SESSION

PERFORMANCE ISSUES AND ENHANCEMENT METHODS + POWER ANALYSIS

Chair(s)

TBA

Virtual Page Placement Guided by DRAM Locality and Latency

Daniel G. Bassett[1], Aabhas S. Agarwal[1] and Brian T. Davis[2]

[1]Department of Electrical and Computer Engineering
Michigan Technological University
{dbassett, asagarwal}@mtu.edu

[2]Department of Electrical and Computer Engineering
Embry-Riddle Aeronautical University
btdavis@btdavis.org

Abstract

The organization of modern DRAM containing multiple banks per device or module allows for non-uniform access latencies. By including the physical organization of the memory system in virtual page allocation algorithms, program execution time can be reduced through aggregate changes in memory access latency. Four virtual page allocation mechanisms are incorporated into the Linux 2.6 kernel and evaluated for impact upon benchmark execution times. These modified kernels are able to better utilize the available parallelism and multiple latency states of DRAM in the main memory system. Each of these mechanisms uses a different approach, or a combination of approaches, to distribute virtual pages across the available DRAM banks. An analysis utilizing several benchmark suites demonstrates that these modifications to page placement algorithms provide a moderate increase in performance in most benchmarks. These modified virtual page allocation mechanisms require limited changes to the code base, with no modifications to the hardware.

Keywords
Virtual Memory; DRAM; Linux Kernel

1. Introduction

We propose to change the page allocation and placement algorithm in the virtual memory component of the operating system to make use of the parallelism and locality available in SDRAM. This will incorporate emerging technologies in compilers, operating systems (OS) and memory hierarchy design.

While processors have increased in performance at a rate of 75 percent per year, DRAM has increased at a mere 7 percent per year. [14]. DRAM has therefore become one of the major limiting factors in computer performance, as evidenced by the steadily increasing cache sizes in modern processors to address the speed discrepancy. Although greater DRAM bandwidth can be achieved by increasing the bus width, or increasing the bus frequency, the latency inherent to the DRAM structure remains. The problem is exacerbated with the demand for greater capacities of DRAM because increased capacity results in both greater overall latency and an increased disparity in the non-uniform latencies.

Since the introduction of Synchronous DRAM, the main memory system has inherently possessed concurrency and parallelism that leads to a non-uniform access latency. The allocation and placement mechanism proposed here will incorporate policies which will be cognizant of the capabilities of SDRAM and make intelligent and adaptive placements based upon memory system capabilities, and run-time program behavior.

The goal of the work proposed is to reduce the latency of compulsory SDRAM accesses through adaptive data placement. Placement of objects within memory affects the latency to retrieve these objects for a number of reasons:

- Data requested from a synchronous DRAM device may reside in one of multiple states. The data access latency will be largely dependant upon the state and interaction between accesses.
- Each unique bank in the SDRAM may be maintained in an independent state. If an SDRAM access is issued in contention with other SDRAM accesses, the ability to parallelize the access stream is dependant upon accesses being distributed among unique banks.
- Virtual memory systems can be utilized to distribute data objects with arbitrary virtual locality, to either common DRAM banks, or unique SDRAM banks, as is appropriate for the application.

The study presented herein modifies the Linux 2.6.23.14 kernel virtual page placement, and presents the performance measured running applications on a notebook computer. In the following section we present relevant background material and in Section 3, discuss related work. Section 4 describes modifications made to the Linux kernel to facilitate exploration of four different virtual page placement algorithms, and Section 5 analyzes the performance impact of these algorithms. Section 6 states conclusions and Section 7 considers future work.

2. Background

Understanding of virtual memory function and SDRAM operation are key to understanding the motivations behind the virtual page placement mechanisms proposed and discussed.

2.1 Virtual Memory

Virtual memory provides each program with a process specific memory space. Memory is managed in pages, the size of which is OS dependant, under the constraints of the hardware memory management unit. The operating system (OS) manages the translation between virtual memory and physical memory addresses and the virtual memory system is transparent to user processes.

Virtual pages are identified using virtual page numbers. Physical memory is divided into page frames, with page frame numbers as identifiers. Every virtual page maps to a single page frame, either

This material is based upon work supported by the National Science Foundation under Award 0133777. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

in physical memory or swapped out to backing storage. Figure 1 shows an example of how this mapping may work. This mapping is managed using page tables, made up of page table entries (PTEs). A PTE must indicate if a virtual page is in memory, on disk or unallocated. PTEs should allow the OS to maintain information required for memory management. [8].

Page table organization is logically a single table with mappings to page frames. An exhaustive table with an entry for each possible virtual page would be prohibitively large. For efficient use of memory, hierarchical page tables are a commonly used alternative. Hierarchical page tables consist of multiple levels of tables. Each level's entries point to a table in the next lower level, except for the lowest level, which contains the mapping to physical memory.

2.2 SDRAM

SDRAM devices {SDR, DDR2, DDR3, XDR, etc...} contain multiple independent banks; each bank contains an array of memory cells organized into banks, rows and columns. The latency required for a DRAM access is dependant upon a number of factors, including but not limited to: (a) the state of the bank containing the data, (b) contention with accesses directed at unique banks of the DRAM, and (c) the sequence of accesses directed to a given bank.

The state of the SDRAM bank (a) determines whether a given access latency will include one, two or three of the phases required to complete an SDRAM access. These three phases are bank precharge, row activation, and column access; all SDRAM accesses require each three of these phases - but through state management and locality - bank precharge and activate can be done once for multiple column accesses. Bank precharge prepares the bank for row activation. Through row activation, the data contained in one row of the memory array is transferred into the bank's sense amplifiers. While a row is active in the sense amplifiers, one or more column accesses can be performed to retrieve or modify the data contained in the sense-amplifiers or I/O buffers. [9] The impact of these phases upon latency is dependant upon application access pattern and device state.

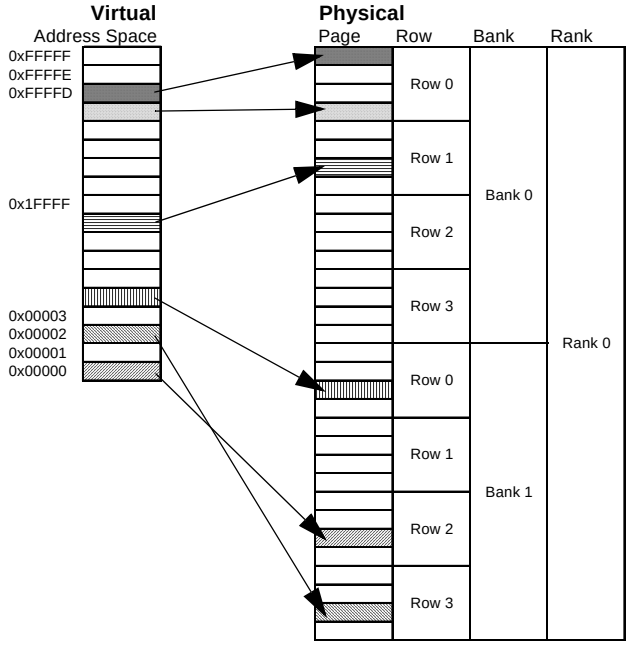


Figure 1. Virtual Address Mapping to Physical Memory

Each unique bank (b) within a DRAM system can be maintained in an independent state, and can thus be processing an independent request in parallel. The implication is that if two data structures are being frequently accessed in parallel, it is advantageous to have these data structures located in unique banks. This is in contrast with (c) accesses to a single bank, which must be serialized. The sequence of accesses to a single bank determines the overall number of precharges and activations through which an SDRAM bank must proceed. This sequence can directly impact the overall access latency.

2.3 Combining VM & SDRAM

In the past, virtual memory and DRAM controller policies and scheduling have been examined in isolation. Virtual memory has been implemented in a manner which is algorithmically efficient in terms of both time and memory usage. DRAM has been designed in a manner which is efficient in circuit topology, bit density, bandwidth and to a lesser degree, access latency. However, the objective of system design is to reduce overall application execution time. Through the use of VM algorithms which are cognizant of the latencies and parallelism available in the DRAM structure application execution time can be reduced, albeit by sacrificing algorithmic efficiency.

3. Related Work

Work examining topics related to reducing average memory access latency or DRAM cognizant virtual memory placement is described in this section. Many approaches have been proposed or implemented to address the DRAM access latency problem, or "Memory Wall" [14].

3.1 Memory Management

Virtual memory is known to impact system performance, and past work has been done to use VM to reduce access latency, but little of this work makes use of knowledge of the DRAM structure, and non-uniform latency constraints.

3.1.1 Data Placement Within a Page

Placement and alignment of data within the pages being dynamically allocated can reduce access time [5][10]. These techniques depend upon the application being programmed for a hardware implementation. These techniques require that the programmer be aware of memory details, and results in an application optimized for a subset of memory architectures. Unlike this approach, the page placement as described herein operates at the operating system level to place virtual pages across all applications resulting in reduced access latency enabled through the non-uniform access latency of DRAM.

3.1.2 Page Prefetch

Glass and Cao [6] examined how to fetch virtual pages from the disk swap area, prior to their request by the application. This will reduce memory access latency, primarily when the virtual memory requirements exceed the physical memory available.

3.2 Reducing Access Latency

The following sections discuss methods which attempt to reduce the latency of any access resulting from a load in the CPU. Not all accesses proceed to main memory, many are handled by on-chip memory which have significantly lower latencies.

3.2.1 Processor Cache

A processor cache consists of a high speed memory. Most processors use a hierarchical cache design, with the smallest and fastest memory being at the lowest level. A cache exploits spatial and temporal locality. Most programs use a limited set of memory locations during the majority of execution time [7].

3.2.2 Prefetching

Prefetching is the concept of retrieving data from memory before the data is required. Prefetching can be controlled by either software or hardware [18]. Software controlled prefetching is generally implemented by the compiler. Loading data into the cache before the data is required increases the effective cache hit rate. Prefetching also increases the number of accesses to main memory, as not all prefetched data is used.

3.3 Reducing DRAM Access Latency

The methods discussed in the following sections detail techniques that exploit a characteristic of DRAM operation to reduce the latency of DRAM accesses observed by the processor.

3.3.1 Open Page Controller Policy

SDRAM controllers have the option to use one of two static access control policies, either Open Page (OP) or Close Page Autoprecharge (CPA). OP leaves the accessed row open after each access, whereas CPA closes it by performing a bank precharge immediately after the access has been satisfied. The static controller policy that yields minimal benchmark execution time largely depends upon the SDRAM access pattern of an application.

By delaying a precharge operation (OP), the latency of a row hit access, is significantly reduced. The open page policy enables the sense amplifiers to act like a cache at the DRAM level [15]. For programs that have high spatial locality, row hits occur frequently and thus the open-page controller policy can reduce overall execution time. If, however, a program has very little spatial locality, using an open page policy will actually increase the latency of DRAM accesses, and the execution time will be longer. Some research [20] has demonstrated that allowing the SDRAM controller to dynamically transition between an open page policy and a closed page auto-precharge policy, based upon program access patterns, can reduce program execution time.

3.3.2 Address Remapping

Address remapping changes the mapping of process addresses to physical addresses, in order to distribute accesses across SDRAM banks within the physical address space. Address remapping in a basic form is present in modern SDRAM controllers such as the 915 chipset. These controllers use lower order address bits to select the bank and rank, distributing SDRAM row sized chunks of memory across SDRAM banks in a round-robin fashion. This simple approach to exploiting non-uniform access latency was the initial motivation for further exploitation of this property.

Bit-reversal address mapping is a more advanced remapping method that swaps the bits that select the row, bank, rank and channel as shown in Figure 2 [17]. Accesses become distributed across the DRAM banks more evenly because spatial locality causes the lower order bits change more frequently and bank conflicts are reduced. Similarly since the higher bits change less frequently, the number of row hits should increase.

The IMPULSE project [3] distributes memory in a manner that increases parallelism and cache utilization, but also attempts to provide a software interface to addresses higher level concerns, not only by distributing data structures across all banks, but by giving software the ability to manage this placement. This technique can provide significant performance gains, but requires the software to be specially coded and recompiled for the machine on which it is to be executed.

Address remapping is known to provide performance benefits when physical memory is being accessed directly. When a virtual memory system is used, because of the additional remapping from a virtual address to a physical address, the performance becomes non-deterministic. Since the operating system already manages the allocations of virtual pages into physical memory, a more suitable approach is to have the OS distribute virtual pages across the DRAM banks.

3.3.3 Balancing Bank Accesses

Rather than trying to distribute the usage of DRAM banks in terms of pages allocated, the more intuitively advantageous strategy is to balance the number of accesses to the banks [16]. If each bank is being access at the same rate, statistically each bank has the same probability of being accessed and therefore the amount of bank conflict is expected to be minimal, therefore maximizing bank parallelism utilization. Implementing this method would require hardware monitoring of the number of accesses to each bank because any software monitoring would require accesses to main memory. This work attempts to approximate balancing bank accesses without the additional hardware requirements.

3.4 Virtual Page Placement

The problem of virtual page placement is the determination of the physical page frame to which a newly allocated virtual page will be placed [4]. Our algorithms significantly change this placement problem by trying to place virtual pages in a manner which reduces latency through the physical capabilities of the SDRAM. Previous virtual page placement algorithms tend to focus on the algorithmic and data structure efficiency of the allocation and replacement tasks [11], rather than attempting to improve run-time performance.

3.4.1 Virtual Page Replacement

The task of virtual page replacement is the determination of which virtual page to eject when a new page is being allocated. This task is orthogonal to our algorithms, because the selection of a page to be ejected can utilize the same criterion as previous work [2]. If the physical memory is fully occupied, where replacement is critical, there will be reduced flexibility for placement, decreasing the performance impact of adaptive data placement.

4. Virtual Page Placement - Linux Implementation

The Linux 2.6 kernel uses a binary buddy allocator to allocate virtual pages for user processes. In general, a buddy system uses an array of free lists. There is a free list for each allowable block size or order of virtual memory pages. In the Linux kernel on the x86 architecture, there are block sizes from 4kB to 4MB in increments

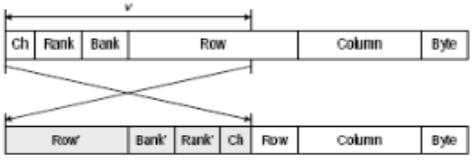


Figure 2. Bit-reversal Address Mapping [17]

based on powers of two. [4] The Linux kernel references blocks of memory as page structures and each block size has a corresponding order.

One of the major advantages of the buddy system is that splitting and coalescing operations are very fast. However, since the allowable block sizes are limited, internal fragmentation can be relatively high. The buddy system uses powers of two block sizes which makes finding the buddy of a block algorithmically simple [19].

The intent of this work is to create an allocation mechanism that utilizes the available bank parallelism to reduce the execution time required for memory accesses [1]. The Linux kernel's current page allocation mechanism is algorithmically efficient in terms of computation and data structures. Exploitation of the available bank parallelism and non-uniform latency present in modern SDRAM, allows average memory access times to be reduced, which in turn can reduce execution time. The changes required may impact the algorithmic efficiency of the allocation, but the expectation is that page accesses occur more frequently than page allocations and the reduction in memory access times will exceed the additional time required for allocation.

4.1 Kernel Changes

Only one file (page_alloc.c) in the linux kernel required changes for the implementations of the algorithms described. A useful tool in understanding the page allocation code is the Linux Cross Reference (LXR) [12]. LXR creates cross references for files names, structures, variables, functions and macro definitions.

Physical memory is allocated upon request from a process by rounding up the requested block size to the next architecturally allowable block size. The kernel function *(static struct page *__rmqueue)* performs the task of retrieving a block of the appropriate size (potentially multiple pages) from the appropriate free page list. This may be directly from the list containing the block size requested, or by splitting a larger block. When a block is freed, if the block's buddy is also free, the two blocks are coalesced together into one larger block.

Another function *(static inline void __free_one_page) exists* for freeing a single page. If the buddy of the page being freed is also free and of the same order, the two pages are combined and moved up to the next order. This continues for the buddy of the resulting combined page until the next buddy is not free. The page that is freed is added to a list of free pages, specific to the resulting order.

The code modifications discussed are dependant upon the specifics of this development environment. If the linux kernel modifications are going to be extended to the general case, they must be modified to work with all memory configurations, memory controllers, DIMM types and address mappings, as described in Section 7, Future Work.

4.2 Balance Bank Allocations

The first virtual page allocation algorithm attempts to balance allocations between all of the DRAM banks, with the motivation of reducing the number of bank conflicts and thus increasing bank parallelism.

The bank balancing algorithm maintains as equal as possible the number of pages in each free list, for orders less than the DRAM row size, which in this environment is 16kB. Higher order blocks or free lists will already encompass multiple banks and are not considered for bank balancing.

When a page is allocated, the bank with the most free pages is identified and a page from that bank is allocated. If no free pages are found in the bank with the most free pages, the default kernel behavior of using the first page in the free list is used. For page allocation requests that are of order greater than the DRAM row size or are in the DMA zone, the default kernel behavior is used.

4.3 Balance Bank Allocations Per Order

The second virtual page allocation algorithm is similar to the bank balancing algorithm, but balances bank allocations for each order of pages separately. While no global information about bank allocations is maintained, if the banks are perfectly balanced at each order, the combination of orders should also be balanced. The advantages of this method include some algorithmic simplifications in the implementation, and a reasonable assurance that an appropriate page can be found. The most significant disadvantage is that a larger array is required to maintain a free page count at each order.

When a page is allocated, the bank with the most free pages in the requested order is used. The advantage of this method is that if any free pages are available at the requested order, a bank match will be guaranteed. If there are no free pages in this order, the order is greater than the DRAM row size, or the request is in the DMA zone, then the default kernel behavior is used.

4.4 Free List Queue

The free list queue allocation algorithm was implemented in the function that frees one page to make the lists of free pages behave like a queue instead of a stack. The motivation for this change is that the page used will be the least recently freed page, whereas by default the most recently freed page would be used. This virtual page allocation algorithm is algorithmically simple and requires no information about the DRAM organization.

4.5 Balance Bank Allocations with Free List Queue

The last allocation algorithm is actually a combination of two of the previously discussed modifications. A queue is used for the free list and the banks are globally balanced based on the number of free pages in each bank. The performance of this benchmark is anticipated to be similar to balancing bank allocations, with some variation depending on the queue performance.

5. Performance Evaluation

The evaluation environment was a notebook computer with an Intel 915PM northbridge chipset, with two 512MB DIMMs of DRAM. Each DIMM has 8 banks divided into two ranks, and due to dual channel operation, there are 8 banks in total. The Intel 915PM chipset datasheet specifies that bits 14 and 15 are the bank select bits and the 29th bit is the rank select bit. The Linux kernel used was version 2.6.23.14 [12].

A variety of benchmarks were used to evaluate the change in performance. Many of the SPEC2000 benchmarks were executed to evaluate the change in performance for integer (CINT) and floating point (CFP) applications. STREAM was used to evaluate the impact of these algorithms upon applications which are heavily dependent upon memory bandwidth. MDBNCH is used to examine performance for applications which produce a large number of irregular memory accesses.

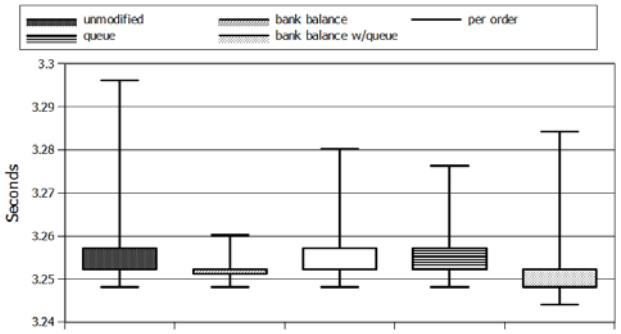


Figure 3. MDBNCH Results

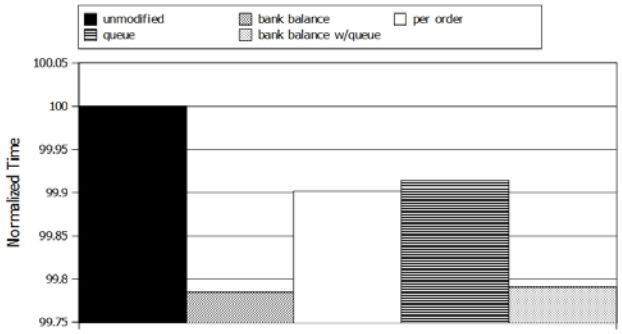


Figure 4. MDBNCH Average Results

5.1 MDBNCH

The MDBNCH was run twenty times on the stock kernel and each of the four modified kernels. Figure 3 shows a box plot of these results. Bank balancing shows the most consistent performance, with a small range of execution times. The distribution among banks through bank balancing provides the opportunity for reduced bank conflicts from the irregular memory accesses performed by MDBNCH, and can explain the small deviation in execution times. The results were averaged and normalized in Figure 4, more clearly showing the change in performance. All of the kernels with modified virtual allocation perform better than the stock kernel, although modestly; bank balancing results in approximately a 0.2% decrease in execution time.

5.2 STREAM

The STREAM benchmark suite [13] is composed of 4 benchmarks {Copy, Scale, Add and Triad} each of which are presented independently. This benchmark suite is intended for testing memory bandwidth. For the purposes of testing the kernel modifications, the STREAM benchmark was configured to use an array of 40,000,000 elements, with a total memory requirement of 915.5 MB. The large memory requirement gives a significant execution time, but is small enough to fit in main memory such that swapping to/from the disk does not become a performance limiting factor. The benchmark was also configured to iterate twenty times and returns the overall minimum, maximum and average execution times.

Figure 5, shows the average execution times for the Copy operation with the modified kernels to be slightly shorter than the unmodified kernel. For all modifications except bank balancing with a free list queue, the maximum execution time is noticeably smaller.

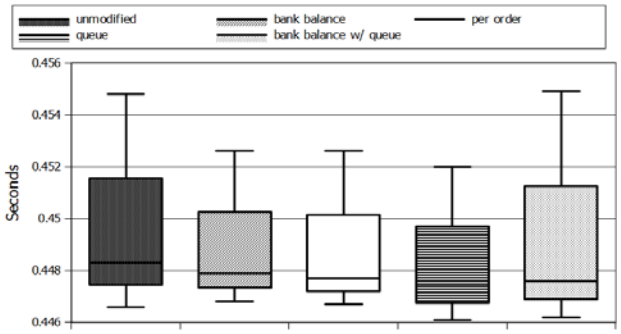


Figure 5. Stream Copy Results

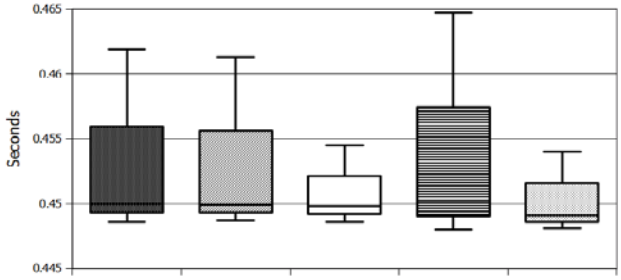


Figure 6. Stream Scale Results

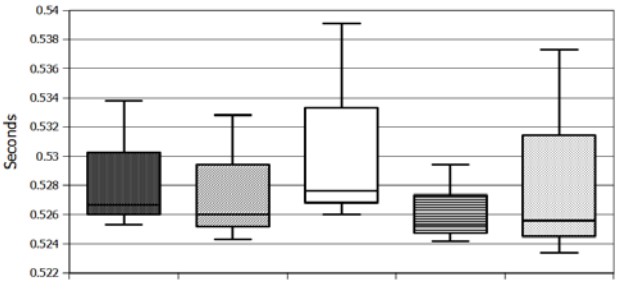


Figure 7. Stream Add Results

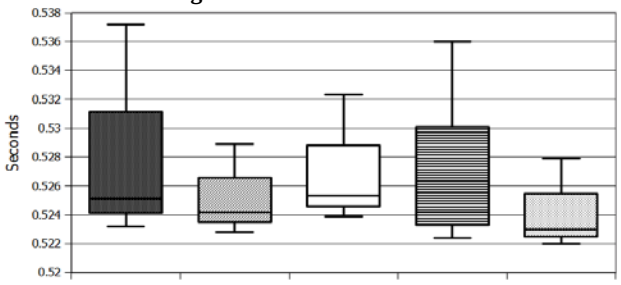


Figure 8. Stream Triad Results

The Scale operation results, shown in Figure 6, demonstrate nearly identical average execution times. Notably, bank balancing per order and bank balancing with a free list queue both provide significantly shorter maximum execution times.

Figure 7 shows that bank balancing per order has the worst performance for the Add operation. Using a free list queue consistently performs better than the other page allocation algorithms, with the smallest difference between maximum and minimum execution times.

The Triad operation results are illustrated in Figure 8. Global bank balancing with and without a free list queue and all execution times are lower than the unmodified kernel. As with the Add

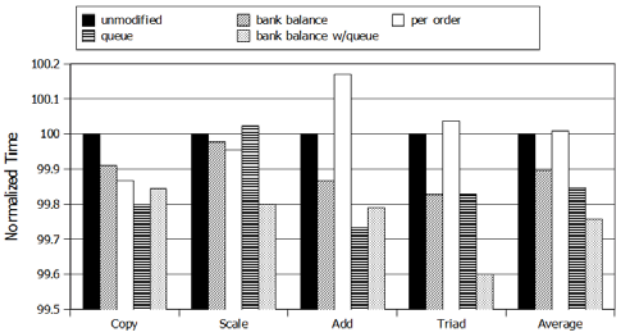


Figure 9. Stream Average Results

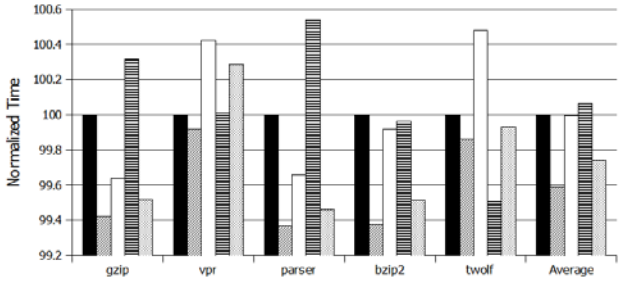


Figure 10. CINT Results

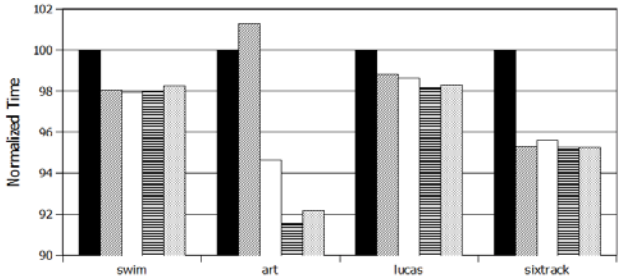


Figure 11. CFP Individual Results Part 1 of 3

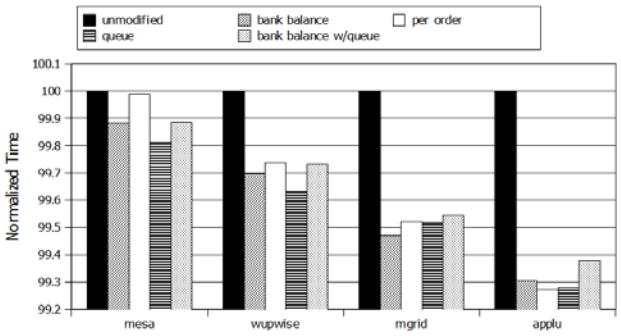


Figure 12. CFP Individual Results Part 2 of 3

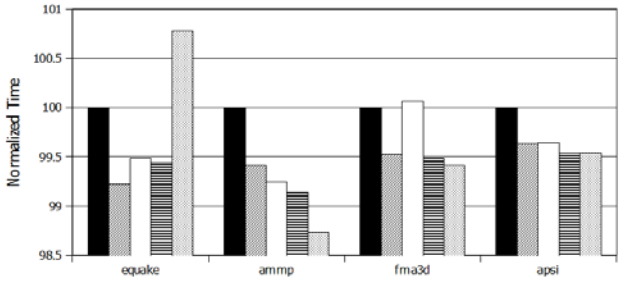


Figure 13. CFP Individual Results Part 3 of 3

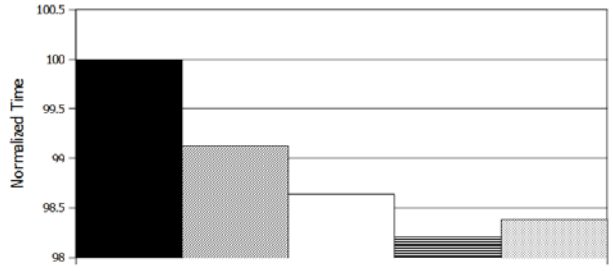


Figure 14. CFP Geometric Mean Results

operation, bank balancing per order has the poorest average performance for the Triad operation.

The performance of all the STREAM operations are summarized in Figure 9, with execution times normalized against the unmodified kernel. Global bank balancing with and without a free list queue consistently performs better than the unmodified kernel. The greatest improvement is in the Triad operation using bank balancing with a free list queue with approximately a 0.4% decrease in average execution time.

5.3 SPEC2000

Figure 10 shows the normalized execution time for five of the SPEC2000 integer benchmarks. Bank balancing results in the greatest reduction in execution time at approximately a 0.6% reduction, for the gzip, parser, and bzip2 benchmarks. Bank balancing shows a slight improvement for the vpr benchmark, whereas the other modifications have the same or worse performance. Using a queue for the free pages reduced the execution time of the twolf benchmark by approximately 0.5%, better than any other modification.

The individual results of the SPEC2000 floating point benchmarks are shown in Figure 11, Figure 12 and Figure 13, with the observation that each figure has a unique scale on the vertical axis. The most interesting results are in Figure 11, which contains the benchmarks that demonstrated the greatest improvement in performance. When a queue of free pages was used for the art benchmark, the execution time was reduced by almost 8.5%. Bank balancing using a queue gave similar results of near 8% improvement. All the modified kernels result in close to 5% improvement in the sixtrack benchmark and 2% for the swim benchmark. The remaining SPEC FP benchmarks only show marginal improvement, if any, generally less than 1%. The geometric mean of all the benchmarks, shown in Figure 14, indicates that using a free list queue has the most performance improvement, closely followed by bank balancing with a free list queue.

5.4 Overall Results

The results of all the benchmarks are summarized in Figure 15. Results were produced by recording execution time on a running workstation with a modified Linux Kernel. They represent many hundreds of executions of more than thirty unique benchmarks. Bank balancing, with and without a queue of free pages, provides improved average performance in all benchmarks. Using per order bank balancing or a queue of free pages provided the highest variability, with typical benchmarks having a small increase in performance while other benchmarks may experience a slight decrease in performance. On average all the modifications provide a comparable reduction in execution time, approximately 0.5%.

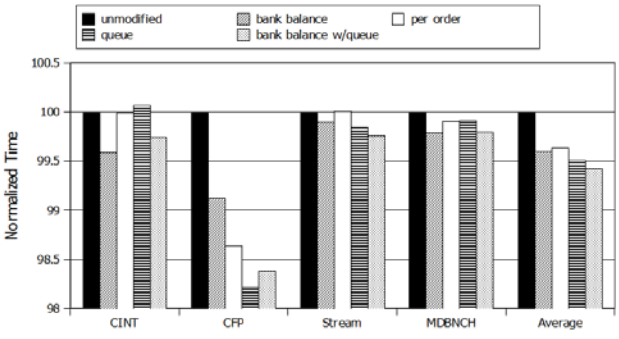


Figure 15. Overall Results

6. Conclusion

Experimental results show that the page allocation mechanisms used provide some improvement in performance with only limited changes to the kernel and no changes to the hardware. In the case of SPEC2000 floating point benchmarks, multiple virtual page allocation mechanisms provided a modest reduction in execution time. These results provide evidence that utilizing available bank parallelism within virtual page allocation is a practical method of improving performance.

These results validate that using DRAM organizational information to drive virtual page placement has the potential to reduce program execution times. In the most significant cases, these reduction in execution time approach 8%. In general the performance improvements were modest, less than 1%, but this improvement comes at the negligible cost of changing only the virtual page allocation algorithm in the Linux 2.6 kernel. These results motivate the development of more advanced algorithms which can determine application behavior, and tailor the allocation mechanism to reduce access time or increase parallelism.

7. Future Work

For DRAM cognizant virtual page placement to gain acceptance, it must be transparent to the end user; this would require the operating system to be able to identify the DRAM construction and behavior of an arbitrary system on boot. The software modifications used in this work have been designed for a very specific hardware configuration, that is an Intel 915PM chipset with two 512MB DIMMs, each DIMM having 8 banks of 64MB each with a DRAM line size of 16KB. For practical use, the specific information required to be identified includes the total number of banks in main memory and which physical address bits select the rank and bank. The most difficult issue is determining the rank and bank select bits because these are determined by the memory controller rather than the user installed DIMMs.

Placement of virtual pages could be improved through the use of hints, profile data, or predictions of access frequency from the compiler. This possible path for research is being examined in collaborative activities.

Lastly, if virtual page placement can be used to distribute pages to unique DRAM banks to improve performance, then virtual page placement can also be used to dynamically migrate all virtual pages to a single DRAM bank in a lightly loaded environment. This could enable removing power from DRAM components and reducing system power consumption.

8. References

[1] Bassett, D. G., “Modifying page allocation in the Linux 2.6 Kernel to increase bank parallelism.”, Master’s Thesis, College of Engineering, Michigan Tech, 2008.

[2] Belady, L. A., “A study of replacement algorithms for a virtual-storage computer,” IBM Systems Journal, pp. 5:78-101, 1966.

[3] Carter, J., Hsieh, W., Stoller, L., Swanson, M., Zhang, L., Brunvand, E., Davis, A., Kuo, C. C., Kuramkote, R., Parker, M., Schaelicke, L. and Tateyama, T., “Impulse: Building a Smarter Memory Controller,” HPCA-5, Washington, DC, pp. 70, 1999.

[4] Denning, P. J. 1970. Virtual Memory. ACM Comput. Surv. 2, 3 (Sep. 1970), 153-189. DOI= http://doi.acm.org/10.1145/356571.356573

[5] Feng, Y. and Berger, E. D. 2005, “A locality-improving dynamic memory allocator,” Proceedings of the 2005 Workshop on Memory System Performance, Chicago, IL, pp. 68-77, June 2005.

[6] Glass, G. and Cao, P., “Adaptive page replacement based on memory reference behavior,” ACM SIGMETRICS international Conference on Measurement and Modeling of Computer Systems, Seattle, WA, pp. 115-126, June 1997.

[7] Hennessy, J. L. and Patterson, D. A., “Computer Architecture: A Quantitative Approach,” Morgan Kaufmann Publishers, San Fransisco, CA, 1996.

[8] Jacob, B. and Mudge, T, "Virtual Memory: Issues of Implementation," IEEE Computer, vol. 31, no. 6, pp. 33-43, June 1998.

[9] Jacob, B., Ng, S. W. and Wang, D.T., “Memory Systems: Cache, DRAM, Disk,” Burlington, MA, Morgan Kaufmann Publishers, 2008.

[10] Jula, A. and Rauchwerger, L, “Two memory allocators that use hints to improve locality,” ISMM 2009, Dublin, Ireland, pp. 109-118, June 2009.

[11] Knowlton, K. C., “A fast storage allocator,” Communications of the ACM, v.8, n.10, pp. 623-624, Oct. 1965

[12] Linux Cross Reference LXR http://lxr.linux.no/

[13] McCalpin, J.. “STREAM: Sustainable memory bandwidth in high performance computers”. http://www.cs.virginia.edu/stream/, 1999.

[14] McKee, S. A., “Reflections on the memory wall,” in Proceedings of the 1st Conference on Computing Frontiers, Ischia, Italy, pp. 162, April 2004.

[15] Prince, B., “High Performance Memories: New Architecture DRAMs and SRAMs -Evolution and Function,” John Wiley & Sons, 1999.

[16] Rokicki, T., “Indexing Memory Banks to Maximize Page Mode Hit Percentage and Minimize Memory Latency,” Hewlett-Packard Laboratories Technical Report, HPL-96-95, June 1996.

[17] Shao, J., “Reducing Main Memory Access Latency through SDRAM Address Mapping Techniques and Access Reordering Mechanisms.”, Master’s Thesis, College of Engineering, Michigan Tech, 2006.

[18] VanderWiel, S. and Lilja, D. J., “A survey of data prefetching techniques,” Tech. Rep. 5, University of Minnesota, Oct. 1996.

[19] Wilson, P. R., Johnstone, M. S. , Neely, M. and Boles, D. “Dynamic Storage Allocation: A Survey and Critical Review.” University of Texas at Austin. 1995.

[20] Xu, Y., Agarwal, A. S., and Davis, B. T. 2009. “Prediction in Dynamic SDRAM Controller Policies.” In Proceedings of the 9th International Workshop on Embedded Computer Systems: Architectures, Modeling, and Simulation, 128-138. DOI= http://dx.doi.org/10.1007/978-3-642-03138-0_14

Standby Power Reduction Techniques for Asynchronous Circuits with Indeterminate Standby States

Liang Zhou and Scott C. Smith (contact author)
Department of Electrical Engineering, University of Arkansas, Fayetteville, AR
kingdom701@gmail.com and smithsco@uark.edu

***Abstract*—In the literature, some schemes were proposed to combine Multi-Threshold CMOS (MTCMOS) with asynchronous circuits to reduce standby power consumption. However, all of these can only be applied to asynchronous circuits in which the values of all signals can be determined in the standby state. As a result, their applications are limited. To solve this problem, this paper develops standby power reduction techniques which not only combine MTCMOS with asynchronous circuits, but also can be applied to asynchronous circuits with indeterminate standby states. Compared to asynchronous circuits implemented with all regular threshold transistors, the proposed implementation has significantly less standby power, reasonable speed penalty, and negligible area overhead.**

Keywords-asynchronous circuits; Multi-Threshold CMOS (MTCMOS); NULL Convention Logic (NCL)

I. INTRODUCTION

With the current trend of semiconductor devices scaling into the deep submicron region, design challenges that were previously minor issues have now become increasingly important. Where in the past, dynamic, switching power has been the predominant factor in CMOS digital circuit power dissipation, recently, with the dramatic decrease of supply and threshold voltages, a significant growth in leakage power demands new design methodologies for digital integrated circuits (ICs). The main component of leakage power is sub-threshold leakage, caused by current flowing through a transistor even if it is supposedly turned off. Sub-threshold leakage increases exponentially with decreasing transistor feature size.

Among the many techniques proposed to control or minimize leakage power in deep submicron technology, Multi-Threshold CMOS (MTCMOS) [1], which reduces leakage power by disconnecting the power supply from the circuit during standby mode while maintaining high performance in active mode, is very promising. MTCMOS incorporates transistors with two or more different threshold voltages (V_t) in a circuit. Low-V_t transistors offer fast speed but have high leakage, whereas high-V_t transistors have reduced speed but far less leakage current. MTCMOS combines these two types of transistors by utilizing low-V_t transistors for circuit switching to preserve performance and high-V_t transistors to gate the circuit power supply to significantly decrease sub-threshold leakage.

Quasi-delay-insensitive (QDI) asynchronous circuits [2] designed using CMOS exhibit an inherent standby behavior since they only switch when useful work is being performed; however, there is still significant leakage power during standby mode. In the literature, some schemes were proposed to combine multi-threshold CMOS (MTCMOS) with asynchronous circuits to reduce standby power consumption, as discussed below.

MTNCL [3-6] combines the MTCMOS technique with full-word pipelined NULL Convention Logic (NCL) asynchronous circuits [7] to sleep the NCL circuit during standby mode. It has significantly less standby power, higher speed, and requires less area than the original NCL circuits implemented with either all low-V_t or high-V_t transistors.

Bit-Wise MTNCL (BWMTNCL) [8] combines the MTCMOS technique with NCL circuits. Compared to original NCL circuits implemented with all low-V_t and high-V_t transistors, respectively, it provides the leakage power advantages of the all high-V_t NCL implementation with a reasonable speed penalty compared to the all low-V_t design, and has negligible area overhead.

Fine-grain leakage power reduction method in [9] combines the MTCMOS technique with asynchronous circuits synthesized with Boolean gates. It utilizes high-V_t transistors for the off-state transistors whose gate input signals are inactive in the standby phase. Compared to original asynchronous circuits implemented with regular-V_t transistors, it has significantly less standby power, reasonable speed penalty, and negligible area overhead.

Static power reduction techniques in [10] combine the MTCMOS technique with Pre-Charge Half Buffer (PCHB) [2] asynchronous circuits. Compared to original PCHB asynchronous circuits implemented with regular-V_t transistors, it has less standby power, reasonable speed penalty, and reasonable area overhead.

However, all of the above can only be applied to asynchronous circuits in which the values of all siganls can be determined in the standby state. As a result, their applications

are limited. To solve the problem, this paper develops standby power reduction techniques which not only combine MTCMOS with asynchronous circuits, but also can be applied to asynchronous circuits with indeterminate standby states. Compared to asynchronous circuits implemented with all regular-V_t transistors, the proposed implementation has significantly less standby power, reasonable speed penalty, and negligible area overhead. Although the proposed techniques are illustrated by extending BWMTNCL, they can also be applied to other asynchronous circuit paradigms.

Section II provides an overview of NCL, BWMTNCL, and an NCL unsigned 32+16×16 Multiply and Accumulate (MAC) unit, as an example with indeterminate standby states. Section III details the proposed techniques to handle indeterminate standby states; Section IV compares the various implementations; and Section V provides conclusions.

II. Previous Work

A. Introduction to NCL

NCL circuits utilize multi-rail logic, such as dual-rail, to achieve delay-insensitivity. A dual-rail signal, *D*, consists of two wires, D^0 and D^1, which may assume any value from the set {DATA0, DATA1, NULL}. The DATA0 state ($D^0 = 1$, $D^1 = 0$) corresponds to a Boolean logic 0, the DATA1 state ($D^0 = 0$, $D^1 = 1$) corresponds to a Boolean logic 1, and the NULL state ($D^0 = 0$, $D^1 = 0$) corresponds to the empty set meaning that the value of *D* is not yet available. The two rails are mutually exclusive, such that both rails can never be asserted simultaneously; this state is defined as an illegal state.

NCL circuits are comprised of 27 fundamental gates [11]. These 27 gates constitute the set of all functions consisting of four or fewer variables. The primary type of threshold gate, shown in Fig. 1, is the THmn gate, where $1 \le m \le n$. THmn gates have *n* inputs. At least *m* of the *n* inputs must be asserted before the output will become asserted. NCL threshold gates are designed with hysteresis state-holding capability such that all asserted inputs must be de-asserted before the output will be de-asserted, as shown in Fig. 2. Therefore, a THnn gate is equivalent to an n-input C-element [12] and a TH1n gate is equivalent to an n-input OR gate. NCL threshold gates may also include a reset input to initialize the output. These resettable gates are used in the design of DI registers [13].

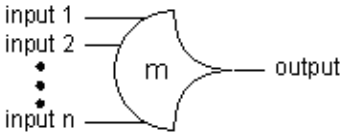


Figure 1. THmn threshold gate.

NCL systems contain at least two delay-insensitive (DI) registers, one at both the input and at the output, and can be finely pipelined by inserting additional registers, as shown in Fig. 3. Two adjacent register stages interact through their request and acknowledge signals, *Ki* and *Ko*, respectively, to prevent the current DATA wavefront from overwriting the previous DATA wavefront, by ensuring that the two DATA wavefronts are always separated by a NULL wavefront. The acknowledge signals are combined in the Completion Logic to produce the request signal(s) to the previous register stage, utilizing either the full-word or bit-wise completion strategy [13].

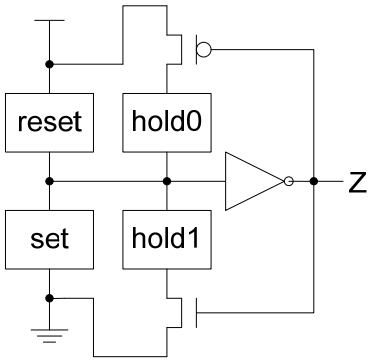


Figure 2. NCL threshold gate design.

To ensure delay-insensitivity, NCL circuits must adhere to the following criteria: Input-Completeness and Observability [14]. Input-Completeness requires that all outputs of a combinational circuit may not transition from NULL to DATA until all inputs have transitioned from NULL to DATA, and that all outputs of a combinational circuit may not transition from DATA to NULL until all inputs have transitioned from DATA to NULL. Observability requires that no orphans may propagate through a gate. An orphan is defined as a wire that transitions during the current DATA wavefront, but is not used in the determination of the output.

B. Introduction to Bit-wise MTNCL

In NCL systems without feedback loops, the inputs of each gate while in the standby state are determinate, since all circuit inputs will be NULL, which causes all Combinational Logic gates and the data inputs and outputs of all registers to be de-asserted, which in turn causes all Completion Logic gates to be asserted (i.e., request-for-data or *rfd*), as shown in Fig. 3.

The *leakage path* is defined as the path formed by "on" transistors and "off" low-V_t transistors in the standby state. To substantially reduce leakage power while degrading speed as little as possible, the following rules should be utilized to determine which transistors should be high-V_t and which transistors should be low-V_t [8]:

1. Determine threshold gate input and output values in standby state.
2. All transistors "on" in standby state should be low-V_t.
3. Replace the minimal number of "off" transistors with high-V_t transistors to eliminate leakage path, and replace the rest with low-V_t transistors.

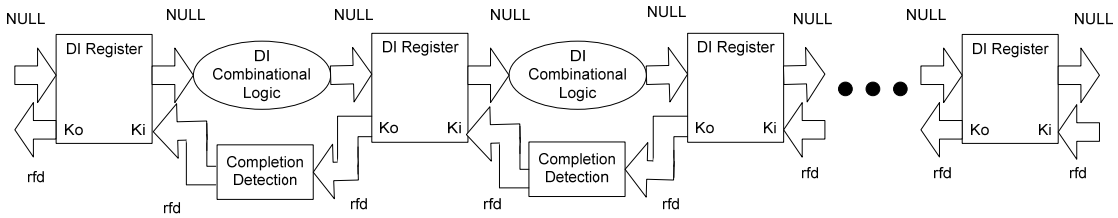


Figure 3. NCL system framework without feedback loops in standby state.

Fig. 4 shows the standby states of a 1-bit dual-rail NCL register, which consists of two TH22 resettable to '0' gates with A = '0', B = '1', reset = '0', and one inverted TH12 gate with both '0' inputs in standby state. After applying those 3 rules, the schematic of the TH22 gate is given in Fig. 5, in which high-V_t transistors are circled and low-V_t transistors are not. *T0*, *T1*, and *T2* are high-V_t because they do not switch except for initialization.

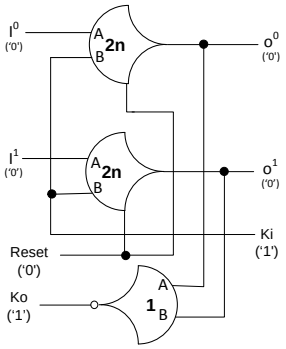


Figure 4. 1-Bit NCL register in standby state.

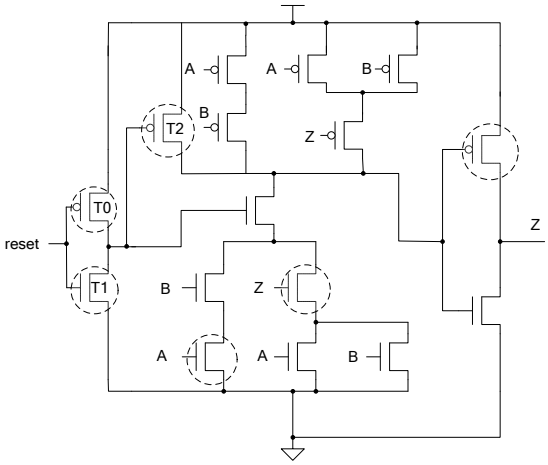


Figure 5. BWMTNCL applied to TH22 resettable to '0' gate with a single standby state: reset = '0', A = '0', B = '1', Z = '0'

Compared to original NCL circuits implemented with all low-V_t and high-V_t transistors, respectively, BWMTNCL provides the leakage power advantages of the all high-V_t NCL implementation with a reasonable speed penalty compared to the all low-V_t design, and has negligible area overhead.

C. Asynchronous Circuits with Indeterminate Standby States

As an example of an asynchronous circuit with indeterminate standby states, an NCL unsigned 32+16×16 MAC is developed. As shown in Fig. 6, it consists of 3 parts: full-word pipelined 7-stage partial product generation and Wallace tree summation circuit (PP1, PP2), a 4-stage feedback loop which feeds back the accumulator as 2 partial products in carry-save form (A1, A2), and full-word pipelined 15-stage 30-bit Ripple-Carry-Adder. The architecture is elaborated in [15], except that it is full-word pipelined and extra control functions are removed for simplicity.

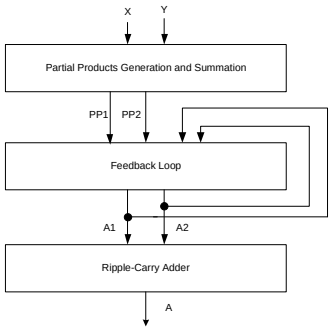


Figure 6. NCL MAC architecture

Fig. 7 shows the standby states of the feedback loop, which has 2-level carry save adders to sum up new partial products, *PP1* and *PP2*, and old accumulator, *A1* and *A2*, to generate new accumulator in carry-save form. Partial DATA in Fig. 7 means that some bits of the register are DATA while the other bits are NULL.

In standby state, the old accumulator values are stored in register, REG0, in Fig. 7. Each bit of REG0 can either be DATA0 or DATA1, which cannot be determined at design time. Similarly, register, REG1, Combinational Logic, COMB1 and COMB2, and Completion Logic, COMP0, also have indeterminate standby states. Therefore, BWMTNCL cannot be applied to this feedback loop.

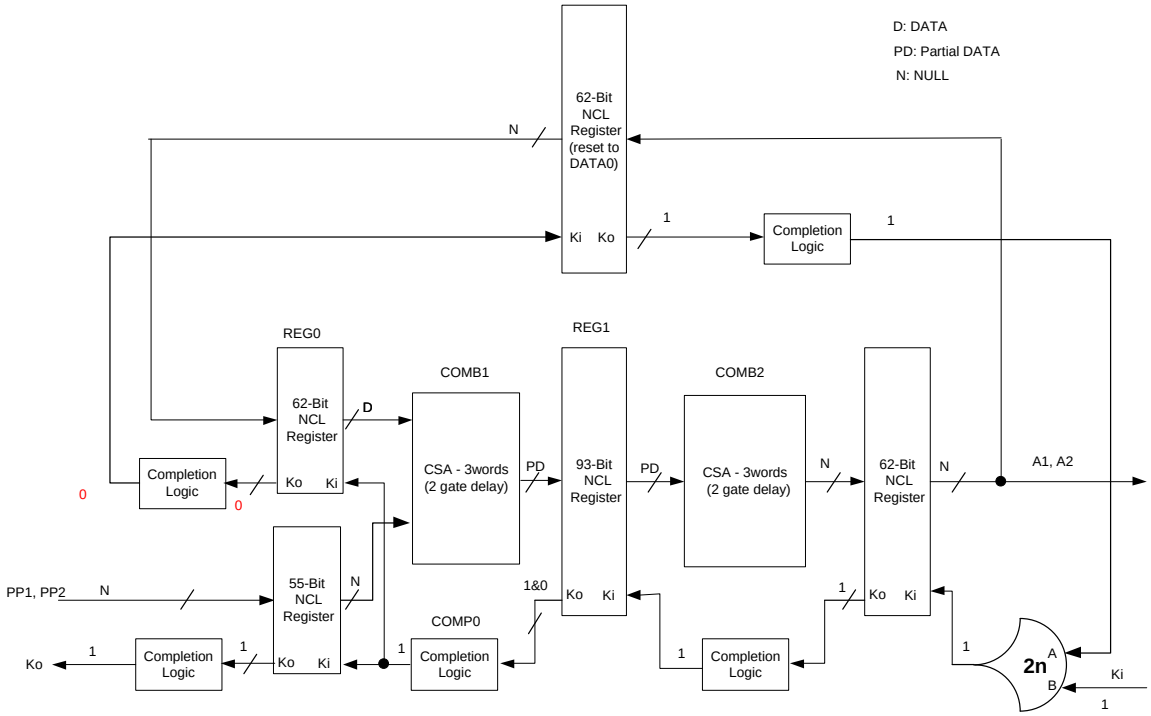


Figure 7. Standby states of the feedback loop without indeterminate states reduction

III. TECHNIQUES TO HANDLE INDETERMINATE STANDBY STATES

In asynchrnous circuits with indeterminate standby states, usually not all of the inputs and outputs of a threshold gate are indeterminate in the standby state. For example, TH22 resettable to '0' gates used in REG0 in Fig. 7 have determinate standby states for A, B, and reset inputs (i.e., A = '0', B = '1', reset = '0'), but an indeterminate standby state for output, *Z* (i.e., Z = '1' or Z = '0'). In other words, it has 2 possible standby states. Instead of using an all high-V_t implementation, those 2 standby states can be analyzed to use the minimal number of high-V_t transistors to eliminate the leakage path in either of the two possible standby states, in order to substantially reduce leakage power while degrading speed as little as possible. The rules used in BWMTNCL can be enhanced as follows:

1. Determine the number of standby states and threshold gate input and output values in each standby state.
2. Replace the minimal number of transistors with high-V_t transistors to eliminate leakage path in any standby state, and replace the rest with low-V_t transistors.

After applying these 2 rules, the schematic of the TH22 gate with 2 possible standby states is given in Fig. 8.

By comparing Fig. 5 with Fig. 8, it can be observed that Fig. 8 has 3 more high-V_t transistors than Fig. 5. These 3 additional high-V_t transistors are required by the extra standby state and make the TH22 gate in Fig. 8 slower than in Fig. 5. Therefore, it is beneficial to reduce the number of gates with indeterminate standby states so that fewer high-V_t transistors are required, to degrade speed as little as possible.

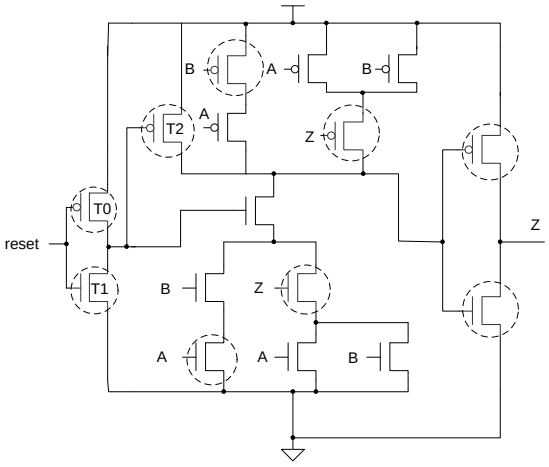


Figure 8. Enhanced BWMTNCL applied to TH22 resettable to '0' gate with 2 standby states: reset = '0', A = '0', B = '1', Z = '0' or Z = '1'

To reduce the number of gates with indeterminate standby states, an inverter U0 and an asymmetric [16] TH22 gate, U1, are added in Fig. 9 to control the *Ki* input of register, REG0. The input *B* with '+' of U1 only takes effect in asserting the asymmetric TH22 gate. In other words, U1 will be asserted if both inputs are '1', and de-asserted if *A* is '0' regardless of the value of *B*. The preceding partial product generation and Wallace tree summation circuit has output register, REG3, whose *Ko* output is asserted if *PP1* and *PP2* are NULL, and de-asserted if they are DATA.

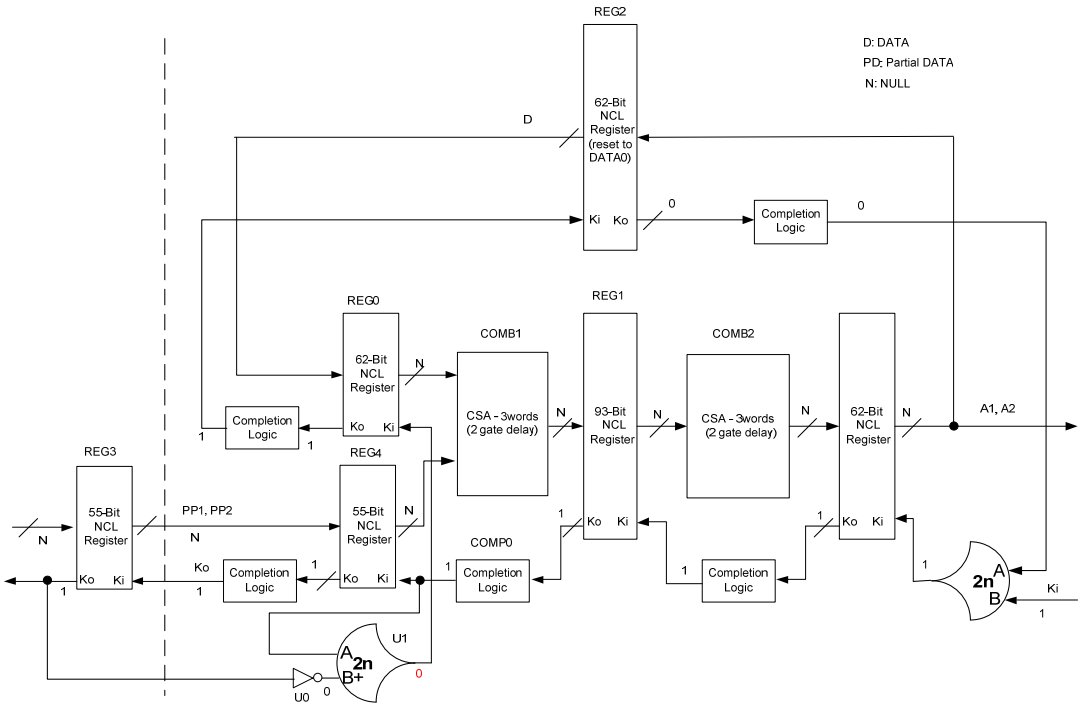


Figure 9. Standby states of the feedback loop with indeterminate states reduction

As a result, the condition for the *Ki* input of REG0 to be de-asserted remains that Completion Logic, COMP0, is de-asserted, but the condition to assert it is changed from COMP0 is asserted to COMP0 is asserted and *PP1* and *PP2* are DATA. The function of the feedback loop in the active state is not changed, because the added condition that *PP1* and *PP2* are DATA is always met in the active state. However, in the standby state, *PP1* and *PP2* are NULL so that the *Ki* input of REG0 cannot be asserted, and the old accumulator values are stored in REG2 instead of REG0. In Fig. 9, only REG2 and REG0 have indeterminate standby states, compared to REG0, REG1, COMB1, COMB2, and COMP0 in Fig. 7. As the number of gates with indeterminate standby states is significantly reduced, fewer high-V_t transistors are required, which degrades speed as little as possible.

The area overhead is negligible as only two gates are added. The added condition is equivalent to adding Combinational Logic with 1 gate delay between REG3 and REG4. As the speed of the whole circuit is limited by the slowest stage [13], which is the feedback loop which has much longer delay than the stage between REG3 and REG4, the speed of the whole circuit will not be degraded by adding those two gates.

Fig. 9 requires 4 types of gates with indeterminate standby states. First, REG2 requires TH22 gates resettable to '0' with standby states reset = '0', A = '0', B = '1', Z='0' or Z = '1', which is already shown in Fig. 8. Second, REG2 requires TH22 gates resettable to '1' with standby states reset = '0', A ='0', B ='1', Z='0' or Z = '1', which is shown in Fig. 10. Third, REG2 requires inverted TH12 gates with standby states A = '1', B= '0' or A = '0', B = '1', which is shown in Fig. 11. Finally, REG0 requires TH22 gates resettable to '0' with standby states reset = '0', A = '0' or A = '1', B = '1', Z = '0', which is shown in Fig. 12.

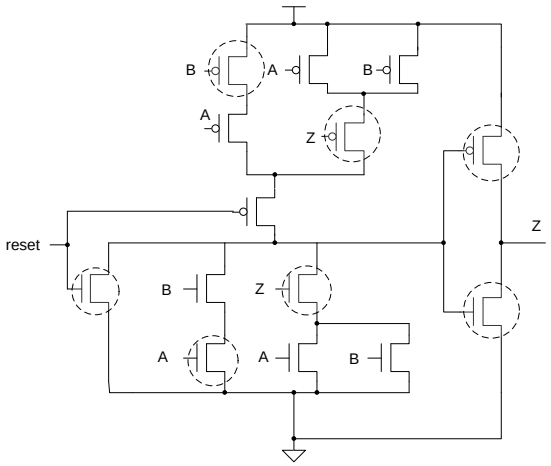


Figure 10. Enhanced BWMTNCL applied to TH22 resettable to '1' gate with 2 standby states: reset = '0', A = '0', B = '1', Z = '0' or Z = '1.

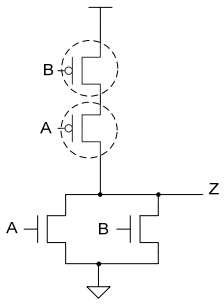


Figure 11. Enhanced BWMTNCL applied to inverted TH12 gate with 2 standby states: A = '0', B = '1' or A = '1' or B = '0'.

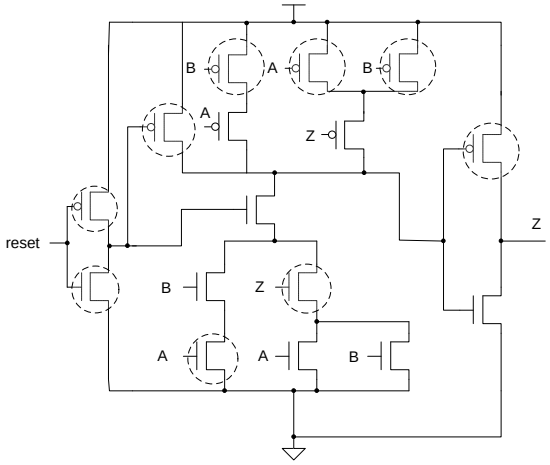


Figure 12. Enhanced BWMTNCL applied to TH22 resettable to '0' gate with 2 standby states: reset = '0', A = '0' or A = '1', B = '1', Z = '0'.

In summary, two techniques to handle indeterminate standby states are proposed. First, add extra gates to reduce indeterminate standby states as much as possible. Second, analyze the standby states of threshold gates and replace the minimal number of transistors with high-V_t transistors to eliminate the leakage path in any possible standby state, and replace the rest with low-V_t transistors.

IV. Simulation Results

To compare the proposed BWMTNCL with indeterminate state reduction design to those without, the 32+16×16 unsigned MAC with feedback loop in Fig. 9 was implemented with enhanced BWMTNCL, and the MAC with feedback loop in Fig. 7 was implemented with regular-V_t transistors, using the 1.2V IBM 8RF-LM 130nm CMOS process. All designs were simulated at the transistor level using Cadence's UltraSim simulator running a VerilogA controller in mixed-signal mode. Note that all transistors are minimum sized except for the buffers used for high fanout signals.

The first two rows of Table I show the results, in which T_{DD} is the average DATA plus NULL processing time, which is comparable to the clock period in a synchronous system. T_{DD} and Energy/Operation are calculated while the circuit is operating at its maximum speed, while Leakage Power is calculated using Cadence Spectre DC analysis after the pipeline is flushed with all NULL inputs. The results show that the proposed design has 36x standby power reduction over the regular design, with 20% speed penalty and 0.02% area overhead.

The 2 MACs were also implemented with all high-V_t and all low-V_t minimum sized transistors, respectively, as the all high-V_t implementation will give the lower bound on standby power and the all low-V_t implementation will give the lower bound on T_{DD}. The results are shown from row 3 to row 6 of Table I, which prove that adding extra gates to reduce indeterminate states does not increase T_{DD}. It also shows that the proposed design provides the leakage power advantage of the all high-V_t implementations with a reasonable speed penalty compared to the all low-V_t implementations.

Table I also shows that the proposed design is more energy efficient than the regular design, and the all low-V_t implementation is the most energy efficient. The relationship between energy/operation, threshold voltage, and transistor sizing will be investigated in the future.

V. Conclusions

None of the asynchronous circuit standby power reduction techniques in the literature are able to handle indeterminate standby states. This paper describes enhancements to the asynchronous circuit standby power reduction techniques developed in [8], to make it handle indeterminate standby states. Simulation proves that compared to the regular design, it has significant standby power reduction, reasonable speed penalty, and negligible area overhead.

TABLE I. Simulation Results

	Transistor#	**T_{DD}(ns)**	**Energy/ Operation(pJ)**	**Standby Power(nW)**
Enhanced BWMTNCL with indeterminate states reduction	118176	6.2	36	159.312
Regular-V_t without indeterminate states reduction	118158	4.9	37.5	5761.8
All high-V_t with indeterminate states reduction	118176	9.3	35.3	129.096
All high-V_t without indeterminate states reduction	118158	9.3	35.3	130.56
All low-V_t with indeterminate states reduction	118176	4	34.5	13123.8
All low-V_t without indeterminate states reduction	118158	4	34.5	13255.8

References

[1] S. Mutoh, T. Douseki, Y. Matsuya, T. Aoki, S. Shigematsu, and J.Yamada, "1-V Power Supply High-Speed Digital Circuit Technology with Multithreshold-Voltage CMOS," *IEEE Journal of Solid-State Circuits*, Vol. 30/8, pp. 847-854, August 1995.

[2] A. J. Martin and M. Nystrom, "Asynchronous techniques for system-on-chip design**,"** *Proceedings of the IEEE*, pp. 1089 – 1120, Vol. 94, No. 6, June 2006.

[3] A. D. Bailey, J. Di, S. C. Smith, and H. A. Mantooth, "Ultra-Low Power Delay-Insensitive Circuit Design," *IEEE Midwest Symposium on Circuits and Systems*, pp. 503-506, August 2008.

[4] A. D. Bailey, A. Al Zahrani, G. Fu, J. Di, and S. C. Smith, "Multi-Threshold Asynchronous Circuit Design for Ultra-Low Power," *Journal of Low Power Electronics*, Vol. 4/3, pp. 337-348, December 2008.

[5] A. Alzahrani, A. D. Bailey, G. Fu, and J. Di, "Glitch-Free Design for Multi-Threshold CMOS NCL Circuits," 2009 *Great Lakes Symposiumon VLSI*, May 2009.

[6] S. C. Smith and J. Di, *Designing Asynchronous Circuits using NULL Convention Logic (NCL)*, Synthesis Lectures on Digital Circuits and Systems, Vol. 4/1, July 2009, Morgan & Claypool Publishers (doi: 10.2200/S00202ED1V01Y200907DCS023).

[7] K. M. Fant and S. A. Brandt, "NULL Convention Logic: A Complete and Consistent Logic for Asynchronous Digital Circuit Synthesis," *International Conference on Application Specific Systems, Architectures, and Processors*, pp. 261-273, 1996.

[8] L. Zhou, S. C. Smith, and J. Di, "Bit-Wise MTNCL: An Ultra-Low Power Bit-Wise Pipelined Asynchronous Circuit Design Methodology," *IEEE Midwest Symposium on Circuits and Systems*, pp. 217-220, August 2010.

[9] M. Imai, K. Takada, T. Nanya, "Fine-grain leakage power reduction method for m-out-of-n encoded circuits using multi-threshold-voltage transistors." *IEEE Int. Symp. on Asynchronous Circuit and Systems*, pp. 209 – 216, 2009.

[10] C. Ortega, J. Tse, and R. Manohar, "Static power reduction techniques for asynchronous circuits", *IEEE Int. Symp. on Asynchronous Circuit and Systems*, May 2010, pp. 52-61.

[11] Gerald E. Sobelman and Karl M. Fant, "CMOS Circuit Design of Threshold Gates with Hysteresis," *IEEE International Symposium on Circuits and Systems (II)*, pp. 61-65, 1998.

[12] D. E. Muller, "Asynchronous Logics and Application to Information Processing," in *Switching Theory in Space Technology*, Stanford University Press, pp. 289-297, 1963.

[13] S. C. Smith, R. F. DeMara, J. S. Yuan, M. Hagedorn, and D. Ferguson, "Delay-Insensitive Gate-Level Pipelining," *Elsevier's Integration, the VLSI Journal*, Vol. 30/2, pp. 103-131, October 2001.

[14] S. C. Smith, R. F. DeMara, J. S. Yuan, D. Ferguson, and D. Lamb,"Optimization of NULL Convention Self-Timed Circuits," *Integration, the VLSI Journal*, Vol. 37/3, pp. 135-165, August 2004.

[15] L. Zhou and S. C. Smith, "Speedup of a Large Word-Width High-Speed Asynchronous Multiply and Accumulate Unit," *IEEE Midwest Symposium on Circuits and Systems*, pp. 499-502, August 2009.

[16] S. B. Furber and P. Day, "Four-phase micropipeline latch control circuits," *IEEE Trans VLSI Syst*., vol. 4, pp. 247–253, June 1996.

The Impact of Dust on On-chip Temperature And Performance of Microprocessors

Hyung Beom Jang[1], Cheol Hong Kim[2], and Sung Woo Chung[1]
Division of Computer and Communication Engineering[1],
Korea University, Seoul 136-713, Korea
{kuphy01, swchung}@korea.ac.kr
School of Electronics and Computer Engineering[2],
Chonnam National University, Kwangju 500-757, Korea
chkim22@chonnam.ac.kr

Abstract - *Temperature is a dominant factor in the performance, reliability, and leakage power consumption of modern microprocessors. With the advance of technology, thermal problems in microprocessor design get more crucial due to power density of microprocessors. As a result, more researchers have investigated thermal-aware techniques considering many factors such as ambient temperature [1]and characteristics of an application, which affect on-chip temperature of microprocessors. In fact, there is much possibility that dust in a heat sink deteriorates thermal problems of microprocessors. However, the impact of dust on on-chip temperature has never been studied. In this paper, we show that a heat sink covered with dust increases on-chip temperature, ultimately leading to performance degradation. Our evaluation results show that the performance of the same DTM (Dynamic Thermal Management) scheme is decreased, when we use a heat sink covered with dust (compared to that when we use a clean heat sink). Therefore, we recommend that computer users should clean their heat sinks for better performance.*

Keywords: Microprocessor, On-chip Temperature, Thermal Management, Performance Evaluation, Dust

1 Introduction

As process technology scales down, power density of microprocessors continuously increases leading to higher on-chip temperature. Excessively high on-chip temperature of microprocessors in turn threatens timing stability and lifetime reliability, resulting in physical damages in the worst case. In addition to reliability, temperature is tightly related to performance, leakage power, and cooling cost; i) performance can be gracefully sacrificed to alleviate thermal problems, ii) leakage power can be reduced by the decreased temperature, since leakage power is more than linearly proportional to temperature, and iii) cooling cost is increased to reduce temperature. In modern microprocessor design, temperature has become one of the most crucial considerations. In the past, thermal problems were resolved in the device or circuit level, which is not enough in the current technology. Thus, architectural thermal management techniques such as DTM (Dynamic Thermal Management) [6] were proposed. Naturally, there has been a significant increase of thermal-related publications in architectural conferences and journals.
On the other hand, there are some factors to affect on-chip temperature of microprocessors in addition to on-chip power dissipation. Ambient temperature affects on-chip temperature of microprocessor, since it varies across applications depending on convective heat flux from heat sources such as HDD, DRAM, and so on. Additionally, dust in a heat sink also increases on-chip temperature by preventing a heat sink from dissipating heat flux. However, as far as we know, there has not been any study on the impact of dust on on-chip temperature. Since on-chip temperature of microprocessors is

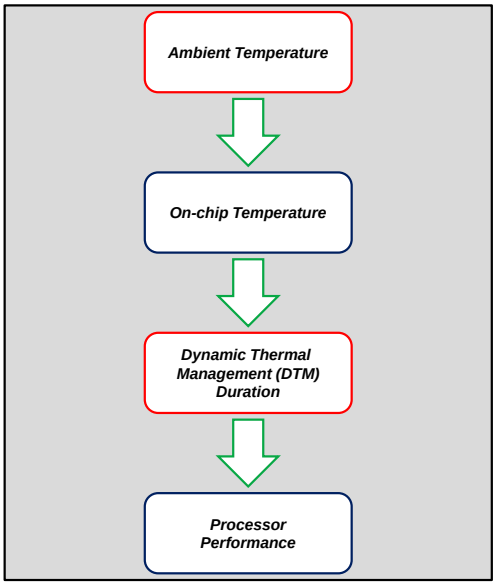


Fig. 1. The effect of ambient temperature on performance

[1] In this paper, ambient temperature denotes air temperature between processor and heat sink. Note that it is not room temperature.

highly affected by dust in the heat sink, considering the impact of dust on on-chip temperature is crucial. Note that it is easy to find dust on the heat sink of running processors. In this paper, we analyze the impact of dust on on-chip temperature and performance.

