN card. Also, updating the medical card with data related to the current visit by the doctor's client terminal is done with the authorization of the patient by entering the PIN. The client terminal application will interact with the smart card application by sending and receiving APDU (Application Protocol Data Unit) to exchange the medical data between the client terminal and the smart card in a secure way.

The patient's smart card application has a hierarchical modular structure. It consists of three levels, where each level is responsible to handle specific part of the data, as shown in Figure.3. The *HealthAPDU* module is the first level of abstraction. It defines the external interface with the smart card. It is responsible to handle all communications involving APDU operations that can be executed by the card. The *HealthATL* module introduces a second level of abstraction that can communicates with the low level module (*HealthAPDU*) and the high level module (*SCHS GUI*). The *HealthATL* module implements functions like *OpenSession* that call the *HealthAPDU* commands: *allocOpenCard, establishContext, connectToCard* and *selectFileByAid.* These APDU commands allocate the card structure, establishes the resource manager context with the card if no session is opened with the card, connect to the card if the smart card is inserted in the reader, and select the healthcare application. If the command is not executed as expected, these functions should return a specific error code. This error code is interpreted by the *HealthATL* module and a specific error message is displayed.

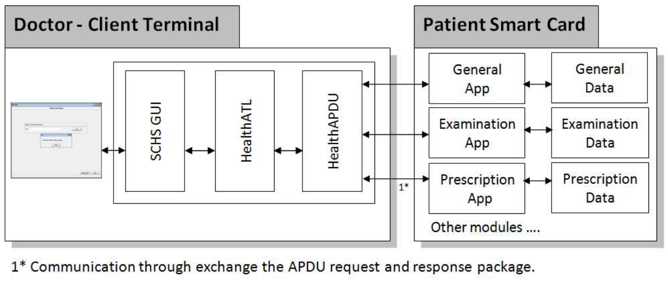


Fig.3 - Architecture of the Smart Card Healthcare System.

Patient's smart card application has three modules. Each module stores and processes a group of medical data. These modules can be accessed by client terminal to exchange the medical data. Each smart card module has a buffer for more than 200 characters to store a group of a related medical data into the smart card and defines a specific instruction to process these data. For example, Prescription Application which when loaded into the smart card is responsible for all patients' prescription data. These applications contains the related functions such retrieving and sending these data to the client terminal application or writing a new prescription on the patient's smart card.

The client terminal communicates and exchanges medical data with the patient's smart card module. It is implemented through three levels. Each level groups the related commands to allow communication with the smart card reader. Running commands is performed by sending APDU buffer to the smart card. These APDU commands read the responses sent from the smart card that specifies the execution status. In case of failure, the patient's smart card should return an error code. These three levels have been depicted in Figure.3.

The real communication with the smart card is performed in the lowest level of the client terminal. The communication with the card include connecting to the card, selecting the related application, sending APDU command to read or write medical data in the related module, and finally disconnecting from the card. The middle level module of the client terminal (*HealthATL*) interacts with both the lowest and highest level. The *HealthATL* module is responsible for preparing and manipulating patient's data before sending them to the next level (HealthAPDU). The highest level of client terminal application (*SCHS GUI*) handles the graphical user interface operations. The SCHS GUI takes the entered patient's data related to the current visit and sends them to the *HealthATL* module to be prepared for the *HealthAPDU* module. The *HealthAPDU* module sends the command through an APDU buffer to the smart card and the received response or the error code to the *HealthATL*. Once received, the *SCHS GUI* displays the patient's data requested, or a message that confirm the update, or a message error explaining the problem.

The client terminal system has a modular structure and has been designed within three levels. Each level contains one specific module. The patient's smart card application also has been divided over several modules and data groups as show in the figure.3. This architecture will facilitate modification, because each group of the medical data is stored in a separate module. The change will affect only in one module. Also, if we need to implement another application for the healthcare professionals for example, we can use modules like *HealthAPDU* to communicate with the smart card. Some of the client terminal applications will interact only with a specific group of the medical data and prevented from the interaction with other data. Thus, this modular structure offers data protection and ensures that the client terminal application interacts with the required module only. The pharmacy client application will interact with prescription smart card module only. An administrative personalization tool was developed to load experimental data, keys and certificates.

5 Conclusion

The smart card healthcare model was developed to allow each patient to carry a secure medical record in a smart card and use it in the authentication processes and emergency cases. This model aims to improve the quality of smart card services in the healthcare sector in the region. Additionally, the smart card services in healthcare sector will be enhanced with the increase capacity and lower costs of the smart cards with a higher capacity. Smart card would have the capability of storing extra medical information such as x-ray images.

The smart card healthcare model can be extended to integrate healthcare professional's data. These cards will store the healthcare professional personal information needed by the medical centers. Consequently, each healthcare professional can be authenticated using their own card and will able to access the patient's medical data with the patient's card. In the proposed model, the healthcare professionals can retrieve data and update the smart card only if the patient enters his PIN. In the next step, the healthcare professional will not be allowed to update the card or to execute any transaction if he doesn't insert his card first and enter his PIN. In this case, each transaction data present in the smart card is allowed and identified.

Furthermore, pharmacies and insurance companies can be integrated in the model to process the electronic prescriptions currently stored in smart cards. The pharmacies can proceed the prescriptions only when they receive an approval from the insurance companies. Hence, that will provide a paperless communication between hospitals, pharmacies and insurance companies.

The insurances companies should adhere to the ministry health organization in order to trust the reimbursement request coming from the pharmacies. The healthcare organization will play the trust party of the pharmacy and the insurance company. The healthcare organization will permit the reimbursement transaction. Due to the modular architecture, those modifications can be added in the proposed healthcare model. The government healthcare organization can also revoke the certificate for a healthcare professional. This alternative was not included in the developed prototype. In this case, the healthcare organization will reject the transaction request made by the healthcare professionals.

6 References

[1] Kardas G. and Tunali E. T. ”Design and implementation of a smart card based healthcare information system”. Computer Methods Programs Biomedicine, 81, 1 66-78, Jan. 2006.

[2] Moon D., Chung Y., Pan S. B., and Park J. ”Integrating fingerprint verification into the smart card-based healthcare information system”. EURASIP J. Adv. Signal Process, 5-5, Jan. 2009. DOI= http://dx.doi.org/10.1155/2009/845893

[3] Mayes K., Markantonakis K.. ”Smart Cards, Tokens, Security and Applications”, Springer, 2008.

[4] A. Alkhateeb, T. Takahashi, S. Mandil, and Y. Sekita. ”The changing role of health care IC card systems”. Computer Methods & Programs in Biomedicine, 60, 2, 83-92, 1999.

[5] M. Marschollek and E. Demirbilek. “Providing longitudinal health care information with the new German Health Card - a pilot system to track patient pathways”. Computer Methods & Programs in Biomedicine, 81, 3, 266-271, 2006.

[6] Savostyanova N. and Velichko V. ”Plastic card fraud: a survey of current relevant card and system properties”, 2004.

[7] Bernd Blodel and Peter Pharow. ”A model driven approach for the German health telematics architectural framework and security infrastructure”. International Journal of Medical Informatics, 76, 169-175, 2007.

[8] Rankl W. and Effing W. ”Smart card handbook”, Wiley & Sons, 2003.

[9] Alvin T.S. Chan, Jiannong Cao, Henry Chan, and Gilbert Young. ”A Web-Enabled FRAMEWORK for SMART CARD Application in Health Services”. COMMUNICATIONS OF THE ACM, 44, 9, 77-82, September 2001.

[10] Won Jay SONG, Byung Ha AHN and Won Hee KIM. ”Healthcare Information Systems Using Digital Signature and Synchronized Smart Cards via the Internet”. Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC’02) IEEE Computer Society.

[11] Walid ATTIAOUI, Pr. Mohamed BEN AHMED, Pr. Moncef TAGINA and Dr. Boutheina CHETALI. ”Integrating USB Smart Card with Flash Memory to Web based Medical Information Systems: Application for the pathology of cancer”, IEEE Explore, 971-977, 2006.

[12] Rodrigues R., Piccolo U., Hernandez A. and Oliveri N. ”Integrated circuit Health data Cards”, Pan American Health Organization, 2003.

[13] Ebtisam AlAbdulqader. ”Smart Card in Healthcare Systems”. Project Report submitted for the degree of Master in Information Systems, College of Computer and Information Sciences, King Saud University, June 2009.

[14] ”Multos development tools and manuals”. Available at: http://www.multos.com/

[15] Hendry M.” Multi-application Smart Cards: Technology and Applications”. Cambridge University, 2007.

Design and Implementation of SOC and BIST based Wave-Pipelined Circuit

Rengaprabhu Paramasivam[$], Venkatasubramanian Adinarayanan[*], Parasuraman Sakkarapani[@], Marimuthu Murugesan[^], Vivek Karthik Perumal[%] and Seetharaman Gopalakrishnan[#]

[$]Research Scholar, Dept. of Info. & Comm. Engg, Anna University of Technology, Trichy, India

[*]Research Scholar, Department of CSE, Sathayabamma University, Chennai, India

[@^%]Final year Students, Department of ECE, Oxford Engineering College, Trichy, India

[#]Principal, Oxford Engineering College, Trichy, India

***Abstract*— Wave-pipelining is proposed as one of the techniques for the design of high speed digital circuits in the literature. Higher operating frequencies can be achieved in wave-pipelined digital circuits, by adjusting the clock periods and clock skews so as to latch the outputs of combinational logic circuits at the stable periods. Major contribution of this paper is the proposal for automation of the above tasks, one using the Built In Self Test (BIST) approach and another using the System-On-Chip (SOC) approach. Both of these approaches is studied by a multiplier using dedicated AND gate by adopting three different schemes: wave-pipelining, pipelining and non-pipelining. BIST approach is used for the implementation on Xilinx Spartan-II device. The SOC approach is adopted for implementation on Altera Field Programmable Gate Arrays (FPGAs) based SOC kits with Nios II softcore processor. From the implementation results, it is verified that the wave-pipelined circuits are faster compared to non-pipelined circuits. The automation scheme has considerably reduced the effort required for the design and testing of the wave-pipelined circuits. The pipelined circuits are found to be faster than the wave-pipelined circuits and this is achieved at the cost of increase in area.**

Keywords: SOC; wave-pipelining; FPGA; self-testing.

1. Introduction

Hardware components in a SOC may include one or more processors, memories and dedicated components for accelerating critical tasks and interfaces to various peripherals [1], [2]. In addition, the development of IP cores for the FPGAs for a variety of standard functions including processors, enables a multimillion gate FPGA to be configured to contain all the components of a platform based FPGA. Development tools such as the Altera System-On-Programmable Chip (SOPC) builder enable the integration of IP cores and the user designed custom blocks with the soft-core processors such as the Nios II processor [3]. The increased gate count in a complex SOC results in increased power dissipation, clock routing complexity and clock skews between different parts of a synchronous system. These limitations may be partially overcome by adoption of circuit design techniques such as wave-pipelining. Wave-pipelining enables a combinational logic circuit to be operated at a higher frequency without the use of registers and may result in lower power dissipation and clock routing complexity compared to a pipelined circuit. However, the maximization of the operating speed of the wave-pipelined circuit requires the following three tasks: adjustment of the clock period, clock skew (δ) and equalization of path delays. The automation of these three tasks are proposed for the first time in this paper. Effectiveness of the automation scheme is studied by a multiplier using dedicated AND gates as well as fast carry logic.

The organization of the rest of the paper is as follows: In section II, the previous work related to wave-pipelining and the challenges involved in the design of wave-pipelined circuits are described. In section III, automation schemes for wave-pipelined circuits are presented. In section IV, an overview of the multiplier using dedicated AND gate as well as fast carry logic is presented. In section V, BIST approaches for the implementation of wave-pipelined circuits are discussed and the implementation results are presented. In section VI, SOC approaches for the implementation of wave-pipelined circuits are discussed and the implementation results are presented. Section VII summarizes the conclusions.

2. Review of previous work

Pipelining achieves high speeds in digital circuits at the cost of increased area, power dissipation and routing complexity [4]. Wave-pipelining is proposed as one of the techniques for achieving high speed without the above limitations. Wave-pipelining has been employed for implementing a number of systems on both ASICs and FPGAs [5], [6]. The concept of wave-pipelining has been described in a number of previous works [7], [8]. To illustrate this concept, graphical representation of the data flow through combinational logic circuit, is used [8]. Fig. 1 shows a typical combinational logic circuit along with the input and output registers [8]. Fig. 2 depicts the flow of data through the above circuit [8]. The skew between the clocks at the input and output registers is denoted as δ. At the beginning of each clock cycle, data is initiated into the combinational logic block through the input register.

A number of paths may exist between the inputs and output of a logic block. A change in the input causes the output to change after a delay of D_{min}, D_{max} through the shortest and longest path respectively. The shaded regions bounded by (D_{min} and D_{max}) depict the periods where the logic levels of the logic block vary with time. The nonshaded areas depict the stable duration of the logic block. In the conventional system, the output register is clocked in the nonshaded region and the minimum clock period, T_{clk} is chosen to be greater than $D_{max.}$ In the Wave-Pipelined (WP) system, the clock period is chosen to be (D_{max} - D_{min}) + clocking overheads such as set up time, hold time etc.

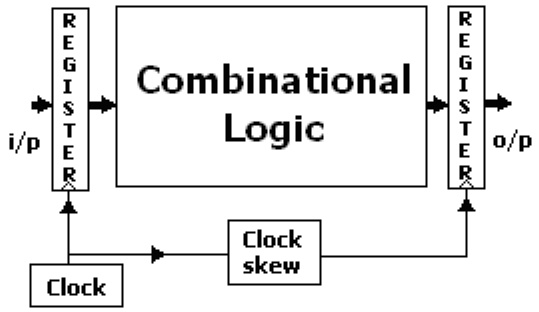


Fig. 1. A combinational logic circuit with input and output registers.

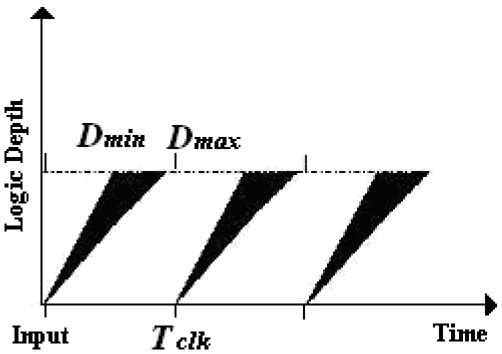


Fig. 2. Temporal/spatial diagram of data flow through the combinational logic circuit.

To ensure correct operation, δ should be adjusted so that the active clock edge occurs in the stable period. As the shaded region increases with increase in the logic depth, the operating clock frequency should be reduced with increase in logic depth. Moreover, to maximize the frequency of operation of the wave-pipelined system, the difference (D_{max} - D_{min}) is minimized by equalizing the path delays. Hence, adjustment of the clock period, clock skew (δ) and equalization of path delays, are the three tasks required for maximizing the operating speed of the wave-pipelined circuit. All the three tasks require the delays to be measured and altered if required. Layout editors, such as FPGA editor from Xilinx, may be used for this purpose.

These tasks are carried out manually in [9], [10]. The wave-pipelined circuit designed using the layout editor may be tested using simulation. However, the simulation is inadequate for testing due to the difference between the actual delays and the delays calculated by the layout editor. This is because, the layout editor considers only the worst case delays and the actual delays may be significantly different due to fabrication variations. This difference becomes important as the logic depth of the circuit increases. Hence, the design is downloaded to the actual FPGA and its operation is checked using a PC based test system [10]. If correct results are not obtained, delays are altered and the design is downloaded for testing again. A number of iterations of place and route, simulation, downloading and testing in the actual device may be required till the correct results are obtained. The design of wave-pipelined circuit in this fashion requires human intervention and is time consuming. Automation of the above three tasks are considered in the next section.

3. Automation schemes for wave-pipelined circuits

Equalization of the path delays is considered first. This cannot be completely automated as the commercially available syntheses tools do not support the specification of interconnect delays. However, the difference in path delays can be minimized by specifying the physical location of logic cells (slices) or logic elements used for the implementation, through either the User Constraints File (UCF) or the Logic lock feature supported by the FPGA CAD tools. UCF approach is proposed for Xilinx FPGAs in [10], [11]. The logic lock feature is adopted for the Altera FPGAs in this paper.

The adjustment of the clock skew and clock period can be automated by adopting programmability. The programmable clock and clock skew generator may be implemented in the FPGAs. Fig. 4 gives the circuit diagram of a clock generation scheme which consists of a delay block and an inverter. The actual clock period depends on the interconnect delay. The select input of the multiplexer is varied with either a processor or a Finite State Machine (FSM) to achieve different clock frequencies. Similarly, for the clock skew generator, the same circuit is used, but the feedback connection is removed and the select line is varied through processor or FSM to achieve different clock skew ranges.

The wave-pipelined circuit using the programmable clock and skew generator can be operated at a higher frequency than that can be achieved using the commercially available synthesis tools which use D_{max} for fixing the operating frequency. The automation may be carried out using either off-chip processor or on-chip processor. The off-chip processor is used when the FPGA is used as a coprocessor or hardware accelerator for a main processor or microcontroller.

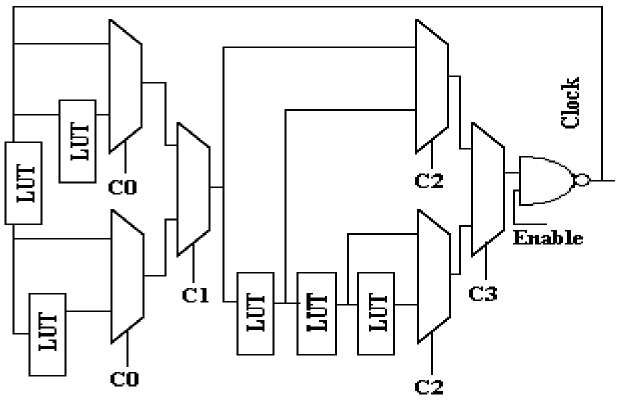


Fig. 3. Programmable clock generator.

Since off-chip communication between the FPGA and a processor is bound to be slower than on-chip communication, in order to minimize the time required for adjustment of the parameters of the wave-pipelined circuit (clock frequency and skew), the built in self test approach using design for testability [12] technique, is proposed for this case. In the SOC approach, a processor is assumed to be available on-chip and it is used for adjustment of the parameters of the wave-pipelined circuit.

3.1 BIST Approach for Wave-Pipelined Circuit

Testing a large chip requires a large test sequence and application of these test sequences to the Circuit Under Test (CUT) using external testers is time consuming. Built in self test scheme is an alternative for minimizing the testing time. In the BIST scheme, the test sequences are internally generated, applied to the CUT at full speed and a signature is generated for finding whether it is good or bad. The block diagram of a wave-pipelined circuit with BIST is given in Fig. 4. This is obtained by including the FSM block and self-test circuit. The self-test circuit contains programmable clock, clock skew generator, signature analyzer and test vector RAM to the circuit given in Fig. 4.

3.1.1 FSM Block

The Finite State Machine (FSM) is realized using the off-chip processor. The FSM block generates the control signal to choose between the normal mode and the self test mode and this is applied to the select input of multiplexer. In the self test mode, the FSM systematically varies the clock skews and clock periods. For each clock frequency and skew, the self test circuit generates the test inputs, applies them, generates the signature, compares it with the expected result and finally generates a flag indicating the match. The FSM progresses with the testing till the frequency at which the DUT works for at least 3 or more skew values is found. The operating skew value is chosen to be the middle value so that the DUT would reliably work even if the delays change due to environmental conditions. In order to minimize the time required to determine the correct value of clock skew and clock period, a two step procedure is adopted. The clock frequencies are varied by large steps to determine the range of frequency in which the circuit works. This is achieved by varying only the higher order two bits of the select inputs of the programmable clock. After the range is determined, fine tuning is achieved by varying the lower order bits. For every frequency at which the circuit is tested, the clock skews are varied gradually and the results are tested for its correctness and the clock skews for which the circuit works satisfactorily is noted. The testing time can be minimized by using the optimal test vector set and a signature analyzer [10].

3.1.2 Signature Generator

For testing the correctness of the circuit, N test vectors may be fed one after another and the N outputs obtained should be compared with the expected outputs. In order to minimize the number of comparisons, a unique signature is generated out of the N outputs and it is compared with the signature corresponding to the expected outputs. The signature generator consists of a Pseudo Random Binary Sequence (PRBS) generator with multiple data input [Smith 2003] as shown in Fig. 5. The successive output of the output register is XOR'ed with the state of the PRBS to generate the next state.

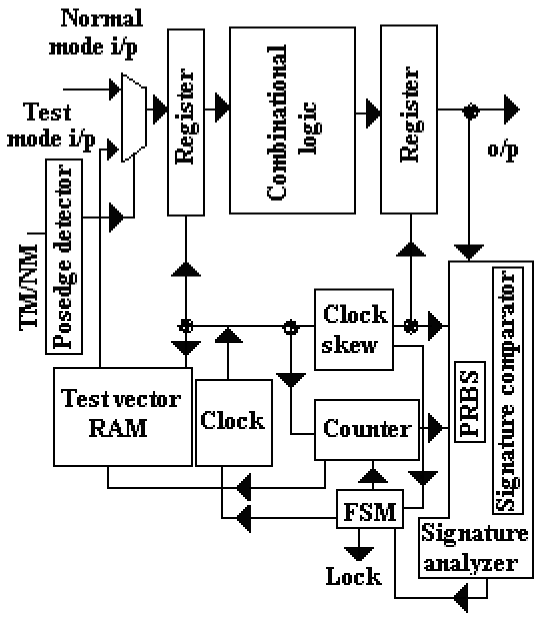


Fig. 4. Self tuned wave-pipelined circuit.

If the test vector set consists of N vectors, the PRBS generator output contains the signature after application of N clock pulses. However, due to the propagation delay in the Random Access Memory (RAM), I/O registers and the combinational logic block, the time at which signature generation begins should be delayed with respect to the time at which the application of test vectors begins. The delay depends on the depth of the combinational logic blocks.

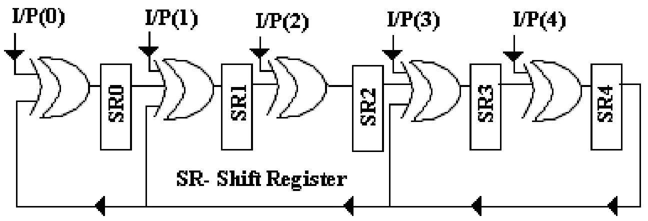


Fig. 5. Signature Generator.

3.1.3 Test Vector Generation

In principle, the number of test vectors required for an M input combinational logic circuit is 2^M. If the value of M is small, exhaustive testing of the circuit may be carried out by generating the test inputs through an M bit counter and checking the signature after the counter completes one full cycle. However, some of the inputs may contribute more to D_{max} than the others. For example, in the case of the multipliers, the maximum propagation delay occurs only when MSBs of the operands are 1. If the multiplier works for this case, it will work for the other cases where at least one of the MSBs is zero. Hence, a (M-2) bit counter is adequate for testing. For circuits with large number of inputs, exhaustive testing would require very large testing time. Minimal test vector set, which reduces the testing time without compromising the quality of detection of faults, may be obtained using the automatic test pattern generator (ATPG) algorithms [Smith 2003]. Computed Aided Design tools may also be used for generating the minimal test vectors using ATPG algorithm and assessing their fault coverage ratio. However, the generation of test patterns for wave-pipelined circuit is non trivial because we have to account for data dependent delays (delay for 001 is different from that for 101) [Thomas Gray et al 1993] and this is compounded by the absence of accurate models for interconnects in FPGAs. Since the conventional ATPG techniques are not applicable for wave-pipelined circuits, we have to content with only random test vectors. By choosing different test vector sets consisting of different combinations and different ordering of test vectors, we can improve the confidence level.

The BIST approach requires a number of overheads such as FSM, signature generator and test vector RAM. These blocks are useful only when the clock frequency and skew are to be varied. If the operating frequency is chosen so that the stable period in Fig. 2 is greater by at least twice the worst case variation in the delay due to temperature, neither the clock frequency nor the skew need to be adjusted again. After these initial selection, the wave-pipelined circuit requires no further tuning and work satisfactorily without any external intervention. Instead of using a dedicated circuit such as BIST, a processor may be used to carry out the above tuning task. For example, FPGA based speech recognition with SOC may perform the various tasks required by optimally partitioning between hardware and software [Amudha et al 2007]. The tasks performed in software uses the on-chip processor. The hardware block may use wave-pipelining and it may be tuned by the on-chip processor at the beginning.

For the SOC approach, PRBS generator, signature comparator blocks in Fig. 4 may be replaced by a block RAM which is used to store the outputs of the circuit under test corresponding to the test inputs. Since the communication interface between the on chip processor and the circuit under test is faster, the outputs can be directly read and compared with the expected output for every combination of skew and clock frequency. The algorithm for the FSM given in section 3.1 can be modified accordingly. The select inputs for the clock as well as skew blocks and the data inputs to the wave-pipelined circuit may be applied and varied through the on-chip processor.

A variety of choices exist for the implementation of SOC. The SOC may consist of a hard core processor such as power PC or ARM processor and an FPGA coprocessor or DSP block. Alternatively, it may consist of soft-core processors such as Nios II or Micro blaze and a custom DSP block implemented in FPGA. In this paper, Altera FPGA based SOC consisting of Nios II soft-core processor is used for the implementation. Fig. 6 shows the interface diagram of a Nios II processor along with the custom block (wave-pipelined circuit).

3.2.1 Procedure for Adjusting the Clock Period and Skew

In this paper, the multiplier block is implemented as custom hardware for the soft-core processor. The period of the clock signal and delay introduced by the clock skew blocks depend on the interconnect delays, the location of the logic elements and the interconnects used for the implementation of these blocks should be fixed so that when this block is integrated with the multiplier or the processor, the interconnect delays are not altered. This is achieved by using the Logic Lock feature in Altera.

The operating frequency of the wave-pipelined circuit is expected to lie between that of non-pipelined circuit and pipelined circuit. Hence, the minimum and maximum frequency of the clock generator should correspond to the maximum operating frequencies of the non-pipelined and pipelined circuits respectively. The approximate values of these two frequencies are found for the circuit to which the clock is to be applied using the synthesis report. After determining the range of the frequencies to be generated by the clock circuit, the number of delay blocks are adjusted.

4. Design of multiplier unit

The effectiveness of the approach proposed is studied by implementing self tuned wave-pipelined circuits with combinational logic block: multipliers using dedicated AND gate.

The implementation of multipliers is considered first. Xilinx FPGAs such as Spartan-II as well as Virtex devices and Altera FPGAs such as APEX and Cyclone II devices have fast carry logic and dedicated AND gate for each of the Look Up Tables (LUTs) in the Slices/Logic

Elements (LEs). Since multiplying an N bit number by 2 requires only AND gates and adders, fast Nx2 multipliers can be implemented using this dedicated hardware. To implement a Nx4 multiplier, output of two Nx2 multipliers has to be added.

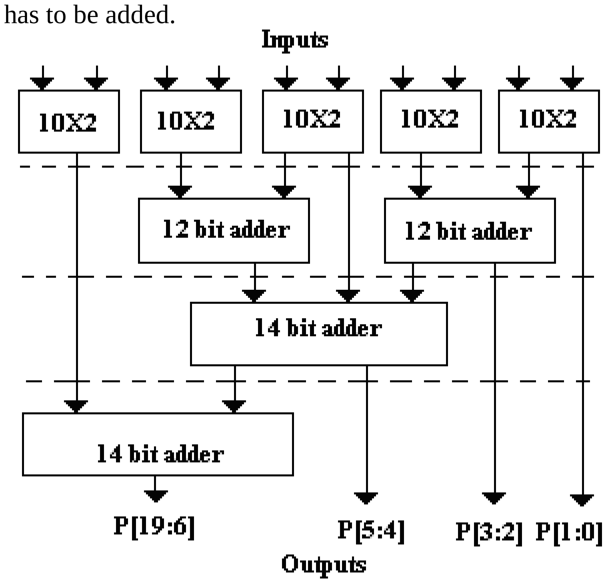


Fig. 9. 10X10 Multiplier dedicated AND gate and fast carry logic.

To implement an NxM multiplier, the output of ⌈log$_2$M⌉, Nx2 multipliers have to be added, 2 at a time in parallel in ⌈log$_2$M⌉ stages appropriately. The circuit diagram of the 10x10 multiplier using dedicated AND gate and fast carry logic is shown in Fig. 9. The dotted line indicates points where registers may be inserted for pipelining. For wave-pipelining all the stages are directly connected without registers. The registers are used only at the inputs and outputs.

5. Implementation of self tuned wave-pipelined circuits using BIST approach

The multipliers using dedicated AND gate is implemented on Xilinx Spartan II FPGA using BIST approach. It may be noted that the BIST approach is also applicable for Altera FPGAs.

5.1 Implementation results on Multiplier using Spartan-II XC2S100

The multipliers of size 4x4, 6x6 and 10X10 are implemented on Spartan-II XC2S100-5PQ208 device. The multipliers with and without pipelining are also implemented and the results are shown in Fig. 11. Overhead required for wave-pipelined circuits are also shown in Fig. 11.

From Fig. 11, it may be noted that the wave-pipelined multipliers are faster by a factor of 1.43-1.46 compared to the non-pipelined multipliers. The pipelined multipliers are faster by a factor of 1.16-1.23 compared to the wave-pipelined multiplier. This is achieved by increasing the number of registers by a factor of 1.75-4.1 and slices by a factor of 1.21-1.26.

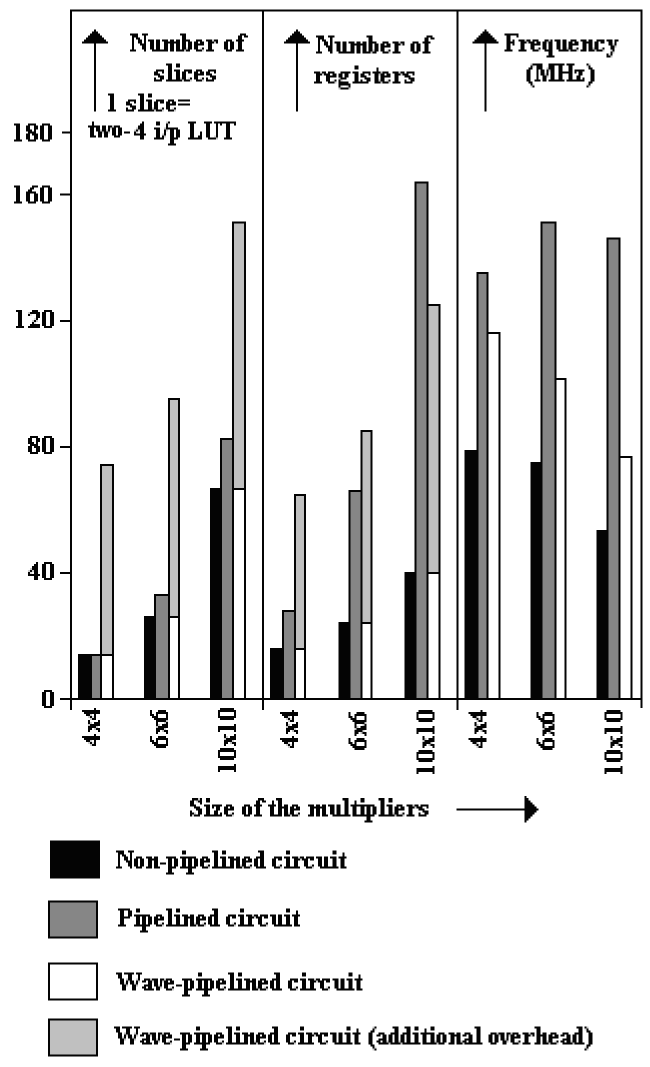


Fig. 11. Implementation results of multipliers using dedicated AND gate and fast carry logic.

5.2 Implementation results on Multiplier using ASIC

The self tuned wave-pipelined circuit in Fig. 4 is studied by using 10X10 multiplier as the combinational logic block. These circuits are implemented using 180nm technology in ASIC. Verilog HDL language is used to describe the functionality of the circuit and after the circuit is described in HDL, functionality is verified modelsim simulation tool. Leospectrum is used for synthesizing the circuit.

Table. 1. Implementation results of multipliers using dedicated AND gate and fast carry logic.

Schemes	Area	Freq.(Mhz)
Pipelining	1623	343.4
Non-pipelining	916	197.1
Wave-pipelining	1431	203.2

From table1, it may be noted that the wave-pipelined multipliers are faster by a factor of 1.03 compared to the non-pipelined multipliers. The pipelined multipliers are faster by a factor of 1.6 compared to the wave-pipelined multiplier.

6. Implementation of self tuned wave-pipelined circuits using SOC approach

For the SOC approach, the soft-core processor, Nios II, is implemented on Altera FPGAs and 12X12 multiplier is implemented as custom hardware.
The optimal clock period and clock skews are determined using the procedure described in section III.B. Since the period of the clock signal and delay introduced by the clock skew block depend on the interconnect delays, the location of the LEs and the interconnects used for the implementation of these blocks should be fixed so that when these blocks are integrated with either the multiplier or the Nios II processor, the interconnect delays are not altered. This is achieved by using the Logic lock feature in Quartus II.

As mentioned in section III, simulation is inadequate to test the wave-pipelined circuit. Hence, this circuit is implemented along with the Nios II softcore processor and the former is added as the custom block to the Nios II using SOPC builder. The program to be executed by the Nios II is written in C/C++ and the custom block is invoked as a function in the C/C++ program. A C++ program is written to read and write from the block RAM in the custom block.

6.1 Implementation results on Multiplier using Apex 20K200EFC484

The results obtained for the 12X12 wave-pipelined multiplier using dedicated AND gate and fast carry logic are shown in Fig. 13. The multipliers with and without the pipelining are also implemented and these results are also shown in Fig. 13. Overhead required for wave-pipelined circuit is also shown in Fig. 13. From Fig. 13, it may be noted that the wave-pipelined multiplier is faster by a factor of 1.5 compared to the non-pipelined multiplier. The pipelined multiplier is faster by a factor of 2.02 compared to the wave-pipelined multiplier. This is achieved with increase in the no. of registers by a factor of 4.17 and LEs by a factor of 1.05. Quartus II does not have the power estimation feature for Altera Apex devices and hence powers dissipated by the different multiplier schemes are not compared.

7. Conclusion

Two automation schemes proposed in this paper for the FPGA implementation of the wave-pipelined circuit is tested using the multipliers with dedicated AND gate as well as fast carry logic. It is observed that wave-pipelined circuits operate faster by a factor of 2.6 compared to non-pipelined circuits. The pipelined circuits are in turn faster than the wave-pipelined circuits and this is achieved with the increase in the number of registers and LEs or slices. Out of the two automation approaches, SOC approach requires less overhead compared to BIST approach.

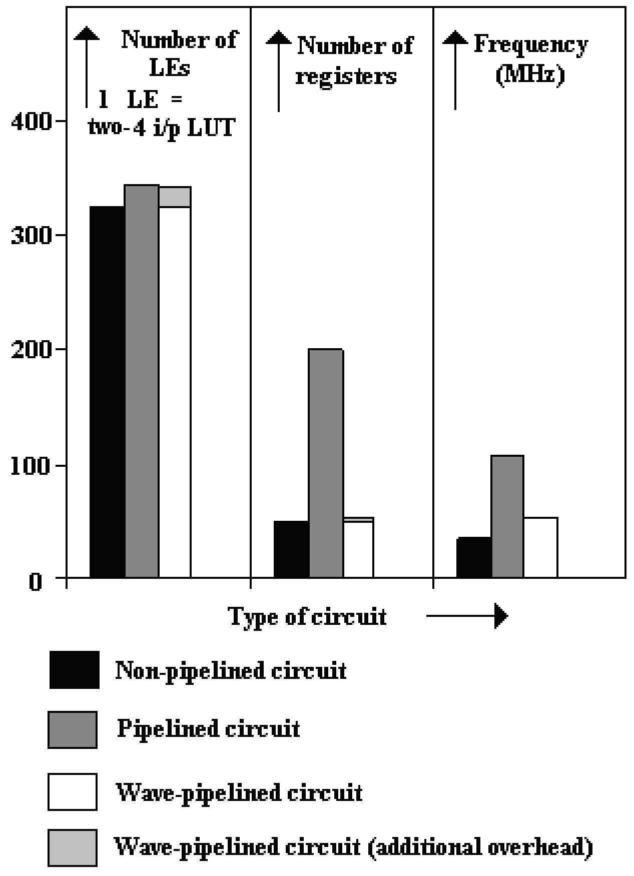


Fig. 13. Implementation results of 12X12 multiplier using dedicated AND gate fast carry logic.

References

1. Flavio R. Wagner, Wander O. Cesario, Luigi Carro and Ahmed A. Jerraya, "Strategies for the integration of hardware and software IP components in embedded systems-on-chip," *Elsevier Integration, The VLSI Journal*, pp. 1-31, Nov. 2003.
2. G. Martin and H. Chang, "System-on-Chip design," *Proc. of Intl. conf. on ASIC*, pp. 12 – 17, 2001.
3. Altera documentation library- 2003, Altera corporation, USA.
4. K. K. Parhi, "VLSI signal processing systems," *John Wiley & Sons*, 1999.
5. J. Nyathi and J. G. Delgado-Frias, "A hybrid wave pipelined network router," *IEEE Transactions on Circuits and Systems I*: Fundamental Theory and Applications, vol. 49, no. 12, pp. 1764 –1772, Dec. 2002.
6. O. Hauck., A. Katoch and S. A. Huss, "VLSI system design using asynchronous wave pipelines: a 0.35 µm CMOS 1.5 GHz elliptic curve public key cryptosystem chip," *Proc. of Sixth Intl. Symposium on Advanced*

Research in Asynchronous Circuits and Systems, 2000, (ASYNC 2000), pp. 188 –197, April 2000.

7. W. P. Burleson, M. Ciesielski, F. Klass, and Liu, "Wave-pipelining: a tutorial and research survey," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. 6, no. 3, pp. 464 –474, Sep.1998.
8. C. Thomas Gray, W. Liu and R. Cavin, "Wave Pipelining: Theory and Implementation," *Kluwer Academic Publishers*, 1993.
9. E. I. Boemo, S. Lopez-Buedo and J. M. Meneses, "Wave pipelines via look-up tables," *IEEE International Symposium on Circuits and Systems ISCAS '96*, vol. 4, pp. 185 -188, 1996.
10. G. Lakshminarayanan and B. Venkataramani, "Optimization techniques for FPGA based wave-pipelined DSP blocks," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 13, no. 7, pp 783-793, July 2005.
11. Xilinx documentation library, Xilinx Corporation, USA.
12. M. J. S. Smith, "Application Specific Integrated Circuits," *Pearson Education Asia Pvt. Ltd,* Singapore, 2003.
13. G. Seetharaman, B.Venkataramani and G. Lakshminarayanan, "Design and FPGA implementation of self-tuned wave-pipelined filters," *IETE journal of research*, vol 52, no. 4, pp. 305-313, July-August 2006.
14. V. Amudha, B. Venkataramani, R. Vinoth Kumar and S. Ravishankar, "SOC Implementation of HMM Based Speaker Independent Isolated Digit Recognition System," *20th International IEEE Conference on VLSI Design (VLSID'07)*, pp. 1-6, 2007.
15. G. Seetharaman, B.Venkataramani, V. Amudha, Anurag Saundattikar, "System on chip implementation of 2D DWT using lifting scheme," *Proc. of the International Asia and South Pacific Conference on Embedded SOCs (ASPICES 2005),* July 5-8, 2005, Bangalore.
16. U. Meyer Baese, "Digital Signal Processing with FPGAs," *Springer Verlag*, 2001.

System on Chip implementation of wave-pipelined 2D DWT

Venkatasubramanian Adinarayanan[*], Rengaprabhu Paramasivam[$] and Seetharaman Gopalakrishnan[#]
[*]Research Scholar, Department of CSE, Sathayabamma University, Chennai, India
[$]Research Scholar, Dept. of Info. and Comm. Engineering, Anna University of Technology, Tiruchirappalli, India
[#]Principal, Oxford Engineering College, Tiruchirappalli, India

Abstract

This paper presents the design and implementation of hybrid wave-pipelined 2D DWT using lifting scheme. In this approach different lifting blocks are interconnected using pipelining and the individual blocks are implemented using Wave-Pipelining (WP). For the purpose of comparison, non pipelined scheme as well as the scheme with pipelining within the blocks and between the blocks is implemented. For images of size 512×512, one level 2D DWT scheme is implemented on Xilinx based SOC Kits, using all the three schemes. From the implementation results, it is concluded that the hybrid WP is faster than non-pipelined and requires less area, less clock routing complexity and lower power than pipelined. The one level 2D DWT scheme is also implemented, in ASICs using pipelining. This is first of its kind.

Key-Words: FPGA, SOC, ASIC, DWT, lifting.

1. Introduction

Programmable logic devices such as Field Programmable Gate arrays (FPGAs) offer an alternative solution for the computationally intensive functions performed traditionally by Programmable Digital Signal Processors (P-DSP). They are also employed at the radio frequencies as the digital down converter. [1]. The ability to design, fabricate and test ICs with gates count of the order of a few tens of million has led to the development of complex embedded System On-Chip (SOC) with FPGAs and ASICs. Development tools such as the Altera SOPC builder and the Xilinx EDK enable the integration of IP cores and the user designed custom blocks with the soft-core processors such as the NiosII from Altera and Micro blaze from Xilinx [2], [3], [4].

The VLSI implementation of image encoders with 2D DWT has been addressed in number of previous works. In Ritter [5], FPGA (XC4085XLA) based implementation of 2D DWT using lifting scheme and compression using EZT algorithm is reported. For storing the image and the transform an external RAM is used in this work. The image block encoder proposed in [6], uses block RAMs in FPGAs for storing the sub image and its transform. As 2D DWT is not a block transform, the image is split into overlapping blocks of sub images for the 2D DWT computation. For the implementation of the 2D DWT, lifting scheme proposed by Swelden [7] is quite suited. However, pipelined systems require more power dissipation, clock routing complexity and result in larger clock skews between different parts of the system. A circuit design technique such as wave-pipelining is one of the techniques for achieving high speed without the above limitations. A number of systems have been implemented using wave-pipelining on ASICs and FPGAs [9], [10]. The concept of wave-pipelining has been described in a number of previous works [11], [12], [13]. Wave-pipelined circuit dispenses with the need for registers for storing the intermediate results and instead uses the inherent capacitance at the input to the various combinatorial blocks. This results in lower power dissipation at the cost of speed. The hybrid scheme is aimed at combining the advantages of both pipelining and wave pipelining.

II. Review of previous work on 2D DWT

2D wavelet transform may be computed using filter banks. Fig .1 shows One level 2D DWT, x[n] shows the input image, LL1 shows the subset of the transform coefficients represents the coarse form of the input image. The input samples x(n) are passed through the 2 stages of analysis filters as shown in Fig .1. They are first processed by the low pass h[n] and high pass g[n] horizontal filters and are sub sampled by two. Subsequently, the outputs (L1, H1) are processed by low pass and high pass vertical filters.

The horizontal and vertical filters contains 5 lifting blocks (α, β, γ, δ, ξ). The lifting scheme uses a poly-phase structure for the analysis filter [7]. For the two level 2D DWT the input is LL1 component and for further decomposition the same procedure is followed. For every level the image gets reduced by a factor of four. In the lifting scheme, the odd and even input samples are processed by the lifting blocks (α, β, γ, δ, ξ

(ξ_1 & ξ_2)) in cascade as shown in Fig. 1a. ξ_1, ξ_2 are scaling blocks. Details of α and β blocks are shown in Fig. 2a and Fig. 2b. γ and δ blocks are obtained by replacing the constants α, β with γ, δ.

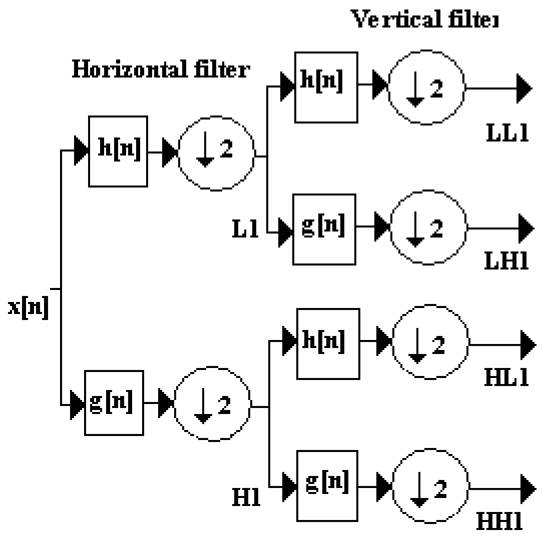


Fig.1 One level 2D DWT

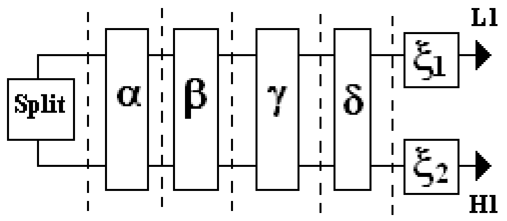


Fig.1a Simplified block diagram of Lifting Scheme for 9/7 filter.

In Fig. 2, since the output from one block is fed as the input to the next block, the maximum rate at which the input can be fed to the system depends on the sum of the delays in all the four stages. The speed is increased by introducing pipelining at the points indicated by dotted lines in Fig. 2 [b]. In this case, the input rate is determined by the largest delay among all the four blocks. The delay in the individual stages is reduced further by using Constant Coefficient Multiplier (KCM). KCM uses a ROM for finding the product of a constant and a variable. The variable is fed as address to the ROM, which contains the products corresponding to all possible combinations of the operands. When the ROM is implemented using 4 input Look Up Tables (LUTs), a no. of stages of LUTs and adders are required to find the product. For example a 12x12 bit KCM requires one ROM stage consisting of three 16X16 ROMs and two stages of 16 bit adders. The speed of the KCM can be increased by introducing the pipelining registers at the outputs of ROMs and adders.

The Pipelined Constant Coefficient Multiplier (PKCM) using the BW content is referred to as BW-PKCM in [6] and is shown to be superior compared to the other approaches. Hence, only this multiplier is considered for wave pipelining in this paper. The detailed diagram of the α block implemented using BW-PKCM is shown in Fig. 3. The same scheme can be adopted for the β, γ, δ, $\xi_{1,}\xi_2$ blocks.

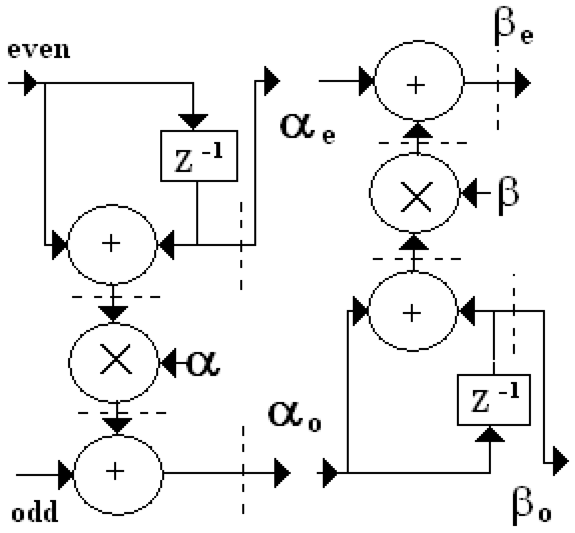


Fig. 2a α block Fig. 2b β block

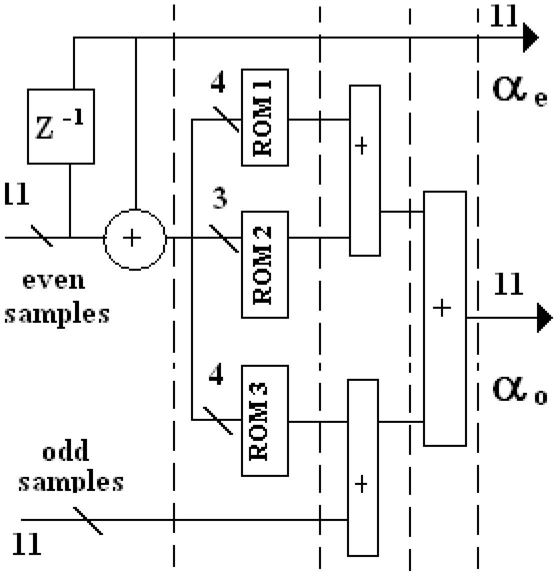


Fig.3 α block using BW-PKCM

III. Design of wave-pipelined lifting blocks on FPGAs

An RTL model of a circuit consists of a combinational logic circuit separated by the input and output registers. The combinational logic circuit may be considered to be a wave-pipelined circuit if a number of waves are made to simultaneously propagate through it is shown in Fig. 4a [14]. In other words, at any point of time, a sequence of data is processed in the combinational logic block. In the case of pipelining, only one data is processed in the combinational logic block at a time. Further, the maximum data rate in the pipelined circuit depends only on Dmax, the maximum propagation delay in the combinational logic block.

Fig. 4b shows temporal/spatial diagram of combinational logic circuits [14]. If Dmin denotes the minimum propagation delay of the signal through the combinational logic block, the maximum data rate of the wave-pipelined circuit depends on (Dmax – Dmin). Traditionally, in a wave-pipelined circuit, higher speeds are achieved by equalizing the Dmax and Dmin [9]. The output of the wave-pipelined circuit alternates between unstable and stable states. The stable period decreases with the increase in the logic depth. By adjusting the latching instant at the output register to lie in the stable period, the wave-pipelined circuit can be made to work properly. But, for large logic depths, there may not be any stable period. Hence adjusting the latching instant by itself may not be adequate for storing the correct result at the output register. For such cases, the clock period has to be increased to increase the stable period.

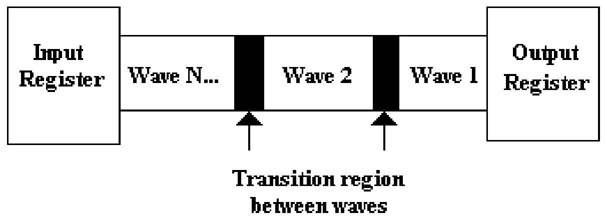


Fig. 4a Wave-pipelined circuit

Equalization of path delays, adjustment of the clock period and clock skew are the three tasks carried out for maximizing the operating speed of the wave-pipelined circuit. All the three tasks require the delays to be measured and altered if required. Layout editors, such as EPIC editor from Xilinx, may be used for this purpose. In [15], [16], these tasks are carried out manually. The wave-pipelined circuit designed using the layout editor may be tested using simulation. However, the simulation is inadequate for testing due to the difference between the actual delays and the delays calculated by the layout editor. This is because, the layout editor considers only the worst case delays and the actual delays may be significantly different due to fabrication variations.

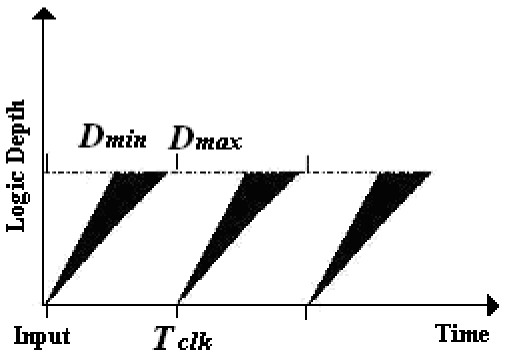


Fig.4b Temporal/spatial diagram of combinational logic circuits

This difference becomes important as the logic depth of the circuit increases. Hence, the design is downloaded to the actual FPGA and its operation is checked using a PC based test system in [16]. If correct results are not obtained, delays are altered and the design is downloaded for testing again. A number of iterations of place and route, simulation, downloading and testing in the actual device may be required till the correct results are obtained. The design of wave-pipelined circuit in this fashion requires human intervention and is time consuming. Automation of the above three tasks is considered in this paper.

IV. Automation schemes for wave-pipelined circuits

To maximize the operating speed of the wave-pipelined circuit, the equalization of the path delays is considered first. This cannot be completely automated as the commercially available synthesis tools do not support the specification of interconnect delays. However, the difference in path delays can be minimized by specifying the physical location of logic cells used for the implementation, through User Constraints File (UCF). This approach is proposed in [16].

The adjustment of the clock skew and clock period can be automated by using programmable clock and skew generator. The programmable clock and clock skew generator (without feedback) may be implemented in the FPGAs. When the custom block is integrated to the Nios II core using the SOPC builder or Micro Blaze, the CPU clock is available as one of the inputs to the custom block.

Sequential circuits may use this clock as the basic clock from which other clock frequency are derived. However, if the custom block requires a large number of sequential operations, it would be preferable to use a higher clock frequency to minimize the computation time. This may be achieved by generating the clock internally in the custom block. Fig. 5 gives the circuit diagram of a clock generation scheme which consists of a delay block and an inverter. The actual clock period depends on the interconnect delay. The select inputs s(0)-s(18) are connected to one data input and a(0)-a(18) are connected to the other port of the Nios II or Micro Blaze CPU. When prefix code is 0, the clock is reset to 0 and it keeps toggling periodically for other prefix codes. The select input of 8:1 multiplexer (s1, s2, s3) is varied by the processor to achieve different clock frequencies. Similarly, for the clock skew generator, the same circuit is used, but the feedback connection is removed and the select line is varied through processor to achieve different clock skew ranges. On implementation of the clock, it is found to contain glitches of the order of 1ns. This is suspected to be due to transients in the data ports, which occur when the prefix code is changed. To overcome this, a majority logic circuit with 3 inputs is used.

The circuit using the programmable clock and skew generator is a suboptimal wave pipelined circuit but can operate at a higher frequency than that reported by the commercially available synthesis tools which use Dmax for fixing the operating frequency. The automation (choosing the correct values for the select inputs for clock and skew generators) may be carried out using either off-chip processor or on-chip processor. The off-chip processor is used when the FPGA is used as a coprocessor or hardware accelerator for a main processor or microcontroller. The off-chip communication between the FPGA and a processor is bound to be slower than on-chip communication. In order to minimize the time required for adjustment of the parameters of the wave-pipelined circuit (clock frequency and skew), the Built In Self Test (BIST) approach for design for testability [17], [18] may be used.

In the SOC approach, a processor is assumed to be available on-chip and it is used for adjustment of the parameters of the wave-pipelined circuit. Automation of the tuning process using SOCs based on both Xilinx and Altera FPGAs are considered in this paper.