The rest of this paper is organized as follows. Section II explains previous work on architectural thermal management methods including architectural methodology to analyze the impact of ambient temperature. Section III describes experimental environments. Section IV analyzes the impact of dust on the heat sink on ambient temperature and performance. Section V concludes this paper.

2 Previous Work

For thermal management of processors, various techniques have been proposed. One of them is the DTM technique which dynamically controls power dissipation of processors referring to on-die temperature from CMOS thermal sensors, while trying to minimize performance degradation. In [1], thermal control techniques, such as DVFS (Dynamic Voltage Frequency Scaling) and decode throttling, were proposed for DTM. In DVFS, the clock frequency is dynamically scaled along with the supply voltage to reduce processor power, resulting in on-chip temperature reduction. On the other hand, in decode throttling, processor core is throttled by restricting the flow of instructions for reducing power consumption and on-chip temperature. However, in their efforts, they did not consider the impact of ambient temperature on on-chip temperature. Recognizing the importance of ambient temperature on on-chip temperature variation, researchers have tried to propose systematic thermal management schemes to reflect ambient temperature that does change depending on applications. Choi et al. incorporated ambient temperature analysis capabilities with their simulation methodology based on the CFD (Computational Fluid Dynamics) simulation [12]. In [3], Jang et al. found that different ambient temperatures should be used to evaluate a thermal management technique for different applications. They also evaluated the impact of application-dependent ambient temperature on the thermal evaluation results, such as performance and leakage power.

One can easily guess the effect of the dust in a heat sink on performance. Since dust in a heat sink deteriorates heat dissipation from a processor to outside, ambient temperature is increased; note ambient temperature represents air temperature between processor and heat spreader, not room temperature. The increased ambient temperature in turn leads to higher on-chip temperature, causing more frequent DTM invocations and eventually performance degradation, as shown in Fig. 1. Eventually, the dust degrades performance. Since computer users do not generally clean their computers, it is very probable that their heat sinks are covered with heavy dust. However, the effect of dust in a heat sink has never been quantitatively evaluated. In this paper, we analyze the impact of dust on on-chip temperature.

3 Experimental Environments

In this paper, we investigate the impact of dust on on-chip temperature by measuring ambient temperature that is not room temperature but air temperature between processor and heat spreader. If there is heavy dust in the heat sink, heat is not efficiently dissipated due to lowered heat dissipation capability of the heat sink. Naturally, ambient temperature between processor and heat spreader is increased. For measuring ambient temperature of the processor, we place off-chip thermal sensors (SEN-AP002P from Koolance Corporation [7]) as shown in Fig. 2 for two different heat sinks (a heat sink covered with dust and a clean heat sink). We use the computer system equipped with the 45nm Intel Core2Duo processor typically consuming 35 watts [9], 2GB Hynix DDR3 synchronous DRAM with the maximum TDP (Thermal Design Power) of 4.3 watts [11], a Hitachi 320GB SATA HDD with the activate-state TDP of 2.2 watts [8], an Intel GM45 north bridge chipset, a printed circuit board (PCB), a finned heat sink, an exhaust fan, and un-powered devices. The Intel GM45 north bridge I/O chipset of 12.0 watts [9] includes a graphic processing unit. For benchmark applications, we select ten applications from SPEC CPU2000 benchmark suite [10].

Additionally, to investigate the impact of dust on processor performance, we evaluate the DTM duration ratio [2] of the DTM schemes. We use the DVFS and stop-go (also known as global clock gating) schemes for DTM with the emergency temperature of 358K (85ºC) [9]. For the DVFS scheme, the Core2Duo processor supports three voltage/frequency pairs of 800, 1600, and 2533MHz that correspond to 0.95, 1.1, and 1.25V supply voltage, respectively. However, to consider the imprecision caused by on-chip thermal sensors; random noise and potential [6], we conservatively set the DTM invocation temperature to 355K (83ºC). In case of the stop-go scheme, when the temperature of the processor reaches 357K (84ºC)

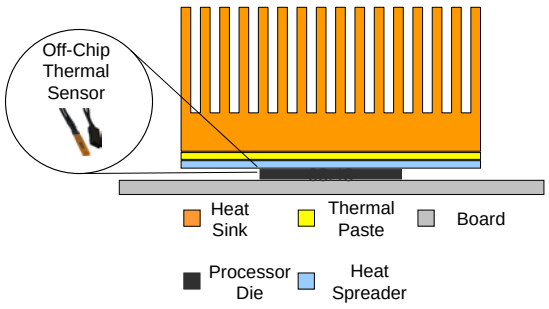


Fig. 2. Off-chip thermal sensor placement to measure ambient temperature

[2] DTM duration ratio = (accumulated time when DTM is invoked) / (total execution time)

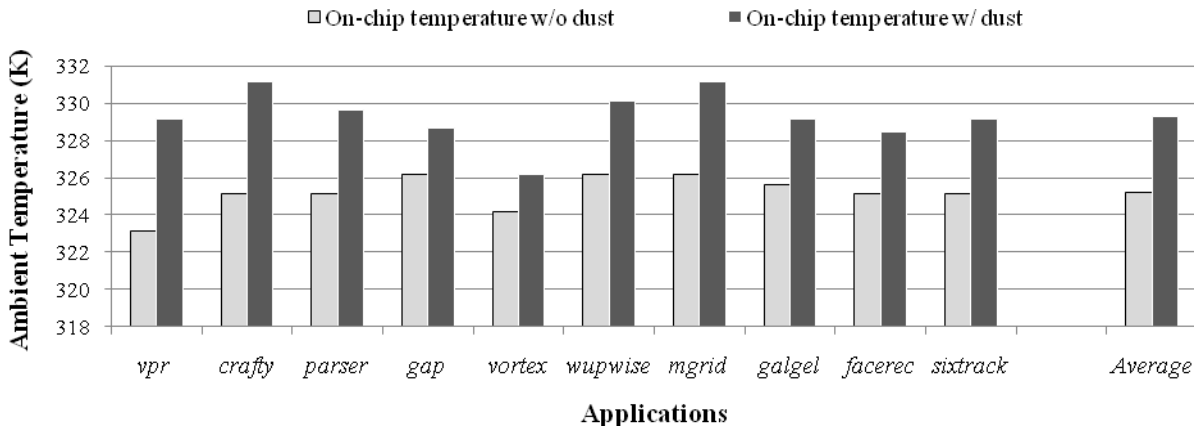


Fig. 3. Ambient temperature comparison.

[4], the processor is stalled for 10ms (doing nothing) by a thermal interrupt. When the temperature of the processor goes down below the threshold temperature, the processor resumes executions at the highest voltage/frequency pair.

4 Experimental Results

Fig. 3 shows ambient temperature for ten applications from SPEC CPU2000 benchmark suite, where ambient temperatures for two different heat sinks (a heat sink covered with dust and a clean heat sink) are depicted. The dust prevents a heat sink from dissipating heat flux from processors. Thus, heat flux from the processor is not be transferred to the cooling fan resulting in an increase of ambient temperature. With the heat sink covered with dust, ambient temperature of the processor is increased by up to 6 degrees, in case of *vpr* and *crafty*, since these applications consume large power over TDP with long execution time. In case of *vortex*, the ambient temperature is increased by 2 degrees showing the smallest ambient temperature increase due to the short execution time. When the execution time is short, the effect of dust is not so noticeable since heat is not propagated to the heat sink. The applications of ambient temperatures, which have very long execution time and large power consumption as much as the TDP, are increased over 3 degrees in *parser*, *mgrid*, *facerec*, and *sixtrack*. On average, ambient temperature of the processor is increased by 4 degrees due to the dust. The increased ambient temperature also leads to an on-chip temperature increase and the increased on-chip temperature of the processor in turn degrades performance. When the ambient temperature difference is 4.5, 1.1, and 2.6 degrees, on-chip temperature difference is 3.5, 0.7, and 2.0 degrees, on average, respectively for SPEC CPU2000 benchmark suite.

We estimate the performance differences of the processor with a dusty heat sink compared to that with a clean heat sink. Table 1 represents the DTM duration ratio and execution time for two DTM schemes (DVFS and stop-go) with different on-chip temperature, which are obtained from [3]; they ran all the applications from SPEC CPU2000 benchmark suite. When the difference of on-chip temperature ranges from 3 to 4 degree, the DTM duration ratio of the processor varies from 0% to 6.91% in both of DTM schemes, which is directly affected by on-chip temperature. The execution time of the DVFS scheme and stop-go scheme varies from 0.35% to 5.4% and from 0.67% to 9.77%, respectively. As ambient temperature difference is increased leading to on-chip temperature difference, the DTM duration ratio of the processor is increased, which it also increases execution time. In other words, the DTM technique frequently lowers the DVFS level to reduce the temperature of the processor below emergency

Table 1. The impact of on-chip temperature variation on performance of the processor

On-chip Temperature Difference	DTM Duration Ratio Variation		Execution Time Variation	
	DVFS Scheme	Stop-go Scheme	DVFS Scheme	Stop-go Scheme
0 ~ 1 degrees	0% ~ 0.08%	0% ~ 0.08%	0% ~ 9.88%	0.03% ~ 2.71%
1 ~ 2 degrees	0.11%~ 32.32%	0% ~ 2.66%	0.08%~ 3.96%	0.15% ~ 5.31%
2 ~ 3 degrees	0.24% ~ 0.28%	0.64% ~ 1.69%	0.24% ~ 0.5%	1.09% ~ 2.62%
3 ~ 4 degrees	0% ~ 6.91%	0% ~ 4.85%	0.35% ~ 5.4%	0.67% ~ 9.77%
Over 4 degrees	0.03% ~ 77.5%	0.02% ~ 25.74%	0.22% ~ 54.66%	0.02% ~ 73.88%

temperature, resulting in longer execution time. From evaluation results, we can know that the dust affects on-chip temperature variation and performance results which are not so small to be ignored.

5 Conclusion

In conventional thermal researches, on-chip temperature and performance have been evaluated without consideration of application/system dependent ambient temperature. Recently, there was a study to consider different ambient temperatures when evaluating different applications. However, there has not been any quantitative study to evaluate the impact of the dust in a heat sink on on-chip temperature, though the dust definitely deteriorates heat dissipation efficiency. In this paper, we found that the dust that has been ignored by computer users may deteriorate performance by up to 73.9%. Hence, we recommend that future computer users should remove the dust that is in their heat sinks for better performance.

Acknowledgement

This research was supported by the MKE(The Ministry of Knowledge Economy), Korea, under the ITRC(Information Technology Research Center) support program supervised by the NIPA(National IT Industry Promotion Agency) (NIPA-2011-C1090-1121-0010).

References

[1] D. Brooks and M. Martonosi, "Dynamic Thermal Management for High-Performance Microprocessors," in *Proc. 7th Int'l Symp. High-Performance Computer Architecture*, pp. 171-82, Jan. 2001.

[2] J. Donald and M. Martonosi, "Techniques for Multicore Thermal Management: Classification and New Exploration," *ACM SIGARCH Computer Architecture News*, pp. 78-88, May 2006.

[3] H. B. Jang, J. Choi, I. Yoon, S. –S. Lim, S. Shin, N. Chang, and S. W. Chung, "Exploiting Application-dependent Ambient Temperature for Accurate Architectural Simulation," in *Proc. 28th IEEE Int'l. Conf. on Computer Design*, Accepted, Oct. 2010.

[4] Y. Kim, S. Gurumurthi, and A. Sivasubramaniam, "Understanding the Performance-Temperature Interactions in Disk I/O of Server Workloads," in *Proc. 20th Int'l Symp. High-Performance Computer Architecture*, pp. 176-186, Feb. 2006.

[5] S. Liu, S. O. Memik, Y. Zhang, and G. Memik, "An Approach for Adaptive DRAM Temperature and Power Management," in *Proc. 22nd Annu. Int'l Conf. Supercomputing*, pp. 63-72, Jun. 2008.

[6] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan, "Temperature-Aware Microarchitecture," in *Proc. 30th Annu. Int'l Symp. Computer Architecture*, pp. 2-13, Jun. 2003.

[7] Koolance, Flat Thermal Sensor SEN-AP002P Available: http://www.koolance.com/water-cooling /product_info.php?product_id=194

[8] *Hitachi Travelstar 5K320 Hard Disk Drive Specification*, Hitachi Corporation, Apr. 2008.

[9] *Intel Core™2Duo Mobile Processor, Intel Core™2 Solo Mobile Processor and Intel Core™2 Extreme Mobile Processor on 45-nm Process Datasheet*, Intel Corporation, Mar. 2009.

[10] SPEC, *Standard Performance Evaluation Corporation*. Available: http://www.spec.org/cpu2000/

[11] Micron, *System-Power Calculator*. Available: http://www.micron.com/support/part_info/powercalc

[12] *User's guide, Icepak, 4.4.6*. ANSYS/Fluent Inc., Lebanon, NH, 2007.

Performance Improvement by N-Chance Clustered Caching in NoC based Chip Multi-Processors

Rakesh Yarlagadda, Sanmukh R Kuppannagari, and Hemangee K Kapoor
Department of Computer Science and Engineering
Indian Institute of Technology Guwahati
Guwahati - 781 039, Assam, India

Abstract— *Cache management is one of the key factors that affect the performance of present day Chip Multi-Processors. The main aspects that govern the cache management are the access latency and the cache space utilization. This paper proposes 3-chance clustered caching cache management scheme in NoC based multi-core systems, where it targets to address both the issues. The L2 banks are formed into a cluster and are non inclusive. The cache management policy concentrates on increasing the life time of a cache block by giving up to 3 chances by rotation of data among the L2 banks and clustering keeps the data close to the processors thereby decreasing the access latency. The caches act as non-inclusive for increasing the cache space, which increases the access latency but is reduced by 2 level directory protocol implemented for cache coherence and cache clustering. The evicted L1 cache bocks are stored in the Cluster home L2 bank and the evicted L2 cache blocks follow the 3-Chance rotation algorithm. Experimental results based on full-system simulation show that for 16 core 2D-MESH the scheme increases the performance by* $9 \sim 15\%$.

Keywords: Cache coherence; Clustering; Cached block lifetime; Performance; Victim rotation

1. Introduction

The number of processors residing in the CMPs is increasing day by day. Instead of complex processors more number of simple processors are placed on the chip. IBM introduced Power 6 [1] processor with dual high performance cores and Niagara 2 [2] by Sun Microsystems with 8 SPARC cores each supporting 8 hardware threads all on a single chip. This increase in number of processors arises new challenges for computer architects mainly the cache management and the interconnection network for communication. Network-on-Chip is an effective solution for interconnection. This paper addresses the cache management in NoC based multi-core systems.

Many cache management schemes have been proposed and the basic ideas are the Private and Shared Schemes. In a CMP the chip consists of tiles where in each tile there resides a processor with L1 cache and an L2 bank along with the Directory for cache coherence and off-chip memory accesses. In private scheme the L2 cache is private and can be accesses by its own processor, whereas in shared scheme the L2 banks are distributed. In both the cases the L2 banks are inclusive. The factors which affect the performance of the CMPs are the memory access latency and the available cache space. The private scheme even though reduces the access latency, it suffers in the cache space utilization. The number of copies of a shared data can be more thereby reducing the effective cache space. The shared scheme provides high cache space utilization by distributed L2 banks as only one copy of shared data is present, but suffers in access latency as the L2 bank which contains the data can be far away. The shared scheme is also known as Non-Uniform Cache Architecture (NUCA) [3], [4].

The cache management scheme proposed in this paper tries to balance these two factors and mainly concentrates on increasing the life time of a data block on the chip and tries to reduce the access latency by clustering and keeping the data inside the cluster. For cluster of size (N+1) we can have a N-chance scheme. In this paper we experiment on clusters of size 4 and hence we have a 3-chance scheme. The 3-Chance Clustered Caching scheme can be divided as follows

- Forming of Clusters
- Storing of L1 evicted cache blocks in Cluster Home L2
- Rotation of L2 evicted cache blocks
- 2-Level Directory Protocol for Cache Coherence

The rest of the paper is organized as follows. The following section explains the previous proposed designs and identified draw backs. Section 3 introduces the 3-Chance Clustered Caching scheme and Section 4 experimental results and Section 5 future work and the last section deals with the conclusion.

2. Related Work

Many cache management policies have been proposed in the recent years trying to balance the factors (latency and cache capacity). In [5] they divide 16 tiles into four groups where each group contains four tiles with four L2 banks. The four L2 banks are in shared scheme and the distributed directories control the cache coherence through

query messages. By this the access latency is reduced as the L2 is nearby in the cluster. When the data block is already present in some L1 cache within the cluster then a new L1 miss request for that block can be redirected to the L1 cache in the cluster having the data closer to the requestor than wasting the L2 cache space. Non inclusive L2 caches are better than inclusive in a cluster as the access latency would be the same and effective L2 capacity is more in non inclusive. Detailed explanation will be given in next section.

In [6], [7] and [8] the evicted L1 cache blocks called victims are removed and are stored in the L2 bank of the same tile and these blocks are called replicas. The hybrid scheme which combines the private and shared scheme lacks in localization i.e. the replica created can only be accessed by its own processor but not by nearby cores and these cores have to again get it from the global L2 bank(which may be far away). This can be avoided by clustering and localizing, the evicted L1 blocks are sent to the L2 banks in the cluster which can be accessed by all the processors in the cluster.

In [9] the L2 banks are private to start with but become shared due to spilling of data. The evicted L2 cache blocks are spilled onto neighbor L2 bank thereby increasing the life time of the cache block but this could not be scaled as the number of processors increase and the cache coherence engine is an overhead. This can be removed through clustering where in each cluster the spilling of data is done for evicted L2 cache blocks to increase their life time. For the cache clustering to work the directory has to be 2-level one for intra cluster and other inter cluster queries.

In [10] [11] the processors are divided into clusters where each cluster has a L2 cache bank and directory which maintains the information about that cluster and another sits near the shared main memory which maintains information about all the clusters. A variant of this scheme is used in our scheme where a directory sits on each tile. It is elaborately discussed in the next section. Such a scheme reduces the access latency as the clusters information is within the cluster at cluster home directory.

In [12] where each processor has its own cache dimension and depending on it the number of L2 banks it can access vary dynamically. A mapping function is used at each L1 for sending query messages to L2 cache depending on the cache dimension of that core the L1 belongs and if none of the L2 has it the directory is contacted as other processors might have already brought the data. The number of shared data copies on the chip will increase thereby decreasing the overall cache capacity, and if a neighbor L1 has the data, requestor has to wait for the directory to respond which adds up a lot to the access latency. The next section explains the 3-Chance Clustered Caching scheme and how it overcomes the drawbacks of the previous designs as discussed earlier.

3. 3-Chance Clustered Caching

3.1 Non Inclusive Formation of Clusters

Before going into the Cluster formation the architecture of the CMP is explained. Fig: 1 shows the architecture of the [13] 2D-MESH 16-tile CMP model. Tiled CMP architectures can scale well as the number of processors increase. The architecture employs a 2D mesh switched network, and each tile is connected to the 2D mesh network via the router attached to each tile. Fig: 1(b) shows the inside of a tile which contains a processor, a dedicated L1 cache, a L2 bank and a Directory which serves its purpose for coherence between the caches and for off chip memory accesses. The cache controllers of L1, L2 and Directory communicate via the NoC fabric through messages for data transfer as well as for coherence requests. The MESH network employs XY routing algorithm [14].

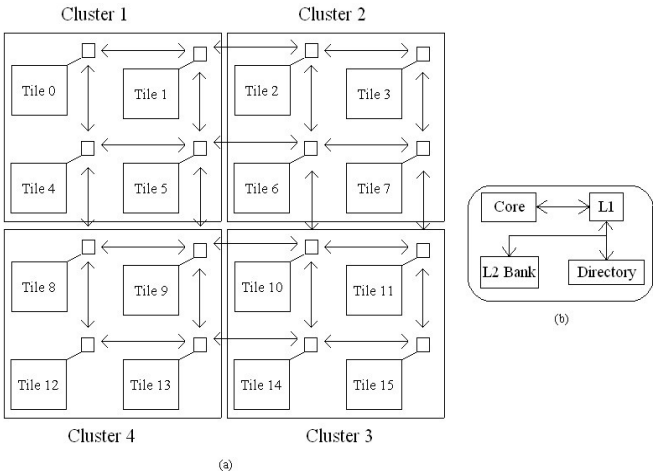


Fig. 1: (a) Clusters formed in a 16-core Tiled CMP (b) Architecture inside a tile

The 16 tiles are divided into four clusters and each cluster containing four cores. In a cluster the L2 act as non inclusive caches, where either one L2 has a data or any L1 has the data. The advantage of this policy is increase in the effective cache capacity of a cluster than [5] but comes with a disadvantage of access latency for example some L1 in the cluster has already cached the block and if an L1 requests for the same block none of the L2 banks will have the cache block because of the non inclusiveness. If the information regarding the L1 data is kept within the cluster at the directory then the directory will forward the request to the nearest L1 having the cache block thereby effectively reducing the overhead latency due to the non inclusiveness. As L1 hit time is much less than the L2 it further reduces the access latency caused by forwarding. The aim of the 3-Chance Clustered Caching is to increase the life time of a cache block as explained below.

3.2 L1 evicted Cache Blocks

If an L1 evicts a cache block and if it is the only copy left in the cluster instead of completely removing it from the cluster it is stored in the Cluster Home L2. This way the life time of a unique cache block is increased. The addresses are mapped across the L2 banks and the Cluster Home L2 is the L2 bank that maps to that address. Here the stored cache block in the Cluster Home L2 can be reused again by any core in the cluster i.e. the evicted L1 block is localized in the cluster and is not private to any core which is in the case of [6].

This way the number of shared copies in the cluster will always remain to the minimum thereby increasing the cache capacity yet attending to the access latency by the 2-Level Directory.

3.3 L2 evicted Cache Blocks

Every cache block in the L2 bank in the cluster is unique data block and is not present in any other cache in the cluster. So when a L2 eviction occurs the life time of the block can still be increased by forwarding the evicted cache block to neighbor L2 bank in the cluster in some order. We move the block in clockwise manner. This block can also be evicted from neighbouring L2 in future and thus rotated again until the cache block is given 3 chances, beyond that the cache block is evicted from the cluster. The 3-Chance rotation algorithm is variant of [15] where the value of N in N-chance algorithm is set to 1. After the last chance the data block is been sent to the Cluster Home Directory where it is written to the main memory if needed.

3.4 2-Level Directory Protocol

Every directory acts as a Cluster Home Directory (CHD) and Global Home Directory (GHD). The address space is mapped across the four directories in a cluster acting them as Cluster Home Directory to maintain coherence and for forwarding of data requests inside a cluster. Also the address space is mapped across the 16 directories on the chip which makes each directory Global Directory which maintains coherence across the clusters. Every entry in Directory can be a Cluster, Global or both thereby every entry in the directory has to accommodate for both the information. Fig 2 shows an entry in a directory.

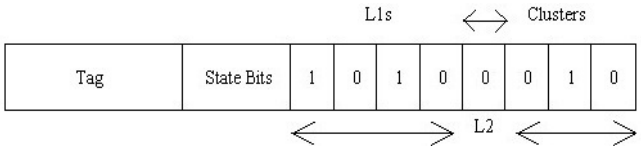


Fig. 2: An entry in the directory containing both cluster and global information of a data block.

The four L1 bits tells the information of L1s having the data in the cluster and L2 bit tells that the data is in some L2 bank in the cluster and the remaining three cluster bits is for Global information telling which other clusters have data.

When a L1 read miss occurs 5 query requests are sent, 4 to all the L2 banks in the cluster and one to the cluster home directory. If any L2 bank has data it sends the data and invalidates itself for non inclusiveness and if no L2 bank has the data it either means that any L1 has the data or there is no data in the cluster Fig 3.

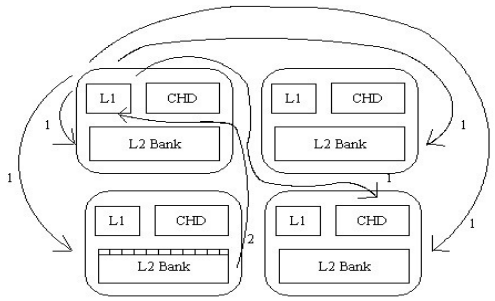


Fig. 3: Query request messages 1 are sent to the 4 L2 banks and CHD by requestor L1 for a particular address B which maps to that CHD. Response message 2 to the requestor L1 from a L2 bank having the data.

The Cluster Home Directory comes to know whether any L2 bank has the data or not. It forwards the request to the closest L1 cache near the requestor having the data if the cache block is present in any L1s of the cluster Fig 4.

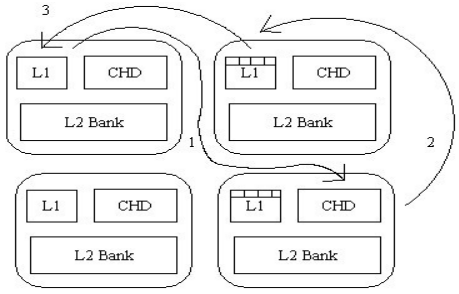


Fig. 4: If L2s dont have the data and some L1 in the cluster has, data request message 1 sent to the CHD is forwarded to the nearest sharer 2, and data response message 3 is sent by the L1 having the data.

If there is no data in the cluster the CHD forwards the request to the GHD if some other cluster has already brought the data. In that case the GHD searches the cluster bits and forwards the request to the closest CHD that contains the data and thereby the data is sent to the requestor. If there is no data present in any of the cluster the GHD makes an off-chip memory access and sends the data to the requestor updating the cluster bits Fig 5.

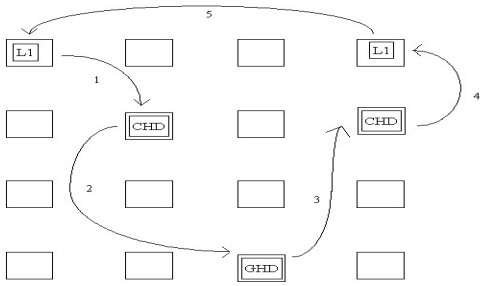


Fig. 5: If there is no data in the cluster the data request message 1 is forwarded to the GHD 2, and if some cluster has the data block the GHD forwards the request message 3 to the CHD of the cluster having the data which forwards 4 to the member of the cluster havind data and finally data response message 5 is sent to the requestor.

If an eviction comes in the Directory entry preference is given to the Global entries than the local cluster ones. Due to 3-Chance rotation extra query request messages are required for other L2 bank as the data block can be present in any one of the L2 bank. This increases the network traffic which is an overhead but it increases the performance by increasing the life time of a cache block. The experimental setup and the results of the 3-Chance Clustered Caching are shown in the next section.

4. Experimental Evaluation

4.1 Setup

In this paper we ran shared memory benchmarks on multi-core simulator GEMS [16]. GEMS uses an existing full-system functional simulation infrastructure Simics [17] as the basis to build a set of timing simulator modules for modeling the timing of memory systems and microprocessors. Table lists the configurations of the processor, cache and memory used.

Component	Parameters
Processor	UtraSPARCIII+
L1 I/D cache	64KB, 4-way, 3 cycles
L2 cache bank	2MB, 4-way, 12 cycles
Memory bank	1GB, 4KB/page, 158 cycles
distributed directory	52KB, 6 cycles

We used [18] Princetons Garnet network simulator to simulate on-chip networks which calculate the latencies of accessing L1, L2banks and directories. GEMS generate practical traffics and simulate these on Garnet. Garnets fixed pipeline model is used which models the pipeline routers and its parameters are listed in the table below.

Network Configuration	Parameters
Flit Size	16 bytes
Buffer Size	4
Pipeline Stage	5-stage
VCs per virtual network	4
Number of virtual networks	5

For evaluation of the energy consumption Orion [19] is used, it is integrated with Garnet to evaluate network energy consumption by the routers, switches etc using 100nm technology. Shared memory benchmarks from SPLASH2 suite [20] are used in our experiment and their parameters are listed below.

Benchmarks(transactions)	Environment
Ocean	514 x 514 grid
Barnes	64K particles
Water	512 molecules
Fmm	16384 particles

4.2 Evaluation

As can be seen from the graph below that shows the performance improvement of the 3-Chance Clustered Caching over Shared L2 bank scheme, the improvement is $9 \sim 15\%$. The reasons as explained earlier are that the increase in life time of the data block by non inclusive L2 banks and 3-chance rotation and reducing the access latency through clustering thereby keeping the cache block nearer to the requestor.

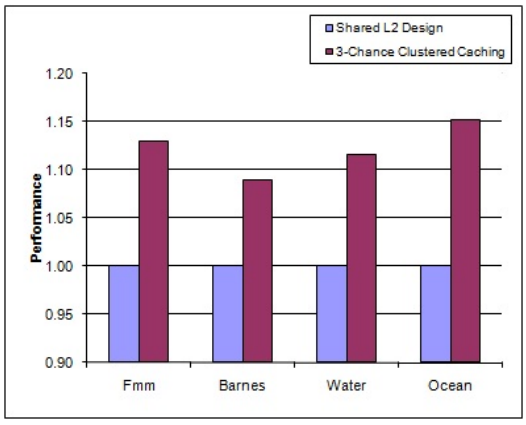


Fig. 6: Performance comparisons between Shared L2 Scheme and 3-Chance Clustred Caching for 16-node 2D-MESH CMP

The below graph shows the energy consumption and the 3-Chance Clustered Caching consumes $19 \sim 34\%$ more energy than the Shared L2 bank scheme and this can be explained due to the overhead of extra query messages sent to the L2 banks and forwarding of requests by the CHD.

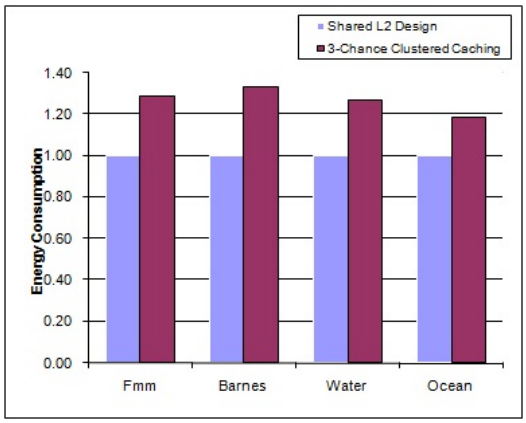


Fig. 7: Energy Consumption comparisons between Shared L2 Scheme and 3-Chance Clustred Caching for 16-node 2D-MESH CMP

5. Conclusions

For the upcoming CMPs cache management plays a key role in its performance. Its main goals are to reduce the access latency and to increase the effective cache capacity. Increase in the life time of a cached data block reduces costly off-chip memory access and non inclusiveness of L2 banks increases the cache capacity. We propose 3-Chance Clustered caching scheme which aims at the above two and ultimately reducing the access latency and increasing the cache capacity. The main principle of this scheme is to keep the data block within the cluster as long as possible. For this the evicted L1 cache blocks are stored in the cluster home L2 thereby localizing the data and making it accessible to other cores and rotating the L2 evicted data blocks using 3-chance algorithm. For this to happen a directory at cluster level and at global level is required and a 2 level directory is implemented which takes care of the query requests and the coherence issues of the caches. The experiment results show that the proposed scheme shows increase in the performance by $9 \sim 15\%$ compared with Shared L2 bank scheme.

Due to increase in the query messages because of the rotation of data there is an overhead in the energy consumption and the new scheme increases the energy consumption by $19 \sim 34\%$ Since the overheads of communications inside cache cluster play hinders the performance of the CMP our future work focuses on decreasing the query messages and also supporting n-chance rotation (intra cluster and inter cluster rotation) of L2 evicted data blocks further thereby clusters unused cache space can be utilized more efficiently.

References

[1] B. Stolt, Y. Mittlefehldt, S. Dubey, G. Mittal, M. Lee, J. Friedrich, and E. Fluhr, "Design and Implementation of the POWER6 Microprocessor," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 1, pp. 21–28, 2008.

[2] U. Nawathe, M. Hassan, L. Warriner, K. Yen, B. Upputuri, D. Greenhill, A. Kumar, and H. Park, "An 8-Core 64-Thread 64B Power-Efficient SPARC SoC," in *IEEE International Solid-State Circuits Conference. ISSCC 2007.*, 2007, pp. 108–590.

[3] J. Huh, C. Kim, H. Shafi, L. Zhang, D. Burger, and S. W. Keckler, "A NUCA Substrate for Flexible CMP Cache Sharing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 8, pp. 1028–1040, 2007.

[4] Z. Chishti, M. D. Powell, and T. N. Vijaykumar, "Distance associativity for high-performance energy-efficient non-uniform cache architectures," in *International Symposium on Microarchitecture, MICRO 2003. MICRO-36.*, 2003, pp. 55–66.

[5] W. Zuo, S. Feng, Z. Qi, J. Weixing, L. Jiaxin, D. Ning, X. Licheng, T. Yuan, and Q. Baojun, "Group-caching for NoC based multicore cache coherent systems," in *Design, Automation Test in Europe (DATE)*, 2009, pp. 755–760.

[6] M. Zhang and K. Asanovic, "Victim replication: maximizing capacity while hiding wire delay in tiled chip multiprocessors," in *International Symposium on Computer Architecture. ISCA 2005*, 2005, pp. 336–345.

[7] B. M. Beckmann and D. A. Wood, "Managing wire delay in large chip-multiprocessor caches," pp. 319–330, 2004.

[8] M. D. Powell, Z. Chishti, and T. N. Vijaykumar, "Optimizing replication, communication and capacity allocation in CMPs," in *International Symposium on Computer Architecture, ISCA*, 2005, pp. 357–368.

[9] J. Chang and G. S. Sohi, "Cooperative Caching for Chip Multiprocessors," in *International Symposium on Computer Architecture, ISCA*, 2006, pp. 264–276.

[10] Y. Zhang, Z. Lu, A. Jantsch, L. Li, and M. Gao, "Towards hierarchical cluster based cache coherence for large-scale network-on-chip," in *International Conference on Design Technology of Integrated Systems in Nanoscal Era DTIS*, 2009, pp. 119–122.

[11] M. E. Acacio, A. Ros, and J. M. Garcia, "Scalable Directory Organization for Tiled CMP Architectures," pp. 112–118, 2008.

[12] M. Hammoud, S. Cho, and R. Melhem., "Dynamic cache clustering for chip multiprocessors," in *International Conference on Super Computing ICS*, 2009, pp. 56–67.

[13] M. B. Taylor, J. Psota, A. Saraf, N. Shnidman, V. Strumpen, M. Frank, S. Amarasinghe, A. Agarwal, W. Lee, J. Miller, D. Wentzlaff, I. Bratt, B. Greenwald, H. Hoffmann, P. Johnson, and J. Kim, "Evaluation of the Raw microprocessor: an exposed-wire-delay architecture for ILP and streams," in *International Symposium on Computer Architecture ISCA 2004*, 2004, pp. 2–13.

[14] R. Mullins, A. West, and S. Moore, "Low-latency virtual-channel routers for on-chip network," in *International Symposium on Computer Architecture, ISCA*, 2004, pp. 188–197.

[15] M. D. Dahlin, R. Y. Wang, T. E. Anderson, and D. A. Patterson, "Cooperative caching: Using remote client memory to improve file system performance," EECS Dept, University of California, Berkeley, Tech. Rep. UCB/CSD-94-844, Dec 1994.

[16] M. M. K. Martin, D. J. Sorin, B. M. Beckmann, M. R. Marty, M. Xu, A. R. Alameldeen, K. E. Moore, M. D. Hill, and D. A. Wood, "Multifacet's General Execution driven Multiprocessor Simulator (GEMS) Toolset," in *Computer Architecture News (CAN)*, 2005.

[17] *Virtutech AB. Simics Full System Simulator http://www.simics.com/.*

[18] *http://www.princeton.edu/ niketa/publications/garnet-tech-report.pdf.*

[19] H.-S. Wang, X. Zhu, L.-S. Peh, and S. Malik, "Orion: a power-performance simulator for interconnection networks," in *International Symposium on Microarchitecture, (MICRO)*, 2002, pp. 294–305.

[20] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The SPLASH-2 programs: characterization and methodological considerations," in *International Symposium on Computer Architecture, ISCA*, June 1995, pp. 24–36.

SESSION

ALGORITHMS, LOGIC, CIRCUIT/HARDWARE DESIGN, AND TOOLS

Chair(s)

TBA

Synchronization of different chaotic systems and electronic circuit analysis

J.H. Park[1], T.H. Lee[1], D.H. Ji[2], H.Y. Jung[3], S.M. Lee[4]
[1]Department of Electrical Engineering, Yeungnam University, Kyongsan, Republic of Korea.
[2]Digital Media and Communications, Samsung Electronics, Suwon, Republic of Korea.
[3]Department of Information and Communication Engineering, Yeungnam University, Republic of Korea.
[4]Department of Electronic Engineering, Daegu University, Gyungsan, Republic of Korea.

Abstract— *This paper investigates the problem of synchronization between the Chen-Lee and Lorenz chaotic systems. Based on the Lyapunov stability theory and active control method, an effective controller is designed for asymptotic stability of the null solution of an error dynamics between master and slave chaotic systems. In order to verify the effectiveness of proposed control scheme, the computer simulation via Matlab software is applied to the Chen-Lee and Lorenz chaotic systems. Then, the realization model of the Chen-Lee and Lorenz chaotic systems is revised for electronic circuit simulation. Finally, the circuit simulation via NI (National Instruments) Multisim is performed to confirm the efficiency of our results.*

Keywords: Chen-Lee chaotic system, Lorenz chaotic system, Chaos synchronization, Lyapunov method, Circuit analysis.

1. Introduction

Synchronization of one system with another is very important process in the control of complex physical, chemical and biological systems as well as engineering. Therefore, many researchers have focused on this topic and developed several efficient synchronization techniques for various dynamic systems including chaotic systems, which are very sensitive to variations in the parameters and initial conditions. Since Pecora and Carroll [1] introduced the concept of synchronization in chaotic systems, the study of chaos synchronization has received increasing interest from scientists and engineers. It makes very big issue in nonlinear society. Up to date, not only various applications of chaos synchronization [2-7] but also diverse methods for control of the chaos synchronization have been introduced [8-10]. Originally, the chaos synchronization refers to the state in which the master (or drive) and the slave (or response) systems have precisely identical trajectories for time to infinity. We usually regard such a synchronization as complete synchronization or identical synchronization.

During the last several years, for more practical and real applications, the investigation of synchronization between different chaotic systems has been researched. For example, Yassen [11] studied the synchronization problem of different unified chaotic systems such as Lorenz-Chen, Lorenz-Lü and Lü-Chen. Park [12] proposed a method to synchronize between Genesio-Tesi and Rössler chaotic systems. In more development point of view, Huang [13] studied the chaos synchronization between hyperchaotic Lorenz system and hyperchaotic Lü system which has more complicated chaotic behavior and more than four Lyapunov exponent. By the way, when the mathematical model of chaotic system is implemented to electronic circuit, some adjustment due to difference of time scale between the mathematical model and electronic circuit model is sure to conduct. However, this is not easy job. Hence, that's why the numerical simulations are only provided to verify their synchronization algorithms without circuit simulation in most of literatures.

In addition, most of chaos circuit analysis dealt with only Chua's circuit and single chaos system which are a simple electronic circuit that exhibits chaotic behavior [14-17]. Sometimes, even though the circuit analysis for chaos synchronization is conducted, these researches only deal with two identical chaotic systems not different chaotic systems. For example, Cuomo et al. [18] and Lian et al. [19] presented a solution to the synchronization problem for identical Lorenz systems. They used transformed Lorenz equation because of some errors between theoretical system and practical system. In [20], Du et al. investigated the synchronization of Qi hyperchaotic master and slave systems with parameters mismatch using high order differentiator. Also, Xiao et al. [21] studied the synchronization problem between two identical Van der Pol oscillators using adaptive control method.

As is well-known, some difference between theoretical system parameters and practical system parameters exists. So, it is difficult and significant to materialize theoretical system to real one. In addition, the electrical circuit simulation of different chaotic systems have more complicated problems such as readjustment of time range or difference of limitation in power supply and electronic device and so on. Therefore, in this paper, the synchronization scheme between the revised practical Chen-Lee chaotic master system and the revised practical Lorenz chaotic slave system will be showed by applying our control law via NI Multisim. To the best of authors' knowledge, this is the first circuit analysis between different chaotic systems.

This paper is organized as follows. In Section 2, system description is given. In Section 3, the theoretical synchronization scheme between Chen-Lee and Lorenz chaotic systems is illustrated. In Section 4, a numerical simulation via Matlab is given to demonstrate the effectiveness of the proposed control method. In Section 5, the electronic circuit implementations are presented to show real applications of the method. Finally, some conclusions are given in Section 6.

2. System description

Consider the following master (drive) and slave (response) chaotic systems

$$\dot{x}(t) = f(t,x), \tag{1}$$
$$\dot{y}(t) = g(t,y) + u(t,x,y), \tag{2}$$

where $x(t) = (x_1, x_2, \ldots, x_n)^T \in \mathbb{R}^n$ and $y(t) = (y_1, y_2, \ldots, y_n)^T \in \mathbb{R}^n$ are master and slave state vectors, respectively, $f : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$ and $g : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$ are continuous nonlinear vector functions and $u(t,x,y) = (u_1, u_2, \ldots, u_n)^T \in \mathbb{R}^n$ is the control input for synchronization between master (1) and slave system (2).
As previously stated, we deal with the Chen-Lee master system and Lorenz slave system for synchronization problem.

Now let us consider following Chen-Lee master chaotic system

$$\begin{aligned}\dot{x}_1(t) &= ax_1(t) - x_2(t)x_3(t)\\ \dot{x}_2(t) &= -bx_2(t) + x_1(t)x_3(t)\\ \dot{x}_3(t) &= -cx_3(t) + \frac{1}{3}x_1(t)x_2(t),\end{aligned} \tag{3}$$

where $a = 5, b = 10, c = 3.8$.
In order to see chaotic motion of the system (3), let us take an initial condition $x(0) = (-5, -7, -10)^T$. Then, Fig. 1 shows chaotic behavior of Chen-Lee system.
Next, the Lorenz chaotic systems as slave system is given as follows

$$\begin{aligned}\dot{y}_1(t) &= a_1(y_2(t) - y_1(t)) + u_1(t)\\ \dot{y}_2(t) &= b_1y_1(t) - y_1(t)y_3(t) - y_2(t) + u_2(t)\\ \dot{y}_3(t) &= y_1(t)y_2(t) - c_1y_3(t) + u_3(t),\end{aligned} \tag{4}$$

where $a_1 = 10, b_1 = 28, c_1 = 8/3$.
The chaotic behavior of system (4) with an initial condition $y(0) = (0, -1, -1)^T$ is presented in Fig. 2.

3. Synchronization between the Chen-Lee and Lorenz systems

In this section, we design control law for achieving synchronization between the Chen-Lee and Lorenz systems.

Definition 1. It is said that synchronization occurs between master system (1) and slave system (2) such that $\lim_{t\to\infty} \|y_i(t) - x_i(t)\| = 0,\ (i = 1, 2, 3)$.

Now, for our synchronization scheme, let us define error signals between the Chen-Lee chaotic system and Lorenz chaotic system in the sense of Definition 1 as

$$\begin{aligned}e_1(t) &= y_1(t) - x_1(t)\\ e_2(t) &= y_2(t) - x_2(t)\\ e_3(t) &= y_3(t) - x_3(t).\end{aligned} \tag{5}$$

The time derivative of error signal (5) is

$$\begin{aligned}\dot{e}_1(t) &= \dot{y}_1(t) - \dot{x}_1(t)\\ \dot{e}_2(t) &= \dot{y}_2(t) - \dot{x}_2(t)\\ \dot{e}_3(t) &= \dot{y}_3(t) - \dot{x}_3(t).\end{aligned} \tag{6}$$

By substituting (3) and (4) into (6), we have the following error dynamics

$$\begin{aligned}\dot{e}_1 &= a_1y_2 - a_1y_1 + x_2x_3 - ax_1 + u_1\\ &= -a_1e_1 - (a_1 + a)x_1 + a_1y_2 + x_2x_3 + u_1\\ \dot{e}_2 &= b_1y_1 - y_1y_3 - y_2 - x_1x_3 + bx_2 + u_2\\ &= -be_2 + (b-1)y_2 + b_1y_1 - y_1y_3 - x_1x_3 + u_2\\ \dot{e}_3 &= y_1y_2 - c_1y_3 - \frac{1}{3}x_1x_2 + cx_3 + u_3\\ &= -c_1e_3 + (c - c_1)y_3 + y_1y_2 - \frac{1}{3}x_1x_2 + u_3.\end{aligned} \tag{7}$$

Here, our goal is to achieve synchronization between the Chen-Lee and Lorenz systems. For this end, the following theorem shows that chaotic systems (3) and (4) can be synchronized effectively by the following designed controller.

Theorem 1. Chaotic Chen-Lee system (3) and Lorenz system (4) can be synchronized asymptotically for any different initial conditions with the following controller:

$$\begin{aligned}u_1 &= -x_2(t)x_3(t) - a_1y_2(t) + (a_1 + a)x_1(t)\\ u_2 &= y_1(t)y_3(t) + x_1(t)x_3(t) - b_1y_1(t) - (b-1)y_2(t)\\ u_3 &= -y_1(t)y_2(t) + \frac{1}{3}x_1(t)x_2(t) - (c - c_1)y_3(t).\end{aligned} \tag{8}$$

Proof. Let us take the following Lyapunov function candidate

$$V = \frac{1}{2}(e_1^2 + e_2^2 + e_3^2). \tag{9}$$

By differentiating Eq. (9), we get

$$\dot{V} = e_1\dot{e}_1 + e_2\dot{e}_2 + e_3\dot{e}_3. \tag{10}$$

By applying our controller (8) and error dynamics (7) to Eq. (10), we obtain

$$\begin{aligned}\dot{V} &= e_1\big(-a_1e_1-(a_1+a)x_1+a_1y_2+x_2x_3+u_1\big)\\ &\quad e_2\big(-be_2+(b-1)y_2+b_1y_1-y_1y_3-x_1x_3+u_2\big)\\ &\quad e_3\big(-c_1e_3+(c-c_1)y_3+y_1y_2-\frac{1}{3}x_1x_2+u_3\big)\\ &= -a_1e_1-be_2-c_1e_3\\ &= -\begin{bmatrix} e_1\\ e_2\\ e_3\end{bmatrix}^T\begin{bmatrix}10&0&0\\0&10&0\\0&0&\frac{8}{3}\end{bmatrix}\begin{bmatrix}e_1\\e_2\\e_3\end{bmatrix}\\ &\equiv -e^TPe<0, \end{aligned} \tag{11}$$

which guarantees the stability of error systems in the sense of Lyapunov theory. This implies that the error signals satisfy $\lim_{t\to\infty}\|e_i(t)\|=0\ (i=1,2,3)$. This completes the proof. □

4. Numerical simulation

In order to demonstrate the validity of proposed ideas, numerical simulation via Matlab software is presented. Fourth-order Runge-Kutta method with sampling time 0.001[sec] is used to solve the system of differential equations (3) and (4).

The system parameters are used by $a=5, b=10, c=3.8, a_1=10, b_1=28, c_1=8/3$ in numerical simulation. The initial conditions for master and slave system are given by $x(0)=(-5,-7,-10)^T$ and $y(0)=(0,-1,-1)^T$, respectively. Fig. 3 shows that error signals go to zero asymptotically. It means synchronization occurs between state of $x_i(t)$ and state of $y_i(t), (i=1,2,3)$.

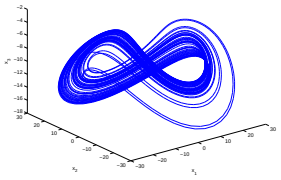


Fig. 1: Chaotic motion of Chen-Lee system

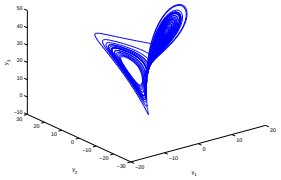


Fig. 2: Chaotic motion of Lorenz system

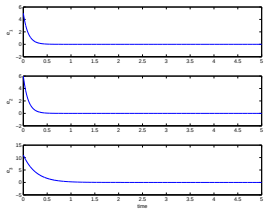


Fig. 3: Error signals of numerical example

5. Circuit design and analysis

In this section, we present circuit design and analysis for proposed synchronization scheme. As previously stated, chaotic systems have some errors between theoretical system parameters and practical system parameters. So we will conduct some process for elimination of these errors.

5.1 Chen-Lee circuit

For the circuit design of mathematical dynamic model (3), we use transformed Chen-Lee chaotic system because of some problems. Based on electronic circuit of Eq.(3), the range of state variables is over the limit of power supply. So, the reasonable transformation is to multiply 10 by nonlinear term.

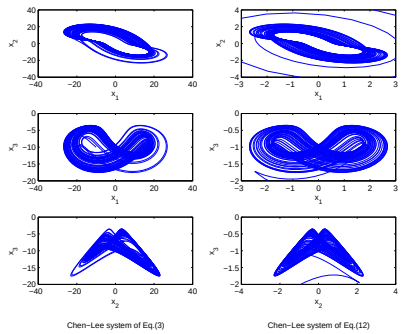


Fig. 4: Comparing with original Chen-Lee and modified Chen-Lee systems

Consider the following transformed Chen-Lee equations

$$\begin{aligned}\dot{x}_1(t) &= ax_1(t)-10x_2(t)x_3(t)\\ \dot{x}_2(t) &= -bx_2(t)+10x_1(t)x_3(t)\\ \dot{x}_3(t) &= -cx_3(t)+\frac{10}{3}x_1(t)x_2(t),\end{aligned} \tag{12}$$

where $a=5, b=10, c=3.8$.

This system can be more easily operated with analog circuit because all the state variables gave similar dynamic range and circuit voltages remain well within the range of typical power supply limits. In order to present effect of previous process, Fig. 4 is given which shows phase to phase portrait of original Chen-Lee system (3) and modified Chen-Lee system (12) of x_1-x_2, x_1-x_3, x_2-x_3 respectively. In

Fig. 4, we can note that the state value of modified Chen-Lee system (12) is similar the state value of original Chen-Lee system (3) divided by 10 but inherent chaotic behavior is not changed. It means we can use the transformed Chen-Lee system (12) for our synchronization scheme because this process keep the range of state variables less than the limit of electronic device and transformed equations behave same chaotic motions. The analog circuit of transformed Chen-Lee Eq.(12) is shown in Fig. 5.

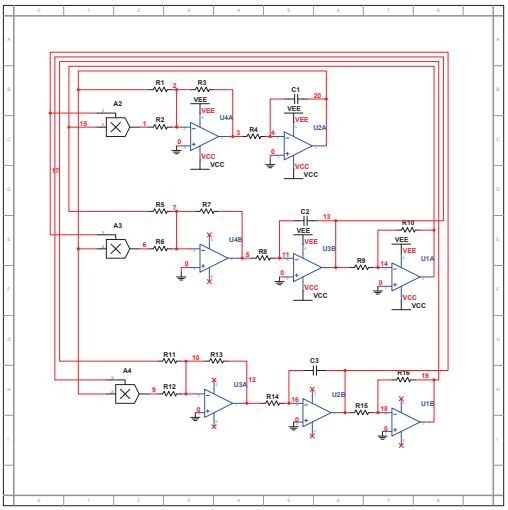


Fig. 5: The circuit of Chen-Lee system

The electrical equations of the circuit are given by

$$\begin{aligned}
\dot{x}_1 &= \frac{1}{R_4C_1}\Big(\frac{R_3}{R_1}x_1 - \frac{R_3R_{10}}{R_2R_9}x_2x_3\Big)\\
\dot{x}_2 &= \frac{1}{R_8C_2}\Big(-\frac{R_7R_{10}}{R_5R_9}x_2 + \frac{R_7}{R_6}x_1x_3\Big)\\
\dot{x}_3 &= \frac{1}{R_{14}C_3}\Big(-\frac{R_{13}R_{16}}{R_{11}R_{15}}x_3 + \frac{R_{13}}{R_{12}}x_1x_2\Big), \qquad (13)
\end{aligned}$$

where we can note that Eq.(13) is equivalent to Eq.(12) after some calculation and applying the required electrical parameters such as: $R_2, R_5, R_6 = 10k\Omega$; $R_1 = 20k\Omega$; $R_4, R_8, R_{14} = 1M\Omega$; $R_3, R_7, R_9, R_{10}, R_{11}, R_{13}, R_{15} = 100k\Omega$; $R_{12} = 30k\Omega$; $R_{16} = 380k\Omega$; $C_i = 1\mu F, (i = 1, 2, 3)$. The operational amplifiers are considered to be ideal, the time step is 0.001 [s] and the initial condition of master circuit is $x(0) = (0.02, 0.02, 0.02)$ [V]. Fig. 6 displays phase to phase portrait of master system of $x_1 - x_2$, $x_1 - x_3$, $x_2 - x_3$, respectively, in left side and time to state x_1, x_2, x_3, respectively, in right side.

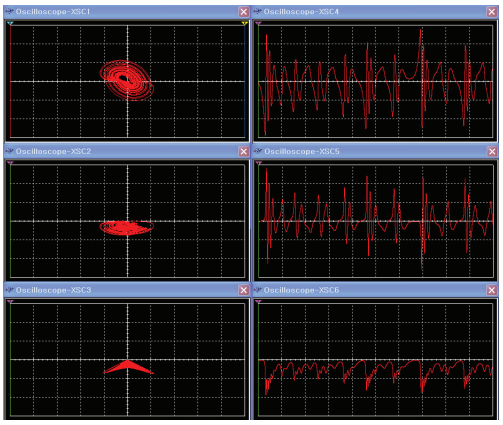


Fig. 6: Chaotic phase of Chen-Lee system

5.2 Lorenz circuit

As the same reason, we transformed Lorenz chaotic system(4) into following equations

$$\begin{aligned}
\dot{y}_1(t) &= a_1(y_2(t) - y_1(t))\\
\dot{y}_2(t) &= b_1y_1(t) - 20y_1(t)y_3(t) - y_2(t)\\
\dot{y}_3(t) &= 20y_1(t)y_2(t) - c_1y_3(t), \qquad (14)
\end{aligned}$$

where $a_1 = 10, b_1 = 28, c_1 = 8/3$.

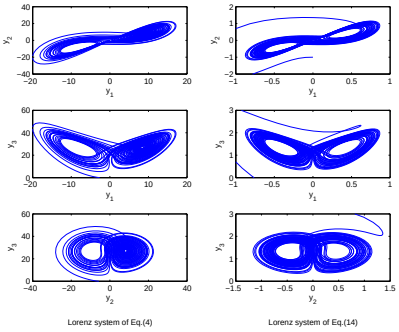


Fig. 7: Comparing with original Lorenz and modified Lorenz systems

As comparing with Eq.(4), the transformed equation is changed nonlinear terms which are multiplied by 20. Fig. 7 displays phase to phase portrait of original Lorenz system (4) and modified Lorenz system (14) of $x_1 - x_2$, $x_1 - x_3$, $x_2 - x_3$ respectively. Like the preceding, we can note that the state value of modified Lorenz system (14) is similar the state value of original Lorenz system (4) divided by 20. But we can also know inherent chaotic behavior is not changed. The analog circuit of transformed Lorenz equation (14) is shown in Fig. 8.

The electrical equations of the circuit are given by

$$\begin{aligned}
\dot{y}_1 &= \frac{1}{R_5C_1}\Big(\frac{R_4}{R_1}y_2 - \frac{R_3}{R_2+R_3}(1+\frac{R_4}{R_1})y_1\Big)\\
\dot{y}_2 &= \frac{1}{R_{13}C_2}\Big(\frac{R_{12}}{R_{11}}y_1 - \frac{R_{12}R_9}{R_{10}R_8}y_2 - \frac{R_{12}R_9}{R_{10}R_7}y_1y_3\Big)\\
\dot{y}_3 &= \frac{1}{R_{18}C_3}\Big(\frac{R_{17}}{R_{16}}y_1y_2 - \frac{R_{17}R_{14}}{R_{15}R_6}y_3\Big), \qquad (15)
\end{aligned}$$

where we can note that Eq.(15) is equivalent to Eq.(14) after rescaling time by a factor of 100. And the required electrical parameters are as following: $R_1, R_2, R_{11} = 10k\Omega$; $R_3, R_4, R_8, R_9, R_{15}, R_{17} = 100k\Omega$; $R_5, R_{13}, R_{18} = 1M\Omega$; $R_6 = 300k\Omega$; $R_7, R_{16} = 5k\Omega$; $R_{10}, R_{12} = 280k\Omega$; $R_{14} = 800k\Omega$; $C_i = 1\mu F, (i = 1, 2, 3)$. The operational amplifiers are considered to be ideal, the time step is 0.001 [s] and the initial condition of master circuit is $x(0) = (0.01, 0.01, 0.01)$ [V]. Fig. 9 displays phase to phase of master system of $y_1 - y_2$, $y_1 - y_3$, $y_2 - y_3$, respectively, in left side and time to state y_1, y_2, y_3, respectively, in right side.

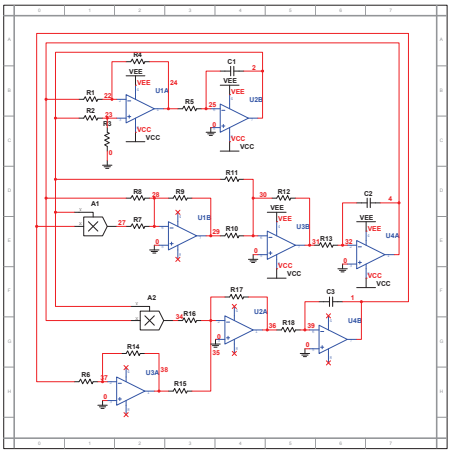


Fig. 8: The circuit of Lorenz system

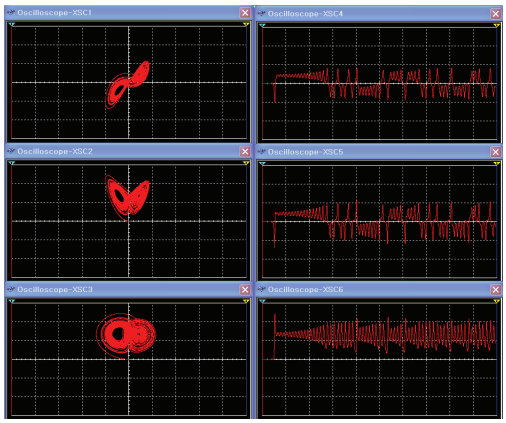


Fig. 9: Chaotic phase of Lorenz system

5.3 Synchronization circuit

As transforming Eqs. (3) and (4) to Eqs.(12) and (14), respectively, the control inputs of Theorem 1. should be also changed as follows:

$$\begin{aligned}
u_1 &= -10x_2x_3 - a_1y_2 + (a_1+a)x_1\\
u_2 &= 20y_1y_3 + 10x_1x_3 - b_1y_1 - (b-1)y_2\\
u_3 &= -20y_1y_2 + \frac{10}{3}x_1x_2 - (c-c_1)y_3. \qquad (16)
\end{aligned}$$

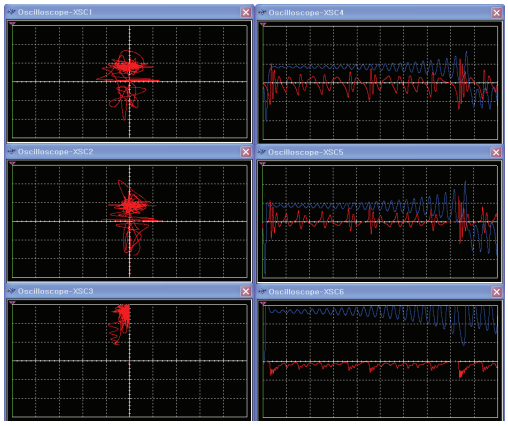


Fig. 10: Simulation results without control

To show the effect of control input, first of all, we run the circuit without control inputs. Fig. 10 displays phase to phase and time to phase portraits of master and slave systems for this case. One can see that the errors do not approach to zero as expected since the control inputs are not applied.

Finally, the circuit of the whole synchronizing system is given in Fig. 11. The circuit consists of three parts: master systems, slave systems, and controllers. Then, Fig. 12 displays that synchronization between Chen-Lee chaotic system and Lorenz chaotic system is achieved by control inputs as expected.

6. Conclusion

In this paper, we have investigated the synchronization problem for the Chen-Lee and Lorenz chaotic systems. Our proposed control scheme is verified by numerical simulation of the system. It should be noted that we included circuit analysis for the synchronization between the different chaotic systems for the first time.

Acknowledgements

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2010-0009373).

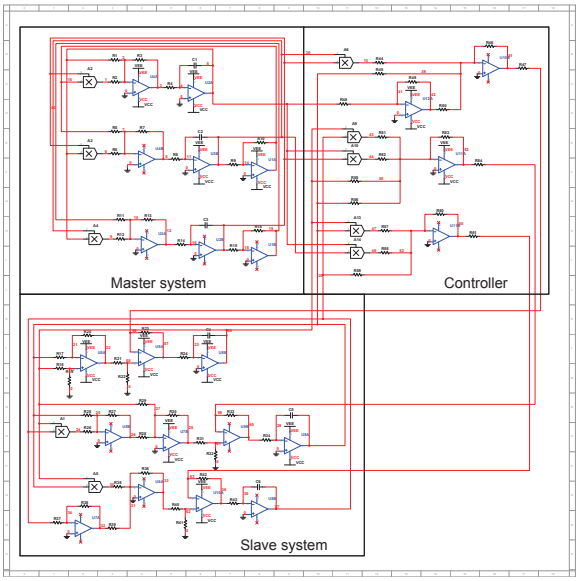


Fig. 11: The circuit for controlled systems

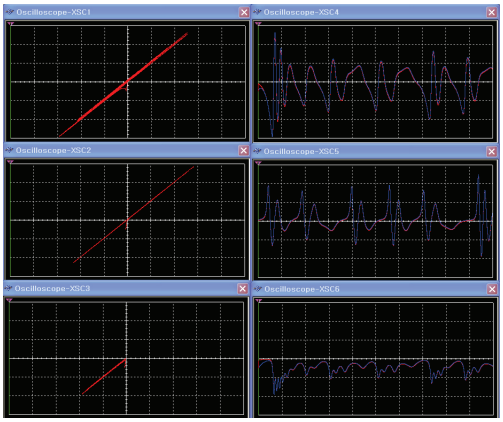


Fig. 12: Simulation results with control

References

[1] L.M. Pecora, and T.L. Carroll, Synchronization in chaotic systems. *Physical Review Letters* **64**, 821-824, 1990.

[2] U.E. Vincent, Synchronization of Rikitake chaotic attractor using active control. *Physics Letters A* **343**, 133-138, 2005.

[3] S. Oancea *et al.*, Master-slave synchronization of Lorenz systems using a single controller. *Chaos Solitons Fractals* **41**, 2575-2580, 2009..

[4] J.H. Park, Adaptive controller design for modified projective synchronization of Genesio-Tesi chaotic system with uncertain parameters. *Chaos Solitons Fractals* **34**, 1154-1159, 2007.

[5] T.H. Lee, and J.H. Park, Adaptive Functional Projective Lag Synchronization of a Hyperchaotic Rössler System. *Chinese Physics Letters* **26**, 090507, 2009.

[6] Yassen, M.T. Adaptive chaos control and synchronization for uncertain new chaotic dynamical system. *Physics Letters A* **350**, 36-43, 2006.

[7] J. Lü, X. Yu, and G. Chen, Chaos synchronization of general complex dynamical networks. *Physica A* **334**, 281-302, 2004.

[8] X.Q. Wu, and J.A. Lu, Parameter identification and backstepping control of uncertain Lu system. *Chaos Solitons Fractals* **18**, 721-729, 2003.

[9] J.H. Park, Chaos synchronization of a chaotic system via nonlinear control. *Chaos Solitons Fractals* **25**, 579-584, 2005.

[10] J.H. Park, and O.M. Kwon, LMI optimization approach to stabilization of time-delay chaotic systems. *Chaos Solitons Fractals* **23**, 445-450, 2005.

[11] M.Y. Yassen, Chaos synchronization between two different chaotic systems using active control. *Chaos Solitons Fractals* **23**, 131-140, 2005.

[12] J.H. Park, Chaos synchronization of between two different chaotic dynamical systems. *Chaos Solitons Fractals* **27**, 549-554, 2006.

[13] J. Huang, Chaos synchronization between two novel different hyperchaotic systems with unknown parameters. *Nonlinear Analysis* **69**, 4174-4181, 2008.

[14] H.R. Pourshaghaghi *et al.* Reconfigurable logic blocks based on chaotic Chua circuit. *Chaos Solitons Fractals* **41**, 233-244, 2009.

[15] G. Qi *et al.*, A new hyperchaotic system and its circuit implementation. *Chaos Solitons Fractals* **40**, 2544-2549, 2009.

[16] G.G. Dong *et al.*, Spectrum Analysis and Circuit Implementation of a New 3D Chaotic System with Novel Chaotic Attractors. *Chinese Physics Letters* **27**, 020507, 2010.

[17] L.J. Sheu *et al.*, Alternative implementation of the chaotic Chen-Lee system. *Chaos Solitons Fractals* **41**, 1923-1929, 2009.

[18] K.M. Cuomo, A.V. Oppenheim, and S.H. Strogatz, Synchronization of Lorenz based chaotic circuits with applications to communications. IEEE Transactions Circuit Systems II **40**, 626-633, 1993.

[19] Lian, K.Y. and Liu,P. Synchronization with message embedded for generalized Lorenz chaotic circuits and its error analysis. *IEEE Transactions Circuit Systems I* **47**, 1418-1430, 2000.

[20] S. Du *et al.*, Chaotic system synchronization with an unknown master model using a hybrid HOD active control approach. *Chaos, Solitons Fractals* **42**, 1900-1913, 2009.

[21] M. Xiao and J. Cao, Synchronization of a chaotic electronic circuit system with cubic term via adaptive feedback control. *Communications Nonlinear Science Numerical Simulation* **14**, 3379-3388, 2009.

Improved Cost Reversible Multiplier Design

Mihail Cutitaru, Lee A. Belfore, II
Department of Electrical and Computer Engineering, Old Dominion University, Norfolk, VA 23529, USA

Abstract— *Reversible logic is an emerging research area and is a promising technology for the next generation of computers. The focus of this paper is improving the design of a multiplier that can be part of any future reversible computer. In this paper, we propose an improvement to the manner in which partial products are generated that improves the efficiency of implementation using repetition codes. The final implementation reduces the total number of gates, constant inputs, garbage outputs, and quantum cost over other proposed models.*

***Index Terms*— Reversible logic, multiplier, quantum cost.**

1. Introduction

Modern logic circuits offer a great deal of computing power in a small footprint. As technology evolves and many more transistors can fit in a given area, the concern for power dissipation as heat arises. Landauer has shown in [1] that for every erased bit at least $kT \log_2$ J of energy are lost, where k is Boltzmann's constant, T is the circuit temperature in degrees Kelvin. Also, according to Moore's law, the number of transistor elements doubles roughly every two years and if this trend continues to hold, in the near future more and more energy will be lost due to bit erasures and cooling than doing computations. Bennett has shown in [2] that this problem could be avoided if reversible logic is used. In reversible logic computations, there are no bit erasures and energy is conserved, thus no heat is dissipated. In order to be able to have reversible logic computations a new set of gates are required. Reversible gates are gates that have a one-to-one and onto relationship between inputs and outputs, that is they are bijective.

A relatively large number of reversible gates and circuits using these gates have been proposed. Some other considerations include reducing the number of constant input lines (CI), number of quantum gates, and quantum cost (QC). Quantum cost is defined as the number of quantum gate primitives (1×1 or 2×2 gates) required to construct a given circuit. For example, the quantum cost of the CNOT gate is 1, Toffoli gate is 5, etc.

Multiplier circuits are of particular interest because they are used in a wide variety of computing applications. Several reversible multiplier designs have been proposed aimed at decreasing the costs explained above and this paper aims at decreasing those costs even further.

The rest of the paper is organized as follows: Section 2 gives a brief introduction to the reversible gates used in this paper and previous reversible multiplier designs. Section 3 describes the proposed design in detail. Section 4 shows the results of circuit simulation and compares them with other implementations, and Section 5 concludes the paper.

2. Background

Several reversible logic gates have been proposed, ranging from a simple inverter (NOT gate) to 4×4 full adders. Some of the gates were specifically designed for certain tasks (adders, etc) and other were designed as general-purpose gates (Peres, Toffoli, etc).

A few reversible multiplier designs were proposed in [3], [6]–[9], but their total costs do not appear to be minimal. In [9] the authors have introduced a new gate (TSG gate) and constructed a reversible multiplier using it. In [6]–[8], multipliers were proposed using new gates (i.e. MKG, HNG, and PFAG respectively) with different total costs. The proposed gates have different quantum costs in their implementations, so some of the multipliers were more efficient than others. All the gates above are 4×4 gates that can act as full adders and are a few of the many possible variations to achieve the given functionality on a 4×4 gate. Lastly, in [3], another design was proposed that used a Double-Peres Gate (DPG), which was also based on the 4×4 full adder gates. This design achieved lower costs than previous designs, but it still was not minimal. All four multipliers above had a fanout for both the multiplier and multiplicand, which increased total costs significantly.

The proposed design aims at decreasing the costs of the multiplier by tackling the Partial Products (PP) generation stage and using fanout on only one of the multiplicands. The gates chosen for the proposed design were Controlled-NOT (CNOT) and Peres gates. Below is a short description of these gates.

a) Controlled-NOT Gate (CNOT): CNOT gate is a 2×2 gate that passes-through the first input and produces the XOR of the two inputs on the second output [4]. The CNOT gate has a Quantum cost of one.

b) Peres Gate: The Peres gate is a 3×3 gate with quantum cost of four. The organization and logic equations defining the Peres gate are given in Fig. 2 [5].

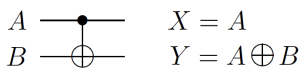


Fig. 1
CNOT GATE

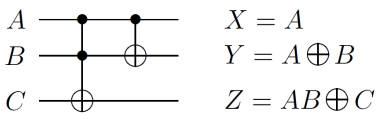


Fig. 2
PERES GATE

3. Proposed Design

The proposed multiplier is designed with Partial Products (PP) circuit to generate partial product and a Parallel Adder (PA) circuit that adds the partial products to form the product. The operation of the multiplier is shown in Fig. 3.

				y_3	y_2	y_1	y_0
				x_3	x_2	x_1	x_0
				P_{30}	P_{20}	P_{10}	P_{00}
			P_{31}	P_{21}	P_{11}	P_{01}	
		P_{32}	P_{22}	P_{12}	P_{02}		
	P_{33}	P_{23}	P_{13}	P_{03}			
R_7	R_6	R_5	R_4	R_3	R_2	R_1	R_0

Fig. 3
MULTIPLIER OPERATION

The first operation is generating the PP's from the given four-bit inputs. To generate all PP's, a repetition code is applied to each bit in the multiplier using CNOT gates as shown in Fig. 4. With this approach it is not necessary to apply the repetition code to the multiplicand bits. The bits from the multiplier do not have to pass through a repetition code as the design allows them to be passed through unchanged.