As shown in Fig. 5a, the combinational block used is the one level 2D DWT. A variety of choices exist for the implementation of system on-chip. The SOC may consist of a hardcore processor such as power PC or ARM processor and an FPGA coprocessor or DSP block. Alternatively, it may consist of a softcore processor such as MicroBlaze or Nios II and a custom DSP block implemented in FPGA. For adjustment of the parameters of the wave-pipelined circuit, any of these processors may be used and they need not be dedicated for this purpose. Even in the worst case, the parameters of the wave-pipelined circuit may have to be tuned only once in few hours. Hence, the speed of a combinational logic circuit can be enhanced with very little overhead using the SOC approach.

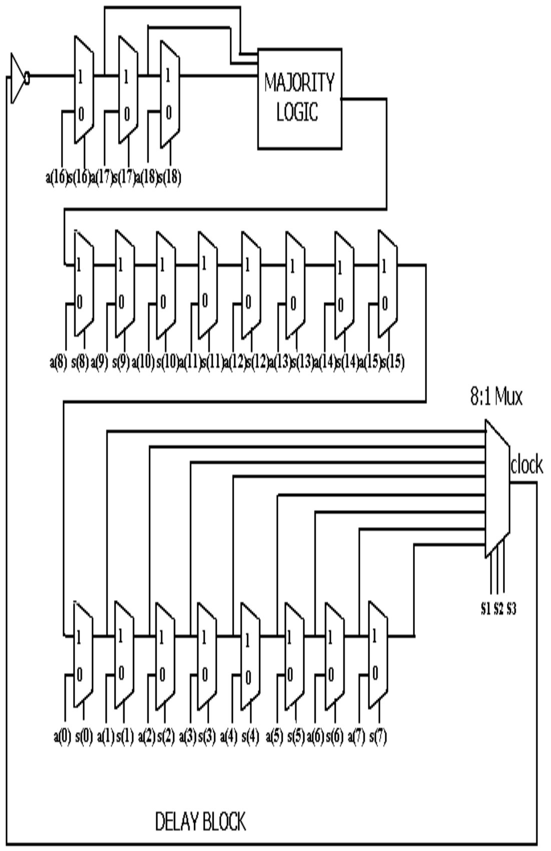

MAJORITY LOGIC
a(16) s(16) a(17) s(17) a(18) s(18)
a(8) s(8) a(9) s(9) a(10) s(10) a(11) s(11) a(12) s(12) a(13) s(13) a(14) s(14) a(15) s(15)
8:1 Mux
clock
S1 S2 S3
a(0) s(0) a(1) s(1) a(2) s(2) a(3) s(3) a(4) s(4) a(5) s(5) a(6) s(6) a(7) s(7)
DELAY BLOCK

Fig. 5 Clock generation scheme

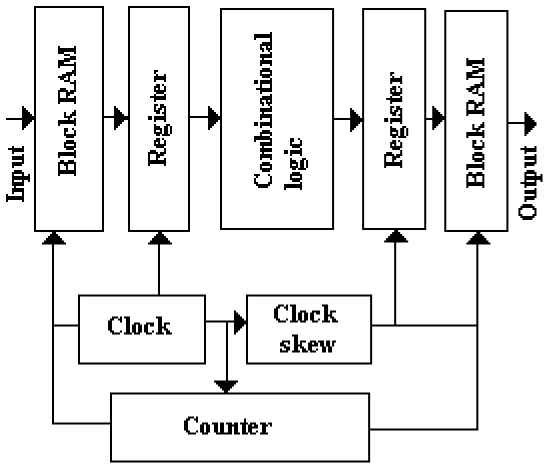


Fig. 5a SOC approach for wave-pipelined circuit

V. Procedure for adjusting the clock period and skew

The Altera softcore processor, Nios II, is implemented on Cyclone-II EP2C35F672C6 device and Xilinx softcore processor, MicroBlaze, is implemented on Spartan-III-XC3S200. The one level 2D DWT is implemented as custom hardware in both cases. As shown in Fig. 5a, the inputs to these blocks are applied from the block RAM and the results of these blocks are stored into the block RAM. During normal operation, the block RAM contains the array of data to be processed. In the test mode, the block RAM contains the test data.

During the testing mode, the processor writes the test vectors into block RAM, systematically applies the select inputs for the clock generator and clock skew blocks and uploads the results to be stored into the output block RAM for each combination of select inputs (s1-s3). It then checks the results with the expected results. It keeps varying the select inputs and repeats the above steps till the operating frequency at which the circuit works for three different skew values is found. The same procedure is followed for the next set of input vectors.

Alternatively, all possible combination of input vectors can be stored into block RAM, and the above procedure can be used to find the operating frequency. But this is feasible only for small word sizes.

Since the period of the clock signal and delay introduced by the clock skew block depend on the interconnect delays, the location of the logic elements and the interconnects used for the implementation of these blocks should be fixed so that when these blocks are integrated with the 2D DWT or the processor, the interconnect delays are not altered. This is achieved by using the Logic Lock feature in Altera. In the case of Xilinx FPGAs, this is achieved by using the Macros.

The operating frequency of the wave-pipelined circuit is expected to lie between that of non-pipelined circuit and pipelined circuit. Hence, the minimum and maximum frequency of the clock generator should correspond to the maximum operating frequencies of the non-pipelined and pipelined circuits respectively.

In the case of 2D DWT, the α block shown in Fig. 3 can be wave-pipelined and tuned using the circuit shown in Fig.4a. Similar circuits may be used for the β, γ, δ, ξ_1,ξ_2 blocks. However, the self tuning blocks need not be replicated. α block may be tuned using the circuit given in Fig. 5. After determining the optimum value of clock period, β block may also be tuned. In this case the output of the α block is assumed to be fed as input to the β block. After determining the optimum value of clock period for β block, γ block is tuned assuming the input to originate from the cascaded output of α, β blocks. Similarly δ and ξ blocks are tuned. Then the α, β, γ, δ, ξ_1,ξ_2 blocks are interconnected using pipelining. The clock frequency should be chosen to be the smallest frequency at which all the 5 blocks function individually. For proper operation of the hybrid WP-P BW-KCM, the skew value of each individual block is to be retuned. This is carried out after downloading the design onto the device.

VI. Implementation and study of the 2D DWT using lifting scheme

The 512x512 gray-level Lena with 8 bits per pixel is used for testing the three schemes. The 2D DWT scheme is implemented on Xilinx device using the lifting blocks with 9/7 biorthogonal filters and BW-PKCM multipliers. The lifting multiplier constants (α, β, γ, δ, ξ_1, ξ_2) are assumed to be of 8 bits each and the input samples are assumed to be of 11 bits. For 2D DWT, image block of size 32x32 is assumed. Fig. 6 overall block diagram of one level 2D-DWT. The input image and the output of the horizontal filters as well as vertical filters are assumed to be stored in the block RAMs. For the horizontal filters, the even and odd inputs are applied from two block RAMs of size 512x11. For testing, the image is assumed to be loaded into the block RAMs using Memory Initialization File (MIF). The result is written into 4 block RAMs of size 256x1.
In this paper In order to minimize the area required for implementation, a single horizontal filter may be reused to compute the one level 2D DWT. This procedure is repeated for the all the 25 sub blocks, then all are merged suitably. The one of the advantage of wavelet transform is that a subset of the coefficients represents a coarse form of the image and can be displayed (LL1) without computing the inverse transform.

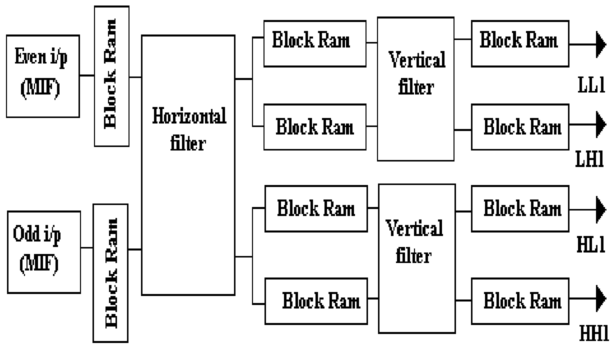


Fig. 6 Overall block diagram of one level 2D DWT

A. Implementation results on Spartan-III XC3S200

Implementation results for one level 2D DWT on Xilinx Spartan-III XC3S200 using all the three approaches and the results are given in Fig. 7. The programmable clock and clock skew blocks are implemented as Macro blocks using Xilinx ISE 8.1i project navigator. For tuning the wave-pipelined circuit, the Micro blaze softcore processor is used. Xilinx Embedded Design Kit (EDK) software is used to integrate the custom block to the Micro blaze processor. The rest of the steps are similar to what is used for the Altera SOC kit.

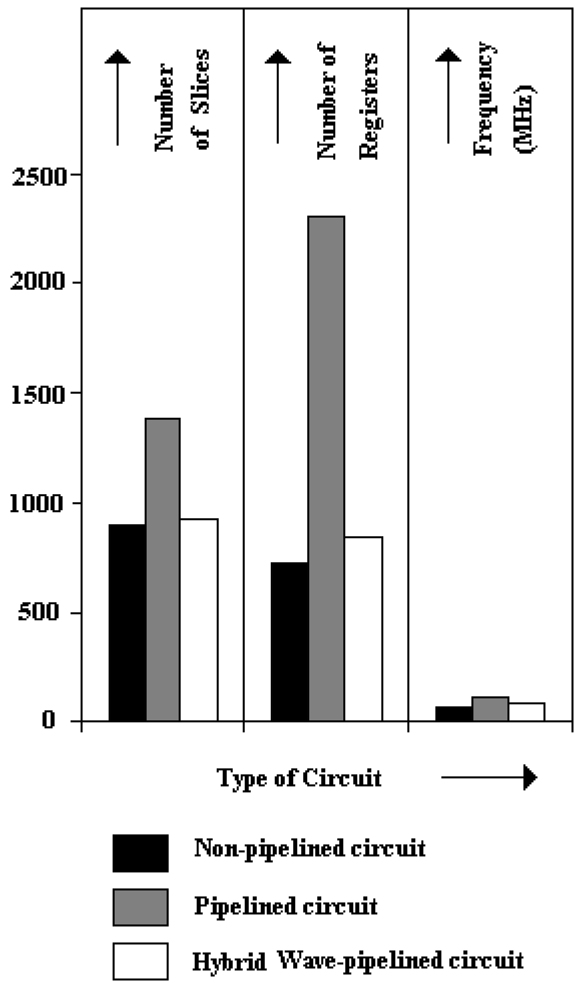


Fig. 7 Implementation results on Spartan-III XC3S200

For the all three schemes, the no. of logic elements, no. of registers, maximum operating frequency and power dissipated are computed and the results are given in Fig. 7. From this Fig. 7, it may be concluded that for the lifting scheme, the method using hybrid WP-P BW-KCM is faster than non pipelined BW-KCM by a factor of 1.07. The scheme with BW-PKCM is in turn faster than the hybrid WP-P BW-KCM by a factor of 1.56 and this is achieved with the increase in the number of registers by a factor of 3.157 and increase in the number of LEs by a factor of 1.54 compared to the hybrid WP-P unit.

B. Implementation of 1 level 2D DWT using ASIC

The 2D DWT scheme is implemented on ASIC using the lifting blocks with 9/7 biorthogonal filters and BW-PKCM multipliers. The 2D DWT is implemented using 180nm technology in ASIC. Verilog HDL language is used to describe the functionality of the circuit and after the circuit is described in HDL, functionality is verified modelsim simulation tool. Leo spectrum is used for synthesizing the circuit. First time the 2D DWT is implemented using ASIC. In future, it can be extended to compare with three schemes.

Table 1. Implementation results of 2D DWT

Scheme	Area	Freq.(Mhz)
Pipelining	6896	346.8

VII. Conclusions

In this paper, techniques for implementation of the 9/7 bi-orthogonal filters using hybrid WP-P KCM with Baugh-Wooley multiplication algorithm are proposed. The 9/7 bi-orthogonal filters implemented on Xilinx SOC device using the lifting scheme with the following three multipliers: with BW-PKCM, BW-KCM and hybrid WP-P BW-KCM. From the implementation results, it is verified that hybrid WP-P BW-KCM is faster compared to non pipelined BW-KCM and is register efficient, less clock routing complexity and less power dissipation compared to BW-PKCM. The one level 2D DWT scheme is also implemented, in ASICs using pipelining. This is first of its kind. It can be extended for non-pipelining and WP for future work.

References

1. G. Martin and H. Chang, “System-on-Chip design,” *Proc. of Intl. conf. on ASIC,* pp.12 – 17, 2001.
2. Altera documentation library- 2003, *Altera Corporation*, USA.
3. Xilinx documentation library, *Xilinx Corporation*, USA.
4. W. Tuttlebee, “Software defined radio,” *John Wiley & Sons ltd*. USA, 2004.
5. Jorg Ritter and Paul Molitor, “A pipelined architecture for partitioned DWT based lossy image compression using FPGA's,” *Proc IEEE conf. FPGA 2001*: 201-206, 2001.

6. G. Lakshminarayanan, B. Venkataramani, J. Senthil Kumar, A. K. Md. Yousuf and G. Sriram, "Design and FPGA implementation of image block encoders with 2D-DWT," *Proc. TENCON 2003*, vol. III, pp 1015-1019, Oct 15-17, Bangalore, 2003.
7. I. Daubechies, and W. Sweldens, "Factoring Wavelet Transforms into Lifting Steps," *Journal of Fourier Analysis and Applications*, vol. 4, pp 247-269, Nov. 3, 1998.
8. A. D. Bruce, J. Ross, B. Böhm, A. P. W. Charles , and C. Monica, "Accelerated Image Processing on FPGAs," *IEEE Transactions On Image Processing*, vol. 12, no. 12, pp. 1543-1551, Dec. 2003.
9. J. Nyathi and J. G. Delgado-Frias, "A Hybrid wave-pipelined network router," *IEEE Transactions on Circuits and Systems- I, Fundamental Theory and Applications*, vol. 49, no. 12, pp. 1764 –1772, Dec. 2002.
10. O. Hauck, A. Katoch and S. A. Huss**,** "VLSI system design using asynchronous wave pipelines: a 0.35 μm CMOS 1.5 GHz elliptic curve public key cryptosystem chip," *Proc. of Sixth Intl. Symposium on Advanced Research in Asynchronous Circuits and Systems 2000 (ASYNC 2000)*, pp. 188 –197, April 2000.
11. W. P. Burleson, M. Ciesielski, F. Klass and Liu, **"**Wave-pipelining: a tutorial and research survey,**"** *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 6, Issue. 3, pp. 464 -474, Sept. 1998.
12. C. Gray, W. Liu and R. Cavin, "Wave-pipelining: Theory and Implementation," *Kluwer Academic Publishers*, 1993.
13. K. K. Parhi, "VLSI signal processing systems," *John Wiley & Sons*, 1999.
14. C. Thomas Gray, Wental Liu and Ralph K. Cavin, III, "Wave-pipelining: Theory and CMOS Implementation," *Kluwer Academic Publishers*, 1994.
15. E. I. Boemo, S. Lopez-Buedo and J. M. Meneses, "Wave pipelines via look-up tables," *IEEE International Symposium on Circuits and Systems (ISCAS '96)*, vol. 4, pp. 185 -188, 1996.
16. G. Lakshminarayanan and B. Venkataramani, "Optimization techniques for FPGA based wave-pipelined DSP blocks," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 13, no. 7, pp 783-793, July 2005.
17. M. J. S. Smith, "Application Specific Integrated Circuits," *Pearson Education Asia Pvt. Ltd.*, Singapore, 2003.
18. G. Seetharaman, B. Venkataramani and G. Lakshminarayanan, "Design and FPGA implementation of self tuned wave-pipelined filters," *IETE journal of research*, vol. 52, no. 4, pp. 281-286, July-August 2006.

Design Space Exploration to Find the Optimum Cache and Register File Size for Embedded Applications

Mehdi Alipour [1], **Mostafa E. Salehi**[1], **Hesamodin shojaei baghini**[2]
[1] Islamic Azad University, Qazvin Branch, Qazvin 34185-1416 Iran.
[2] Computer Engineering Department, Javid University of Jiroft, Azadi Ave, Jiroft, Iran.

Abstract - *In the future, embedded processors must process more computation-intensive network applications and internet traffic and packet-processing tasks become heavier and sophisticated. Since the processor performance is severely related to the average memory access delay and also the number of processor registers affects the performance, cache and register file are two major parts in designing embedded processor architecture. Although increasing cache and register file size leads to performance improvement in embedded applications and packet-processing tasks in high traffic networks with too much packets, the increased area, power consumption and memory hierarchy delay are the overheads of these techniques. Therefore, implementing these components in the optimum size is of significant interest in the design of embedded processors. This paper explores the effect of cache and register file size on the processor performance to calculate the optimum size of these components for embedded applications. Experimental results show that although having bigger cache and register file is one of the performance improvement approaches in embedded processors, however, by increasing the size of these parameters over a threshold level, performance improvement is saturated and then, decreased.*

Keywords: Embedded processor, design space exploration, cache, optimum size of register file, cache access delay.

1 Introduction

In recent years embedded application and internet traffic become heavier and sophisticated so, future embedded processors will be encountered by more computation-intensive embedded applications, in this way, designing high performance processors is recommended. By scaling down the feature size, technology and presentation of chip multiprocessors (CMP) that are usually multi-thread processors, somehow the user's performance necessity have guaranteed. Inseparable parts in designing these processors are cache and register file because the performance of a processor is severely related to cache access and also having enough registers.

Recently in numerous researches, multi-thread processors are used to design a fast processor especially in network processors [4], [9], [11], [23], [25], and [26]. In [3] a Markov model based on fine grain multithreading is implemented. Analytical Markov model is faster than simulation and has dispensable inaccuracy. In this chain, stalled threads defined as states and transitions are based on cache contention between threads.

Cache memories are usually used to improve the performance and power consumption by bridging the gap between the speed and power consumption of the main memory and CPU. Therefore, the system performance and power consumption is severely related to the average memory access time and power consumption which makes cache as a major part in designing embedded processor architectures.

In [4] cache misses are introduced as a factor for reducing memory level parallelism between threads. In [5] thread criticality prediction has been used and for better performance, resources are given to the threads that have higher L2 cache misses which are called the most critical threads.

To improve packet-processing in network processors, [6]-[8] have applied direct cache access (DCA). In [9] processor architecture is based on the simultaneous multithreading (SMT) and cache miss rate is used to show the performance improvement.To find out the effect of cache access delay on performance, a comparison between multi-core and multi-thread processors has been performed in [10]. Likewise, victim cache is an approach to improve the performance of a multi-thread processor [11].

All recent researches are based on the comparison results with single-core single-thread processors. In the other word multi-thread processors are the heir of the single thread processors [23], [25], and [26]. Hence, evaluating the effective parameters like cache and register file sizes are required for designing a multithread processor. The basic purpose of this paper is to study the effect of the cache size on the performance because embedded processors process computation and data intensive applications and larger cache sizes will present better performance.

Generally, one of the easiest way to improve the performance of embedded and network processors is increasing the cache size [2], [12], [13], [14], and [22]-[26] but this improvement, severely increase the occupied area and power consumption of the processor. So, it is necessary to find a cache size that creates tradeoffs between performance and power-area of the processor.

From other point of view, because of the performance per area parameter, higher performance in a specified area budget is one of the most important needs of a high performance embedded processor. A negative point of the recent researches is that they don't have any constraints on the cache size.

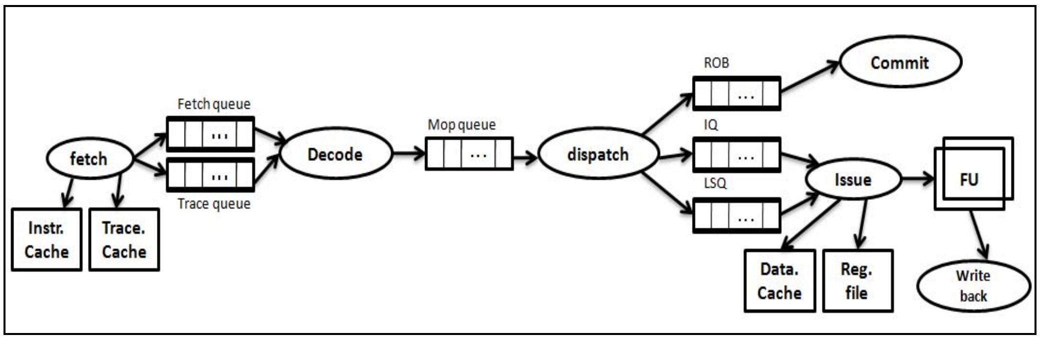


Fig1. Processor pipeline of Multi2sim simulator [19].

Because of the limited area budget in embedded processors, in this paper we have found the optimum size of L1 and L2 cache and also, because of the longer latency of bigger caches, best size of memory hierarchy in relation to this parameter has been calculated.

As mentioned above, another inseparable part in designing embedded processors is register file. Like the cache, size of this parameter has the fundamental effect on the better processor performance. To improve the performance of a embedded processor, a large register file must be implemented. However, larger register files occupy more area and make a worse critical path [18]. Therefore, exploring the optimum size of the register file is the second purpose of this paper. The high importance of this issue is based on the fact that some parameters encourage designer to have a large register file. Generally embedded processors are implemented in multi-issue architecture and out of order (OOO) instruction execution that has renaming logic [16]-[18], [23], [25], and [26]. On the other hand, because register files are shared in multi-thread processors, observing the fact: increment the common parts in design, force the designer to have a larger register file [1]. These parameters also make higher importance for register file size. In [15] effects of register file size in SMT processors have been studied. However, high budget for the number of registers has used. In recent researches effect of register file and cache size in the same time is not studied, so in this paper this issue will be studied too.

2 Simulation environment

For simulation, we have used Multi2sim version 2.3.1 [19], a superscalar multi-thread multi-core simulation platform which has 5 stages of pipeline named *fetch, decode, dispatch, issue, writeback, and commit.* This simulator executes x86 instruction sets. Fig.1 shows a block diagram of the processor pipeline modeled in Multi2Sim. In the fetch stage, instructions are read from the instruction or the trace cache. Depending on their origin, they are placed either in the fetch queue or the trace queue. The former contains raw macroinstruction bytes, while the latter stores pre-decoded microinstructions (uops). In the decode stage, instructions are read from these queues, and decoded if necessary. Then, uops are placed in program order into the uop queue. The fetch and decode stages form the front-end of the pipeline [19]. The dispatch stage takes uops from the uop queue, renames their source and destination registers, and places them into the reorder buffer (ROB) and the instruction queue (IQ) or load-store queue (LSQ). The issue stage is in charge of searching both the IQ and LSQ for instructions with ready source operands, which are schedule to the corresponding functional unit or data cache. When and uop completes, the writeback stage stores its destination operand back into the register file. Finally, the completed uops at the head of the ROB are taken by the commit stage and their changes are confirmed. So the commit stage is where we can log and count the number of committed instructions for performance comparison. Detail of this simulation is described in the simulation method and results section.

Multi2sim can run programs in multi issue platform, but to evaluate the requirements of each thread we have used the single issue model for comparison. We changed and compiled the source code of simulator on a 2.4GHz, dual core processor with 4GB of RAM and 6MB of cache that run fedora 10 as an operating system. Base on this configuration the average time of each simulation is about 20 minutes.

3 Benchmarks

The aim of this paper is to calculate the optimum cache and register file size. Because embedded applications are so pervasive homogenous applications they cannot be a good choice for DSE. Hence we have applied our DSE on heterogeneous applications, such that in some of them data cache is more important and in the others instruction cache is more important. So we apply PacketBench [20] and MiBench [27] respectively. PacketBench is a good platform to evaluate the workload characteristics of network processors. It reads and writes packets from and to real packet traces, and manages packet memory, and implements a simple application programming interface API. This involves reading and writing trace files and placing packets into the memory data structures used internally by PacketBench. On a network processor, many of these functions are implemented by specialized hardware components and therefore should not be considered part of the application. Programs in this tool are categorized in 3 parts: *1- IP forwarding* which is corresponding to current internet standards. *2- Packet classification* which is commonly used in firewalls and monitoring systems. *3- Encryption*, which is a function that actually modifies the entire payload of the packet.

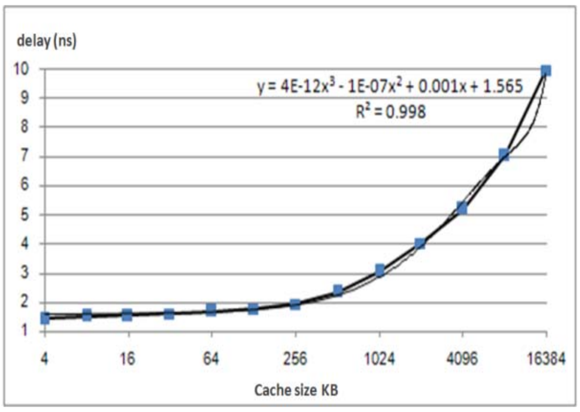


Fig.2. Effect of cache size on cache access delay

Specific applications that respectively we used from each category are IPv4-Lctrie, Flow-Classification and IPSec.IPv4-trie performs RFC1812-based packet forwarding. This implementation is derived from an implementation for the Intel IXP1200 network processor. This application uses a multi-bit trie data structure to store the routing table, which is more efficient in terms of storage space and lookup complexity [20].Flow classification is a common part of various applications such as firewalling, NAT, and network monitoring. The packets passing through the network processor are classified into flows which are defined by a 5-tuple consisting of the IP source and destination addresses, source and destination port numbers, and transport protocol identifier. The 5-tuple is used to compute a hash index into a hash data structure that uses link lists to resolve collisions [20]. IPSec is an implementation of the IP Security Protocol [27], where the packet payload is encrypted using the Rijndael Advanced Encryption Standard (AES) algorithm [28]. This is the only application where the packet payload is read and modified.

MiBench is a combination of six deferent categories. We have selected 3 of them: *1- Dijkstar* from network category, *2- Susan (corners)* from automotive and industrial control category, and *3- String-search* from office category. The Dijkstra benchmark constructs a large graph in an adjacency matrix representation and then calculates the shortest path between every pair of nodes using repeated executions of Dijkstra's algorithm [49]. *Susan* is an image recognition package. It was developed for recognizing corners and edges in magnetic resonance images of the brain [27]. *String-search* searches for given words in phrases using a case insensitive comparison algorithm.

4 Simulation method and results

Purpose of this paper is to evaluate optimum size of cache and register file. At first, we describe the methodology to extract proper size of cache. For this purpose, it is necessary to configure the simulator in the way that just the size of cache be the parameter that has affects on the performance. So, for each application the execution number of the main function is calculated in different sizes of L1 and L2 caches. For this purpose we made changes in some parts of simulator source code to calculate the cycles of sending a packet (the cycles that are used to execute the main function of each application).

To calculate the beginning address and the end address of the main function we disassemble the executable code of each benchmark application and extract these addresses and then these parameters are back annotated to commit.c and processor.h file of Multi2sim simulator where a thread is executed.

By these changes we can calculate the number of x86 instructions and macroinstructions and the execution cycles for each specific function.

The second step is to run the simulator with different cache sizes. But the worthwhile point is that although based on the recent researches that recommend doubling the cache size for improving the performance of a processor, however during doubling the cache size, important parameters like area power and cache access delay must be considered. For this purpose we have used CACTI 5.0 [21], a tool from HP that is a platform to extract parameters relevant to cache size considering fabrication technology. Most important parameters that we used in this research are in table 1.

To compare the performance based on the cache size, extracted results from cacti (L1 and L2 cache access delay) are back annotated to Multi2sim. This work has been done by calculating the simulator cycle time and comparing it to the results of cache access time from CACTI. In this way when the cache size is changed, actual cache access delays are considered.

As can be seen in Fig.2, increasing the cache size, leads to more cache access delays.For exploring the cache size, the other simulator parameters are set to the default value, because the purpose is to find the best cache size for a single-thread single-core processor for embedded applications. i.e. width of the pipeline stages must be one (issue-width =1).

Table 1
The most important parameters used in cacti

	L1 cache	L2 cache
Cache size	Variable	Variable
Cache line size	Variable	Variable
Associatively	Variable	Variable
Number of banks	1	1
Technology node (nm)	90nm	90nm
Read/write ports	1	1
Exclusive read ports	0	0
Exclusive write ports	0	0
Change tag	No	No
Type of cache	Fast	normal/serial
Temperature (K)	300-400	300-400
RAM cell/transistor type in data array	ITRS-HP	Global
RAM cell/transistor type in tag array	ITRS-HP	Global

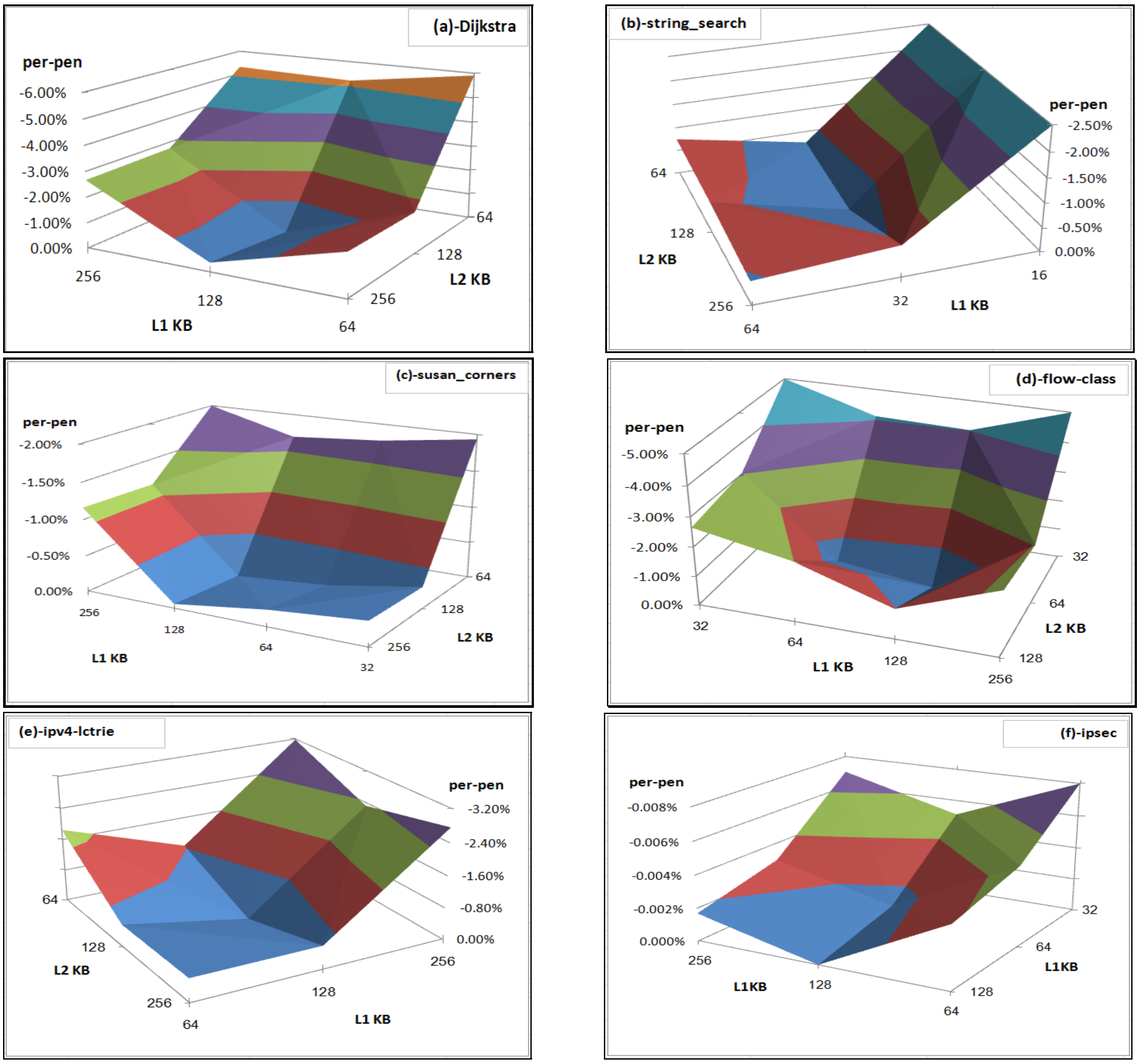


Fig.3. Effect of cache size on the performance (a):Dijkstra, (b): String_search, (c):Susan.corners, (d):flow_class, (e):ipv4_lctrie, (f): ipsec

5 Analysis of the simulation results

Fig.3 shows the extracted result of our simulations. In this figure each axis is labelled and the vertical axis (per-pen) shows the performance penalty of related cache size configuration. Based on these results, by increasing the cache size we can achieve more hit rates, however, because of the longer cache access time of larger caches, from a specific point (best cache size) performance improvement is saturated and then even decreased. In other word, doubling the cache size cannot always improve the performance. From other point of view, area budget is limited and always we can't have a large cache, so, considering the sizes smaller and near the best cache size, performance degradations are negligible (3% in average).

To calculate the optimum size of register file, we have applied the parameters used for calculating best cache size, however, to find out just the effect of register file size on the performance, we used the best cache size (L1 and L2) concluded in the previous section for the cache size and run the simulator accordingly. Fig.4 shows the results of this part. In this picture vertical column shows the performance effect (performance penalty) of register file size.

Numbers in this column are relative to the best size of register file. It shows that although for all applications, in average, the best size of register file is 68 and above but in sizes near the half of this size performance penalty is lower that 5%.

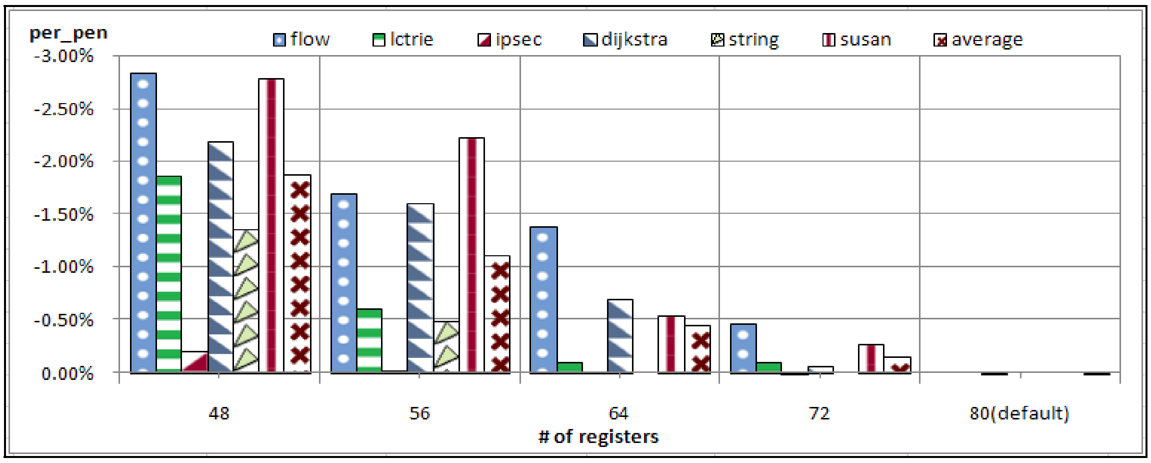


Fig.4. Effect of register file size on performance of different embedded applications.

Also fig.4 shows that reducing the register file size always decrease the performance but sometimes, by doubling the register file size we don't have noticeable performance improvement. So the first point that the highest performance is reached, is introduced as the best size for register file.

It is worthwhile to say that in Fig.4 the effect of cache size and register file size can be seen. In other side of view based on recent researches [23, 25, 26] to have a multi-thread architecture we need more area budget, and to run this architecture in the best performance that can be met, multi issue architecture with renaming logic , ROB, LSQ, IQ and other OOO components which occupy large area budget are needful. Base on these simulations, we calculated 2 point for cache and register file sizes: *1- best size* that has no performance penalty and occupy bigger area budget and *2- optimum size* that has about 3% performance penalty and occupy smaller area budget so, we can deduce that in the optimum size of cache and register file we have saved the area budget of the processors and qualification to run multi-threads in the higher issue widths is obtained. In other word in lower area more performance is achieved and causes to growth the most important parameters for embedded applications that is performance per area.

As mentioned before the multi-thread processors are the heir of single thread processors. So, extracting the best size for important parameters like cache size and register file size is necessary.

6 Conclusion

In this paper we have studied the effect of cache and register file size on the performance of an embedded processor and extracted the best size of these two parameters for embedded applications. Simulation results show that for selected benchmarks the best size of L1 and L2 caches are 64KB and 128KB respectively, and the best size of register file is 80. Experiments show that although by increasing the cache size performance will improve, but in a specific point the performance improvement is saturated and then decreased. Also increasing register file size cannot always improve the performance and in a specific size the performance improvement will be saturated. From the area point of view, based on the results of this research, when we select half of the best size of the cache and register file, performance penalty is about 3% in average. In other word in sizes lower than best size the acceptable performance can be met. It means we can reach performance requirements in lower area and also have a better performance/area parameter.

7 References

[1] David A. Patterson, John L. Hennessy.: Computer *organization and design*: the hardware/software interface, Morgan Kaufman 2007.

[2] Davanam, N., Byeong Kil Lee., "Towards Smaller-sized Cache for Mobile Processors Using Shared Set-Associatively," international conference on information technology. pp. 1-6. 2010.

[3] Chen, X.E., Aamodt, T.M., "A First-Order Fine-Grained Multithreaded Throughput Model," International Symposium on High Performance Computer Architecture (HPCA), pp.329-340.2009.

[4] Shailender, Chaudhry. Robert, Cypher. Magnus, Ekman. Martin, Karlsson. Anders, Landin. Sherman, Yip. Haakan, Zeffer. Marc,Tremblay," Simultaneous speculative threading: a novel pipeline architecture implemented in sun's rock processor," international symposium on computer architecture(ISCA 2009). pp. 484-495, 2009.

[5] Abhishek, Bhattacharjee. Margaret, Martonosi. "Thread Criticality Predictors for Dynamic Performance, Power, and Resource Management in Chip Multiprocessors," International Symposium on Computer Architecture (ISCA). pp. 168-176, 2009.

[6] Huggahalli, R. Iyer, R. Tetrick, S.,"Direct cache access for high bandwidth network I/O," International Symposium on computer Architecture (ISCA). pp. 50-59, 2005.

[7] Kumar, A. Huggahalli, R.,"Impact of Cache Coherence Protocols on the Processing of Network Traffic," International

symposium on Microarchitecture. (MICRO), pp. 161-171, 2007.

[8] Kumar, A. Huggahalli, R. Makineni, S.,"Characterization of Direct Cache Access on Multi-core Systems and 10GbE," International Symposium on High Performance Computer Architecture (HPCA). pp. 341-352, 2009.
[9] Kyueun, Yi. Gaudiot, J.-L.,"Network Applications on Simultaneous Multithreading Processors," Journal of IEEE Transaction on Computer (TCOMPUTER). pp. 1200-1209, 2009.

[10] Guz, Z. Bolotin, E. Keidar, I. Kolodny, A. Mendelson, A. Weiser, U.C.," Many- Core vs. Many-Thread Machines:Stay Away From the Valley," Journal Computer Architecture Letters (L-CA), pp.25-28, 2009.

[11] Colohan, C.B. Ailamaki, A.C. Steffan, J.G. Mowry, T.C.,"CMP Support for Large and Dependent Speculative Threads," Journal IEEE Transaction on Parallel and Distributed systems (TPDS), pp. 1041-1054, 2007.

[12] Bienia, C. Kumar, S. Kai Li.," PARSEC vs. SPLASH-2: A quantitative comparison of two multithreaded benchmark suites on Chip- Multiprocessors," International Symposium on Workload Characterization (IISWC), pp.47-56, 2008.

[13] Guanjun, Jiang. Du, Chen. Binbin, Wu. Yi, Zhao. Tianzhou, Chen. Jingwei, Liu.," CMP Thread Assignment Based on Group sharing L2 Cache," International Conference on Embedded Computing, pp. 298-303, 2009.

[14] McNairy, C. Bhatia, R.," Montecito:a dual core dual thread Itanium processor," IEEE Journal MICRO, pp.10-20, 2005.

[15] Alastruey, J. Monreal, T. Cazorla, F. Vinals, V. Valero, M.,"Selection of the Register File Size and the Resource Allocation Policy on SMT Processors Policy on SMT Processors," International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD), pp.63-70, 2008.

[16] A, Yamamoto. Y, Tanaka. H, Ando. T, Shimada.," Data prefetching and address pre-calculation through instruction pre-execution with two-step physical register deallocation," in MEDEA-8, pp. 41–48, 2007.

[17] Yamamoto. Y, Tanaka. H, Ando. T, Shimada.,"Two-step physical register deallocation for data prefetching and address precalculation," IPSJ Trans. on Advanced Computing Systems. vol. 1, no. 2, pp. 34–46, 2008.

[18] Tanaka, Y. Ando, H.,"Reducing register file size through instruction pre execution enhanced by value prediction,"IEEE International Conference on Computer Design, pp. 238 – 245. 2009.

[19] R, Ubal. J, Sahuquillo. S, Petit. P, L'opez.," Multi2Sim: A Simulation Framework to Evaluate Multicore-Multithreaded Processors," journal Proc. of the 19th Int'l Symposium on Computer Architecture and High Performance Computing. Oct 2007.
[20] Ramaswamy, Ramaswamy. Tilman, Wolf.," PacketBench: A tool for workload characterization of network processing," in Proc. of IEEE 6th Annual Workshop on Workload Characterization (WWC-6), Austin, TX. pp. 42-50, Oct. 2003.

[21] Shyamkumar Thoziyoor, Naveen Muralimanohar, and Norman P. Jouppi "CACTI 5.0 technical report", form Advanced Architecture Laboratory, HP Laboratories HPL-2007. [Online]. Available:www.hpl.hp.com/research/cacti/

[22] Hyunjin, Lee. Sangyeun, Cho. Childers, B.R.," StimulusCache: Boosting Performance of Chip Multi-processors with Excess Cache," IEEE 16th International Symposium on High Performance Computer Architecture (HPCA), pp, 1 – 12, 2010.

[23] Chung, E.S. Hoe, J.C.," High-Level Design and Validation of the BlueSPARC Multithreaded Processor," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD). vol. 29, no. 10, pp. 1459–1470, 2010.

[24] Davanam, N. Byeong, Kil. Lee.," Towards Smaller-sized Cache for Mobile Processors Using Shared Set-Associativity," international conference on information technology, pp. 1-6, 2010.

[25] Kyueun Yi, and Gaudiot J. L, "Network Applications on Simultaneous Multithreading Processors," *IEEE Transaction on Computer (TCOMPUTER).*vol. 59, no. 9, pp. 1200-1209, SEPTEMBER 2010.

[26] Wang, H. Koren, I. Krishna, C.," Utilization-Based Resource Partitioning for Power-Performance Efficiency in SMT Processors," IEEE Transactions on Parallel and Distributed Systems, (TPDS) vol. 22, no. 99, pp. 191-216, 2010.

[27]M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T.Mudge, and R. B. Brown, “MiBench: a free, commercially representative embedded benchmark suite,” *in Proceedings of the IEEE InternationalWorkshop onWorkload Characterization,* pp. 3-14,2001.

[28]S.K. Dash, T. Srikanthan, "Instruction Cache Tuning for Embedded Multitasking Applications," *IEEE/IFIP International Symposium on Rapid System Prototyping,* pp. 152-158, 2009.

A Wearable Trajectory Reconstruction System Using Inertial and Magnetic Sensors

Jeen-Shing Wang[1], **Yu-Liang Hsu**[1], **Ping-En Cheng**[1], **Lun-Chia Kuo**[2], **and Jen-Chieh Chiang**[2]
[1]Department of Electrical Engineering, National Cheng Kung University, Tainan, Taiwan, R.O.C.
[2]Information and Communications Research Laboratories, Industrial Technology Research Institute, Hsinchu, Taiwan, R.O.C.

Abstract - *This paper presents an inertial-sensor-based wearable device and its associated pedestrian trajectory reconstruction algorithm to reconstruct indoor pedestrian trajectories. The wearable device is composed of a triaxial accelerometer, a triaxial gyroscope, a triaxial magnetometer, a microcontroller, and a Bluetooth wireless transmission module. Users can carry the device to walk in indoor environments at normal speed. A pedestrian trajectory reconstruction algorithm composed of the procedures of data collection, signal preprocessing, and trajectory reconstruction has been developed for reconstructing pedestrian trajectories. In order to minimize the integral errors caused by the intrinsic noise/drift of inertial sensors, we utilize sensor fusion techniques with the Kalman filter for correcting orientation estimation and trajectory reconstruction. The advantages of this wearable device include the following: 1) It can be used anywhere without any external indoor localization technique; and 2) the pedestrian trajectory reconstruction algorithm can reduce integral errors effectively and thus can construct the pedestrian trajectories accurately. Our experimental result on pedestrian trajectory reconstruction has successfully validated the effectiveness of the inertial-sensor-based wearable device and its pedestrian trajectory reconstruction algorithm.*

Keywords: Indoor navigation system, pedestrian trajectory reconstruction algorithm, Kalman filter

1 Introduction

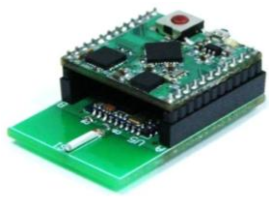
Recently, a wearable device embedded with inertial sensors has been provided for indoor pedestrian navigation or localization [1], [2]. The salient superiority of inertial sensors for general indoor location sensing is that they can be operated without any external indoor localization technique (such as radio frequency (RF), WiFi, or Zigbee) [3]. This kind of devices usually consists of two types of sensors: accelerometers for sensing translational accelerations, and gyroscopes for detecting angular velocities. However, most inertial-sensor-based indoor pedestrian navigation systems suffer from poor accuracy due to various errors generated by the inertial sensors. The errors not only involve the effects of sensor uncertainty, namely scale factor (sensitivity) and bias (offset), but also include contingent error sources, such as intrinsic drift of inertial sensors, circuit thermal noise, time discretization, quantization error, vibration, or friction, etc. To improve the accuracy of trajectory reconstruction, many researchers have focused on the subject of error reduction and compensation of inertial sensors by developing effective algorithms. For example, Bang *et al.* [4] used an error compensation method, called zero velocity compensation (ZVC), to reduce the cumulative error based on the assumption that velocity should be zero at the start and the end of a motion. Bird *et al.* [1] applied the zero velocity updates (ZUPTs) to pedestrian navigation systems, where zero velocity was fed into the Kalman filter, during the stance phase of a gait cycle to reduce the cumulative error. Huang *et al.* [5] integrated the strap-down inertial navigation and pedestrian dead-reckoning techniques for pedestrian tracking tasks. In addition, the ZUPTs was applied to reduce the integral errors for reconstructed trajectories in the stance phase of a gait cycle.

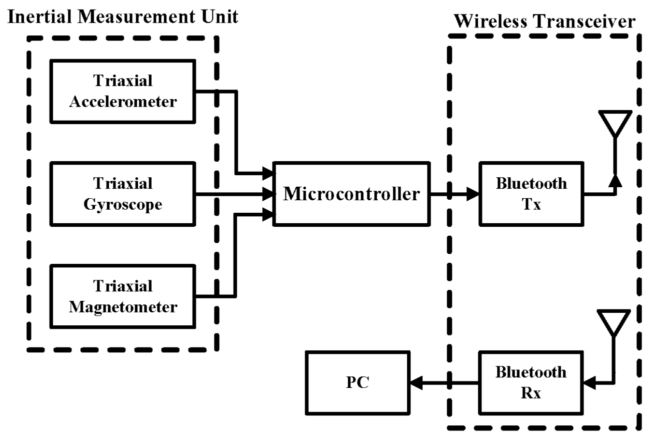
An inertial-sensor-based wearable device and a pedestrian trajectory reconstruction algorithm have been developed in this study. The wearable device is composed of a triaxial accelerometer, a triaxial gyroscope, a triaxial magnetometer, a microcontroller, and a Bluetooth wireless transmission module. The measured inertial and magnetic signals from the sensors are transmitted to a computer via the wireless module. Users can carry the device to walk in indoor environments at normal speed without any space limitation. The human walking trajectories can be reconstructed by the pedestrian trajectory reconstruction algorithm, which consists of the procedures of data collection, signal preprocessing, and trajectory reconstruction. In order to minimize the cumulative error caused by the intrinsic noise/drift of sensors, we have utilized sensor fusion techniques with the Kalman filter for further improving the accuracy of the estimated orientations and reconstructed trajectories. The effectiveness of the wearable device and its associated pedestrian trajectory reconstruction algorithm has been validated by an indoor positioning experiment. The advantages of our proposed wearable device include: 1) It is portable and can be used anywhere without any external indoor localization technique, and 2) the pedestrian trajectory reconstruction algorithm can reduce orientation and integral errors effectively and thus can reconstruct the trajectories of human walking accurately.

2 Methods

2.1 Apparatus

The wearable device generates and transmits the inertial and magnetic signals of a user's walking motion. The device includes a triaxial accelerometer (LSM303DLH), a triaxial gyroscope (L3G4200D), a triaxial magnetometer (LSM303DLH), a microcontroller, and a Bluetooth wireless transmission module. The size of the board is 37.5mm × 25mm × 10mm as shown in Fig. 1 (a). The output signals of the accelerometer, gyroscope, and magnetic sensors are all sampled at 60 Hz. The overall power consumption of the wearable is 30 mA at 3.7 V. The schematic diagram of the wearable hardware system is shown in Fig. 1 (b). The pedestrian trajectory reconstruction was performed on a PC running Microsoft® Windows 7 operating system with an Intel Core Processor i5-450 and 4G RAM.


(a)



(b)

Fig. 1. (a) The wearable hardware device for indoor localization system. (b) The schematic diagram of the wearable device.

2.2 Pedestrian trajectory reconstruction algorithm

A pedestrian trajectory reconstruction algorithm has been developed to perform the trajectory reconstruction of human walking using the signals measured by the inertial and magnetic sensors embedded in the wearable device. The pedestrian trajectory reconstruction algorithm is composed of the following procedures: 1) data collection, 2) signal preprocessing, and 3) trajectory reconstruction. At the beginning of the procedure, the acceleration, angular velocity, and magnetic signals are transmitted to a computer via the Bluetooth transceiver. Before using the wearable device, we need to calibrate the inertial and magnetic sensors to reduce the errors of sensitivity and offset of the sensors first. Then a moving average filter is applied to remove high frequency noise from the raw data. The trajectory reconstruction procedure includes the following steps: 1) orientation estimation, 2) coordinate transformation and gravity compensation, and 3) position estimation. In the orientation estimation step, the quaternion representation is employed to express the orientation of the wearable device. To obtain correct orientation of the wearable device, a sensor fusion technique with the Kalman filter is used to estimate orientation angles based on the corresponding filtered acceleration, angular velocity, and magnetic signals. Once we obtain the quaternion-based orientation, the filtered acceleration can be transformed from the body coordinate to the reference coordinate, and the gravitational acceleration can then be removed. In the position estimation step, the velocities and positions (or trajectory) of the wearable device in a walking motion can be obtained through the single integral and double integral of the compensated acceleration signals, respectively. The pedestrian trajectory reconstruction process of the wearable device is shown in Fig. 2.

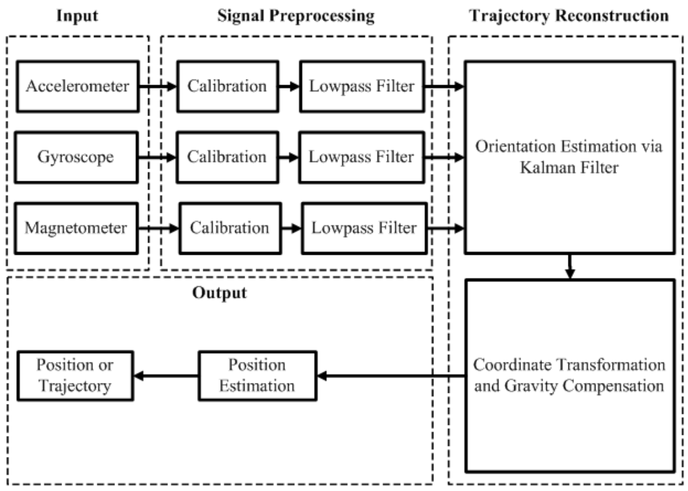


Fig. 2. Pedestrian trajectory reconstruction process of the wearable device.

3 Experimental results

The proposed wearable device was mounted on the users' foot. 10 users were asked to walk along a straight line of 23.6 meters on a flat floor of a building with no speed and pace limit. Fig. 3 shows the result of the experiment. There are about 1.0±0.6 meters short compared with the total path of 23.6 meters. Hence, the average error of the reconstructed pedestrian trajectory is around 4.23%±2.54% as to the distance traveled. In addition, to validate the effectiveness of the proposed pedestrian trajectory reconstruction, we compared the trajectory reconstruction algorithm proposed in [6] for the same path. There are about 2.3±1.4 meters short compared with the length of the path. Hence, the error of the reconstructed pedestrian trajectory is around 9.7%±5.93% as to the distance traveled. Obviously, the result of the proposed pedestrian trajectory reconstruction algorithm is better than one of the algorithm proposed in [6].

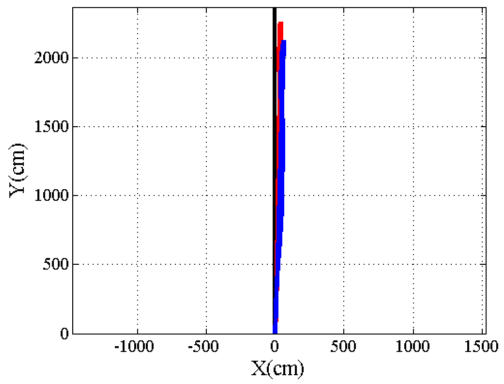


Fig. 3. Walking along a straight line. (Block color: reference path; Red color: reconstructed pedestrian trajectory via the proposed pedestrian trajectory reconstruction algorithm; Blue color: reconstructed trajectory via the trajectory algorithm proposed in [6])

4 Conclusion

A systematic framework of the trajectory reconstruction of human walking for pedestrians has been presented in this paper. The acceleration, angular velocity, and magnetic signals are transmitted to a computer via the Bluetooth transceiver. The proposed pedestrian trajectory reconstruction algorithm consists of data collection, signal preprocessing, and trajectory reconstruction. To obtain correct orientation of the wearable device, a sensor fusion technique with the Kalman filter is used to estimate orientation angles based on the corresponding filtered acceleration, angular velocity, and magnetic signals. In the pedestrian trajectory reconstruction experiments, the average error of the reconstructed pedestrian trajectory is around 4.23%±2.54% as to the distance traveled. The effectiveness of the wearable trajectory reconstruction system and its pedestrian trajectory reconstruction algorithm has been successfully validated by the experiments.

5 Acknowledgment

This research was supported by Information and Communications Research Laboratories (ICL), Industrial Technology Research Institute (ITRI), Taiwan, Republic of China under project number A301AR2100.