Once four copies of each bit of the multiplicand are obtained, PP's are obtained by passing two input bits (one from the multiplier, the other form the multiplicand) through an AND gate. The Peres gate with a constant zero as its third input and two input bits provides the AND function. Only four copies of the multiplicand bits are needed because the proposed design makes use of the propagation of the first input of the Peres gate as an output. This construct results in fewer gates, decreased number of constants used, and a lower quantum cost compared to other designs. The circuit diagram for generating the first four PP's is given in Fig. 5. The rest of the PP's can be generated in a similar fashion by adding new constant input lines.

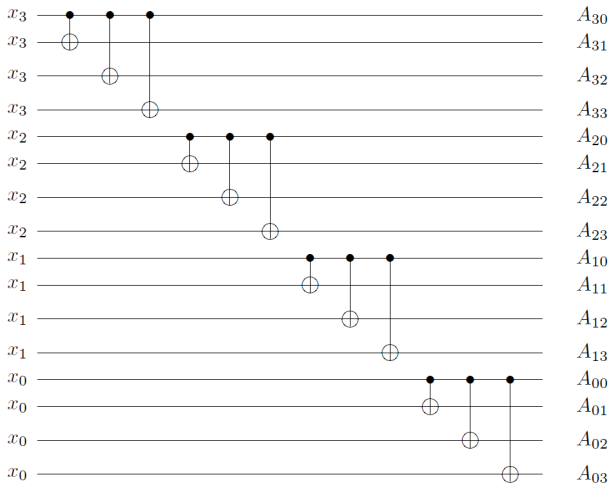


Fig. 4
REPETITION CODES FOR THE MULTIPLIER QUBITS

The next part is the generation of the Parallel Adder (PA), which, given a collection of binary inputs, outputs the number of ones in the inputs. A PA can be constructed in many ways, but the proposed design uses a cascade of Full-Adders and Half-Adders (for example [12]). For this four-bit by four-bit multiplier, the minimum number of full-adders and half-adders are eight and four, respectively. A single Peres gate can act as a half-adder given a constant zero as its last input. The full-adder can be implemented with two Peres gates linked together and re-arraged and a constant input, as shown in Fig. 6.

The full adder above has a quantum cost of six, which is the minimum found in the literature. The full adder can also be implemented with any other gate(s) that can be used as a full adder given a constant input. The PA diagram along with all connections for the final multiplier is given in Fig. 7. In addition, Fig. 8 shows how the Peres and Double Peres Gates can be substituted into the PA circuit.

4. Results and Comparison

In previous designs in [6]–[9], several gates were used to produce the fan-out. In [3], three BVF gates were used to produce four copies of two input bits. This was fewest number of gates and quantum cost noted in literature, however it was not minimal as the multiplier and multiplicand were copied four times. This is not necessary in the proposed design because it uses the pass-through capability of the Peres gate to mirror the first input to the output. Using the mirroring function of the Peres gate, the number of CNOT gates used is decreased by 12 (or 50%). This also helps decrease the number of constant inputs to 12 for the fan-out and the number of garbage outputs to 20. The number of Peres gates used to generate the PP's stays constant at

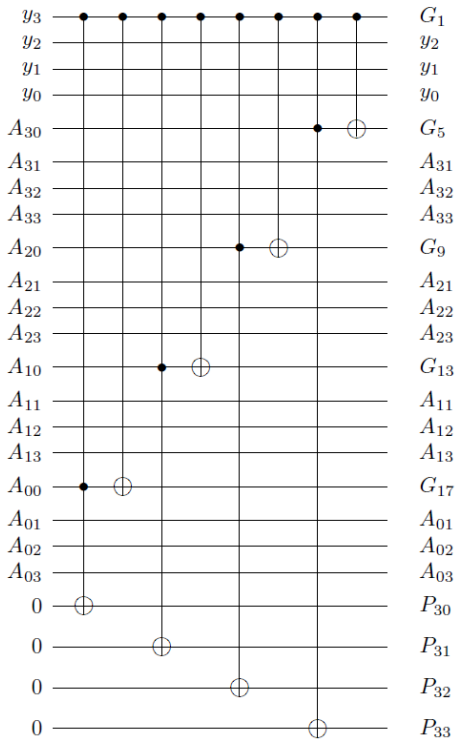


Fig. 5

GENERATION OF THE FIRST FOUR PARTIAL PRODUCTS (*PP*'S) USING PERES GATES

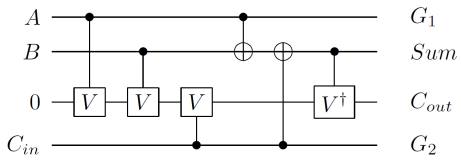


Fig. 6

FULL ADDER USING TWO PERES GATES

16 gates. Overall, the number of gates used decreases to 28 (compared to 40 with other designs) and quantum cost decreases to 76. A comparison with [10] also shows that the quantum cost of the proposed PP circuit is lower. This is because Peres gates used in this circuit have a lower cost compared to the Toffoli gates used in [10], and even though using Peres gates involves using repetition code for half of the inputs, the final quantum cost is still lower. Table 1 shows a wider comparison between several proposed PP implementations. Here, the gates cost gives the number of quantum gates needed to implement a given design and is technology-independent. At best, a given technology will be able to achieve the shown gates count, otherwise the numbers will vary depending on the technology used.

Parallel adders implementation did not improve the number of gates or the quantum cost compared to [3], [6] because it uses the minimum number of gates for implementation and has lowest quantum cost. However, due to the improvement in the PP generation, the total quantum cost is still lower than any other circuit studied. A comparison between the total costs for the reversible multiplier is given in Table 2.

PP Generation	Gates	CI	GO	QC
Proposed	28	28	20	76
DPG [3]	28	40	32	88
MKG [6], HNG [7], PFAG [8]	40	40	32	88
TSG [9]	40	40	32	104

Table 1

PARTIAL PRODUCT COSTS

PP Generation	Gates	CI	GO	QC
Proposed	40	40	40	140
DPG [3]	40	52	52	152
MKG [6]	52	52	52	152
HNG [7]	52	56	56	208
PFAG [8]	52	52	52	168
TSG [9]	53	58	58	221

Table 2

MULTIPLIER TOTAL COSTS

5. Conclusions/Future Work

In this paper a new approach to implementing the repetition codes for generating partial products in a reversible multiplier is proposed. Using CNOT and Peres gates, the number of gates, constant inputs, garbage outputs, and quantum cost are reduced compared with existing circuits. This design can be used to construct larger multiplier circuits ($N \times N$ bit) and in more complex reversible systems. Future work includes further attempts at minimizing total multiplier costs as a whole, not just for each component.

References

[1] R. Landauer, "Irreversibility and heat generation in the computing process," in *IBM Journal of Research and Development*, Vol. 5, pp. 183-191, July 1961.

[2] C. Bennett, "Logical Reversibility of Computation," in *IBM Journal of Research and Development*, Vol. 17, No. 6, pp. 525-532, Nov. 1973.

[3] H. R. Bhagyalakshmi, M. K. Venkatesh, "An Improved Design of a Multiplier using Reversible Logic Gates," in *International Journal of Engineering Science and Technology*, Vol. 2, No. 8, 2010.

[4] R. Feyman, "Quantum Mechanical Computers," in *Optics News*, Vol. 11, pp. 11-20, 1985.

[5] A. Peres, "Reversible Logic and Quantum Computers," in *Physical Review A*, 32:3266-3276, Dec. 1985.

[6] M. Shams et al., "Novel Reversible Multiplier Circuits in Nanotechnology," in *World Applied Science Journal*, Vol. 3, No. 5, pp. 806-810, 2008.

[7] M. Haghparast et al., "Design of a Novel Reversible Multiplier Circuit using HNG Gate in Nanotechnology," in *World Applied Science Journal*, Vol. 3, No. 6, pp. 974-978, 2008.

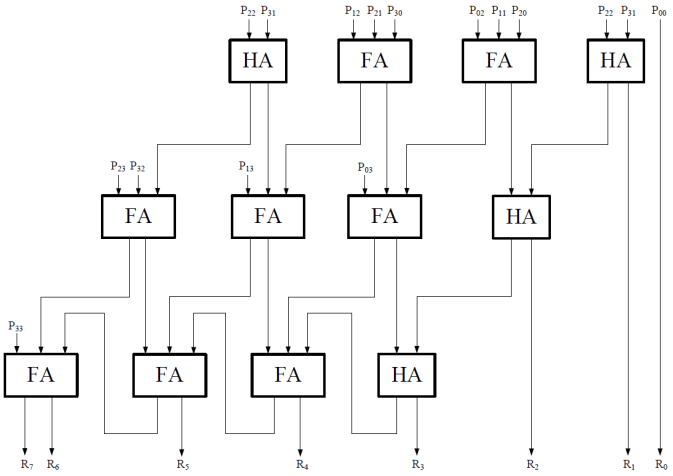


Fig. 7

PARALLEL ADDERS WITH INTERCONNECTIONS (*PA*'S) [3] AND E.G. [12]

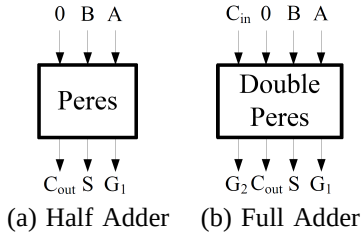


Fig. 8

PERES GATE CONFIGURATIONS REQUIRED IN FIG. 7

[8] M. S. Islam et al., "Low Cost Quantum Realization of Reversible Multiplier Circuit," in *Information Technology Journal*, Vol. 8, No. 2, pp. 208-213, 2009.

[9] H. Thapliyal, M. B. Srinivas, "Novel Reversible Multiplier Architecture using Reversible TSG Gate," in *Proceedings of the IEEE International Conference on Computer Systems and Applications*, pp. 100-103, Mar. 2006.

[10] A. Banerjee, A. Pathak, "An Analysis of Reversible Multiplier Circuits," *arXiv:0907.3357*, 2009.

[11] E. Fredkin, T. Toffoli, "Conservative Logic," in *International Journal of Theoretical Physics*, Vol. 21, pp. 219-253, 1982.

[12] M. J. Flynn, S. F. Oberman, *Advanced Computer Arithmetic Design*, Wiley, New York, 2001.

Advancement On Optical Transition Logic

A. Mr. Parteek Sharma[1], **B. Er. Gagandeep Sharma**[2], **and C. Ms. Richa Sharma**[3]
[1]Computer Science Engineering, Punjab Technical University, Jullundur, Punjab, India
[2]Computer Science Engineering, RIMT-IET, Mandi Gobindgarh, Punjab, India
[3]Master of Computer Application, CIET, Jalvehra, Punjab, India

Abstract – *This research paper is about advancement in Optical Computing an emerging field of computer design and hardware with very fast speed and performances. The optical Computing (also known as Photonic computing) is a technique based on photons of visible light or infrared region rather than Electrons flowing in electric current which are used to perform digital Computations using electronic logic gates. In this technique we are using logic gates which will show the logic transition using photons of visible light which forms the basis of our research. We studied here that we can achieve a logic gate transition through photons of light employed using chemical compounds which behalves accordingly to incident photons of light on it. This photonic logic will be used to make optical transistors. This will in turn used to make processors working on the principal of light rather than on the principal of electric current.*
Keywords:-Optical Computing, Photonic Logic, Photo luminescence, optical transistor.

1 Introduction

An optical computing (also known as photonic computing) is a technique based on photons of visible light or infrared beams. This uses photons of light rather than electrons in electric current to perform digital computations. Electric current creates heat in computer systems and as processing increases the energy requirement also increases so does the heat the extra heat is extra damaging to the hardware. Light however creates insignificant amount of heat regardless of how much it is used. Thus due to need of more powerful processing systems alternate way has to be found and its scope lies in optical computing. So by applying science of photonics someday in future a computer able to perform operations significantly faster than the convention electronic computer can be achieved. Another advantage could be coherent light beams, unlike metal conductor, pass through each other without interfering (at least not after the intersection). Electrons repel each other while photons do not. This is why the signal from copper wire gets poorer the further you are from the telephone exchange while Fiber optic cables do not have this problem. Several laser beams can be shone so their paths intersect, but there is no interference among the beams, even when they are confined essentially to two dimensions. Electric currents must be guided around each other, and this makes three-dimensional wiring necessary. Thus, an optical computer, besides being much faster than an electronic one, might also be smaller. Most research projects focus on replacing current computer components with optical equivalents, resulting in an optical digital computer system processing binary data. This approach appears to offer the best short-term prospects for commercial optical computing, since optical components could be integrated into traditional computers to produce an optical/electronic hybrid.

However, optoelectronic devices lose c.30% of their energy converting electrons into photons and back, this also slows down transmission of messages. All-optical computers eliminate the need for switching. [1]

2. Optical computing

2.1 Detail review

Optical computing is a computing technology in the research and theory stage. The idea would be to make a computer that relies entirely on light (photons) instead of electricity (electrons) to do computing. The appeal of optical computers is limited, because with short distances, they require more power than electronic computers to do the same computation. Still, optical computing may allow the construction of computers physically impossible using electronics. Optical computing is still in the early stages of development -- only a few very limited prototypes have currently been constructed in the lab.

An optical computer primarily uses lasers to send signals. Unfortunately, lasers can't interact directly with one another in any meaningful way, so performing computations requires an intermediary in the form of matter somehow. Attempts to make "optical transistors" have tended to revolve around materials that re-emit light selectively in response to the intensity of the incoming light. Putting together these components into a huge web can allow the construction of an optical computer.

2.2 Technology

So far, optics has been enthusiastically adopted for data transmission over long distances, as in fiber optics. Over short distances, however -- and this is one of the main downsides of optical computing -- the energy loss experienced by the light requires more power to send a signal than using electrons to send the same signal over the same distance. Over long distances, light wins out, but part of the point of computers is that they're supposed to be small, and the distances over which light is better (10 Ft. /3 m or more) are pretty big by the standards of computing. Still, it is conceivable that optic channels could be used in large supercomputers to send data more efficiently than electronics.

In theory, optical computing could produce computers tens of thousands of times faster than today's computers, because light can travel that much faster than electric current. In practice, however, the need to use large beams of light to avoid signal loss has precluded that possibility. More recently, however, researchers at Harvard University found a way to flip a register using only a single photon, a milestone which could open the path to efficient optical computing. The researchers took advantage of Plasmon's, tiny surface disturbances in a medium which can be created by bombarding it with photon

2.2 Components

The fundamental building block of modern electronic computers is the transistor. To replace electronic components with optical ones, an equivalent "optical transistor" is required. This is achieved using materials with a non-linear refractive index. In particular, materials exist [3] where the intensity of incoming light affects the intensity of the light transmitted through the material in a similar manner to the voltage response of an electronic transistor. This "optical transistor" effect is used to create logic gates, which in turn are assembled into the higher level components of the computer's CPU. These will be non-linear crystals used to manipulate light beams into controlling others. There has been a large expansion in the optical interconnects for photonic based computing. Currently these interconnects are being tested and expanded by Intel.

The fundamental building block of modern electronic computers is the transistor. To replace electronic components with optical ones, an equivalent "optical transistor" is required. This is achieved using materials with a non-linear refractive index. In particular, materials exist [3] where the intensity of incoming light affects the intensity of the light transmitted through the material in a similar manner to the voltage response of an electronic transistor. This "optical transistor" effect is used to create logic gates, which in turn are assembled into the higher level components of the computer's CPU. These will be non-linear crystals used to manipulate light beams into controlling others. There has been a large expansion in the optical interconnects for photonic based computing. Currently these interconnects are being tested and expanded by Intel.

2.3 Challenges misconceptions and prospect

A claimed advantage of optics is that it can reduce power consumption, but an optical communication system will typically use more power over short distances than an electronic one. This is because the shot noise of an optical communication channel is greater than the thermal noise of an electrical channel which, from information theory, means that more signal power is required to achieve the same data capacity. However, over longer distances and at greater data rates, the loss in electrical lines is sufficiently large that optical communications will comparatively use a lower amount of power. As communication data rates rise, this distance becomes longer and so the prospect of using optics in computing systems becomes more practical. A significant challenge to optical computing is that computation is a nonlinear process in which multiple signals must interact to compute the answer. Light, which is an electromagnetic wave, can only interact with another electromagnetic wave in the presence of electrons in a material, and the strength of this interaction is much weaker for electromagnetic wave light than for the electronic signals in a conventional computer. This result in the processing elements for an optical computer requiring more power and larger dimensions than those for a conventional electronic computer using transistor until recently, electronics was fine for computer processing, but at speeds higher than 40 GHz, only optics can cope.[4]

3.Photonic Logic

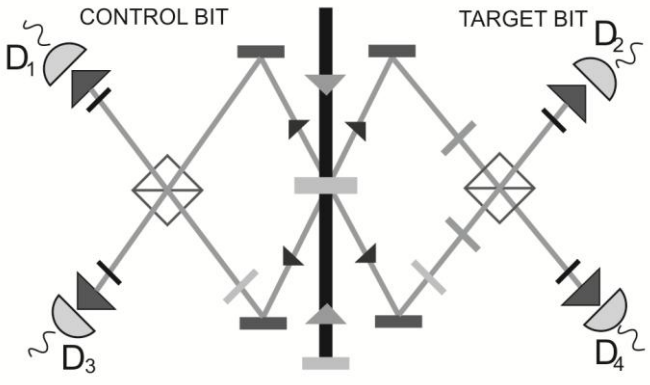


Fig 1Realization of a Photonic Controlled-NOT Gate for use in Quantum Computing

3.1 Photonic logic introduction

Photonic logic is the use of photons (light) in logic gates (NOT, AND, OR, NAND, NOR, XOR, XNOR). Switching is obtained using nonlinear optical effects when two or more signals are combined. Resonators are especially useful in photonic logic, since they allow a build-up of energy from constructive, thus enhancing optical nonlinear effects. Other approaches currently being investigated include photonic logic at a molecular level, using photo luminescent chemicals.

3.2 Photo luminescent

Photoluminescence (abbreviated as PL) is a process in which a substance absorbs photons (electromagnetic radiation) and then re-radiates photons. Quantum mechanically, this can be described as an excitation to a higher energy state and then a return to a lower energy state accompanied by the emission of a photon. This is one of many forms of luminescence (light emission) and is distinguished by photo excitation (excitation by photons); the period between absorption and emission is typically extremely short, in the order of 10 nanoseconds. Under special circumstances, however, this period can be extended into minutes or hours. Ultimately, available energy states and allowed transitions between states (and therefore wavelengths of light preferentially absorbed and emitted) are determined by the rules of quantum mechanics. A basic understanding of the principals involved can be gained by studying the electron configurations and molecular orbital's of simple atoms and molecules. More complicated molecules and advanced subtleties are treated in the field of computational chemistry.

4. Photonic Logic Gate

In supra molecular chemistry and nana materials research have stimulated interest in the design and development of molecular electronic and photonic devices for information processing, sensing and computation.[7] Numerous examples of molecular nano devices operating as wires,[8] switches[9] and sensors[10] have been reported in the literature. The development of molecular scale logic gates responding to multiple inputs has been a particularly active area of research. [11] Specifically, photo luminescent logic gates exhibiting AND, [12] OR, [13] XOR, [14] NOR [15] and INH [16] functionality have all been demonstrated. Exploitation of molecular photonic properties is appealing in this regard due to the high absorption cross-sections and luminescence efficiencies of many molecular systems, as well as the high signal-to-noise ratios that may be achieved. However, the majority of these supramolecular systems operate exclusively in the solution phase, thus limiting the range of inputs that may be used to ions and other chemical species, or simple environmental stimuli such as temperature. [17] The lack of an externally addressable interface makes it uncertain whether these approaches can provide a viable and scalable technology suitable for integration into future hybrid nano electronic devices and circuits. To successfully address this challenge, it will be necessary to take advantage of the structural and electronic properties of molecules for the rational design and fabrication of photonic logic gates that can be addressed by an underlying metal or semiconductor substrate. To this end, we report on the demonstration of an externally addressable molecular photonic logic gate comprising a metal polypyridyl complex self-assembled at the surface of a nano crystalline semiconductor electrode that responds to electrical and chemical inputs provided by the substrate and the ambient solution, respectively. Scheme 1 outlines the design and operating principles of the system selected for study, *cis*-bis(cyano) ruthenium(II)-bis-2,2′-bipyridine-4,4′-dicarboxylate ($Ru(dcbpy)_2(CN)_2$, **I**), adsorbed at the surface of a nano porous nano crystalline TiO_2 thin film. Chelation of **I** onto the TiO_2 substrate *via* its pendant carboxylate groups allows for both electrical and chemical switching functions to be integrated into a single molecular device. The electrical switching function is accomplished by modulating the potential applied to the semiconductor substrate, employed as the working electrode in a three-electrode single compartment electrochemical cell. It is well known that dye sensitization of TiO_2 films usually results in quenching of dye luminescence as a consequence of efficient excited state mediated electron transfer to the TiO_2 conduction band. However, luminescent emission from the dye may also beswitched on at applied potentials more negative than the TiO_2 flat band potential (V_{fb}) due to the sharp reduction in charge injection yields that occurs following increased occupancy of the electronic density of states with the electrode. Therefore, at applied potentials more negative than the TiO_2 V_{fb} (0,0), visible excitation of the Ru^{II} complex results in a strong metal-to-ligand charge transfer (MLCT) based luminescence in the red; see Scheme 1(a). In contrast, application of a more positive bias, (1,0), leads to almost complete luminescence quenching due to charge injection from the ^{1}MLCT and thermalised ^{3}MLCT excited states of the Ru^{II} complex into the TiO_2 conduction band; see Scheme 1(b). The second switching function is accomplished by introduction of Cu^{2+} ions into the adjacent electrolyte solution; see Scheme 1(c). Luminescence quenching by Cu^{2+} ions has been previously reported in solution phase studies of the Ru (bpy) $_2$(CN) $_2$ complex. Here, formation of $[Ru(bpy)_2(CN)(CNCu)]^{2+}$ and $[Ru(bpy)_2(CNCu)_2]^{4+}$ species results in static quenching of complex luminescence. Dynamic quenching, *i.e.*, luminescence quenching by unbound metal cations diffusing from solution, is also a significant additional pathway. These switching functions were combined to demonstrate a two-input molecular photonic logic gate, using the MLCT based luminescence of the complex as the output

signal, and the potential applied to the semiconductor substrate, in the presence or absence of Cu^{2+} ions, as the inputsignals.Complex **I** was synthesized according to previously reported procedures; see Supporting Information. † **I** was adsorbed from solution onto the surface of a nanostructured TiO_2 (anatase) thin film on fluorine-doped tin oxide coated glass. The **I**-functionalized substrate was incorporated as the working electrode in a single compartment spectro electrochemical cell, with a Pt. rod counter electrode and an Ag/AgCl reference electrode. Shows the emission spectra of an **I**-functionalized Nano crystalline TiO_2 film recorded under each of the four possible input conditions. In the (0, 0) state, excitation at 467 nm results in strong complex phosphorescence with a maximum at 668 nm. In the (1,0) state, this emission is switched off by the applied positive bias due to luminescence quenching by charge injection. In the (0,1) state, *i.e.*, at negative applied potentials, but in the presence of added Cu^{2+} ions, the emission is also switched off. Finally, in the (1,1) state, both switching functions can contribute to quenching of the MLCT based luminescence, thus completing the NOR logic gate truth table; see Fig. 1(b). Comparison of the output luminescence intensity of the (0,0) state with that of each of the three other states gives an average device on/off ratio of ca. 30 1 (at 668 nm), a value that compares very favorably with those obtained for solution based molecular logic gates.

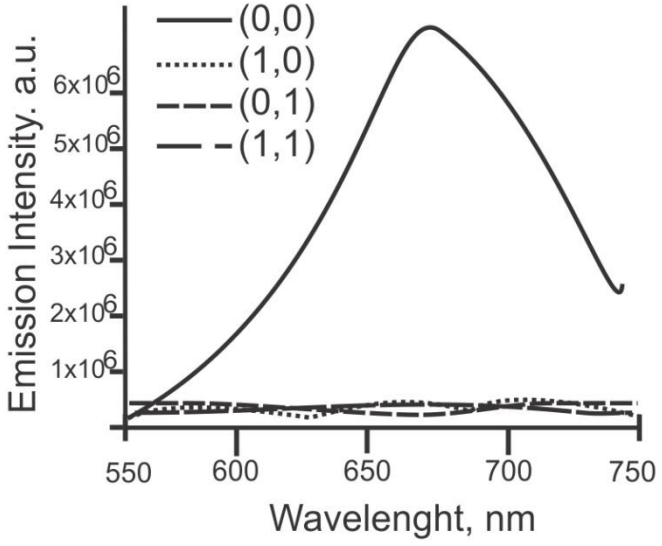


Fig 2 Emission spectra of an I-functionalized nanocrystalline TiO2 film recorded as a function of applied potential and the presence of Cu2+ ions. Electrolyte solution: 0.1 M LiClO4 in CH3CN

Corresponding NOR gate truth table, where Vapp= 0 indicates Vapp Vfb, Vapp= 1 indicates Vapp < Vfb, and [Cu2+]= 1 indicates the presence of added Cu2+.) Neither corresponding NOR gate truth table, where Vapp= 0 indicates Vapp Vfb, Vapp= 1 indicates Vapp < Vfb, and [Cu2+] = 1 indicates the presence of added Cu2+.

Input 1 (V_{app})	**Input 2** ($[Cu^{2+}]$)	**Output** (MLCT Emission)
0	**0**	**1**
1	**0**	**0**
0	**1**	**0**
1	**1**	**0**

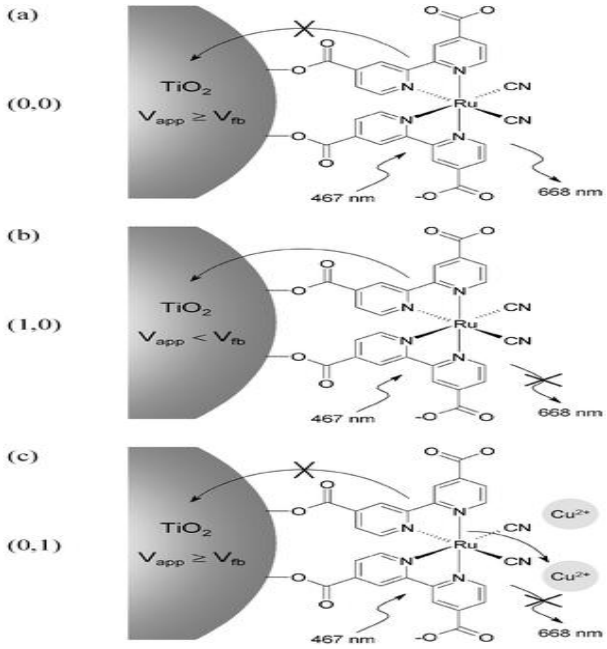


Fig 3 Scheme 1 Design and operating principles of the integrated molecular photonic logic gate. (a) Under negative applied potentials and in the absence of Cu^{2+} ions (0, 0), visible excitation of the Ru^{II} complex results in a strong MLCT based luminescence output. The luminescence output may be switched off by either (b) the application of positive applied potentials in the absence of Cu^{2+} ions (1,0), (c) the introduction of Cu^{2+} ions into the adjacent electrolyte solution under negative applied potentials (0,1) or (not shown) the introduction Cu^{2+} ions into the adjacent electrolyte solution under positive applied potentials (1,1)

5. Conclusion

So in conclusion we can say that by using photonic computing logic based on photons of visible light or infrared beams the computers with better performance could be achieved. We demonstrated an all-optical NOR logic gate based on symmetric GaAs-AlGaAs microring resonators whose resonances are closely matched. Two input pump data streams are tuned close to one resonance of the symmetric microrings

to switch a probe beam tuned to another resonance by two-photon absorption. The switching energy of the gate is 20 pJ/pulse, and the switching window is 40 ps, limited by the carrier lifetime. The use of two rings provides for better cascading in photonic logic circuits because of the higher number of available ports.

References

[1] Mind at Light Speed, David Nolte, page 34

[2] www.wisegeek.com/what-is-optical-computing

[3] Rp photonics

[4] Mind at Light Speed, David Nolte, page 36

[5] Ibrahim TA, Amarnath K, Kuo LC, Grover R, Van V, Ho PT. Photonic logic NOR gate based on two symmetric microring resonators. Opt Lett. 2004 Dec 1;29(23):2779-81.

[6] Ibrahim TA, Amarnath K, Kuo LC, Grover R, Van V, Ho PT.

[7] *Molecular Electronics*, ed. M. A. Ratner and J. Jortner, Blackwell, Oxford, 1997; J.-M. Lehn, *Supramolecular Chemistry*, Wiley-VCH, Weinheim, 1995; V. Balzani and F. Scandola, *Supramolecular Photochemistry*, Ellis-Horwood, Chichester, 1991; *Handbook of Nanoscience, Engineering and Technology*, ed. W. A. Goddard III, D. W. Breener, S. E. Lysehevski and G. J. Iafrate, CRC Press, Boca Raton, 2003; *Nanoelectronics and Information Technology*, ed. R. Waser, Wiley-VCH, Weinheim, 2003.

[8] C. Joachim and S. Roth, Atomic and Molecular Wires, *NATO ASI Ser, Ser. E 341*, Kluwer, Dordrecht, 1996; W. B. Davis, W. A. Svec, M. A. Ratner and M. R. Wasielewski, *Nature*, 1998, 396, 60–63 ; B. Schlicke, P. Belser, L. de Cola, E. Sabbioni and V. Balzani, *J. Am. Chem. Soc.*, 1999, 121, 4207–4214 ; V. Grosshenny, A. Harriman and R. Ziessel, *Angew. Chem., Int. Ed. Engl.*, 1996, 34, 2705–2708 .

[9] *Molecular Switches*, ed. B. L. Feringa, Wiley-VCH, Weinheim, 2001; R. W. Wagner, J. S. Lindsey, J. Seth, V. Palaniappan and D. F. Bocian, *J. Am. Chem. Soc.*, 1996, 118, 3996–3997 ; M. P. Debreczeny, W. A. Svec and M. R. Wasielewski, *Science*, 1996, 274, 584–587 ; P. R. Ashton, V. Balzani, J. Becher, A. Credi, M. C. T. Fyfe, G. Mattersteig, S. Menzer, M. B. Nielsen, F. M. Raymo, J. F. Stoddart, M. Venturi and D. J. Williams, *J. Am. Chem. Soc.*, 1999, 121, 3951–3957 ; P. L. Boulas, M. Gomez-Kaifer and L. Echegoyen, *Angew. Chem., Int. Ed.*, 1998, 37, 216–247; S. H. Kawai, S. L. Gila and J.-M. Lehn, *J. Chem. Soc., Chem. Commun.*, 1994, 1011–1013.

[10] *Fluorescent Chemosensors of Ion and Molecule Recognition*, ed. W. W. Czarnik, *ACS Symp. Ser. 538*, American Chemical Society, Washington DC, 1993; A. P. de Silva, H. Q. N. Gunaratne, T. Gunnlaugsson, A. J. M. Huxley, C. P. McCoy, J. T. Rademacher and T. E. Rice, *Chem. Rev.*, 1997, 97, 1515–1566 .

[11] V. Balzani, A. Credi and M. Venturi, *Molecular Devices and Machines*, Wiley-VCH, Weinheim, 2003; A. P. de Silva and N. D. McClenaghan, *Chem. Eur. J.*, 2004, 10, 574–586 .

[12] A. P. de Silva, H. Q. N. Gunaratne and C. P. McCoy, *Nature*, 1993, 364, 42–44 ; A. P. de Silva, H. Q. N. Gunaratne and C. P. McCoy, *J. Am. Chem. Soc.*, 1997, 119, 7891–7892 .

[13] A. P. de Silva, H. Q. N. Gunaratne and G. E. M. Maguire, *J. Chem. Soc., Chem. Commun.*, 1994, 1213–1214 .

[14] A. Credi, V. Balzani, S. J. Langford and J. F. Stoddart, *J. Am. Chem. Soc.*, 1997, 119, 2679–2681.

[15] A. P. de Silva, I. M. Dixon, H. Q. N. Gunaratne, T. Gunnlaugsson, P. R. S. Maxwell and T. E. Rice, *J. Am. Chem. Soc.*, 1999, 121, 1393–1394 .

[16] T. Gunnlaugsson, D. A. MacDónaill and D. Parker, *J. Am. Chem. Soc.*, 2001, 123, 12866–12876 ; T. Gunnlaugsson, D. A. MacDónaill and D. Parker, *Chem. Commun.*, 2000, 93–94 .

[17] S. Uchiyama, N. Kawai, A. P. de Silva and K. Iwai, *J. Am. Chem. Soc.*, 2004, 126, 3032–3033.

Designing a Secure 32-bit ALU using (63, 36) BCH code

A. Vahid Khorasani[1], B. Mohammad Morazavi[2], and C. Bijan Vosoughi Vahdat[2]
[1]Department of Electrical Engineering, Sharif University of Technology, International campus, Tehran, Iran
[2]Department of Electrical Engineering, Sharif University Of Technology, Tehran, Iran

Abstract - *Reliable communication has become very crucial in the transmission applications. Hence, to design hardware to handle reliability is most important part of communication. In this work, we propose a new secured ALU (Arithmetic and Logic Unit) against fault attacks that is used in ARM processor which can correct any 5-bit error in any position of 32-bits input registers of ALU. In this work, we also designed a BCH (Bose, Chaudhuri, and Hocquenghem) codec (encoder, decoder) using the prototyping FPGA. Further, we designed (63, 36) BCH encoding and decoding system to tolerate the 5-bit faults. Since the usages of the ARM (Advanced RISC Machine) processors are more applicable for control system, we give the fault tolerance characteristic through the error control coding to this processor. As a result, the core for implementation of an ALU employing BCH code on Spartan-3 FPGA has been provided. Our Fault tolerant ALU has high reliability. Moreover, it consumes low hardware overhead with acceptable fault coverage*.

Keywords: Fault tolerant, ALU, BCH codes, Residue codes, TMR, Encoding, Decoding, FPGA.

1 Introduction

Nowadays, by increasing the usage of digital systems and improvement of modern technology, working on reliable communication transmission play an important role. So that a single error may shutdown the whole system and give rise to incredible or erroneous data. System reliability is one of major issues in embedded processors designs for space application such as satellite, military, communications etc. Various attacks exist in space on integrated circuits that comes from sun activity [1]. Such as solar rays which are composed of charged particles. The radiation from sun effects in integrated circuits make digital damage and upsets such as SEU (Single Event Upset), SET (Single Event Transient) and etc as presented in [2]. Such attacks can upset either combinational logic or sequential logic. In other words a bit flip can occur in memory or register bits and if one bit of main storage system is changed the mission of system would be completely different. In such scenario the error control or fault tolerant methods are employed to keep integrated circuits against these attacks in space. To achieve such purpose we consider Error detection and correction codes (EDAC) method. It is usually used to mitigate SEU in integrated circuits which are required that the encoder and the decoder blocks to be able to detect and correct errors respectively. This technique gives strong faults coverage and less overhead hardware. For this reason we consider the BCH (Bose, chaudhuri, and Hocquenghem) codes and in this project a binary BCH codes is considered. As a result of using BCH codes, we have achieved to design encoder and decoder circuits to detect and correct 5-bit faults. Further we have designed a 32-bit ALU. Our 32-bit ALU model consists of the following function units: Arithmetic operation consists of Full-Adder and Subtractor. Bitwise logic operation such as: XOR, AND, OR, and NOT. Bit-shifting operations such as: shifting to the left or right Encoder and decoder block and 32-bit ALU is based upon the use of Verilog description language. Furthermore, we have presented design of a secure ALU (Arithmetic and Logic Unit) against faults. This ALU is able to correct any 5-bit error in any position of its 32 bits input registers. Consequently, the core for implementation of an ALU employing BCH code on Spartan-3 FPGA has been provided. The proposed system has been simulated on Modelsim 6.2b and its performance has been verified by ISE 8.2i.

1.1 Motivation

Our goal is to overcome the difficulties of designing a new 32-bit ALU that is robust against many attacks or faults and able to correct any 5-bit fault in any position of its 32 bits input register of ALU. Because the radiation effects on electronic circuits may cause to be inverted data bits of registers or memories. If one bit of main storage system is changed the mission of system would be completely different. In fact this project is an essential and vital part for OBC (On Board Computer) of a satellite due to the fact that there are high radiations in space such as solar rays and cosmic radiations. The high motivation in choice of BCH codes is that, it is able to correct multiple errors and these classes of codes are kind of powerful random error correcting cyclic codes. Moreover, by choosing two essential parameters n and t, the designer is able to design any BCH code. These implies that a crucial motivation because the structure of BCH encoder and decoder in presence of two parameters *n* and *t* can be notably different.

1.2 Problem Definition

Since the usages of the ARM processors are more applicable to control systems, we need to focus on the fault tolerance characteristic through the ECS (Error Control System) to ARM processor. Consequently, the core for implementing on FPGA will be provided.

1.3 Application

Apart from many applications such as military applications to protect against intentional enemy interfaces one considerable point for us in this project is On Board Computer (OBC) of a satellite.

1.4 Overview

The rest of this paper is presented as follows: In section 2, mathematical background of BCH code is described. In section 3, proposed method based on BCH code for decoding are described. In section 4, Hardware implementation of an encoding and decoding of (63, 36) BCH code applied in our work are presented. In section 5, our designing 32-bit fault tolerant ALU are described. section 6, describes conclusion and future work are described.

2 Background to the research objectives

Error detection and correction (EDAC) **codes** technique is usually utilized to mitigate SEU in integrated circuit and it requires extra hardware. Nevertheless this technique gives strongly faults coverage. Hence, we are considering this technique to cover our goals. EDAC codes can be implemented in two ways and it depends on transmission data.

If error control system can transfer data bidirectionally, where the receiver detects an error in a data frame. It automatically requests the transmitter to resend the data frame. The systems are known as an automatic repeat request (ARQ). If transmission occurs in only one direction to overcome error, the error control system works out through the forward error correction (FEC) [5].

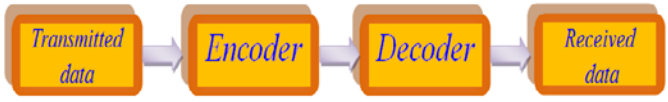


Figure 1. Scheme of a forward error correction (FEC).

In communication theory FEC is a method used for data transmission so that, the sender including m-bits of redundant data, thereby the receiver is able to detect and correct the corrupted data without asking to retransmit it. Therefore, FEC method will be consider in our design. The schematic of an FEC method is shown in figure 1. One of the significant classes of FEC codes is linear block codes and binary BCH codes are classes of linear block codes.

BCH codes are a generalization of Hamming code for a multiple error correcting. Further, these codes are a large class of powerful random-error correcting cyclic codes [4].

2.1 Mathematical background

Error control codes such as BCH codes strongly depend on algebraic structures that are called Finite fields or Galois field. This field consists of element set which could be done addition, subtraction, multiplication and division operations over field elements. The number of elements in a field is known as the order of the field. A field with a finite number of elements is known as a finite field which is called Galois field.

The prime field is a field that built under modulo p-addition and p-multiplication where p in this field is set of integers {0,1,2,…, P-1} that is called prime field and it is denoted by GF (P). It can be possible to extend the prime field to a field of P^m elements which is known as an extension field of GF(P) and is denoted by GF(P^m). For P=2 the field is denoted by GF(2) and it is called binary field. In this paper this field is considered. This field consists of binary numbers. The binary field GF(2) extensively used in digital computer, digital data transmission and storage systems [5]. Furthermore, the representations of elements in Galois field easily are mapped into the digital domain. Operating in GF(2) has been extended to generate the extension GF(2^m). The GF(2^m) is so specified as a field with 2^m elements and each of element is binary m-tuple. In fact GF(2^m) form a m-tuple vectors with element belong to GF(P). In this paper GF(2^6) consider to do designing the codec system.

In field theory an irreducible polynomial P(X) of order m is called primitive if the smallest positive integer n, for which P(x) divides X^n-1 is equal to 2^m-1 [5].

The primitive element is used to construct GF(2^m) from GF(2). Further, it is one of the most important factors to determine the BCH code. In fact a primitive element of GF(2^m) is an element α such that every field element except zero that can be shown as a power of α every finite field contains a primitive element [5].

Table 1. Primitive polynomial over GF(2^m).

m	Primitive polynomial	m	Primitive polynomial
3	$1+x+x^3$	7	$1+x^3+x^7$
4	$1+x+x^4$	8	$1+x^2+x^3+x^4+x^8$
5	$1+x^2+x^5$	9	$1+x^4+x^9$
6	$1+x+x^6$	10	$1+x^3+x^{10}$

For each element α in GF(2^m) there exist a unique monic polynomial m(x) of minimal degree in GF(2^m) such that in the following terms are true.

1. m (α) = 0.
2. The degree of m(x) is less than or equal to m.
3. f (α)= 0 implies that f(x) is a multiple of m(x).
4. m (x) is irreducible in GF(2^m) [6].

2.2 Binary primitive BCH codes

For an (n, k) BCH code with any positive integer $m \geq 3$, and $t<2^m-1$, there are an (n, k) binary BCH code as follows:

t ; The most number of errors that can be corrected

$n = 2^m - 1$; Length of codeword (1)

$n - k \leq mt$; Number of parity bits (2)

$2t + 1 \leq d_{\min}$; Minimum Hamming distance (3)

In order to encode and decode, at the beginning, the below stages are required to determine the BCH code.

1. Construct GF(2^m), with choice of primitive polynomial P(x).
2. Get the minimal polynomial $m_i(x)$ of α^i, i=1, 2,…,2t.
3. Obtain the generator polynomial to encode the data.

Further, with choice of k, we obtain *n-k* which gives us the degree of generator polynomial of BCH code. Finally, minimum distance is determined from equation (3) if we choose t=1, $d_{min} \geq 6$. An (63, 36) code is capable of correcting any combination of t=1 in a block of n=63 digits. BCH codes are multiple error correcting and these codes also are a class of cyclic codes whose generator polynomial with length of 2^m-1 has $\alpha^1,\alpha^2,\alpha^3,...,\alpha^{2t}$ ($1\leq i \leq 2t$) as it roots. The generator polynomial g(x) of the t-error correcting BCH code is determined to be the least common multiple of minimal polynomial $m_i(x)$.

$$g(x)=LCM\{m_1(x), m_2(x)\ldots m_{2t}(x)\} \quad (4)$$

We can generate a codeword for an (n, k) t-error-correcting BCH code. The 36 data bits are formed in to the data part in figure 2; Where $i_i \in GF(2)$ and the 27 parity bits are formed in the left most.

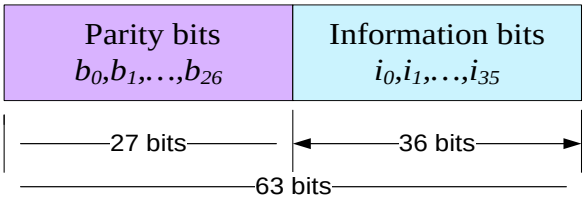


Figure 2. Systematic format of a codeword for an (63,36) BCH code.

3 Proposed method based on BCH code

The decoding of BCH code is composed of three main steps that are expressed as follows:

1) Compute the syndromes from the received codeword.
2) Obtain the error locator polynomial σ(x) (ELP) through the BMA (Berlekamp-Massy-Algorithm)
3) Determine the error-location numbers by finding the roots of error location polynomial (identifying the position of erroneous bit).

3.1 The syndrome computation

The syndromes identify whether error has occurred. If the syndromes all are zero, we will have no error in codeword and if the syndromes not be zero we will have error in codeword. Assuming that r(x) be the received codeword

$$r(x) = r_0 + r_1 x + r_2 x^2 + \ldots + r_{n-1} x^{n-1} \quad (5)$$

And e(x) be the error pattern

$$e(x) = e_0 + e_1(x) + e_2 x^2 + \ldots + e_{n-1} x^{n-1} \quad (6)$$

W(x) be the code word or transmitted polynomial

$$w(x) = w_0 + w_1 x + w_2 x^2 + \ldots + w_{n-1} x^{n-1} \quad (7)$$

Therefore, the received codeword can be shown :

$$r(x) = w(x) + e(x) \quad (8)$$

If $1\leq v\leq t$, where t is the number of error which may be occurred in unknown location $j_1,j_2,\ldots j_v$,that is

$$e(X) = X^{j1} + X^{j2} + \ldots + X^{jv} \quad (9)$$

Whereas, $\alpha, \alpha^1, \alpha^2, \ldots, \alpha^{2t}$ are roots of each code polynomial, $W(\alpha^i) = 0$, for $1\leq i \leq 2t$. Therefore, we have [5].

$$r(\alpha^i) = e(\alpha^i), \quad i = 1, 2, \ldots, 2t \quad (10)$$

The first step of decoding procedure is called "syndrome computation". For computing the syndromes, the syndrome S_i is defined as:

$$\begin{aligned} S_i &= r(\alpha^i) = r_{n-1}\alpha^{(n-1)i} + r_{n-2}\alpha^{(n-2)i} + \ldots + r_1\alpha^i + r_0 \\ &= (\ldots((r_{n-1}\alpha^i + r_{n-1})\alpha^i + r_{n-3})\alpha^i + \ldots + r_1)\alpha^i + r_0 \end{aligned} \quad (11)$$

Where $1 \leq i \leq 2t$. The syndromes are the set of the field elements in GF(2^m) . Therefore, each syndrome component is calculated by dividing r(x) by the minimal polynomial $m_i(x)$ of α^i

$$r(x) = q_i(x)\, m_i(x) + b_i(x) \quad (12)$$

$m_i(x)$ is the minimal polynomial and $b_i(x)$ is the reminder So, by evaluating $b_i(x)$ with $X = \alpha^i$, we can be found the syndrome components. Since,

$$m_i(\alpha^i) = 0$$

We have the following equation,

$$S_i = r(\alpha^i) = b_i(\alpha^i) \quad (13)$$

Hence, with equation (13) the syndrome component is

$$S_i = r(\alpha^i) = b_i(\alpha^i) = e(\alpha^i), \quad i = 1, 2, \ldots, 2t \quad (14)$$

Therefore, the above equation represent that syndrome just depends on the error pattern. This subject is the significant characteristic of the syndromes. Now, we have a set of equations with unknown parameters.

$$S_i = (\alpha^{j1})^i + (\alpha^{j2})^i + \ldots + (\alpha^{jv})^i \quad 1 \leq i \leq 2t \quad (15)$$

We see that computing the 2t syndrome components $S_1,S_2,\ldots,S_{2t}$ can be computed by substituting the field element $\alpha, \alpha^2, \ldots, \alpha^{2t}$ in to the received polynomial r(x) in decoding a t-error –correcting BCH code.

3.2 Finding the error location polynomial through the simplified Berlekamp-Massy Algorithm

We assume that the numbers of errors $v\leq t$ have occurred and error locator polynomial σ(x) is:

$$\sigma(x) = \sigma_0 + \sigma_1 x^1 + \sigma_2 x^1 + \ldots + \sigma_v x^v \quad (16)$$

$$\sigma(x) = (1+\beta_1 x)(1+\beta_2 x)\ldots(1+\beta_v x) \quad (17)$$

The coefficient of error locator polynomial and the error location numbers are related by the following set of equations [5].

$$\begin{aligned}\sigma_0 &= 1\\ \sigma_1 &= \beta_1 + \beta_2 + \ldots + \beta_\nu\\ \sigma_2 &= \beta_1\beta_2 + \beta_2\beta_3 + \ldots + \beta_{\nu-1}\beta_\nu\\ &\cdot\\ \sigma_\nu &= \beta_1\beta_2 + \ldots + \beta_\nu .\end{aligned} \qquad (18)$$

Where the coefficient of error locator polynomial σ_i , $1 \le i \le \nu$ are related to the syndrome components S_i , $1 \le j \le \nu+1$ [8].

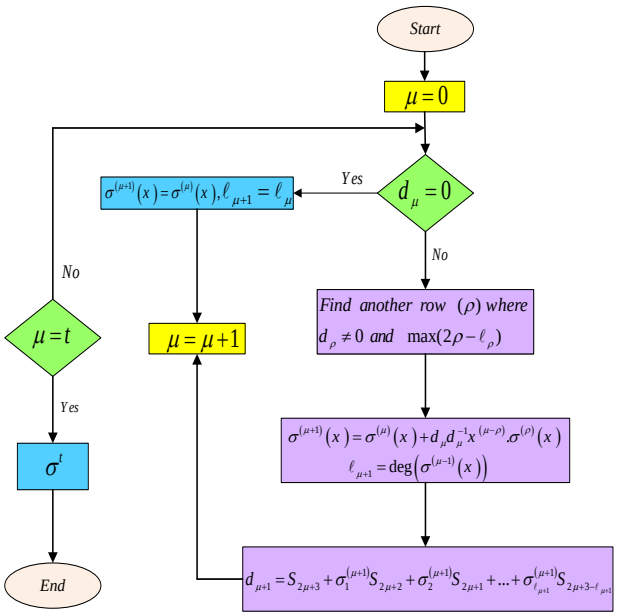


Figure 3. The inversion Berlekamp Massy Algorithm.

The BMA procedure begins with the 2t syndrome. Components (S_1, S_2,…, S_{2t}) which it is possible to determine the coefficients $\sigma_1, \sigma_2, \ldots, \sigma_t$ of the error-location polynomial. This algorithm is simplified to t-steps for calculating σ (x), and when we carry out the step t-1 , perfectly σ^{μ} (X) is the final error location polynomial and if the degree of σ (x) is greater than t , we will have more than t errors then , the received codeword cannot be corrected [5]. The inversion Berlekamp massy algorithm is shown in Fig. 3.

4 Hardware implementation of a (63, 36) BCH encoder and decoder for t=5

The selected BCH code parameters for designing an encoder and decoder for this project is (63,36,5) which is able to correct 5-bit error. Consider 5-error-correcting (63,36) BCH code. If α be a primitive element in GF(2^6) such that $1+\alpha+\alpha^6=0$, We have the list of the minimal polynomials of the elements in GF(2^6) for an (63,36) BCH code that are shown in Table 2. The table is indicating this fact that for some roots we have the same minimal polynomials. We utilized these minimal polynomials as hardware implementation of decoding such as syndrome calculation step. The generator polynomial is required to encode the information. The generator polynomial is computed for an (63, 36, 5) BCH code and is shown in Table 3.

Table 2. List of minimal polynomial of the elements in GF(2^6) for an (63,36,5) BCH code.

Elements	Minimal polynomial
$\alpha, \alpha^2, \alpha^4, \alpha^8$	$m_1(x) = 1+x+x^6$
α^3, α^6	$m_2(x) = 1+x+x^2+x^4+x^6$
α^5, α^{10}	$m_3(x) = 1+x+x^2+x^5+x^6$
α^7	$m_4(x) = 1+x^3+x^6$
α^9	$m_5(x) = 1+x^2+x^3$

Table 3. The generator polynomial for an (63,36) BCH code.

# of error correction	BCH code	Generator polynomial
t=5	(63,36)	$g_2(x)=LCM\{m_1(x),m_2(x),m_3(x),m_4(x),m_5(x)\}$ $=1+x+x^4+x^8+x^{15}+x^{17}+x^{18}+x^{19}+x^{21}+x^{22}+x^{27}$

4.1 Hardware implementation of a (63, 36) BCH encoder

The encoder circuit [4], calculates the parity bits using the LFSR (Linear Feedback Shift Register) whose composed of 2-input XOR gates. The generator polynomial of the (63, 36) BCH code is required to implement the encoding and this polynomial is given as follows:

$$G(X) = X^{27} + X^{22} + X^{21} + X^{19} + X^{18} + X^{17} + X^{15} + X^{8} + X^{4} + X^{1} + 1.$$

This code is equivalent to the following binary code which is

1 000 011 011 101 000 000 100 010 011

used instead of g_1 to g_{26} values in the following circuits fig.4. So the feedback connections of the LFSR are formed in this manner.

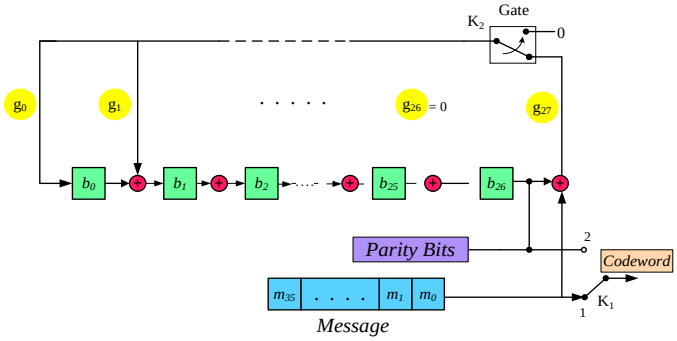


Figure 4. Encoding circuit for an (63, 36) BCH code.

The degree of G(x) is *m*. So the feedback connections of the LFSR are formed in the following manner that is shown in figure 4. In this circuit, the input data is 36 bits and the output is a serial bit of 63 bit data generated by BCH encoder. This operation is done in two steps: In the first step the gate K_2 is

on and for clock cycle 1 to 36 the message digits, at the same time are shifted in an unchanged manner into the output further, switch K_1 is in position 1 and the parity check bits are calculated in LFSR. In the second step from clock cycle *37* to *63* (where switching K_1 in position 2) the parity-check bits in the LFSR are shifted and K_2 in this stage is off and then these *27* parity bits concatenate with the *36* messages to form a systematic codeword. In Fig. 5 the output of encoding logic circuit is a serial bit of 63 bit data.

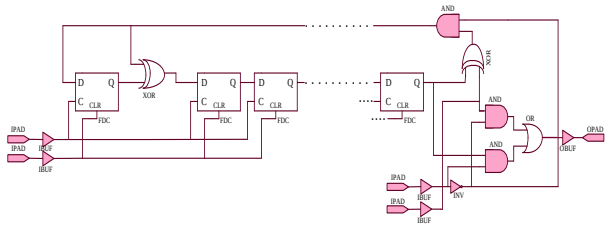


Figure 5. A (63, 36) BCH encoding logic circuit implemented in FPGA

4.2 (63, 36) BCH decoder implementation

The decoding process consists of three steps that are described in section III. The following fig.6 is more clear what is require for decoding technique.

We have implemented the syndrome computation circuit for 5-error-correcting (63, 36) BCH code. In this task we are stored the received polynomial in a buffer register to compute the syndrome from the received codeword r(x).

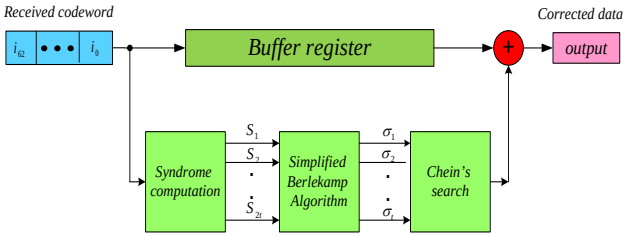


Figure 6. Outline for decoder technique using (63,36) BCH code.

When the entire received codeword has entered the decoder, 10 syndrome components (s_1, s_2,…, s_{10}) are formed. It takes 63 clock cycles to complete the computation. Since, the generator polynomial is a product of at most 5 minimal polynomials Therefore at most 5 feedback shift register , each consist of at most 6 stages, are required to form the 10 syndrome components. The fraction of syndrome computation circuit for 5-error correcting (63, 36) BCH code is shown in Fig. 7.

This process stored the received codeword in a buffer register to compute the syndrome. It takes 63 clock cycles to complete the computation. A Chein's searching circuit [3], for the 5-error correcting (63, 36) BCH code is shown in figure 8. This circuit is applied to identify the position of erroneous bits into 63-bit received codeword and correct it.

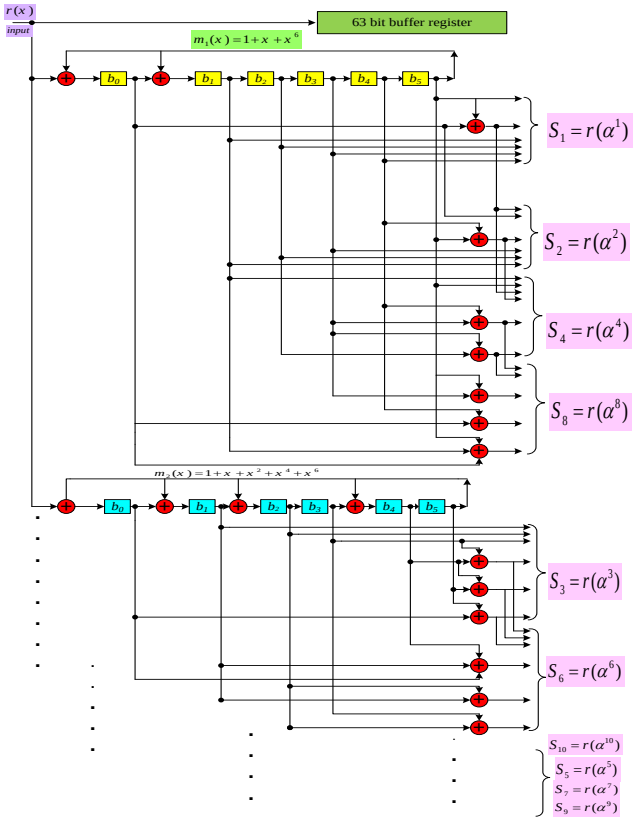


Figure 7. Syndrome computation of (63,36) BCH code in GF(2^6).

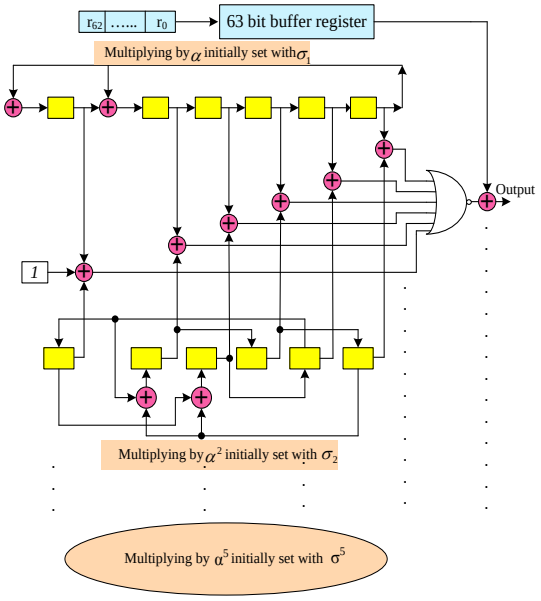


Figure 8. LFSR circuit for Chein's error location searching.

5 Proposed Design & Experimental result

Our 32-bit ALU model consists of the following function units: Arithmetic operation consists of Full Adder and Subtractor. Bitwise logic operation such as: XOR, AND, OR,

and NOT. Bit-shifting operations such as: shifting to the left and right. The codec circuits are applied to correct any five error occurred in any position of 32-bits input registers of ALU. Our algorithm of fault tolerant ALU is shown in Fig. 10. In this algorithm at first the 63 bits input register A and B is read out one by one, by the decoding system. Registers A, B consist of 27-bits parity check; 36-bits data which are 4-bits are extra. We use these extra bits as parity check bits as shown in fig.10. If any error occurs in any position of 63-bits the decoder will correct the erroneous bit at once. In output the decoding system gives 36-bits where we need only 32-bits as 2-inputs of ALU (we don't store 4-bits of 36-bits). So, the output of 32-bit ALU is the C´ resister. We add the 4-bits zero for leftmost of C´ register to convert the 36-bits for the input of encoder system. After encoding the data of register C is keep in register A. 4-bit of 36-bit data are extra then, we have utilized them as parity check bits to detect and correct the error in presence of parity maker and XOR gate. In figure 9, The 32nd bit is the parity check for the first 8-bit (0-7), the 33nd bit is checked the next 8-bit of data (8-15), the 34nd bit is the parity check for the 16 to 23 and finally the 34nd is the parity check for the last position bits of data from 24 to 31.

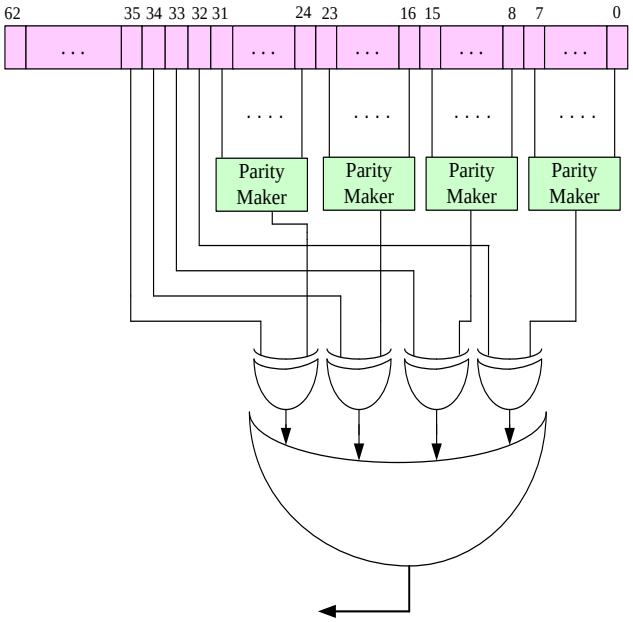


Figure 9. Decoding of register by making extra parity check bits.

The circuits have been programmed in ISE version 8.2i, and are simulated in Modelsim SE 6.2b, before it is implemented in XC3S400 from Spartan-3 FPGA family.

The Xilinx company for XC3S400 offer 400000 system gates, 8064 D-flip flops and the number of CLB (Configured Logic Blocks) is 896 (one CLB = four slices). The important features of this device consists of sixteen dedicated 18×18 multipliers in it, low cost and high-performance logic solution for high volume [7]. The design performance has been verified by ISE 8.2i which is shown in figure 11.

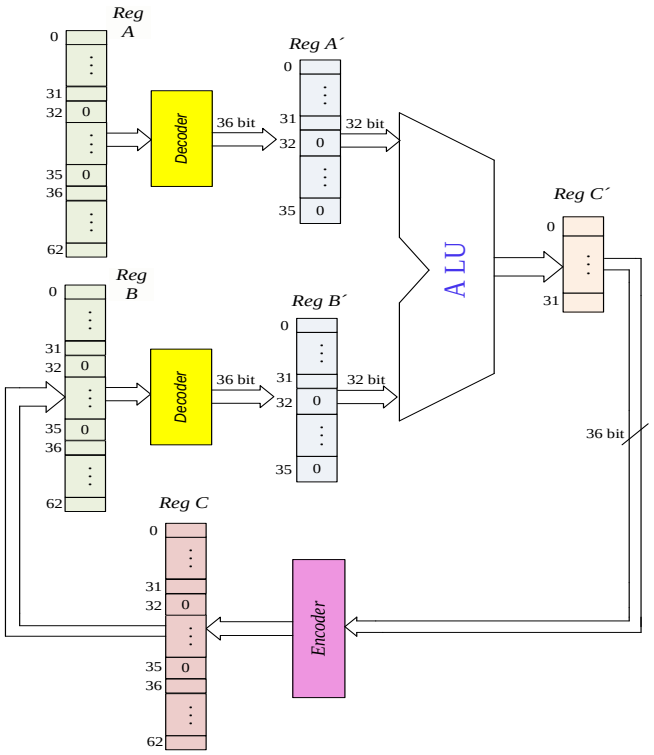


Figure 10. The fault tolerant ALU Algorithm using (63,36) BCH code.

ALU Project Status			
Project File:	ALU.ise	Current State:	Programming File Generated
Module Name:	BIT_ALU	• Errors:	No Errors
Target Device:	xc3s400-5pq208	• Warnings:	4 Warnings
Product Version:	ISE 8.2i	• Updated:	Tue Jan 11 19:27:04 2011

ALU Partition Summary
No partition information was found.

Device Utilization Summary				
Logic Utilization	Used	Available	Utilization	Note(s)
Number of 4 input LUTs	285	7,168	3%	
Logic Distribution				
Number of occupied Slices	145	3,584	4%	
Number of Slices containing only related logic	145	145	100%	
Number of Slices containing unrelated logic	0	145	0%	
Total Number of 4 input LUTs	285	7,168	3%	
Number of bonded IOBs	100	141	70%	
IOB Latches	32			
Number of GCLKs	1	8	12%	
Total equivalent gate count for design	2,068			
Additional JTAG gate count for IOBs	4,800			

Performance Summary			
Final Timing Score:	0	Pinout Data:	Pinout Report
Routing Results:	All Signals Completely Routed	Clock Data:	Clock Report
Timing Constraints:	All Constraints Met		

Figure 11. The Design performance of Fault Tolerant ALU

6 Conclusions

The radiation effect causes faulty data in satellite communication. In this paper, we proposed to remove these faulty data by using a (63,36) BCH code. We have implemented a codec with a new ALU (inside an ARM processor) system employing fault tolerant algorithm using BCH code with Verilog hardware description language. The design has been simulated with Modelsim 6.2b and its performance verified by ISE 8.2i. The results indicate any five bits error in any position of 32-bits input registers of ALU can be corrected. Moreover, we can conclude that proposed system have essential advantages in terms of reliability and area occupation. It is high reliability in presence of faults and

susceptibility is very low. Thus it consumes less hardware overhead with high fault coverage. Future work can be extended to implement this algorithm on the 64-bit microprocessor.

ACKNOLEGMENT

The financial support of the Iran Telecommunication Research Center is gratefully acknowledged.

7 References

[1] S.Bourdarie and M.Xapsos, senior member, IEEE, "The near earth space radiation environment", IEEE Transaction on Nuclear Science, August, 2008.

[2] NASA, preferred reliability practices "space radiation effects on electronic components in low-earth orbit", April, 1996.

[3] R.T. Chien, "*Cyclic decoding procedure for the Bose-Chaudhuri-Hocquenghem codes,"*IEEE Trans. Inf. Theory, IT10, pp. 357-363, October 1964.

[4] W.W. Peterson, "*Encoding and error-correction procedures for the Bose-Chaudhuri Codes"*, IRE Trans. Inf. Theory, IT-6, pp. 459-470, September 1960.

[5] Lin, Shu, and Daniel J. Costello, Jr., "*Error Control Coding: Fundamentals and Applications*", Englewood Cliffs, NJ, Prentice-Hall, 1983.

[6] Rudolf Lidl, H. Niedereiter "Introduction to Finite field and their application" Cambridge university press, 1986.

[7] www.xillinx.com, Spartan-3 FPGA family data sheet, Jun, 2008.

Dealing with the "Itanium Effect"

Steve Richfield
Consultant
5498 124th Avenue East
Edgewood, WA 98372
00-1-505-934-5200
Steve.Richfield@gmail.com

ABSTRACT

The "Itanium Effect" is a subtle organizational phenomenon leading to the wide adoption of a few widely applicable technologies, and the abandonment of many powerful but more narrowly applicable technologies.

The main elements of the Itanium Effect are:

1. Technology loops
2. Compartmentalized conferences
3. Little PhD student participation
4. Procedural exclusion of futurist and top-down discussions
5. Keeping problems secret, so that no one else can help

The Itanium Effect has become the leading barrier to advancement of high performance computing. This is why defects continue to impair yield. This is what now stands in the way of wafer scale integration.

Prospective glue technologies examined in this paper include:

1. Logarithmic arithmetic
2. Medium-grained and multi-grained FPGAs
3. Coherent memory mapping
4. Variable data chaining
5. Fast aggregation across ALUs
6. Blurring the SIMD/MIMD distinction
7. A simple horizontal microcoding interface for applications
8. Failsoft configuration on power-up
9. Failsoft partial reconfiguration during execution
10. Symmetrical pinout to use of defective components.
11. An architecture-independent universal compiler to compile a new APL-level language

Category and Subject Descriptor

B.0 **[Hardware]:** General.

Keywords

coherent memory mapping, failsoft reconfiguration, logarithmic arithmetic, medium granularity, symmetrical pinout, universal compiler.

1. INTRODUCTION

A really incredible thing happened in 2001, which went completely unnoticed throughout the chip-making industry. Intel released the Itanium, a nearly precise monolithic copy of the biggest boondoggle in computer history – the 1961 IBM Project STRETCH. Further, there had been much contemporary analysis of the mistakes leading to the STRETCH boondoggle, partially documented in the book ***Planning a Computer System***. All of the recognized mistakes made on STRETCH, like the absence of a "guess bit" in conditional branch instructions, were recreated in the Itanium, despite recommendations by various commentators of 40 years earlier.

Now this same process is continuing with GPU designs that are paralleling the products of Floating Point Systems Inc. of ~30 years earlier. Sure, we have jumped about one decade ahead in this loop, but why not simply skip 3 more decades of now-obsolete designs of the past and move forward?

How could such crazy repetitions of history possibly happen? How is this same phenomenon continuing to radically depress the capabilities of all present-day CPU, GPU, and FPGA designs? How could correcting this phenomenon in your company be worth billions of dollars to your shareholders? Let's examine this continuing phenomenon.

Photo 1. IBM 7030 STRETCH Control Panel

2. TECHNOLOGY LOOPS

That things could proceed along a "linear" development path to perfectly close a gigantic loop as the Itanium did, suggests a path with technological cobblestones laid in gigantic circles. Present technology is defined by the hundreds of college courses and thousands of textbooks that teach the various past methods. Often, new methods appear on the scene, but remain in product-specific manuals that disappear as soon as products become obsolete (like the superior methods of fault tolerance pioneered by Tandem Corp), or appear at a time when industry just isn't ready

for them (like Ashenhurt and Metropolis' 1959 paper on significance arithmetic). If you look at the many technologies that appear in textbooks as points in a multidimensional space, people will predictably add points indefinitely until a closed loop is formed.

The STRETCH boondoggle foreclosed on the future of extensive instruction lookahead for 40 years, until engineers at Intel read books whose content was traceable to the last computer with extensive instruction lookahead, and substantially recreated the STRETCH, complete with its dead ends. In short, they didn't truly reinvent the STRETCH, but rather they copied its concepts and arrived at essentially the identical architecture, as you might reasonably expect from good engineers developing a common concept. STRETCH and its descendant, the Itanium, constitute just one of those many points in the multidimensional technological space. The precision of this re-creation shows that we have very nearly reached the end of incremental improvements in computer architecture, so now it is time to take some radical steps if we are to continue to move forward.

Note that some present-day coarse-grained FPGA proposals are traceable all the way back to the 1949 IBM-407 accounting machine that was able to use similar methods to achieve electronic speeds using slow electromechanical components.

Once a few closed loops are formed, there is no more pressing need to add additional technologies. Project development will then predictably jump from point to nearby point, as needed to push out new products, without ever reaching a point where anything radically new is needed to produce the next product. This has created a situation where corporate managers now think they need only hire PhDs who understand the various technologies, and then those PhDs can work for the rest of their lives, without ever having to leave the comfort zone of their own knowledge.

This would ordinarily leave everyone in the industry vulnerable to a maverick corporation that willingly adopts obscure technologies, if not for the extreme expense in developing new chips. Now, any such maverick corporation would probably need a billion or so dollars just to "ante in" to this game. Hence, the challenge is to find some way for existing corporations to morph their methods to break out of this loop and conquer their competition. Laggards will be forced out of business.

3. ENGINEERING ISOLATION

I usually attend the yearly WORLDCOMP, which consists of 20 separate conferences on all aspects of computing. This keeps me current on the very latest thinking in everything ranging from computer design to AI. WORLDCOMP includes separate conferences on computer and FPGAs design, including various panel discussions about hot topics of the year. Dozens of PhDs present their work, pushing the frontiers of computing a bit further ahead. One thing these conferences do not include is any representation from major hardware or software vendors, with some rare curious exceptions countable on the fingers of one hand. I routinely look up these very rare individuals to determine their place in their respective corporations, and their reasons for attending. Invariably I discover that they are not in a position to design anything for their employers, and have traveled on their own time and money, for the same reasons that I have come – to stay on top of computer technology.

Meanwhile, major players like Microsoft hold their own conferences covering their own new products, and there are various separate conferences on Supercomputers, FPGAs, etc., sprinkled around at various times and places. In short, the entire industry is highly compartmentalized, and thereby effectively isolated from outside innovation.

4. SACRIFICING OUR FUTURE

The argument that most of the PhD theses and other outside work presented at conferences like WORLDCOMP is sophomoric, and hence not worth studying, is one industry defense. This is the predictable result of keeping outside work in isolation. Entire development departments exist where no PhD students are being mentored, thereby sealing the last avenue of cross-pollination. With no common conferences other than WORLDCOMP, no cross-specialty interaction because of the lack of manufacturer participation in WORLDCOMP, very few PhD students, etc., there can be little if any significant technological advancement (outside of the usual technology loops) from the isolated developers in each corporation. This is the underlying problem, not the questionable quality of outside work that is kept completely isolated from the harsh realities of fabrication. Sure, keep secret the proprietary methods that are presently being used to address the technological challenges at hand, but keeping the challenges themselves secret, e.g. the nature and distribution of real-world defects, is suicidal to the entire industry.

5. CORRUPTION OF PROCESS

Much of the computing community has lost the "re" in "research", corrupting it to mean the exposition of past developments, and specifically excludes speculations as to where things could go with careful (re)direction. Indeed, an earlier version of this paper was denounced on this basis by reviewers, with comments like "there are no analyses of any of these suggestions, only proposals", "it does not fit easily into any subtopic area" and "ideas are cheap, especially recycled ideas."

As a result, the entire field of high performance computing has been sucked into a bottom-up design process, by simply dismissing top-down discussions on the basis that they are not "research". As a result, high performance computing still lacks an effective architecture, when all indications are that a top-down approach could probably have produced a good architecture a decade ago.

This obviously can not be corrected at compartmentalized conferences like the FPGA conference. Only multidisciplinary conferences like WORLDCOMP have any real hope of encouraging the top-down methods needed to merge the disparate technologies to forge the future of high performance computing.

6. WHERE WE SHOULD BE HEADING

It seems pretty obvious that each segment of the industry has some of the "missing pieces" needed by the other segments. Intel is now developing CPUs with embedded GPU-like capability, which could be greatly enhanced with some reconfigurable logic. Similarly, a processor embedded into FPGAs could orchestrate power-on fault-tolerant reconfiguration and reconfigurable logic. Just about everyone already seems to agree that all memory must be coherent. In short, it appears that each segment of the industry is proceeding toward a single common point, a computational

singularity promising a hundredfold improvement in performance over traditional methods. However, the present looping of technology is tugging at every segment to pull them away from that computational singularity. The main blockage seems to come from ignorance of various obscure enabling technologies.

7. ENABLING TECHNOLOGIES

The following synergistic enabling technologies are not additive, or even multiplicative. Projected advanced technology processors will need nearly all of these enabling technologies to function at all, because a very high level of parallelism is needed to support power-on and on-the-fly reconfiguration, and dynamic reconfiguration is needed to economically manufacture chips having so many components. Possibly the most challenging application that will ever be run on these processors will be contained in an attached ROM that is executed when power is applied, or when a fault is detected during execution, to reconfigure many faults "into the ether" leaving a fully functioning chip.

This "wall of obscure technology" presents a sort of chicken-or-egg situation that has so far blocked the construction of envisioned advanced technology processors, and hence denied the hundredfold improvement in performance that is expected from such architectures. Corporations limit their technological risks by taking risks one-at-a-time, and therefore wouldn't think of trying many new things on a single new device. These enabling technologies are not individually transformative, so they have remained obscure.

Note that taken together, these enabling technologies define a processor that is very different from present day processors, and would look a bit like "alien technology" to present day technologists. I think of this as the "rubber band effect". These methods were individually rejected because they presented no great advantage, providing an ever increasing incentive to embrace other new methods as they appear. Now, that rubber band is stretched quite tight, providing a gigantic incentive for adopting enabling technologies.

WARNING: Most of these technologies have outward characteristics that are very similar to other more common but less powerful methods. This has resulted in many experts misdismissing them during the varying number of decades since they were first proposed, thinking that they were something else. Indeed, this phenomenon has become an important driving force behind the continuation of the Itanium Effect.

7.1 Logarithmic Arithmetic

Everyone is taught in school that you can multiply and divide with logarithms, but not add and subtract. This is simply not true (see below). Pipelined logarithmic ALUs are much simpler than floating-point ALUs, as they only need 3 adders, a small ROM, and some glue. Unfortunately, they are consigned to low precision. These enable super performance for low precision applications, like image and speech recognition, neural networks, etc., and make medium granularity designs quite practical.

Note that, for the most part, future supercomputers will be working on very different problems than do present day computers. Future computing will involve AI applications centered on visual, audio, and neural network (NN) applications, all of which deal in low precision that is within the range of logarithmic arithmetic.

Adding with logarithms is easy:

1. Take the ratio of the two arguments, which since they are represented as the logarithms of numbers; you can divide by simply subtracting their logarithms.
2. Use that ratio to look up the appropriate entry in a "fudge factor" table that contains the logarithms of fudge factors. The size of this table limits the precision available with this method to less than IEEE-754 single precision. Interpolation methods extend the available precision. Carefully computed table entries avoid consistent round-off errors much like IEEE-754 does.
3. Multiply the numerator by the fudge factor, which is accomplished by adding their logarithms.

Subtraction and signed arguments are handled with obvious extensions of this simple strategy, that involve the use of two sign bits, one sign bit for the sign of the logarithm of the absolute value of the number being represented, and the other sign bit for the sign of the number being represented.

Since most of the "logic" of logarithmic arithmetic is contained in the contents of its tables, SECDED error correction logic can correct most faults, and detect the faults that it cannot correct.

Note that the practical success of power-on and on-the-fly reconfiguration depend on having a high enough flops/transistor ratio to support real-time diagnosis and reconfiguration, so the speed and simplicity of logarithmic arithmetic makes it a clear winner in these devices.

7.2 Medium Grain and Multi-Grain FPGA Architectures

Until now, everyone designing FPGAs either designed with full ALUs (coarse granularity), or with just gates (fine granularity). However, logarithmic arithmetic and "fixed point" (like integer, but the decimal point can have any pre-assigned location) arithmetic require much simpler blocks. Full IEEE-754 floating point ALUs can be chopped both horizontally (into functional units) and vertically (into digits). These blocks can be combined to achieve full pipelined result-per-clock-cycle capability as needed, though at the cost of additional flow-through time.

The pieces of a logarithmic ALU are adders and tables with attached lookup logic. The pieces of full IEEE-754 floating point ALUs are priority encoders, shift matrixes, adders, multipliers, etc., depending on the implementation. These pieces can be used for other things, e.g. binary multiplying a logarithm by an integer is actually raising the number represented by the logarithm to the integer exponent. By providing a "parts store" of both ALU pieces and full ALUs in typically needed proportions, Akin to the letter assortment in a Scrabble game, users can create "super duper operations" that use most of them, then instantly reconfigure (see Horizontal Microcoding below) them to do other different "super duper operations" as needed. The goal here is not (yet) to provide everything needed to do entire programs in a single data chained operation, but rather to simply reduce the number of times that the same data needs to be (re)handled by an order of magnitude or so.

Note that in most cases this obviates some of the need for an optimizing compiler, because there is no benefit to optimizing a

configuration, unless without optimization, there aren't enough components available to perform a complex series of computations and produce a result every clock cycle.

Multi-grain would doubtless be more complex to use than either fine or coarse grained approaches, but its use would improve the flops/transistor ratio to facilitate real-time reconfiguration.

7.3 Coherent Memory Mapping

Often, "local" memory has been attached to a particular ALU, and "global" memory has been attached to a particular bus. This isolation of local memory is needless, and associating "global" memory to a particular bus sucks performance. Multiple memory busses and interleaved memories have been used since the 1960s to exceed single-bus performance, and it takes very little logic to be able to attach local memories to global bus systems so that all memory is uniquely addressable (coherent).

Note that it is important to provide redundant busses if vary large chips are to be reliably made. The trick is to provide many redundant busses, and then use whatever works and is available. In addition to "in use" bits, they would also need "functional" bits, along with the logic to honor and operate these bits, and use whatever bus is ready to carry the traffic.

Switch fabric architecture is evolving. Note that simply organizing clusters of functional components in a 2-D fashion on the chip, and providing redundant busses for every row and column, it becomes possible for many cluster-to-cluster communications to be simultaneously taking place. As a result, traditionally slow operations like scatter and gather can run at many times traditional speeds.

However, the biggest payoff from coherent memory is in its ability to easily save the state of a subsystem by simply copying out the memory in the subsystem. Without coherent memory, partial reconfiguration would be extremely difficult.

7.4 Variable Data Chaining

The idea of pasting ALUs together in chains, as is now done in some coarse-grained FPGA designs, was called "data chaining" in early supercomputers, like the CDC Cyber 205. They simply switched ALU connections from memory to pipeline registers, in order to interconnect ALUs to get a sort of simplistic coarse-grained FPGA-like performance. Hence, perceived distinctions between coarse-grained FPGA designs and multi-ALU data chaining in supercomputers is illusory.

By simply providing pipeline registers between 2-D arranged clusters, and configuring ALU ports to connect to those pipeline registers, otherwise isolated slave processors can chain together to perform extremely complex operations.

7.5 Fast Aggregation across ALUs

One thing missing from all contemporary designs, and now presenting a major barrier to GPU advancement, is the interaction between parallel ALUs needed to perform aggregation functions like adding the elements of an array, finding the maximum value, etc., in $\log n$ time instead of n time. First every even ALU interacts with the adjacent odd ALU, and then the winners interact with other winners, etc. Graphics applications are unique in their general lack of need for aggregation, which has been the basis for GPU successes in that arena, and the basis for their lackluster performance in other areas.

Aggregation also facilitates the merging of thousands of individually performed diagnostics in thousands of slave processors, thereby speeding up real-time reconfiguration.

7.6 Blurring the SIMD/MIMD Distinction Using Small Local Program Memories

A certain amount of temporal autonomy is needed for slave processors to deal with slow operations, re-routing over busy and defective busses, memory cycles lost to other processors, waiting for slow global busses, etc. A small amount of FIFO or memory would provide the buffering needed to hold a few slave processor instructions broadcast by the central processor. However, it would take little more to implement a rudimentary instruction set in that memory, complete with conditional branch instructions, etc. With this, individual slave processers could each do their part to perform complex operations, without holding up the entire processor when they individually slow down or stumble into each other. This could provide the combined advantages of SIMD and MIMD architectures.

Note that code running in faulty configurations will doubtless run into many roadblocks. Slave processors must be able to function autonomously, in order to continue running when other slaves have died.

7.7 A Simple Horizontal Microcoding Interface for Applications

HP is believed to have been the first to make some of their horizontal microprogramming memory accessible to applications programmers. This was implemented as an option on HP 21-MX minicomputers. This way, users could define new operations that ran at hard-wired speed. In FPGA terms, this is akin to instant partial reconfiguration from memory. Instead of the usual FPGA design that uses long shift registers to hold a particular configuration, suppose that some of the configuration bits are replaced with several bits and a global mechanism of selecting which bit from each group to use. This could be a simple as having short circular shift registers controlling each potential connection, and a global mechanism of rotating all circular shift registers in unison. Loading would work as usual, only when all of the bits have been loaded for one configuration, the short circular shift registers would all be rotated by one bit and loaded with new contents for that position, with this process continuing until all of the bits in every circular shift register have been loaded. Runtime reloading, equivalent to re-defining operation codes of the computer being implemented, could be accomplished by rotating the circular shift register into position, and transferring the contents of a dedicated place in memory into the main FPGA programming input, while inhibiting changing any memory or registers on the device during reprogramming. This way, compilers could invent perfect "operation codes" that are custom made to perform the work of a page of code, and use just about everything on large devices to achieve incredible speed.

This is critical for diagnostics in preparation for reconfiguration, because it allows access to the basic "grains" of the system. Without this, really complex diagnostics would be needed that

deduce the sources of malfunctions from combinations of observed failures from various complex configurations.

7.8 Failsoft Configuration on Power-Up

Blowing fuses during manufacture to deal with faults makes no provision for in-service failures. However, (re)configuring on power-up cures in-service failures and assures a nearly limitless lifespan. This is no problem with a general purpose programmable device that is fast enough to do the job in a reasonable amount of time. However, this functionality establishes a high lower limit on the performance of future processors, as processors must be able to fully diagnose and repair numerous malfunctions on power-up within a second or so. Further, devices must incorporate appropriate technologies to support dynamic reconfiguration. This presents a chicken-or-egg challenge, as super-performance is needed for practical power-up reconfiguration, and power-up reconfiguration is a practical necessity to implement super-performance at the high levels envisioned here.

Once this has been achieved, there is **no limit** on size and complexity of future computers while maintaining ~100% yield, providing that sufficient spares are included in the design, and there is fall-back to smaller configurations in the event of an excessive number of failures. For example, a chip might have a dual main-processor where only one is needed, and, say, 4096 slave processors plus a few hundred spares, configured in a way that the system could run with 2048, or even 1024. Further, the main processors might have IEEE-754 floating-point ALUs, which could be emulated in software should the ALUs be faulty. There would be countless fuses in the power distribution network to isolate any shorted logic, etc. In short, if much of anything worked, the chip would work well enough to sell, at least into applications that didn't need their maximum potential performance. This would completely remove the present tradeoffs between chip size and yield.

Note that advanced configuration methods typically involve genetic algorithms (GA) to discover workable configurations, so the time needed to configure is variable and potentially unbounded. Hence, a large engineering margin in performance will be needed to assure timely (re)configuration.

7.9 Failsoft Partial Reconfiguration During Execution

It is also possible to survive most new faults during operation. Mostly implemented in the firmware running on a future processor, applications that run as multiple tasks that post their results when done; could continue operation, even with the failure of associated computational components. Applications would use **Assert** logic to confirm correct operation, and watchdog timers to recognize dead tasks. When a task is seen to be malfunctioning, a partial reconfiguration would be triggered, using the same (re)configuration logic used on power-up, but restricted to the subset of hardware used by the malfunctioning task.

When reconfiguration is complete, the failed task would then be restarted on the reconfigured hardware. A repeated identical failure would indicate a programming error.

Fail-soft partial reconfiguration would require suitable application program architecture, and would introduce a considerable momentary "glitch" into DSP applications. The alternative to such glitches is the present situation of permanent irreversible component failure.

To further improve the likelihood that in-service failures will be recognized and corrected, idle time should be spent running diagnostics, diagnostic failures should be used to invalidate recent results, and diagnostics should be invoked before accepting highly unusual results.

7.10 Physically Symmetrical Pinout to Facilitate the Use of Defective Components

By carefully assigning the pinout of new chip designs, it could easily become possible to plug them in any of up to four different ways, with as many separate-but-equal processors and associated I/O pins as there are ways of plugging them in. This way, there would be up to four prospective pins #1. After testing, a factory technician would then cut off the corner pin associated with each fully functional processor. A circuit board designer could then eliminate any combination of corner pin holes, depending on which processors and I/O pins were not essential to the design.

Typical circuit board designs would have 3 missing corner pin holes on the circuit board, so that a fully functioning chip could be plugged in any of 4 different ways (because all of its corner pins would be missing), However, a chip with a malfunctioning processor or I/O pin(s) could only be plugged in only one way, with the associated pin #1 connecting to the one remaining corner pin hole. Note that hexagonal chips could be plugged in any of 6 ways, and only 1/6 of the chip would be lost to a bad I/O pin. Designers would be motivated to use fewer I/O pins, because malfunctioning chips would be less expensive then ones where all of the processors and I/O pins function correctly. Designs needing less than half of the I/O pins would provide additional corner pin holes on the circuit board design, so that chips with even more bad sections could be utilized.

Another pin (perhaps pin #2) would be asserted on the circuit board to identify the "primary" section of the chip, so that the chip can tell which section is to control the others, and identify the chip's rotational position on the board, so that the functions of I/O pins can be properly determined.

Of course a designer could use all of the processors and all but two pins in every corner, but then only fully functional chips could by utilized.

Symmetrical pinout is the last line of defense for when fail-soft methods fail. This provides the factory with a method of selling devices that cannot fully repair themselves, and provides service personnel with a method for repairing equipment incorporating devices that can no longer fully repair themselves. Repair personnel would simply remove a malfunctioning device, rotate it, and reinstall it.

7.11 An Architecture-Independent Universal Compiler To Compile A New APL-Level Language

There are several candidate compiler architectures that could compile to almost any imaginable computer. Methods range from rewriting the code-generator portion, to rewriting table entries containing snippets that perform various functions, to using a syntax directed meta-compiler and simply altering the output

instructions. All of these methods and more have been used in commercial production compilers, but complicating factors, like the desire for code optimization, muddy the waters. Note that there are orders of magnitude to be gained from a radically new compiler, yet optimization typically only buys <2:1. Hence, for the time being, until architectures make their radical jump and then settle down, optimization concerns should be set aside in favor of getting the technology off of its present hump.

A major complication is the need for a new source language with semantics akin to APL in which to program high performance applications. A truly flexible and capable compiler for compiling from an APL-level language to variable targets is desperately needed for the high-performance industry to progress. This can only come as an industry-wide effort, or at government expense.

If a new language is being designed, it should probably also have ADA-like variable declarations to facilitate both compile-time and run-time checking, and support a COBOL-like verbose listing mode to facilitate "desk checking".

7.12 Putting it All Together

It seems clear that it is possible to build self-repairing processors of nearly limitless size and performance, and manufacture them with ~100% yield, with reasonable extensions to the technology at hand. However, its architecture won't be a CPU, GPU or FPGA. Instead, it will be a combination of all of these and a little more. It could run languages like C++ just as inefficiently as do present products, but would need a language with APL-like semantics to support future capabilities. Further, there are some big money hurdles to overcome. Probably the biggest hurdle is the need for a universal compiler to support future high-performance computing.

Is a hundredfold improvement in performance worth the ~billion dollar cost to engineer such devices? That is **the** question that should logically determine whether this path is taken.

Note the prior call for technological revolution of computation, made 15 years ago by Andŕe DeHon in his PhD thesis *Reconfigurable Architectures for General-Purpose Computing*. His technology was eventually commercialized into a company called Silicon Spice, which was sold to Broadcom at the height of the tech boom for ~$1.2 billion. Technology of that time restricted practical implementation to DSP applications, whereas no such limitations now persist. However, with more complex methods supporting loftier goals, including general purpose computing, the cost will be considerably higher.

8. BEFUDDLEMENT

If you ask others why they are in their particular corner of their industry, or you ask yourself why you are in your particular corner of the industry, the answer usually boils down to befuddlement with the unknown worlds outside of a particular domain. Nowhere else is this more clear than with memory designers, who usually want absolutely nothing to do with the internal complexities of computer architecture, operating systems, etc. There is a similar though lesser effect with FPGA designers. At the opposite end of the spectrum are the operating systems and applications designers, few of whom have any experience with an oscilloscope or logic analyzer. Just about everyone sees the industry as way too complex to think about in any sort of gestalt way. However, with the industry chopped up into various fixed domains that are gradually becoming less and less appropriate for future computational needs, the gaps in other-domain capability are being filled in with extreme in-domain complexity. In this mode, we all become part of the problem, rather than part of the solution.

There was a similar situation during the 1960s, when radically new computers were being introduced on a yearly basis. Many excellent technologists simply dropped out of this scramble to keep up with technology. I anticipate the same phenomenon during a transition to the envisioned advanced technology processors, where few present technologists will be able to keep up with the coming radical changes.

9. COMPUTATIONAL SINGULARITY

Here are three differently stated but identical architectures, stated from three different points of view:

1. An array processor whose slave processors are much more powerful than in the past, including reconfigurable logic, so that on-the-fly array processing instruction definition becomes practical.
2. A multi-grain FPGA with ALUs and parts of ALUs, memory banks, etc., organized into reconfigurable clusters under the control of central processor(s).
3. GPUs with reconfigurable logic and integrated into a CPU.

Each of these descriptions presumes the application of the necessary enabling technologies as explained earlier.

So, where does the hundredfold improvement in performance come from?

1. Future projected applications won't need the precision of present devices, so precision can be traded for speed, e.g. more logarithmic ALUs in place of fewer floating-point ALUs. Also, this speed will facilitate fast power-on diagnostics.
2. With the methods presented, much larger chips can be fabricated at the same overall cost, because defects will no longer impair the yield. Greater size brings the components needed to do more in parallel, along with the assortment of components needed to implement long computational pipelines.

10. CONCLUSION FOR DEVELOPERS

Just because your co-workers remain isolated is no reason for you to do the same. Attend WORLDCOMP and other conferences that are out of your present scope of work. Present papers that discuss your vision for the distant future of your present scope of work. Make it known at local universities that you will mentor PhD students. Make friends who are familiar with potentially important future technologies, e.g.:

1. Someone who worked on 1980s supercomputers
2. Someone who has extensive computing experience that predates microcomputers.
3. A CS/EE professor who is up on just about everything that has ever been made.

Soon, you will become known as THE guy who knows just about everything. You will soon be able to move into management, whereupon…

11. CONCLUSION FOR MANAGEMENT

The issues presented in this paper all center around what would have been called "professionalism" in the era before microcomputers. Professionals would represent their employers at multidisciplinary conferences, present papers at conferences; stay up on all potential enabling technologies, mentor students, etc. This sense of professionalism has been completely lost in "modern" technology, and with it has gone the productivity of the designers at the major chip makers. Everyone is now threatened by this lack of a sense of professionalism. Thirty years ago I might have recommended simply firing such designers for cause and hiring more professional designers, but that time has passed and we must now "dig our way out" of this mess, using the people that we now have working in the industry.

Rather than taking years/decades to do so in any slow way, my present recommendation is to draw up clear company standards of professionalism that ALL product designers must follow, and demote or fire those who don't follow them. Sure you may have to make an example of one or two technically capable developers, but this is necessary to demonstrate your resolve. Soon everyone will be attending conferences other than just the "inside" conferences that only pertain to their present narrow work. Everyone will mentor PhD students, will be presenting papers inviting public scrutiny before committing designs to silicon, will bring in both very new and very old enabling technologies, etc.

12. OVERALL CONCLUSION

This paper advances an approach to achieve a renaissance in computational architecture, promising a hundredfold increase in performance. However, the big challenge here is organizational rather than architectural. Existing microcomputer, GPU, and FPGA manufacturers have been unable/unwilling to adapt to produce these products, or this would have happened a decade or more ago. This appears to require revolutionary management changes, presenting an excellent opportunity for takeover bids by astute investors who are not attached to present methods of engineering management.

13. REFERENCES

[1] Strenski, D., Sundararajan, P., Wittig, R. 2010. The Expanding Floating-Point Performance Gap Between FPGAs and Microprocessors. *HPC wire*. November 22. http://www.hpcwire.com/features/The-Expanding-Floating-Point-Performance-Gap-Between-FPGAs-and-Microprocessors-109982029.html?viewAll=y

[2] http://www.computerhistory.org/collections/ibmstretch Computer History Museum's collection of IBM Project STRETCH materials.

[3] http://www.world-academy-of-science.org/ *WORLDCOMP* web site

[4] http://www.fundinguniverse.com/company-histories/TANDEM-COMPUTERS-INC-Company-History.html

[5] http://en.wikipedia.org/wiki/Floating_Point_Systems

[6] DeHon, Andŕe. 1996. *Reconfigurable Architectures for General-Purpose Computing*. A.I. Technical Report No. 1586. http://www.seas.upenn.edu/~andre/pdf/aitr1586.pdf

[7] Pickett, L. 1993. *U.S. Patent 5,184,317*. A discussion of advanced logarithmic arithmetic methods.

[8] Richfield, S. 1987, A Logarithmic Vector Processor for Neural Net Applications. *Proceedings of the IEEE First Annual International Conference on Neural Networks*. IEEE Catalog #87TH0191-7.

[9] Buchholz, Werner. 1962. *Planning a computer system: Project Stretch*. McGraw-Hill, Inc. Hightstown, NJ, USA. ISBN:B0000CLCYO

[10] Ashenhurt, R. L., Metropolis, N. 1959. Unnormalized Floating Point Arithmetic**.** *Journal of the ACM (JACM)* Volume 6, Issue 3. Pp 415-428. ISSN:0004-5411

Analysis of 32-bit Fault Tolerant ALU Methods

A. Vahid Khorasani[1], **B. Bijan Vousoghi Vahdat**[2], **and C. Mohammad Mortazavi**[2]
[1]Department of Electrical Engineering, Sharif University of Technology, International campus, Tehran, Iran
[2]Department of Electrical Engineering, Sharif University of Technology, International campuse, Kish, Iran

Abstract - *`This paper presents a BCH based hardware implementation of 32-bit Fault-tolerant ALU in which is compared with the current techniques such as Residue code, Triple Modular Redundancy (TMR) with single voting and TMR with triplicated voter that are widely used in space application to mitigate the upsets, in terms of area penalty. We consider BCH (Bose, Chaudhuri, and Hocquenghem) codec (encoder, decoder) that is implemented FPGA hardware. The new implementation of ALU employing BCH code on Spartan-3 FPGA has been provided. The results show reduced area requirements compared to the other technique and it can correct any 5-bit error in any positions of 32-bits input registers of ALU*

Keywords: Fault Tolerant, BCH codes, ALU, Residue code, TMR, Encoder, Decoder, FPGA.

1 Introduction

In recent years, working on reliable communication transmission plays a vital role. So that a single error may shutdown the whole system and give rise to erroneous data. System reliability is one of major issues in embedded processors designs for space application such as satellite, military, communications and etc. Various attacks exist in space on integrated circuits that comes from sun activity. Such as solar rays which are composed of charged particles. The radiation from sun effects in integrated circuits make digital damage and upsets such as SEU (Single Event Upset), SET (Single Event Transient) and etc as presented in [1]. Such attacks can upset either combinational logic or sequential logic. In other words a bit flip can occur in register bits and if one bit of main storage system is changed the mission of system would be completely different. In such scenario the error control or fault tolerant methods are employed to keep integrated circuits against these attacks in space. To achieve such purpose we consider error detection and correction codes (EDAC) method. It is usually used to mitigate SEU in integrated circuits which are required that the encoder and the decoder blocks to be able to detect and correct errors respectively. This technique gives strong faults coverage and less overhead hardware. For this reason we consider the BCH (Bose, chaudhuri, and Hocquenghem) codes and a binary BCH codes is considered. As a result of using BCH codes, we have achieved to design encoder and decoder circuits to detect and correct 5-bit faults. Also we have designed a 32-bit ALU. Our 32-bit ALU model consists of the following function units: Arithmetic operation consist of Full-Adder and subtractor. Bitwise logic operation such as: XOR, AND, OR, and NOT. Bit-shifting operations such as: shifting to the left or right. We previously have presented design of a secure ALU (Arithmetic and Logic Unit) against faults. This ALU is able to correct any 5-bit error in any position of its 32 bits input registers. Consequently, the core for implementation of an ALU employing the BCH code on Spartan-3 FPGA has been provided.

1.1 Motivation

Our goal is to designing a new 32-bit ALU that is secure against many attacks or faults and able to correct any 5-bit fault in any position of its 32 bits input register of ALU. Because the radiation effects on electronic circuits may cause to be inverted data bits of registers or memories. If one bit of main storage system is changed the mission of system would be completely different. The high motivation in choice of BCH codes is that, it is able to correct multiple errors and these classes of codes are kind of powerful random error correcting cyclic codes. Moreover, by choosing two essential parameters n and t, the designer is able to design any BCH code. These implies that a crucial motivation because the structure of BCH encoder and decoder in presence of two parameters n and t can be notably different.

1.2 Overview

The rest of this paper is presented as follows: In section 2, types of fault tolerant techniques are discussed. In section 3, an encoding and decoding implementation of (63, 36) BCH code applied in our work are described. In section 4, our fault tolerant 32-bit ALU are presented. In section 5, the results of fault tolerant methods are presented. Section 6, describes conclusion.

2 Types of fault tolerant technique

There are many design-based techniques to give the fault tolerant scheme such as detection techniques and mitigation techniques. Detection techniques consist of hardware redundancy, time redundancy and information redundancy scheme and mitigation techniques consist of TMR (triple modular redundancy), multiple redundancy with voting as presented in [3], and EDAC (Error Detection and Correction coding) scheme.

Hardware redundancy (HR) is based on extra component that can be composed of extra same circuit to perform the same operation at the same time. In this way the faults can be

identified by duplication or masked by triplication and then by comparing the outputs through a voter. Error detection and correction codes can also be a type of hardware redundancy due to the requirements of extra bits enabling to detect and correct errors. So, consume 100% hardware overhead however it gives powerfully high fault coverage. The proposed technique can identify an SEU in a sequential logic.

The time redundancy (TR) consists of extra execution time with sampling at different time then comparing results to detect errors. These techniques are usually utilized to detect a SET in the combinational logic and also it requires lower hardware overhead than the HR and EDC. However this method has a major drawback due to the extra delay during recompilation.

The information redundancy (IR) is based on EDC (Error Detection Code). Although this technique requires extra hardware, it gives a low hardware overhead than the HR. This method can be considered as a good choice to meet our criteria.

The TMR (triple modular redundancy) technique uses three components. In fact, the entire device triplicate to fault masking by a single voter then the voter gives the correct value in the presence of an upset by the majority opinion as an output. Similar to previous techniques, IT needs extra logic. It corrects up to three faults per 3-bit word, if each fault is placed in a distinct bit. This technique votes the true value but it doesn't correct it. In addition faults will accumulate, if there is no additional device to correct them. Hence this approach is also more applicable to space applications.

Error detection and correction (EDAC) **codes** technique is usually utilized to mitigate SEU in integrated circuit and it requires extra hardware. Nevertheless this technique gives strongly faults coverage. Hence, we are considering this technique to cover our goals. EDAC codes can be implemented in two ways and it depends on transmission data.

If transmission occurs in only one direction to overcome error, the error control system works out through the forward error correction (FEC) [7].

Therefore, FEC method is useful for a satellite transmission and it is considerable for us. The schematic of an FEC method is shown in figure 1. One of the significant classes of FEC codes is linear block codes and binary BCH codes are classes of linear block codes. BCH codes are a generalization of Hamming code for a multiple error correcting. Further, these codes are a large class of powerful random-error correcting cyclic codes [6].

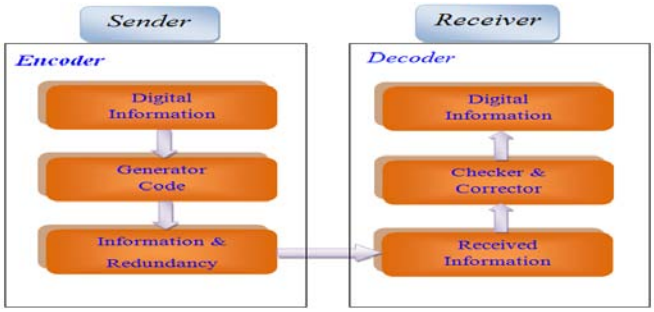


Figure 1. Scheme of a forward error correction (FEC).

2.1 Introduction of Fault Tolerant methods

In this section the similar works in the literature that has been applied in the ALU to keep it safe from logic and arithmetic operation using Residue code are presented [4]. The other works which are considered here are triple modular redundancy (TMR) scheme with single voting [3], and with triplicated voting. Then we have compared these methods with our tasks. Finally, Fault Tolerant methods for 32-bit ALU in terms of overhead hardware are compared.

2.1.1 Residue code

To design the 32-bit Fault Tolerant ALU (Arithmetic Logic Unit), V.S. Veeravalli et al [4], has used the residue code and duplication of hardware mechanism in order to achieve a less hardware overhead. In residue codes, data parts and check parts are separated to be able to detect the errors. The residue of X modulo A is denoted by $|x|_A$. There exists the following equation.

$$|X+Y|_A = \big||X|_A+|Y|_A\big|_A$$
$$|X.Y|_A = \big||X|_A.|Y|_A\big|_A \qquad (1)$$

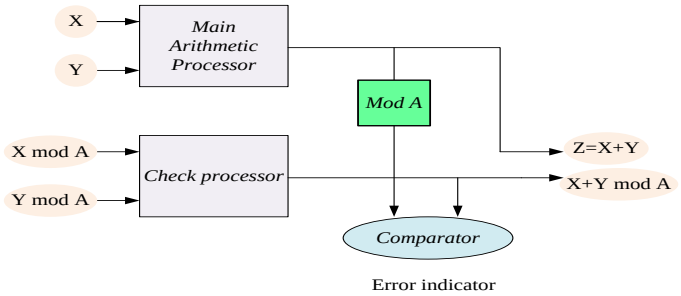


Figure 2. Arithmetic processor with residue checker.

The structure of such scheme is depicted in Fig. 2. V.S.Veeravalli et al in [4] has been presented implementation of error detection mechanism to detect error in ALU. Also for Boolean operations of the ALU duplication of hardware has been used to detect the error. Since the residue code can only detect the error, they have to devise other spare ALU to make error correction possible. In such technique, if one error has occurred in ALU, then it should replace the original ALU with the spare ALU. In such method it is required to compute the remainder of both inputs of the ALU. Also they had to compute the remainder of the output of the ALU. Consequently, after computing the remainders it is required to compare whether they match. The main drawback of residue code with A as a check modulus is that it has the same undetectable fault magnitudes. For A=3, only errors that modify the result by such multiple of 3 will go undetected and therefore single bit faults are always detectable [5]. The hardware overhead composed by residue codes concerning the ALU is around 45.596%. The hardware overhead of duplication for Boolean unit is 3%. So, the extra ALU has been increased the hardware overhead by 100%. Therefore the total hardware overhead for the 32-bit ALU is 148.596%. The other main drawback for duplication of hardware mechanism is that it uses two copies of the same hardware. It has more

than 100% hardware overhead. In such procedure the input is processed in both modules and is compared with the output results. If there is a mismatch or an error in the circuit, it will be informed by the comparator; moreover such scheme can only detect errors and cannot correct errors.

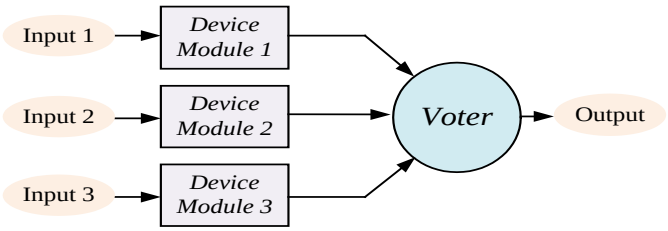


Figure 3. Triple Modular Redundancy scheme.

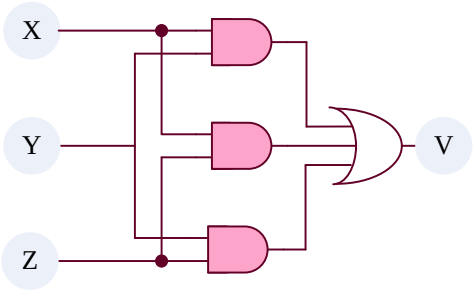
2.1.2 Triple modular redundancy with single voting

The other work in such field is designing a 32-bit ALU using Triple Modular Redundancy (TMR) with single voting. In fact, it uses hardware redundancy technique in the combinational logic and allows voting the correct output value in the presence of faults. The majority voter scheme is depicted in Fig. 3.

The concept of TMR is devised firstly by Von Neumann. The circle Voter in Fig. 4 is called a majority organ by Von Neuman. In TMR technique the logic is triplicated in the output, the voters identify the correct value. The entire scheme is shown in Fig. 4.

In this technique all registers should be tripled to protect circuits against radiation effects. The voter must be added in the output. The error will not be reflected in the output of voter if, one component fails. The hardware overhead in TMR is the addition of the two registers of the same size.

Moreover there exist n voters for each n-bit register. So, such method is also need extra two spare 32-bit ALU and it is give rise to 200% hardware. Hentschke et al [2], have shown that Triple modular redundancy scheme is more effective according to area and performance to preserve registers and small memory structure. Nevertheless Hamming code is more suitable to preserve large register files and memories.



X	Y	Z	V
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Figure 4. Majority Voter Schematic.

2.1.3 Triple modular redundancy with triplicated voting

The other mechanism to fault tolerant 32-bit ALU is called Triple modular redundancy with tripled voting. Since in such scheme the reliability of the voter circuit proportionately is increased, it is currently utilized in industry.the schematic of this mechanism are shown in fig.5.

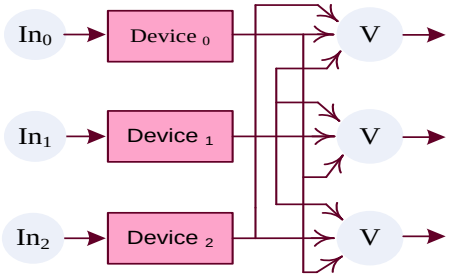


Figure 5. Triplicated Modular Redundancy with Tripled Voting.

3 Proposed Fault Tolerant methods based on (63, 36) BCH code

In this procedure we have designed (63,36) BCH encoder circuit to find the codeword and we have designed (63,36) BCH decoder to correct any faults in codeword. The designing of (63, 36) BCH encoder and decoder for t=5 has been provided in [8] by the way we describe encoder and decoder implementation summarily.

The encoder circuit [4], calculates the parity bits using the LFSR (Linear Feedback Shift Register). The generator polynomial of the (63, 36) BCH code is

$$G(X) = X^{27} + X^{22} + X^{21} + X^{19} + X^{18} + X^{17} + X^{15} + X^{8} + X^{4} + X^{1} + 1.$$

Encoding circuits are shown in fig.6.For more information refer to [8].

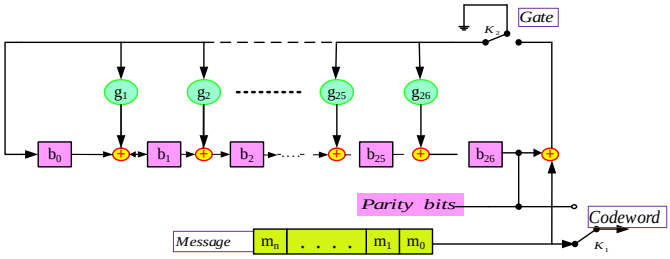


Figure 6. Encoding circuit for an (63, 36) BCH code.

The decoding of BCH code is composed of three main steps that are expressed as follows:

1) Compute the syndromes from the received codeword.
2) Obtain the error locator polynomial σ(x) (ELP) through the BMA (Berlekamp-Massy-Algorithm).
3) Determine the error-location numbers by finding the roots of error location polynomial (identifying the position of erroneous bit). All these steps shown in the following block diagram in figure 7.

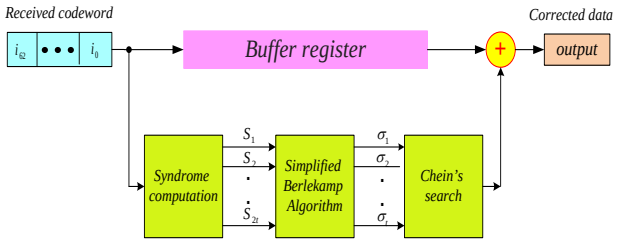


Figure 7. Block diagram for decoder system using BCH code.

3.1 The syndrome computations

The first step of decoding procedure is "syndrome computation". The syndromes identify whether error has occurred. If the syndromes all are zero, we will have no error in codeword and if the syndromes not be zero we will have error in codeword. For computing the syndromes, the syndrome S_i is defined as:

$$S_i = r(\alpha^i) = r_{n-1}\alpha^{(n-1)i} + r_{n-2}\alpha^{(n-2)i} + \ldots + r_1\alpha^i + r_0 \qquad (2)$$
$$= (\ldots((r_{n-1}\alpha^i + r_{n-1})\alpha^i + r_{n-3})\alpha^i + \ldots + r_1)\alpha^i + r_0$$

Where i is $1 \le i \le 2t$. Each syndrome component is calculated by dividing r(x) by the minimal polynomial $m_i(x)$ of α^i

$$r(x) = q_i(x)\, m_i(x) + b_i(x). \qquad (3)$$

$b_i(x)$ is the reminder. When the entire received codeword has entered the decoder, 10 syndromes components (s_1, s_2,…, s_{10}) are formed. It takes 63 clock cycles to complete the computation. Since, the generator polynomial is a product of at most 5 minimal polynomials Therefore at most 5 feedback shift register , each consist of at most 6 stages, are required to form the 10 syndrome components. For more information about the computation of syndrome refer to [8]. Also We have implemented the syndrome computation circuit for (63, 36,5) BCH code which has been presented in [8].

3.2 The Berlekamp-Massy Algorithm

The second step of decoding for finding the error location polynomial has been done through the simplified Berlekamp-Massy Algorithm which is shown in fig 8. We assume that the numbers of errors $\le t$ have occurred and error locator polynomial σ(x) is:

$$\sigma(x) = \sigma_0 + \sigma_1 x^1 + \sigma_2 x^1 + \ldots + \sigma_\nu x^\nu$$
$$\sigma(x) = (1+\beta_1 x)(1+\beta_2 x)\ldots(1+\beta_\nu x) \quad (4)$$

The coefficient of error locator polynomial and the error location numbers are related by the following set of equations: [7].

$$\begin{aligned} \sigma_0 &= 1 \\ \sigma_1 &= \beta_1 + \beta_2 + \ldots + \beta_\nu \\ \sigma_2 &= \beta_1\beta_2 + \beta_2\beta_3 + \ldots + \beta_{\nu-1}\beta_\nu \\ &\cdot \\ \sigma_\nu &= \beta_1\beta_2 + \ldots + \beta_\nu . \end{aligned} \qquad (5)$$

Where the coefficient of error locator polynomial σ_i , $1 \le i \le \nu$ are related to the syndrome components S_i , $1 \le j \le \nu+1$ [7].

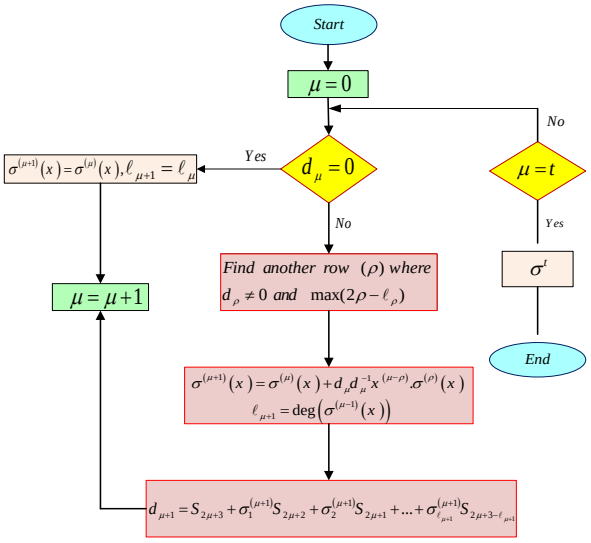


Figure 8. The inversion Berlekamp massy algorithm.

3.3 Chein's searching circuit

This process stored the received codeword in a buffer register to compute the syndrome. It takes 63 clock cycles to complete the computation. We have implemented a Chein's searching circuit [3], for the 5-error correcting (63, 36) BCH has been implemented in [8]. This circuit is applied to identify the position of erroneous bits into the 63-bit received codeword and then correct it.

4 Fault tolerant ALU using BCH code

Our 32-bit ALU model consist of the following operations Full Adder, Subtractor XOR, AND, OR, NOT, shifting to the left and right. The codec circuits are applied to correct any

5-bit error occurred in any position of 32-bits input registers of ALU. Our algorithm of fault tolerant ALU is shown in Fig. 9. In this algorithm at first the 63 bits input register A and B is read out one by one, by the decoding system. Registers A, B consist of 27-bits parity check; 36-bits data which are 4-bits are extra. We use these extra bits as parity check bits as shown in fig.10. If any error occurs in any position of 63-bits the decoder will correct the erroneous bit at once. In output the decoding system gives 36-bits where we need only 32-bits as 2-inputs of ALU (we don't store 4-bits of 36-bits). So, the output of 32-bit ALU is the C resister. We add the 4-bits zero for leftmost of C register to convert the 36-bits for the input of encoder system. After encoding the data of register C´ is kept in register A.

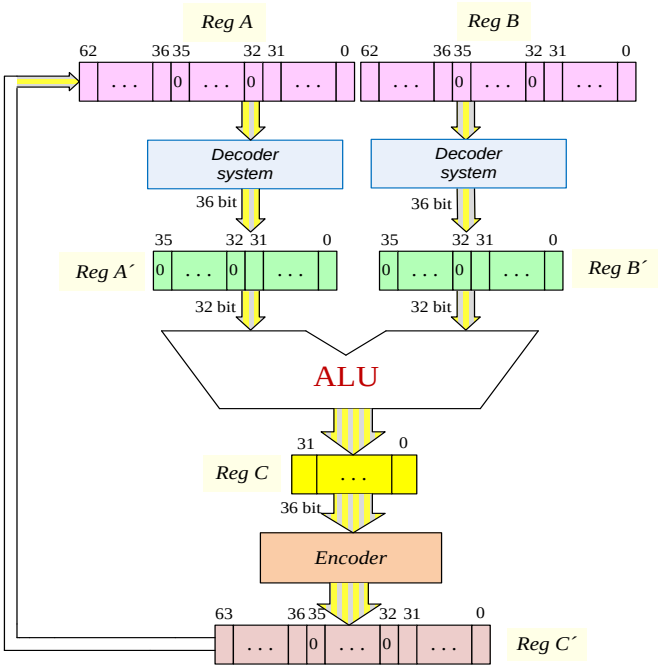


Figure 9. The fault tolerant ALU Algorithm using (63,36) BCH code.

5 Results

As shown in fig. 9, our proposed fault-tolerant ALU has been implmented in XC3S400 from Spartan-3 FPGA family. The system has been simulated on Modelsim 6.2b and its performance has been verified by ISE 8.2i that is shown in figure 10. In the comparison of the previous methods with using the BCH code and TMR, the result implies that the performance of the proposed fault tolerant ALU algorithm, an encoding and decoding block have partially degraded. This is due to the delay of the number of XOR gates in serial form in codec circuits. Moreover, as the number of error bits increases, the time for error correction may take longer. In the TMR method the delay is occured in the voter scheme, then the performance is not deeply affected and it is constant with the number of bits to be detected. In comparison with area penalty of methods, 32-bit fault tolerant ALU using BCH code is a better choice in terms of area as compared to TMR and Residue code. This is due to the fault tolerant method for 32-bit ALU using TMR with single or triplicated voting need single voting scheme or tripled voter and two extra 32-bit ALU which has been increased the hardware overhead by 202% and 208% respectively. In comparison with fault tolerant method using Residue code, we need Hardware duplication for boolean operations, residue coedes for arithmetic operation and extra 32-bit ALU which has been increased the hardware overhead by 148.9%. However, in comparison with fault tolerant method using BCH code, we need encoding and decoding block then the hardware overhead is 75%. Thus our fault tolerant hardware overhead has lower hadware compared to the others.

ALU Project Status

Project File:	ALU.ise	Current State:	Programming File Generated
Module Name:	BIT_ALU	• Errors:	No Errors
Target Device:	xc3s400-5pq208	• Warnings:	4 Warnings
Product Version:	ISE 8.2i	• Updated:	Tue Jan 11 19:27:04 2011

ALU Partition Summary

No partition information was found.

Device Utilization Summary

Logic Utilization	Used	Available	Utilization	Note(s)
Number of 4 input LUTs	285	7,168	3%	
Logic Distribution				
Number of occupied Slices	145	3,584	4%	
Number of Slices containing only related logic	145	145	100%	
Number of Slices containing unrelated logic	0	145	0%	
Total Number of 4 input LUTs	285	7,168	3%	
Number of bonded IOBs	100	141	70%	
IOB Latches	32			
Number of GCLKs	1	8	12%	
Total equivalent gate count for design	2,068			
Additional JTAG gate count for IOBs	4,800			

Performance Summary

Final Timing Score:	0	Pinout Data:	Pinout Report
Routing Results:	All Signals Completely Routed	Clock Data:	Clock Report
Timing Constraints:	All Constraints Met		

Figure 10. The Design performance of Fault Tolerant ALU

ACKNOLEGMENT

The financial support of the Iran Telecommunication Research Center is gratefully acknowledged.

6 Conclusions

The paper describes a new implementation of the ALU for BCH code .We also compared our 32-bit fault tolerant ALU by using a (63, 36) BCH code with the other Fault tolerant methods (Residue code, TMR with single voting scheme and TMR with triplicated voting method). In the comparisons, for instance, fault tolerant method using BCH shows 75% hardware overhead. The implementation provides a high level of fault tolerance with relatively and small hardware penalty. Further, the proposed system has been simulated on Modelsim 6.2b and its performance has been verified by ISE 8.2i. The results indicate any five bits error in any position of 32-bit input registers of ALU would be corrected.

7 References

[1] S.Bourdarie and M.Xapsos, senior member, IEEE, "The near earth space radiation environment", IEEE Transaction on Nuclear Science, August, 2008.

[2] R.Hentschke, F.Marques, F.Lima, L.Carro et al. "Analyzing area and performance penalty of protecting different digital modules with Hammnig code and Triple Modular redundancy". IEEE International Conference on Integrated Circuits and Systems Design, 2002.

[3] Fernanda Lima Kastensmidt, L.Carro, R.Reis, "Fault tolerant techniques for SRAM-based FPGA" June, 2006.

[4] Veeravalli, V.S. "Fault tolerant Arithmetic and Logic Unit", IEEE international conference, Rutgers State Univ. of New Jersey, Piscataway, NJ, USA, March, 2009.

[5] Israel Koren and C.Mani K rishna. "Fault-Tolerant System".Morgan Kaufmann Publishers 2007.

[6] W.W. Peterson, "Encoding and error-correction procedures for the Bose-Chaudhuri Codes", IRE Trans. Inf. Theory, IT-6, pp. 459-470, September 1960.

[7] Lin, Shu, and Daniel J. Costello, Jr., "*Error Control Coding: Fundamentals and Applications*", Englewood Cliffs, NJ, Prentice-Hall, 1983.

[8] Vahid Khorasani, B.Vousoghi et al. " Designing a secure 32-bit ALU using (63, 36) BCH code", Worldcomp conference, July, 2011.

SESSION

HPC AND DESIGN ISSUES + FPGA + GPU + NOC + EMBEDDED SYSTEMS

Chair(s)

TBA

High-Performance and Area-Efficient Hardware Design for Radix-2^k Montgomery Multipliers

Liang Zhou, Miaoqing Huang, Scott C. Smith
University of Arkansas, Fayetteville, Arkansas 72701, USA

Abstract— *Montgomery multiplication is one of the fundamental operations used in cryptographic systems. The now-classic hardware architecture for implementing Multiple-Word Radix-2 Montgomery Multiplication (MWR2MM) was proposed by Tenca and Koç in CHES 1999. This architecture performs a single Montgomery multiplication in approximately $2n$ clock cycles, where n is the size of operands in bits. In this paper we propose an improved design that is capable of carrying out the same computation in n clock cycles using equivalent amount of hardware resource at higher frequency rate. The improved design first assumes the most significant bit of the previous word to be zero and adds its real value later when it becomes available. This design is particularly desirable when dealing with high-radix Montgomery multiplications. Experimental results show that the proposed improved design can reduce hardware resource utilization by up to 60% compared with other previous architectures.*

Keywords: Montgomery Multiplication, MWR2^kMM Algorithm, Hardware Optimization

1. Introduction

Since the introduction of the RSA algorithm in 1978, high-speed and space-efficient hardware architectures for modular multiplication have been a subject of constant interest for more than 30 years. During this period, one of the most useful advances came with the introduction of Montgomery multiplication algorithm due to Peter L. Montgomery [1]. Montgomery multiplication is the basic operation of the modular exponentiation, which is required in the RSA public-key cryptosystem. It is also used in Elliptic Curve Cryptosystems, and several methods of factoring, such as ECM, p-1, and Pollard's "rho" method, as well as in many other cryptographic and cryptanalytic transformations.

At CHES 1999, Tenca and Koç introduced a word-based algorithm for Montgomery multiplication, called Multiple-Word Radix-2 Montgomery Multiplication (MWR2MM), as well as a scalable hardware architecture capable of executing this algorithm [2], [3]. There is a 2-clock-cycle latency between the computation of two consecutive iterations due to the right shift of the intermediate result. This latency brings the overall computation time to approximately $2n$ clock cycles where n is the number of bits of the operands. Two recent architectures, proposed by Harris *et al.* [4] and Huang *et al.* [5], [6] respectively, are capable of reducing this latency to 1 clock cycle. Huang's architecture outperforms Harris' for radix-2 multiplication; however, it suffers from demanding resource requirement for high-radix cases. In this paper, we propose two optimizations applied to Huang's architecture. The first one is to reduce the resource requirement, particularly for high-radix cases. It is demonstrated that the computation redundancy of the processing element (PE) in Huang's original architecture can be removed by calculating only one version of a word with the assumption that all of the unknown bits are zeros in current clock cycle, and adding their actual values in the next clock cycle. The second one is to improve the performance when the architecture is used to process a stream of operands. It is shown that carefully designed gating logic can get rid of the dedicated reset clock cycle and make every PE always process data without stall. As a result, the maximum normalized throughput of the architecture is achieved, which is $\frac{1}{n/k}$ operands per clock cycle for radix-2^k cases. These two improvements render a more practical architecture on top of Huang's by reducing the resource utilization and increasing the throughput.

The remainder of the text is organized as follows. Section 2 describes the Montgomery multiplication, the MWR2MM algorithm, and the related work. The optimization design is presented in Section 3, followed by implementation results in Section 4. Section 5 concludes this work.

2. Montgomery Multiplication

2.1 Multiple-Word Radix-2 Montgomery Multiplication Algorithm

In many cryptosystems, such as RSA, computing $X \cdot Y \pmod M$, in which $M > 0$ is an odd integer, is a crucial operation. The reduction of $X \cdot Y \pmod M$ is a more time-consuming step than the multiplication $X \cdot Y$ without reduction. In [1], Montgomery introduced a method for calculating products $\pmod M$ without the costly reduction $\pmod M$, since then known as Montgomery multiplication. Montgomery multiplication of X and Y $\pmod M$, denoted by $MP(X, Y, M)$, is defined as $X \cdot Y \cdot 2^{-n} \pmod M$ for some fixed integer n.

Since n is generally quite large in cryptosystems, such as 1024 or 2048, the direct implementation of Montgomery multiplication in hardware is impractical. In [2], [3], Tenca

Algorithm 1: Multiple-Word Radix-2 Montgomery Multiplication Algorithm [2]

Input: odd $M, n = \lfloor \log_2 M \rfloor + 1$, word width w, $e = \lceil \frac{n+1}{w} \rceil$, $X = \sum_{i=0}^{n-1} x_i \cdot 2^i$, $Y = \sum_{j=0}^{e-1} Y^{(j)} \cdot 2^{w \cdot j}$, $M = \sum_{j=0}^{e-1} M^{(j)} \cdot 2^{w \cdot j}$, with $0 \leq X, Y < M$

Output: $Z = \sum_{j=0}^{e-1} S^{(j)} \cdot 2^{w \cdot j} = MP(X, Y, M) \equiv X \cdot Y \cdot 2^{-n} \pmod{M}$, $0 \leq Z < 2M$

1.1 $S = 0$; /*initialize all words of S*/
1.2 **for** $i = 0$ **to** $n - 1$ **do**
1.3 $q_i = (x_i \cdot Y_0^{(0)}) \oplus S_0^{(0)}$;
1.4 $(C^{(1)}, S^{(0)}) = x_i \cdot Y^{(0)} + q_i \cdot M^{(0)} + S^{(0)}$;
1.5 **for** $j = 1$ **to** e **step** 1 **do**
1.6 $(C^{(j+1)}, S^{(j)}) = C^{(j)} + x_i \cdot Y^{(j)} + q_i \cdot M^{(j)} + S^{(j)}$;
1.7 $S^{(j-1)} = (S_0^{(j)}, S_{w-1..1}^{(j-1)})$;
1.8 $S^{(e)} = 0$;
1.9 **return** $Z = S$;

and Koç proposed a Multiple-Word Radix-2 Montgomery Multiplication Algorithm (MWR2MM), as shown in Alg. 1. In Alg. 1, the operand Y (multiplicand) is scanned word-by-word, and the operand X is scanned bit-by-bit. The operand width is n bits, and the word width is w bits. $e = \lceil \frac{n+1}{w} \rceil$ words are required to store S since its range is $[0, 2M - 1]$. The original M and Y are extended by one extra bit of '0' as the most significant bit. Presented as vectors, $M = (0, M^{(e-1)}, \ldots, M^{(1)}, M^{(0)})$, $Y = (0, Y^{(e-1)}, \ldots, Y^{(1)}, Y^{(0)})$, $S = (0, S^{(e-1)}, \ldots, S^{(1)}, S^{(0)})$, and $X = (x_{n-1}, \ldots, x_1, x_0)$. The data dependency graph of the proposed scalable architecture [2], [3] is shown in Fig. 1, in which each column represents the computation of a whole S for one iteration. The computation of a word is represented as a circle. Once a word $S^{(j)}$ is updated, its least significant bit is concatenated with the $w - 1$ most significant bits of $S^{(j-1)}$, which becomes the "new" $S^{(j-1)}$ and is forwarded to the neighbor PE. The concatenation is shown in Line 1.7 of Alg. 1. The forwarding is illustrated as the arrows between two columns in Fig. 1. Due to the 2-clock-cycle latency between two consecutive columns, the whole computation process takes approximately $2n$ clock cycles when performance is optimized.

2.2 Related Work

Several follow-up designs based on the MWR2MM algorithm have been proposed in order to reduce the computation time [4]–[17]. In [7], a high-radix word-based Montgomery algorithm (MWR2^kMM) was proposed using Booth encoding technique. Although the number of scanning steps was reduced, the complexity of the control and computational logic increased substantially at the same time. In [4], Harris *et al.* implemented the MWR2MM algorithm in a quite different way, i.e., left shifting Y and M instead of right shifting S. Their approach was able to process an n-bit precision Montgomery multiplication in approximately n clock cycles, while keeping the scalability of the original

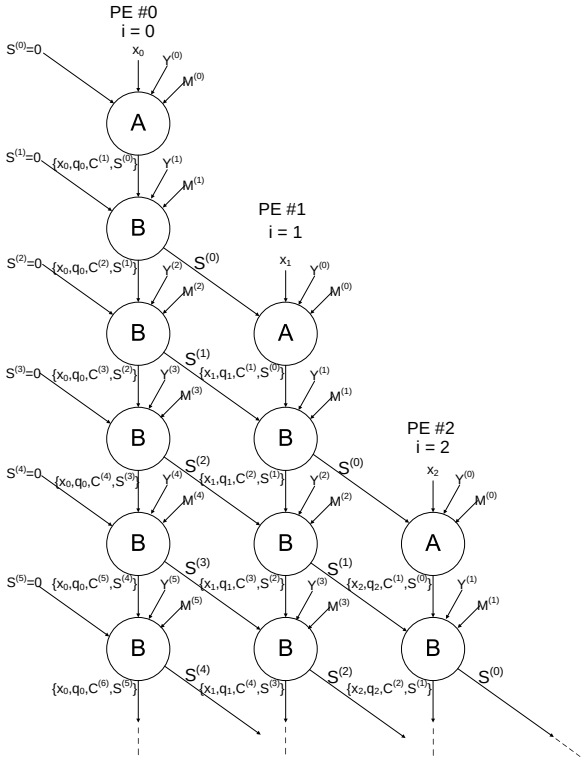


Fig. 1: Data dependency graph of the hardware architecture [2] for MWR2MM algorithm (each PE computes all words of S in the same iteration)

implementation. In [8] and [9], the left-shifting technique was applied on the radix-2 and radix-4 versions of the parallelized Montgomery algorithm [10], respectively. However, several additional cycles are necessary to complete the most significant words at the end of each iteration, which complicates the control logic. In [11], Michalski and Buell introduced an MWRkMM algorithm, which is derived from *The Finely Integrated Operand Scanning Method* described in [12]. MWRkMM algorithm requires the built-in multipliers in the FPGA device to speed up the computation. This feature makes the implementation expensive. The systolic high-radix design by McIvor *et al.* described in [13] is also capable of very high speed operation, but suffers from the same disadvantage of large area requirements for fast multiplier units. A different approach based on processing multi-precision operands in carry-save form has been presented in [14]. This architecture is optimized for the minimum latency and is particularly suitable for repeated sequence of Montgomery multiplications, such as the sequence used in modular exponentiations (e.g., RSA).

The work presented in this paper is the improved work based on the architecture proposed by Huang *et al.* [5]. The main contribution of Huang's architecture is to reduce the computation time to approximately n clock cycles while using the equivalent amount of hardware resource and running

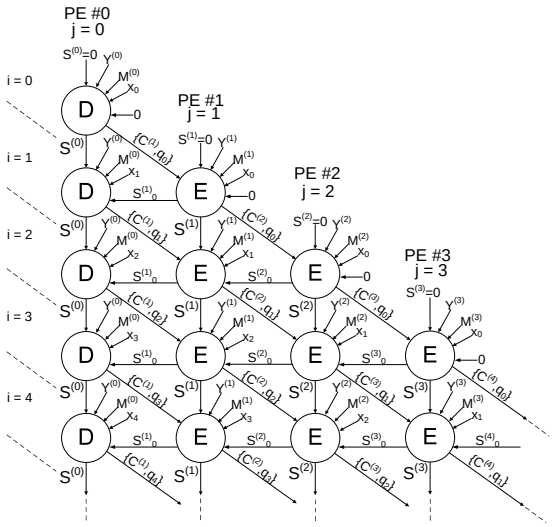


Fig. 2: Data dependency graph of architecture [5] of MWR2MM algorithm (each PE focuses on the computation of a particular word of S across all iterations)

at almost the same clock frequency. The data dependency graph of Huang's architecture is shown in Fig. 2, in which each column represents the computation of a word $S^{(j)}$ across all iterations and is computed by a PE. Tenca's architecture and Huang's architecture map the computation of $S^{(j)}$s differently. On Tenca's architecture, the computation of all words belonging to the same iteration maps to the circles in a column. Contrastingly, the same amount of computation maps to the circles on a diagonal in Huang's architecture. In other words, circle *A* and *B* in Fig. 1 correspond to circle *D* and *E* in Fig. 2, respectively.

As shown in Fig. 3, the 2-clock-cycle latency in Tenca's architecture is reduced to 1 clock cycle in Huang's architecture by pre-computing partial results using two possible assumptions regarding the most significant bit of the previous word, which becomes available after the two possible results are latched into the register. The forwarding of $S_0^{(j)}$ is illustrated as the horizontal arrow between two neighbor columns in Fig. 2. One major drawback of the previous architecture, is the demanding resource requirement for the case beyond radix-4 since 2^k branches need to be covered exhaustively for radix-2^k cases. The proposed optimization techniques in this work are meant to resolve this issue to make it viable for high radix cases. Furthermore, the throughput of the optimized architecture is improved compared with Huang's architecture by integrating gating logic.

3. Optimization

In the data dependency graph in Fig. 1, there is a 2-clock-cycle latency between the computation of two consecutive iterations due to the right shift of S. As shown in Fig. 4, the computation of $S^{(j)}$ of iteration i requires the least

Algorithm 2: Computation in Optimized PE $\#0$

Input: x_i, $Y^{(0)}$, $M^{(0)}$, $S_0^{(1)}$, $S_{w-1..1}^{(0)}$
Output: q_i, $C^{(1)}$, $S_{w-1..1}^{(0)}$
2.1 $q_i = (x_i \cdot Y_0^{(0)}) \oplus S_1^{(0)}$;
2.2 $(CE^{(1)}, SE_{w-1}^{(0)}, S_{w-2..0}^{(0)}) = (0, S_{w-1..1}^{(0)}) + x_i \cdot Y^{(0)} + q_i \cdot M^{(0)}$;
2.3 $(C^{(1)}, S_{w-1}^{(0)}) = (CE^{(1)}, SE_{w-1}^{(0)}) + S_0^{(1)}$;

Algorithm 3: Computation in Optimized PE $\#j$

Input: q_i, x_i, $C^{(j)}$, $Y^{(j)}$, $M^{(j)}$, $S_0^{(j+1)}$, $S_{w-1..1}^{(j)}$
Output: $C^{(j+1)}$, $S_{w-1..1}^{(j)}$, $S_0^{(j)}$
3.1 $(CE^{(j+1)}, SE_{w-1}^{(j)}, S_{w-2..0}^{(j)}) =$
$(0, S_{w-1..1}^{(j)}) + C^{(j)} + x_i \cdot Y^{(j)} + q_i \cdot M^{(j)}$;
3.2 $(C^{(j+1)}, S_{w-1}^{(j)}) = (CE^{(j+1)}, SE_{w-1}^{(j)}) + S_0^{(j+1)}$;

significant bit of $S^{(j+1)}$ of iteration $i-1$. Since only the most significant bit in $S^{(j)}$ is missing at the beginning of a clock cycle, it is possible to carry out the computation of $S^{(j)}$ (of iteration i) and the computation of $S^{(j+1)}$ (of iteration $i-1$) in the same clock cycle. Huang's approach [5] is to pre-compute all possible results of $S^{(j)}$ corresponding to different values of its most significant bit. Once the value of the most significant bit of $S^{(j)}$ (i.e., the least significant bit of $S^{(j+1)}$) is determined at the beginning of the following clock cycle, the correct result of $S^{(j)}$ can then be selected, as shown in Fig. 3. For a radix-2^k case, the number of possible results is 2^k, which increases exponentially as the value of k increments. The proposed optimization is meant to remove this demanding resource requirement.

As shown in Fig. 4, each PE carries out the computation of the same $S^{(j)}$ across all n iterations. Once the computation of iteration i finishes, the w-bit $S^{(j)}$ performs a right shift. Then the computation of $S^{(j)}$ in iteration $i+1$ will start in the following clock cycle, say *clk* $\#n$. Due to the right shift, the most significant bit of $S^{(j)}$ is not determined until the beginning of *clk* $\#n+1$ when the least significant bit of $S^{(j+1)}$ becomes available. In the proposed optimization, the most significant bit of $S^{(j)}$ is assumed to be '0' at the beginning of each clock cycle. At the end of each clock cycle the intermediate sum and the carry are latched into the register. Since the most significant bit of $S^{(j)}$ is assumed, the value of the most significant bit of the intermediate sum and the carry need to be adjusted by adding the "real value" of the most significant bit of $S^{(j)}$ onto them. In Fig. 4, this adjustment is represented as the link pointing from the least significant bit of $S^{(j+1)}$ to the most significant bit of $S^{(j)}$. This link corresponds to the horizontal arrow in the data dependency graph of Fig. 2.

The overall architecture of the optimized design is illustrated in Fig. 5, which consists of e PEs. These e PEs belong to three different types, head PE (i.e., PE $\#0$), middle PE (i.e., PE $\#j$, $0 < j < e-1$), and tail PE (i.e., PE $\#e-1$).

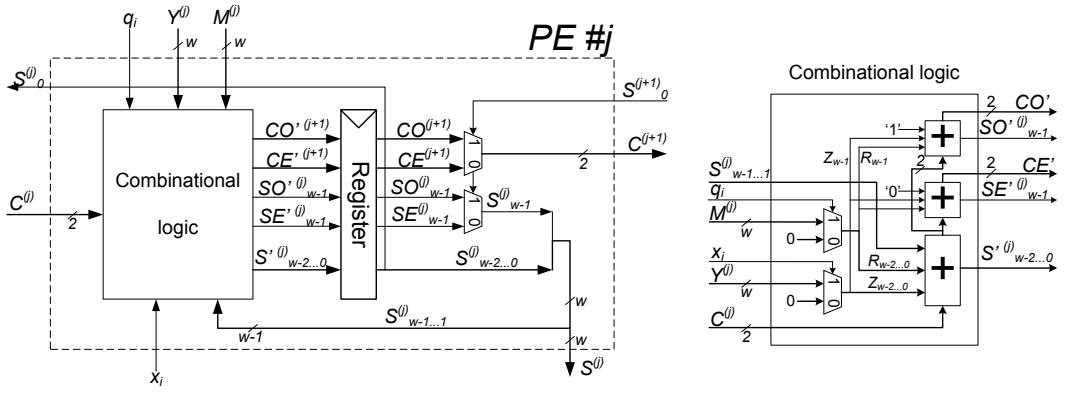


Fig. 3: Internal logic of a processing element (type E) in Huang's architecture [5]

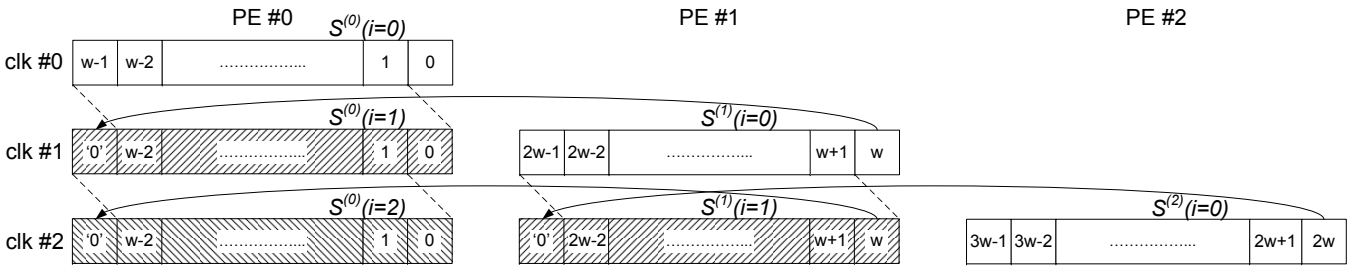


Fig. 4: Data operation diagram in the proposed architecture (*words belonging to the same iteration share the same background pattern*)

Algorithm 4: Computation in PE $\#e-1$

Input: q_i, x_i, $C^{(e-1)}$, $Y^{(e-1)}$, $M^{(e-1)}$, $S^{(e-1)}_{w-1..1}$, $C^{(e)}_0$
Output: $C^{(e)}$, $S^{(e-1)}_{w-1..1}$, $S^{(e-1)}_0$
4.1 $(C^{(e)}, S^{(e-1)}) =$
$(C^{(e)}_0, S^{(e-1)}_{w-1..1}) + C^{(e-1)} + x_i \cdot Y^{(e-1)} + q_i \cdot M^{(e-1)}$;

The internal architectures of these three types of PEs are illusrated in Fig. 5 too. The pseudocode representing their functions are listed in Alg. 2, Alg. 3, and Alg. 4, respectively. The proposed technique is applied to the head PE and the middle PEs. In the tail PE, i.e., PE $\#e-1$, $C^{(e)}_0$ becomes the most significant bit of $S^{(e-1)}$ after the right shift; therefore, it is not necessary to apply the technique. The bits in the operand X are pushed down from the head to the tail through the shift register. The parity signal q is generated by PE $\#0$ and pushed down through another shift register. By using this architecture, the Montgomery multiplication between two n-bit operands takes $n + e - 1$ clock cycles.

3.1 Extension to High-Radix Multiplications

The major issue of Huang's architecture is the huge resource requirement when it is extended to high-radix multiplications. Contrastingly, the proposed optimization technique in this work can be applied to high-radix cases conveniently. Instead of scanning one bit of X every time,

Algorithm 5: Computation in Optimized PE #j (radix-2) with Gating Logic

Input: q_i, x_i, $C^{(j)}$, $Y^{(j)}$, $M^{(j)}$, $S^{(j+1)}_0$, $S^{(j)}_{w-1..1}$, $start$, $start'$
Output: $C^{(j+1)}$, $S^{(j)}_{w-1..1}$, $S^{(j)}_0$
if $start$ **then**
  $S^{(j)}_{w-1..1} = 0$;
$(CE^{(j+1)}, SE^{(j)}_{w-1}, S^{(j)}_{w-2..0}) =$
$(0, S^{(j)}_{w-1..1}) + C^{(j)} + x_i \cdot Y^{(j)} + q_i \cdot M^{(j)}$;
if $start'$ **then**
  $S^{(j+1)}_0 = 0$;
$(C^{(j+1)}, S^{(j)}_{w-1}) = (CE^{(j+1)}, SE^{(j)}_{w-1}) + S^{(j+1)}_0$;

several bits of X can be scanned together for high-radix cases. For a radix-2^k case, the k most significant bits of a word $S^{(j)}$ are assumed to be '0's after the right shift. Correspondingly, the k least significant bits of word $S^{(j+1)}$ are used to adjust the intermediate result of $S^{(j)}$ and its carry. The Montgomery multiplication between two n-bit operands takes $\frac{n}{k} + e - 1$ clock cycles for radix-2^k case.