6 References

[1] J. Bird and D. Arden, “Indoor navigation with foot-mounted strapdown inertial navigation and magnetic sensors,” *IEEE Wireless Communications*, vol. 18, no. 2, pp. 28-35, 2011.

[2] S. Godha and G. Lachapelle, “Foot mounted inertial system for pedestrian navigation,” *Measurement Science & Technology*, vol. 19, no. 7, pp. 1-9, 2008.

[3] H. Liu, H. Darabi, P. Banerjee, and J. Liu, “Survey of wireless indoor positioning techniques and systems,” *IEEE Trans. Systems, Man, and Cybernetics – Part C: Applications and Reviews*, vol. 37, no. 6, pp. 1067-1080, 2007.

[4] W. C. Bang, W. Chang, K. H. Kang, E. S. Choi, A. Potanin, and D. Y. Kim, “Self-contained spatial input device for wearable computers,” in *Proc. IEEE Int’l Conf. Wearable Computers*, 2003, pp. 26-34.

[5] C. Huang, Z. Liao, and L. Zhao, “Synergism of INS and PDR in self-contained pedestrian tracking with a miniature sensor module,” *IEEE Sensors Journal*, vol. 10, no. 8, pp. 1349-1359, 2010.

[6] Jeen-Shing Wang, Yu-Liang Hsu, and Jiun-Nan Liu, “An inertial-measurement-unit-based pen with a trajectory reconstruction algorithm and its applications,” *IEEE Trans. Industrial Electronics*, vol. 57, no. 10, pp. 3508-3521, 2010.

Development of Test Interface for Test Automation of Automotive Embedded System

Kabsu Han[1]**, Jeonghun Cho**[1]
[1]School of EE, Kyungpook National University, Daegu, Republic of Korea

Abstract - *To certify safety and reliability of automotive embedded system, analysis, verification and validation stages are critical parts in software development process. Analysis of requirements and functionality, embedded software test and embedded system test must be fulfilled on each stage of development process. For testing and test automation, well-defined test interface which support controllability and visibility is mandatory. Although there are many approaches, such as simulation and emulation to test embedded system efficiently, their approaches have a gap between target real hardware and simulated hardware. Therefore we propose a rapid prototyping approach using OCD(on-chip debugger) to perform software testing on a real target microprocessor at early development process. Our proposed approach is independent on programming languages and capable of automating software test process which is time consuming and labor intensive work.*

Keywords: Embedded system testing, Rapid Prototype, Test automation, Testability, BDM

1 Introduction

In 1960, Volkswagen Beetle equipped electric circuits firstly. In 1978, Mercedes S-series equipped embedded systems for ABS. Now, in 2010s, almost a hundred embedded processors are equipped in a Luxury car.

Early automotive embedded system consists of a embedded processor, a sensor and a actuator. And rarely embedded system shares data among different embedded system. Automotive embedded systems get complicated with time, test process is needed to certify safety and reliability and test cases are increased exponentially.

There are various ways to certify safety and reliability in automotive embedded system [1][9]. Analysis : examine software, requirement documents and design to find defects. Verification : evaluate result of each development process to check conditions of its phase. Validation : evaluate the end product to check its requirements. Generally, formal methods, reviews, logical modeling are required to find logical defects at the analysis of requirement phase and the analysis of logical system architecture phase. At the software development phase, testing of software component, software integration test must be needed to assure the software functionality and find errors. After that, embedded software is loaded to embedded hardware, embedded system is integrated finally. But not finished yet. It is important embedded system have to pass system integration test and acceptance test which assures its safety and reliability.

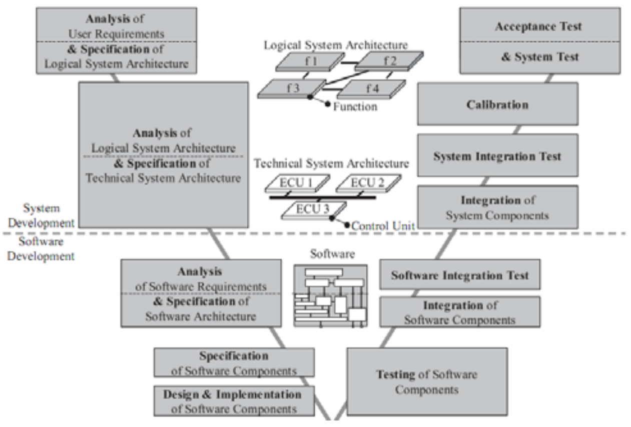

Analysis of User Requirements
& Specification of Logical System Architecture
Logical System Architecture
f 1
f 2
f 3
f 4
Function
Acceptance Test
& System Test
Calibration
Analysis of Logical System Architecture
& Specification of Technical System Architecture
Technical System Architecture
ECU 1
ECU 2
ECU 3
Control Unit
System Integration Test
Integration of System Components
System Development
Software Development
Software
Analysis of Software Requirements
& Specification of Software Architecture
Software Integration Test
Integration of Software Components
Specification of Software Components
Design & Implementation of Software Components
Testing of Software Components

Figure 1 Design flow of automotive embedded system[9]

At least, Controllability and visibility is needed to test embedded software components and embedded system [7]. Controllability means ability to control inputs to software and system under test. Visibility means ability to observe data of software and system under test. It is easy to test software component and software integration on host computer. But, because of lack of controllability and visibility, it is hard to test software and system on embedded hardware. To solve these problems, to support test automation, we propose a test interface using Rapid prototype.

In section 2, we review related work. Section 3 states proposed test interface. Test environment is described in section 4. Finally we give a conclusion and future work.

2 Related Work

Automotive embedded system is a kind of distributed real-time system and needs special test environment which consider these characteristics.

Simulated Hardware approach simulate all of the embedded system include sensors and actuators [3]. This approach can control inputs of embedded software test easily. But there is no actual embedded hardware which is a part of embedded system, we could not observe real data of embedded software. We could not assure safety and reliability of embedded system also.

-Human Resource Training Project for Regional Innovation
This research was financially supported by the Ministry of Education, Science Technology (MEST) and National Research Foundation of Korea(NRF) through the Human Resource Training Project for Regional Innovation

SoC design approach emulate embedded system with SoC based on FPGA [2]. Configurable test approach uses HiL modules to test embedded system and support test automation [5]. These approaches, which use hardware work like embedded hardware, show dependable test result but could not control inputs and observe data easily. However, these approaches are hard to build test environment and need a high cost of testing.

Modeling-in-the-loop approach use embedded hardware, simulated sensors and actuators [6]. This approach can control inputs of embedded system under test and can build test environment easily. But could not observe data of embedded system relatively.

On this paper, we propose test environment, which can control inputs and can observe data of embedded system under test easily, using on-chip debugging interface on rapid prototype.

3 On-chip debugging interface

Most of the embedded system support test interface that allows debugging of embedded system like BDM(Background Debug mode), JTAG(Joint Test Action Group), etc.

In proposed approach, We use HCS12x starter kits, which supports BDM, of Freescale semiconductor, Inc. for automotive rapid prototype [15]. BDM provides control commands like a single step execution, break, reset, etc. Also, can access registers and memory to control inputs and to observe data of embedded system through serial interface [12].

3.1 BDM Basic

Commands are transferred by 10 pins or 26 pins BDM port connector. Each command represents a single 17-bit transfer across the bus sequentially. When BDM is enabled, several signals (BKPT, BGND, Double bus fault) can cause the transition from normal mode to BDM mode which can execute debugging instructions.

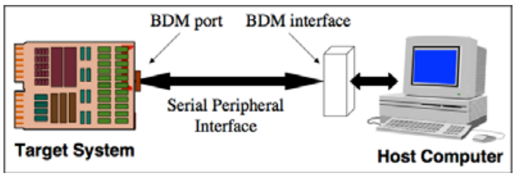


Figure 2 BDM Interface Environment

Table 1 BDM Interface Commands

RAREG/RDREG	read address or data register
WAREG/WDREG	write address or data register
RSREG	read system control register
WSREG	write system control register
READ	read memory
WRITE	write memory
DUMP	read memory block
FILL	write memory block
GO	runCPU
CALL	call user patch code
RST CPU	reset instruction
NOP	null command

3.2 USB-based BDM Interface

In proposed approach, we develop a BDM interface module based on USB. A BDM interface module consist of a MC68HC908JB8 processor and ICs[18]. USB module is integrated on MC68HC908JB8 processor of Freescale semiconductor, Inc. , so we can connect BDM interface to test environment easily. BDM interface assure maximum 100KB/s data transfer rate. BDM interface translate test commands into BDM signals. BDM signals consist of data signals and control signals. After that, transfer BDM signals to BDM port on the embedded system.

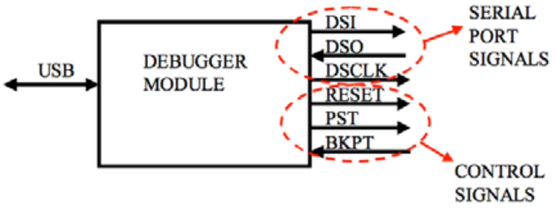


Figure 3 USB-based BDM interface signal

Control software of BDM interface module and BDM APIs are implemented with standard C and C library. BDM APIs run on MS windows, Linux. Also, It is possible import BDM APIs for Java, TTCN-3 with JNI(Java Native Interface).

Table 2 BDM APIs

unsigned char tbdml_read_byte(unsigned int address) - read a byte from memory
void tbdml_write_byte(unsigned int address, unsigned char data) - write a byte to memory
void tbdml_read_block(unsigned int address, unsigned int count, unsigned char *data) - read bytes from memory
void tbdml_write_block(unsigned int address, unsigned int count, unsigned char *data) - write bytes to memory
unsigned int tbdml_read_reg_pc(void) - read data from program counter register
void tbdml_write_reg_pc(unsigned int value) - write data to program counter register
unsigned char tbdml_set_target_type(target_type_e target_type) - set target device (only HC12, HCS08)
unsigned char tbdml_target_sync(void) - sync target device
unsigned char tbdml_target_go(void) - execute code from current program counter
unsigned char tbdml_target_halt(void) - halt debug mode

4 Experimental

4.1 Test Environment

To build test environment with proposed approach, We construct IILS(Intelligent vehicle Interior Lamp System) on Rapid Prototype. IILS is a kind of automotive embedded system which gets inputs from sensors, switches, CAN network and control Interior Lamps on the vehicle.

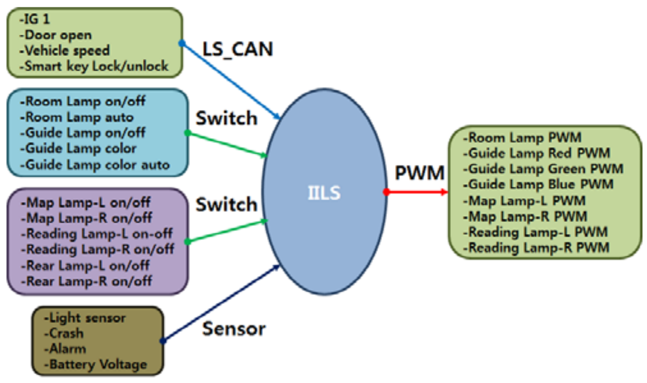


Figure 4 Test Environment

4.2 Test Scenario

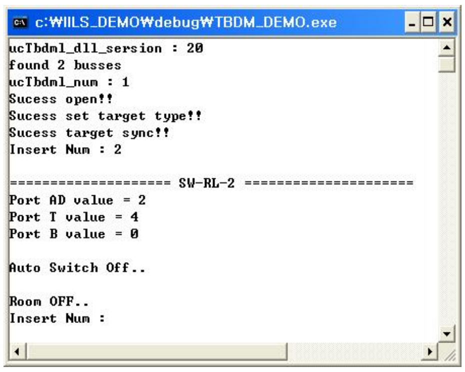
There are a lot of requirements for IILS. We demonstrate test scenarios that focus on Room Lamp control among test scenarios. Test scenarios of IILS for Room Lamp control are from SW-RL-1 to SW-RL-6.

Table 3 IILS Test Scenario for Room Lamp

SW-RL-1	Turn off the Room Lamp when IILS receives Smart key Lock signal
SW-RL-2	Turn off the Room Lamp if Auto switched off
SW-RL-3	Turn off the Room Lamp if Auto switch is on and Light sensor is true, Manual switch is off
SW-RL-4	Turn on the Room Lamp if Auto switch is on and Light sensor is true, Manual switch is on
SW-RL-5	Turn off the Room Lamp if Auto switch is on and Light sensor is true, Manual switch is on, vehicle speed is over 40Km/h
SW-RL-6	Turn on the Room Lamp if Auto switch is on and Light sensor is false, Door Open signal is true

4.3 Test with BDM interface

We implement test scenarios for Room lamp control test with C APIs.



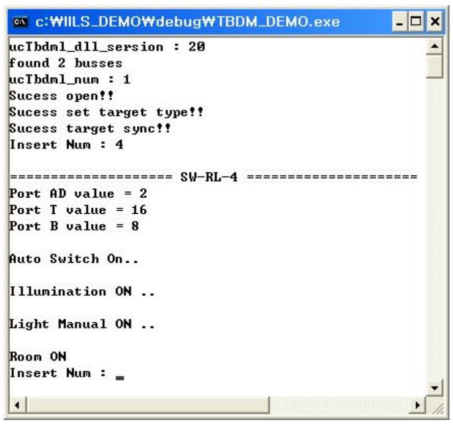
```
ucTbdml_dll_sersion : 20
found 2 busses
ucTbdml_num : 1
Sucess open!!
Sucess set target type!!
Sucess target sync!!
Insert Num : 2

==================== SW-RL-2 =====================
Port AD value = 2
Port T value = 4
Port B value = 0

Auto Switch Off..

Room OFF..
Insert Num :
```

Figure 5 Result of SW-RL-2



```
ucTbdml_dll_sersion : 20
found 2 busses
ucTbdml_num : 1
Sucess open!!
Sucess set target type!!
Sucess target sync!!
Insert Num : 4

==================== SW-RL-4 =====================
Port AD value = 2
Port T value = 16
Port B value = 8

Auto Switch On..

Illumination ON ..

Light Manual ON ..

Room ON
Insert Num : _
```

Figure 6 Result of SW-RL-4

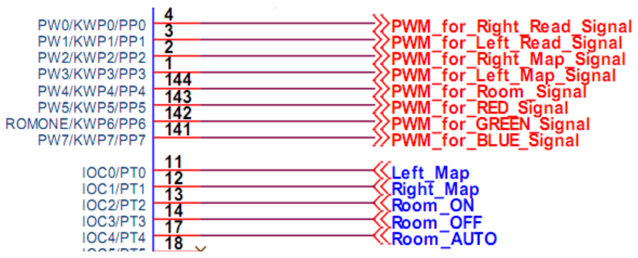


Figure 7 I/O ports for Room Lamp Control

5 Conclusion

We propose a test interface using rapid prototype which support testability with low cost. BDM interface, based on USB, provide simplicity that can build test environment easily and scalability that can control several serial interfaces within bandwidth. BDM APIs provide operability that can run on various test environment. Test interface provide controllability and visibility also.

To support various rapid prototype, We will research JTAG interface and others. Also, research test automation for distributed embedded system. We can build a whole system of vehicle, when we finish construction of the rapid prototype of the sub system of a vehicle.

And, we will research test scenario auto generation with former verification results of the each development phase simultaneously.

6 References

[1] A. Pretschner, C. Salzmann, T. Stauner, “SW engineering for automotive systems”, *ICSE’05*, 2005.

[2] M. Horauer et. Al, “An FPGA based SoC Design for Testing Embedded Automotive Communication Systems employing the FlexRay Protocol”, *Austrochip*, 2004.

[3] Jakob Engblom, Guillaume Girard, Bengt Werner, “Testing Embedded Software using Simulated Hardware”, *ERTS conference*, Toulouse, France, 2006.

[4] Klaus Lamberg, Michael Beine, Mario Eschmann, Rainer Otterbach, “Model-based Testing Of Embedded Automotive Software Using Mtest”, *J. Passenger Cars-Electronic and Electrical Systems*, vol.7, pp.132-140, July, 2005.

[5] Peter Kruse, Joachim Wegener, Stefan Wappler, “A highly configurable test system for evolutionary black-box testing of embedded systems”, *GECCO '09: Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, 2009.

[6] A R Plummer, “Model-in-the-loop testing”, *IMechE 2006*, System and Control Engineering Vol.220, 2006.

[7] Bret Pettichord, “Design for Testability”, *Pacific Northwest Software Quality Conference*, 2002.

[8] Bret Pettichord, “Seven steps to Test Automation Success”, *STAR west*, 1999.

[9] Jorg Schauffele et al., *Automotive Software engineering*, SAE International, 2005.

[10] *CAN Specification Version 2.1. datasheet*, Robert Bosch GmbH., 2006.

[11] *MC68300 Family MC68332 User's Manual*, Freescale semiconductor, Inc., http://www.freescale.com/, 2000

[12] *CPU32 Reference Manual*, Freescale semiconductor, Inc., 1996

[13] *MC68376 User's Manual*, Freescale semiconductor, Inc., 2000

[14] *MCF5206 User's Manual*, Freescale semiconductor, Inc., 1998

[15] *S12XDP512 User’s Manual*, Freescale semiconductor, Inc., 2005

A Difference-Based Data Compression for Daily Activity Signals and Its Realization in an Embedded System

Jeen-Shing Wang, Che-Wei Lin, Yen-Shun Chen, Wei-Hsin Wang
Department of Electrical Engineering,
National Cheng-Kung University
Tainan, Taiwan, R.O.C
jeenshin@mail.ncku.edu.tw

Abstract - *—This paper presents a difference-based data compression algorithm for daily activity signals and its realization in an embedded system. Because daily activity signals have little variances between consecutive data samples, the proposed data compression algorithm compresses a sequence of digital numbers of daily activity signals into an initial value followed by a sequence of difference values. The proposed data compression algorithm had been realized in an embedded system. The realization of the proposed data compression algorithm utilized little hardware resources; only one buffer, one subtraction, one look-up table, and one encoding procedure were utilized. The effectiveness of the proposed data compression algorithm was validated by two experiments. For 11 different daily activities, the average data compression rates for the wrist sensor and ankle sensor were 58.16% and 52.50% respectively. For 24-hour daily activity signals, the data compression rates for the wrist sensor and the ankle sensor were 35.37% and 31.97% respectively.*

Keywords: Data compression; Accelerometer; Daily Activity

1 Introduction

Importance of daily activity monitoring has been gaining much attention in recent years due to the healthy activity styles could help us to increase our fitness levels, improve our health conditions, and prevent diseases [1]. Owing to the advances in the embedded system and sensing technology, accelerometer-based activity monitor were employed in the studies regarding to the quantification of daily activities such as sleep-wake period detection [2]-[4], daily activity detection [5]-[6], energy expenditure [7]-[8], and gait analysis [9]-[11]. From the aforementioned literature, the effectiveness of using accelerometer-based activity monitor has been validated. However, to widely promote the accelerometer-based activity monitor in the commercial market, there are still some technical problems needs to be solved. Hanson *et al.* concluded that the wireless communication, power saving, and data storage are the key factors of widespread adoptions of wearable sensors such as accelerometer-based activity monitor [12]. The issue of data storage became notable in the accelerometer-based activity monitor because large amount of data collected from accelerometer. For example, the data rate will reach up to 720bps and 62.2MB per day when it continuously works at 30Hz sample rate and 8 bits precision by 3 axes. The large amount of data causes the problem of high storage space requirement and high energy consumption. In our literature survey, the data compression of daily activity acceleration can be categorized into the implementing in the embedded system and simulating in the computer. In the field of implementing the data compression algorithms in the embedded system, Tanaka *et al.* employed Golomb-Rice coding in the data compression and reported that the compression rate is typically 30% [13]. Reinhardt *et al.* implemented run-length encoding (RLE) and adaptive Huffman coding (AHC) in an embedded system with tiny OS. Reinhardt *et al.* concluded that applying data compression may allow saving energy, even if additional microcontroller operations are required [14]. In the field of simulating data compression algorithms in the computer, Chen *et al.* employed wavelet transform and adaptive differential pulse code modulation (ADPCM) to compress running and sprinting data sets. The data compression rate in Chen *et al.*'s study is around 40%~50% [6]. Yang *et al.* examined the performance of the state of art compression scheme: the compressed sensing (C-S) framework. The experimental results showed that the C-S framework could effectively reduce the amount of data [14]. Although the state-of-art data compression algorithms such as the wavelet and the C-S compression can provide satisfactory data compression results, to implement those data compression algorithms in an embedded system is a problem due to the limited computing power of the embedded systems. Therefore, this paper aims at developing a hardware-friendly data compression algorithm for daily activity signals that can be easily implemented in the embedded system.

The rest of this paper is organized as follows. In Section 2, we present the hardware platform employed in our daily activity acceleration collection and data compression. In Section 3, the proposed data compression algorithm for daily activity signals is introduced. The results of simulation and hardware implementation of the proposed data compression algorithm are reported in Section 4. Finally, conclusions are given in Section 5.

2 iHelath daily acceleration sensor

A portable embedded system which is responsible for collecting signals and executing data compression called "iHealth daily acceleration sensor" is introduced in this section. The iHealth daily activity acceleration sensor is responsible for collecting signals and executing data compression algorithm in our study. The hardware block diagram of the iHealth daily acceleration sensor is shown in Fig. 1. The hardware consists of a microcontroller (PIC24FJ64GA002), a triaxial accelerometer (MMA7455L), a flash memory (MX25L128), a wireless RF transceiver (Nordic nRF24L01+), a Bluetooth® module (BTM-162), and a power management IC (LTC4063). The sampling rate of the iHealth daily acceleration sensor is 30Hz (f_s = 30Hz), and the signals data are stored in a flash memory with an 8-bit format (Sensitivity: ±8g). The circuit size of the iHealth daily acceleration sensor is 32mm × 30mm × 5mm. The circuit board of the iHealth daily acceleration sensor is enclosed by a case which can be worn on subjects' wrists and ankles to collect the daily activity signals.

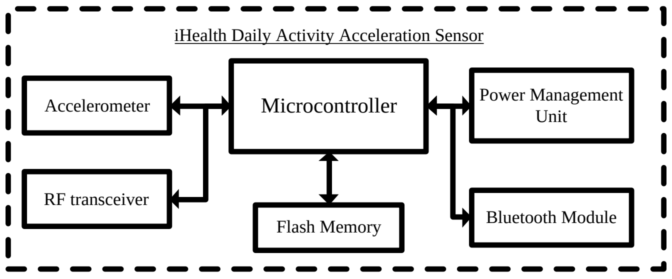


Fig. 1. Block Diagram of iHealth daily acceleration sensor

3 Difference-based data compression algorithm for daily activity signals

In this section, the proposed difference-based data compression algorithm for daily activity signals will be introduced. We will first analyze the characteristics of daily activity signals in Subsection 3.1. In Subsection 3.2, a block diagram of the proposed data compression algorithm will be introduced to explain the detailed procedures. The realization of the differenced-based data compression algorithm for daily activity signals is explained in Subsection 3.3.

3.1 Characteristics of daily activity signals

Daily activities can generate signals during the movement and can be collected by accelerometers. Commercial accelerometers commonly use number of bits to represent the signals they generate in an interval. For example, the sensor described in section 2 generates signals in an 8-bit format to represent signals in an interval (±2g, ±4g or ±8g, user-specified).

The characteristics of daily activity signals are associated with the types of activities. Daily activities can be categorized into four levels: sedentary, light, moderate and vigorous [16]. Signals of the four levels of daily activities exhibit various characteristics, having little variance in sedentary and light activities while varying a lot in moderate and severe activities. However, sedentary and light activities account for the main proportion of a day. Therefore, the proposed difference-based data compression algorithm aims at using less data to represent the sedentary/light daily activities, and further reduce the data size of all-day activity signals.

Table I. Percentage of the Successive data difference smaller than 1 unit in all-day Activity signals

Sensor Placement	Axis	Successive Data Difference (bit)	Percentage (%)
Wrist	*x*-axis	0	55.14
		1	14.33
		-1	14.32
	y-axis	0	61.24
		1	13.26
		-1	13.27
	z-axis	0	58.26
		1	14.82
		-1	14.87
Average (Wrist)		**0**	**58.35**
		1	**14.14**
		-1	**14.15**
Ankle	*x*-axis	0	69.96
		1	9.76
		-1	9.78
	y-axis	0	71.06
		1	9.76
		-1	9.59
	z-axis	0	66.64
		1	12.36
		-1	12.32
Average (Ankle)		**0**	**69.22**
		1	**10.62**
		-1	**10.56**

3.2 Difference-based data compression algorithm

Based on the characteristics observed in all-day activity signals, Table I reveals the percentage of successive data difference in 0, +1, -1 is 58.35%, 14.14% and 14.15% respectively (sensor worn on the wrist), and those on the ankle were 69.22%, 10.62% and 10.56% respectively.

In order to achieve the goal of data compression, we proposed a difference-based data compression algorithm of daily activity signals. Originally, a series of signals are represented by a sequence of digital numbers (8-bit digital number in our study). In the proposed algorithm, a sequence of digital numbers is transformed into an initial value followed a sequence of difference values. Based on the statistical analysis in Table I, the data difference of 0, 1, and -1 accounted for 86.64% and 90.41% in the wrist and the ankle for all-day data respectively. The proposed data compression algorithm uses digital number $(11)_2$, $(10)_2$, $(01)_2$ to denote the successive difference in 0, 1, and -1 respectively. When the data difference greater or smaller than 1 bit, the difference-based data compression algorithm will give a digital number

$(00)_2$, and then we have to put the original value of the acceleration data behind the digital number $(00)_2$. Block diagram of and one example of the proposed difference-based data compression algorithm are shown in Table II and Table III.

3.3 Embedded System Realization of the Difference-based Data Compression Algorithm for Daily Activity Signals

The proposed difference-based data compression algorithm for daily activity signals had been realized as a firmware program in the hardware platform mentioned in the Section 2. The proposed data compression algorithm utilized little hardware resource in the embedded system; only one buffer, one subtraction, one look-up table, and one encoding procedure were utilized.