3.2 Achieving Maximum Normalized Throughput

In Huang's architecture, the operand X is pushed into the shift register by k bits every clock cycle in radix-2^k operation. Once the k most significant bits of X are pushed

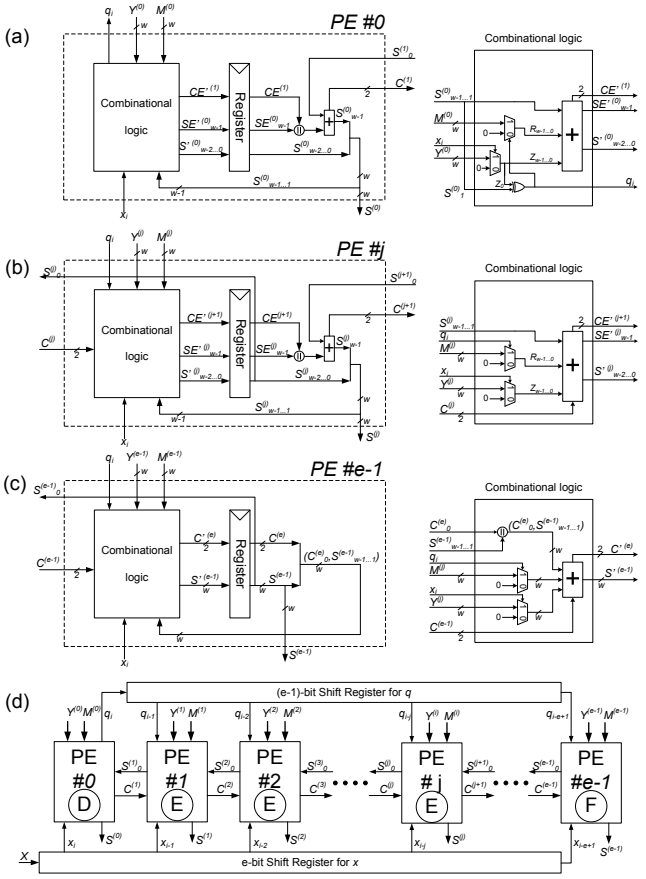


Fig. 5: (a)The internal logic of PE #0; (b)The internal logic of PE #j; (c)The internal logic of PE #e-1; (d)The overall architecture

into the shift register, it will take another $e-1$ clock cycles for the k bits to travel down to the tail PE. Therefore, it would take $\frac{n}{k}+e-1$ clock cycles to compute one operand. After the computation of one operand is finished, the whole architecture needs to be reset to the default status so that the computation of a new operand can start, which takes another extra 1 clock cycle. Under this scenario, the throughput of Huang's architecture would be $\frac{1}{(n/k)+e}$ operands per clock cycle when it is dealing with an operand stream (assuming another operand Y is fixed during the operation).

The $e-1$ clock cycles plus the extra global reset cycle can be removed by integrating gating logic into the design of each PE. Taking the internal logic of PE #j in Fig. 6 as an example, the contents of $S^{(j)}$ and $S_0^{(j+1)}$ are gated at two continuous clock cycles for the updating of $S^{(j)}$ in the first iteration. First, when the PE #j is processing the least significant bit of an input operand (i.e., $x_i = x_0$ for radix-2), $S^{(j)}_{w-1..1}$ (belonging to the operation of previous operand) is invalid and must be gated to all zeros. Second, when it is time to adjust the most significant bit of $S^{(j)}$ and the corresponding carry in the following clock cycle, $S_0^{(j+1)}$

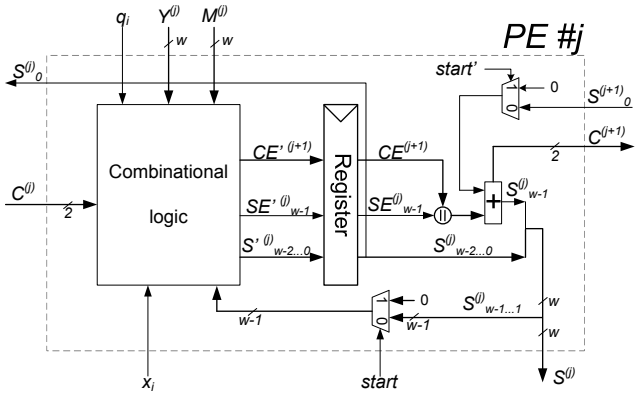


Fig. 6: The internal gating logic of optimized PE #j

has to be gated to '0' since the whole S is initialized to zero at the beginning of the computation for each operand. These two gating logic are implemented as two multiplexers, which are controlled by signal $start$ and $start'$ respectively. The corresponding pseudocode of the computation in PE #j is shown in Alg. 5.

The condition signal $start$ can be obtained by shifting it along with x. $start$ is asserted to '1' when x_0 is pushed into the shift register. For the general radix-2^k case, $start$ is asserted when the k least significant bits of X are pushed into the shift register. The condition signal $start'$ can be obtained by delaying $start$ for 1 clock cycle internally in each PE.

By integrating these two gating logic, the computation of two operands can start one after the other without stall. Therefore, the throughput of the optimized architecture can reach $\frac{1}{n/k}$ operands per clock cycle, which is the maximum rate when dealing with a stream of operands.

4. Implementation Results

The word-based Montgomery multiplication can be carried out in either non-redundant form (using carry-ripple adder) or redundant format (using carry-save adder). The proposed optimization techniques can be applied to both formats. For the sake of simplicity, all the discussion including all figures in this work are in the non-redundant format. Similarly, the implementation of the optimized architecture is in the non-redundant format too. Therefore, the results reported in this section are not comparable to the results in [2]–[4], [9] since their implementations are in redundant format. In [6], a comprehensive comparison has been made between Huang's architecture and other architectures, including Tenca's and Harris'. Through the comparison, it has been demonstrated that Huang's architecture is superior to other architectures in terms of latency×area. Therefore, we only compare the proposed optimized architecture with Huang's architecture in this work.

Table 1: Resource Utilization and Speed Comparisons ($w = 16$)

		Without Gating Logic				With Gating Logic			
Number of bits in operands (n)		1024	2048	3072	4096	1024	2048	3072	4096
Number of PEs ($\lceil (n+1)/w \rceil$)		65	129	193	257	65	129	193	257
Radix-2									
Normalized Throughput (Operands per Clock Cycle)		1/1089	1/2177	1/3265	1/4353	1/1024	1/2048	1/3072	1/4096
Huang's Architecture [5]	Frequency(MHz)	114.39	111.11	111.11	111.11	111.98	106.41	106.41	106.41
	Number of LUTs	4553	8843	13259	17675	5465	10778	16154	21530
	Number of Registers	4621	9229	13837	18445	4619	9227	13835	18443
	Number of Slices	2311	4615	6919	9223	2733	5389	8077	10765
	Number of Multiplier Blocks	0	0	0	0	0	0	0	0
Optimized Architecture	Frequency(MHz)	123.67	119.85	119.85	119.85	113.7	113.7	113.7	113.7
	Number of LUTs	4426	8588	12876	17164	5401	10777	16153	21529
	Number of Registers	4429	8845	13261	17677	4427	8843	13259	17675
	Number of Slices	2215	4423	6631	8839	2701	5389	8077	10765
	Number of Multiplier Blocks	0	0	0	0	0	0	0	0
Improvement	LUT Saving(%)	2.79	2.88	2.89	2.89	1.17	0.01	0.01	0
	Register Saving(%)	4.15	4.16	4.16	4.16	4.16	4.16	4.16	4.16
	Slices Saving(%)	4.15	4.16	4.16	4.16	1.17	0	0	0
	Frequency Speedup	1.08	1.08	1.08	1.08	1.02	1.07	1.07	1.07
Radix-4									
Normalized Throughput (Operands per Clock Cycle)		1/577	1/1153	1/1729	1/2305	1/512	1/1024	1/1536	1/2048
Huang's Architecture [5]	Frequency(MHz)	81.26	81.26	81.26	81.26	81.26	81.26	81.26	81.26
	Number of LUTs	9089	18177	27265	36353	9940	19860	29780	39700
	Number of Registers	4925	9853	14781	19709	4922	9850	14778	19706
	Number of Slices	4545	9089	13633	18177	4970	9930	14890	19850
	Number of Multiplier Blocks	0	0	0	0	0	0	0	0
Optimized Architecture	Frequency(MHz)	91.13	91.13	91.13	91.13	91.13	90.7	90.7	90.7
	Number of LUTs	7810	15618	23426	31234	8532	18442	27658	36874
	Number of Registers	4093	8189	12285	16381	4090	8186	12282	16378
	Number of Slices	3905	7809	11713	15617	4266	9221	13829	18437
	Number of Multiplier Blocks	0	0	0	0	0	0	0	0
Improvement	LUT Saving(%)	14.07	14.08	14.08	14.08	14.16	7.14	7.13	7.12
	Register Saving(%)	16.89	16.89	16.89	16.89	16.9	16.89	16.89	16.89
	Slices Saving(%)	14.08	14.08	14.08	14.08	14.16	7.14	7.13	7.12
	Frequency Speedup	1.12	1.12	1.12	1.12	1.12	1.12	1.12	1.12
Radix-16									
Normalized Throughput (Operands per Clock Cycle)		1/321	1/641	1/961	1/1281	1/256	1/512	1/768	1/1024
Huang's Architecture [5]	Frequency(MHz)	40.33	42.29	42.29	42.29	38.99	41.27	41.27	41.27
	Number of LUTs	10253	29643	50315	70987	11024	31217	52657	74097
	Number of Registers	11447	22903	34359	45815	11442	22898	34354	45810
	Number of Slices	5724	14822	25158	35494	5721	15609	26329	37049
	Number of Multiplier Blocks	131	144	144	144	131	144	144	144
Optimized Architecture	Max Frequency(MHz)	45.02	48.93	48.93	48.93	43.53	47.57	47.57	47.57
	Number of LUTs	4168	17478	32070	46662	4748	18669	33837	49005
	Number of Registers	4215	8439	12663	16887	4210	8434	12658	16882
	Number of Slices	2108	8739	16035	23331	2374	9335	16919	24503
	Number of Multiplier Blocks	131	144	144	144	131	144	144	144
Improvement	LUT Saving(%)	59.35	41.04	36.26	34.27	56.93	40.2	35.74	33.86
	Register Saving(%)	63.18	63.15	63.15	63.14	63.21	63.17	63.15	63.15
	Slices Saving(%)	63.17	41.04	36.26	34.27	58.50	40.19	35.74	33.86
	Frequency Speedup	1.12	1.16	1.16	1.16	1.12	1.15	1.15	1.15

We made two types of comparison in this paper. First, we applied the optimization technique to remove the computation redundancy in Huang's architecture to investigate its impact on two parameters, the speed and the resource requirement, particularly for high-radix cases. Second, we applied the gating logic on both Huang's architecture and the optimized architecture to explore its impact on the two same parameters. The comparison is shown in Table 1, which includes post-synthesis results. All designs are written in Verilog and the synthesis tool is Mentor Graphics Precision 2008a.47 with default settings. The target FPGA device is Xilinx VirtexII6000FF1517-4, which has 2 LUTs and 2 registers in each slice and is the same device used in [5], [6].

By observing the results in Table 1, it can be found that the operating frequency is slightly increased by removing the computation redundancy in Huang's original architecture. In the meantime, it is evident that the optimization technique is capable of reducing the resource requirement. The resource saving increases as the radix goes higher, which falls into our expectation and clearly demonstrates the advantage of the proposed optimization. It is worth mentioning that the focus of our implementation is not to design high frequency logic for high-radix operation. In the implementation of radix-16 (radix-2^4) case, the direct multiplication between a 4-bit variable and a 16-bit variable is implemented. The direct multiplication is not fully pipelined; therefore, it brings the frequency down to the range of 40 MHz. By observing Alg. 2, Alg. 3, Alg. 4, it can be found that the head PE consists of three multiplications, and the middle PE and the tail PE both include two multiplications. For the 1024-bit operation, 65 PEs are required to perform 131 4×16 multiplications concurrently. These 131 4×16 multipliers are implemented using built-in Multiplier Blocks available on VirtexII6000 devices. For the other three operand lengths, the required number of 4×16 multipliers is bigger than the 144 available Multiplier Blocks on the device. Then the synthesis tool starts using LUTs to implement the multiplication. The lack of built-in multiplier blocks contributes to the large LUTs requirement when dealing with long operands.

By comparing the results of the design with and without the gating logic, it can be found that the gating logic increases normalized throughput and does not necessarily reduce the operating frequency. However, it may increase the resource requirement by up to 20%.

5. Conclusions

In this work, we proposed two techniques to design an optimized hardware architecture to perform Montgomery multiplication. This architecture is capable of carrying out the multiplication in approximately $\frac{n}{k}$ clock cycles for radix-2^k operations, where n is the number of bits of the operands. The first technique assumes the k most significant bits of the previous word to be zeros and adds their real value later when it becomes available. For the second technique, gating logic is integrated into the PEs to remove the computation stalls when the architecture is performing multiplication on a stream of operands. The optimized architecture has a slightly higher frequency, achieves the maximum throughput of $\frac{1}{n/k}$ operands per clock cycle, and reduces resource utilization by up to 60% compared with previous work.

References

[1] P. L. Montgomery, "Modular multiplication without trial division," *Mathematics of Computation*, vol. 44, no. 170, pp. 519–521, Apr. 1985.

[2] A. F. Tenca and Ç. K. Koç, "A scalable architecture for Montgomery multiplication," in *CHES '99, Springer-Verlag Lecture Notes in Computer Sciences*, vol. 1717, 1999, pp. 94–108.

[3] ——, "A scalable architecture for modular multiplication based on Montgomery's algorithm," *IEEE Trans. Comput.*, vol. 52, no. 9, pp. 1215–1221, Sep. 2003.

[4] D. Harris, R. Krishnamurthy, M. Anders, S. Mathew, and S. Hsu, "An improved unified scalable radix-2 Montgomery multiplier," in *Proc. 17th IEEE Symposium on Computer Arithmetic (ARITH 17)*, Jun. 2005, pp. 172–178.

[5] M. Huang, K. Gaj, S. Kwon, and T. El-Ghazawi, "An optimized hardware architecture for the Montgomery multiplication algorithm," in *Proc. 11th International Workshop on Practice and Theory in Public Key Cryptography (PKC 2008), Springer-Verlag Lecture Notes in Computer Sciences*, vol. 4939, Mar. 2008, pp. 214–228.

[6] M. Huang, K. Gaj, and T. El-Ghazawi, "New hardware architectures for Montgomery modular multiplication algorithm," *to appear in IEEE Trans. Comput., retrieved from http://www.csce.uark.edu/∼mqhuang/papers/mmj_TC.pdf*.

[7] A. F. Tenca, G. Todorov, and Ç. K. Koç, "High-radix design of a scalable modular multiplier," in *CHES 2001, Springer-Verlag Lecture Notes in Computer Sciences*, vol. 2162, 2001, pp. 185–201.

[8] N. Jiang and D. Harris, "Parallelized radix-2 scalable Montgomery multiplier," in *Proc. IFIP International Conference on Very Large Scale Integration, 2007 (VLSI-SoC 2007)*, Oct. 2007, pp. 146–150.

[9] N. Pinckney and D. M. Harris, "Parallelized radix-4 scalable Montgomery multipliers," *Journal of Integrated Circuits and Systems*, vol. 3, no. 1, pp. 39–45, Mar. 2008.

[10] K. Kelly and D. Harris, "Parallelized very high radix scalable Montgomery multipliers," in *Proc. Thirty-Ninth Asilomar Conference on Signals, Systems and Computers, 2005*, Oct. 2005, pp. 1196–1200.

[11] E. A. Michalski and D. A. Buell, "A scalable architecture for RSA cryptography on large FPGAs," in *Proc. International Conference on Field Programmable Logic and Applications, 2006 (FPL 2006)*, Aug. 2006, pp. 145–152.

[12] Ç. K. Koç, T. Acar, and B. S. Kaliski Jr., "Analyzing and comparing Montgomery multiplication algorithms," *IEEE Micro*, vol. 16, no. 3, pp. 26–33, 1996.

[13] C. McIvor, M. McLoone, and J. V. McCanny, "High-radix systolic modular multiplication on reconfigurable hardware," in *Proc. 2005 IEEE International Conference on Field-Programmable Technology (FPT'05)*, Dec. 2005, pp. 13–18.

[14] ——, "Modified Montgomery modular multiplication and RSA exponentiation techniques," *IEE Proceedings – Computers and Digital Techniques*, vol. 151, no. 6, pp. 402–408, Nov. 2004.

[15] L. Batina and G. Muurling, "Montgomery in practice: How to do it more efficiently in hardware," in *Proc. Cryptographer's Track at the RSA Conference on Topics in Cryptology (CT-RSA'02)*, Feb. 2002, pp. 40–52.

[16] C. D. Walter, "Precise bounds for Montgomery modular multiplication and some potentially insecure RSA moduli," in *Proc. Cryptographer's Track at the RSA Conference on Topics in Cryptology (CT-RSA'02)*, Feb. 2002, pp. 30–39.

[17] D. Suzuki, "How to maximize the potential of fpga resources for modular exponentiation," in *Proc. 9th International Workshop on Cryptographic Hardware in Embedded Systems (CHES'07)*, Sep. 2007, pp. 272–288.

Sensitivity Analysis on Hardware Resources in SMT Processors

Naveen Davanam, Ho Young Kim, Byeong Kil Lee and Eugene John

Department of Electrical and Computer Engineering
The University of Texas at San Antonio
One UTSA Circle, San Antonio, Texas 78249-0669
byeong.lee@utsa.edu

Abstract

The microprocessor performance is increased by allowing multiple threads per clock cycle to issue instructions in simultaneous multithreading processors. Shared hardware resources are the key components of SMT processor performance. In SMT processors, impact to the performance will not be same among many hardware resources. To design an optimal hardware configuration for SMT processors, sensitivity analysis on hardware resource variation is required. In this paper, we evaluate the impact of parameters such as cache, physical register sizes, issue queue, instruction fetch policies and branch prediction accuracy on the performance improvement in the SMT processors, which demonstrates the efficacy of hardware resources among the threads. This research helps in choosing the best hardware resource sizes and the techniques in allocating these resources among the threads.

Keywords: simultaneous multithreading, sensitivity analysis, parallel processing, performance evaluation and optimization, workload characterization

1. Introduction

The speed (throughput IPC) of the processor is improved by increasing the number of the threads running on the processor [1]. Simultaneous multithreading develops the effectiveness (IPC) by using multiple threads and also by issuing instructions from multiple threads per clock cycle. It increases both the thread level parallelism and instruction level parallelism. The shared resources are the key factors which influence the performance in SMT Processors [5-7][10-11]. A SMT processor allows the multiple running threads to share the resources which potentially increase the processor utilization and performance (IPC).

In SMT processors, as the number of threads increases, resources should be increased and properly shared. An improper use of these shared resources can seriously damage the performance of SMT processors. In order to prevent it, these resources should be used in an accomplished way to enhance the performance. The multiple running threads demand high resources because as the number of threads increases, the resource starvation increases. In this paper, we present how the main resources make an impact to the performance, such as the variations on hardware configurations such as cache size, fetch policies, branch prediction, issue queue and physical registers.

We investigate the impact of resource allocation on SMT processor efficiency. The goal of this paper is to determine the sensitivity of shared hardware resources and what the particular processor resources among the running threads are significant in SMT processor efficiency. We focused on fetch policies because instruction fetch policy determines how the resources are allocated to the threads, and also examined the cache configurations and branch prediction resources. We found that the proficient way of utilizing resources and the size of resources can outweigh any losses due to the division of resources among multiple threads. We also apply our insight to find the benefit of particular hardware resource sizes and the methodologies in sharing the resources among the threads to improve the processor efficiency.

In this research, we limited ourselves to two to four threads to observe the sensitivity in hardware resources, although our methodology is relevant to application specificity and diversity. Different benchmarks are classified according to *High*, *Medium*, and *Low* IPC benchmarks, while running with single threads on the processor. We used several possible combinations of multithreaded workloads for evaluating the performance of the SMT Processor. The appropriate cache configurations in SMT processors can improve the performance among the threads. The overall performance mainly depends on the fetch policies in SMT processor, since the way the resources are allocated to the threads is based on the different fetch policies. The Fetch policies choose to allocate the resources to the threads by using the cache behavior. The instruction fetch policy can indirectly control the distribution of these shared resources, which drives the performance of SMT processors. *Dcra* (Dynamically Controlled Resource Allocation) is one of the fetch policies. This policy mainly depends on the frequent L1 data cache misses and dynamically distributes the processor resources. *Dcra* provides the better performance compared to the other fetch policies.

The primary result is with sensitivity of cache structure and how exactly it can improve the performance in a multithreaded environment. In the case of cache structure, we mainly focused on the impact of cache line sizes. The results show that performance gain (IPC) increases when there is an increase in number of threads, which increases

benefits the processor in performance from two-threaded to four-threaded mode. At the same time, as the number of threads increases, the pressure on the resources will be high and more hardware resources should be provided. The L2 cache size should be taken as a large size in SMT processors, and it is shared among the threads. Also, cache misses increases the load latency on threads. With proper fetch policies, cache performance can be improved in order to prevent the clogging in the pipeline.

The rest of the paper is organized as follows: Section 2 describes general overview of simultaneous multithreading processor. Section 3 provides our experimental framework for the sensitivity analysis, and also we analyzes the results of our experiments. Finally, we present conclusion and future work in Section 4.

2. Simultaneous Multithreading Processor

Simultaneous multithreading has been implemented to enhance the performance (IPC) by parallel execution of multiple threads on a single core of a processor. The main idea of the multithreading processor is the execution of multiple threads in a parallel form. The SMT Processor allows for multiple running threads to share the resources, which increases the performance (IPC) accuracy and reduces the processing time.

The simultaneous multithreading processor design emerged from the environment of the Superscalar processor, which tries to increase the instruction level parallelism (ILP). In the Superscalar processor, a single thread is performed and the instructions are executed in parallel. In order to improve the chip parallelism in the processor design, computer architects tend to increase the number of threads in the single core. An increase in the number of threads improves the performance by using the other threads that are available when the working thread is stalled; this optimization increases the thread level parallelism.

The SMT processor can increase the thread level parallelism while the Superscalar processor can enhance instruction level parallelism. In SMT processors, as the number of threads increases, resources should be increased and shared as well. The major parts that are shared and are accountable for the performances are the fetch unit and the execution unit. The purpose of the fetch unit is to fill the instructions in the instruction queue (IQs). Fetch units can issue multiple instructions in each cycle from multiple threads. The fetched instructions by the fetch unit are drained by the execution unit. The most significant parts that improve the SMT processing are the fetch unit and the execution unit. These two resources are shared by the threads in the SMT processors.

The SMT processor features resources that are shared by the running threads. Moreover, in every clock cycle, the running threads tend to use these shared resources at the same time. The main resource, the L2 cache, is shared among the running threads; when more threads are running, they may contend to accessing the same block of memory in the cache. This leads to the load latency in the threads and may cause a miss. The instructions in the SMT processors are fetched by the fetch unit and executed by the shared resources. Since threads can fetch instructions according to the fetch policy, the threads should be allowed to fetch instructions, with no instruction cache miss.

The resources that are shared in order to fetch and execute the instructions are: (i) Fetch unit, a unit to recover the instructions from the cache; (ii) Decode unit, which decodes the instruction and checks for the operands; (iii) Register renaming, a unit used to rename the registers, which helps to avoid the waiting time to retrieve operands for the instructions that in turn helps the instructions to execute out of order; and (iv) Execution unit, which uses the instructions that contain the operands ready by that time to execute the process with the functional unit.

3. Sensitivity Analysis

3.1 Experimental Framework

To analyze the overall impact of the resources in SMT processors, the M-sim [8] simulator is used to estimate the performance impact in SMT processors. This simulator supports the simultaneous multithreading model which is a modified version of Simplescalar 3.0d simulator [2]. The experimental evaluation of key resources in SMT processors such as the issue queue, reorder buffer and physical registers was done on different processor configurations. These key resources are shared by the running threads in the SMT processors. Table 1 shows the details of the processor baseline configurations.

Table 1. Details of the processor baseline configurations

Parameter	Configuration
Machine width	8-wide fetch,8-wide issue, 8-wide commit
Window size	Issue queue – 32 entry, 48-entry load/store queue, 96-entry ROB per thread
Function Units and Lat	4 - integer Add, 1- integer mult, 4 - floating point Add, 1 - floating point mult
Physical Registers	256 Integer + 256 floating point physical registers
L1 I-cache	64KB, 2-way set- associative, 128 byte line
L1 D-cache	32KB, 4-way set associative, 256 byte line
L2 Cache unified	2 MB, 8-way set associative, 512 byte line 10 cycles hit time
BTB	512 number of sets, 4-wat set associative
Memory	300 entry width of memory bus in bytes

The performance of the processors is evaluated according to the shared resources by simulating on different processor configurations. These processors simulate SMT architectures with 2-threads and 4-threads respectively.

We simulated the benchmarks from the SPEC 2006 both integer and floating point benchmarks [9]. These

benchmarks are precompiled alpha binaries. All the simulations skip the initialization part which is 20 million instructions for each benchmark. Our SMT architecture is analyzed using 2-threads and 4-threads, and we also simulate all the possible combinations of the simulated benchmarks. Initially, we developed a base case which is a superscalar case - in other words, the processor is simulated in a single threaded environment. This base case is employed to classify the combinations of multithreaded environment into *High*, *Med* and *Low* IPC in accomplished way. A base case simulation provides the point of reference in comparing between multithreaded environment and single threaded environment. The single-threaded simulations are simulated with default configurations unless stated otherwise. Table 2 shows the possible combinations of two-threaded and four-threaded benchmarks.

Table 2: Possible combinations of two-threaded and four-threaded benchmarks

Classification	Mixed benchmarks
2 High ILP	gromacs & leslie; gromacs & namd
2 Medium ILP	gromacs & bzip2; leslie & lbm
2 Low ILP	bzip2 & gobmk; gobmk & lbm
4 High ILP	gromacs, leslie, gamess, namd
4 Medium ILP	gromacs, leslie, bzip2, lbm
4 Low ILP	Bzip2, gobmk, lbm, perlbench

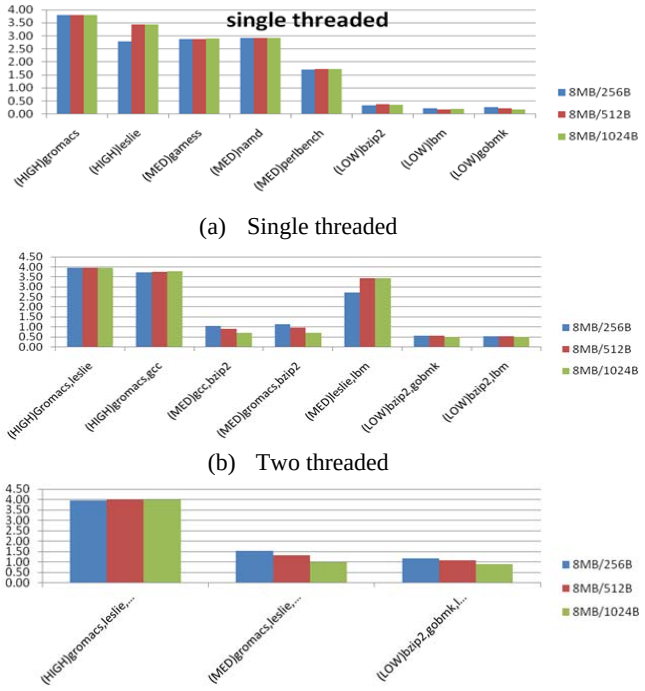
3.2 Impact of Hardware Resources

We started off by looking at the effect of cache size on the performance when running in single threaded versus multithreaded mode. The key parameters for a multithreaded processor which has multithreading support are the L2 cache size and the block size. So, our first step is to estimate the sensitivity of the cache structures and hardware impact in a simultaneous multithreading processor.

In order to select the multithreading workloads, we simulated the benchmarks in the single threaded superscalar environment and classified them using the results as high, med and low. We used several possible combinations of multithreaded workloads for evaluating the performance of the SMT processor. Several metrics are used for analysis, the first of which is the total throughput IPC.

3.2.1 Sensitivity analysis on Cache structures:

Figure 1 shows that how the IPC is changed on L2 cache size variation, in which the L2 Cache size is shared among the active threads in the processor. The performance and the miss rate of the L2 cache size is the one of the key factors in the efficiency of SMT processors. The miss rate in L2 cache size increases the load latencies for each thread processing, and the threads with high load latency needs more resources which leads to clogging in the resources. This improper utilization of resources can potentially impact on the performance of the SMT processor. In order to have better utilization of the resources, we need to have L2 Cache which has better performance with additional costs. In order to choose proper cache configurations in early design stage, we investigate the sensitivity on cache configurations vs. performance on SMT processors. Cache structure contains block size, associativity and number of sets. We analyzed each detailed part to observe the behavior of the cache in order to have a better performance and less miss rate when more threads are running simultaneously. We also observed the performance according to the cache size in single threaded mode and multithreaded mode and factored out the impact of each detailed part of the cache structure.



(a) Single threaded

(b) Two threaded

(c) Four threaded

Figure 1: Throughput IPC of 1-, 2- and 4- threaded workloads on different cache line configurations

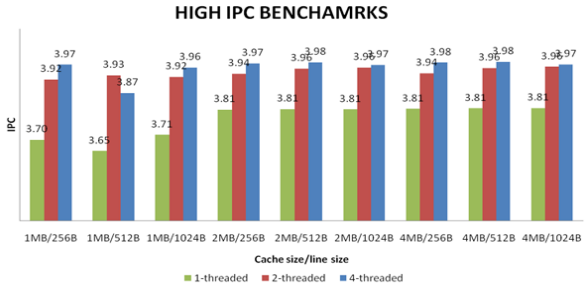
In Figure 1, the cache behavior in a SMT Processor is demonstrated on single, two and four threaded workloads. The bars in the figures plot the throughput IPC as the line sizes are varied. From the single threaded result, for a given cache size of 8MB, the line size of 512 has a better performance comparing to other line sizes. Several of the benchmarks in one- and two-threaded workloads show almost the same performance from increasing line size. In the multithreaded case, the L2 Cache is shared so both the threads may need the same block of data in it. This may increase the load in one of the threads and can cause a slight performance loss, so it is better to have an optimum line size.

In Figure 1 (c), the four-threaded workloads show a decrease in IPC for increasing line sizes in Med and Low IPC benchmarks. This performance loss is very small (only a fraction of the overall IPC); this effect arises because of the workload combination used for the four-thread workloads. However, in these results, we focused on the impact of cache line sizes in multithreading and concluded that 512-byte line has more benefit compared to other line sizes in this configuration.

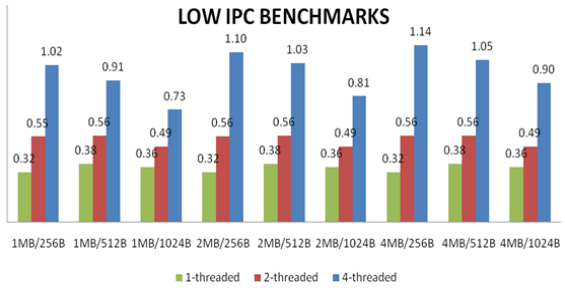
The primary result pertains to the sensitivity of the cache structure, and shows how exactly it can improve the performance in a multithreaded environment. Here, the L2 cache is shared within the threads in order to access the data so the load on the cache size is higher compared to the single-threaded environment. In other words, the benefit of sharing the L2 cache in a multi-threaded environment has more potential in performance gains, causing a slight performance loss like latency on the threads.

3.2.2 Issue Queue and Physical registers:

As the number of threads increases, the pressure on the resources will be high, so the resources should be at a high to begin with as well. According to the performance and fairness, we choose to direct the SMT performance feedback to different Issue Queue size (IQ Size) and physical registers (PRF size). As expected, we can see the performance difference according to these shared resources (Figure 2), and the performance gain is much worthier than the cost from extra hardware.



(a) High IPC benchmarks



(b) Low IPC benchmarks

Figure 2: Impact of PRF Size/IQ size in four threaded workload Benchmarks

It is significant to establish a superscalar case as a base case; it is employed to serve as a point of reference. The base case is provided by simulating the non-multithreaded mode (single-threaded) on a processor using SPEC benchmarks with the same configuration as described in the methodology. In order to compare the results using the single-threaded and multithreaded modes, the base case is programmed to run with the default configuration unless stated otherwise.

3.2.3 Fetch Policy:

In several SMT processor resources, the commonly shared asset by the different executing threads is Issue Queue size or the PRF. On the other hand, the instruction fetch policy can indirectly control the distribution of these shared resources, which is driving the performance of the SMT processors. Therefore, it is more appropriate to talk about the distribution of different fetch policies [3].

In the *Icount fetch policy*, the threads with fewer instructions in the front-end stages have the highest priorities. Unfortunately, the main drawback in the Instruction count fetch policy is that even one thread suffers many long latency loads, and keeps allocating resources to it. This results in the clogging of the pipeline, holds many resources and reduces the throughput of the processor.

Dcra (Dynamically Controlled Resource Allocation) is one of the techniques that try to overcome this drawback of the fetch policy. This fetch policy mainly depends on the cache behavior and dynamically distributes the processor resources among the threads to prevent the stalled threads form clogging resources. It allots a limited share of resources to the threads with frequent L1 data cache misses. Furthermore, the threads are able to share the resources from other threads that do not require them.

The goal in both these cases is to improve the performance by distributing the resources among threads according to their priorities; the priorities of these threads are given by different heuristics such as pending data cache misses, number of unresolved branches, L1 data cache misses and instructions in the pre-issue stages.

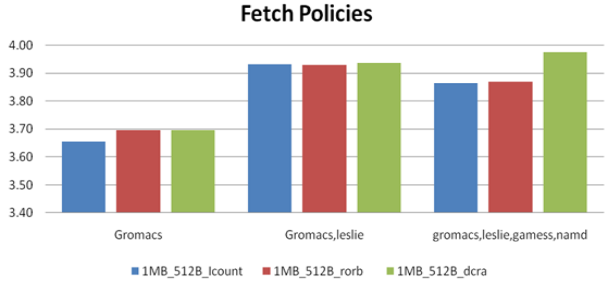


Figure 3: Instruction throughput IPC for different fetch policies with same cache size

Another mechanism, known as *the round robin policy*, allocates one fetch cycle to all threads from 0 through N-1 in a cyclic order. This is implemented without any priority to the threads. At every clock, the thread equal to the value of counter is allowed to fetch a block of instructions.

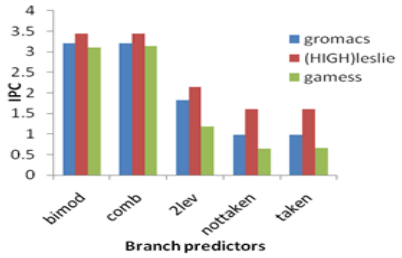
Three different mechanisms are compared using the single-threaded and multi-threaded processes. Figure 3 shows that the Icount in a single-threaded mode obtains the low performance comparing to the round robin fetch policy. This is because complete resources are allocated to the single-thread process with no need of sharing. On the other hand, switching the fetch unit is not required, therefore affecting the performance in the Icount fetch policy. An interesting point is that the three fetch policies show almost the same performance in the two-threaded process, where the resources are allocated equally to the threads. Finally, we compare the performance for two-threaded and four-threaded workloads. Resource pressure on the four-threaded workloads is high, and Dcra obtains the better performance here as compared to the other fetch policies. The Icount allocates resources to the stalled threads which lead to clogging in the pipeline and degrading the performance, and the threads with long latency loads are predicted by the Dcra fetch policy by frequent L1-data cache misses in the cache behavior.

3.2.4 Branch Predictors sensitivity on SMT Processors:

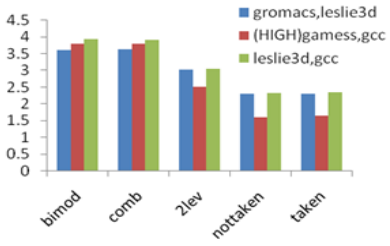
To explore the sensitivity of branch prediction that occurs between the threads in SMT processors [4], we simulated different branch predictors with different applications. This study is primarily concerned with the impact in the multi-threaded environment compared to the single-threaded mode. In order to evaluate it, we targeted five branch prediction schemes and observed the branch predictors accuracy, performance and misprediction rate in both the single-threaded and multi-threaded environment.

In our research, we first simulated the branch predictors in the single-threaded mode which is the basis of our approach. The IPC for different branch predictors are recorded and the results are shown in the Figure 4. Figure 4 (a) shows how each branch predictor can perform in a single-threaded environment. Figure 4 (b) and (c) show how these same branch predictors would perform in a multi-threaded environment, where the threads share the branch predictors. This segment provides a brief outline of each prediction scheme. The result shows that the both bimodal and Comb branch predictors have better IPC compared to other branch predictors. Same performance gain is seen in multi threaded and single threaded modes. On the other hand, schemes taken and not taken branch predictors are showing low performance which is expected, and this results provides the enough information to factor out that the both bimodal and comb branch predictors have better performance comparing to other branch predictors. The behavior of branch predictors are evaluated according to High and Low IPC benchmarks.

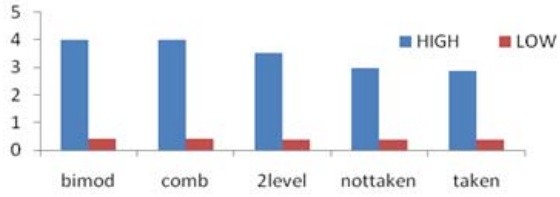
Figure 4 presents each branch predictor performance, IPC of the prediction scheme increases as the number of thread increases. To demonstrate this point more clearly, let us look at the Improvement in performance which is computed according to the formula.



(a) Single threaded mode



(b) Two threaded mode



(c) Four threaded mode

Figure 4: Performance of the five targeted branch predictions for HIGH and LOW IPC

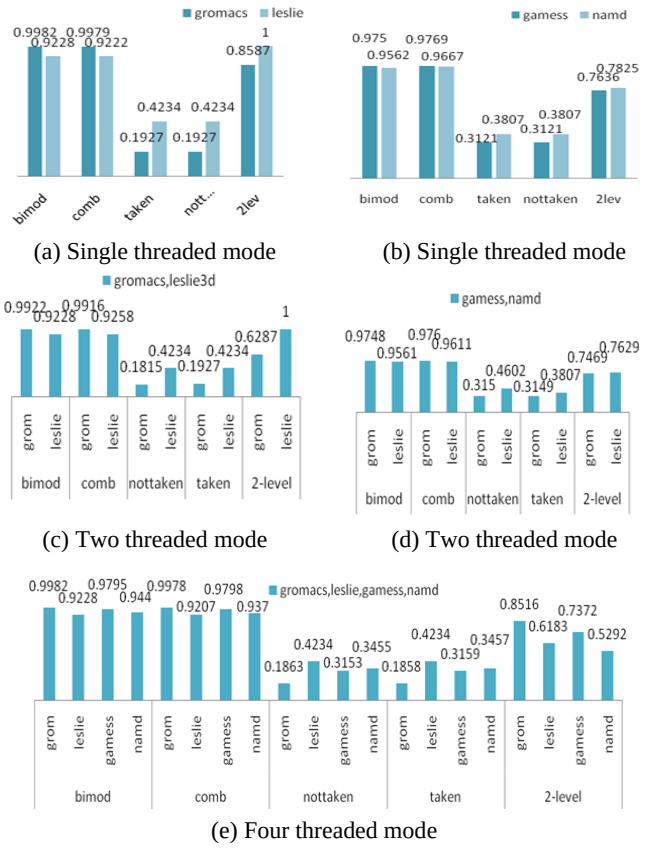
In Figure 5, we clearly showed the connection between the number of threads and strengthening of prediction rate. We can see that the more working threads can give more benefit. In addition, as the number of threads increases from two to four, the prediction rate increases more than others. Figure 5 (a) and (b) shows the branch prediction rate of unique threads in single threaded environment, and Figure 5 (c) and (d) shows the prediction rate of same threads running as 2-threads. Also, when there are four-running threads, the branch prediction rate is shown in Figure 5 (e).

Multithreading can develop the performance, and by using this result we can clearly show that the prediction rate can be enhanced by increasing number of threads. In Figure 5 (d), we can say evidently that the prediction rate of same threads is improved from two-running threads to four running-threads.

4. Conclusions and Future Work

The Characterizations of hardware resources give an idea how the resources are utilized in a way to improve the performance in SMT processors. Proper analysis of these hardware resources helps us to use the resources in an accomplished way among the running threads in the

processor. The fetch policies are the key factors to improve the performance because it can control the allocation of the hardware resources.



(a) Single threaded mode (b) Single threaded mode

(c) Two threaded mode (d) Two threaded mode

(e) Four threaded mode

Figure 5: Performance difference in single and multi threaded modes

Since the fetch policies allocate the resources among the threads by using different factors, it mainly depends on the cache behavior. The Cache missrates are predicted by these fetch policies and the resources are allocated according to the prediction, and another fetch policies allocates the resources to the threads which has less instructions in the initial stages. The Icount and Dcra both the techniques were more efficient in allocating the resources and improving the efficiency. Dcra had more performance gain with more number of threads running on the processor as compared to the other fetch policies Icount and Round Robin. More fetch policies can be used to further increase the through put IPC, but the analysis focused on these fetch policies where the threads are allocated according to the cache behavior. Based on these results, we can conclude that the Dcra fetch policies with a proper L2 cache size shared among the threads can be utilized in a efficient way among the two-running threads and four-running threads in the processor. This research helps in choosing the best hardware resource sizes and the techniques in allocating these resources among the threads. The future work of this research can include other hardware resources which improves performance.

References

[1] John Patterson, David Hennessy, Computer Architecture – A Quantitative Approach, Morgan Kaufmann, 2007

[2] D. Burger, T. Austin, "The SimpleScalar tool set: Version 2.0," Technical Report, Department of Computer Science, University of Wisconsin-Madison, June 1997 and documentation for all SimpleScalar releases.

[3] F. Cazorla, E. Fernandez, A. Ramirez, and M. Valero, "Improving Memory Latency Aware Fetch Policies for SMT processors," in the Proceedings of the 5th International Symposium on High Performance Computing, October 2003.

[4] M. Ramsay, C. Feucht, M. Lipasti, "Exploring Efficient SMT Branch Predictor Design," in the Workshop on Complexity Effective Design, 2003.

[5] D. M. Tullsen, S. Eggers, H. Levy, "Simultaneous Multithreading: Maximizing On-Chip Parallelism," in the Proceedings of the 22nd Annual International Symposium on Computer Architecture, June 1995.

[6] D. M. Tullsen, S. Eggers, J. Emer, H. Levy, J. Lo, and R. Stamm, "Exploiting Choic: Instruction Fetch and Issue on an Implementable Simultaneous Multithreading Processor," in the Proceedings of the 23rd Annual International Symposium on Computer Architecture, May 1996.

[7] D. M. Tullsen, "Simulation and Modeling of a Simultaneous Multithreading Processor," in the 22nd Annual Computer Measurement Group Conference, December 1996.

[8] J. Sharkey, "M-Sim: A Flexible, Multithreaded Architectural Simulation Environment," Technical Report CS-TR-05-DP01, Department of Computer Science, State University of New York at Binghamton, 2005

[9] SPEC, http://www.specbench.org/

[10] Jack L. Lo, Susan J. Eggers, Joel S. Emer, Henry M. Levy, Rebecca L. Stamm, Dean M. Tullsen, "Converting Thread-Level Parallelism to Instruction-Level Parallelism via Simultaneous Multithreading," ACM Transactions on Computer Systems, August 1997.

[11] Susan J. Eggers, Joel S. Emer, Henry M. Levy, Jack L. Lo, Rebecca L. Stamm, Dean M. Tullsen, "Simultaneous Multithreading: A Platform for Next-generation Processors," IEEE Micro, pages 12-18, September/October 1997.

Hardware Design and Implementation of Digital Pulse Processor using FPGA

Sanghoon Yang[1], Cheoljin Choi[1], Sook Yoon[2], Dong Sun Park[1]
[1]Department of Electronics Engineering , Chonbuk National University, Jeonju, Korea
[2]Department of Multimedia Engineering, Mokpo National University, Mokpo, Korea

Abstract - *A new data acquisition (DAQ) system was developed to fulfill the requirements of the X-ray spectrometer, providing high-resolution spectroscopy at very high-count rate. This system is based on FPGA, able to perform real time algorithm for data reduction and digital pulse processing. The DAQ system consists of digital filter, edge detector, energy resolver and so on. The main filter is based on the conventional digital time-invariant trapezoidal shaper operating. The DAQ system is implemented by Altera FPGA Cyclone III. Our proposed design is solved the previous design problem that baseline is increased gradually.*

Keywords: Digital Pulse Processing, FPGA, Digital Spectroscopy, XRF, USB

1 Introduction

European Union (EU) were fermented the RoHS directive since July 2006. According to this rule, if it contains more than standard concentration, the electrical and electronic equipment does not sell them in the European market[1],[2]. And so "IEC 62321 RoHS Test Method" has been adopted analytical method such as XRF, AAS, ICP-OES, ICP-MS, GC-MS, so on. AAS, ICP-OES, ICP-MS, GC-MS methods are as accurate and precise analysis is possible[3]~[5].

But these methods contain disadvantages that are complexity of pre-processing. The sample preprocessing comes some errors and time consuming. Also preprocessing is drawback the requiring skills. XRF is provided poor precision compare to AAS, ICP-OES, ICP-MS, GC-MS. But XRF is not limited time, place and size of sample. XRF has some advantages. These are that fast time to analyze the various elements and simple preprocessing or preprocessing process can be measured non-destructively. Portable XRF limits of the place, nor the size of the samples are less affected and proficiency. Portable XRF market will be increased gradually. In this study, DAQ system has been implemented for obtaining spectral data. XRF system has been implemented by the Altera FPGA Cyclone III. It operates 25MHz. ADC is 14 bits resolution. To improve the compatibility, external I/O uses USB interface.

2 Hardware Structure

Many DAQ systems have been developed based on the gamma-ray[6]~[9]. But in this paper, system is based on X-ray. DAQ system's hardware architecture is proposed as shown in Fig. 1. The system through input of the ADC(resolution 14bits) was connected directly to the FPGA.

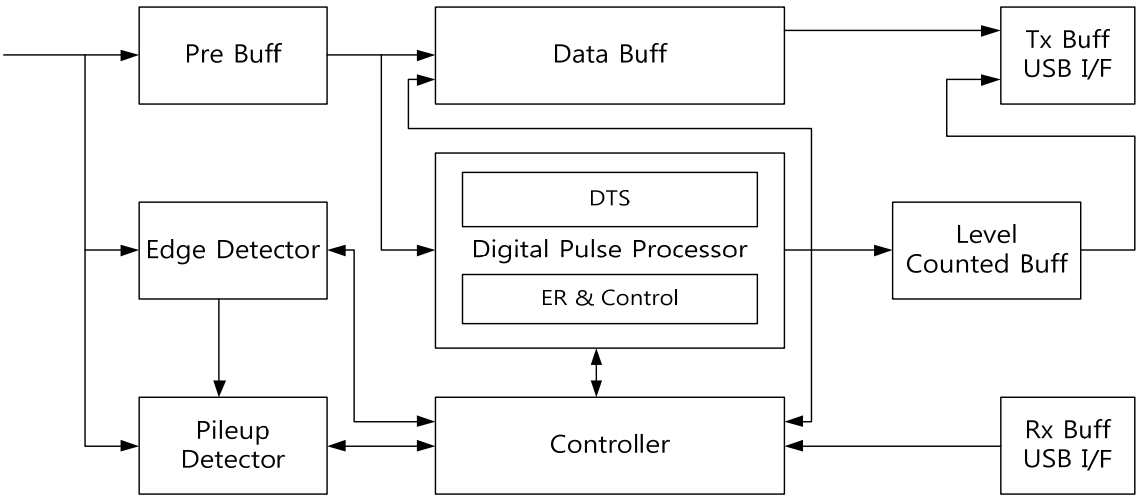


Figure 1. DAQ system Block Diagram

From the ADC input data temporarily stored at the pre-buffer. In the same time to determine whether the data is validated. Depending on the results validity of the data, the temporarily data is stored in the Data Buffer. The stored data is able to transfer to another device, or to calculate the time constant of the pulse data.

Hardware configuration is divided into three major blocks. First part is peak detecting part. It is to search the amplitude and position of the peak. Second part is DPP(Digital Pulse Processing) part. In this part, we calculate the maximum value of the pulse.

This value is flat-top. spectrum data is obtained by the DPP. The last part is controller part. Controller block consists of the register, external I/O interface, counter and memory controller.

3 Hardware Structure

Peak detecting part, as shown in Fig 1, is composed of edge detector and pileup detector. Edge detector is to detect the pulse from ADC input data. And pileup detector is to decide block through input pulse data.

To obtain spectral data, Digital Pulse Processor calculates the maximum of input pulse data. In this paper, to distinguish input pulse data from ADC, edge detector is designed based on the differentiator.

It is continuously monitoring and detecting pulse data. And it judges validity of the data using generated pulse interval or time constant.

3.1 Edge Detector

Edge Detector is to detect the pulse and to calculate the difference between the input data. Differential pulse occurs when the value is changed sharply. But also noise suddenly changes the derivative. We must remove the noise effect. So, We adapt two method.

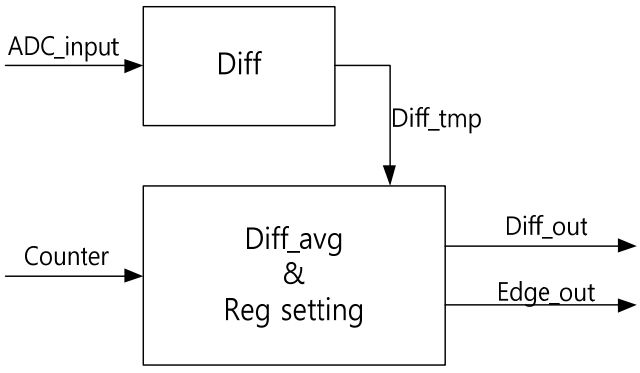


Figure 2. Design of Edge Detector

- First step, when the pulse occurs, the differential value is more than a certain threshold.
- Second step, if pulse is generated, we calculate weighted averages using the around differential value.
- Conventional methods used in the differential, using data from multiple channels and found the pulse. But our design is used just a single channel data. Previously method use the gamma-ray, but proposed design is used X-ray.
- Fig 2 is block diagram of the Edge Detector. Hardware blocks compose of 'Diff' and 'Diff_avg' blocks. Diff block is calculated differential value from input data. Diff_avg block is compared pulse conditions.

3.2 Pileup Detector

Spectral data can be obtained from the DAQ system. When the pulse occurs, spectrum data is generated. However, sometimes overlapping pulses can be generated.

The pileup is that multiple pulse are generated overlapping in a set of data. If pulse contains overlapping, it is difficult to obtain the correct peaks of the multiple pulses.

At this reason, hardware designer does not use the overlapping pulse data. They only use the correct input data. In this paper, pileup detector module was designed by Edge Detector(ED) and Counter.

The pileup generates when the pulse is detected several times by the ED. Then this block performs initialization of each register and the input data set removal.

Fig 3 is pileup detector's block diagram. Hardware design is very simply. But this module strongly detects the pileup. F/F is reset when counter is reached control value

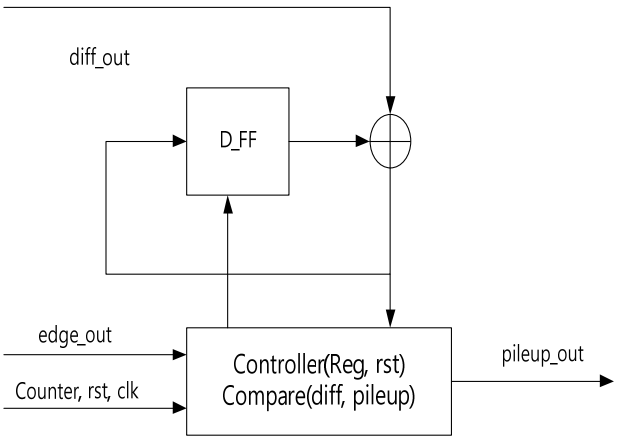


Figure 3. Design of Pileup Detector

4 Digital Pulse Processor

Spectral data can be obtained by the Edge Detector and Pileup Detector modules. Digital Trapezoidal Shaper(DTS) calculates flat-top value using pre-buffer data. At that time, calculated float-top value is the maximum of the pulse.

Flat-top value means spectrum level. Spectral data is consisted of counted value the spectrum level. XRF system uses the spectral data. Our design is used DTS to calculate the level of the pulse. Also, Energy Resolver is applied to remove the baseline.

4.1 Digital Trapezoidal Shaper(DTS)

Many algorithms have been developed to calculate the peak of pulse data[10]~[14]. In this paper, we adapted trapezoidal algorithm. This algorithm finds maximum of pulse using flat-top. Trapezoidal function was implemented according to the following formula:

$$d_n^{k,l}(t) = v_n(t) - v_n(t-k) - v_n(t-l) + v_n(t-k-l) \quad (1)$$

$$p_n(t) = p(t-1) + \sum_{i=0}^{n} d_i^{k,l}(t) \quad (2)$$

$$r_n(t) = p_n(t) + M * d_i^{k,l}(t) \quad (3)$$

$$S_n(t) = S(t-1) + \sum_{i=0}^{n} r_i \quad (4)$$

- DTS module basically consists of adder, subtracter and two kinds of shift register.

- Equation (1), input data delayed k and l clocks then calculate result of adder and subtracter operation.

- These results fed into the high pass filter to remove the pole-zero cancelation, eq (2).

- The pulse energy is calculated using M value that reflects pulse of time constant, eq (3).

- Finally, the spectrum data obtained from flat-top.

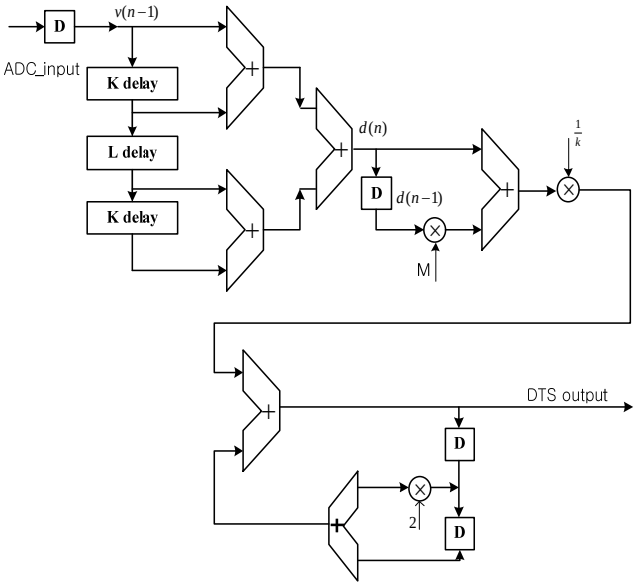


Figure 4. Hardware Design of DTS

- Figure 4 is hardware architecture of DTS. It consists of some shift register, delay and Arithmetic and Logic Unit(adder and subtracter).

4.2 Energy Resolver

In the previous method[15], result of DPP continuously increases such as step function. It is that baseline is not reset because of continuous input.

In this paper, we proposed baseline reset method. This method calculates baseline during detected pulse data before(using the average of the input data).

Proposed method is very simple and DTS module configuration can be applied in real time. Figure 5 is block diagram of energy resolver. It is very simple. And it will be applied real time system.

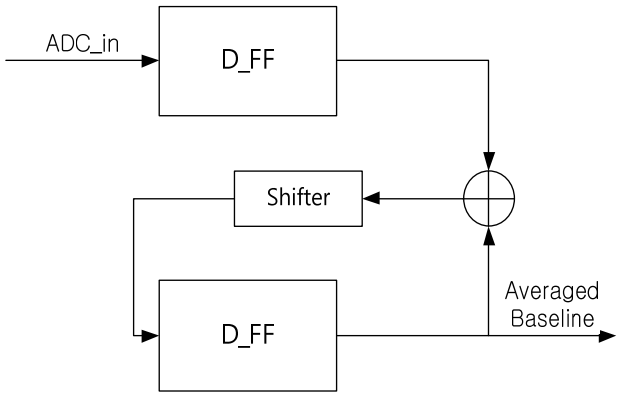


Figure 5. Block Diagram of Energy Resolver

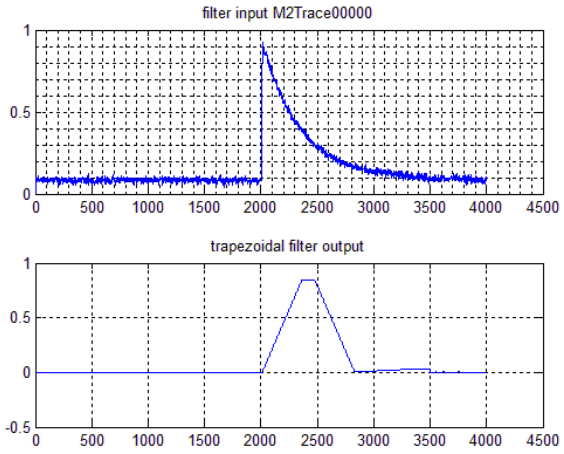


Figure 6. Auctual input and DTS output

5 Control & Extensional I/O

The controller was implemented based on the counter. Differential signal was used as interrupts. For example, pileup generates if additional differential signal is detected during the edge detector generating signals. In addition, ED is detected exceeded at least 24 samples to improve the accuracy of Energy Resolver. XRF system uses DTS operation results that collected during sample period time. Actually, component analysis is done using external devices such as PC, DSP, Embedded system. We were considering USB interface for convenience connection. In this paper, the data set consists of 1000 samples. Figure 6 is DTS resulting waveform using Matlab simulation.

6 Experiment Result

In this paper, we proposed hardware architecture of DAQ system. It is synthesized using Quartus II and simulated by ModelSim 6.5b[16]. DAQ system is implemented using Alter FPGA Cyclone III. Also, we adapts external I/O interface USB. We will be expected that helps to ensure the scalability and versatility. Figure 7 is simulation results of each hardware blocks.

6.1 Edge Detector & Pileup Detector

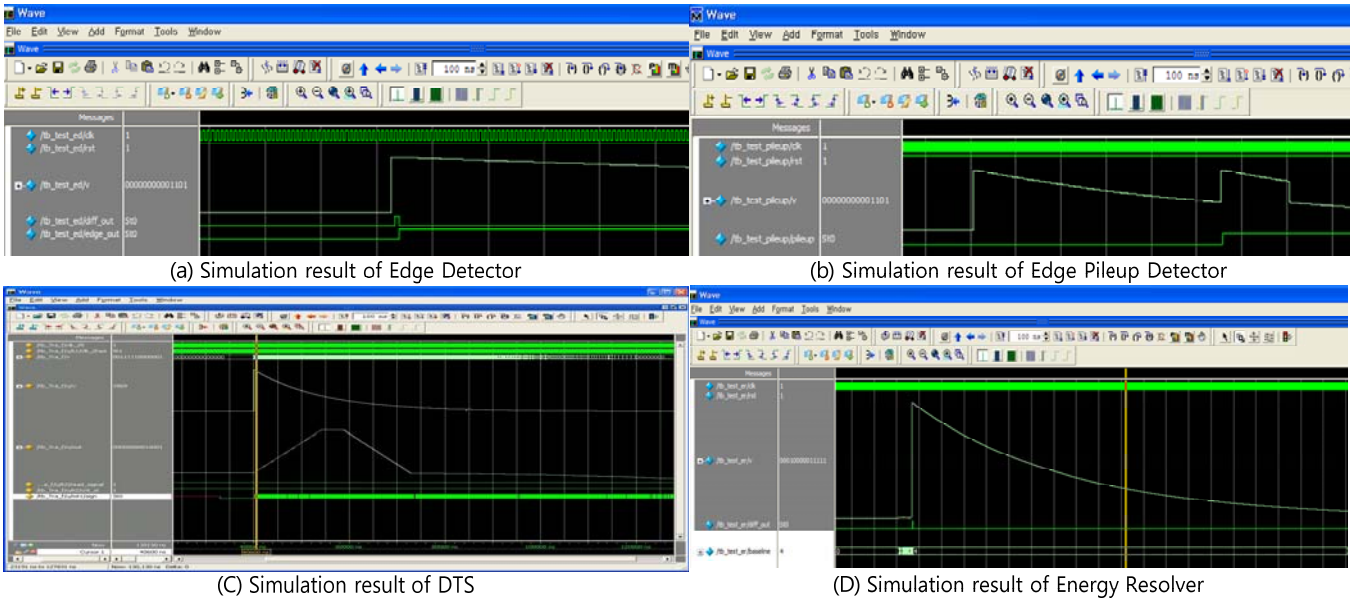
Edge Detector simulation result is Fig.7.(a). In the middle of graph, amplitude of the input signal is changing rapidly. Accordingly, differential signal occurs. Then ED signal is maintained until the end of processing or pileup detecting.

Pileup detector simulation result is Fig.7.(b). As shown in fig.7. pulse is overlapped. As a result, Pileup detecting signal created. In the proposed design, counter continues to operate until it reaches the set value. Then ED, Pileup, ER module turns initial value.

6.2 DTS & Energy Resolver

DTS simulation result is Fig.7.(c). Result of DTS is the flat-top. As you know from the simulation results, flat-top is related the time constant.

Energy resolver simulation result is Fig.7.(d). Pulse occurs, then data occurred prior to calculate the energy resolver(ER). ER pulse is detected, current value is maintained. After processing of the data set, ER is initialized.



(a) Simulation result of Edge Detector

(b) Simulation result of Edge Pileup Detector

(C) Simulation result of DTS

(D) Simulation result of Energy Resolver

Figure 7 Simulation Results of the each Blocks

7 Conclusions

DAQ system was designed and implemented using Alter Cyclone III FPGA. It synthesis of Quartus II[16] and simulates using ModelSim. Our design is operating at 25MHz.

Previous design using gamma-ray had been serious problem. That is continuously increase baseline. To remove this problem, we add reset block. When processing a set of data, the baseline was used to reset the registers.

Also we calculate average of input data is detected pulse before. Our design is very simple and powerful. To improve the compatibility, external I/O uses USB interface.

Acknowledgment

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.

8 References

[1] Directive 2002/95/EG: Restriction of the Use of Certain Hazardous Substances in Electrical and Electronic Equipment(RoHS)

[2] F. L. Melquiades and C. R. Appoloni, J. Radioanal Nucl, Chem., 262, 533-541(2004)

[3] V.G. Kiptily, et al., _ ray diagnostics of energetic ions in JET, Nucl. Fusion 42 (2002) 999–1007.

[4] M. Tardocchi, et al., Gamma ray spectroscopy at high energy and high time resolution at JET, Rev. Sci. Instrum. 79 (10) (2008) E524..

[5] JET-EP2 - Project Plan: "Gamma-ray spectrometry-GRS", 2006, JET internal document.

[6] AdvancedTCA®, PICMG® 3.0 Revision 2.0, AdvancedTCA® Base Specification,2005.

[7] A.J.N. Batista, J. Sousa, C.A.F. Varandas, ATCA digital controller hardware for vertical stabilization of plasmas in tokamaks, Review of Scientif Instruments 77 (10) (2006).

[8] A. Neto, J. Sousa, B. Carvalho, H. Fernandes, R.C. Pereira, A.M. Fernandes, et al., The control and data acquisition software for the gamma-ray spectroscopy ATCA sub-systems of the JET-EP2 enhancements, Fusion Eng. Des. 83 (2008) 346–349.

[9] R.C. Pereira, J. Sousa, A.M. Fernandes, F. Patricio, B. Carvalho, A. Neto, et al.,ATCA data acquisition system for gamma ray spectrometry, J. Fusion Eng. Des.83 2008) 341–345.

[10] FIFO Generator v4.2 User Guide, UG175, 2007 http://www.xilinx.com/.

[11] V.T. Jordanov, F. Glenn, Knoll, Digital synthesis of pulse shapes in real time for high resolution radiation spectroscopy, Nucl. Instrum. Method Phys. Res. A 345 (1994) 337–345.

[12] V.T. Jordanov, et al., Digital techniques for real-time pulse shaping in radiation measurements, Nucl. Instrum. Method Phys. Res. A 353 (1994) 261–264.

[13] A. Combo, R. Pereira, J. Sousa, N. Cruz, P. Carvalho, C.A.F. Varandas, et al., A PCI transient recorder module for the JET magnetic proton recoil neutron spectrometer, Fusion Eng. Des. 71 (2004) 151–157.

[14] Wuyun Xiao, Yixiang Wei, Xianyun Ai, Qi Ao, System simulation of digital pulse spectrometer, Nucl. Instrum. Method Phys. Res. A 555 (2005) 231–235

[15] A.M. Fernandes et al., Parallel processing method ofr high-speed real time digital pulse processing for gamma-ray spectroscopy, Fusion Engineering and Design 85 (2010) 308–312.

[16] Quartus II Handbook v10.1 Design and Synthesis, QII5V1-10.1.0,2010 http://www.altera.com/literature/hb/qts/quartusii_handbook.pdf

Numerical Precision and Benchmarking Very-High-Order Integration of Particle Dynamics on GPU Accelerators

K.A. Hawick, D.P. Playne and M.G.B. Johnson
Computer Science, Institute for Information and Mathematical Sciences,
Massey University, North Shore 102-904, Auckland, New Zealand
email: { k.a.hawick, d.p.playne, m.johnson }@massey.ac.nz
Tel: +64 9 414 0800 Fax: +64 9 441 8181

ABSTRACT
GPUs offer a powerful acceleration platform for many scientific applications. Numerical integration of classical Newtonian dynamical particles often requires very high-order numerical accuracy. We assess the floating-point precision and performance of various GPUs for applications involving high-order time-step integration methods for particle model simulations using N-squared interactions. We demonstrate how high-order algorithms can be expressed in Compute Unified Device Architecture (CUDA) and present some detailed benchmark data. We show the high numerical power of high-order integration methods such as Hairer's 10^{th} order method and relate its performance to its precision requirements.

KEY WORDS
time-stepping; numerical precision; GPU; CUDA; high-order integration.

1 Introduction

Graphical Processing Units(GPUs) have risen to great prominence recently with their deployment in many of the leading supercomputers in the Top500 world-wide list. GPUs offer a very-many core solution that is particularly effective at SIMD structured calculations such as field models and other problems where there is a great deal of separable parallelism available in the problem.

In this paper we build on ideas put forward in a previous work[1], on using N-body particle dynamics as a benchmark application problem for accelerator devices such as GPUs. A key open issue for GPUs and related devices is the extent to which they can support double (or even higher) precision floating point calculations. GPUs have proven excellent in data-parallelising integer and 32-bit float problems, and indeed many of the current and forthcoming generation devices do indeed support 64-bit floating point[2]. This support does vary however and in some cases the double precision floating point units are in fact shared across a group of compute cores within the GPU.

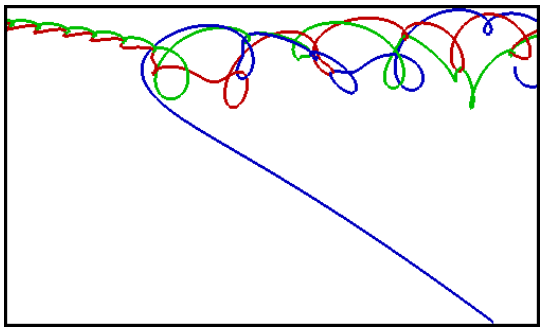


Figure 1: The motion trails of three interacting particles. Numerically integrated using the Hairer 10^{th} order integration method.

Many problems and simulation applications do not need double precision but some important ones quite definitely must be executed with the highest precision available. The case we discuss in this paper is that of high-order numerical integration in time of the Newtonian equations of motion of classical dynamics for rigid bodies[3]. Many time-integration problems can make do with standard "work-horse" algorithms such as the well known fourth-order Runge-Kutta method[4] and there are several good coded implementations of these widely available[5]. Increasingly modern applications make use of fifth and sixth order algorithms such as those of Dormand and Prince[6], and indeed routines for these algorithms are now widely available in libraries and even in tools such as Matlab[7]. However for precise orbital trajectories or indeed just for cases where conservation of energy is important, even these algorithms are insufficient. Unfortunately, from a mathematical perspective beyond fourth and fifth order the algorithms become significantly more complex and do not scale linearly in the number of floating point

operations with the order and precision required.

We have implemented a tenth-order integration method described by Hairer[8] and show that it is very necessary to conserve energy and momentum and so forth in N-body planetary orbital trajectories as might be used in planning spacecraft movements or in simulating a solar system[9], or other astronomical[10] or molecular particle dynamical problems[11, 12]. We show how a sophisticated algorithm like this can be implemented on GPU and use this as a benchmark to compare a number of GPU models with different core and memory combinations. This order and its precision requirements really need a minimum of 64 bit floating point precision to avoid truncation errors and this is therefore a good application driver to explore double and precision floating point requirements for accelerators such as GPUs.

In Section 2 we discuss the simulation of a N-body particle system and in Section 3 we discuss some of the low- and high-order integration methods that can be used to integrate their motion over time. In Section 4 we show how these simulations and integration methods can be programmed for computation on a GPU. Section 5 gives some performance and stability results which we then discuss in Section 6. Finally we offer some conclusions and discuss future work in Section 7.

2 N-Body Particle Application

We consider a set of N particles, labelled $i = 0, 1, 2, ..., N-1$ that interact via pair-wise interactions that are dependent on various properties of the particles and most importantly their individual masses. For the purposes of this present paper we consider central forces that depend solely upon the relative distance between particles i and j. For example the Newtonian gravitational potential arising on the i'th particle from the j'th, $V(r_{i,j}$, can be written as:

$$\mathcal{V}(r_{i,j}) = -\frac{Gm_i m_j}{r_{i,j}} \tag{1}$$

The classical (Newtonian mechanical) force can then be written as the gradient of the potential:

$$F = \nabla \mathcal{V}(r) \tag{2}$$

For centralised forces like gravitational systems, we can simply sum pair-wise forces along a vector connecting the particle centres, and for a single such axis the gradient is just a single-variable derivative and hence:

$$F_i = \sum_j G m_i m_j r_{i,j} \tag{3}$$

Given Newton's third law: $F_i = m_i a_i => a_i = F_i/m_i$, we can employ the separate x, y, z components of

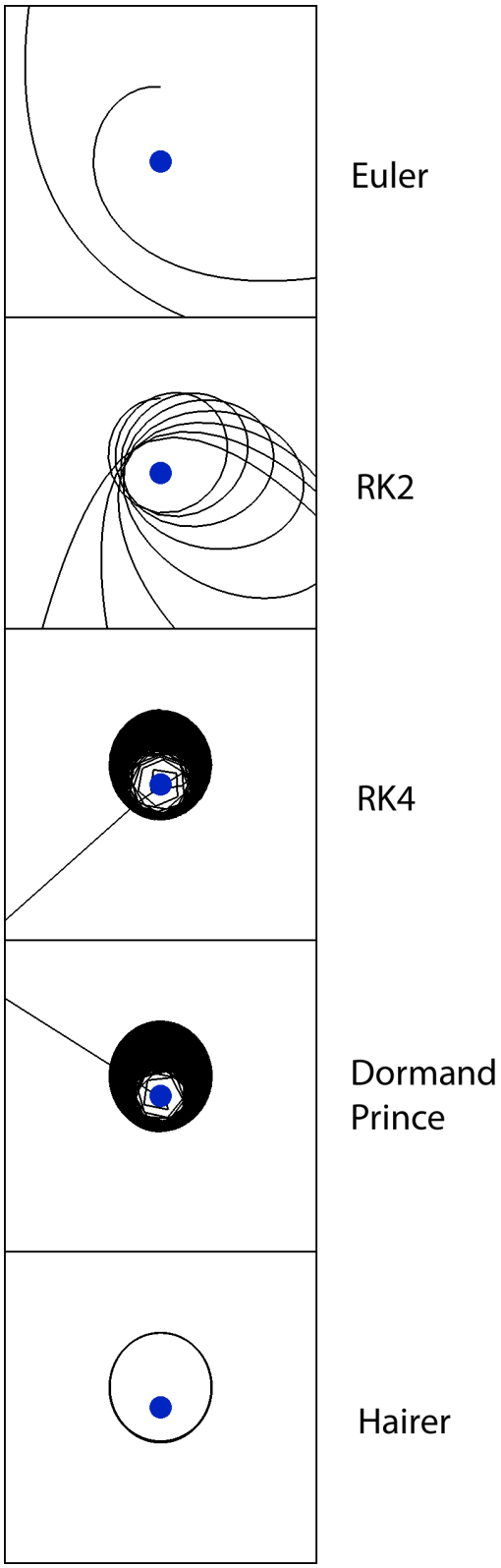


Figure 2: Numerically integrated particle trace as it orbits a single massive particle. Integration methods from top to bottom are: Euler, RK2, RK4, Dormand-Prince 5^{th} and Hairer 10^{th}. Only the Hairer method remains stable with this time-step of 0.1 for N=2.

the acceleration in the Newtonian classical rigid body equations of motion so that we compute changes in particle i's velocity and position.

For the purposes of the benchmarking application we use here, we simplify to a planar solar system so there is effectively only x, y and no z. A useful benchmark problem is obtained of we model a large central mass with a distribution of very much smaller planetary masses arranged to have close to stable orbital velocities for a randomly chosen distribution of individual masses and initial positions.

So choosing the m_i and $\mathbf{r}_i$ randomly we arrange that for each particle, its kinetic energy ($\frac{1}{2}m_i\mathbf{v}^2$) is initially set equal to its potential energy $\mathcal{V}$ - in the frame of reference of the large-massed central star (indexed by $i = 0$), so that for the remaining planetoid particles we initialise:

$$\mathbf{v}_i = \sqrt{(Gm_0/|\mathbf{r}_i|)} \tag{4}$$

and we decompose $\mathbf{v}_i$ into its v_x, v_y components by arbitrarily choosing a direction of rotation around the star for each particle. We thus have a near stable solar system that is very sensitive to the order and precision supported by the time integration algorithm and hardware.

In the limit of infinite precision we should be able to time-integrate the trajectories in near perfect (energy conserving) orbital ellipses around the central star, and even the small perturbations arising from planet-planet interactions should not give rise to large errors.

In practice of course the numerical integrations routes are very much limited and we can illustrate the effect of different algorithms of increasing order (and computational cost) both visually with plotted trajectory snapshots and also with time plots of the total energy of the system. A good algorithm will conserve energy or at worst exhibit small non-systemic fluctuations. A poor algorithm will exhibit systemic drifts in energy and will fail to conserve it even over short timescales.

3 Numerical Integration

Numerical integration[13] of ordinary differential equations or partial differential equations is a well studied problem with a range of good mathematical algorithm families to choose from. Generally speaking, for the sort of particle trajectory problem we discuss in this paper, higher order methods[14, 15] are better and necessary for systems that have large numbers of particles that interact with inverse square or similar force laws. High-order methods[16] will generally yield better numerical precision if the floating point hardware is available to perform the arithmetic appropriately.

The main tradeoff is that higher order methods generally require more intermediate force function calculations and so the computational cost scales considerably worse than linearly with order. Until the widespread availability of accelerator devices such as GPUs the increasingly prohibitive cost of high-order would likely tip the tradeoff balance in favour of methods such as the standard fourth order Runge-Kutta method (RK4) or at best a fifth-order method such as Dormand and Prince (DP5). It is not always trivial to generate software for even higher order methods and still maintain correctness. We have implemented a 10-th order RK method due to Hairer[8].

For the work reported in this paper we employ three-dimensional x,y,z coordinate system even though most test cases consider planar particle systems.

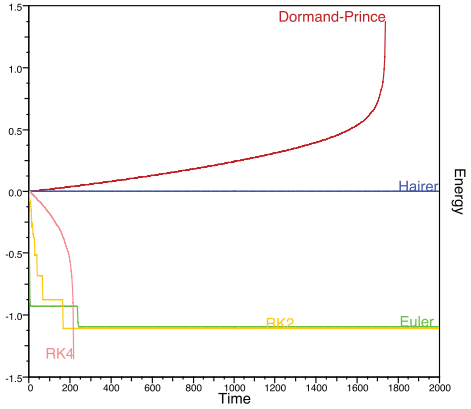


Figure 3: A plot of the energy of the system shown in Figure 1. The Hairer integration method is stable while the other methods introduce a large degree of error.

Generally explicit RK methods[17] as we discuss here can be expressed as Butcher Tableaux of coefficients: $a_{i,j}; b_i; c_i = \sum_{j=1}^{s} a_{i,j}$. We construct an s-stage method in terms of approximates Y_i with derivative F_i with $F_i = f(Y_i)$ where function f defines our differential equation: $y' = f(y(x)), y(x_0) = y_0$ and for an explicit fixed step h we simply take $x_1 = x_0 + h$. Butcher and other textbooks show that for an s-stage method we derive:

$$y_1 = y_0 + h\sum_{i=1}^{s} b_i F_i \tag{5}$$

with:

$$Y_i = y_0 + h\sum_{j=1}^{s} a_{i,j} F_j, i = 1, 2, ..., s \tag{6}$$

The work of course involves solving these and obtaining useable coefficients which are usually expressed in the form of a Butcher tableau.

Table 1: General form of a tableau representing an explicit integration method.

$$\begin{array}{c|ccccc} 0 & & & & & \\ c_2 & a_{21} & & & & \\ \vdots & \vdots & \vdots & \ddots & & \\ c_s & a_{s1} & a_{s2} & \cdots & a_{s,s-1} & \\ \hline & b_1 & b_2 & \cdots & b_{s-1} & b_s \end{array}$$

Table 1 shows the tableaux of coefficients for a generalised explicit RK method. Hairer's paper gives the coefficients in full so we do not reproduce them here, but simply note that for a 10'th order accurate method we need $s = 17$ stages and the coefficients need to be expressed to the full precision supportable for the floating point data type.

1. Initialise particle positions and velocities
2. Synchronise with the other thread
3. Compute sum of all forces on each particle and thus compute accelerations that determine the first differential equation
4. Integrate this to obtain velocities
5. Integrate the other differential equation to obtain distances moved
6. Update all particles accordingly
7. Repeat

The number of force evaluations for different algorithms will constitute the greatest computational cost and for the algorithms we employ are:

Table 2: Integration methods, their order of accuracy and number of stages required to compute them.

Name	Accuracy	Stages	Force Function Evaluations
Euler	1^{st} Order	1-Stage	1(1)
RK2	2^{th} Order	2-Stage	2(2)
RK4	4^{th} Order	4-Stage	4(7)
DP5	5^{th} Order	7-Stage	7(26)
Hairer10	10^{th} Order	17-Stage	17(99)

Table 2 emphasizes the difference in the order of accuracy for each integration method and the number of stages required to compute them. By saving the force evaluations in memory the number of force evaluations can be made equal to the number of stages. The number in brackets shows the number of force evaluations required if they are not stored to reduce memory use.

It is thus seen that a performance accelerator such as a many-cored GPU is important to offset the increasing operation count needed to obtain high numerical accuracy for the particle trajectories.

4 Graphical Processing Units

Graphical Processing Units or GPUs have steadily increasing in popularity for computing scientific simulations [18]. The high computational throughput and relatively low cost of GPUs have made them an attractive platform for parallel processing. This is known as General Purpose computation on GPUs or GPGPU and there are a number of APIs available. The most popular and powerful GPGPU API is NVidia's CUDA [19] which only works on the NVidia GeForce and Tesla graphics cards, this is the API used for this research.

Two generations of NVidia GPUs are considered in this work the Tesla and Fermi architectures. Both of these GPUs contain many (up to 512) cores known as scalar processors which are organised into multiprocessors. Tesla multiprocessors contain 8 SPs which multiprocessors on Fermi GPUs each contain 32. Programs can be run on these cores by splitting a problem into a large number of threads which are managed and scheduled by the GPU hardware. These threads are organised into blocks, each block of threads will be processed by one multiprocessor.

GPUs can manage this many cores by dropping the cache hierarchy common in most CPU architectures, eliminating the cache coherency problem that limits the expansion of cores in CPUs. To replace this cache, GPUs contain a number of specific memory types which can be used explicitly by developers to cache memory access. These memory types are:

- **Global** memory - the main memory of the GPU device. Can be accessed by the CPU host as well as any thread executing on any multiprocessor.
- **Shared** memory - is shared between the threads of a multiprocessor. It can be accessed very quickly by all the threads executing in the same block.
- **Texture** memory - is a cached access for global memory. Designed for threads in the same block accessing values from global memory in the same spatial locality.
- **Constant** memory - is another cached method of accessing global memory. Designed to allow all

threads in a block to read the same value from global memory at the same time.

The new generation Fermi architecture GPUs have also introduced a L1/L2 cache back into the design. This cache only guarantees that data written to the cache will be written to global memory once the block has finished executing. This way the problem of cache coherency is avoided.

The Tesla architecture GT200 GPUs and all of the Fermi GPUs support double precision floating point calculations, however there is still a performance loss associated with it. The higher-order integration methods discussed in Section 3 are pointless without the use of double precision floating point. We aim to compare the performance of different GPUs with respect to computing high-order integration methods using double precision.

4.1 CUDA N-Body Implementation

The CUDA implementation of the particle dynamics simulator uses the all-pairs tile calculation method described in "Fast N-Body Simulation with CUDA" [20, 1]. This method uses shared memory to cache particles and reduce global memory access. Each thread block will load a tile of particles into the shared cache, process it and then load the next tile. To process N particles with tiles of size T this process must be performed $\frac{N}{T}$ times.

This method is used to compute the acceleration of each particle due to gravity. To implement the higher order integration methods, the CUDA kernel has been adapted to use an array of coefficients (taken from the butcher tableau of the integration method) to compute the next state. The kernel must be called s times to compute the intermediate and final system according to the integration method. A set of s system states are computed which are then used in the final computation. Each state is computed as the sum of the derivatives of the previous saved states multiplied by the coefficients of the table. This must be performed for both the position and velocity of the particles.

The CUDA kernel that computes a single system state is shown in Listing 1. This kernel will compute a single state $s+1$ using *pos* to store the positions of the particles, *vel* to store the velocities, *acc* to store the accelerations, *mass* to store the masses and *coeff* to store the coefficients.

Listing 1: CUDA kernel to compute a single stage of the integration method defined by `coeff`.

```
__shared__ double4 cache[BLOCK_SIZE];

__global__ void kernel(double4 **pos, double4 **vel,
                       double4 **acc, double **coeff,
                       double *mass,   int s) {
  int i=blockIdx.x*blockDim.x+threadIdx.x;
   //Load particle i from step s
  double4 p_i = pos[s][i];
  double4 a_i = make_double4(0.0,0.0,0.0,0.0);
  //Go through all tiles
  for(int t = 0; t < N/blockDim.x; t++) {
    //Calculate tile index
    int ti=blockIdx.x+t;
    ti = (ti < gridDim.x) ? ti : ti-gridDim.x;
    ti = ti*blockDim.x;
    //Load and process tile
    cache[threadIdx.x] = pos[s][bi+threadIdx.x];
    __syncthreads();
    for (int j = 0; j < blockDim.x; j++) {
      double4 pn = cache[j];
      if((ti+j) != i) {
        double d = distance(pi-pn);
        double av = -(G* mass[tile_index+j])/(d*d);
        ai = (av/d) * (pi-pn);
      }
    }
    __syncthreads();
  }
  acc[s][i] = ai;
  double4 pn = pos[0][i];
  double4 vn = vel[0][i];
  for(int c = 0; c <= s; c++) {
    pn += vel[c][i]*coeff[s][c]*h;
    vn += acc[c][i]*coeff[s][c]*h;
  }
  pos[(s+1)][i] = pn;
  vel[(s+1)][i] = vn;
}
```

This kernel can be used to integrate the system for any of the integration methods discussed in Section 3.

5 Performance Results

Figure 4 shows the time taken by a GTX580 to compute 1000 steps of the N-body simulation using the different integration methods. The results scale as expected with the Hairer method taking the longest.

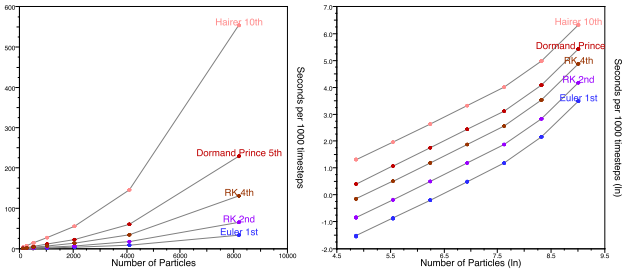


Figure 4: The performance results of the Fermi architecture GTX580 for processing the N-body simulation with different integration methods.

Figure 5 shows a comparison between the different NVidia GPUs and their performance for computing the N-Body simulation with the Hairer 10^{th} order integration method. The figure shows the performance for both the single and double precision floating point

calculations for reference. The GPUs compared are the Tesla architecture (GTX260 and GTX295) and the Fermi architecture (GTX480, GTX580 and C2070).

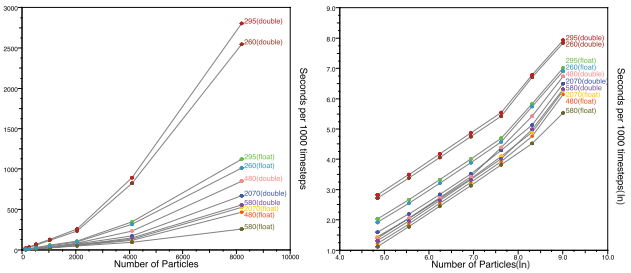


Figure 5: A comparison of different GPUs computing the N-Body simulation using Hairer 10^{th} order integration method. This plot shows the performance for the GTX260, GTX295, GTX480, GTX 580 and Tesla C2070. Timings for both double and single precision floating point calculations are shown.

As expected all graphics cards showed higher performance processing single- rather than double-precision floating point numbers. The GTX 580 showed the best overall performance for both the single- and double-precision simulations. This is expected as the GTX580 has the highest clock speeds and number of cores. One point of interest is the C2070s double-precision performance as compared to the GTX 480.

The GTX480 performs slightly better than the C2070 when processing single-precision floating point values but the C2070 provides significantly better performance when processing double-precision.

Table 3: Comparison of the different integration methods to compute a single unit of simulation time with error $e < 1 \times 10^{-12}$.

Method	Time per Simulated Second
Euler	N/A
RK2	0.2292 seconds
RK4	0.0070 seconds
Dormand-Prince	0.0042 seconds
Hairer	0.0010 seconds

Table 3 gives a comparison of the time taken for the different integration methods to compute a single simulation second for a fixed error. While the higher-order methods are more expensive to compute they are more accurate and therefore capable of numerical stability in problem regimes inaccessible to the lower-order algorithms. It can be seen from the table that the 17-stage Hairer integration method can compute a simulated second the fastest as it can support the largest simulation time-step.

6 Discussion

The performance curves in Figure 4(right) and Figure 5 (right) both show a kink at $N = 1024 \cdots 4096$ depending on the GPU model. This is attributable to the the limited processors of the GPU, above this transition multiple thread block must be executed on a single multiprocessor. Systems below this size cannot utilise the all of the GPU multiprocessors and thus do not make optimal use of the GPU hardware. For the purposes of benchmarking the system must be sufficiently large to fully utilise all available multiprocessors. The optimal point benchmarking system is at the position of this kink, however this not constant for all GPU models.

The benchmark described in [1] describes a general purpose application with low precision integration methods and floating point numbers. Whereas, this paper is aimed at specific scientific computing applications where the precision of the integration methods and double precision floating point numbers are required. Using these methods it allows us to exploit the differences between different generations and data types of modern NVidia GPU's.

7 Conclusions

We have shown that high-order time integration algorithms for N-Body classical particle dynamics make an interesting and challenging benchmark application for computational performance accelerators such as GPUs. We have explored the computational load of methods ranging from the simple Euler single stage method, through common workhorse algorithms such as fourth-order Runge-Kutta and emerging algorithms such as the Dormand and Prince fifth order method. We have also implemented the relatively unknown, superior but computationally expensive 10th order Hairer algorithm and shown the increasing number force-functions used by such seventeen-stage methods. Nevertheless we have confirmed that such numerically superior methods do support problem configurations with high curvature of variable values, that are simply not feasible with lower-order methods. We have also explored the notion of simulated seconds per second and have shown that there are problem regimes where it is actually faster in terms of the number of simulated seconds per clock second to use a high order method such as Hairer's.

We have explored properties of the NVidia GPUs and have found it quite feasible to implement these codes

using CUDA although we note the increasing code complexity required in making a correct implementation. The 10th order method uses the full numerical precision available in 64-bit arithmetic and it therefore represents a threshold in utility. An even higher order method would perhaps necessarily be crippled if only run with 64 bit floating point precision.

We have confirmed the float and double performance of NVidia's high-end GPUs and reaffirmed the ongoing importance of double precision accelerator units for use of massively multi-core data parallel devices such as GPUs for these sort of high accuracy N-Body calculations.

There is further scope for using these algorithms on GPU-enabled clusters rather than just on a single GPU-accelerated CPU node as we have discussed in this paper. There is also scope to explore other numerical integration algorithms such as the family of adaptive step-size explicit Runge-Kutta methods or implicit methods[17]. It is likely that this class of problem will remain an important benchmark for floating-point accelerators.

References

[1] Playne, D.P., Johnson, M.G.B., Hawick, K.A.: Benchmarking GPU Devices with N-Body Simulations. In: Proc. 2009 International Conference on Computer Design (CDES 09) July, Las Vegas, USA. Number CSTN-077 (2009)

[2] Knuth, D.: The Art of Computer Programming: Seminumerical Algorithms. 3rd edn. Volume 2. Addison-Wesley (1997)

[3] Hairer, E., Vilmart, G.: Preprocessed discrete MoserVeselov algorithm for the full dynamics of a rigid body. J. Phys. A: Math. and Gen. **39** (2006) 13225

[4] Hairer, E., Wanner, G.: Algebraically Stable and Implementable Runge-Kutta Methods of High Order. SIAM J. Numer. Anal. **18** (1981) 1098–1108

[5] Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes in C. 2nd edn. Cambridge University Press (1992) ISBN 0-521-43108-5.

[6] Dormand, J., Prince, P.J.: A Family of Embedded Runge-Kutta Formulae. J. of Computational and Applied Maths. **6** (1980) 19–26

[7] The MathWorks: Matlab. available at http://www.mathworks.com (2007)

[8] Hairer, E.: A Runge-Kutta Method of Order 10. J. Inst. Maths. Applics. **21** (1978) 47–59

[9] Playne, D.P.: Notes on particle simulation and visualisation. Hons. thesis, Computer Science, Massey University (2008)

[10] Warren, M.S., Salmon, J.K.: A parallel hashed oct-tree n-body algorithm. In: Supercomputing. (1993) 12–21

[11] Barnes, J., Hut, P.: A hierarchical o(n log n) force-calculation algorithm. Nature **324** (1986) 446–449

[12] Allen, M., Tildesley, D.: Computer simulation of liquids. Clarendon Press (1987)

[13] Cohen, G.C.: Higher-Order Numerical Methods for Transient Wave Equations. Number ISBN 3-540-41598-X in Scientific Computation. Springer (2002)

[14] Enright, W.H., Higham, D.J., Owren, B., Sharp, P.W.: A Survey of the Explicit Runge-Kutta Method. Technical Report 291/94, Computer Science, University of Toronto (1995)

[15] Shampine, L.F.: Some practical Runge-Kutta Formulas. Mathematics of Computation **46** (1986) 135–150 ISSN: 0025-5718.

[16] Tu, V.P., Tlusty, J.: Higher-order (2,8) fdtd method for solving multiconductor transmission-line equations. In: 2004 Large Engineering Systems Conf. on Power Engineering. (2004)

[17] Butcher, J.C.: Numerical Methods for Ordinary Differential Equations. Second edition edn. Number ISBn 978-0-470-72335-7. Wiley (2008)

[18] Leist, A., Playne, D., Hawick, K.: Exploiting Graphical Processing Units for Data-Parallel Scientific Applications. Concurrency and Computation: Practice and Experience **21** (2009) 2400–2437 CSTN-065.

[19] NVIDIA® Corporation: NVIDIA CUDA^{TM} C Programming Guide Version 3.2. (2010) Last accessed December 2010.

[20] Nyland, L., Harris, M., Prins, J.: Fast n-body simulation with cuda. In Nguyen, H., ed.: GPU Gems 3. Addison Wesley Professional (2007)

Clustered Caching for Improving Performance and Energy requirements in NoC based Multiprocessors

Hemangee K. Kapoor, Lopamudra Chatterjee and Rakesh Yarlagadda
Department of CSE, Indian Institute of Technology Guwahati, Guwahati - 781 039. Assam. India.

Abstract—*Advancement in semiconductor technology is allowing to run larger applications on chip multiprocessors. Parallelism is achieved by running different threads of applications on separate processors. This leads to coherence issues of shared data. As wire delays are dominating in current SoCs, added communication over the interconnect also adds to latency and power requirements.*

In this paper we propose to form small size clusters of cores which will share the same high-level cache instead of one global, large-size banked cache. Experimental evaluation shows that clustering gives improvement in performance and power requirements. Research on application mapping on NoC has shown that assigning nearby cores to an application improves performance. We performed experiments by localising an application within a cluster and obtained improvements in performance as well as power.

Keywords: Cache coherence, Clustering, Chip Multiprocessors, Performance, Energy

1. Introduction

Advancement in semiconductor technology is allowing to pack more and more transistors on a single die, thus increasing the complexity of the systems. This increasing amount of logic (transistors and logic-gates) available on the same die is helping to improve the efficiency of the CPU. Physical limitations like heat dissipation and data synchronisation tend to create an upper bound on the CPU performance. To improve CPU utilisation the alternative generally used is instruction level parallelism, like pipelining. However issues related to instruction prediction are not always easy to handle for all applications. Many applications are better suited for thread level parallelism which is the major impetus behind Multiprocessors systems, where each core can handle one thread at any moment of time.

Considering these issues, the trend is to design architectures where different asynchronously working processing elements can communicate with each other to provide us with the end result. In a multiprocessor system, the processors are shared among different processes. When a job is to be allocated, a subset of all the available processors is considered. This subset is chosen based on the factors of communication cost between the processors allocated to the same application, the network contentions due to flow of data between processors, etc. Many papers in the area of application mapping for SoC address these issues [1].

The complexity and the increase in communication demand for large size SoCs will no longer be met by the bus-based architectures. Network-on-chip (NoC) [2] is the solution for establishing fast and efficient links between the cores. Related to the interconnect architecture for next generation SoCs, some of the challenges [3] posed are:
(i) *Predictability and Wiring delay*: As wire delays are dominating gate delays and the performance of the system depends on the propagation delay on the chip topology, it is getting difficult to predict system performance based on the physical parameters; and
(ii) *Power dissipation* in the interconnect is increasingly becoming a considerable component of the power budget.

Now, consider the combined scenario of a multiprocessor system connected using an NoC. As the processor cores cooperate to run a bigger application or perform small but related tasks, they need to share data across processors. Such shared data, residing in a central store is ideal for consistent access, however it leads to longer latencies. Caching is done to improve the average memory access time. Multiple copies of shared data in caches of different processors leads to coherence issues. In this paper we address the problem of the relation between cores and cache in order to reduce the number of hops, and in turn to improve performance and energy requirements.

A recent work which is similar to the contributions of this paper appears in [4]. Here, the authors have divided the cores into clusters and the cores within the cluster share a set of cache banks. The modified directory stores coherence information about the availability of cache lines with other clusters. The cache replacement policy is also changed to suit the new settings. The focus here is on improving performance. Our work differs from [4] as we focus on improving performance as well as concentrate on network latency and link and router power consumption. Towards achieving this we give a detailed analysis of the effect of various cluster sizes on performance and power.

Most of the current designs use low-radix networks and develop effective cache management strategies to improve performance [4], [6], [5]. Efforts on reducing the network contention and access latencies are discussed in [7], [8]. The overall efforts are towards improving performance by either modifying coherence algorithms or by improving the

interconnect to reduce wire delays: giving better network latency and energy values. Working towards this objective, in the present paper we form clusters of cores (possibly running the same or related application and thus sharing more data), and use different coherence methods within and across the clusters to improve performance. The experimental evaluation shows that clustering helps to reduce the network latency and power dissipation along with better performance. The idea behind clustering is that instead of every core accessing a particular cache bank for a given address, the cores within the cluster will always access a fixed cache for all requests. Thus the cores in a cluster do not need to send requests to far-off cache banks, thus reducing the network latency and power consumption in the interconnection network. Also, reduced traffic going out from the cluster reduces the overall network contention.

The remaining part of this paper is organised as follows. We give a brief background in section 2. The experimental setup and results are presented in section 3. Discussion on the effects of change in cluster sizes in section 4. We present experimental results for process localisation in section 5. Finally we conclude in section 6.

2. Background

2.1 Cache coherence

Variety of protocols are designed that take advantage of the NoC [2], [9] based communication. For example, instead of storing the sharers information in a directory at a central place, one can build a virtual tree at network routers [10]. These help in faster access of the data in cases where the block in available with a nearby node instead of reading it from the memory. A priority based scheme for sending cache coherence related control packets is given in [11] which allows these smaller packets to bypass the longer data packets. A hierarchical cluster based protocol is discussed in [12] which creates virtual tree cluster among the nodes.

The results presented in this paper show improved performance by clustering the cores and using the original MESI protocol (without modifications).

2.2 Cache layout and interactions

Recent chip multiprocessors (CMPs) use low-radix (2D-mesh) network of interconnected tiles. Each tile can be a single processor or a set of few processors. The tiles usually have their private L1-cache. Each tile may have a private L2-cache, alternatively all tiles may share a global L2-cache. The global L2-cache is logically a monolithic block, but due to physical constraints it is efficient to divide this L2 into banks and place each bank in separate location. The banks of L2 are shared by all tiles/processors. Each bank maintains its own directory to store coherence information. This is the distributed directory style. Another variant is to store all the directory information in one centralised directory. In a shared-banked L2-cache, a L1 cache miss is mapped to exactly one bank depending on the address of the data block. As the requesting L1 may be located far from the serving L2 bank, this can lead to varying access times for the same memory block. However, in this style the total size of L2 available to all tiles is very large.

Alternative methods, as pointed out earlier in the paper, try to reduce the access latencies by modifying the interconnect or the cache organisation. Modify the memory architecture to make each L2-bank a new L2-cache having the total cache size same as that of the original L2. If we allow each cluster to share one L2-cache, we can give access to the same size cache to each cluster.

3. Experimental evaluation

In this section we describe in detail the experiments we performed on a 16-core mesh network. Similar evaluation was done for a 32-core network. The results section show graphs for both 16 and 32-core setups.

3.1 Simulation setup

We use 16 cores connected together by a 2D-Mesh, as shown in figure 1. The boxes containing 'R' are the routers and those containing 'L1' are the cores. Each core has its own private L1 data and instruction cache. The figure also shows the position of the L2-cache: banks in case of non-clusters and independent L2s in case of clustered arrangement.

The cores are divided into 4 clusters each of size 4 (shown by dotted rectangles in the figure). Each cluster has a L2 cache and associated local directory. The L2 is placed in a way so that it has average minimum distance to all the cluster members. The local directory only keeps track of cluster member's L1 cache states. The centralized directory associated with the main memory keeps information only of the L2s i.e. on a per cluster basis. We used a similar architecture for a 32-core setup for simulations.

3.1.1 Message-flow for non-clustered design

When there is an L1 miss the request goes to the L2 bank to which the block is mapped. Since there is only one copy of a block there is only one location for it. If the data is there (L2 hit) then it is brought from there. Otherwise (L2 miss) the request is sent to the main memory and the data is brought from the memory.

3.1.2 Message-flow for clustered design

In case of an L1 miss the request goes to the local L2 (i.e. the cache within the cluster). If the data is there (cluster hit) then it is brought from there. In the clustered scheme the L2 is closer than the L2 bank in shared banked L2 scheme where the data can be on any of the L2 banks. Therefore, in the clustered case the access latency is expected to be less

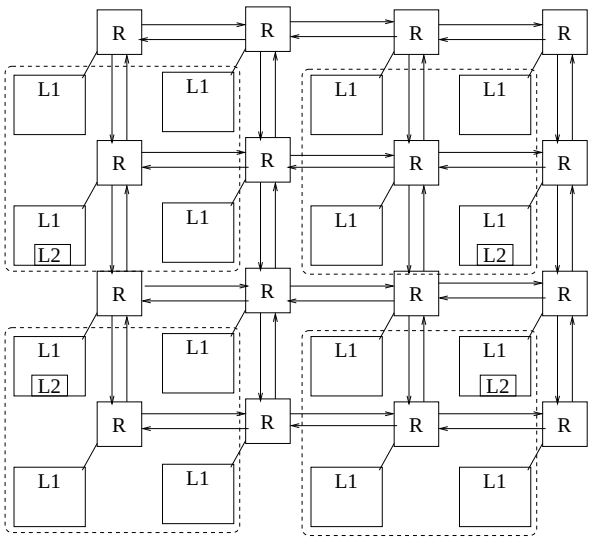


Fig. 1: 16-core 2D-mesh architecture.

and hence we expect improved performance. If there is an L2 miss (cluster miss) then the request will go to the main memory and bring the data from there.

As the number of hops have reduced by clustering we get better performance. For example, consider the top-left corner tile in figure 1. If this processor wants to access the cache, we have

- Cluster case: 1 hop to reach local L2-cache.
- Non-cluster case: Hops required to reach any of the L2 banks are either: 1, 2, 4 or 5. In other words, average 4 hops to access a cache bank.

The mapping of cores to threads should be done in a way that most members in a cluster have shared data, therefore the sharing across clusters would be less, and most of the requests would be satisfied by the local L2 (which has less access latency). This is also the main idea behind all application mapping techniques.

3.2 Experiments

The simulation was performed using Simics [13] full system simulator and GEMS toolset [14]. GEMS was used to simulate the memory timing model. It uses the existing full-system functional simulation infrastructure of Simics as the basis around which to build a set of timing simulator modules for modeling the timing of memory systems and processor cores.

In clustered system 4 L2 caches of 4 MB each were used and in non clustered system 4 banks of L2 were used where each of the banks had 4 MB capacity. Therefore total L2 size was 16 MB for both the clustered and non clustered cases. The system parameters are given below:

Component	Parameters
Processor	UtraSPARCIII+
L1 I/D cache	64KB, 4-way, 3 cycles
L2 cache bank (non-cluster)	4MB, 4-way, 6 cycles
L2 caches (clustered)	4
L2 cache size (clustered)	4MB, 4-way, 6 cycles
Memory bank	1GB, 4KB/page, 158 cycles
Distributed directory	

The network was modeled using Princeton's Garnet [15] network simulator. The fixed pipeline model was used. Garnet also provides power estimates as it incorporates the Orion [16] power models. Garnet with the help of Orion evaluates the network energy consumption using 100nm technology. A mesh network was used to connect the cores. The network parameters are given below.

Network configuration	Parameters
Flit size	16 byte
Buffer size	4
Pipeline stage	5 stage
VCs per virtual network	4
Number of virtual networks	5

Unmodified Solaris 10 operating system was loaded in the simulated system and benchmarks from Splash2 [17] benchmark suite were used to measure the performance. The benchmark applications used were: Ocean, Water-Spatial, Barnes and FMM. We compared MESI protocol [18], [19] to do the comparison of cluster against non clustered architecture.

3.3 Results and Analysis

In this section we present the results using a comparative analysis on various parameters:

- Performance.
- Miss latency: Average memory request latency.
- Average Network Latency: The latency in the network routers, network links and latency at the interface.
- Total Link Power: Sum of power consumption at all the links in the mesh.
- Total Router Power: Sum of the power consumption at all routers.
- Link Utilisation.

Objective is to increase performance and reduce all other parameters.

All the graphs show the normalised values for each of the above parameters. We have also computed the percentage improvements as will be mentioned below.

The performance for the 16-core design is plotted in the graph of Figure 2. The performance in the clustered architecture, has improved between 6.5% to 51% and in one case remained almost at par with non-clustered case. The corresponding normalised values are given in the graph.

As the benchmark programs are sharing lot of data, travelling to far-off L2-bank will result in more latency. This increased latency over the network is hampering the performance in the non-clustered case. However, due to

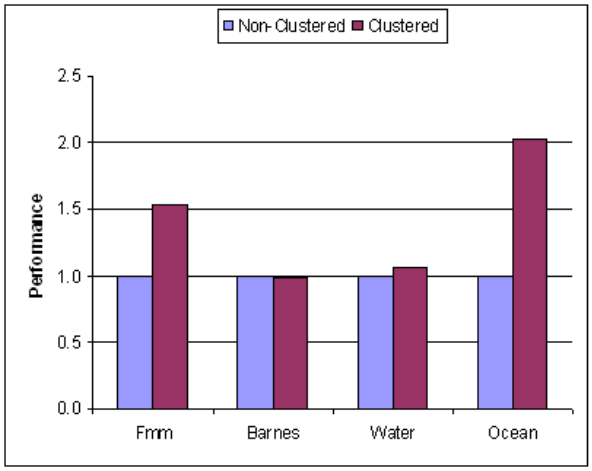


Fig. 2: Performance comparison: 16-cores

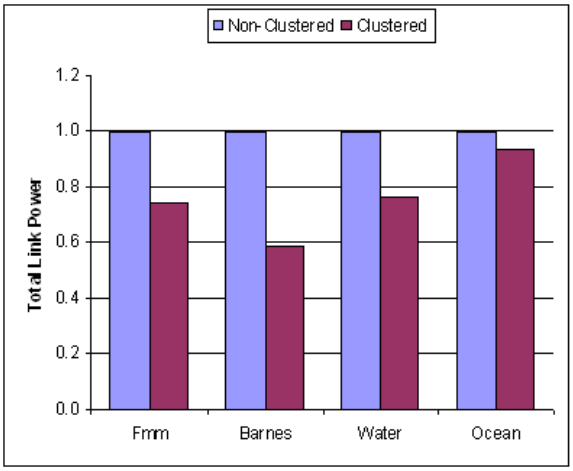


Fig. 5: Total Link Power: 16-cores

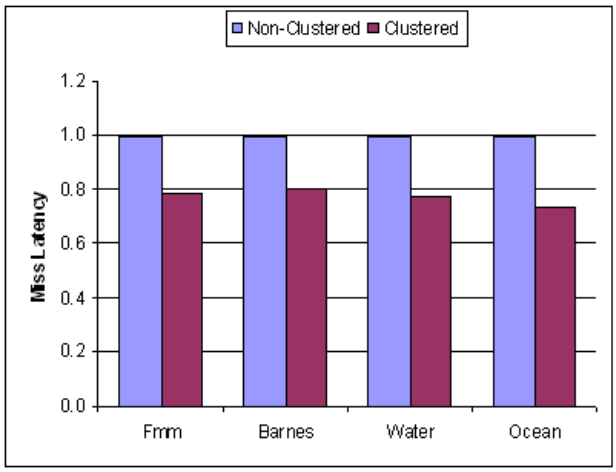


Fig. 3: Cache-miss latency: 16-cores

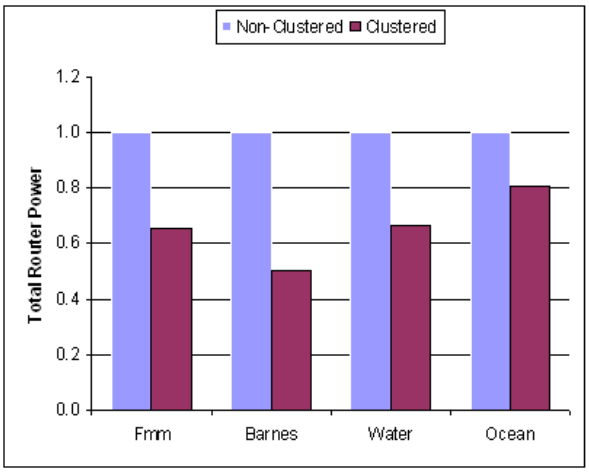


Fig. 6: Total Router Power: 16-cores

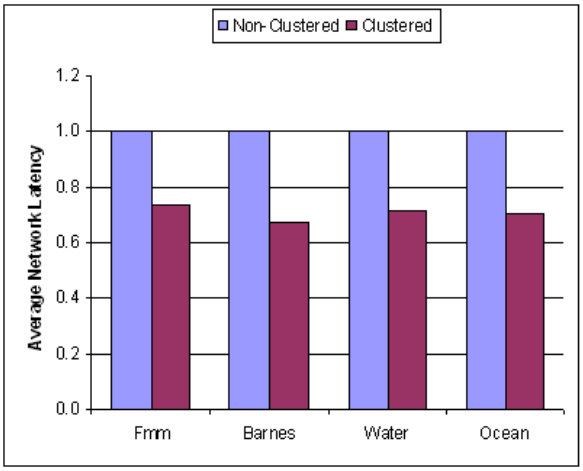


Fig. 4: Average network latency: 16-cores

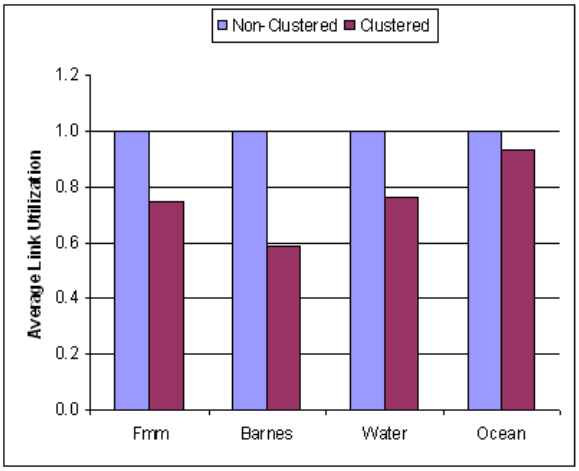


Fig. 7: Link Utilisation: 16-cores

clustering, as the data is always fetched from a fixed nearby L2-cache, we are seeing an improved performance.

The improvement in network latency and cache-miss latency are plotted in the graphs below. For the 16-core setup, we are getting 20-27% improvement in cache-miss latency and 27-32% improvement in the average network latency.

The cache access in the clustered design are localised to a nearby location and hence the network traffic has reduced. This reduced traffic helps in saving power in the links and routers. For 16-cores we have savings from 7-41% in link power and 19-50% in router power.

Reduced traffic going out from the cluster reduces the overall network contention as can be seen from the difference in link utilisation for the clustered versus non-clustered case (figure 7). We got improvement of 7-42%.

We have used inclusive cache for both clustered and non-clustered setup. In case of the clustered design, due to inclusiveness, the effective cache space for the cluster is one-fourth compared to the non-clustered (as we have not made any changes to the cache size). Even after having lesser cache size the clustered designs have performed better mainly due to lesser network traffic.

In the non-clustered case, coherence had to be maintained across all L1-caches alone. In the clustered scenario, coherence was maintained both at the L1 as well as L2-cache levels. This lead to increased coherence messages for the clustered case. However, these extra coherence messages are still small in number and size compared to the data request messages travelling to L2-banks in the non-clustered case. Hence we get better performance in clustered design.

We repeated the same experiment by increasing the number of cores to 32. Here the performance has improved between 26-55% but at the same time the link power has gone up by minimum 6% and maximum 32%. The cache miss latency and average network latency are improved by 57% and 39% respectively. In an attempt to reduce the link power and see its effect on performance, we did simulations by changing the cluster sizes. Results are presented in the next section.

4. Discussion

We performed some experiments to check the effect of cluster size and number of cores on the performance and power. The results and their analysis are presented below. Note that in all the cases we have kept the total L2 cache capacity same for the whole network.

4.1 Effect of Increase in Number of Cores on Performance

As seen earlier, in general, clustering the cores provides better performance in comparison with non-clustered architectures. The percentage improvement increases when the number of cores increase.

The performance improvement in clustering occurs because the number of hops needed in case of an L1 miss to access the L2-cache is less here. This effect is more pronounced when the number of cores is large. For example here we present the number of hops needed for the core in the top-left corner in 16 and 32 core setup:

Number of cores	Average number of hops to L2	
	Clustered	Non-clustered
16	1	3
32 (cluster size 8)	2	5

The number of hops saving is more when the number of cores are more. Correspondingly the performance gain is better as shown in the table below.

Benchmark	Percentage Improvement	
	16 cores	32 cores
FMM	34.31	45.69
Barnes	-1.89	35.91
Water	6.53	26.02
Ocean	50.58	55.96

As in the future the number of cores will continue to increase the clustered approach will provide better performance gains.

4.2 Effect of decrease in cluster size on Performance

We see that as the cluster size is changed from 8 to 4 the performance improvement for clustering reduces (although it is still better then non clustered approach). The values are given below:

Benchmark	Percentage Improvement	
	Cluster size 8	Cluster size 4
Barnes	35.91	5.8
Water	26.02	16.11

Let the total L2 capacity in the non-clustered architecture be M. Assume we divide this into N clusters. The phenomenon of relative decrease in performance due to cluster size happens because: in the clustering approach the effective cache capacity available to each cluster is (M/N), as compared to M available to the non-clustered design.

For example, in 32 cores when the cluster size is 8 (i.e. number of clusters 4) the effective cache capacity for clustering is one-fourth of that of the non-clustered approach. But when the cluster size is reduced to 4, this capacity reduces to one-eighth of the non-clustered. Whereas, in both cases the non-clustered gets full capacity cache.

Therefore the number of L1 misses goes up in case of clustered architecture and the time needed to service these extra misses takes away from the savings in execution time contributed by the reduced hop count. Hence the performance improvement reduces though it is still better than the non-clustering architecture. Below we show the increase in L1 misses for clustered against non-clustered for the two different cluster sizes. One can try to reduce the number of misses by increasing the cache size allotted to each cluster.

Benchmark	Percentage increase in L1-D misses	
	Cluster size 8 (no. of clusters 4)	Cluster size 4 (no. of clusters 8)
Barnes	30.27	61.36
Water	45.76	40.89

4.3 Change in Power consumption

In clustered case most of the traffic is contained within the cluster. For 16 cores we see that the average link utilization is less in clustered case and this results in a percentage improvement in clustered against non clustered.

However when we have a cluster size of 8 in 32 cores the number of hops needed inside the cluster is more than the number of hops necessary for 16 cores with cluster size of 4 (as depicted in table below).

	Maximum hops to L2-cache inside the cluster
16 core	2
32 core (cluster size 8)	3
32 core (cluster size 4)	2

Since the number of hops inside cluster goes up for cluster size 8 and since most of the traffic is contained the within cluster, the average link utilization goes up and it becomes more than that of non clustered. Therefore the power consumption in links also goes up. But in the case of cluster size 4, the the number of hops inside the cluster is less and the average link utilization is lower and the power consumption is also lower for clustered architecture as shown in table below.

Benchmark	Percentage Improvement	
	Average Link Utilisation	Total Link Power
Barnes 16 core	41.55	41.55
Barnes 32 core (cluster size 8)	-5.9	-5.9
Barnes 32 core (cluster size 4)	31.54	31.54
Water 16 core	23.73	23.73
Water 32 core (cluster size 8)	-5.85	-5.85
Water 32 core (cluster size 4)	1.88	1.88

4.4 Power Performance Trade off

As we go from 16 core to 32 core, the performance improvement caused by clustering is large, as shown below:

Benchmark	Percentage Improvement in Performance		
	16 core	32 core cluster size 8	32 core cluster size 4
Barnes	-1.89	45.69	5.8
Water	6.53	35.91	16.11

However, this increase comes with a penalty: as the total link power consumption increases over the non-clustered approach as shown below:

Benchmark	Percentage Improvement in Power		
	16 core	32 core cluster size 8	32 core cluster size 4
Barnes	41.55	-5.9	31.54
Water	23.73	-5.85	1.88

Therefore we can come to a power performance trade-off and use the 32 cores setup with cluster size 4. In this design, the performance improvement is not as high as when the cluster size is 8 but the power consumption is lesser than the non-clustered approach.

5. Process Localisation

We have already seen that clustering the cores improves performance. As chip multiprocessor is expected to run many applications at the same time, mapping of application should reduce communication cost due to flow of data between the processors allocated to the same application. This dictates that threads of the same application have to run on nearby cores. We propose that in such a multiple workload scenario clustering will provide better results than the traditional single banked cache.

To evaluate our hypothesis, we performed some experiments. In the 16 core setup, we executed three applications: in particular we had three instances of the Water benchmark, each with 4 threads. In the localised version, the threads belonging to the same applications are mapped to nearby cores, in that we map each instance to a cluster. In non-localized version the threads are allowed to run anywhere in the system. The scenarios considered are:
(a) *localised clustered:* Each instance running within a cluster and accessing cache within the cluster.
(b) *localised non-clustered:* Each instance running within a cluster and accessing a single global cache.
(c) *non-localised clustered:* Threads of each instance are allowed to run anywhere in the system; and threads (that may belong to a mix of instances) running within a cluster access the cache within the cluster.
(d) *non-localised non-clustered:* Threads of each instance are allowed to run anywhere in the system; and they access the single global cache.

Results obtained a given in Table 1. We computed the percentage improvements obtained using option (a) versus (b): given in row-1; and using option (c) versus (d), given in row-2. Therefore we conclude that clustering the cores delivers better performance both in single and multiple workload scenarios and can further benefit from localising the applications.

6. Conclusion

In low radix networks, the cores are arranged in a grid style with all cores sharing a set of L2-cache banks. We presented in this paper the idea of clustering the cores and making all cores within a cluster to share one bank of the L2-cache, as its independent L2-cache. As the cores have to travel smaller number of hops in the clustered case we get improved performance and also save energy in the interconnect. The interconnect power is consumed mainly due to the amount of network traffic and the utilisation of

Percentage Improvement							
	Performance	Miss latency	Avg. network latency	Link power	Router power	L1-D misses	Avg. link utilisation
localized: cluster over non-cluster	4.32	6.55	33.79	51.48	44.72	47.01	51.48
non-localized: cluster over non-cluster	4.24	25.39	35.72	10.55	24.62	-60.11	10.55

Table 1: Performance and Power improvement due to localising applications

routers. As the number of messages across the network is now mostly confined to the cluster, the overall interconnect utilisation is reduced, thus saving power. The results show that the network access is the bottleneck in the performance of an application over multiple cores.

We could achieve performance and power improvements with caches of smaller capacity each, compared to four times the cache size in the non-clustered case. We also did a comparative analysis for the 32 core setup by varying the cluster sizes and keeping the total L2-cache capacity same for all combinations. The results show that bigger cluster size increases link utilisation and hence increases power consumption, but gives better performance compared to small size clusters. Thus, there is a tradeoff between performance and power depending on the cluster size. However, in all cases the performance is better than the non-clustered. For small size clusters we get better power requirements but little less performance. Thus for a clustered design if we can allocate more cache capacity to each cluster, by increasing the total L2-cache on the network, we can get better performance and power values.

The overall results show that mapping the application to nearby cores and having them share a nearby cache can give better performance and energy savings. As future work, we intend to extend the experiment to more number of cores and try different topological placements of caches and directory to check their effect on performance and power.

Acknowledgment

The authors would like to thank Wind River for providing academic licence for the use of Simics functional simulator and the people in The Virtutech Simics Forum for their help in clearing doubts.

References

[1] C.-L. Chou and R. Marculescu, "Contention-aware Application Mapping for Network-on-Chip Communication Architectures," in *IEEE International Conference on Computer Design (ICCD)*, 2008, pp. 164–169.

[2] L. Benini and D. Micheli, "Networks on Chips: A New SoC Paradigm," in *IEEE Computer*, 2002.

[3] M. Sgroi, M. Sheets, A. Mihal, K. Keutzer, S. Malik, J. Rabaey, and A. Sangiovanni-Vencentelli, "Addressing the system-on-a-chip interconnect woes through communication-based design," in *Proceedings of the 38th annual Design Automation Conference (DAC)*, 2001, pp. 667–672.

[4] W. Zuo, S. Feng, Z. Qi, J. Weixing, L. Jiaxin, D. Ning, X. Licheng, T. Yuan, and Q. Baojun, "Group-caching for NoC based multicore cache coherent systems," in *Proceedings of the conference on Design, automation and test in Europe (DATE)*, 2009, pp. 755–760.

[5] M. H. Hammoud, S. Cho, and R. Melhem, "Dynamic Cache Clustering for Chip Multiprocessors," in *Proceedings of the 23rd Intl. Conference on Supercomputing (ICS)*, 2009, pp. 56–67.

[6] M. Zhang and K. Asanovic, "Victim replication: Maximizing capacity while hiding wire delay in tiled CMPs," in *32nd International Symposium on Computer Architecture*, 2005, pp. 336–345.

[7] B. M. Beckmann and D. A. Wood, "Managing wire delay in large chip multiprocessor caches," in *International Symposium on Microarchitecture*, 2004, pp. 319–330.

[8] C. Liqun, N. Muralimanohar, K. Ramani, R. Balasubramonian, and J. Carter, "Interconnect-Aware Coherence Protocols for Chip Multiprocessors," in *33rd International Symposium on Computer Architecture*, 2006, pp. 339–351.

[9] W. Dally and B. Towles, "Route Packets, Not Wires: On-Chip Interconnection Networks," in *Proc. of Design Automation Conference*, 2001.

[10] N. Eisley, L.-S. Peh, and L. Shang, "In-network cache coherence," *Computer Architecture Letters*, vol. 5, pp. 34–37, 2006.

[11] E. Bolotin, Z. Guz, I. Cidon, R. Ginosar, and A. Kolodny, "The Power of Priority: NoC based Distributed Cache Coherence," in *Proc. of 1st International Symposium on Networks-on-Chip*, May 2007, pp. 117–126.

[12] Y. Zhang, Z. Lu, A. Jantsch, L. Li, and M. Gao, "Towards Hierarchical Cluster based Cache coherence for Large-scale Network-on-Chip," in *4th Intl. Conference on Design and Technology of Integrated Systems in Nanoscale Era (DTIS)*, April 2009.

[13] P. S. Magnusson, M. Christensson, J. Eskilson, D. Forsgren, G. Hållberg, J. Högberg, F. Larsson, A. Moestedt, and B. Werner, "Simics: A Full System Simulation Platform," *IEEE Computer*, vol. 35, no. 2, pp. 50–58, 2002.

[14] M. M. K. Martin, D. J. Sorin, B. M. Beckmann, M. R. Marty, M. Xu, A. R. Alameldeen, K. E. Moore, M. D. Hill, and D. A. Wood, "Multifacet's General Execution-driven Multiprocessor Simulator (GEMS) Toolset," *Computer Architecture News (CAN)*, 2005.

[15] N. Agarwal, L.-S. Peh, and N. Jha, "Garnet: A detailed interconnection network model inside a full-system simulation framework," Princeton University, Tech. Rep. CE-P08-001, 2008. [Online]. Available: http://www.princeton.edu/ niketa/garnet

[16] A. Kahng, B. Li, L.-S. Peh, and K. Samadi, "Orion 2.0: A Fast and Accurate NoC Power and Area Model for Early-Stage Design Space Exploration," in *Proceedings of the conference on Design, automation and test in Europe (DATE)*, 2009, pp. 423–428.

[17] S.C.Woo, M.Ohara, E.Torrie, J. Singh, and A. Gupta, "The splash-2 programs: characterization and methodological considerations," in *22nd Annual Symposium on Computer Architecture*, 1995, pp. 24–36.

[18] D. E. Culler, J. P. Singh, and A. Gupta, *Parallel Computer Architecture*. Morgan Kaufmann, 1999.

[19] P. G. de Massas and F. Pétrot, "Comparison of memory write policies for NoC based multicore cache coherent systems," in *Proceedings of the conference on Design, automation and test in Europe (DATE)*, 2008, pp. 997–1002.

SESSION

GENERAL TOPICS AND DISCUSSIONS

Chair(s)

TBA

Evaluation of Enhanced Replacement Scheme (ERS) on Simplescalar Simulation Tool

A. Manoj Kumar[1], **B. Gaurav Gupta**[2], **and C. Anand Mohan**[2]
[1]Department of Computer Engineering, Delhi Technological University, New Delhi, India
[2]Department of Computer Engineering, University of Delhi, New Delhi, India

Abstract - *Computing machines are getting superior by the day, especially, in terms of their processor efficiencies and speeds. This enhanced capability of the processors needs to be matched by an appropriate betterment of memory performance, which can be brought about by improving the hit ratio (hit ratio is the total number of cache hits divided by the total number of cache lookups) of cache memory. This paper focuses on enhancing the hit ratio by consolidation of two popular policies for cache replacement which are Least Recently Used (LRU) and Least Frequently Used (LFU). We have implemented this new policy in Simplescalar [4] simulation tool to show its effectiveness.*

Keywords: *cache, Simplescalar, hit-ratio, replacement policy.*

1 Introduction

In the presence of advanced VLSI technologies, the processors are evolving fast. To improve the output of a computing machine, the memory system should also evolve at the same rate. The cache memory forms an important part of computer systems today to enable faster access to memory blocks which are likely to be accessed again. This paper deals with the improvement of cache hit ratio by combining two popular policies which are Least Recently Used (LRU) and Least Frequently Used (LFU).

As the name suggests, LRU is a policy which replaces the data which has not been used for the longest period of time with fresh data that does not exist in the memory. LRU is popular as it is simplistic and works well with some datasets but at times becomes erroneous, especially where frequency of data plays a major role. In such cases, the LFU policy is useful. It replaces the data which has the least frequency of being used. But again, LFU has its limitations for recently used data. The new policy, called Enhanced Replacement Scheme (ERS) provides the consolidation of features of both policies to improve the hit ratio of cache. Significantly, with an improved hit ratio, the efficiency of a machine, especially with respect to cache accesses, increases.

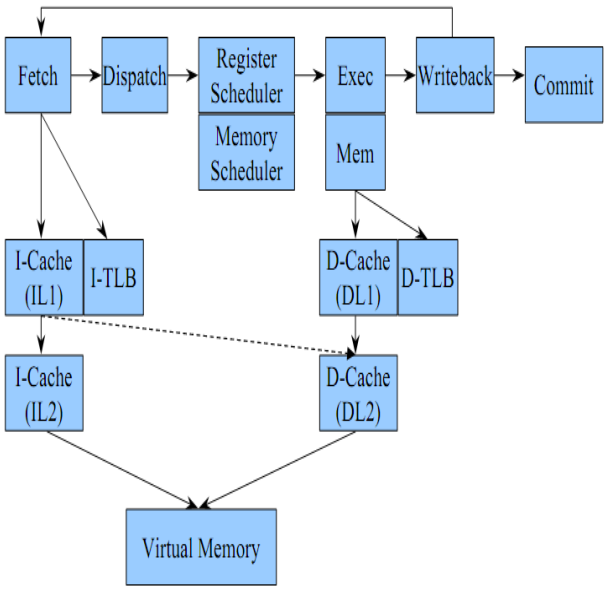
The implementation of ERS has been described in Section 2. Observations of the simulation are presented in Section 3 with an analysis of its improved performance. Section 4 concludes the paper. References are mentioned in Section 5.

2 Enhanced Replacement Scheme

The Enhanced Replacement Scheme (ERS) combines the advantages of two popular cache replacement policies which are Least Recently used (LRU) and Least Frequently Used (LFU) to form a single effective cache replacement scheme. ERS overcomes the shortcomings of LRU, by improving the performance which, in LRU, degenerates under quite common reference patterns. For example, if there are N pages in the LRU pool, an application executing a loop over array of N + 1 pages will cause a page fault on each and every access. This certainly degrades the performance of cache with respect to hit ratio and the efficiency of the system as a whole. On the other hand, the Least Frequently Used (LFU) causes 'cache pollution' in which unnecessary blocks are loaded into the cache. To counter such problems ERS chooses the block according to both LRU and LFU. The remainder of the paper discusses the implementation of ERS on Simplescalar [4] simulation tool and the results obtained in terms of hit ratio.

2.1 Hardware environment

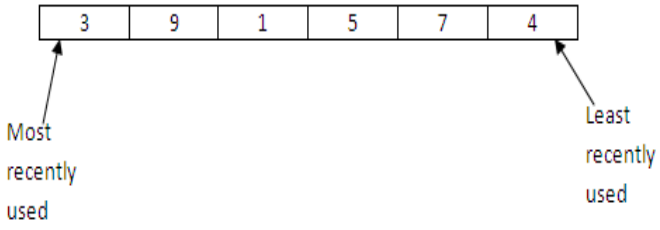
The Simplescalar [4] simulation tool provides the hardware environment as shown in the figure below.



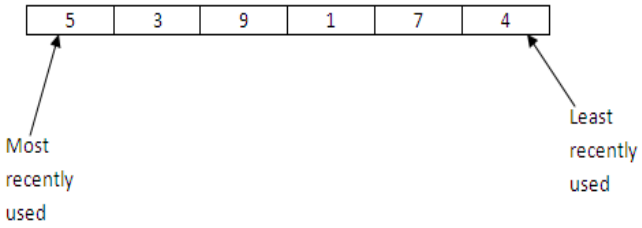
The benchmarks were run on a pipelined architecture as shown in the above figure with a two-level cache [3] i.e. if the block is not found in the level 1 cache, level 2 cache is accessed.

2.2 LRU and LFU Implementation

The Least Frequently Used algorithm is implemented using a linked list which is updated whenever a cache hit or cache miss occurs. The linked list contains all the blocks in memory. At the back of this list is the least recently used block, and at the front is the most recently used block. Suppose at a given instance of time the LRU list is as shown below.



In the above figure the block 3 is the most recently used while block 4 is the least recent. Now, if the block 5 is accessed the resulting LRU list will become as shown below.



The block 5 now becomes the most recently used and therefore, is inserted at the top of the list.

The Least Frequently Used algorithm is implemented using a counter which stores the information about the frequency of the block to which it belongs. The more the value of this counter, the more frequently accessed is the block.

2.3 ERS Algorithm

This section explains the algorithm used in ERS.

Case 1: If a Hit occurs

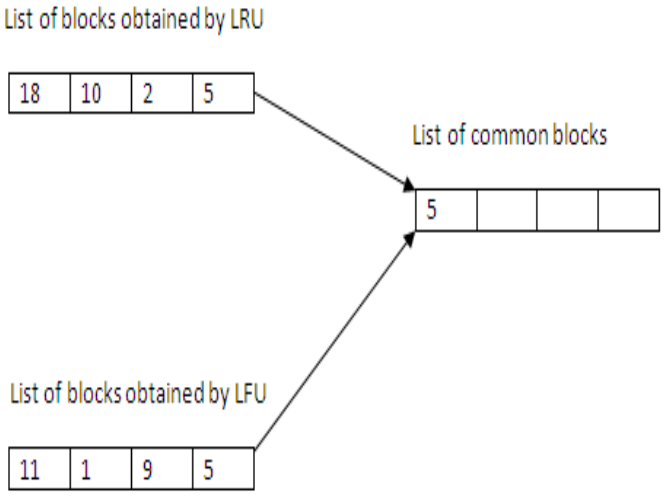
1. Increase the frequency count of the block hit by one and decrease the frequency count of other blocks by half.
2. Rearrange the LRU list, so that the least recent block appears at the tail of the list and most recent at its head.

Case 2: If a Miss occurs

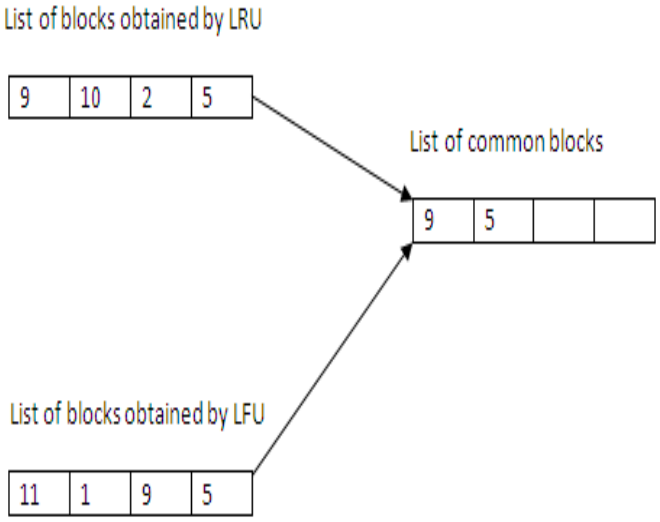
1. A list of blocks as selected by LRU is created by taking blocks from the end of the LRU linked list.
2. A list of blocks created by LFU is created based on frequency counter of each block.
3. If there is only one common block in the two lists, go to step 6**.**
4. If there is more than one common block, then take the block according to LRU list.
5. If there are no common blocks in the two lists, take sufficient blocks from the LRU linked list such that at least one block is common.
6. Replace the block obtained with the new one.

Please see the following diagrams for description of case 2 i.e. when a cache miss occurs.

When there is only one common block between the two lists, we choose that common block for replacement as shown in the figure below.



As we can see that the block 5 is the only block common between the two lists, so this block is best suited for replacement.
When there are more than one common blocks between the two lists, we choose the block as per LRU for replacement.



Since there are two blocks in common, we choose the block 5 for replacement as per LRU.
There can be a case when there is no block in common between the two lists. In that case, we can choose the block as per LRU. In our experiments we found that keeping a fixed size of four blocks in each list returned at least one common block.

3 Observations and Analysis

3.1 Level – 1 cache statistics

No. of instructions	Hit ratio (LRU) (in %age)	Hit ratio (ERS) (in %age)
5000000	94.192	94.405
7000000	94.123	94.323
8000000	94.265	94.460

The table above shows the comparison between the hit rates for level - 1 cache obtained with LRU and ERS for different number of instructions.

3.2 Level – 2 Statistics

No. of instructions	Hit ratio (LRU) (in %age)	Hit ratio (ERS) (in %age)
5000000	89.681	90.122
7000000	88.339	88.971
8000000	87.858	88.532

The table above shows the comparison between the hit rates for level - 2 cache obtained with LRU and ERS for different number of instructions.

3.3 Percentage improvement

No. of instructions	%age change Level 1 cache	%age change Level 2 cache
5000000	0.213	0.441
7000000	0.200	0.632
8000000	0.195	0.674

The table above shows the percentage improvement in hit ratio of level – 1 and level – 2 caches.

3.4 LRU vs. ERS

Level – 1 Cache

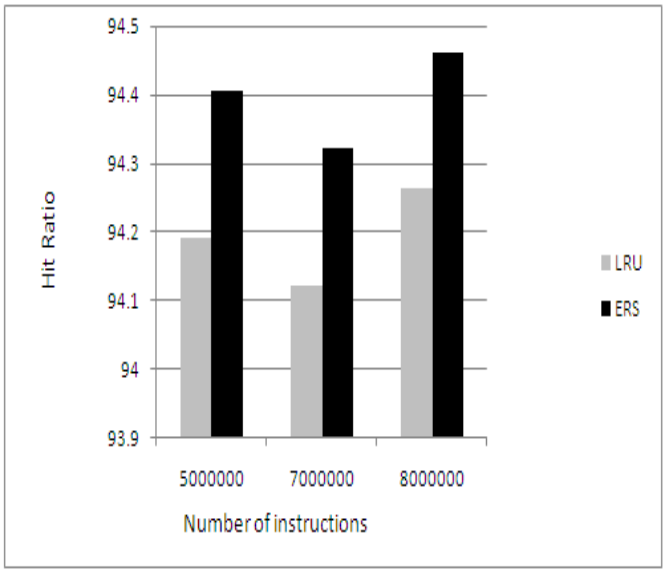


Fig. Level 1 Cache stats: Hit ratio vs. number of instructions executed on Simplescalar [4] tool

The graph above shows the comparison between the level – 1 cache hit ratios with LRU and ERS for different number of instructions. As we can see there is an improvement in the hit ratio with ERS.

Level – 2 Cache

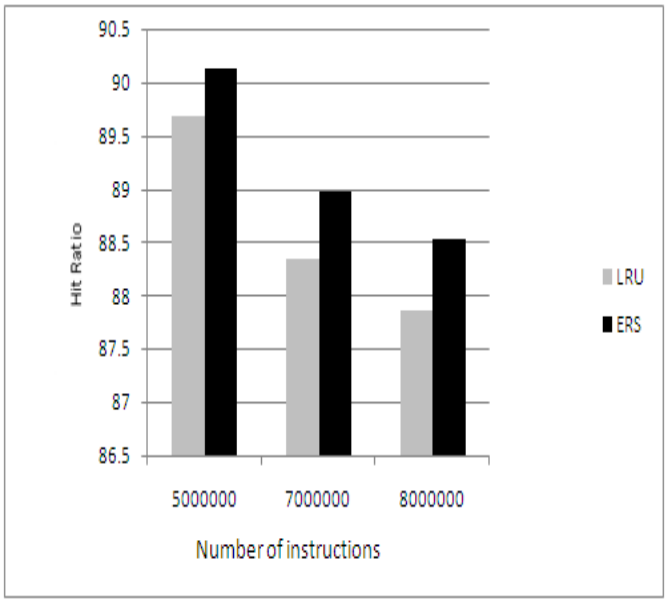


Fig. Level 2 Cache stats: Hit ratio vs. number of instructions executed on Simplescalar [4] tool

The graph above shows the comparison between the level – 2 cache hit ratios with LRU and ERS for different number of instructions. As we can see there is an improvement in the hit ratio with ERS.

3.5 Improvement over IRP

The Improved Replacement Policy (IRP) [1] policy has an extra counter for recent hits, rather than associating an extra counter, we can maintain a linked list structure which is organized so that the least recent block appears at its tail and the most recent at its head. In this way, the searching time to find a list of lesser recency is reduced. In the case when no block is common between the two lists formed by LRU and LFU, we are considering only LRU to find a block common to both the lists. Maintaining a fixed size of 4 blocks in the LRU list always gave a common block between the two lists. This means that a block which is least recently accessed is also one the less frequent. Therefore, instead of using two control parameters as in IRP [1], we are left with only one which is much easier to vary.

4 Conclusion

This paper presents the evaluation of Enhanced Replacement Scheme (ERS) which is the combination of two policies which are Least Recently Used (LRU) and Least Frequently Used (LFU). The experiment results show that the hit ratio obtained in the case of ERS is better than that in the case of LRU. Thus, ERS can be deployed a more effective cache replacement policy.

5 References

[1] Jaragh M, Hasswa A. Implementation, analysis and performance evaluation of IRP replacement policy, report. Computer Engineering Department, Kuwait University, 2003.

[2] Lee D, Choi J, Choe H, Noh SH, Min SL, Cho Y. Implementation and performance evaluation of the LRFU replacement policy. In: Proceeding of the 23rd Euromicro conference. 1997. p.106–11.

[3] M. S. Obaidat .A trace-driven simulation study of two-level cache systems. Original Research Article Computers & Electrical Engineering, Volume 21, Issue 3, May 1995, Pages 201-210.

[4] Simplescalar Tool (designed by Todd Austin www.**simplescalar**.com/docs/simple_**tutorial**_v2.pdf).

[5] Silberschatz A, Galvin B, Gagne G. Operating systemconcepts. 7th ed. John Wiley & Sons, Inc.

Computerized Sale and Distribution System and its Economical Analysis

M Hamedanian

Islamic Azad University, Central Organization, Tehran, Iran

Abstract

The aim of this research is planning and performance of a computerized system so that parts of management information system (MIS) in the companies which have several sale and distribution agencies prove to be true. Recognition of manual sale and distribution system and relation between the elements and method of changing to a computerized system includes of two sub-systems: Distribution and sale sub-system is in distributing centers, and Distribution and sale MIS sub-system is in central office. Distribution and sale management information system is able to prepare management reports, diagrams and statistics for management team. For economic analysis of this system, research theories in case of speed, accuracy, cost and user satisfaction are studied. This study is done by means of information and test application of difference between averages and estimating the differences between averages.
Finally, following results and suggestions were presented: Concerning the time, accuracy, expense and user satisfaction, the planned system was significantly superior to the manual system. Also, it has the ability of providing diagrams, special reports and so on, which were not possible in the manual system. So, it is proposed to the companies to promote the management information system.

Keywords: Computerized sale and distribution system, Economical analysis and Management Information system

1- Introduction

In these days, accessing to accurate and correct information for management decision due to rapid development of organizations activity is essential and vital. Information volume accretion in different organizational stages and sensitiveness of making decision in circumscribed time makes more and more manifest why management needs updated information. Therefore, using computerized systems in today's technique and knowledge is crucial. Understanding of positive and negative aspects in information techniques will direct to better selection and complete productivity. Management information systems are just computerized and connected system design.

One of the organizational branches which produce massive information is sale and distribution branch. If this information gathers and registers precisely, it is possible to achieve valuable information about company in order to make strategic decisions. Sale and distribution system has a products storage system and a customer service system. Customer service system controls financial affairs between company and customers. In the companies with massive and various sales and distributions, processing and keeping manual information for making on time decision is very laborious and make problems for company. Rapid progress in computer knowledge especially in data bases make it possible to create related systems regarding sales and distributions.

Meantime it is possible to save time, decrease costs and increase speed. Therefore, the object of this research is management information system design in sale and distribution and its economical analysis. Case study in this research is Darugar Company. In this regard, methods of changing manual sale and distribution system to a computerized system were designed then the result of designed system in performance steps were evaluated regarding different factors such as speed, accuracy, cost and user satisfaction.

This paper has forth parts: first part is introduction, second part is related to a sale and distribution computerized system presentation, third part is belong to System evaluation, theories, information gathering and system economical analysis and last part is deals to conclusion and suggestions.

2- A sale and distribution computerized system presentation

2-1- Sale and distribution manual system

In each production and Business Company, sale and distribution is very crucial. In today's world every company which wants to grow and develop or at least survive must ability to sell and distribute its own products. This task needs information's knowledge, suitable view from

market, having proper system for gathering information and processes, and preparing on time and exact reports:
A sale and distribution system has several parts: 1- product storage 2- visitor 3- customer 4- bank or treasury
Every sale center has a product storage which has been nourished from Tehran center and from this place products have been sold and distributed. One complete cycle is when product enters to storage, sells to customer and registers in the offices. Product request happens orally from center by phone or sometimes it happens in writing, however it does not register at the end. Visitors contact sellers and write their request plus its quantity then return it to distribution center. According to visitors' sheets, sale factors are written then daily sale report is prepared. In this report, code, client name, factor number and factor amount are written manually, and some columns have been prepared for receiving money (cash, check, money order ...).Then exit sheet is proceeded individually (individual sheet) in which for every report proceeds separate sheet. According to this sheet, collectors get product from storage with sale reports, go to each area to give product and sale's factor, and then complete sale report in which they receive money. If product returns for any reason, related returnee d's column is marked and according to this column, returned sheet is created in order to return product to storage. After receiving these factors, collectors go to related bank in distributed center and pay funds to company's account and get receipt. Then collector gives bank receipt and sale report to distributed center. Finally, bank receipt with sale factors send to Tehran.

- If product does not sell for any reason, it will be sent to Tehran by rejected sheets (There are not any damages sheets).
- After delivering products from Tehran to sale center, if there are not enough products in each box, deficit sheet will proceed from warehouse.
- According to the whole sheets, storage man updates warehouse and employee calculates balances' accounts of clients.