Table II. Definitions of the operation code in the proposed data compression scheme

d_i	Relations	O_i = OP + INF
0	Current acceleration is equal to previous acceleration	$(11)_2$
1	Current acceleration is greater than previous acceleration in 1 unit	$(10)_2$
-1	Current acceleration is greater than previous acceleration in 1 unit	$(01)_2$
Else	Current acceleration is greater or smaller than previous acceleration in more than 1 unit	$(00)_2 + a_i$

Table III. An example of the proposed difference-based data compression algorithm

i	a_i	d_i	O_i = OP + INF
1	$(00000000)_2$	0	$(11)_2$
2	$(00000001)_2$	1	$(10)_2$
3	$(00000010)_2$	1	$(10)_2$
4	$(00000010)_2$	0	$(11)_2$
5	$(11111010)_2$	Else	$(00)_2 + (11111010)_2$
6	$(11111001)_2$	-1	$(01)_2$
7	$(00000000)_2$	Else	$(00)_2 + (00000000)_2$
8	$(00000001)_2$	1	$(10)_2$
9	$(00000001)_2$	0	$(11)_2$
10	$(11111001)_2$	Else	$(00)_2 + (11111001)_2$

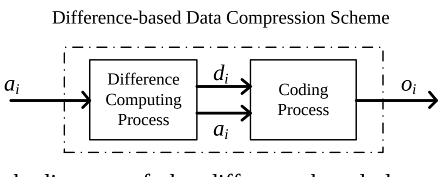

Difference-based Data Compression Scheme
a_i
Difference Computing Process
d_i
a_i
Coding Process
o_i

Fig. 2. Block diagram of the difference-based data compression algorithm

4 Experimental results

The effectiveness of the proposed difference-based data compression algorithm was evaluated in two different experiments. The first experiment was to examine the data compression rate in 11 different activities. The second experiment was to evaluate the data compression rate of the signals from subjects in a 24-hour period. The data compression rate in our experiment is defined as (1).

$$\text{Recogtion Rate} = \frac{\text{Compressed Size}}{\text{Original Size}} \times 100\% \qquad (1)$$

4.1 Data compression rate of daily activity accelerations

In our first experiment, the effectiveness of the proposed data compression algorithm for daily activity signals was examined. Total 18 subjects participated in this experiment; they were asked to collect their daily activity signals. All subjects were asked to conduct 11 different types of daily activities. During the experiment, the subjects were asked to wear two acceleration sensors mentioned in Section II. After all subjects performed the assigned daily activities, the signals were processed by the proposed differenced-based data compression algorithm. The data compression rates of different daily activities are shown in Table IV. There were two different data compression rate corresponding to the signals measured by the wrist activities and ankle activities. For the 11 daily activities, the data compression rates ranged from 25.07% to 101.63% for the wrist sensor, and 25.23% to 111.38% for the ankle sensor. The average data compression rate of 11 daily activities was 58.16% (wrist) and 52.50% (ankles). The results showed that the proposed data compression algorithm achieved a satisfactory performance in the sedentary and light level activities, such as sit and walking, whereas yielded a poor performance in the moderate and severe level activities, such as walking downstairs and running. However, the proposed data compression algorithm still improves the data compression rate of 24-hour daily activity signals since sedentary and light activities are accounted for most of the day.

4.2 Data Compression Rate in Daily Condition

In addition to compute the data compression rates of the 11 daily activities, we also asked the subjects to wear the acceleration sensors and did their daily activities as usual to examine the effectiveness of the proposed data compression algorithm in the daily condition. The signals were processed by the proposed data compression algorithm in the embedded system and the results are shown in Table V. The average data compression rate is 35.37%/31.97% in the wrist/ankle sensor respectively.

TABLE IV. Data compression rate of different activities

Activity Types	**Data Compression Rate (%)**	
	Wrist	**Ankle**
Sit	25.07	25.66
Washing Dishes	46.45	25.23
House Work	67.03	26.49
Cleaning Tables	50.88	43.38
Walking	55.73	93.82
Vacumming	61.41	32.59
Mopping	69.05	30.21
Upstairs	46.82	68.56
Downstairs	69.21	94.98
Running	101.63	111.38
Bicycling	46.45	25.23
Average Data Compression Rate	**58.16**	**52.50**

TABLE V. Data compression rate of 24-hour daily activity signals

Subject	**Data Compression Rate (%)**	
	Wrist	**Ankle**
Average of Data Compression Rate	35.37	31.97

5 Conclusions and future works

Development of a difference-based data compression algorithm for daily activity signals its realization in the embedded system was proposed in this paper. Originally, a series of signals is represented by a sequence of digital numbers (8-bit digital number in our study). In our difference-based data compression algorithm, a sequence of digital numbers is transformed into an initial value followed by a sequence of difference values. The proposed data compression algorithm had been realized in the embedded system and only utilized little hardware resource. The effectiveness of the proposed data compression algorithm was validated by two experiments. For the 11 daily activities, the average data compression rates for the wrist sensor and ankle sensor were 58.16% and 52.50% respectively. For the 24-hour daily activity signals, the data compression rates for the wrist sensor and the ankle sensor were 35.37% and 31.97% respectively. The existing drawback in the data compression of moderate and severe daily activity signals will be further investigated in our future work.

6 References

[1] http://www.americanheart.org/presenter.jhtml?identifier=4563

[2] M. Enomoto, T. Endo, K. Suenaga, N. Miura, Y. Nakano, S. Kohtoh, Y. Taguchi, S. Aritake, S. Higuchi, and M. Matsuura, "Newly developed waist actigraphy and its sleep/wake scoring algorithm"; Sleep and Biological Rhythms, vol. 7, 17—22, 2009.

[3] J. Tilmanne, J. Urbain, M. V. Kothare, A. V. Wouwer, and S. V. Kothare, "Algorithms for sleep-wake identification using actigraphy: a comparative study and new results"; Journal of Sleep Research, vol. 18, 85—98, 2009.

[4] P. Dürr, W. Karlen, J. Guignard, C. Mattiussi, and D. Floreano, "Evolutionary selection of features for neural sleep/wake discrimination"; Journal of Artificial Evolution and Applications, vol. 2009, 1—9, 2009.

[5] Y. P. Chen, J. Y. Yang, S. N. Liou, G. Y. Lee, and J. S. Wang, "Online classifier construction algorithm for human motion detection using an accelerometer"; Applied Mathematics and Computation, vol. 205, 849—860, 2008.

[6] J. Y. Yang, J. S. Wang and Y. P. Chen, "Using acceleration measurements for activity recognition: An effective learning algorithm for constructing neural classifiers"; Pattern Recognition Letters, vol. 29, 2213—2220, 2008.

[7] M. P. Rothney, E. V. Schaefer, M. M. Neumann, L. Choi, K. Y. Chen, "Validity of physical activity intensity predictions by Actigraph, Actical, and RT3 Accelerometers"; Obesity Society, vol. 16, 1946—1952, 2008.

[8] S. E. Crouter, J. R. Churilla, D. R. Bassett Jr, "Estimating energy expenditure using accelerometers"; Eur. Journal of Applied Physiology, vol. 98, 601—612, 2006.

[9] A. Salarian, H. Russmann, F. J. G. Vingerhoets, C. Dehollain, Y. Blanc, P. R. Burkhard, and K. Aminian, "Gait assessment in Parkinson's disease: toward an ambulatory system for long-term monitoring"; IEEE Transactions on Biomedical Engineering, vol. 51, 1434—1443, 2004.

[10] M. S. Angelo, M. Chiara, S. Sergio, and C. Filippo, "Assessment of walking features from foot inertial sensing"; IEEE Transactions on Biomedical Engineering, vol. 51, 486—494, 2005.

[11] H. K. Lee, J. You, S. P. Cho, S. J. Hwang, D. R. Lee, Y. H. Kim, and K. J. Lee, "Computational methods to detect step events for normal and pathological gait evaluation using accelerometer"; Electronics Letters, vol. 46, 1185—1187, 2010.

[12] M. A. Hanson, H. C. Powell Jr., A. T. Barth, K. Ringgenberg, B. H. Calhoun, J. H. Aylor, and J. Lach, "Body area sensor networks: challenges and opportunities"; Computer, vol. 42, 58—65, 2009.

[13] T. Tanaka, S. Yamashita, K. Aiki, H. Kuriyama, and K. Yano, "Life Microscope: Continuous daily-activity recording system with tiny wireless sensor"; in Proc. Networked Sensing Systems, 162—165, 2008.

[14] Andreas Reinhardt, Delphine Christin, Matthias Hollick, Ralf Steinmetz, "On the energy efficiency of lossless data compression in wireless sensor networks"; in Proc. of the 4th IEEE International orkshop on Practical Issues in Building Sensor Network Applications, 873—880, 2009.

[15] S. Yang and M. Gerla, "Energy-efficient accelerometer data transfer for human body movement studies"; in 2010 IEEE Int. Conf. on Sensor Networks, Ubiquitous, and Trustworthy Computing, 304—311, 2010.

[16] B. E. Ainsworth, W. L. Haskell, M. C. Whitt, W. L. Irwin, A. M. Swartz, S. J. Strath, W. L. O'Brien, D. R. Bassett, K. H. Schmitz, P. O. Emplaincourt, D. R. Jacobs, A. S. Leon, "Compendium of physical activities: an update of activity codes and MET intensities"; Medicine and Science in Sports and Exercise, vol. 32, S498—504, 2000.

GSM-based Embedded Water Meter System

Hesham H. Hallal[1], May Haidar[2], Taha Barake[1],
Sami AlKhaldi[3], Mohammad AlOrayfij[3], Amal AlBalawi[2], Raneem Aljehani[2]
[1]Department of Electrical Engineering
[2]Department of Computer Science
[3]Department of Computer Engineering
Fahad Bin Sultan University, Tabuk, Saudi Arabia P.O.Box 15700, Tabuk 71454

Abstract - *We present an embedded system implementation of a wireless water meter system. The deployment of the proposed system uses the existing GSM network, where the water meter system can send its readings directly to a server application using a GSM modem. The application itself can notify subscribers of their bills using SMS messages as well.*

Keywords: Embedded Systems, Water Meter, Smart Home, GSM

1 Introduction

Water resources around the world are getting scarcer day after day. Climate, global warming, and irresponsible usage are major factors the make the situation even harder.

In the absence of any real solution for global warming, governments are putting large efforts to compensate for the shortage of water either through awareness campaigns to reduce consumption or through more taxes on usage of water. While raising awareness is usually a long process that requires a lot of resources, increasing usage fees is highly dependent on the availability of human resources to make measurements and collect appropriate fees.

An alternative, more practical, solution consists of reusing existing technologies that are already deployed in other fields, where the results are promising. The main idea is to customize existing solutions to the context of water billing and usage management. The proposed approach in this paper relies on two main technologies that have made significant contributions to wellbeing of individuals and societies: Application specific embedded systems and the Global System for Mobile Communications (GSM). The latter has been increasingly used in different applications ranging from phone calls to Internet browsing and remote control of electronic, electrical, and mechanical devices [7]. Meanwhile, applications based on embedded systems are also being introduced almost in every domain, especially for purposes of cost reduction and portability. Examples of such applications include watches, microwave ovens, cars, digital cameras, and security systems. In additions, networks of embedded wireless sensors are being used in many [2] domains like weather forecasting and smart home development.

In this paper, we propose a system to automate the billing of the consumption of water and the control of the water meters using a blend of both technologies: GSM networks and embedded systems. The proposed system consists of three main components:

1. Embedded water meter (E-WATER) system for measurement and control of water consumption.

2. Server application to manage the measurements and prepare invoices and bills. In addition to that, the application performs some predefined control operations that can be transmitted to the embedded water meter.

3. Communication medium that is based on the existing GSM networks. The water billing and control operations will be performed simply using the Short Messaging System service that is available over GSM. For this, no modification or even customization is needed in the networks themselves.

Automating the billing process remains an appealing objective especially with tendency of governments in many countries to go electronic (paperless). Some proposals already exist to automate the billing of some basic services like water, gas, phone, and electricity. In [1], a proposal is presented to combine in one meter the measurements of all the services needed for a house. The proposed system relies on a microcontroller for the readings but requires the user to pay on site with his credit card. This means the meter system needs to host the hardware and software necessary to complete a credit card transaction. In addition, the proposed system does not allow any remote control of the meter of any sort.

Recently, some implementations of remote water meter systems have reached a commercial level like in the case of [5], where a mixed RF/GPRS network is used to convey the readings of the meter to the billing center. The use of GPRS for communications makes possible to access meter information online from any place with Internet access. The use of both RF and GPRS in the same network for the reading system adds to the complexity of the system, especially in terms of adapting the protocols on both sides. Another commercial example of water meter systems is the IkTech [6], which involves using a point to point communication system between the meter and a mobile reader from which collected readings can be downloaded to a central system with a billing/management application. The proposed system still requires human intervention to do the reading though not from each and every meter in a large network. In additions to the limitations mentioned in both cases, the idea of controlling the meter remotely is absent, which makes the system proposed in this paper a proper solution. In particular, the possibility of controlling the meters remotely adds to the range of services that could be offered along with the system. For example, it is possible control the water supply to regions hit by disasters or contamination. In addition, services like suspending the account or reducing/increasing supply become possible.

The remainder of this paper is organized as follows. Section 2 describes the architecture of the proposed system. Section 3 describes the embedded system implementation of the water meter. Section 4 presents the software application of the system. Finally, Section 5 concludes the paper.

2 Architecture of the GSM-based Water Billing System

Figure 1 shows the architecture of the automated water billing system. The system consists of three main components:

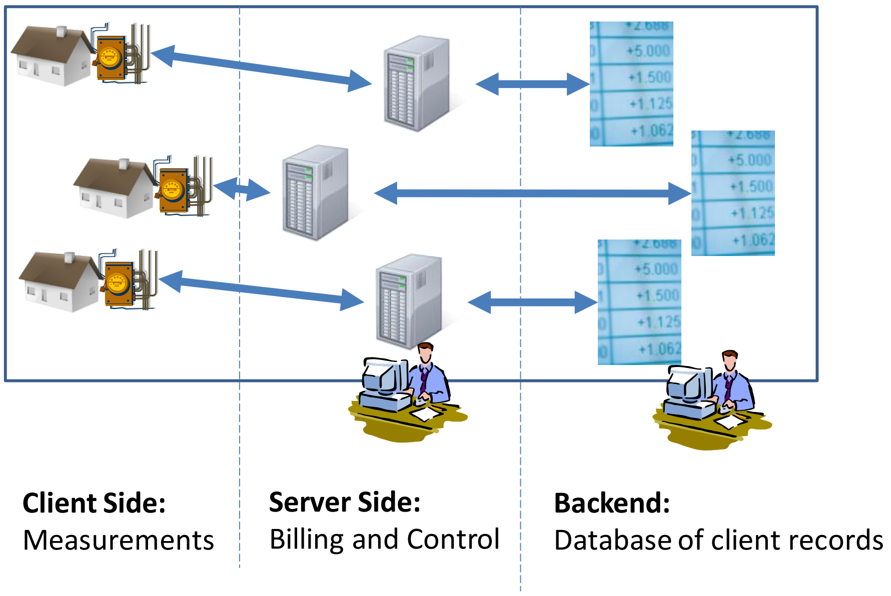


Figure 1. Architecture of the automated water billing system.

1. Embedded water meter system [2] has the responsibility of providing the reading and filtering of data while receiving control commands and executing them. We denote this system E-WATER for Embedded WAter meTER.

2. Server application has the task to receive the readings from the E-WATER system and process the information by accessing a database at the backend. The application on the server has two main modules:
   a. Database access module that handles the records of different E-Water systems.

   b. Control module that receives instructions from a user, translates them into commands, and transmits the commands to the E-WATER system, which executes them on site. Such commands might include closing the supply and changing the type of reading or its frequency.

3. GSM network for communication between the E-Water systems and the server application. The communications is possible between the server applications and the subscribers directly by sending them SM messages about their consumption and bills.

The communication between the E-WATER system at the client side and the server application is carried out over the existing GSM networks using the SMS standard/protocol. In the following, we detail the description of the components of the proposed system.

3 E-WATER System

The E-WATER system, Figure 2, consists of a water flow sensor (e.g., Signet 515 Rotor-X Flow sensor [4]), microcontroller, memory, LCD display, GSM modem, and a power supply.

Using a microcontroller, e.g., PIC16F876A [3] which is programmable in both assembly and C language, is driven by the need to start managing the billing at the client side and to control the flow of water. The former includes changing the reading process from time to time, displaying preliminary information on the bill to the client. The interface of the E-WATER system with the external world, mainly the server application, is implemented using the GSM modem (interfaced to the microcontroller using a MAX232 converter) that is capable of sending SMS messages in which the microcontroller can report the readings and related data. The F1003 modem is a primary for its availability and relatively reduced cost. The control part of the operation, meanwhile, can be initiated either by the client or by the user of the software application. In either case, the GSM modem will receive a command to change the status of the E-WATER system. Examples of such control operations include: shutting down the supply for emergency reasons, minimizing the flow for payment problems, changing the frequency of sending the readings to the station, and changing the flow based on the request of the user. In general, this control process is divided into two parts. In the first, the software application generates a command and sends it as SMS to the GSM modem of the E-WATER system. In the second phase, the microcontroller converts the control command into signals that pass through actuators and physically change the status of the flow sensor.

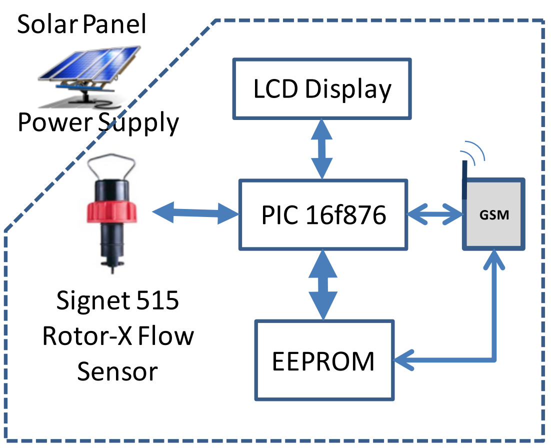


Figure 2. Design of the E-WATER system.

In order to optimize the use of the GSM modem and control the number of SMS messages exchanged, the microcontroller uses a memory unit, the 24LC16B EEPROM, to save the readings obtained from the flow sensor. Based on the type of program executed by the microcontroller it can select specific data from the memory unit and send it to the server via the GSM modem at different intervals. In addition, the microcontroller drives an LCD display to show the readings at a programmable pre-defined rate. To save power, the display will be off by default unless selected otherwise. The power supply of the E-WATER system is a combination of long term battery and a solar panel based supply. The solar power supply is the default source of power with the battery acting as backup for when the sun light is reduced or at night.

4 Server Application

The software application has two main functions:

1. Processing the readings of the E-WATER system received via GSM communication. The main objective is billing and statistics. The main role of this module is to process the readings of the E_WATER system and prepare bills and reports on the types and amounts of usage for different clients. A database is designed to host the records for the different clients. Figure 4 shows the Entity Relationship (ER) diagram of the proposed database. The proposed database consists of four main tables: Customer, Invoice, Reading, and Rate.

The proposed database structure will be distributed to areas of supply. This approach keeps the databases of manageable sizes and allows for ease of interface. In addition, the distributed approach ensures efficiency in using the existing communication networks. Currently our objective is to respond to the need for a data storage model where different records of clients can be easily accessed. In the future, the proposed database can be extended to include variety of data that is useful to different types of reports like the area where the client is located, the fines a client receives for delayed payments, as well as various other records belonging to the same client.

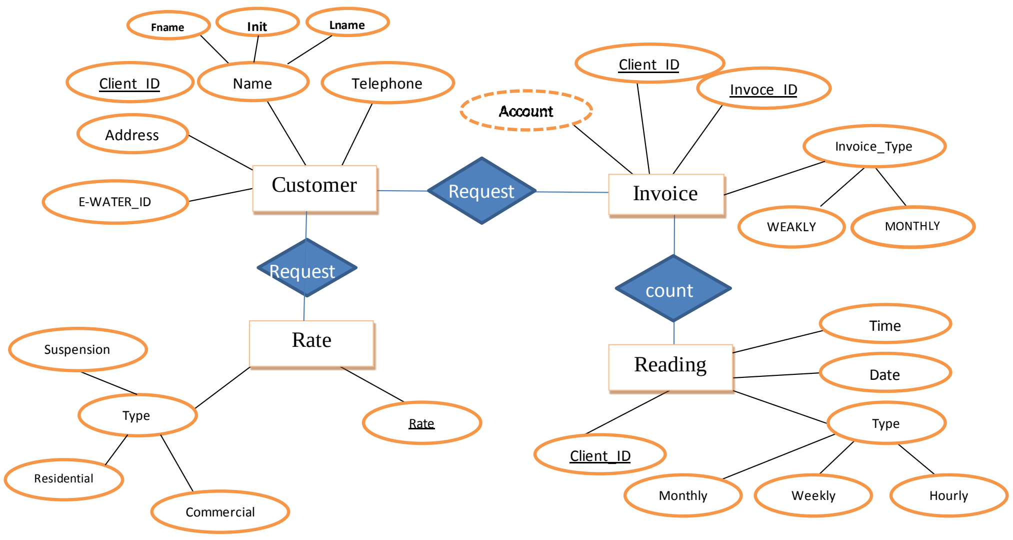


Figure 3. Entity Relationship diagram of the client DataBase.

2. Controlling the E-WATER system based on predefined commands executed a user who can control the E-WATER system remotely with some predefined commands that should be used only in specific situations as discussed earlier in Section 3. These commands include two main sets:

   a. Managing the water supply such as turning it off upon the client's request.

   b. Changing the frequency of readings; e.g., from hourly to daily.

5 Conclusion

We presented a system for water billing and consumption management based on an embedded system implementation of the water meter that is capable of reporting readings over the GSM network to a server application capable of billing and of controlling the meter itself.

The proposed system can be extended as follows: other sensors for other services (telephone, electricity, and gas) can be included in the E-Water system. Another future step is to evaluate the use of the GSM network databases and billing procedure instead of building a separate database and server application. This makes the water invoice/bill look like just adding a new phone bill for the customer.

Acknowledgement: We acknowledge the contributions of Ahmad Hattab, Firas Hoteit, and Farah Kourdi.

References

[1]. Al-Qatari, S. A.; Al-Ali, A. R. Microcontroller-based Automated Billing System. In Proceedings of International IEEE/IAS Conference on Industrial Automation and Control: Emerging Technologies, Taiwan, 1995.

[2]. Peter Marwedel, Embedded System Design. *Kluwer Academic Publishers*, Nov. 2003, ISBN 1-4020-7690-8

[3]. www.microchip.com

[4]. www.panner.com/assets/pdf/**Signet515**.pdf

[5]. http://www.baylanwatermeters.com/en/fixed-remote-meter-reading-systems.php

[6]. http://www.lktech-ic.com/elongkang/product_detail.asp?picid=21

[7]. Joerg Eberspaecher, Hans-Joerg Voegel, Christian Bettstetter. GSM Switching, Services, and Protocols, John Wiley & Sons, 2001.

An Embedded Platform for Patient Monitoring and Care System

Vivek Kumar Sehgal[1*], Nitin[2], Shubhrangshu Naval[3], Abhinav Gulhar[4], Sayed Jeeshan Ali[5] and Mudit Singhal[6]
[1,3,4,5]Department of Electronics and Communication
[2,6]Department of Computer Science and Engineering
Jaypee University of Information Technology
Waknaghat, Solan, HP (INDIA)
e-mail: vivekseh@ieee.org, vivekseh@acm.org

Abstract - *This paper proposed a prototype that analyses various Bio-medical parameters like temperature and heartbeat obtained from the sensors used &with the help of a microcontroller, all the parameters obtained are displayed on an LCD screen. Based on the parameters obtained the patient is continuously monitored and if in case of any critical mishap when the parameters go out of a particular range then it is prevented by the care system attached to the Patient Monitoring System. The primary function of this system is to sense the temperature and heartbeat of the patient and sensed data is sent to the embedded processor port. The processor is programmed to continue monitor the data and send the actuation signals to patient care system.*

Keywords*:* sensors, microcontroller, bio signals, medicine injection system

1 Introduction

It has been long recognized in the health care industry that long-term, continuous monitoring is a key element in preventive care for people with chronic conditions such as cardiovascular disease. A typical example of patient monitoring is a home care device, such as an electronic blood pressure or glucose meter. An ambulatory system that allows long-term monitoring of mobile patients is also desirable. The ambulatory electrocardiogram (ECG) Holter device, used since the 1960s, provides a reliable measurement of the wearer's heartbeat but is heavy and cumbersome to wear over an extended period of time. In addition, its substantial power consumption forbids continuous operation using low-capacity batteries. In recent years, lightweight devices have emerged as a viable technology for continuous measurement of vital biomedical parameters [7]. Wearable, biosensors connected to self-organizing allows physicians to continuously monitor vital signs, and helps in preventing any critical mishap and also helps physicians to record long-term trends and patterns that provide invaluable information about a patient's ongoing condition, ease of Use [1]. The availability of advanced

Fig.1. Patient Monitoring and Care Systems

sensing devices combined with sophisticated; self-organizing care system will enable new applications and represents a significant opportunity for remote health monitoring. This system will serve 3 requirements

1) The first is a portability factor so that these health monitoring devices can fit or attach easily to a wrist or arm band, ring sensor or other wearable or implantable device.
2) The second requirement is extremely low power so that small batteries can be used for an extended period of time.
3) The third requirement is a highly sophisticated protocol for low latency, high scalability and high responsiveness.

The organization of paper is as follows:
Section II describes the block diagram. Section III gives the detail of component used in proposed system. Section IV and V overviews the software and hardware parts of this proposed work. Finally some conclusion and future scopes are drown from the work done in section VI.

2 Description

Various biometric signals are sensed by the sensors and sensed signals are conditioned through signal conditioning circuits. After getting the appropriate shape and value these bio signals are converted in to digital signals for processing. Embedded processor continues monitors these bio signals and display their values on LCD times to time. Any variation in these signals makes processor to send the actuating signals to patient caring system as shown in Fig. 2.

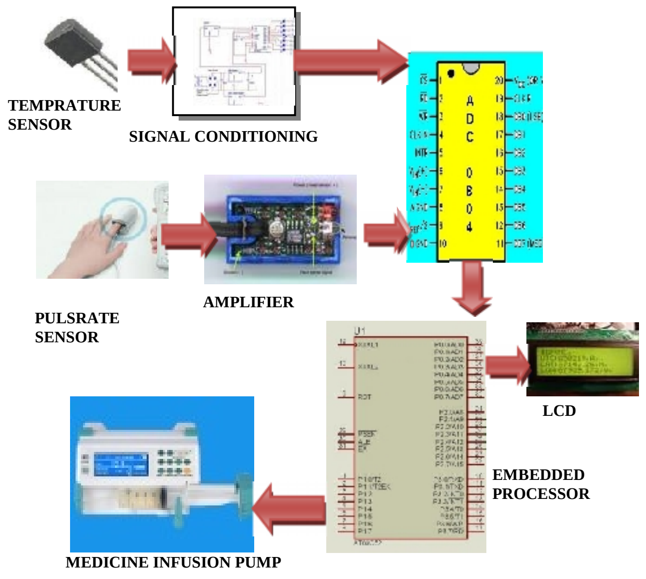


Fig.2. Block Diagram

3 Components Used

The whole system is composed of three components sensing, processing and actuating. Apart from these components, the signal conditioning circuit, local display system and programmed algorithm are also integrated part of this application specific embedded system. A few components are discussed below

3.1 Microcontroller

89c52 from ATMEL is being used. The AT89C52 is a low-power, high-performance CMOS 8-bit microcomputer with 8K bytes of Flash programmable and erasable read only memory (EPROM). The on-chip Flash allows the program memory to be reprogrammed in-system or by a conventional nonvolatile memory programmer. We use this because it reduces the amount of external hardware or internal software necessary to process the sensory data. Functions of micro controller in this prototype are:

1) Used to display Biomedical Parameters on LCD.
2) It is also used to interface the temperature sensor, the heartbeat sensor, the LCD and the Infusion Pump.

3.2 Liquid Crystal Display

Display used here is the LCD display. It is an intelligent LCD. It is a 16*2 LCD, which displays 32 characters at a time 16 will be on the 1st line and 16 will be on the 2nd line. There are two lines on the LCD and it works on extended ASCII code i.e. when ASCII code is send it display it on the screen. On the LCD total no of pins are 16 out of which 14 pins are used by the LCD and 2 are used for backlight. LCD is an edge trigger device i.e. from high to low. The data can also be monitored on mobile devices using DTMF [4].

3.3 Temperature Sensor

In this a precision centigrade temperature sensor LM35 is used. It is a precision integrated-circuit temperature sensor, whose output voltage is linearly proportional to the Celsius (Centigrade) temperature [2]. The LM35 thus has an advantage over linear temperature sensors calibrated in ° Kelvin, as the user is not required to subtract a large constant voltage from its output to obtain convenient Centigrade scaling. For every ^{0}C change in temperature, it shows a variation of 10mV in the output [3].

3.4 Heart Beat Sensor

A Heart Beat Sensor is implemented with a pair of LED and LDR. (Fig.3). This transducer works with the principle of light reflection, in this case the light is infrared.

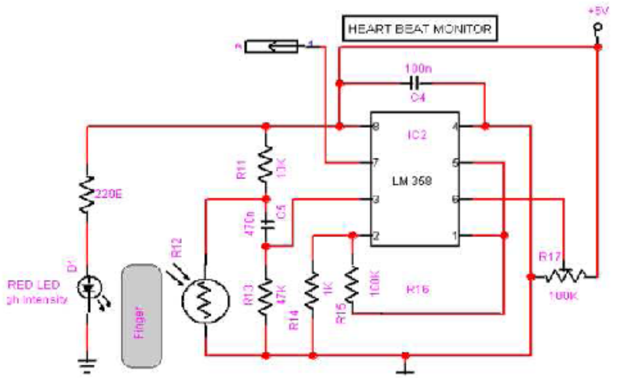


Fig.3. Heart Beat Sensor

3.5 Medicine Injection System

In this part we have implemented syringe to a DC motor with help of a screw such that, when the relay is switched on, the DC motor starts, which in turn moves the screw n the screw changes the rotatory motion of the motor into linear motion which moves the piston of the syringe back & forth.

4 Software Implementation

The software design is a key element in the development of a project. For visualization of the different parameter on the LCD display, the microcontroller is burnt in assembly level language. The microcontroller chosen for the development of the system is Atmel89c52. The Atmel89c52 has 8K bytes of Flash programmable and erasable read only memory (EPROM) and has the capability to write to its own memory. The use of a FLASH device for development also provides the option to use FLASH microcontrollers in the final design making the system fully upgradable. This allows modification of the microcontroller software to expand.

Software Code.

```
;-TEMP LOGER..--- -
;LCD at PORT1
;ADC at PORT0
; ------------BELOW 20 DEGREE------COLD
; ------------UPTO 35 DEGREE------WARM
; ------------ABOVE 35 DEGREE------HOT
org 0000h
mov a,#38h ;initialise two line 5x7 matrix
35
acall command ;sub routine
mov a,#38h ;initialise two line 5x7 matrix
acall command ;sub routine
mov a,#0ch ;display on,cursor blinking
acall command ;sub routine
mov a,#01h ;clear lcd
acall command ;sub routine
mov a,#80h ;shift cursor TO 1st line
acall command ;
---Temperature----
mov a,#'P'
acall data1
mov a,#'a'
acall data1
mov a,#'t'
acall data1
mov a,#'i'
acall data1
mov a,#'e'
acall data1
mov a,#'n'
acall data1
mov a,#'t'
acall data1
mov a,#0c0h ;shift cursor TO 1st line
acall command ;
mov a,#'M'
acall data1
mov a,#'o'
acall data1
mov a,#'n'
acall data1
mov a,#'i'
acall data1
mov a,#'t'
acall data1
mov a,#'o'
acall data1
mov a,#'r'
acall data1
mov a,#'i'
acall data1
mov a,#'n'
acall data1
mov a,#'g'
acall data1
acall delay2
acall delay2
acall delay2
acall delay2
--Temp--
mov a,#01h
acall command
mov a,#80h
acall command
mov a,#'T'
acall data1
mov a,#'e'
acall data1
mov a,#'m'
acall data1
mov a,#'p'
acall data1
mov a,#'.'
acall data1
mov a,#20h
acall data1
mov a,#'i'
acall data1
mov a,#'s'
acall data1
mov a,#0c0h ;shift cursor TO 1st line
acall command ;
mov a,#'H'
acall data1
mov a,#'/'
acall data1
mov a,#'R'
acall data1
mov a,#'='
acall data1
acall delay2
acall delay2
acall delay2
acall delay2
-----ADC-----
mov p0,#0ffh
go:
setb p2.5
clr p2.5 ; INTR=p2.6 ; start conversion
setb p2.7 ; WR = p2.7
; RD = p2.5 active low
hee:jb p2.6,hee
acall delay2
clr p2.5
mov a,p0 ; a contain temp in hex
MOV 40H,A
-----HEX to BCD conversion-
lop: cjne a,#35d,next ; if a is smaller carry=1
sjmp next
next: jnc gom
setb p3.7 ; below 30 led on
40
sjmp hoi
gom:
MOV A,40H
cjne a,#45d,next2 ; if a is smaller than 35 c=1
sjmp next2
next2: jnc gom2
clr p3.7
SJMP HOI
gom2:
setb p3.7 ; above 40 led off
hoi:
MOV A,40H
mov b,#10d
div ab
mov r6,b ; 0ne
mov b,#10d
div ab
mov r7,b ; tens
mov r2,a ;hundred
mov a,#89h ;shift cursor TO 1st line
acall command ;command subroutine
mov a, r2
orl a,#30h
acall data1
mov a, r7
orl a,#30h
acall data1
mov a, r6
orl a,#30h
acall data1
mov a,#20h
acall data1
mov a,#27h
acall data1
mov a,#'C'
acall data1
setb p2.0
jb p2.0,kou
mov a,#0c4h ;shift cursor TO 1st line
acall command ;
42
mov a,#'7'
acall data1
mov a,#'5'
acall data1
acall delay2
acall delay2
acall delay2
acall delay2
acall delay2
acall delay2
acall delay2
mov a,#0c4h ;shift cursor TO 1st line
acall command ;
mov a,#'7'
acall data1
mov a,#'3'
acall data1
acall delay2
acall delay2
acall delay2
acall delay2
acall delay2
acall delay2
acall delay2
mov a,#0c4h ;shift cursor TO 1st line
acall command ;
mov a,#20h
acall data1
mov a,#20h
acall data1
;---- -
kou:
Ljmp go
;------
delay1:
mov r3,#150d
h130: mov r4,#150d
h230: djnz r4,h230
djnz r3,h130
ret
delay2:
mov r3,#255d
h1300: mov r4,#255d
h2300: djnz r4,h2300
djnz r3,h1300
ret
delay:
mov r3,#60d
h13: mov r4,#40d
h23: djnz r4,h23
djnz r3,h13
ret
command:
mov p1,a
clr p3.2
clr p3.1
setb p3.0
clr p3.0
acall delay1
ret
data1:
mov p1,a
setb p3.2
clr p3.1
setb p3.0
clr p3.0
acall delay1
ret
END
```

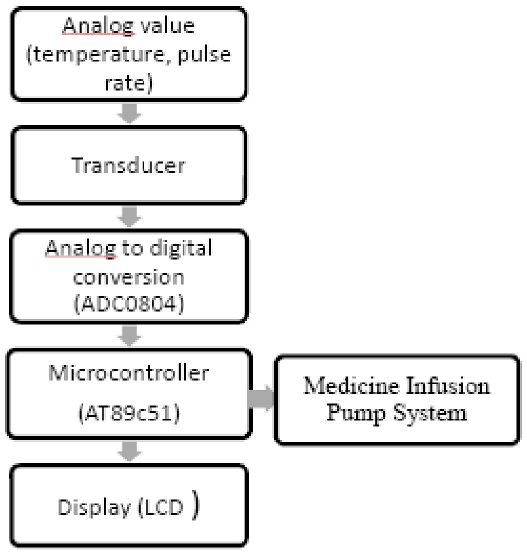

Analog value (temperature, pulse rate)
Transducer
Analog to digital conversion (ADC0804)
Microcontroller (AT89c51)
Medicine Infusion Pump System
Display (LCD)

Fig.4. Flow Diagram

The proposed work is focused on the body temperature measurement device and heart rate measurement monitor, taking up the analog values using the sensor LM35 and LDR and LED; these signals were fed into an ADC (Analog to Digital Convertor) ADC0804. The digital value of the temperature measurement and heart rate measurement from the ADC is then fed to the microcontroller (AT89c52). The LCD (Liquid Crystal Display), is interfaced with the microcontroller which displays the value of the temperature sensed and the heart beat. In case the values of the temperature and the heart beat goes out of a particular range prescribed by the doctor the relay gets triggered and hence the Care System responds accordingly as shown in Fig.4.

4.1 Software Code

The microcontroller along with its various interfaces requires software to work on. The logic involved in achieving the desired operation has been carefully prepared and is noted down in form of software code. The Software Code is in the form of assembly language.

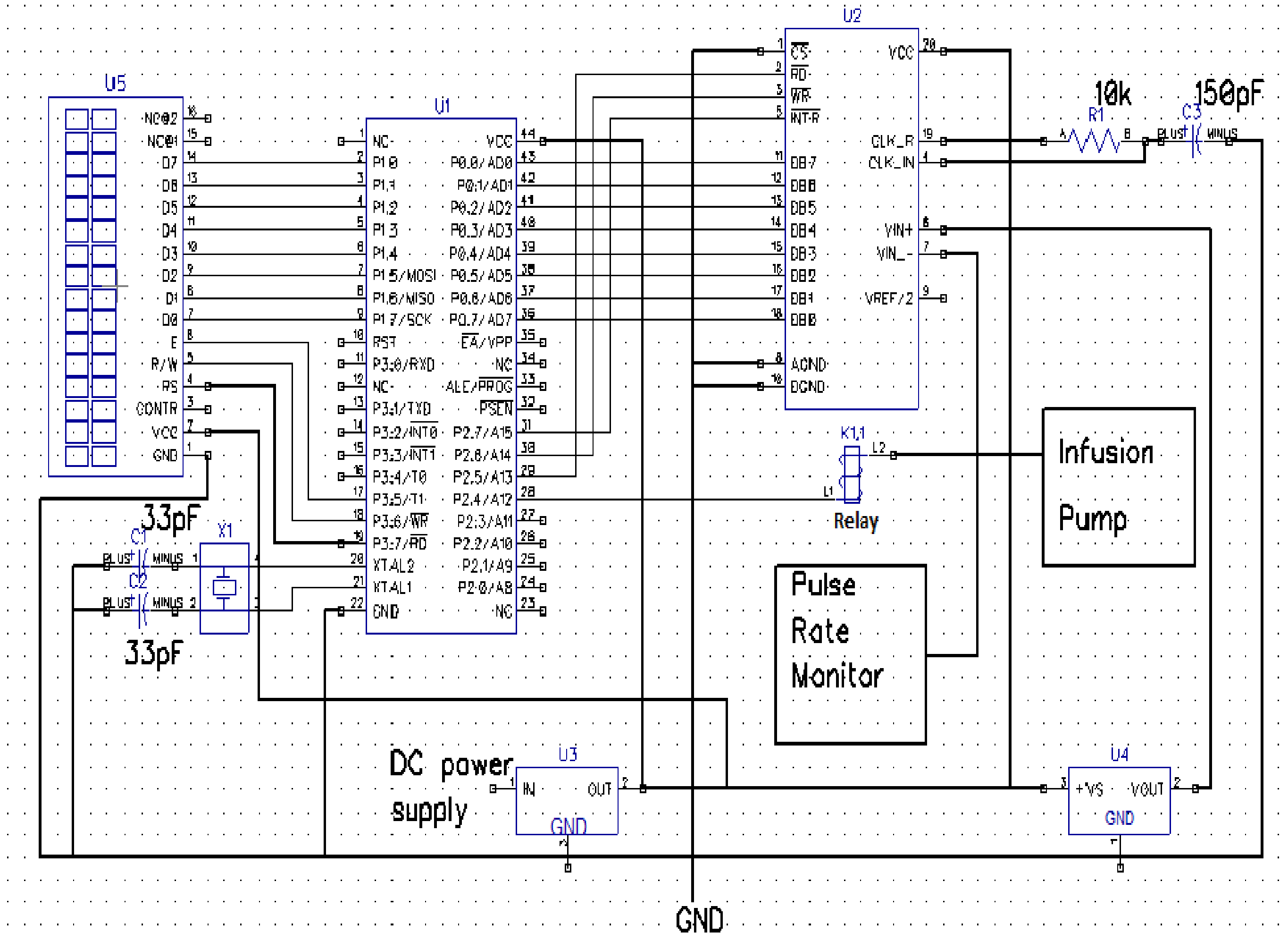


Fig.5. Schematic Diagram

5 Hardware implementation

By using various electrical circuits the bio-medical like temperature and heart beat parameters can be found. The output of the circuits is amplified by means of an amplifier and fed into an A/D converter. The digitized signal is then fed into the input port of the microcontroller. The microcontroller displays the parameters in digital value in the display device. And the injector connected to the prototype works accordingly as shown in Fig. 5. Hardware implementation of proposed work is shown in Fig.6.

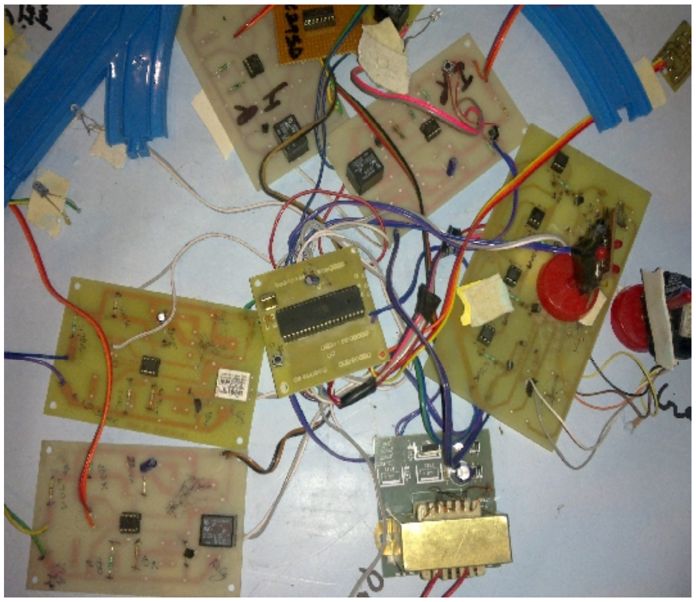


Fig.6. Hardware Implementation

6 Conclusion and Future Scopes

The project has been successfully completed within the stipulated time frame with the prototype displaying bio-medical parameter and the Care System i.e. the Infusion Pump working accordingly. We have achieved the desired outputs of the body temperature and the heartbeat of the patient on the LCD displays and according to which the Care System i.e. the Medicine Infusion Pump performs if these parameters go out of a particular set range. Despite lots of research in this field of Monitoring and Care of patient, there has been very little effort in actual implementation of the concept which provides ample scope for the further developments of this project. Over the past few decades, technology has touched lives, literally. While use of technology in healthcare has been made in a hospital environment, a larger scope lies for technology to become simple. The complete system can be condensed in to a SoC by using on-chip network [5][6].

Patient Monitoring and Care today is fast becoming a common reality. From Cardiac Monitoring to Diabetes Management and more, healthcare services that were once restrained within doctors being around the patient 24 hours are now finding their spot under technologically sound and improved healthcare. That's a win-win for both doctors/caregivers and patients. Patient Monitoring and Care makes objective, pertinent information available to caregivers in a timely manner, or as and when the need arises, prevent any kind of critical disaster to occur. This way, the patients are taken care of and the doctors are able to perform their job effectively too. Also, this addresses the issue of ever-less-available resources like healthcare staff and physical presence of the doctor. Additionally, it helps improve patient health, thanks to early diagnosis and preventive care.

7 References

[1] Michael, M., Djamal, B.,Chirine, G., and Zakaria, M. "A Mediation for Web Services in a Distributed Healthcare Information System", Proceedings of the IDEAS Workshop on Medical Information Systems: The Digital Hospital, pp. 15- 22, Sept. 1-4, 2004.

[2] V.K. Sehgal, Nitin, D.S. Chauhan, R. Sharma," Smart Wireless Temperature Data Logger Using IEEE 802.15.4/ZigBee Protocol", IEEE Region 10 Conference TENCON 2008, PP 1-6.

[3] V.K. Sehgal, N. Chanderwal, R. Sharma, V. Hastir, Y.S. Dadwhal, M. Bansal, R. Puri, S.A. Pathania, and A. Thakur, "Smart Wireless Temperature Data Logger", in Proc. ESA, 2008, pp.140-144.

[4] V.K. Sehgal, N. Chanderwal, S. Garg, A. Jain, N. Shah, and S. Gupta, "RJ-11 Interfaced Embedded Platform for DTMF Based Remote Control System", in Proc. ESA, 2007, pp.239-239.

[5] V.K. Sehgal and D.S. Chauhan, "State observer controller design for packets flow control in networks-on-chip", presented at The Journal of Supercomputing, 2010, pp.298-329.

[6] V.K. Sehgal, N. Chanderwal, and D.S. Chauhan, "A New Approach for Inter Networks-on-Chip Communication in Networks-in-Package", in Proc. ESA, 2008, pp.16-23

[7] Schwaibold, M., Gmelin, M., Wagner, G., Schochlin, J., and Bolz, A. "Key factors for personal health monitoring and diagnosis devices". Workshop of Mobile Computing in Medicine, Germany, 2002, pp. 143-150.).

Flexible Embedded System Design Using Flowpaths

Darrin M. Hanna, Bryant Jones, Lincoln Lorenz, and Mark Bowers

Abstract—**A typical hardware system can be segmented into components that each define a specific task. Components could be divided among a design team for creation, acquired from another source in the form of legacy work, or an IP core. A finalized system could be composed of several heterogeneous components. Proper integration of these components can often be complex and time consuming for designers. Available tools to aid in rapid hardware design often lack in this area and require confusing pre-compilation procedures. Previous work has been done to create a compiler, called flowpaths, for converting high-level stack-based languages (e.g. Java) to VHDL for use on an FPGA or ASIC. Introduced in this paper is an extension to the flowpaths compiler to allow easier integration of system-components using object-oriented methodologies. System-components can be described by high-level Java classes with methods for interactions with components. These methods are filled with the custom system-components during generation, such that flowpaths acts as the glue logic between separate components. In comparison to handwritten component interconnections, a design integrated with flowpaths shows a decrease in implementation time and design complexity.**

Index Terms—Field Programmable Gate Arrays, Program Compilers, Embedded Systems, Glue Logic

I. Introduction

Often times, large hardware systems are composed using a combination of several heterogeneous components. Usually these components are legacy-based, created by different designers, or acquired from a vendor in the form of an IP core. Integration of several such components into a cohesive system is often referred to as “glue logic”. Design of a system using a collection of modules such as these can be quite a difficult task. This can become very time consuming, even for a skilled computer engineer. The need arises for a tool to add flexibility to the design and implementation phases of a multi-component system.

Tools have been created to allow a hardware designer flexibility of design through the use of high-level languages. Examples of such tools include Handel-C and flowpaths; the latter being designed by us. Flowpaths is an architecture described in HDL that can be generated using a stack-based language, rather than a variable/register language which often introduces large fan-out and delay when implemented in hardware. Such stack-based languages include Java bytecode, Common Interface Language (CIL), and Forth. Currently, flowpaths are implemented using Java bytecode, and produce circuits described in VHDL. However, this could be extended to other languages, and different HDLs such as Verilog. Flowpaths are further described in [1, 2]. Table I shows a summary of benchmark results comparing flowpaths to a jStamp microprocessor that natively executes Java bytecode.

Experiment	*Method*	*Time*		*Energy*	
		ms	*ratio*	*mW·ms*	*ratio*
Mandelbrot	JStamp	2.7	1	319	1
	Flowpaths	0.065	0.024	15.55	0.048
FFT	JStamp	237.6	1	44669	1
	Flowpaths	3.84	0.016	714	0.015
Linpack	JStamp	2800.0	1	526,400	1
	Flowpaths	122.2	0.043	34,065	0.064

Table I. Flowpaths versus JStamp performance

These results show improvements in nearly all areas when compared to another embedded system. More results can be found in [1].

These tools allow users to design in a high-level language. Handel-C, for example, uses a subset of C with hardware-specific extensions, while flowpaths uses stack-based languages. Our implementation in Java makes no modification to the standard Java language. On the other hand, Handel-C requires the knowledge of a modified language, allows for design in a high-level language resulting in hardware descriptions that are not practical to modify at the HDL level, and often generates less efficient hardware as described in [2, 3].

Flexible integration of a system is possible in a number of ways. Aside from doing it by hand, tools offer simpler solutions to this problem. Some tools like Handel-C require the pre-compilation of a module for use in a greater system. This can often be confusing and requires many steps. Flowpaths, as we will show, allows for easy integration of custom hardware components into a system through the use of object-oriented design within Java.

This paper describes the flexibility of the flowpath compiler for use in a hardware system. Section 2 outlines how custom VHDL modules can be integrated into an existing flowpath design. Section 3 describes a Mandelbrot fractal explorer using flowpaths to integrate legacy and custom VHDL modules including drivers for VGA, a PS/2 mouse, and user I/O. Section 4 elaborates further with an example of a full robotics system integrated together using flowpaths with custom hardware for GPS, Light Detection and Ranging (LIDAR), a motor controller, a camera with an image processing pipeline,

Manuscript received March 31, 2011.

Darrin M. Hanna, Bryant Jones, Lincoln Lorenz, and Mark Bowers are with the School of Engineering and Computer Science at Oakland University in Rochester, MI (dmhanna@oakland.edu).

and an Inertial Measurement Unit (IMU). The paper closes by providing concluding remarks and future work.

II. Object-Oriented System Design

Flowpaths have the flexibility to utilize custom hand-crafted VHDL components for use in generating hardware. This gives a computer engineer several options for design of a hardware system. Examples of where this can be used include: replacing a complex portion of a generated flowpath with an optimized custom component for greater efficiency, or the integration of several modules with a flowpath as the interconnection fabric. This is described by Fig 1.

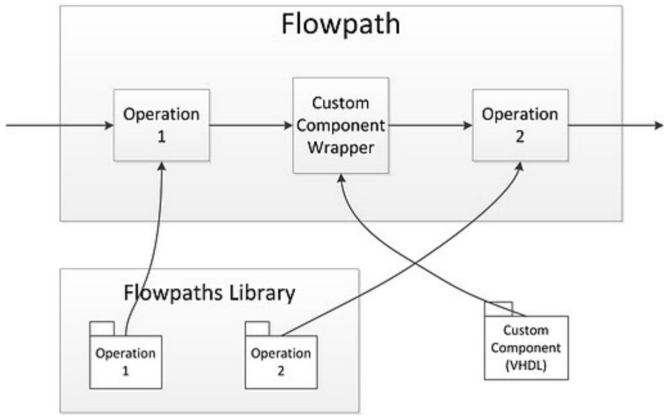


Fig 1. Overview of flowpath component integration

Legacy, custom, or IP core components, can be incorporated easily into an algorithm for an embedded system using flowpaths. Each component can be described using object-oriented methodologies. Components are characterized as objects of the system, with its interactions as functions of that object. This can be easily described in Java by creating a class for each component. Within this class, a method must be made for each function of the component. Alternatively, since a component's interface could be described by a single method, one class could be created with a method for each component. Since the generated flowpath for a method will contain a custom component, the Java method can be left empty. The only parameters that need to be established for a method are the inputs and outputs. These will correspond to stack inputs and outputs of a datapath. The generated flowpath will handle parameter passing into and out of the component. The only change needed to adapt the corresponding hardware component for use in the flowpath, is to make the component conform to a standard operation interface. This interface consists of the changing execution stack (StackIn and StackOut buses) and propagation signals (enable-in and done-out).

Events are usually handled in an embedded system with one of two methods; polling or interrupts. Flowpaths does not currently support the use of interrupts, and therefore handles events using polling. A main execution loop can be created, as in most embedded systems, to allow the generated flowpath logic to interact with the components. When an event occurs within a component it must be registered until the next time it is polled. Once that component is polled again, the event will be handled and reset. Examples given in this paper use this polling method.

III. An Embedded System for Exploring the Mandelbrot Set

A complete Mandelbrot explorer system was implemented using the flowpaths compiler. The Mandelbrot set is a fractal image produced by iterating a quadratic polynomial across points in the complex plane [5]. The system starts at an initial image of the Mandelbrot set, and allows the user to zoom further into a desired portion of the set, recalculating the image and yielding more and more detail. This zooming behavior is shown in Fig 2.

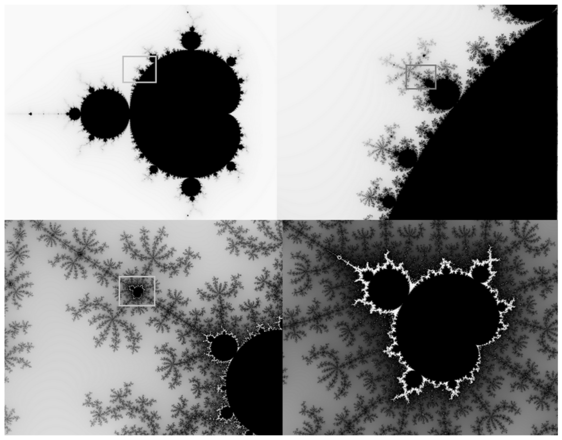


Fig 2. Example Mandelbrot fractal zoom sequence

A VGA driver is used for displaying the Mandelbrot fractal. A PS/2 mouse driver is used for zooming into and out of the fractal. Various calculation parameters are set via switches and buttons using a handwritten component. The VGA and mouse drivers are legacy components. The Mandelbrot calculation core consists of two parallel flowpaths which were generated from a Java algorithm. Each of these components is integrated easily using the object-oriented method presented in this paper.

A class was created containing methods that describe the interface to the existing components. A single method was created for each component. Methods called *PollMouse*, *PollSwitches*, *MandelbrotCalc*, and *PlotPixel* were created. These methods describe the inputs and outputs of the components. The *PlotPixel* and *PollMouse* methods are shown as examples in Listing 1.

```
static void PlotPixel(int addr, int color){
    // write 16-bit VRAM word -> two 8-bit pixels side
    // color(16 downto 8) -> pixel1
    // color(7 downto 0)  -> pixel2
}

static int PollMouse(){
    // returns:
    // 11 downto 0   <= X position
    // 23 downto 12  <= Y position
    // 24 <= r click detected
    // 25 <= l click detected
    return 0;
}
```

Listing 1. *PlotPixel* and *PollMouse* interface methods

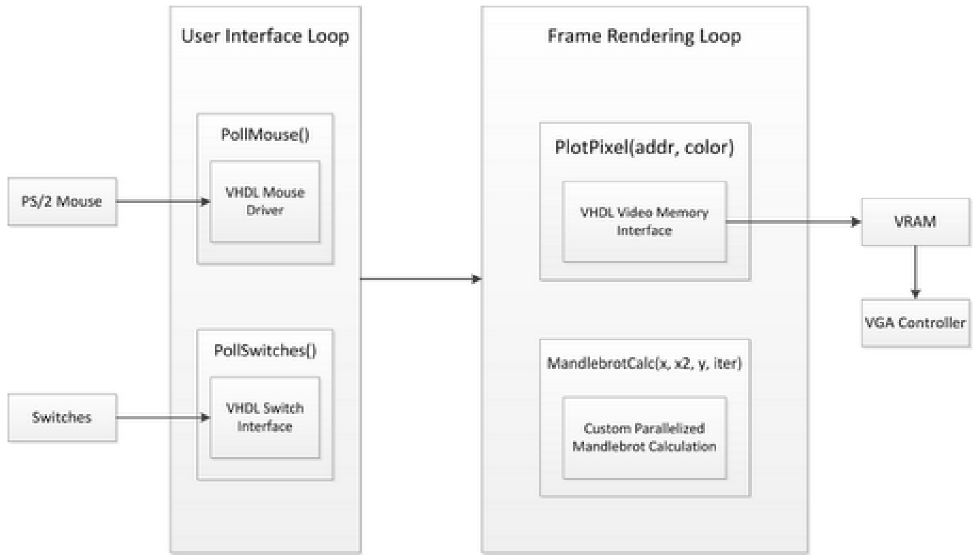


Fig 3. Overview of the Mandelbrot explorer flowpath

The functional interface of these components is shown in Fig. 3. The user interface loop continuously polls the status of the mouse and switches using the *PollMouse* and *PollSwitches* methods, respectively. Momentary events (button presses, mouse clicks) are captured within these methods and handled when the device is polled again.