- Collected information is sent from distributed centers to Tehran, and archived in Tehran center. However, manual system is not able to do particular process on the information because of its high volume.

2-2-Changing manual system to computerized system

After recognizing system's initiation, relation diagrams' forms, data flow diagrams and entity relation diagrams were drawed. Selected tools and environment have strong power in designing pages between program, user interface and data base.
Then programming language, database and report generator were chosen. Designing and preparing pages between program and user interface are in which user and operator did not feel any strangeness with new designed system.

2-3-Sale and distribution computerized system

Process is similar to manual system however the difference is when product delivers from Tehran to distributed center. Information enters to computer in deposit sheet that has been sent from Tehran and a storage receipt prints by computer which has a serial number. If product has been deficited, after again delivery, another storage receipt is printed by computer. Visited sheets which are completed by visitors give to operators for entering information to computer. This information can enter to computer as batch, at the end of day. Then control list is prepared for any error in typing. Afterwards we can print invoices at one time. After printing factors, there is a possibility of printing sales reports and completing columns related to code, client name, factor number and amount. There are daily sale reports which produce automatically by computer based on different sales which has its own serial number. Thereafter computer according to each daily sale report creates an individual sheet (exit sheet) that storage man according to this sheet can deliver product. This is important to pay attention that after these processes which still storage's asset and client's account have not been changed There are still possibility of correcting information (visiting sheet information) in case of visiting any error. After making sure that data is corrected, it is possible to print sale factors. Computer in one batch process makes storage as creditor by decreasing asset of it and clients as debtors according to factor amount. Collectors get product from storage according to exit sheets and deliver it with daily sale report and sale factors. Then related column is completed by collector in daily sale report according to type of receiving money (cash, check) and if product is returned, related column in daily sale report is completed as well. Collector gives receiving money to bank also gives receipt and daily sale report sheet to operator. Operators import information to computer according to columns which was completed by collectors like type and amount of receiving money. Actually, clients' statuses are changed as creditors.

- If product has not been sold for any reason, It would be returned to Tehran with rejected sheets.(Actually there are not any losses sheets)
- After delivering product from Tehran to sale and distribution center, if there are fewer products in some boxes, it is possible to create deficit sheet by computer.

  According to these sheets, computers update storage and calculate clients' accounts on line in details and it is possible to visit or print storage flow and clients' accounts with selected data.

All processes are done by office employees in manual system which there is a possibility of error in writing or calculating. On the other hand, in computerized system there is a possibility of error just at the time of data entry of visiting sheet or receiving fee which by controlling system it is reduced compare to manual system. In addition, in computerized system high speed processes is due to particular design (batch data entering and printing) that makes possible to do high volume of processes in short time. Diversity of reports is very low in manual system and it is limited to amount of sale by visitor (for Sale's commission), however in computerized system there are too many different reports which in short time, particular report prepares like:

- Amount of sale by visitor as total
- Amount of sale of every product by visitor
- Amount of sale of every product in each city
- Amount of sale of all products in each city
- Comparison of selling products monthly and amount of imported products to storage

Collected information is sent to sale branch in Tehran monthly or weekly by interment or external memory and is used in another system which is totally graphical for managerial reports. Manager just needs to select particular area by clicking mouse's button from Iran's map, and visits or prints selected report from reports on the screen related to distributed center. This system was designed in which every manager access to all information of sale centers just by selecting subject.

3-System evaluation

3-1-Theories explanation, information gathering, sampling way

Darugar Co. has 800 employees and the statistic society to study is 160 which they work with sale and distribution system. For computerized sale and distribution evaluation and its economical analysis compare to manual system, four below theories were considered and they have been answered by test application of differences between averages and estimating the differences between averages.

1-Designed system in execution step needs less time, how much could be this decrease compare to computerized system?

2-Designed system in execution step has more accuracy, how much accuracy could be this increase compare to computerized system?

3-Designed system in execution step has less cost, how much could be this decrease compare to computerized system?

4-Did designed system satisfy users?

Field sampling has been taken by completing questionnaire and rangy also sample volume was 30 which it was proved less reliance interval and high accuracy.

3-2- Main variables in theories exam

Time

In these days, speed and time are crucial in doing tasks such as products' sale and distribution, therefore customer's request, product's distribution and creating sale's factor must be answered in minimum time and maximum speed.

Doing these steps particularly whenever distribution is done in central cities, is very time consuming by manual system. Even though fast processes in computer is because of new system design such as: Data entry and printing batch information, it is possible to process numerous volume of information in less time, on the contrary of manual system.

Accuracy

In today's knowledge, goal is to solve systems' problems. In an organization which all processes and calculations are done by office employees in manual system, there are errors in writing and calculating. These errors will be increased due to variety of complicated steps in doing tasks at manual system. Mean time large volume of processes cause to increase these errors also nobody can ignore recapitulation of exhausting processes by human. Because of these reasons, computerized system is able to control large volume of information and processes with top accuracy. If an error happens in computerized system, at the time of receiving funds and data entering in visit sheet, this error will be much less than manual system because of controlling list and other controlling

systems which in the computerized system is available.

Cost

Sale and distribution system is costly by manual system. In computerized system just one person with one computer is able to answer all sale and distribution tasks with minimum error, cost and maximum speed.

User satisfaction

Information's numerous volumes, complicated processes, too many errors, low speed and too much cost were caused managers to create computerized system. Computerized system not only creates user friendly atmosphere for operators, but also it creates on-line reports and fast processes for management information.
In this regard, tables are created based on answering questionnaires related to time, accuracy, cost and user satisfaction average. It is possible with calculating 30 questionnaires regarding above variables between manual and computerized systems complete statistical specification table based on time, accuracy, cost and user satisfaction variables for using them in related assumptions exam in future.

3-3- System statistical and economical Analysis

In table number 1 below page , average and standard deviation for all variables in manual and computerized systems were calculated individually.

Source: It organized based on averages' tables of time, accuracy, cost and user satisfaction

	Computerized Standard Deviation	Computerized Average	Manual Standard Deviation	Manual Average
Time	0.05	0.08	0.12	2,21
Accuracy	0.01	99,97	1,84	49.56
Cost	1.97	69.90	6.08	179.10
User Satisfaction	2.06	97.47	6.75	40.07

Table 1: Statistical characteristics for time, accuracy, cost and user satisfaction variables.

3-3-1-Averages Difference Test:

For this test related to each variable, we have below assumptions:

Mx = Average Variable in Manual System
Cx = Average Variable in Computerized System

For test we have Mx = Cx which is main assumption and we represent it with H0 On the contrary of above, there is alternative assumption like: Mx > Cx or Mx < Cx
Which we represent them with H1

H_0: Mx = Cx
H_1: Mx > Cx

Average difference test indication is based on formula (1):

$$t = \frac{\bar{X}_{M_x} - \bar{X}_{C_x}}{S_p\sqrt{\frac{1}{n_{Cx}} + \frac{1}{n_{Mx}}}} \quad (1)$$

$X_{Mx:}$ Average Variable in Manual System
$X_{Cx:}$ Average Variable in Computerized System
We have S_p from formula (2):

$$S_{p^2} = \frac{(n_{Cx}-1)S_{Cx^2} + (n_{Mx}-1)S_{Mx^2}}{n_{Cx}+n_{Mx}-2} \quad (2)$$

Freedom degree for this test is: $n_{Cx} + n_{Mx} - 2$
$n_{Mx:}$ Statistical society's volume in Manual System
$n_{Cx:}$ Statistical society's volume in Computerized System

Economic analysis for management information system (sale and distribution) were done by collected information in Darugar Co. with test of four assumptions.

3-3-2-Estimating the difference between averages

For all variables, estimating the difference between averages is done by below formulas:

$$d = \bar{X}_M - \bar{X}_C$$

$$Var(d) = Var\,(X_{M\_}X_C)$$

$$Var(d) = Var\,(X_M)_+(X_C) - 2COV(X_{M\_}X_C)$$

Due to the fact that variables are independent in manual and computerized systems, therefore we can write formula (3):

$$Var(d) = Var\,(X_M)_+(X_C) \quad (3)$$

Therefore variance of sampling differences between averages is shown in formula (4) and $S_{\bar{d}}$ is presented in formula (5):

$$Var(\bar{d}) = \frac{Var(d)}{n} \quad (4)$$

$$S_{\bar{d}} = \sqrt{\frac{Var(d)}{n}} \qquad (5)$$

If we assume that computerized and manual systems have equal variances, for estimating we use variances relating to weight average which is S_P^2, therefore we have equation (6):

$$P\left(d - tS_p\sqrt{\frac{2}{n}} \le \delta \le d + tS_p\sqrt{\frac{2}{n}}\right) = 1-\propto \qquad (6)$$

For example, Averages Difference Test and estimating the difference between averages for Time variable in manual and computerized systems were calculated according to below:

Averages Difference Test for Time variable:

$H_0 : M_t = C_t$
$H_1 : M_t > C_t$

$$S_{p^2} = \frac{(30-1)0.122 + (30-1)0.052}{30+30-2} \Longrightarrow S_{p^2} = 0.00845 \Longrightarrow S_p = 0.09192$$

$$t = \frac{2.21-0.08}{0.09192\sqrt{\frac{1}{30}+\frac{1}{30}}} = \frac{2}{0.02373} = 89.759$$

Amount of t in t table with 58 freedom degree in meaning surface of 0.01 is equal to: 2.3924. However, amount of t from above test is 89.759 and it is more than t table therefore it is in refused area and main assumption (H_0) is refused in meaning surface of 0.01. As a result, necessary time for Completing one distribution and sale cycle in manual system is much more than computerized system.

Estimating the difference between averages for time variable in manual system and computerized system is calculated according to below:

d= 2.21 -0.08
d= 2.13
t = 2.3924

$$S_d = 0.09192\sqrt{\frac{1}{30}+\frac{1}{30}}\ , \text{n=30}$$

$$P[2.13 - 2.39(0.09\sqrt{\frac{2}{30}} \le \delta \le 2.13 + 2.39(0.09\sqrt{\frac{2}{30}})] = 99\%$$

$$P(2.13 - 0.056 \le \delta \le 2.13 + 0.056) = 99\%$$
$$P(2.074 \le \delta \le 2.186) = 99\%$$

In computerized system with 99% probability, necessary time for completing one sale and distribution cycle is between 2.074 till 2.186 hour less than manual system. On the other hand, in manual system with 99% probability, necessary time for completing one sale and distribution cycle is 2.13 hour more than computerized system. This is very interesting to know that because of very low confidence distance (2.074, 2.186), accuracy of this estimating is very high and sample's volume has been enough.

4-Conclusion and suggestions

4-1-Conclusion

After doing theories exams by means of information and test application of differences between averages and estimating between averages were cleared that, computerized system has lots of advantages compare to manual system. In computerized system accuracy, user satisfaction and speed indexes were increased and cost index was decreased. One important notice is that reliance distance is so low regarding each index which proves that accuracy of this estimation is very high also sample's volume has been enough.

Estimating the difference between indexes with 99% reliance which proves above increase and decrease related to one distribution and sale cycle is in below:

Time index was decreased 2.074 hours till 2.186 hours in the computerized system compare to manual system. Accuracy index was increased 49/607% till 51/212% in the computerized system compare to manual system. Cost index was decreased 106/41 Toman till 111/99 Toman in the computerized system compare to manual system. User satisfaction index was increased 54/33% till 60/47% in the computerized system compare to manual system.

In addition of above matter, designed system has ability to generate different reports and charts which was not available in manual system and mangers have access to the information which have not had access before.

Another important matter is that designed system in addition of working in the single user operating system, can execute in multi users. This facility makes multi users' data entry simultaneously. It is possible to use different users for data entry and use different printers simultaneously. In summary, based on these systems: Managers can be independent regarding information and they are able to do lots of jobs alone with software also they can use this system as a decision support system which able them to make decision simply and confidently.

4-2-Suggestions

Due to communication knowledge improvement with using computer such as Internet, it is possible to design system that latest charts, statistics and information be available on-line for internal and external customers on the web. By this way, it is possible to improve company's activity for better import and export. For this goal, it is possible to use programming languages like Java, J2ee, Asp.net and Oracle application with a relational database system (RDBS) such as Oracle and MS SQL Server, to design web pages and applets for better communication. This new system could be platform independent and run in every operating system like Windows, Linux, Unix and apple mackintosh.

This cost difference will be cleared especially when communication happens with abroad, also with designing other on line related systems such as accounting, warehouse, production planning, payroll, budget, human resources, filling and archiving systems on the web and design one reporting system for decision support managers we can achieve a total MIS in the web. In addition of above matters, suggest that because of intranet improvement, design an intranet inside organization which each branch can use web pages for sending information to other branches of same organization or other organizations. We can use Web-based information system (WIS) in this regard for marketing information system.

Due to multimedia improvement which can transmit sound and image in computer, it is possible to make attractive computerized system. With these tools and a video conference can have meeting with long distance.

Mean time with having an advance archiving system which works on the web, it is possible to have scan documents and communications. These systems can store high volume of documents. We are going forward to paperless office with advance communication systems.

Unfortunately, with lack of up-to-date knowledge, in existing organizations, improving new idea is very low. For fixing this deficiency, organizational managers must use experts in order to have meetings and conferences with managers and employees till implementing of these new systems encounter with less problems.

5-References

[1] Salehi Sadaghiani, Jamshid, Ebrahimi, Iraj (2009). Estimating the difference between averages, *Advanced statistical analysis*, 4th Edition. 28-36

[2] Kroenke, David M. (2009). Managing development. *Using MIS*, Upper Saddle River, NJ: McGraw-Hill/ Pearson Prentice Hall, 3rd Edition, chapter 10, 1-68.

[3] Jessup, Leonard M.; Joseph S. Valacich (2008). *Information Systems Today*, 3rd Edition, Pearson Publishing. 416 - 417.

[4] Ralph M. Stair, George W. Reynolds (2003). Knowledge management and specialized information system, *Principles of Information Systems*, 9th Edition. 441-442.

[5] Whitten, Jeffrey L., Bentley, Lonnie D., Barlow, Victor M. (1994). The phases of the system development life cycle. *System Analysis & Design Methods*. Irwin INC.,development Toppan Co.3rd Edition,101-102.

[6] Oz, effy (2006). Data base and Data ware house. *Management information system*s. 5th Edition, 216-217.

[7] Aryanezhad, M. B., Zahabion, Mohamad (1992). Estimating the difference between averages**.** *Introduction to probability and utilized statistic*, Iran University of Science and Technology, 20th edition, 207-219.

[8] Momeni, hoshang, (1996). Management information system. Etehad, 190-220, 317-320.

[9] Madhoshi, Mehrdad, (1995). Management information system. *Designing management Information system*, Mazandaran technology and Knowledge University, Babol. 15-45.

[10] Farshad, Mehdi, (1993). System recognition. *Systematic view*, Amir kabir, Tehran. 42-61.

[11] *Lucas*, H. C., J. *Baroudi* (1994). The Role of Information Technology in Organizational Design. *Journal of Management information system*, *Vol*. 10, *No*. 4, *9-23.*

[12] Zakeri, Batool, Behnami, josef, (1998). Data flow diagrams or Data process Entity models, *Analyzing structural ways and designing information system, Industrial Management Organization*, 3rd Edition, 153-206.

[13] Industrial engineering group, (1990). Designing warehouse information system. *Designing warehouse information system (Industrial branches)*, Sharif Industrial University, 4th Edition, 29-87.

[14] Nofresti, Mohammad, (1997). Estimating the difference between averages in two societies. *Statistic in Business and economic*, Rasa, 4th edition. 59-62.

[15] Beheshtian, Mehdi, (1994). Management Information System. *Management Information system*, Bonyad , 100-117, 445-488, 501-505.

NeoMovingLife Garden

Raj Shankar

R & D Division, Panacea Infotech & Consultancy Pvt Ltd(Old) / (YOU-ME & TRIUMPH (Private Individual Enterprise Company), Kathmandu, Baghmati, Nepal)

NeoMovingLife Garden — If Disney's have the revenue problem, who is left ? We heard it recently, if you are following the news. Indeed we are talking about places that belong to fun for kids. Then we never thought we will be digging deeper to find the root cause. I mean literally. When the cost of operations is significantly high and cost of total ownership is equally high we look at the sky for science to help us. Cut the cost for me please. I mean, please come up with an efficient and cost effective attractive product for places where recreational activities are available.

Mechanics and electronics are two branches of science that sometimes go hand in hand. However, computer science allows us to solve problems involving these two areas of science by aiding with intelligence. Let's call it parellex for the sake of simplicity. The concept is pretty simple.

It is moving x square inches of soil at a fixed angle. When soil moves, the objects on the soil can move in specific direction which can be just opposite to the direction of the soil movement. Since it is all about parallex to deceive the eyes and senses of viewers and those experiencing it, objects on the soil gets moved from the coordinate x,y to x1,y1 replacing what was there at x,y earlier.

We need help of an automated process to monitor the heart beat, blood pressure etc. for a good reason. We know what happens when blood pressure in the brain of a person goes down to a specific level. In other words, he is asleep.

Here is how it works. You go to a fun place such as Redwood forest, Yosemite, hotel chains and/or Ghost Garden/Orchid. You get them to sleep in your Ghost Garden. You get them to sleep in the night and monitor their blood pressure remotely. You can also move their beds to go under/beneath another tree. Ascertain when and whether or not they are asleep. Use Neo Blood Pressure Remote Monitoring Tool. Remind them of their weather.

Move their physical body together with the tree to another location while they are fast asleep. This time the adjacent tree is a Redwood Tree rather than a pine tree.

Lets call it an artificial intelligence module that will ascertain the sleeping condition that allows a human to get a sound sleep very quickly based on the heuristic knoledge. It will be a self improving software module that will use a Relational Database Management Server to store data that will be used for comparision. The schema of the RDBMS will be pretty simple to store the basic variables that will be changed.

While the artificial intelligence module will be able to know the body condition of a human, they will be asked to fill a questionnaire based on which the AI module will control the Ghost Garden environment. There are ways to get someone to sleep real quick. It has to be without using a drug that might have a side effect even if takes longer.

Vaccums will be used internally to ensure a sound proof internal for the metallic platform.

Keywords:
RDBMS : Relational Database Management System

AI : Aritificial Intelligence

Neo Blood Pressure Remote Monitoring Tool : A micro controller based electronic device(Hybrid Computer)

Hybrid Computer : A special purpose digital and analog computer

Neo Controller : An intelligent monitoring and controlling device

1 Introduction

Let's talk about force required to move heavy objects from d1(initial distance, say 0) to d2(final distance, say 100m) with a displacement of 90 m²

D1=0m
D2=100m
Dimension of the movable soil beaneath the tree=LBH=15mX15mX15M
Dimension of the Metallic Platform including the soil D1 to D2=20mX20mX20m
Let's consider the following. When we move the object O1 (Tree + Soil + Metallic Platform) from D1 to D2 there has to be encapsulation of the internal implementation of the physical movement of the tree making you believe that only the object O1 move from D1 to D2 physically. Abstracting these details means physical replacement of one or more objects in the garden (Parent Object). In other words, smaller trees or plants in dimensions such as 10m X 10m X 10mX can replace the O1 which, means D1 to D2 has to replaced with objects at D3,D4 and D5. Besides, there can be alternate routes. Let's call them individual path's for every object. Snaking through the curves or not so straigth path is easier to implement with hydraulic multiplication. There is a good news. The height of the O1 and O2 can change if the height of the surface from the sea level is different for both the objects. After all it is a garden, and the surface doesn't have to be flat.
Force Multiplication = 1 :9
Trade Off : Depression :9 :1

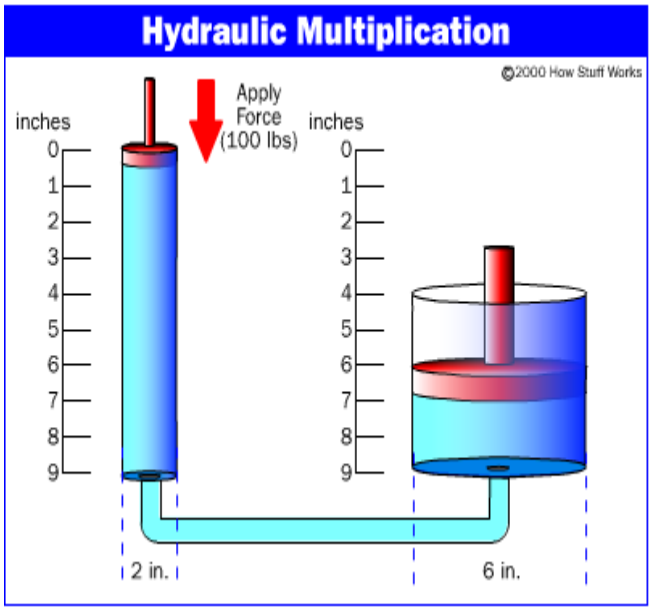


Figure 1 Hydraulic Energy

There is one more point. Just so that we discuss this in our complexities topic. We need to use an algorithm to compute the best path in terms of cost efficiency, time taken and shortest distance travelled. Shortest distance may not be the best solution all the time. Since this model can be used in very huge projects as well, the implementation will have to consider these factors. We will discuss very shortly how this solution is more cost effective than using initial combustion (energy) as a boost factor for mechnical force that is otherwise applied to push and move these heavy objects. Having said that, using mechanical energy is an alternate solution to this problem.

We will have to take a look at Force required to push the metallic platform. It's velocity V1 and velocity V2 will allow us to compute whether there is a consistent time involved in reaching a displacement as well as distance. Since the movement of the trees will generate kintetic energy there paths and time taken are of significant importance. A wrong path can mean very small or large breakdown of elements around the trees that can result in soil erosion among other things. Grasses to protect the soil erosion will be a must.

Energy Source : 2 H ₂ O + One time heater (Energy never dies)

The concept of steam energy is not new. What is important however is the fact that we can retain the energy produced either in gaseous or in liquid state. This solution will focus mainly on preserving the energy produced by breaking the chemical bonds of the Hydrogen and Oxygen ions and letting them remain far from the outer most shells of their respective atomic structure. The good news is that the atoms and molecules of this compound is not as strong as a solid. In other words, they will require less energy to break the chemical bond.

The process of conduction and convection involves looking at the flow of energy from higher to lower pressure zone. However, the external temperature will have a significant impact on the chemical bond. If there is lower temperature near the area that occupies the hydrogen and oxygen ions, the nearest outermost shell of a hydrogen or oxygen element will grab them and make them the part of the same compound.

Lets assume that there are six objects and each one of them need a force of x Newton or multiple of X Newtons. Once enough energy is generated at O1 to give initial momentum to the Hydraulic Machine, the same energy in gasesou state will be passed on to O2. The energy will then move onto O3 to O6. A thermal insulation of the paths will take care of the amount of energy produced since you can lose energy in your way. All six paths will have thermal insulation. However, there will be additional energy reserve for correction to the thermal insulation which will be realesed in the system with the help of a timer. It is like six objects moving in their individual paths for ever.

Neo Blood Pressure Remote Monitoring Tool:

One nano-technology based micro-controller device on the bed[s] of those in the ghost garden for every person is fixed. You can use a small range infra red client as part of another micro-controller device that propogates the information to the control device inside a building. The microcontroller on the tree can have a technology as simple 802.1b. A 8051 compatible ADC chip ADC804 transducer (Sensor) with a powerful range of pressure sensing will do. Conceptually the result is the same as taking the results by providing right inputs i.e, the blood flow from the heart and from the vains and measuring them while a mean is derived by a process called oscillometric detection and measurement of mean arterial pressure. ADC simply takes the analog signals and converts them to electrical signals processed and displayed on a digital LCD display.

Since the person sleeping in the garden can't be predicted to have his body in a one fixed position on the bed, it is only safe to keep the measuring device behind his head somewhere. An infrared device will shoot the place between arm and elbow and increase the temperature of the body by sending sound waves which will be picked up by the thermister sitting next to the ADC804. The 8051 will pass on the input to the ADC8041 with distance and displacement relayed. The sensor will then read the pressure. Alternatively another chip (nano technology based) can be used to shoot to the arm of the person sleeping in the garden which will read the pressure and pass it on to the microcontroller. Bringing the chip can be as easy as using a suction pump. It is not as unrealistic if you consider the use of a stepper motor with a connector such as ULN2003 integrated with XTAL based timer that takes the input from the controller to initiate the movement

of the objects in the garden. It is as easy as getting the same results using a thin robotic hands scanning the pressure.

Controller: Controller is a special purpose computer that receives input from the 8051 through its serial ports. It is like any other computer which is good in processing data. It can be as simple as dual processor based motherboard that takes two way input-output from the 8051. The design implementation of this solution begins with a simple model that has all the feature of the "NeoMovingLife Garden".

Rather than depending on an LCD as an interface to the 8051, a computer LCD flat screen with 6 different windows each displaying the real image of O1-O6 and its surrounding is required. Besides, the real time movement is visible on the screens. This ensures that every object in the garden is fast asleep when it all moves. Good news is that there will be vaccum used to keep the process sound proof so that they do not hear the sound. Having said that the objects should not feel the inertia that can wake them up. Their individual beds can be lifted one inch from the surface so that they do not feel the movement. Besides, there has to be cushion for suppressing triggers to body tissues from the platform. There are shock absorbers on each four corners of the bed that will distribute the shock to the exterior of the bed saving any jolt to the person sleeping in the garden. It maintains a state machine for the AI object to deduce useful inferences. The AI object relies solely on this information to self improve. The controller fetches this information through the Software object. Information in the table below shows the registers of the 8051 controller.

Register holding metadata for the State Machine (Table 1)

Regi-sters	Alarm Type	Alarm Level	Trigger	State
P0	Pressure	0-6	Interrupt 0-6	0-6
P1	Heat	0-6	Interrupt 0-6	0-6
P2	Distance	0-6	Interrupt 0-6	0-6
P3	Current	Not Implemented	Not Implemented	0-6

AI Object: The solution is self improving. The heuristic knowledge acquired by the controller is stored in a persistent storage that derives intelligent results from it, and make updates to the system as appropriate. For instance if the average sleep time for 2000 people in the garden is 20 minutes once they go to bed, the module to soothen their body and mind will kick in. The thermister reading the body temperature of the person will change the temperature of the surrounding rather than offering a constant temperature that may not suite the need of an individual.

AI module will monitor the changes in the state of the body of 2000 people and will derive intelligent inference. That knowledge will be used by the controller to configure other attributes of these objects in a typical OO jargan. The AI module will monitor the blood pressure and body temperature to know if the temperature configured by it helps. The AI module will configure a range of temperatures for general and special cases. Undoubtedly, the heuristic knowledge will do wonders for it.

AI module will interpret data filled in manually by the individuals entering and leaving the garden. AI Object will have object factories that they can understand and create new objects based on data deciphered from the database object. For instance if there is a rain outside, how will the body react to it. The AI object will use the <Change> factory object to instantiate a new object if does not exist already and change the attributes of the new object.

This object will be self improving only if it has more and more inputs and it has ability to interpret it and hence the external input object instantiation. In other words, if it does not find useful information to make a decision, a questionnaire factory object will be instantiated and filled by a qualified human. There will be two defaults in its intelligence that can comfort or discomfort the objects in the garden. Since experiments of this sort can have significant impact on the performance of the garden the change has to be subtle. On a sign of discomfort the AI object will stop.
The controller object is responsible for communicating with the 8051 controller and compatible chips.

Imagine, why a tree will be slightly heavier when it rains. Will this factor affects the energy requirement ? Will it change the angle of elevation for D1 ? How will it affect the entire system ? While these factor can be implemented using static attributes a dynamic attribute calculation is more prudent.

What about the quality of overall communication in the system ? The AI object will use these factors to make intelligent decisions. A fogy weather is a good example. It will become more interesting to manage the system when there are heavy winds espcially changing the centure of curvature at a curve.

Software Object: It is segregated from the AI module just to ensure that configuration and monitoring objects can be called by the AI object.

Matrix for object configuration/Monitoring (Table 3)

Object	**State**	**Store for AI**	**Automatic action A1**	**Input driven action A2**
1…n	1…n	1…n	1…n	1…n

There are >=16 objects to begin with excluding contained objects. Since these objects are layered, it is important to get a real time response and hence their availability in the memory.

The data is color coded and displayed using graphical interface for the simple reason that it will be

easier to manage for the person managing it. A historical and current report will give a fair overview of someone questioning the self-improving ability of the AI object which is responsible for using the same data and deciphering it to draw meaningful results.

Database Object: Relational databases are required to store the data and provide data redundancy. It is objects loaded in the memory from the database (persistent storage) that declares and defines the attributes and behavior of the controller object. Once loaded into the memory they are allowing the AI object to improve constantly. The schema of the database will use the attributes of every single object in the solution. In other words, these are the storage for objects in the memory. It is more of an object database than of a typical tuple and row/column thing. Tuple and rows are used to store the attributes of the objects. A query for normalizing the database and a network diagram to show the relationships of the tables and views will be used. A framework that will launch the stored procedure can have the set of rules (AI) to normalize the database.

Let's keep it simple. Enforce referential integrity with one primary key. No duplicate fields. Use views instead, even if it costs more memory and performance. Let's call them rules and scan through all the tables of the given database to enforce the rule. There are 12 rules by default in the RDBMS world, the database object has to focus on maintaining only two that I mentioned. RDBMS will take care of the rest. It is not about implementing a trigger, it is about allowing a schema to implement a trigger.

Why SQL ? Coz, you want to be consistent across all RDBMSs.

There are way two many scenarios in which a collission between two objects is possible. Besides, trees grow in size. A separate calculation is required that will put additional load on the amount of energy required. It is all computed by the solution. Let's further understand that a human body will not be lying idle in the most convenient position for the solution to gather results. In other words, more and more permutations and combinations are required. When the entire garden is in the state of motion, change of 1 degree angle to an object can have an impact on the integrity, reliability, scalability and fault tolerance of the system.

Matrix for Test Case Computation (Table 2)

Test case (T)	Physical state of tree1 (S1)	Physical state of tree2 (S2)	Autom -atic action A1	Input driven action A2
1…n	1…n	1…n	1…n	1…n

$$F(S1,S2)=T \quad (1)$$

$$T=((S1+S2)!/r!(S1+S2-r)!)-((S1!/r!(S1-r)!)+(S2!/r!(S2-r)!)) \quad (2)$$

For security reason these objects allow receiving or sending messages through limited number of public methods.

The 8051 is a passive delegate that passess on the messages to its listeners. Since it is an embedded system implementation, it does not required the listerns to register with the 8051.

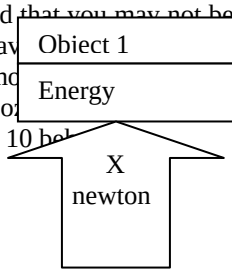
The controller implements an event driven priority based first in first out queue. The controller, software, AI and database together are an independent framework that can be integrated with any other system that requires such framework.

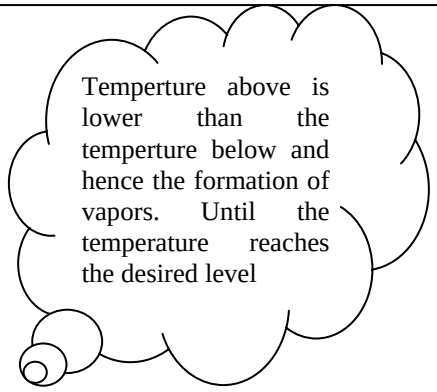
NeoFramework : As far as the difference in behavior is concerned, an attribute and behavior will instantiate classes into objects at run time and use declaration based on the ini file while definition based on the same ini file. To keep the framework loose an ini file is required.

The AI module will have compiler implemented to parse the declaration and definition requirements. An XML file is the way to go. The semantic and grammar will depend on the input from the XML file. The parser will look into its database to match all they key words so that typos can be avoided. Meaningful error generation means, you have error codes with error messages. The event viewer (application) will record the error message and detailed log file will record the error messages with time stamps. If this framework is to be used in several situations, a documented set of interfaces can be exported based on which one can extend the featureset. However, the objective is to let the user use it as is and hence the complexity.

A matix of supported behavior can be found in separate file based on the available interfaces and chips compatible with 8051. In other words, whatever features the chips support, the framework supports them. This is to ensure that you do not have to predict what the next solution will be needing from the framework. Since more on chips is beyond the scope of this solution in the context, the framework will allow add ons that can be used at run time by the next project solution. There are variety of chips 8051 supports. While it looks extensive work, the features are really limited.

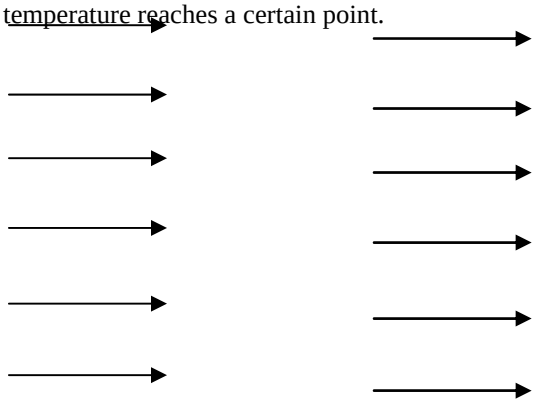
An event listener will be implemented to look for any additional requirement and the implementation will require the framework to load individual definitions from separate dll files. It is as simple as adding one additional card in the system, receiving an event, loading the corresponding definition file and integrating it to the system. Having said that you may not bee needed to load all the supported behav arser can reduce the amount of memo ding the required bahavior. This is co dozen cards and all you need is 8 to 10 bel



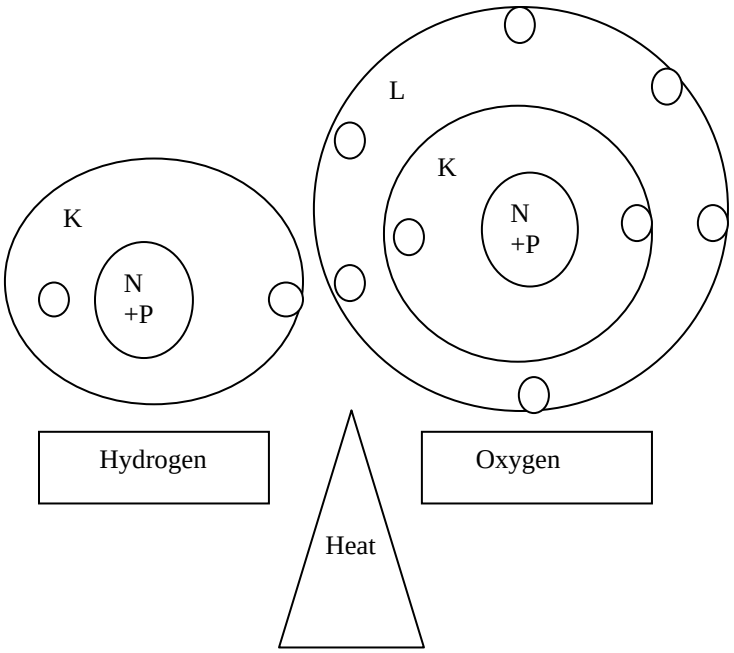


Reservio r for releasing the vapors at will

Way to chamber 2 (For applying force to next object i.e. Object 2)
The vapor comes through an opening that opens after the total temperature reaches a certain point.



Breaking the Barrier – Electrons from the outermost shells will leave a positively charged electron. In other word a field of negatively charged ions will be created.



Object 1: ObjectParent (It is very open and lose model and does not enforce abstract behaviors to be implemented by the child objects)
Object 2: LivingObject (Derives from ObjectParent)
Object 3: Human(Derives from LivingObject)
Attributes: Sex, Body Temp, Blood Press, Height, Age, Medical Condition
Object 4: Tree (Derives from LivingObject)
Attributes: Length, Breadth, Height, Mass, IncreaseInMass
Object 5:Non-LivingObject(Derives from Object Parent)
Attributes: Length, Breadth, Height, Mass
Object 6: ComputerChip (Contained in Computer object [Software,Hardware,Firmware]) ((Derives from Non-LivingObject)
Attributes: type, …
Object 7: MicroController (Derives from Computer Object)
Object 8: Software (Derives from Computer Object)
Object 9: Database (Derives from Software Object)
Object 10: AI (Derives from Software Object)
Object 11: SchemaNormalize(Derives from AI Object, Derives from Database Object)
Object 12:NeoBed(Derives from NonLiving, Derives from bed, Contains Computer Chip – Polymorphic
behavior of the bed object)
Object 13-15: Metallic Platform, Hydraulic Energy, Neo
$2H_2O$ Energy
Object 16…:

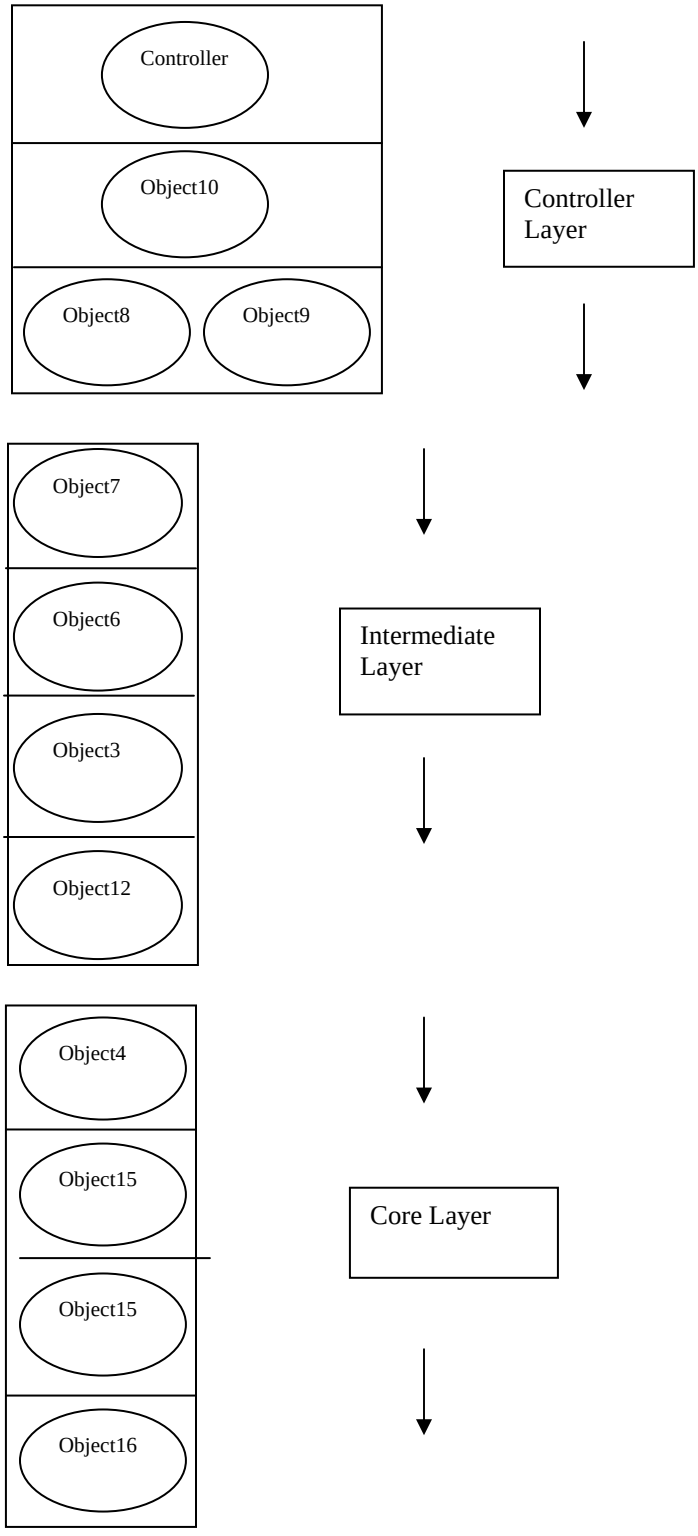


Figure 2 : Layered Architecture

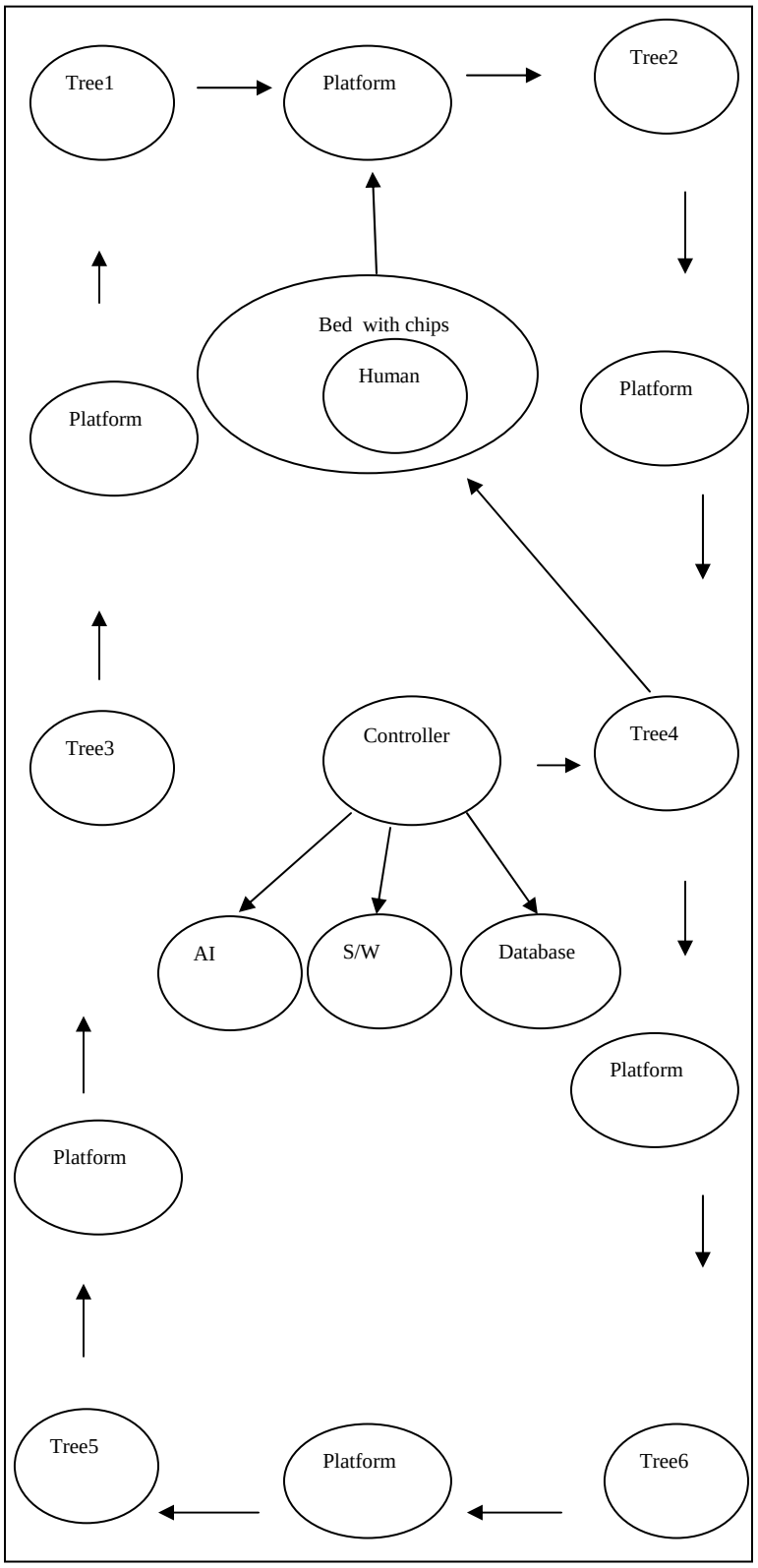


Figure3: Objects in the Garden

Complexities:

The implementation involves knowledge of various fields of science. Electrical, Civil/Construction, Computer, Electronics, and Natural science etc etc. One of the complex issues that you need to deal with is the fact that when an object moves at x velocity it has an inertia. A sudden stop can act as trigger to external body tissues that can wake him up. Hence the movement has to very slow. Since the entire garden will be moving it will be as good as us moving on the earth surface while earth itself is moving. Even difficult challenge is to simulate this when a human enters the garden. It is a moving garden at all time. In other words, it is no less than earth simulation as far as centre of gravity, gravitational pull, pressure on the body is concerened. The model allows us to have fun. Lots of fun but it has greater challenges in implementation so that it looks easy for you as an end user.

Intelligence of the controller limit has only one limit – intelligence of human brain. With this kind of intelligence we can safely state that the model is evolving and is in the state of constant improvement. Error for correction requires mathematical compuation to be very precise. The individual fields such as computer science will require very skilled engineers from the field of hardware, software, and networking which include use of existing drivers for communicating with interfaces. Besides, you will be needing database, AI, OO, C++, Win32, Assembly, 'C', network protocol design, network protocol development, network administrators, application programmers, SNMP programmers on both agent and manager sides. All this makes it a very complex project. Indeed you will need a very experienced management team to handle the project.

The best path detection also means avoiding collission course. Suppressing the noise and external trigger on human body that is in sleeping state involves challenges. The energy solution involves hours and hourse of meticulous lab test result evaluation. There are 12 main layers of data flow and each of these layers are intelligent. Almost all of them have some or other kind of intelligence. Each of these layers have bidirectional data flow and the data integrity is of utmost importance. Besides, these layers use their own individual protocol which may or may not be understood by every layer and hence the controller layer. Scientific calculations involve velocity, distance, momentum, angular momentum, inertia, Hz calculation for sound waves, redirection of sound waves with reduced amplitude through designated solid path, volumes, mass, force, and pressure etc. It is not that big of a challeng except that the stepper motor will require lot of power for lifting 100-200 pounds, which, the robotic hand can wait to do until a human arrives on the bed.

Formulae :

Pressure P is the function of force applied per square meter. (1)

A hardware simulation is warranted before actual production starts. This is due to the number of hardware, software and protocols involved. One wrong result can mean either revision or replacement of the instrument such as the one responsible for generating and controlling the constant energy.

Angle of elevation from the plane surface need to be calculated to compute the amount of energy required to move in each possible direction. The higher the height the more the need of the energy hence need for co-ordinate geomatory, trigonomatory and torgue computation for angles at which the pressure(Force/Sq M) will be applied. These mathetical computations will determine the amount of energy needed. Besides, the best case scenario of having a plane surface may not be a choice in a garden. The computation for curves brings additional variable.

Impact: The impact is on energy sector. There is a huge impact on the energy sector. There is difference now in the way we used to have fun. Let's talk about human body control and management. It is a good beginning. We now know that nano technology based chips, micro-controllers and sophisticated analog and digital devices and can work together in one system solution. Interfaces and kind or number of chips are not limited by the design and hence it allows us to make our system feature rich and robust. There is a hidden impact on the way we look at software applications. The new focus can be on intelligence. We have been talking about pattern recognition and data mining for over a decade now. It is a good move in that direction. It can have impact on space flight simulation programs where you will be allowed to get trained in avoiding objects moving in orbits and changing your course besides deciding the best path to take. Furthermore, the days are not far when you will have 13 ships travelling to different satellite of Saturn and/ man-made satellites docking in their individual orbit. Ships will be calculating the orbit and path collission courses.

2 Conclusion:

This solution is letting us move forward in various areas of science. It will be a as good as a virtual reality for one night which will be so real to experience. The line between virtual and real is really getting blurred here coz it is a parallex not virtual reality. The idea is not to deceive your senses but to let you experience science for fun. In other words, we can safely conclude that this solution will attract lot of attention and hence it worth its while.

3 References

Internet

SESSION

ASYNCHRONOUS DESIGN + MULTI-CORE PROCESSORS + FPGA + HPC + CIRCUIT DESIGN

Chair(s)

Prof. Hamid R. Arabnia

Robust Asynchronous Carry Lookahead Adders

P. Balasubramanian
School of Computer Science
The University of Manchester
Oxford Road
Manchester M13 9PL, UK
padmanab@cs.man.ac.uk

D. A. Edwards
School of Computer Science
The University of Manchester
Oxford Road
Manchester M13 9PL, UK

H. R. Arabnia
Department of Computer Science
University of Georgia
415 Boyd Building
Athens, Georgia 30602-7404, USA

ABSTRACT
Novel gate level synthesis of robust asynchronous carry lookahead adders based on the notion of section carry is discussed in this paper. For a range of higher order addition operations, the carry lookahead adder is found to exhibit reduced latency than the carry ripple version by 38.6%. However, the latter occupies less area and dissipates less power compared to the former by 37.8% and 17.4% respectively on average.

Keywords
Asynchronous design, Carry lookahead adder architectures, Logic synthesis, Delay-insensitive codes, Standard cells.

1. INTRODUCTION
Carry lookahead (CLA) adders represent a widely used high-speed carry propagate scheme for performing addition in logarithmic time [1], unlike the ripple carry adders (RCAs). The CLA adder is based on the principle that by examining the augends and addends, it is possible to predict the carry signals beforehand and thereby reduce the latency (longest path delay) that could be expected in a stage-by-stage propagation scenario. In this work, we consider gate level robust asynchronous realizations of novel CLA adder architectures and evaluate their benefits in comparison with the fundamental RCA structure that employs a cascade of full adder modules.

The de-synchronization approach [2], which is based on the dual-rail combinational logic style, provides an opportunity for straightforward translation of the synchronous CLA adder architecture into an asynchronous format, with the extra requirement of completion detectors to ensure robustness. This, however, would incur at least thrice the area penalty of a synchronous equivalent. Reference [3] presents a full-custom design method to implement fast robust asynchronous CLA adders by utilizing dual-rail data encoding. In this work, our focus is on the direct synthesis of robust asynchronous CLA and combinational adder logic, based on the multi-level extension of a heuristic recently proposed for synthesizing robust asynchronous equivalents of combinational logic descriptions [4]. We consider homogeneous (dual-rail) and heterogeneous (dual-rail and 1-of-4) delay-insensitive data encoding mechanisms. The final physical realization of the robust asynchronous CLA adders is done in a semi-custom design style, corresponding to the proposed novel architectures. The proposed CLA adders are synthesized using the standard cells of a 130nm bulk CMOS process, and satisfy Seitz's weak-indication timing constraints [5], with the CLA logic being early propagative.

The rest of the paper is organized as follows. An analysis of the problem of circuit orphans that manifests in an asynchronous style recursive carry synthesis is dealt with in section 2. The proposed robust section carry based CLA (SCBCLA) adder architectures are discussed in section 3. Also, the simulation results corresponding to 2-bit and 4-bit lookahead based 32-bit robust asynchronous CLA adders are presented. In the next section, a comparative evaluation of asynchronous CLA adders and RCAs with respect to 32, 48 and 64-bit addition operations is performed. Finally, the conclusions are made in section 5.

2. BACKGROUND
Let us consider a few scenarios to illustrate the problem of gate/wire orphans resulting in an asynchronous style recursive carry synthesis. With a and b being the adder inputs and cin representing the input carry, the basic equation governing the carry output, $Cout$, is expressed in single-rail format as,

$$Cout = ab + (a \oplus b)\, cin \quad (1)$$

$$G = ab,\ P = a \oplus b \quad (2)$$

In (2), G and P signify the generate and propagate signals. An output carry is generated if both the operand bits are 1; if the adder inputs are mutually exclusive, the incoming carry is simply propagated to the output. With notations P_i and G_i denoting the generate and propagate functions of an adder stage i, we have,

$$C_i = G_i + P_i C_{i-1} \quad (3)$$

Equation (3) can be thought of as a second-order equation, since G_i and P_i can be further expressed in terms of the primary inputs of an adder stage. Unwinding the recursion implicit in (3) would yield the following i^{th} order equation, where C_{-1} represents the carry input to the least significant adder stage.

$$C_i = G_i + P_i G_{i-1} + P_i P_{i-1} G_{i-2} + \ldots + P_i P_{i-1} P_{i-2} \ldots P_0 C_{-1} \quad (4)$$

Many asynchronous logic designs employ a delay-insensitive data encoding scheme and a return-to-zero handshake protocol to achieve robustness [6]. Among the generic family of delay-insensitive codes [7], the dual-rail and 1-of-4 coding schemes are widely preferred on account of their high coding efficiency and ease of translation between traditional binary switching functions. With dual-rail data encoding, a signal line x is represented using two wires: true-bit – x^1 ($x1$), and false-bit – x^0 ($x0$). A transition on x^1 represents a 1, while a transition on x^0 signifies a 0, however both x^1 and x^0 are not allowed to be high simultaneously since the coding scheme is unordered (i.e. no code word forms a subset of another code word). The spacer state refers to the condition when

both x^1 and x^0 are 0. In return-to-zero handshaking, a spacer state is present between two valid data states.

Representing (3) in dual-rail encoded format, we have,

$$C_i^1 = G_i^1 + G_i^0 P_i^1 C_{i-1}{}^1 \quad (5)$$

$$C_i^0 = G_i^0 P_i^0 + G_i^0 P_i^1 C_{i-1}{}^0 \quad (6)$$

In fact, (5) and (6) incorporate certain logic transformations to satisfy the monotonic cover constraint [8], which implies that the product terms of a sum-of-products expression should be mutually orthogonal [9].

The possibility of the occurrence of gate and wire orphans is clarified through the following discussions. In general, an unacknowledged transition is construed as an orphan. Figure 1 depicts the carry output synthesis using generate and propagate signals, as specified by (5) and (6), where (a_1^1, a_1^0), (b_1^1, b_1^0) and (C_0^1, C_0^0) represent the dual-rail augend, addend and carry inputs of an adder stage and (C_1^1, C_1^0) signifies the dual-rail carry output generated from this stage. In the figures, the C-gate[1] is represented by the AND gate symbol with the marking 'C' on its periphery. In figure 1, *is*1 and *is*3 represent the true and false-rail expressions of the carry-generate signal, while *is*4 and *is*2 correspond to the true and false-rail expressions of the carry-propagate signal. For the condition when the carry-generate function becomes valid, the following sequence of transitions occurs: ($a_1^1\uparrow$, $b_1^1\uparrow$) $\rightarrow$ *is*1$\uparrow$ $\rightarrow$ $C_1^1\uparrow$. Even if the transition $C_0^1\uparrow$ occurs, since the intermediate output signals *is*3 and *is*4 do not fire, the transition at the gate output node *is*2 (*is*1$\uparrow$ $\rightarrow$ *is*2$\uparrow$) gives rise to a gate orphan and $C_0^1\uparrow$ results in a wire orphan.

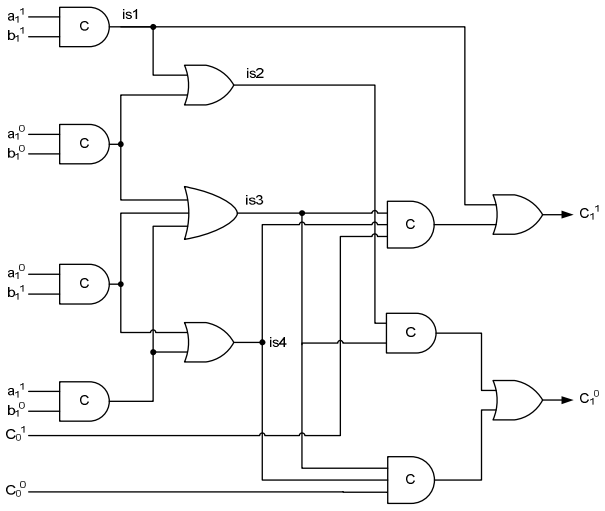


Figure 1. Asynchronous carry output synthesis based on generate and propagate functions

Alternatively, the carry equations can also be represented taking into account the carry-kill condition, apart from generate and propagate conditions, which signifies the state when both the augend and addend inputs of an adder stage assume logic 0. The carry-kill condition prevents the generation of a carry signal from an adder stage and also avoids the propagation of an input carry to the output.

$$C_i^1 = G_i^1 + G_i^0 P_i^1 C_{i-1}{}^1 \quad (7)$$

$$C_i^0 = K_i^1 + K_i^0 P_i^1 C_{i-1}{}^0 \quad (8)$$

The logic realizing the above equations, based on carry-generate, propagate and kill conditions is depicted in figure 2. Here, *im*1 and *im*4 represent the true and false-rail expressions of the carry-generate signal, while *im*5 and *im*2 correspond to the true and false-rail expressions of the carry-kill signal. The equation pertaining to the carry-propagate condition is realized by the intermediate node *im*6. Let us consider a worst-case scenario to describe how orphans could arise in this circuit. It can be seen from figure 2 that ($a_1^1\uparrow$, $b_1^1\uparrow$) $\rightarrow$ *im*1$\uparrow$ $\rightarrow$ $C_1^1\uparrow$. The following sequence of transitions also occurs: *im*1$\uparrow$ $\rightarrow$ *im*2$\uparrow$. But the transition on the intermediate gate output *im*2 will not be subsequently acknowledged by the output *im*3 for a transition on the carry input C_0^1. This leads to the creation of gate and wire orphans in the circuit.

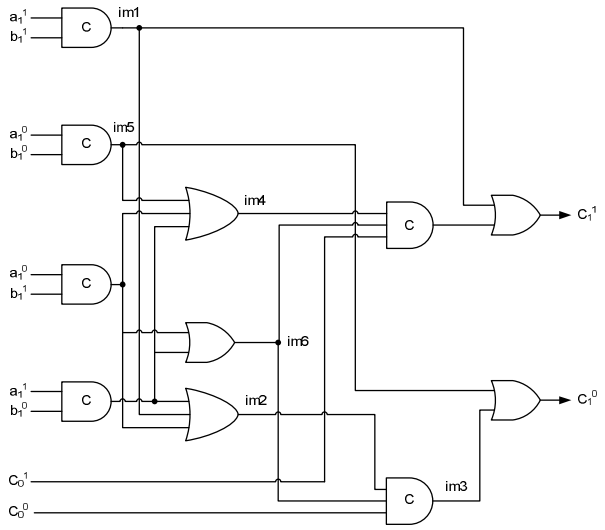


Figure 2. Asynchronous synthesis of output carry logic on the basis of generate, propagate and kill functions

The preceding discussions have demonstrated the problematic issue of gate orphans and/or wire orphans, possible in case of a recursive asynchronous carry synthesis. However, the problem with recursive equations can be overcome if the essential logic transformations to satisfy the monotonic cover constraint are rather performed on a first-order equation. This would therefore necessitate reduction of a k^{th} order carry equation to the first-order that would actually involve unfolding cube expansions. As a consequence, the need for stage-wise propagate and generate signals is deemed unnecessary.

It has been deduced through the principle of mathematical induction that the irredundant disjoint sum-of-products expression of the encoded carry output of a q-bit CLA unit, featuring mutually orthogonal product terms, would consist of a minimum of $(2^{q+1}-1)$ logical conjunctions with the support set of the

[1] The Muller C-element outputs a 1 (0) if all its inputs are 1 (0), otherwise it retains its steady-state.

product of maximum dimension comprising $(2q+1)$ literals. Owing to an exponential relationship between the CLA size and the resulting number of orthogonal product terms, CLA logic sizes in case of a robust asynchronous implementation would have to be restricted so as to gain the maximum benefit in terms of latency on the basis of the proposed CLA adder topologies. Nonetheless, the physical realization of CLA logic of any bit-size would be practically feasible on the basis of speed-independent logic decomposition.

3. SCBCLA ADDER ARCHITECTURES

Two asynchronous CLA adder architectures have been conceived, while acknowledging the spatial demand of a robust asynchronous implementation, and they are discussed in this section.

3.1 Type 1 Architecture

The Type 1 architecture bears some similarity with a block CLA adder featuring intra-group carry ripple [1]. However, it mainly differs in that the propagate and generate signals corresponding to each single-bit adder stage need not be computed – hence the term 'section carry'. Figure 3 portrays the dual-rail encoded Type 1 configuration of the section carry based CLA (SCBCLA) adder.

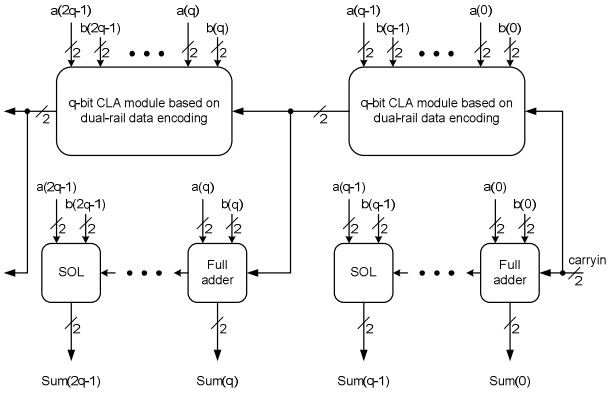


Figure 3. Type 1 SCBCLA adder architecture incorporating dual-rail encoding

The q-bit CLA module generates a CLA signal corresponding to a section/group of q-bits of the adder operands taking into account the carry input from a previous group/section. The CLA signal corresponding to a section is used to feed the subsequent CLA module in the cascade and also the next adder element in the sequence. The acronym 'SOL' expands as 'sum only logic', which accepts an augend, addend and a carry input and processes them to produce only a sum output. Thus, the sum outputs of the adder can be produced simply by a rippling of the carry signal within each section, while the carry output of a section can be produced simultaneously and be quickly passed onto the next section to generate its corresponding lookahead carry output signal. As a result, there arises an opportunity for optimizing the CLA logic at the expense of the sum producing logic, i.e. the sum outputs of a section can assume the collective responsibility of indicating all the input data operands of that section (weak-indication). Hence, the CLA unit corresponding to the adder sections can be freed from the indication (acknowledgement) constraints, allowing them to be early propagative while guaranteeing that the CLA adder is gate-orphan-free. The phenomenon of early propagation means that all the primary outputs (valid/spacer) could be produced even with the arrival of a subset of the primary inputs (valid/spacer).

The Type 1 CLA adder structure for a hybrid input encoded data path is shown in figure 4, which features a combination of dual-rail and 1-of-4 codes. The 1-of-4 code is used for encoding the augend and addend inputs of each adder stage, while the input, output carries and sum outputs are dual-rail encoded. The 1-of-4 encoded values of single-rail inputs given in Table 1 represent one of various possible encodings and a random assignment is shown in the Table below, which is also considered for this work.

Table 1. Data representation based on dual-rail and 1-of-4 delay-insensitive data encoding schemes

Single-rail data		Dual-rail encoded data		1-of-4 encoded data			
a	b	($a1$ $a0$)	($b1$ $b0$)	$e0$	$e1$	$e2$	$e3$
0	0	(0 1)	(0 1)	0	0	0	1
0	1	(0 1)	(1 0)	0	0	1	0
1	0	(1 0)	(0 1)	0	1	0	0
1	1	(1 0)	(1 0)	1	0	0	0

According to the Type 1 topology, for an n-bit adder comprising q-bit CLA units, $\left(\frac{n}{q}-1\right)$ CLA modules would be required as CLA signal generation is necessary until the penultimate adder section.

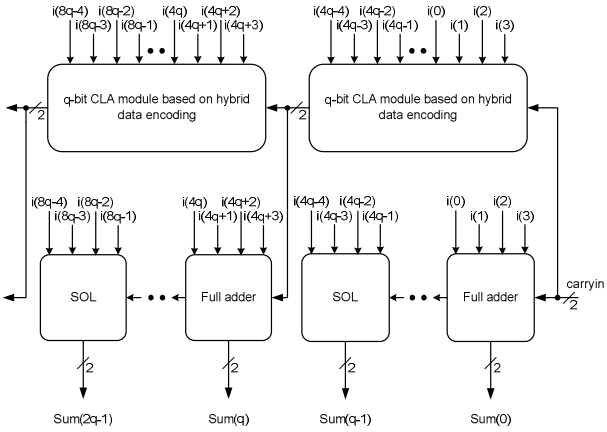


Figure 4. Type 1 SCBCLA adder topology based on hybrid (dual-rail and 1-of-4) input encoding

3.2 Type 2 Architecture

From a physical implementation perspective, it can be anticipated that substantial delay would be encountered in the least significant CLA unit, as opposed to the successive CLA units in case of the Type 1 architecture. This was indeed observed during simulations, where the critical path in case of the least significant dual-rail encoded CLA unit consists of AND4, CE2[2], 3 OR2 gates (5 CE2,

[2] CE2 refers to a 2-input Muller C-element.

5 OR2 gates), while the critical path in the least significant hybrid input encoded CLA block comprises AO2222, CE2, OR2 gates (AND4, CE2, 2 OR3, 2 OR2 gates) for 2-bit (4-bit) lookahead carry signal generation. Since the latency of the least significant CLA module (2-bit or 4-bit) was found to be higher than what could be expected from a series cascade of individual adder sections, the least significant CLA adder section can better be replaced by a simple carry propagate adder section paving the way for a possible reduction in delay, area and power parameters. With this modification, the structure of the Type 2 CLA adder architecture would be as shown in figure 5 for dual-rail encoded data paths.

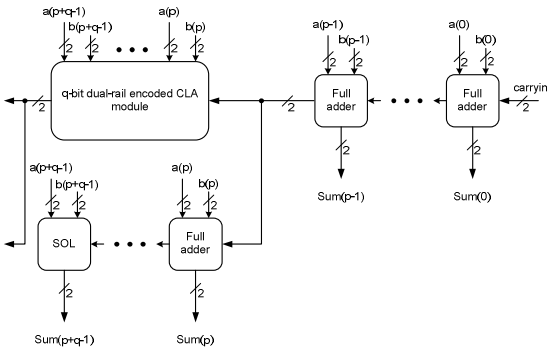


Figure 5. Type 2 SCBCLA adder topology utilizing dual-rail data encoding

The Type 2 SCBCLA adder topology basically involves a slight modification of the Type 1 SCBCLA adder structure primarily targeting latency reduction. As a result, the number of CLA units required would in general be specified by $\left(\frac{n-p}{q}-1\right)$, where n, p and q are assumed to be even. Here, p signifies the number of full adder stages present in the least significant positions of the adder topology. The Type 2 configuration for hybrid encoded data paths is portrayed by figure 6.

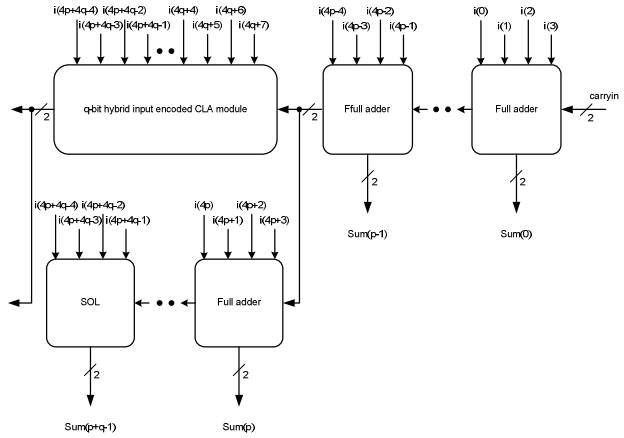


Figure 6. Type 2 SCBCLA adder topology employing hybrid input data encoding

3.3 Evaluation with 2-bit CLA

Simulations have been performed with a 2-bit CLA module embedded in both the Type 1 and Type 2 SCBCLA adder architectures. The CLA logic, full adder and SOL blocks were realized based on a number of self-timed design approaches [5] [10] [11]. Table 2 shows the delay, area and power parameters corresponding to the Type 1 topology, while Table 3 lists the same for the Type 2 architecture.

Table 2. Delay, area and power for 32-bit asynchronous addition based on Type 1 CLA adders with 2-bit CLA logic

Adder realization style	Delay (ns)	Area (μm^2)	Power (μW)
Seitz (dual-rail) [5]	13.7	17229	1168.9
DIMS (dual-rail) [10]	14.9	25245	1233.8
Toms et al. (dual-rail) [11]	10.2	14191	927.4
Proposed (dual-rail)	5.5	9016	755.0
Toms et al. (hybrid) [11]	7.2	12331	807.1
Proposed (hybrid)	**5.1**	**7593**	**659.4**

Table 3. Delay, area and power for 32-bit asynchronous addition based on Type 2 CLA adders with 2-bit CLA logic

Adder realization style	Delay (ns)	Area (μm^2)	Power (μW)
Seitz (dual-rail)	13.3	16593	1145.7
DIMS (dual-rail)	14.8	24273	1208.8
Toms et al. (dual-rail)	10.1	13479	906.1
Proposed (dual-rail)	5.3	8887	749.7
Toms et al. (hybrid)	7.1	12013	794.0
Proposed (hybrid)	**5.2**	**7529**	**656.8**

The proposed (hybrid) adder is found to exhibit the least latency, area and power parameters amongst the Type 1 and Type 2 CLA adder configurations. In comparison with other dual-rail and hybrid encoded adders, the proposed hybrid input encoded adder is preferable as it features optimized design metrics. By comparing Tables 2 and 3, it can be seen that the proposed (hybrid) adder belonging to the Type 2 architecture dissipates the least power and features minimal area occupancy, mainly due to the absence of a CLA module in the least significant stage.

Figure 7 highlights the size of the 2-bit asynchronous CLA logic synthesized based on different methods. It can be seen that the proposed hybrid input encoded 2-bit CLA block is an order of magnitude smaller than the dual-rail encoded 2-bit CLA modules and even in comparison with the strongly indicating 2-bit CLA unit based on hybrid input encoding – thanks to a significant shrinkage of the input state space. In case of Seitz's, DIMS or Toms et al. approaches, for the case of dual-rail encoding, the input space enumeration would be $O(2^9)$ and therefore the resulting CLA logic would be massive. It should be noted here that Seitz's method involves certain timing assumptions with respect to decomposing high fan-in OR gates of the completion detection logic and high fan-in AND gates of the data path logic, while the DIMS method necessitated speed-independent logic decomposition. Toms et al. method, however, involved extra logic optimization to minimize the latency of the adder realization.

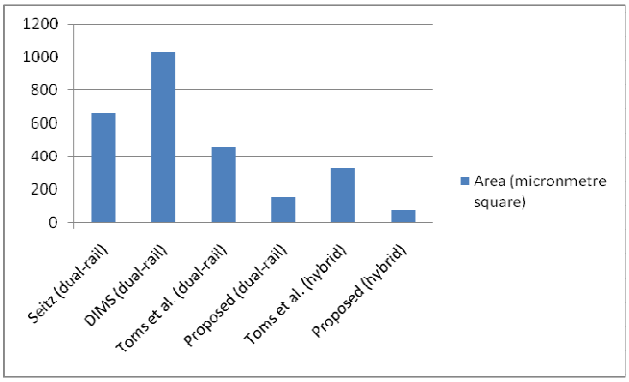


Figure 7. Relative comparison of area occupancy of different 2-bit asynchronous CLA modules

3.4 Evaluation with 4-bit CLA

This section investigates extending the lookahead size from 2-bits to 4-bits. The delay, area and average power values given outside brackets in Table 4 refer to that of the 32-bit asynchronous SCBCLA adder based on our proposed Type 2 architecture. The values mentioned within brackets in Table 4 correspond to the 32-bit robust asynchronous adder based on the basic carry ripple structure. It is evident from Table 4 that synthesis of CLA adders according to our proposed architectures, but on the basis of Seitz, DIMS or Toms et al. self-timed logic design methods, are found to be detrimental and exaggerates the design metrics – this is due to the greater input space demand as highlighted in figure 7.

Table 4. Delay, area and power for 32-bit asynchronous addition based on 2-bit CLA and carry ripple configurations

Adder realization	Delay (ns)	Area (μm^2)	Power (μW)
Seitz (dual-rail)	13.3 (6.5)	16593 (7689)	1145.7 (741.8)
DIMS (dual-rail)	14.8 (12.8)	24273 (10665)	1208.8 (770.5)
Toms et al. (dual-rail)	10.1 (10.6)	13479 (7561)	906.1 (627.6)

The results pertaining to 4-bit CLA based Type 1, Type 2 and mixed SCBCLA adder architectures are shown in Table 5. For the Type 1 and Type 2 asynchronous SCBCLA adders, there is no CLA module present in the most significant nibble position. But the mixed architecture incorporates an additional 3-bit CLA module in the most significant nibble position to effect a marginal reduction in delay, with the output of the 3-bit CLA block acting as the input carry for the most significant full adder stage. This only constitutes a peephole optimization that is accompanied by associated area and power overheads. From Table 5, it is clear that the proposed SCBCLA adder employing hybrid input encoding and utilizing a mixed architecture, involving predominant usage of 4-bit CLA units significantly benefits in terms of minimizing the latency (29.4%) over a 2-bit CLA based asynchronous adder counterpart. Thus comparing Tables 3 and 5, it is seen that this delay (critical path delay) reduction is achieved at the expense of an area overhead (1.3×) and a power penalty (2.4%).