Mouse clicks trigger the frame rendering loop. This loop recalculates the image based on user input. The calculation is performed in the *MandelbrotCalc* function. This function represents a component consisting of two instances of a previously generated Mandelbrot flowpath, combined in a manner such that they operate upon two Mandelbrot points in parallel. Each instance performs an iterative calculation upon a given point in the Mandelbrot set. The number of iterations taken corresponds to the appropriate pixel color, which is then written to video memory with the *PlotPixel* function. This function writes directly to VRAM, by simply wiring the appropriate stack elements to the memory's address and data lines, and the operation enable pulse to the memory's write enable.

Implementing the frame rendering loop in VHDL would require a complex state machine and take a significant amount of time to implement. The Java version, on the other hand, merely consists of two nested *for* loops and two function-calls, which can be created very quickly. A comparison was performed on several implementations of a Mandelbrot system. The results are shown in Table II.

Calc Core	Glue Logic	Logic Elements Consumed	Draw Time (ms)	Draw Time (ratio)	Approx Dev. Time
Flowpath 32-bit	Flowpath	15,772 (47%)	970	1.000	30 min
Flowpath 32-bit	Manual	8,379 (25%)	860	0.886	3 hours
Manual 32-bit	Manual	3,420 (10%)	414	0.426	1 Day

Table II. Relative performance of Mandelbrot implementations

The Mandelbrot explorer system was developed using several different methods. The systems were subjected to a standard benchmark using the same calculation parameters to produce an image of the Mandelbrot set. The flowpath-generated glue logic had a slight impact on the draw time, but was developed much quicker. The flowpath-generated calculation core yielded performance half as fast, and utilized twice as much logic as a handwritten core, but again, was created in significantly less time.

IV. Robotic System Integration

An FPGA-based system for an autonomous ground robot is currently being developed using flowpaths to implement intelligent algorithms that depend on several components. In robotic systems, there are generally multiple sensors, sensor processing, intelligent processing algorithms, and signal conditioning components, among others. Using a high-level object-oriented programming language to create a robotics control algorithm, streamlines the design of a robotic system. The ability to generate special-purpose hardware from a high-level Java description makes it practical to implement such a system on an FPGA. The easy description allows for future changes to the system to be made quite easily.

The system is based on the simple sense-think-act loop model, where a robot acquires the latest sensor data, makes a decision based on its current state estimate, and sends control signals to act upon its decision. Sensors and other modules are connected to the FPGA through external I/O pins. Each custom component polls its sensor and reads data at the interface update rate. Handwritten VHDL components have been developed for interfacing to a GPS sensor, a LIDAR sensor, and motor controller. Other components such as a camera interface, an image processing pipeline, and an IMU interface are in development. Fig 5 illustrates the interfaces and glue logic that will be used to integrate the system.

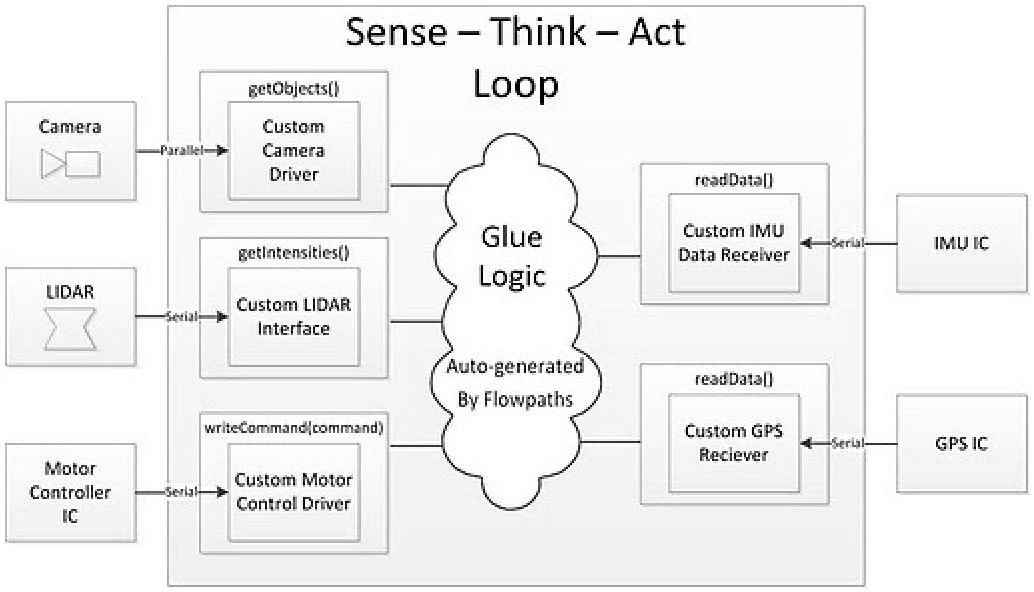


Fig 5. Object-oriented robot design

Individual classes were created to describe the functions of the handwritten components. Each component class contains the methods used to interface with that component and additional helper methods.

The motor controller component utilizes a built-in closed loop control algorithm to control the motors' speeds using encoders for feedback. Its interface requires a system speed and heading angle to compute the velocity for each motor. To interface to the component, a Java class named *MotorController* was created that contains a method called *sendCommand* which takes speed and heading angle as parameters.

A GPS sensor is used for detecting the robot's position on the earth. The GPS component receives the latitude, longitude, altitude, and status information from the GPS sensor. The custom hardware handles all of the details of the interface to the sensor including the parsing of the serial message**.** A Java function, *readData* was created as an interface to the component that reads the latest GPS sensor data. Normally, the *readData* function would return the full GPS message in one read, however, since the Java language limits the number of return values of a function to one, a workaround is needed. It accomplishes this by using an index to select which portion of the data to read. In this way, the data can be read out serially using a call to *readData* for each index. This presents another obstacle since *readData* is called in four places. This normally will generate four instances of the *readData* wrapper including the handwritten hardware. This is redundant and makes connection to an external interface unfeasible. To avoid this, one wrapper file is created and a multiplexer is inserted to control access to the custom hardware from the four different locations. To provide a cleaner external interface, the GPS data is compiled into a *GPSMessage* object. Using the method *readMessage*, the *GPSMessage* passed to it will be populated with the current data. The Java class which provides the interface to the GPS sensor is shown in Listing 2.

```
/**
 * Controller interface to the GPS Sensor
 */
public class GPSDriver{

    public void readMessage(GPSMessage message){
        //populate message
        message.status = readStatus();
        message.longitude = readLongitude();
        message.latitude = readLatitude();
        message.altitude = readAltitude();
    }

    private int readStatus(){
        return readData(0);
    }

    private float readLongitude(){
        return readData(1);
    }

    private float readLatitude(){
        return readData(2);
    }

    private float readAltitude(){
        return readData(3);
    }

    private int readData(int index){
        //get various data from buffer
        // in handwritten component
        return 0;
    }
}
```

Listing 2. GPS interface

To properly navigate the robot, environmental surroundings need to be monitored constantly. A LIDAR and camera are used for this task. The LIDAR operates by receiving laser ranging data in a series of angle-referenced intensity bins which are used to determine surrounding objects. These are read through the Java wrapper method *getIntensities* and added to a LIDAR message created in Java, much like the *GPSMessage*. The camera, on the other hand, acts as the robot's eyes using an algorithm to detect objects in the robot's view. Its corresponding hardware component calculates a list of objects described by their three dimensional location and size. The interface to the camera component reads this list of objects and combines them into a message containing an array of objects.

Furthermore, the robot must be conscious of its orientation relative to the environment. An IMU is used for calculating the robot's relative position and system's current motion. This component receives orientation, velocity, and acceleration data from the IMU. These data are returned from the component using an interface wrapper, as with the other sensors.

The full system can be pulled together using a sense-think-act loop written in Java. Flowpath algorithms can be written on top of the glue logic used to connect the components. The model can be implemented using a main execution loop with the divisions (sense, think, and act) called in order. Sensing can be done by polling the individual sensor interfaces. On each iteration of the loop, the sensors will be queried for data. The robot will have detailed information about its environment and its position relative to it. Using this sense data, the robot will be able to effectively decide how to react. This can be calculated based on its estimated current state and its goal. Currently, response algorithms are unimplemented in this design, however, the creation should be relatively straightforward using Java. Once the system has chosen its path, the robot can use the *sendCommand* interface of the motor controller and the loop repeats.

V. CONCLUSION

This paper shows how the flowpaths compiler can be used for flexible design of a hardware system which uses legacy, custom, and IP core components. These can be integrated easily into the flowpaths structure through the use of custom classes in Java. By creating a class for custom components, an inter-connection layer can be created in flowpaths to wire the modules together. Using this technique allows for a time savings in development with respect to integration of components.

Two examples were given to demonstrate how using object-oriented design and flowpaths reduced the time and complexity in implementing embedded systems using legacy, IP core, and custom components. A Mandelbrot fractal example demonstrated the integration of a few components with minimal complexity. Also, a real-life robotics system that is underway was described including several components for interfacing outside peripherals such as a camera, LIDAR, a motor controller, an IMU, and a GPS module. Through these examples, we have shown how flowpaths can be used to quickly and easily generate efficient inter-connection logic for a system using object-oriented principles.

VI. FUTURE WORK

Currently in Java, returning an object uses memory. To alleviate this issue, a special reserved class could be implemented to handle this without the use of memory. Work also includes the implementation of automated mapping of external signals into a generated flowpath. This includes the ability to utilize pin resources on a specified FPGA to include the proper connection from within a Java algorithm; similar to the classes implemented by a jStamp embedded processor [6]. Creation of timing libraries, optimization techniques, and process threading are also being explored as possible additions to the flowpath compiler for use with flexible design control.

REFERENCES

1. D. M. Hanna, B. Jones, L. Lorenz, and M. Bowers, "Generating Hardware from Java Using Self-Propagating Flowpaths," Submitted to the International Conference on Embedded Systems and Applications, 2011.
2. D. M. Hanna and R. E. Haskell, "Flowpaths: Compiling Stack-Based IR to Hardware," Microprocessors and Microsystems, vol. 30, pp. 125 - 136, 2006.
3. S. A. Edwards, "The Challenges of Hardware Synthesis from C-like Languages," Proc. of Design Automation and Test in Europe (DATE), Munich, Germany, 2005.
4. D. M. Hanna, M. Duchene, L. Kennedy, and B. Carpenter, "A Compiler to Generate Hardware from Java Byte Codes for High Performance, Low Energy Embedded Systems," The 2007 International Conference on Engineering of Reconfigurable Systems and Algorithms, Las Vegas, NV, June 25 - 28, 2007.
5. Benoît Mandelbrot, *Fractal aspects of the iteration of* $z \rightarrow \lambda z(1-z)$ *for complex* λ, z, Annals NY Acad. Sci. **357**, 249/259
6. Systronix, "JStamp: Real-time Native Java Module," 2003.

An Energy-oriented Retargetable Simulator for Instruction-Set Architecture

Kyungyoung So[1], Kwangman Ko[2]
[1]Divisional of IT Information Engineering, Chonbuk National University, KOREA
[2]School of Computer and Information Engineering, SangJi University, KOREA

Abstract - *Retargetability is typically achieved by providing target machine information, ADL, as input. The ADL are used to specify processor and memory architectures and generate software toolkit including compiler, simulator, assembler, profiler, and debugger. Simulators are critical components of the exploration and software design toolkit for the system designer. Instruction-Set Architecture(ISA) simulator an integral part of today's processor and software. In this paper, we design and implement the energy-oriented simulation environments to reduce the time and cost of simulator development through the retargetable technique for ISA. To accomplish our research objectives and goals, Firstly, we describes the energy consumption estimation and monitoring information on the ADL based on EXPRESSION. Secondly, we generate the energy estimation and monitoring simulation library and then construct the simulator. Lastly, we represent the energy estimations results for MIPS R4000 ADL description. From this subject, we contribute to the efficient architecture developments and prompt SDK generation through programmable experiments in the field of mobile software development.*

Keywords: Retargetability, ADL, ISA Simulator, Low power

1. Introduction

Rapid exploration and evaluation of candidate architectures are necessary due to time-to-market pressure and short product lifetime. Without automation and a unified development environment, the design process is prone to error and may lead to inconsistencies between hardware and software representations. The solution is to use a golden specification to capture the architecture and generate the required executable models to enable design automation of embedded processors. The phrase "Architecture Description Language"(ADL) has been used in the context of designing both software and hardware architectures[1]. Retargetability is typically achieved by providing target machine information, ADL, as input. The ADL are used to specify processor and memory architectures and generate software toolkit including compiler, simulator, assembler, profiler, and debugger[2].Simulators are critical components of the exploration and software design toolkit for the system designer. They can be used to perform diverse tasks such as verifying the functionality and/or timing behavior of the system, and generating quantitative measurements(e.g., energy consumption) which can be used to aid the design process[3,4]

Instruction-Set Architecture(ISA) simulator an integral part of today's processor and software. While increasing complexity of the architectures demands high performance simulation, the increasing variety of available architectures makes retargetability as a critical feature of an ISA simulator. Specially, Together with the energy minimization, the estimation and tracing of energy consumption and the construction of energy-oriented simulation environment have been a continuous research topics[5]. Reducing energy consumption has become an important issue in designing hardware and software systems in recent years. Energy-oriented high-level and low-level compiler optimizations have been shown to be particularly useful in reducing the system energy in prior works[6]. Therefore, It is very important research area for embedded software developers to measure and trace of energy consumption on the simulator.

In this paper, we design and implement the energy-oriented simulation environments to reduce the time and cost of simulator development through the retargetable technique and to evaluate and trace the energy consumption of it's applications. To accomplish our research objectives and goals, Firstly, we design of Energy-oriented ADL(EoADL) based on an EXPRESSION[3], LISA[4] and fast simulation model and strategies. And then, we implements of algorithm for the energy evaluation. Finally, we demonstrated the effectiveness and efficiency of the proposed retargetable simulator for ISA

2. Related Works

A fast and retargetable simulation technique is presented in [7]. It improves traditional static compiled simulation by aggressive utilization of the host machine resources. Such utilization is achieved by defining a low level code generation interface specialized for ISA simulation, rather than the traditional approaches that use C as a code generation interface. Retargetable fast simulators based on an ADL have been proposed within the framework of FACILE[8], MIMOLA[9], LISA[10], and EXPRESSION[11]. Specially, Language for Instruction Set Architecture(LISA), an uniform ADL, supports automatic generation of software toolsuite(C Compiler, Assembler, Linker, Simulator, Profiler) and optimized RTL description within a short time, accelerating the design space exploration. EXPRESSION is an ADL for modeling, software toolkit generation, rapid prototyping, design space exploration, and functional verification of SoC. The EXPRESSION follows a mixed level approach-it can capture both the structure and behavior supporting a natural specification of the programmable architectures consisting of processor cores, coprocessors, and memories. And it was originally designed to capture processor/memory architectures and generate software toolkit to enable compiler-in-the-loop exploration of SoC architecture.

In the last two decades, there have been many studies done on instruction set simulation(ISS) to improve their performance. Many ISSs are based on interpretive simulation techniques, where the simulator is looping in the fetch-decode-execute cycle. For such simulator, the decoding overhead is a major bottleneck to performance. To remove the decoding overhead of interpretive simulators, Mills[13] proposed the static compiled simulation technique, where the input application binary is translated into C code that is then compiled with the simulator source code. Even though there are many proposals of static compiler simulation[14] to improve simulation performance, it has two major drawbacks. One is that the compilation time may take longer than the actual simulation time for large applications. The other is that it cannot handle self-modifying code. To overcome these drawbacks, people proposed dynamic compiled simulation[15], which is based on just-in-time compilation techniques that exploit code caches. However, the compiled simulation techniques either translate the target instructions into the native instruction set of the host system, or assume a virtual machine whose ISA is different from the target ISA. But, it cannot exactly model the target architecture, such as pipeline interlocks, resulting in loosing accuracy of application execution time. Most of the simulators that claim cycle accuracy, such as SimpleScalar[16], ARMulator[17], and SESC[18], are based on interpretive simulation.

Low energy consumption is an important metrics that affects choice of hardware and software components in building small and large-scale systems. When designing high-performance, low-power processors, designers need to experiment with software and architectural level trade-offs and evaluate various power optimization techniques. Architectural level power estimation tools are becoming increasingly important with the growing complexity of current designs to provide fast estimates of the energy consumption early in the design cycle[19]. The Simplepower[20] energy simulator was developed based on transition-sensitive energy models and is an execution-driven, cycle-accurate RT level tool. Wattch[21], an architectural simulator that estimates CPU power consumption, are based on a suite of parameterizable power models usage counts generated through cycle-level simulation. Wattch's power modeling infrastructure as a useful and significant enabler of further research on architecture and compiler approaches for power efficiency.

3. Energy-oriented Retargetable Simulator

3.1 Energy-oriented Simulator Model

In this paper, we develops the energy-oriented simulation environments to reduce the time and cost of simulator development through the retargetable technique and to evaluate and trace the energy consumption of it's applications. To accomplish our research objectives and goals, we suggest an implementation model as shown in Figure 1.

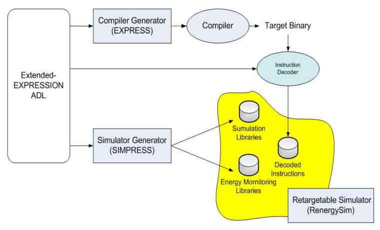

Extended-EXPRESSION ADL
Compiler Generator (EXPRESS)
Compiler
Target Binary
Instruction Decoder
Simulator Generator (SIMPRESS)
Sumulation Libraries
Decoded Instructions
Energy Momitoring Libraries
Retargetable Simulator (RenergySim)

Figure 1. Generation of Energy-oriented ISA Simulator through the EoADL

Firstly, we designed the EoADL, extended EXPRESSION ADL, with the energy consumption monitoring information. Secondly, we generated the architecture structure information libraries, the simulator core engine's libraries, and energy consumption monitoring libraries through the extended EXPRESSION simulator, SIMPRESS[22]. Finally, we generated the energy-oriented retargetable simulator, RenergySim, through the compilation and linking with generated simulation libraries and energy monitoring libraries. Specially, we constructed the energy consumption monitoring libraries from the EoADL descriptions as following Figure 2.

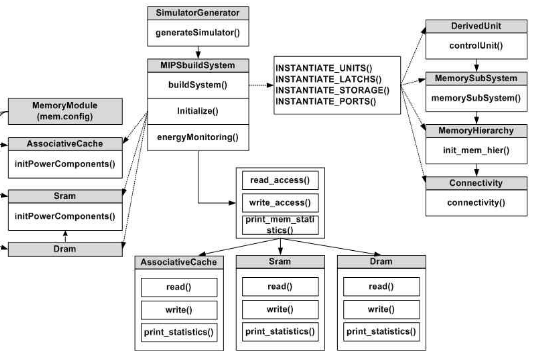


Figure 2. Core Classes of Energy Consumption Monitoring Libraries's

3.2 EoADL Design and description

EoADL syntax are based on the LISA, C++ like forms, and composed of four core classes as following Figure 3, Super_ADL class, Structure class, behavior class, and Energy class. Specially, Energy class captures the energy-efficient target code generation methods, the energy-oriented code optimization methods, and the energy management libraries(evaluation and tracing) that are independent of platforms.

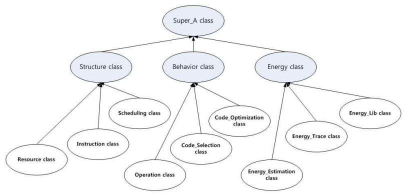


Figure 3. EoADL Classes and Hierarchical Relationships

Practically, we described the EoADL, Figure 5, according to the memory hierarchical design as Figure 4.

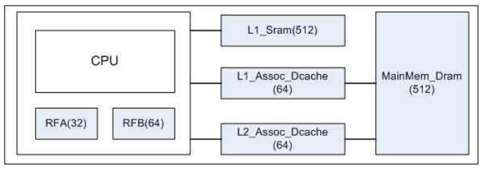


Figure 4. Example Design of Memory structure

```
(STORAGE_SECTION
(RFA
   (TYPE VirtualRegFile)
   (WIDTH 32) (SIZE 32) (MNEMONIC "R")
)
(RFB
   (TYPE VirtualRegFile)
   (WIDTH 64) (SIZE 32) (MNEMONIC "f")
)
(L1_Assoc_Dcache
   (TYPE DCACHE)
   (SIZE 64) (LINESIZE 2) (ASSOCIATIVITY 4)
   (ACCESS_TIMES 1)(ADDRESS_RANGE(0 9995904))
)
(L2_Assoc_Dcache
   (TYPE DCACHE)
   (SIZE 64) (LINESIZE 2) ASSOCIATIVITY 8)
   (NUM_LINES 64) (ACCESS_TIMES 5)
   (ADDRESS_RANGE (0 9995904))
)
(L1_Sram
   (TYPE SRAM) (ACCESS_TIMES 1)
   (ADDRESS_RANGE (9995905 9999999))
)
(MainMem_Dram
   (TYPE DRAM) (ACCESS_TIMES 50)
   (ADDRESS_RANGE (0 9995904))
)
```

Figure 5. EoADL Description according to Figure 4.

4. Experimental and Conclusions

We generated the RenergySim from the EoADL description, such as MIPS R4000 based on EXPRESSION ADL description, and the energy consumption libraries. For the verification of the RenergySim and the estimation of energy consumption, we used EXPRESSION's benchmarking applications(LL1.c-LL19.c) as following Figure 6. The Simulation results show that the RenergySim reduces energy consumptions by up to 5-10% compared to the original EXPRESSION simulator and we had important meanings that energy estimation simulator generated from the ADL description.

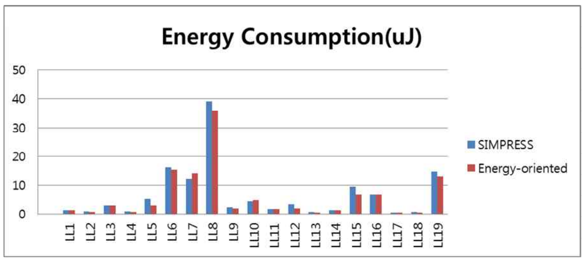


Figure 6. Comparison of Energy Consumption

(SIMPRESS vs. Energy-oriented Retargetable Simulator)

5. References

[1] Prabhat Mishra, Nikil Dutt, Processor Description Languages, Morgan Kaufmann, 2008.
[2] Hoffman, H. Meyr, R. Leupers, Architecture Exploration for Embedded Processors with LISA, Kluwer Academic Publishers(ISBN: 1-4020-7338-0), Dec. 2002.
[3] Anupam Chattopadhyay, Heinrich Meyr, and Rainer Leupers, "LISA: A Uniform for Embedded Processor Modeling, Implementation, and Software Toolsuite Generation", Processor Description Languages: Chap. 5, Morgan Kauffman, 2008.
[4] Prabhat Mishra, Aviral Shrivastava, and NiKill Dutt, "Architecture Description Language(ADL) - driven Software Toolkit Generation for Architectural Exploration of Programmable SOCs", ACM Transactions on Design Automation of Electronics Systems, Vol. 11. No. 2, 2006.
[5] M. Reshadi, Prabhat Mishra, Nikil Dutt, "Instruction Set Compiled Simulation: A Technique for Fast and Flexible Instruction Set Simulation", DAC'03: Proceedings of the 40th conference on Design Automations, pages 758～763, 2003.
[6] A. Parikh, Soontae Kim, M. Kandemir, N. Vijaykrishnan, M. J. Irwin, "Instruction Scheduling for Low Power", Journal of VLSI Signal Processing, Vol 37, pages 129～149, 2004.
[7] Jianwen Zhu, Daniel D. Gasski, "A Retargetable, Ultra-fast Instruction Set Simulator", DATE'99: Proceedings of the Design Automation and Test conference in Europe, pages 1999.
[8] Eric C. Schnarr, Mark D. Hill, James R. Larus, "Facile: A Language and Compiler for High-Performance Processor Simulators", PLDI'99: Proceedings of the ACM SIGPLAN 1999 conference on Programming Language Design and Implementation, pages 1～11, 1999.
[9] R. Leupers, J. Elste, B. Landwehr, "Generation of Interpretive and Compiled Instruction Set Simulators", ASP-DAC'99: Proceeding of the Asia South Pacific Design Automation Conference 1999, pages , 1999.
[10] S. Pees, A. Hoffman, V. Zivojnovic, H. Meyr, "LISA-Machine Description Language for Cycle-Accurate Models of Programmable DSP Architectures", DAC'99: Proceedings of the 36th Design Automation Conference, 1999.
[12] Peter Grun, Ashok Halambi, Vijay Ganesh, Nikil Dutt, Alex Nicolau, "EXPRESSION: An ADL for System Level Design Exploration", Technical Report TR98-29, University of California Irvine, 1998.
[13] Christopher Mills, Stanely C. Ahalt, and Jim Flower, "Compiled Instruction Set Simulation", Software Practice and Engineering, Vol. 21(8), pages 877～889, 1991.
[14] Stefan Kraemer, Lei Gao, Jan Weinstock, Rainer Leupers, Gerd Ascheid, and Heinrich Meyer, "Hysim: A Fast Simulation Framework for Embedded Software Development", CODE+ISSS'07, pages 75～80, 2007.
[15] M. Poncino, Jianwen Zhu, "Dynamosim: A Trace-based Dynamically Compiled Instruction Set Simulator", ICCAD'04: Proceedings of the 2004 IEEE/ACM International Conference on Computer-Aided Design, pages 131-136, 2004.
[16] SimpleScalar: http://www.simplescalar.com
[17] ARM Limited. RealView ARMulator ISS User Guide(ver1.4.3), 2007.
http://infocenter.arm.com
[18] SESC: SuperESCalar Simulator. http://iacoma.cs.uiuc.edu/~paulsack/sescdoc/, 2002.
[19] Uli Kremer, "Compilers for Power and Energy Management", PLDI'03: ACM SIGPLAN 2003 conference on Programming Language Design and Implementation Tutorial, 2003.
[20] W. Ye, N. Vijaykrishnan, M. Kandemir, M. J. Irwin, "The Design and Use of SimplePower: A Cycle Accurate Energy Estimation Tool", DAC'00: Proceedings of the 37th Design Automation Conference, pages 340～345, 2000.
[21] D. Brooks, V. Tiwari, M. Martonosi, "Wattch: A Framework for Architectural-Level Power Analysis and Optimizations", In Proc. of International Symposium on Computer Architecture, pages 83～94, 2000.
[22]Alex Nicolau, et al., "V-SAT: A Visual Specification and Analysis Tool for System- on-Chip Exploration", Journal of System Architecture, Vol. 47, pp.263～275, 2001.