Table 5. Delay, area and power for 32-bit asynchronous addition based on Type 1, Type 2 and mixed SCBCLA adder architectures using 4-bit CLA units

Adder realization style	Delay (ns)	Area (μm^2)	Power (μW)
Proposed (dual-rail) – Type 1	4.3	9769	770.6
Proposed (dual-rail) – Type 2	4.0	**9385**	757.3
Proposed (dual-rail) – Mixed	3.8	9601	765.5
Proposed (hybrid) – Type 1	3.8	10609	681.6
Proposed (hybrid) – Type 2	4.1	10041	**672.8**
Proposed (hybrid) – Mixed	**3.6**	10829	687.3

4. SELF-TIMED RCAs AND CLA ADDERS

It is worth studying the potential benefits of the proposed lookahead scheme for addition involving higher word widths, and the results of this analysis for adders of size 32, 48 and 64 bits are presented in Tables 6 and 7. We consider only dual-rail encoding for this purpose, and the mixed SCBCLA adder architecture referred to here is founded upon the original Type 2 architecture that extensively employs 4-bit robust asynchronous CLA blocks.

Table 6. Comparing delay and area of asynchronous RCAs and mixed architecture SCBCLA adders

Adder size	Realization style	Delay (ns)	Area (μm^2)
32 bits	Proposed (dual-rail) – RCA	5.8	**7081**
	Proposed (dual-rail) – Mixed CLA	**3.8**	9601
48 bits	Proposed (dual-rail) – RCA	8.3	**10611**
	Proposed (dual-rail) – Mixed CLA	**4.8**	14667
64 bits	Proposed (dual-rail) – RCA	10.9	**14129**
	Proposed (dual-rail) – Mixed CLA	**6.6**	19721

Table 7. Comparing total power of asynchronous RCAs and mixed architecture SCBCLA adders for higher word widths

Adder size	Realization style	Power (μW)
32 bits	Proposed (dual-rail) – RCA	**678.8**
	Proposed (dual-rail) – Mixed CLA	765.5
48 bits	Proposed (dual-rail) – RCA	**1011.4**
	Proposed (dual-rail) – Mixed CLA	1150.5
64 bits	Proposed (dual-rail) – RCA	**1351.1**
	Proposed (dual-rail) – Mixed CLA	1697.1

From the simulation results given in Tables 6 and 7, we infer that with respect to higher order addition, the proposed robust asynchronous SCBCLA adder exhibits reduced latency than the asynchronous carry ripple counterpart by 38.6%. However, the latter occupies less area and dissipates less power compared to the former by 37.8% and 17.4% respectively on average. Therefore the important observation here is that the proposed lookahead carry scheme facilitates a reduction in critical path delay by two orders over a simple carry propagate scheme, at the expense of increased power consumption by an order.

5. CONCLUSIONS

The general conclusion is that the proposed robust asynchronous CLA adder architectures (SCBCLA adder structures) enable significant reduction in maximum data path delay compared to the basic carry propagate adder at the expense of increases in area and power dissipation. The main reason for the increased area and power dissipation of the former is attributable to the asynchronous CLA logic realized in a robust fashion, which is associated with an exponential spatial demand. With respect to dual-rail encoding, the 3-bit and 4-bit CLA units are 1.6× and 2.7× bigger compared to the 2-bit CLA module; in case of hybrid input encoding, the 3-bit and 4-bit CLA units are 3.1× and 7.8× bigger in relative comparison with the 2-bit lookahead carry logic.

Also, it appears that increasing the levels of lookahead may not be feasible owing to the problem of circuit orphans that implicitly manifests in a recursive style asynchronous carry synthesis. This reason also appears to negatively impact the direct asynchronous synthesis of parallel prefix adders, where the prefix operation is also recursive. Nevertheless, such adders could be effectively realized following the block-level relaxation approach [12], which constitutes an indirect asynchronous logic synthesis strategy that relies on the proprietary null convention logic method [13] [14]. However, a preliminary theoretical analysis reveals that the proposed SCBCLA adder architectures are competitive with the block-level relaxation based parallel-prefix adders with regard to latency, and possibly area and power.

6. REFERENCES

[1] A.R. Omondi, *Computer Arithmetic Systems: Algorithms, Architecture and Implementations*, Prentice-Hall, 1994.

[2] J. Cortadella, A. Kondratyev, L. Lavagno, and C. Sotiriou, "Coping with the variability of combinational logic delays," *Proc. ACM/IEEE International Conf. on Computer Design (ICCD)*, pp. 505-508, 2004.

[3] F.-C. Cheng, S.H. Unger, and M. Theobald, "Self-timed carry-lookahead adders," *IEEE Trans. on Computers*, vol. 49, no. 7, pp. 659-672, 2000.

[4] P. Balasubramanian, and D.A. Edwards, "Self-timed realization of combinational logic," *Proc. 19th International Workshop on Logic and Synthesis (IWLS)*, pp. 55-62, 2010.

[5] C.L. Seitz, "System Timing" in *Introduction to VLSI Systems*, C. Mead and L. Conway (Eds.), pp. 218-262, Addison-Wesley, Reading, MA, 1980.

[6] J. Sparso, and S.B. Furber, *Principles of Asynchronous Circuit Design – A Systems Perspective*, Kluwer Academic Publishers, 2001.

[7] T. Verhoeff, "Delay-insensitive codes: an overview," *Distributed Computing*, vol. 3, no. 1, pp. 1-8, 1988.

[8] A. Kondratyev, M. Kishinevsky, B. Lin, P. Vanbekbergen, and A. Yakovlev, "Basic gate implementation of speed-independent circuits," *Proc. 31st Design Automation Conference (DAC)*, pp. 56-62, 1994.

[9] V.I. Varshavsky (Ed.), *Self-Timed Control of Concurrent Processes: The Design of Aperiodic Logical Circuits in Computers and Discrete Systems*, Kluwer Academic, 1990.

[10] J. Sparso, and J. Staunstrup, "Delay-insensitive multi-ring structures," *Integration, the VLSI Journal,* vol. 15, pp. 313-340, 1993.

[11] W.B. Toms, and D.A. Edwards, "Efficient synthesis of speed independent combinational logic circuits," *Proc. 10th Asia and South-Pacific Design Automation Conference (ASP-DAC)*, pp. 1022-1026, 2005.

[12] C. Jeong, and S.M. Nowick, "Block-level relaxation for timing-robust asynchronous circuits based on eager evaluation," *Proc. 14th IEEE International Symp. on Asynchronous Circuits and Systems (ASYNC)*, pp. 95-104, 2008.

[13] K.M. Fant, and S.A. Brandt, "Null convention logic system," *US Patent 5828228*, October 1998.

[14] K.M. Fant, *Logically Determined Design: Clockless System Design with Null Convention Logic*, John Wiley & Sons, 2005.

Reducing Thermal Hotspots in Multi-Core Processors using Core Migration Scheduling

Savithra Eratne, Claudia Romo, Eugene John, and Wei-Ming Lin
Department of Electrical and Computer Engineering
University of Texas at San Antonio
San Antonio, Texas, USA

Abstract - *The thermal management of integrated circuits is an increasing challenge, limiting the progression of technology, and making microchip designs complex. The research and development of thermal management has concentrated on reducing the total power consumption of microprocessors at the micro-architectural level. At this level, constraints such as power consumption and its sources are regarded highly. One of the most critical facets of power dissipation is hotspots, which are small localities with high power densities that degrade the microprocessor's efficiency and lifetime.*

Current VLSI technology comprises of microprocessors with multiple cores in a single chip. Multi-core processor technology utilizes two or more independent processors in a single chip in an effort to improve performance through parallel or distributed processing. Modern computers may utilize multiples of such chips. This paper proposes migrating cores as a method to reduce thermal hotspots in multi-core microprocessors with limited thermal management options. Simulations of the multi core microprocessors were completed with the use of multi2sim, McPAT and SPEC2000 benchmarks. Hotspot by HP was utilized to convert the power measurements to temperature, as well as for layout and thermal mapping of the microchip. Dynamic scheduling of cores increased the delay by 1.4% on average, while the peak temperature was observed to reduce by 6°Celsius.

Keywords: Peak-Power, Multi-Core, Microprocessor, VLSI, Power Dissipation

1 Introduction

Technology generations, following Moore's law, have exponentially increased transistor density and increasing the power density of microprocessors. Despite the reduction of dynamic power dissipation per transistor, the power density increases as the area is reduced by half in each subsequent technology increasing number of transistors per unit area. As a result of this exponential increase in thermal density, thermal cooling has become a critical aspect of microprocessor design. Cooling with mechanical means such as heat sinks and fans are still the primary solutions to this problem. Although these mechanical solutions are effective in large fixed computers, in modern slim mobile devices they are impractical.

Multi-core processors' throughput is much higher compared to a single core processor, due to parallel processing, resulting in an increase in thermal energy. On thermal terms, at the 65nm technology level a quad core processor reaches to a maximum temperature of 12° - 14° C higher than the single core microprocessor [1]. This temperature difference can generate hotspots that could impact the performance of the processor, resulting in long term degradation and short term decreased speed. In addition to the overall high temperature, the core-to-core thermal coupling is also a factor that will affect the performance of integrated circuits (ICs). Thermal coupling occurs when the lateral heat dissipation causes increase in temperature between adjacent cores. Sophisticated and expensive microprocessors manufactured by Intel and others are designed with built in power management features such as monitoring and shutting off the processors if they are in the path of destruction due to thermal runoff. There are many inexpensive multi-core processors such as Freescale octal-core, which can have improvements to their thermal power management. For example the Freescale octal-core processor has switching ability to change the state of each core-processor, but the controls in place have limited capacities. This paper presents a technique to reduce thermal hotspots in multi-core microprocessors with limited sources in thermal power management.

The remainder of this paper is structured as follows. Section 2 presents the contemporary techniques researched in reduce power in microprocessors. Section 3 describes the hotspots, migration, and scheduling. Section 4 proposes the new method of scheduling of the migration of cores to reduce hotspots. Sections 5 through 7 are the methodology, results and analysis, and the conclusions to this research.

2 Background

CMOS device technology has gained significant improvements in terms of performance and efficiency, but improvements in reduction of power dissipation and temperature regulation lag in progression. Power dissipation has increased rapidly in multi-core technology, mainly due to increased number of transistors as well as the smaller technology size.

Many direct and indirect techniques are deployed in reducing the power dissipation or removing excessive heat in microprocessors. The temperature of a microprocessor is primarily dependent on the power dissipation with respect to time. Techniques to reduce power consumption could lower the total thermal dissipation of the chip. These techniques were developed to protect the devices from thermal degradation, performance degradation or to improve the battery active life of mobile devices. There could be specific localities in the chip that would continue to experience high power dissipation. Such localities experiencing high temperatures compared to their surrounding area are called hotspots. Hotspots are a threatening phenomenon that causes serious device degradation. Hotspots are localities that can reach over 300 W/cm2 and result in varying temperatures across a chip surface. Such hotspots demand the utilization of exceptional cooling systems for devices such as modern servers that operate continuously: 24-hours a day [3, 4]. Hotspots are critical given that they could be destructive to the chip in the long and short terms. Therefore temperatures at hotspots are an imperative aspect in the design of cooling mechanisms. Designing mechanisms to reduce the occurrence of hot spots or to reduce the temperature in hotspots, will in turn reduce the cooling needs of the microprocessors.

The simple technique of using forced air to cool the microprocessors can be useful as long as the temperature of the environment is lower than the ideal ambient temperature of the microprocessor. In embedded systems, a heat sink and artificial air flow is needed beyond 20W of power dissipation. When the core runs at 7W a cooling fan is rendered obsolete [4]. Cooling and packaging technologies have not developed concurrently with the progression of device technologies, therefore limiting the power accessible to a multi-core processor [2]. The design of previous thermal solutions has been at the package and system level. Although many cooling techniques have been introduced in succession, air cooling has been the industry standard due to its efficiency and low associated costs [3, 4].

Prior research [5] proposed an approach of space exploration by which the centralized and nonpolitical hardware structure is replaced by a distributed architecture substrate structure by which the thermal hotspots can be reduced. This method requires redesign of the core layout for each core separately depending on the configuration of the layout of each core. The speed in each core could be impacted as their routing lengths will change. An alternative to reducing power dissipation is using multiple heterogeneous cores in a processor as proposed by [7], where the cores are of different complexities and performance. Multi-core processors can be classified based on the nature of the cores employed. The microprocessor is considered 'homogeneous' if the same processor is used as the cores, or 'heterogeneous' when different performance-cores are used together, the preference depending on the purpose of the microprocessor. The results were impressive, achieving 80% power reduction, but the impact on performance was 25% latency. The negative performance is attributed to the software dynamically selecting the most appropriate core. Identifying the matching core based on the complexity of the process is an extremely difficult technique to be automated.

Power gating and clock gating has become the most widely used technique of reducing power dissipation during idle/sleep states in multi-core microprocessors [8, 9, 10]. Contemporary high performance microprocessors come with advanced features in thermal power controls such real time temperature monitoring and warning systems. But there are many multi-core processors with few basic features like switches that enable the operating system to control the state of the microprocessor based on its usage, such that it could be placed in active, sleep, or states in between. The objective of multi-state core design is to reduce the consumption of power while keeping the core active so that states can be switched with a minimum transition delay.

Scheduling is a widely used technique in reducing power dissipation in microprocessors. Scheduling of multi-core processors exploits the benefit attributed to sharing resources among cores. There are many approaches to the technique of scheduling that includes scheduling threads, processors, and other resources in microprocessors. The scheduling of resources has expanded to the extent of scheduling virtual machines; whole microprocessors. Since the advent of multi-core processors, scheduling schemes have expanded to include minimizing latency as well as power dissipation.

There are two approaches to types of scheduling threads as resource-based and sampling-based. Resource-based techniques concentrate on distributing the load among the cores based on core parameters. Sampling-based techniques use the nature of the execution code or other parameters to identify the load distribution among the cores. Both symmetric and asymmetric multi-core processors can benefit from these techniques [11]. Virtual machine scheduling and migration can be used as an alternative to thread migration since virtual machines are a wide spread method of resource sharing. Virtual machines can be migrated by transferring the domain between two host machines by transferring its in-memory states seamless to the user [12].

Another method of optimizing performance at this stage has been through the technique of migration. Threads,

processes, and virtual machines are relocated from processors to processors in order to achieve better performance and optimize the utilization of resources. Scheduling threads and migrating threads are common among the multi-core processors as additional hardware resources are available within one chip. The primary objective of this technique is to obtain an optimized performance by utilizing all resources available [11]. Research has been concentrated on scheduling activities i.e. threads and processes in multi-core processor environment [1]. Migrating threads or processes does not cause large transition delays and they can be managed at operating system or software level. Testing and implementing the scheduling of threads and processes are relatively easy as these do not require changes in the hardware or microprocessor design, which are very expensive by nature.

All these techniques proposed are targeting the reduction of the overall power dissipation and not specifically hotspots, although there is some improvement in terms of overall temperature of the chip and thereby hotspots as well. There are considerable benefits in terms of cooling efficiency, if the temperature at hotspots can be reduced. These include increasing the throughput of the microprocessor and reducing power consumption.

The efficiency of migration in multi-core processors is increased when the cores are mesh-connected. Multi-core processors are designed with two primary layouts of cache, either as a mesh-based multi-core structure or a ring-based multi-core structure. Ring-based structure is followed with fewer cores per microprocessor and for increased number of cores they are arranged in a mesh-based structure. In the present state of technology, where the widely marketed processor is a quad-core, the layout can be of either structure.

3 Migration for Optimal Power Dissipation

Power dissipation in a core is at its lowest during sleep mode. When switched to active mode, the core starts dissipating power and the temperature in the core increases in relation to the number of cycles it has been active. As the temperature approaches the upper thermal limits of the microprocessor the frequency of the core is reduced. All cores in a microprocessor may not need to be active at the same time to perform its tasks at any given time. The present approach is to schedule and distribute the load between the cores so that none of the cores are dissipating power over the others in an attempt to reduced hotspots. The end result is that the microprocessor as a whole will be increasing its temperature. As the cores are stacked adjacent to each other one of the key issues that will arise is the core-to-core thermal coupling. The power dissipation in each core affects the operating temperature of its adjacent cores [1].

As a remedy to this situation we propose keeping a minimum number of cores active in the microprocessor. All inactive cores can be placed in the sleep state to reduce power dissipation. This approach will reduce the overall microprocessor power dissipation. Since high cycle counts of processing will occur in the few active cores, we propose migrating the cores so that hotspots will not occur in the active cores that would otherwise continuously active until the end of the process or the thread.

Schedule of the migration of cores should occur in a way that core-to-core thermal coupling does not occur. The basic parameters to observe are the physical distance between the cores and physical layout of the microprocessor. This allows the cores that were active in the earlier step to cool off while the processing is carried out in the other cores. This approach of switching cores is particularly effective in applications such as communications, networking and load balancing computers where the load varies throughout and the load is low at majority of the time. The drawback of switching cores is the inherent latency of migration. But since modern multi-core microprocessors use switch fabrics in place of buses and it allows for faster transitions between cores, the associated latency may not be of great significance.

The heat radiation is proportional to the temperature difference between the core and ambient temperature per Newton's law of cooling, given in Equation 1.

$$\frac{dT}{dt} = -k_1(T - T_a) \qquad (1)$$

Under forced flow of air, the heat convection or energy removed is proportional to the temperature difference between the core and the ambient temperature [4].

$$Q_{out} = k_2 (T - T_a) \qquad (2)$$

Therefore the temperature reduction rate is proportional to the temperature difference of the ambient temperature. The ambient temperature of a microprocessor in an air conditioned environment is around 45°C or 100°F.

Ideally the core to be activated should be the core which is lowest in temperature and is further from the highest temperature cores in the processor. That is, a core to be active next is inversely proportionate to the temperature of the other cores and proportionate to the distance.

At any given instance when the next core to be activated is j;

$$j = 1 \quad \text{where } t = 0 \tag{3}$$

$$j = i \quad \text{where} \quad \sum_{x=1}^{k} \frac{T_x}{d_{xi}^{n}} \Big|_{i=1}^{k} \min; \quad t > 0 \tag{4}$$

where d_{xi} is the unit distance between the selected core i and other core x, T_x the temperature of the core x, t is time, $n \in \{1,2\}$ and k the count of cores

Taking octa-core multi-processor as an example, assume the cores are arranged in two column by four row formation with switch fabric, the cores are identified from left bottom to right top. Distances are in units assuming all cores are homogeneous and lay equidistant to the adjacent cores. Distance calculations and temperature variations would be given by the following matrices. In which the temperature matrix assumes a maximum temperature of 100°C in the active cores; and ambient temperature of 45°C; k1 = -0.1; k2 =0.1.

Distance matrix:

$$D = \begin{bmatrix} 1 & 1 & 1 & \sqrt{2} & 2 & \sqrt{5} & 3 & \sqrt{10} \\ 1 & 1 & \sqrt{2} & 1 & \sqrt{5} & 2 & \sqrt{10} & 3 \\ 1 & \sqrt{2} & 1 & 1 & 1 & \sqrt{2} & 2 & \sqrt{5} \\ \sqrt{2} & 1 & 1 & 1 & \sqrt{2} & 1 & \sqrt{5} & 2 \\ 2 & \sqrt{5} & 1 & \sqrt{2} & 1 & 1 & 1 & \sqrt{2} \\ \sqrt{5} & 2 & \sqrt{2} & 1 & 1 & 1 & \sqrt{2} & 1 \\ 3 & \sqrt{10} & 2 & \sqrt{5} & 1 & \sqrt{2} & 1 & 1 \\ \sqrt{10} & 3 & \sqrt{5} & 2 & \sqrt{2} & 1 & 1 & 1 \end{bmatrix} \tag{5}$$

Temperature matrix:

$$T = \begin{bmatrix} T_1 & T_2 & T_3 & T_4 & T_5 & T_6 & T_7 & T_8 \end{bmatrix} \tag{6}$$

In calculating the suitability (S) for the, core to be activated next,

$$S = [1/T] \times [D^n] \tag{7}$$

The optimal core will have the highest suitability count.

$$\text{Next core j to be activated} = S[j] \Big|_{\max} \tag{8}$$

To identify the correct value for n in Equation (6) we simulated the temperature and the schedule of migration of the cores for n=1 and n=2. The value of n defines the order of the distance. At n=1 the distance is inversely proportional and at n=2 the square of the distance is inversely proportional to unsuitability. In analyzing it was concluded that n=2 gives a better performance scheduler than n=1.

In the case of a single task, the order in which to activate the cores are fixed to a given core layout in the microprocessor as the sequence depends only on the distance between each cores with the other cores. Unlike most scheduling techniques, that need real time calculations, this technique does not require the calculation or measurement of temperature and recalculations at each step. A simple lookup table can be used in switching the cores as the temperature of each core stabilizes after the first full rotation of the scheduling.

4 Methodology

Multi2Sim [13] was used to assess the functional components of the processes in SPEC2000 benchmarks. Multi2Sim is a multi-core, multi-thread environment simulator. It simulates the benchmarks for a given configuration. Simulation provides a wide range of precise parameters and their corresponding values. Statistics include simulation cycles, committed instructions, etc [13]. For the simulations we used the SPEC2000 benchmarks as the processes. Octal-core single thread multi-processors were simulated with the scheduling algorithm modified for the proposed core activation sequence. Migrations were carried out every 100,000 cycles.

McPAT [14] is an architectural modeling tool for CMP, providing accurate power and area for a selected microprocessor architecture of a given technology size. The statistics acquired from Multi2Sim is fed to McPAT to generate area and power statistics. An Alpha like microprocessor was used as the core in this simulation. Simulations were executed for 65nm technology. McPAT uses Hi-K metal gates for technology sizes from 45nm and higher. HotSpot [15] is a tool based on HP Cacti. The tool is primarily used to model the temperature of a microprocessor based on the power dissipation and the area of its components. The statistics needed for the simulation can be acquired from McPAT simulation which provides the power and area data. The steps required to calculate the temperature are carried out with the calculations of the layout of the components, which are based on the area, dimensional ratios, and interconnects between the components. The layout and the power data are simulated to generate the temperature of each component. A grid option for simulation is available to map the thermal variations of the chip surface.

A thermal map is the ideal tool in identifying the hotspots occurring on the chip. We used the thermal map for the chip as a whole to identify the hotspots and the impact of migration on those hotspots. Each of the simulators relies on the previous tool to provide the statistics necessary to carry out the simulations.

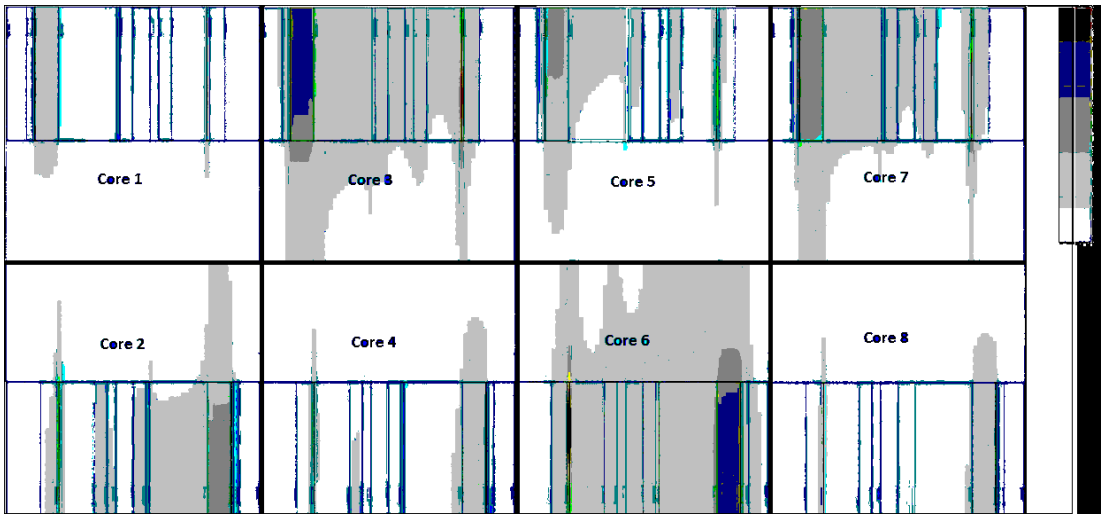


Figure 1: Thermal map of cores at the end of one rotation of migrations.

5 Results

Microprocessor temperature increases with the number of cycles the core was actively processing data according to figure I. This pattern also shows that if the processor can be stopped before the process is completed, and the process thread is migrated to a new core, the maximum temperature the original core will be reached can be limited to a lower temperature. The temperature reduction by this method depends on the process and the cutoff point used for the migration. For the benchmark aspi, if the migration occurred at 100,000 cycles, the peak temperature of the core will be 10% less than the peak temperature.

Table I: Migration delay in number of cycles for SPEC2000 benchmarks

Benchmark	Migration delay (Cycles)
apsi	1443
art	260
bzip2	6259
cc1	435
eon	305
fma3d	89
gzip	8013
mesa	300
perlbmk	330
twolf	171
vortex	558
vpr	106
wupwise	166

The trade-off of power saving in microprocessors is delay. The proposed technique inherently has a higher delay due to its migration of the process from one core to another. Table I lists the delay of migration counted in number of cycles. Migration was tested every 100,000 cycles. The delay depends on the benchmark and its usage of cache memory, as data migration from one cache allocation to another consumes considerable time delay. If the migration is executed at shorter intervals, the maximum temperature can be lowered but the delay percentage will be increased. Similarly if the migration is executed at a higher cycle cutoff, the delay percentage is reduced but the peak temperature will be higher and the power saving will be reduced. The delay with migration at 100,000 cycle migration would be fivefold of delay added with migration at 500,000 cycles. For this set of benchmarks, the average migration delay is 1.41%, if migration is triggered at every 100,000 cycles of processing with a standard deviation of 2.58%. Depending on the nature of the process the delay could be as high as 8% of the processing time.

Table II: Peak temperature of the microprocessor after activity cycles

Number of cycles	Temperature (K)
100,000	327
200,000	330
300,000	332
400,000	333
500,000	333

The power dissipated and the temperature of each component in a core processor changes with respect to its active time duration and the nature of the resources utilized in processing. Table II shows the variation of the highest temperature among the components in the processor. Cooling patterns of the 8-Core simulation is shown in Figure I have the cooling patterns for the scheduled cores with time laps. At 100,000 cycles per migration, the first core that was active

reached a temperature that would allow it be active once again by the end of a full cycle of scheduling. The core-to-core thermal coupling is minimal in this schedule as the activating cores are diagonal to the cores that are still warm.

6 Conclusions

Multi core processors are manufactured and utilized for wide range of applications. In most cases only one or a few of the cores are active while others are in inactive mode. We propose migrating the cores such that each core will be active for a short period of time while other cores are inactive. Migrating the cores will reduce the peak temperature at each core which in turn reduces the temperature of hotspots occurring in the microprocessors. Core scheduling is an alternative to scheduling of threads and processes which are in common practice at present. Reducing hotspots in microprocessors with limited thermal power management options will reduce system degradation and increase performance, resulting in more efficient and reliable use of multi-core processors.

In this research, multi-core scheduling was successfully simulated to reduce peak temperature up to 6° Celsius. The thermal map for an octal-core microprocessor with scheduling is presented with the delay of many SPEC2000 benchmarks. Temperature, power and Power delay Product variations for apsi benchmark is presented as an example. The performance of the benchmark depends on the nature of the process, the use of cache memory and intensity of the use of Arithmetic Logic Unit and Floating Point Unit.

Delay of the processor increased by an average of 1.4% for core migration for every 100,000 cycles as expected trade-off. The delay was observed to be small considering the overall improvement in performance. The thermal savings can be improved by migrating cores at shorter intervals with associated high delay penalties. Increase in number of cycles for the migration result in smaller temperature reductions.

7 References

[1] Janicki, M., Collet, J., Louri, A., Napieralski, A., Hot spots and core-to-core thermal coupling in future multi-core architectures, 26th IEEE SEMI-THERM Symposium, 2010

[2] Jayaseelan, R., Mitra, T., A hybrid local-global approach for multi-core thermal management, ICCAD 09, 2009

[3] Musoll, E., A thermal friendly load-balancing technique for multi-core processing, 26th IEEE SEMI-THERM Symposium, 2010

[4] Lundquist, C., Carey, V., Microprocessor-based adaptive thermal control for air-cooled computer CPU module, IEEE SEMI-THERM Symposium, 2001

[5] Prakash, M., Cooling challenges for silicon integrated circuits, Inter society conference on thermal performance, 2004

[6] Embedded multicore: an introduction, EMBMCRM Rev.0, FreeScale Semiconductors, July 2009,

[7] Cho, C., Poe, J., Li, T., Yuan, J., Accurate, scalable and informative design space exploration for large and sophisticated multi-core oriented architecture, IEEE International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems, pp.1-10, 2009

[8] Kumar, R., Farkas, K., Jouppi, N., Ranganathan, P., Tullsen, D., Processor power reduction via single-ISA heterogeneous multi-core architecture, 2003

[9] Weiping L.; Lei H.; Lepak, K.M.; , Temperature and supply Voltage aware performance and power modeling at microarchitecture level, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, , vol.24, no.7, pp. 1042- 1053, July 2005

Rao, R., Vrudhula, S., Chakrabarti, C., Throughput of multi-core processors under thermal contraints, International Symposium in Low Power Electronics and Design, 2007

[10] e600 PowerPC Core Reference Manual, Freescale Semiconductors, 2006

[11] Sibai, F., Simulation and performance analysis of multi-core thread scheduling and migration algorithm, 2010 International conference on complex, intelligent and software intensive systems, 2010

[12] Chen, P., Lin, C., Huang, S., Chang, J., Shieh, C., Liang, T., A performance study of virtual machine migration vs. thread migration for grid systems, 22nd International conference on advanced information networking and applications, 2008

[13] R. Ubal and J. Sahuquillo and S. Petit and P. L'opez, Multi2Sim: A Simulation Framework to Evaluate Multicore-Multithreaded Processors, Proc. of the 19th International symposium on computer architecture and high performance computing, October 2007

[14] Sheng Li, Jung Ho Ahn, Richard D. Strong, Jay B. Brockman, Dean M. Tullsen and Norman P. Jouppi, McPAT: An integrated power, area and timing modeling framework for multi core and many core architecture, MICRO 42: Proceedings of the 42nd annual IEEE/ACM international Symposium on Micro architecture, 2009, 469-480 pp

[15] W. Huang, K. Sankaranarayanan, R. J. Ribando, M. R. Stan, and K. Skadron. "Accurate, Pre-RTL Temperature-Aware Processor Design Using a Parameterized, Geometric Thermal Model" IEEE Transactions on Computers, 57(9):1277-88, Sept. 2008

Simple QPSK Modulator Implemented in Virtex 6 FPGA Board for Satellite Ground Station

Asraf Mohamed Moubark[1], Mohd Alauddin Mohd Ali[2], Sawal Hamid Md Ali[2]
Hilmi Sanusi[2] , Nasharuddin Zainal[2]
[1]Institute of Space Science (ANGKASA),
[2]University Kebangsaan Malaysia, 43600 Bangi,
Selangor, Malaysia

Abstract - *Modulation is a key feature commonly used in wireless communication for data transmission and to minimize antenna design. QPSK (Quadrature Phase Shift Keying) is one type of digital modulation technique used to transfer the baseband data wirelessly in much efficient way compare to other modulation techniques. Conventional QPSK modulator operates by separation of baseband data into i and q phases and then add them to produce QPSK signal. The process of generating sine and cosine carrier wave to produce the i and q phases consume high power. For better efficiency in power consumption and area utilization, 2 new types of QPSK modulator proposed. The proposed method will eliminate the generation of 2 phases and will produce the QPSK output based on stored data in RAM. Verilog HDL used to implement the proposed QPSK modulators and it has been successfully simulated on Xilinx ISE 12.4 software platform. a comparision has been made with existing modulator and significant improvement can be seen in term of area and power consumption.*

Keywords: QPSK, Verilog, Modulation

1 Introduction

Continuous growing demands from end user for more data have encouraged the engineers to develop many new types of modulation scheme in satellite communication system. New types of modulation technique introduced to increase the efficiency in data transmitting and receiving rate within the same bandwidth. One of the common modulation method used in satellite communication system is QPSK which is one form of PSK (Phase Shift Keying) modulation scheme [7]. In PSK modulation, the phase changed according to the baseband data while the frequency and amplitude remain unchanged. In QPSK quadrature means 4 different states that is used to represent a group of 2 bits input data. The four different inputs are 00, 01, 10 and 11 and each group takes one form of QPSK states as shown in table 1.

Table 1: QPSK phase with different input.

Input	QPSK phase
00	135°
01	225°
10	45°
11	315°

The other most related to QPSK modulation scheme is BPSK (Binary Phase Sift Keying) modulation. In more practical understanding, QPSK is formed from 2 separate BPSK which combined together. However, the data transmission in QPSK is twice when compared to BPSK and the Bit Error Rate (BER) over signal to noise ratio (SNR) for both modulation schemes are same [8]. The symbol period for QPSK is 2 times the bit period, $Ts=2Tb$ while for BPSK the symbol period is same as bit beriod $Ts=Tb$ [1,9]. Data transmission rate is very crucial for Low Earth Orbital (LEO) satellite system where the interaction time with earth ground station is very short. Implementing QPSK in full digital domain not only can save cost for long term but at the same time, it increases the wireless data immunity over surrounding noise [8]. The high configurability and MIPS (million instructions per second) in FPGA (Field Programmable Gate Array) have made the implementation of digital signal processing possible.

2 Conventional modulator

The conventional QPSK modulator operates by dividing the baseband data into 2 main streams, even and odd data. The divided unipolar data then changed into bipolar by using NRZ encoding technique. Continuously, the coded data will be mixed with carrier, which generated from DDFS (Direct Digital Frequency Synthesizer) or also known as DDS (Direct Digital Synthesizer) as shown in figure 1. The DDFS produced the sine and cosine as separate carrier wave signal and it made the mixing process much easier. The intended frequency use for transmission can be set while generating the DDFS core in Xilinx Integrated Software Environment (ISE) [11]. Xilinx provides the DDFS as IP CORE in ISE 11.3 and above version. The DDFS version 4 can produce up to 550

MHz carrier wave. After the process of carrier mixing, the odd data will known as I phase and the even data as Q phase.

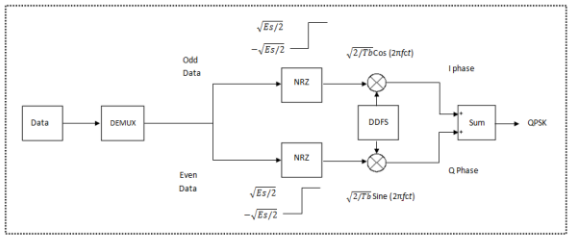


Figure 1: Conventional QPSK modulator block diagram

Table 2 shows the even and odd data represented by I and q phases. These two phases will be added to produce the QPSK signal. The general QPSK wave can be expressed as shown in equation (1) where i equal to 1, 2, 3 or 4, Es represent energy for a symbol,Ts represent period for a symbol and fc represent the carrier frequency.

$$QPSK = \sqrt{\frac{2Es}{Ts}} cos(2\pi fct + \frac{(2i-1)\pi}{4}) \qquad (1)$$

Table 2: NRZ coded odd and even data converted into I and q phase

odd Data	Even data	I phase	Q phase
-1	-1	$-\cos \omega_c t$	$-\sin \omega_c t$
-1	1	$-\cos \omega_c t$	$\sin \omega_c t$
1	-1	$\cos \omega_c t$	$-\sin \omega_c t$
1	1	$\cos \omega_c t$	$\sin \omega_c t$

3 Proposed QPSK modulator

The conventional method used to generate the QPSK signal consumes high power and area in FPGA. The proposed QPSK modulators are fully digital domain and produced QPSK output same as equation (1). The 2 new proposed QPSK modulators used RAM as main data storage to produce same QPSK signal as conventional modulator.

3.1 Proposed QPSK modulator 1

The first proposed QPSK modulator is designed in 2 steps. In first step, the above conventional architecture will be constructed just to collect different QPSK phase data for combinational input data. Once intended data collected, the conventional architecture will not be used for future QPSK signal generation. This will be major dynamic power saving since the DDFS will not be used to generate the 2 different carriers. Collected QPSK data will be stored in 4 different RAMs. Each RAM will store data for one QPSK phase. The multiplexer will be used to choose the correct RAM's according to the splitted input data. Table 3 shows the associated RAM's with different inputs and phases. Figure 2 shows the block diagram for the first proposed QPSK modulator.

Table 3: QPSK phase data stored in 4 Different RAMs according to input data.

RAM	QPSK Phase	Input Data
RAM # 1	135°	00
RAM # 2	225°	01
RAM # 3	45°	10
RAM # 4	315°	11

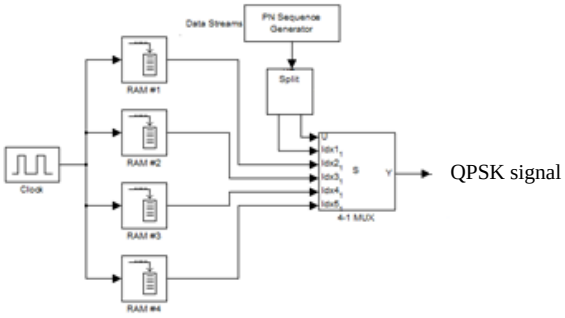


Figure 2: Block diagram for proposed QPSK modulator 1.

3.2 Proposed QPSK modulator 2

As for the second QPSK modulator architecture proposed, further reduction have made from the first architecture. Only 1 block RAM used instead of 4 RAM as in first design. The general equation for QPSK as in (1), give us information that the whole QPSK wave can be represented by a single phase shifted cosine wave. A single cosine wave with different phase representing one symbols in QPSK modulation. With that info, data to generate a cosine wave stored in RAM and for different input data, a pointer assigned at specific RAM address to retrieve the exact phase that needed to generate QPSK wave. Figure 3 show the second proposed QPSK modulator block diagram.

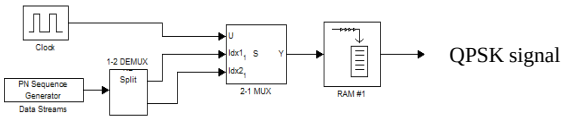


Figure 3: Block diagram for proposed QPSK modulator 2.

4 Simulation

Both proposed designs and conventional QPSK modulator modeled with Verilog HDL and simulated on Xilinx ISE platform. The crucial for simulate the conventional QPSK modulator is for comparison with the other two proposed architecture in term of power and area utilization in FPGA. Each of the design Verilog HDL code synthesizes and tested with a test bench code to simulate it functionality. The synthesizable code translated into RTL(Register Transfer level) schematic diagrams while the Xilinx ISim simulator used to run the test bench code to obtain timing diagram. The simulator also used to produce the decimal data which later used in Microsoft office Excel to plot the waveforms.

4.1 Conventional QPSK modulator

Figure 4 shows the RTL obtain from systhesize Verilog HDL code for conventional QSPK modulator while figure 5 shows a portion of timing diagram where the data (even and odd) change from 00 to 01. The QPSK wave is represented as sum (concatination of Cout and S) of I and Q phase in the timing diagram . Decimal data collected for QPSK wave, I and Q phases from the timing simulation used to plot the waveform .

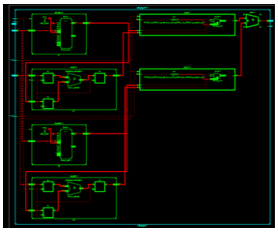
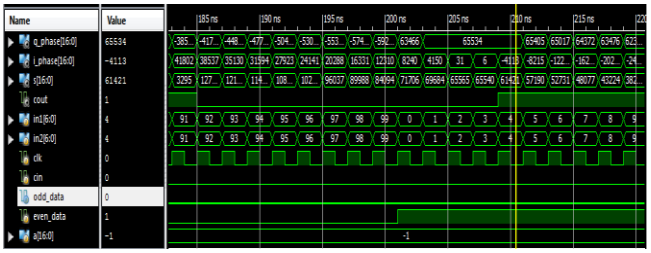


Figure 4: Top level RTL for conventional QPSK modulator



Figre 5: Timing analysis demonstrates a portion of even and odd data transition.

4.2 Proposed QPSK modulator 1

Figure 6 shows the RTL schematic diagram and figure 8 shows timing diagram for proposed QPSK modulator 1. Since the architecture no longer separate the baseband data into 2 single bit (even and odd baseband data), a group of 2 bits used to represent a phase in QPSK waveform. A total number of 100 data stored in each RAM to represent a phase in QPSK wave. Each data need 2 ns (one clock period) and for 100 data 200 ns needed to form a phase. Figure 7 shows the baseband data transition from 01 to 10.

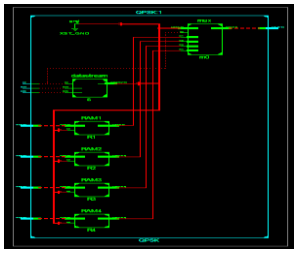


Figure 6: RTL diagram for proposed QPSK modulator 1

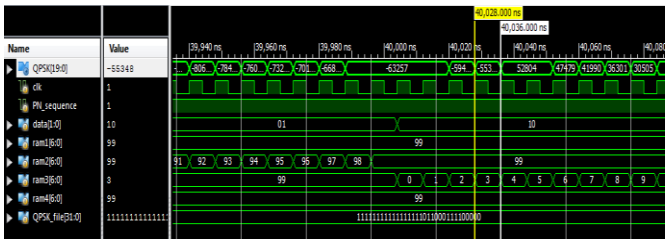


Figure 7: Timing diagram for proposed QPSK modulator 1

4.3 Proposed QPSK modulator 2

As for the proposed QPSK modulator 2, figure 8 shows the RTL schematic diagram and figure 9 shows the timing diagram. 100 data stored in one RAM to represent a complete cosine waveform and a counter assigned to access the data for a specific phase. The timing diagram shows the transition of baseband data from 11 to 00.

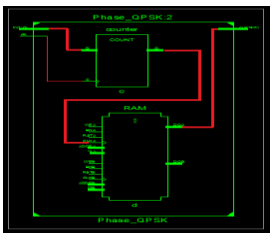


Figure 8: RTL schematic diagram for proposed QPSK modulator 2.

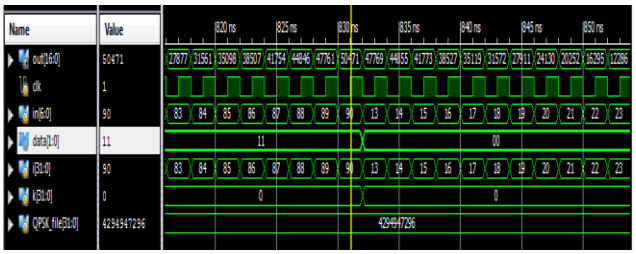


Figure 9: Timing diagram for proposed QPSK modulator 2.

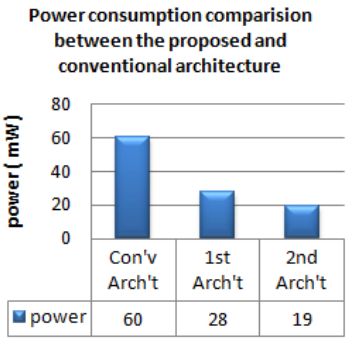


Chart 1: Power consumption for different architecture.

5 Discussion

Power consumption and area utilization is 2 main criteria studied in this project. Both criteria play major roles when the FPGA design translated into ASIC.

5.1 Power

Xilinx provide 3 types of power analysis tools where different tools perform power analysis at different stages. The Xpower estimator tool usually used in the predesign and preimplementation phases of a project [4]. While the XPower Analyzer (XPA) tool performs power estimation at post implementation stages [10]. It is the most accurate tool since it can read from the implemented design database the exact logic and routing resources used for a design. The last power analyzer tools provided by xilinx is PlanAhead RTL power estimator. This software provided an earlier stages of power and area utilaization of a desgin at RTL level. PlanAhead reads the HDL code from a design to estimate the resources needed, and reports the estimated power from a statistical analysis of the activity of each resource [5]. Since the PlanAhead provide an ealier power consumption and area utilaization analysis at same time,the software used instead of the other two power tools analyzer. Chart 1 shows the power comparison between the poposed and conventioanal QPSK modulator. Its cleary shows that the power consumption for both proposed QPSK modulator consume much more less power compare to the typical QPSK modulator architecture using the DDFS. The 1st architecture consume 32mW and the 2nd arcthitecure consume 41mW less power than the conventional architecture. The power analysis mainly carried out on device static power or leakage power where the transistor in FPGA use to hold the device configuration. However it is also known that the FPGA consumed more power compare to ASIC. The power consumption is high in FPGA due to it flexibity in configuration and rerouting . So when the proposed design implemented in ASIC where only dedicated number of transistor used for that design,more power consumption can be reduced.

5.2 Area

Area employment by number of gates in FPGA can directly influence the power consumption, reliability and the cost of a design. A simple design can cut or reduce the number of gates occupied in FPGA and at the same time increase the performance indirectly. The proposed 2nd architecture dramatically consumed more less gates compare to the conventional and the 1st proposed architecture. Even though the 1st architecture have almost double number of LUT from the conventional QPSK modulator, but the power consumption is still less when compared. This is mainly because the 1st architecture proposed does not employ DDFS, DSP48 and arithmetic logic blocks as in the conventional design. Chart 2 shows the number of LUT and I/O used for the proposed and conventional architecture.

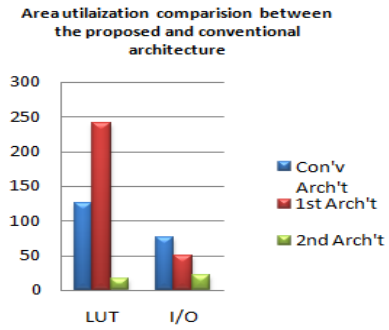


Chart 2: Area utilization in FPGA for different architecture.

6 Conclusion

The proposed QPSK modulators successfully simulated on Xilinx ISE 12.4 software platform and the results obtain shows that the output waveform is same as conventional QPSK modulator. The power analysis tools used on analyze the power consumption and area utilization on the proposed modulator also gives positive feedback. Both proposed architecture consume less power when compared with

conventional architecture. As for future plan, both designs will be implemented on virtex 6 FPGA board and RF front end module will be used to transmit the baseband data. A functional demodulator also will be constructed to retrieve the wirelessly transmitted baseband data.

7 References

[1] M.Kovac, J.kolouch "BPSK, QPSK Modulator Simulation Model" 2004.

[2] Teena Sakla, Divya Jain, Sandhya Gautam "Implementation of Digital QPSK Modulator by Using VHDL /MATLAB" International Journal of Engineering and Technology Vol. 2(9), 2010.

[3] Gihad Elamary,graeme Chester,Jeffrey Neasham " A Simple Digital VHDL QPSK Modulator Designed Using CPLD/FPGAs for Biomedical Devices Application" Proceeding of the World Congress on Engineering Vol 1,2009.

[4] User Guide for Xilinx XPower Estimator, www.xilinx.com/support/documentation/user_guides/ug440.pdf

[5] User Guide for Xilinx Plan Ahead, http://www.xilinx.com/support/documentation/sw_manuals/xilinx11/PlanAhead_UserGuide.pdf

[6] Jouko Vankka, "Direct Digital Synthesizers: Theory, Design and Application", 2000.

[7] Explaination about QPSK modulation, www.complextoreal.com

[8] Bit Error Rate over Signal to Noise Ratio, http://en.wikipedia.org/wiki/File:PSK_BER_curves.svg

[9] Václav VALENTA, "Error Performance of BPSK and QPSK in Satellite Communications".

[10] LogiCORE IP DDS Compiler v4.0 http://www.xilinx.com/support/documentation/ip_documentation/dds_ds558.pdf

[11] User Guide for Xilinx Xpower Analyzer, http://www.xilinx.com/support/documentation/sw_manuals/xilinx11/ug733.pdf

Improved Floating-Point Matrix Multiplier*

Wei-Ting Weng and Dyi-Rong Duh†
Department of Computer Science and Information Engineering
National Chi Nan University
Puli, Nantou Hsien 54561, Taiwan
{s98321508,drduh}@ncnu.edu.tw

***Abstract* – Floating-point matrix multiplier is widely used in scientific computations. A great deal of efforts has been made to achieve higher performance. The matrix multiplication consists of many multiplications and accumulations. Yang and Duh proposed a modular design of floating-point matrix multiplier which reserving intermediate result as two vectors. It brings shorter delay but more cost. This work modifies Yang and Duh's design with Booth encoding in multiplication to reduce the number of partial products. As the result, the improved floating-point matrix multiplier has better performance with shorter delay and much less hardware cost than Yang and Duh's design.**

Keywords - Matrix multiplication; floating-point multiplication; Booth encoding; partial products generator.

I. INTRODUCTION

Floating-point computation is widely used in managing scientific data. Recently, there has been a great deal of research on efficiently improving the performance of floating-point matrix multiplication, some of which were implemented in FPGA devices [1, 2]. Some previous researches performed floating-point matrix multiplication with modular processing elements and divided floating-point matrix multiplication into few sub-blocks, such as floating-point multiplication and floating-point addition. This way is also a common approach for improving the performance of floating-point matrix multiplication.

The multiplication can be partitioned into three sequential steps: generating partial products, summing up the partial products which were produced in former step until two vectors remain, and finally summing up the remaining two vectors by using a carry look-ahead adder (CLA).

Two methods are generally adopted to form the partial products in the first step. The first method uses 2-input AND gates to generate partial products, and the second method applies radix-4 Booth encoding, which has been used by Yeh and Jen [7]. The former is quiet simple and ordinary. The latter is more complex on implementation, but it is widely used to reduce the number of partial products.

The second step uses partial product reduction tree to sum up the partial products generated in the above step. Wallace tree [5] and Dadda tree [3] are well-developed algorithms for solving this problem. Both approaches utilize full adder and half adder as their basic elements to reduce the partial products until two vectors remain. Moreover, Dadda tree needs fewer hardware cost without having more delay as the trade-off. After reduction, a CLA is used to generate the product by summing up the remaining two vectors in final step.

Based on the architecture of floating-point matrix multiplier designed by Yang and Duh [6], this work modifies the original design including partial products generation and partial products reduction. Radix-4 Booth encoding is employed during generating partial products. The height of the partial product matrix can be reduced to nearly a half. In the mean time, the segments of sign-extension bits of all partial products are also merged into only one singular vector and a prominent bit.

In the above methods, the increment of delay in partial products generation is fewer than the decrement in partial products reduction. The hardware cost shows more obvious difference in similar way. As the result, both the delay and cost are reduced. Comparing with Yang and Duh's design, the proposed multiplier unit has 5.1% and 17.9% of improvement in delay and cost, respectively.

II. BACKGROUND

The double precision format of IEEE standard 754 [4] is composed of three parts: sign part, exponent part and mantissa part. The first part uses a bit to represent the sign of this number. One represents negative, otherwise positive. The exponent part is composed of 11 bits. There are 52 bits in the mantissa part. Since scientific representation is applied in this format, there always a leading one (hidden bit) precedes the binary point. That is only the fraction part of the binary floating-point number should be reserved. As the result, the inputs of floating-point multiplier in this work are both 53-bit wide.

In 2009, Yang and Duh proposed an efficient floating-point matrix multiplier [6]. The macro block diagram of Yang and Duh's design is shown in Fig. 1.

Specially, this modification reserves the intermediate result as two vectors instead of one as Bensalli's design [2]. With this scheme, the CLA is saved when generating product in floating-point multiplication, and vast delay is also eliminated simultaneously. As the trade-off, Yang and Duh's design sustains lots of cost.

* This work is partially supported by NSC 99-2221-E-260-010-.
† Correspondence to: D.-R. Duh; E-mail address: drduh@ncnu.edu.tw

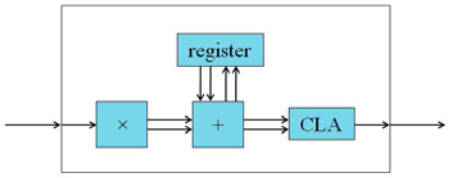


Fig. 1 The macro block diagram proposed by Yang and Duh.

Nevertheless, the floating-point multiplication and floating-point adder are modular. With duplicate multipliers and adders, floating-point matrix multiplier with better performance can be achieved.

III. Proposed Algorithm

The floating-point multiplier gets two n-bit wide inputs, multiplicand and multiplier, which are denoted by $A = (a_{n-1}a_{n-2}\ldots a_0)_2$ and $B = (b_{n-1}b_{n-2}\ldots b_0)_2$, respectively.

The weighted representation is employed such that B can be expressed as:

$$B = \sum_{j=0}^{n-1} b_j 2^j = -b_0 + \sum_{i=0}^{\lceil n/2\rceil-1} (b_{2i}+b_{2i+1}-b_{2i+2})2^{2i+1} \quad (1)$$

where $b_j=0$ for $j\geq n$, and $\lceil k \rceil$ denotes the smallest integer greater than or equal to k. Obviously, $b_{2i}+b_{2i+1}-b_{2i+2}$ corresponds to radix-4 Booth encoding. The value is bounded among 0, ±1 and ±2. For making the following equations much clearer, let $E_i = b_{2i}+b_{2i+1}-b_{2i+2}$ where $E_i \in \{0, \pm1, \pm2\}$, for $0\leq i\leq \lceil n/2\rceil-1$. Equation (1) can be formulated as:

$$B = -b_0 + \sum_{i=0}^{\lceil n/2\rceil-1} E_i 2^{2i+1} \quad (2)$$

The product $C = A\times B$ can thus be expressed as:

$$C = A\times B = -b_0A + A\times \sum_{i=0}^{\lceil n/2\rceil-1} E_i 2^{2i+1} = -b_0A + \sum_{i=0}^{\lceil n/2\rceil-1} E_i A 2^{2i+1} \quad (3)$$

The numbers to be accumulated in (3) can be represented by $\lceil n/2\rceil$ partial products, $PP_i = E_iA2^{2i+1} = (p_{i,2n-1}p_{i,2n-2}\ldots p_{i,0})_2$ for $0\leq i\leq \lceil n/2\rceil-1$, and a reparation vector, $-b_0A$, denoted by $R = (r_{n-1\ldots}r_0)_2$. Thus C can be restated as:

$$C = R + \sum_{i=0}^{\lceil n/2\rceil-1} PP_i \quad (4)$$

As shown in (4), the number of accumulations is $\lceil n/2\rceil+1$. Theses accumulations can be partitioned into two parts. First part is the reparation vector R. When $b_0=0$,

$$R = -b_0A = 0 = (00\ldots0)_2 \quad (5)$$

Otherwise,

$$R = -b_0A = -A = \overline{A} + 1 = (11\ldots1\overline{a_{n-1}a_{n-2}}\ldots\overline{a_1\,a_0})_2 + 1 \quad (6)$$

Notice that $\overline{K}$ represents 1's complement of K. The afterward adding one makes it become 2's complement of A.

The second part is partial products PP_i, for $0\leq i\leq \lceil n/2\rceil-1$. The weighted value here is left for clearly indicating the occupied spaces in the partial products. Those empty spaces will not occupy any hardware cost in partial products generator. According to the probabilities of E_i, PP_i is discussed in the following cases:

When $E_i = 0$,

$$PP_i = (00\ldots0)_2 \times 2^{2i+1} \quad (7)$$

When $E_i = 1$,

$$PP_i = (00\ldots0a_{n-1}a_{n-2}\ldots a_0)_2 \times 2^{2i+1} \quad (8)$$

When $E_i = 2$,

$$PP_i = (00\ldots0a_{n-1}a_{n-2}\ldots a_0)_2 \times 2^{2i+2} \quad (9)$$

When $E_i = -1$,

$$PP_i = (11\ldots1\overline{a_{n-1}a_{n-2}}\ldots\overline{a_1\,a_0})_2 \times 2^{2i+1} + 2^{2i+1} \quad (10)$$

When $E_i = -2$,

$$PP_i = (11\ldots1\overline{a_{n-1}a_{n-2}}\ldots\overline{a_1\,a_0})_2 \times 2^{2i+2} + 2^{2i+2} \quad (11)$$

The leading zeros and ones in (5), (6), (7), (8), (9), (10) and (11) indicate the sign-extended segments. This work reduces these segments into equivalent weighted bits, and it can be easily implemented by partial products generator. Since the leading zeros can be reduced to nothing, only the negative partial products with leading ones are discussed. The following equations present all the possible cases where a negative number occurs.

When $b_0 = 1$,

$$R = -A = \overline{A} + 1 = 2^{2n} - 2^n + (\overline{a_{n-1}a_{n-2}}\ldots\overline{a_1\,a_0})_2 + 1 \quad (12)$$

When $E_i = -1$,

$$PP_i = 2^{2n} - 2^{n+2i+1} + (\overline{a_{n-1}a_{n-2}}\ldots\overline{a_1\,a_0})_2 \times 2^{2i+1} + 2^{2i+1} \quad (13)$$

When $E_i = -2$,

$$PP_i = 2^{2n} - 2^{n+2i+2} + (\overline{a_{n-1}a_{n-2}}\ldots\overline{a_1\,a_0})_2 \times 2^{2i+2} + 2^{2i+2} \quad (14)$$

Furthermore, the product of $n\times n$ multiplication is bounded in a $2n$-bit wide number. Hence, the 2^{2n} ahead of equations (12), (13) and (14) can be automatically disregarded. The remaining sign-extended part can be easily represented by a row of sign-extended flag denoted by SE. Let SE_2^i and SE_1^i denote the second terms 2^{n+2i+2} and 2^{n+2i+1} in (13) and (14), respectively, for $0\leq i\leq \lceil n/2\rceil-1$. Also let SE_R represent the second term 2^n in (12). Thus, SE can be expressed with weighted representation as:

$$SE = (SE_2^{\lceil n/2\rceil-1}SE_1^{\lceil n/2\rceil-1}\ldots SE_2^0SE_1^0SE_R)_2 \quad (15)$$

In (15), $(SE_2^iSE_1^i)_2$ is equal to $(00)_2$, $(01)_2$ or $(10)_2$ when E_i is nonnegative, –1 or –2, respectively. Since $n = 53$ in this work, and $b_{54} = b_{53} = 0$, $E_{26} = b_{52}+b_{53}-2b_{54} = b_{52}$ is always a nonnegative number. Both SE_2^{26} and SE_1^{26} can be ignored automatically.

When SE ≠ 0, it denotes the number should be subtracted. To implement this operation, the partial products generation transforms SE into 2's complement as its negative form, which can be expressed as:

$$-SE = (\overline{SE_2^{\lceil n/2\rceil-1}SE_1^{\lceil n/2\rceil-1}}\ldots\overline{SE_2^0SE_1^0SE_R})_2 + 2^n \quad (16)$$

Symbol S is used to denote 2^n in (16) in the following sections.

IV. Proposed Architecture

This section shows the architecture of improved floating-point multiplier involving refined partial products generator and revised partial product reduction in the architecture designed by Yang and Duh [6]. This work utilizes a 53×53 multiplier to model the floating-point multiplication. The new contributed partial products generator composes of well-developed Booth encoder (BE) [7], revised Booth selector (BS) [7], reparation

vector unit, and reduced sign-extended unit.

Each BE block produces 3 bits: s^i, $2x^i$, and $1x^i$ which are used to represent the binary value of E_i. Fig. 2 and TABLE I present the BE block and its truth table, respectively. The generated three bits are used with the multiplicand A to form the partial product PP_i in the modified BS block.

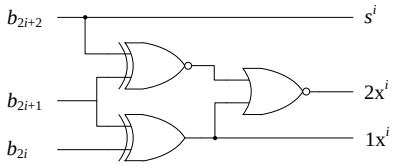


Fig. 2 The BE block.

TABLE I Truth Table of BE Block

b_{2i+2}	b_{2i+1}	b_{2i}	E_i	s^i	$2x^i$	$1x^i$
0	0	0	0	0	0	0
0	0	1	1	0	0	1
0	1	0	1	0	0	1
0	1	1	2	0	1	0
1	0	0	–2	1	1	0
1	0	1	–1	1	0	1
1	1	0	–1	1	0	1
1	1	1	–0	1	0	0

The BS is modified for generating not only the partial products PP_i, but also the sign-extended flags, SE_1^i and SE_2^i. Furthermore, $p_{i+1,2i+1}$ ($p_{i+1,2i+2}$) represent the adding one (two) in 2's complement of PP_i if $E_i = -1$ ($E_i = -2$). Fig. 3 and TABLE II show the new Booth selector block and its truth table for PP_i, where $0 \le i \le 25$.

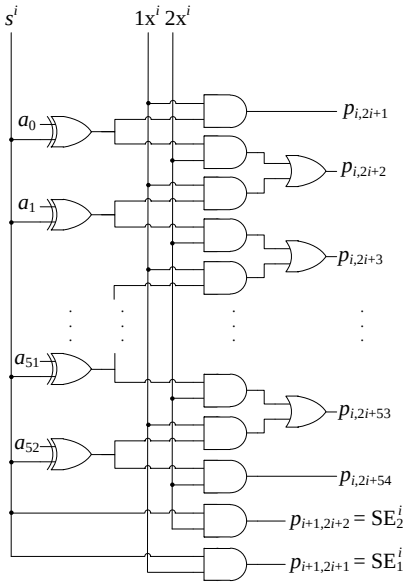


Fig. 3 The new BS block.

TABLE II Truth Table of BS Block.

E_i	s^i	$2x^i$	$1x^i$	$p_{i,2i+1}$	$p_{i,2i+2} \sim p_{i,2i+53}$	$p_{i,2i+54}$	SE_2^i	SE_1^i	$p_{i+1,2i+2}$	$p_{i+1,2i+1}$
0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	a_0	$a_1 \sim a_{52}$	0	0	0	0	0
2	0	1	0	0	$a_0 \sim a_{51}$	a_{52}	0	0	0	0
–0	1	0	0	0	0	0	0	0	0	0
–1	1	0	1	$\overline{a_0}$	$\overline{a_1} \sim \overline{a_{52}}$	0	0	1	0	1
–2	1	1	0	0	$\overline{a_0} \sim \overline{a_{51}}$	$\overline{a_{52}}$	1	0	1	0

Since this floating-point multiplier is used to manage two 53-bit numbers in this study, $b_{53} = b_{54} = 0$. As the result, $E_{26} = b_{52} + b_{53} - 2b_{54} = b_{52}$ is represented by one bit. Fig. 4 and TABLE III show the logic circuit for PP_{26} and its truth table, respectively.

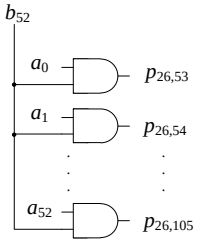


Fig. 4 The logic circuit for partial product PP_{26}.

TABLE III Truth Table for Partial Product PP_{26}

b_{52}	E_{26}	$p_{26,53} \sim p_{26,105}$
0	0	0
1	1	$a_0 \sim a_{52}$

Fig. 5 and TABLE IV demonstrate the design of reparation vector unit. Since forming reparation vector R only depends on b_0 and multiplicand A. The s^R is the sign bit of R, and $p_{0,0}$ represents the adding one for forming 2's complement of A when $s^R = 1$.

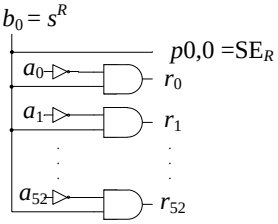


Fig. 5 The logic circuit of reparation vector unit.

TABLE IV Truth Table of Reparation Vector Unit.

b_0	s^R	$p_{0,0}$	SE_R	$r_0 \sim r_{52}$
0	0	0	0	0
1	0	1	1	$\overline{a_0} \sim \overline{a_{52}}$

Fig. 6 illustrates the sign-extended unit. Significantly, when $n = 53$, the 54^{th} column consists of $p_{i,53}$, $0 \le i \le 26$, S and SE_R ($\overline{SE_R}$) if $S = 0$ ($S = 1$). It makes the height of the partial product matrix become 29, which needs one more reduction stage than the matrix with height 28 does in partial products reduction. To avoid this situation, this work calculates S, SE_R ($\overline{SE_R}$) and SE_1^0 ($\overline{SE_1^0}$) if $S = 0$ ($S = 1$) previously by a particular logic circuit, and the height of partial product matrix is then reduced to 28.

By implementing the above blocks, the height of the partial products matrix is 28. This work adopts Dadda's algorithm [3] to form partial products reduction tree. It needs 7 full adder stages to reduce the 28 rows of the partial products matrix into 2 vectors remain. Then these two remaining vectors are processed by other blocks in the original multiplication designed by Yang and Duh [6] to generate the intermediate result.

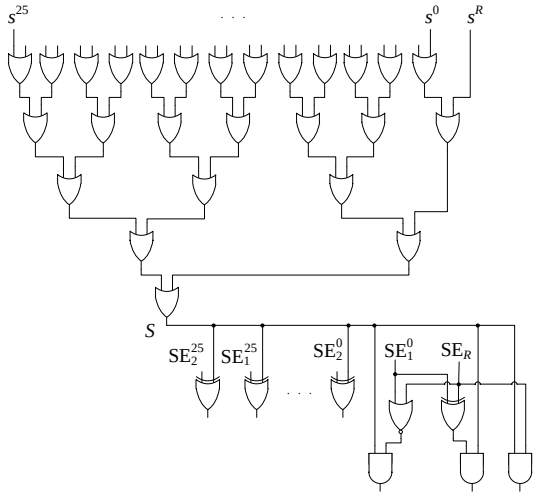


Fig. 6 The logic circuit design of reduced sign-extended unit.

V. COMPARISON

In this section, a common approach is used to estimate the improved floating-point matrix multiplier. This method takes each 2-input monotonic gate, such as AND, NOR, etc., has one gate delay and cost, and it makes each 2-input XOR or XNOR has two gate delays and costs. TABLE V shows the delay and cost of each component unit for our, Yang and Duh's and Benssali's works.

TABLE V Delay and Cost for Each Component Unit

	Our Work		Yang and Duh's		Benssali et al.'s	
	Delay	Cost	Delay	Cost	Delay	Cost
Multiplication Unit	37	17650	39	21489	119	13208
Addition Unit	33	14406	33	14406	126	6370
Register	4	355	4	655	4	320
CLA (Final Addition)	96	4829	96	4829	×	×

The comparison as shown in TABLE VI is under the three different designs for resolving one element of an 8×8 floating-point matrix multiplication. Figs. 7, 8 and 9 present the first, second and third design, respectively. Each of them has different cost and delay. The first one is the simplest design of the above. The second one is the fastest but with the largest cost. The last one is the balance design between cost and delay. TABLE VII is the improvement of our work comparing with Yang and Duh's and Benssali's.

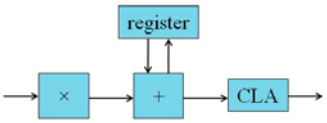


Fig. 7 The first design of comparison.

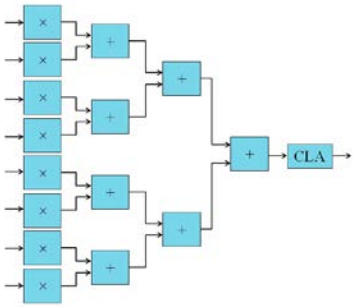


Fig. 8 The second design of comparison.

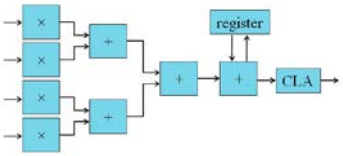


Fig. 9 The third design of comparison.

TABLE VII Improvement Compared with Other Works

	Improvement (%)					
	Compare with Yang and Duh's			Compare with Benssali et al.'s		
	Delay	Cost	Delay×Cost	Delay	Cost	Delay×Cost
First	3.6	9.3	12.6	63.2	–95.5	28.0
Second	0.7	11.1	11.7	42.5	–64.3	5.5
Third	1.2	10.3	11.4	48.5	–70.0	12.4

VI. CONCLUSION

The cost and delay of multiplication unit in our work is 37 and 17650, respectively. By contrast, those in Yang and Duh's work are 39 and 21489. Thus the delay and cost in multiplication unit improve 5.1% and 17.9%, respectively. In the simplest design of an 8×8 floating-point matrix multiplication, the improvements compared with Yang and Duh's design are 3.6%, 9.3% and 12.6% in delay, cost and delay×cost, respectively.

REFERENCES

[1] F. Bensaali, A. Amira, and A. Bouridane, "Accelerating matrix product on reconfigurable hardware for image processing applications," *IEE Proceedings of Circuits, Devices and Systems*, vol. 152, no. 3, pp. 236–246, 2005.

[2] F. Bensaali, A. Amira, and R. Sotudeh, "Floating-point matrix product on FPGA," in: *Proc. ACS/IEEE International Conference on Computer Systems and Applications*, pp. 466–473, May. 2007.

[3] L. Dadda, "Some schemes for parallel multipliers," *Alta Frequenza*, vol. 34, pp. 349–356, 1965.

[4] *IEEE Standard for Binary Floating-Point Arithmetic*, ANSI/IEEE Standard 754-1985.

[5] C. S. Wallace, "A suggestion for a fast multiplier," *IEEE Trans. Comput.*, vol. 13, no. 2, pp. 14–17, Feb. 1964.

[6] L. C. Yang and D. R. Duh, "Optimized design of a floating-point matrix multiplier," in: *Proc. National Computer Symp.*, pp. 300–308, Nov. 2009.

[7] W. C. Yeh and C. W. Jen, "High-speed Booth encoded parallel multiplier design," *IEEE Trans. Comput.*, vol. 49, no.7, pp. 692–701, Jul. 2000.

TABLE VI Comparison with Other Works Under Three Structures

	Our Work			Yang and Duh's			Benssali et al.'s		
	Delay	Cost	Delay×Cost	Delay	Cost	Delay×Cost	Delay	Cost	Delay×Cost
First	425	37540	15954500	441	41379	18248139	1155	19198	22173690
Second	286	246871	70605106	288	277583	79943904	497	150254	74676238
Third	323	133708	43187684	327	149064	48743928	627	78632	49302264

High Speed Performance and Yield Optimization Technique for Analog Integrated Circuits

Sawal Ali[1], Md. Shabiul Islam[1], Siti A. Ahmad[2]

[1]Department of Electrical Electronic & System Engineering,
University Kebangsaan Malaysia, 43600 UKM, Bangi, Selangor.

[2]Department of Electrical and Electronic Engineering,
University Putra Malaysia, 43400 UPM, Serdang Selangorl.

Abstract— *This paper presents a novel predictive modeling technique for yield and performance of analog integrated circuits. Trade-offs between performance functions can be explored through the use of a multi-objective evolutionary algorithm and Monte Carlo simulations. When compared to conventional simulation based approaches, the results show a significant improvement in overall simulation time and efficiency without a corresponding drop in accuracy. The behavioral model has been developed in Verilog-A and tested extensively with practical designs using the Spectre simulator. Two OTA topologies are used to demonstrate the proposed algorithm and their behavior has been verified through transistor level simulations. The examples have demonstrated that accurate performance and yield prediction can be achieved using the proposed method in a fraction of the time taken by conventional simulation based methods.*

Keywords—Analog circuit design, Behavioral modeling, circuit synthesis, design automation, multi-objective optimization, yield optimization.

1 Introduction

IN the last few decades, developments in technology have revolutionized the consumer electronics market. Advances in device technology have led to a significant reduction in transistor size, allowing millions to be integrated onto a single chip. This trend has seen a movement from several functional blocks occupying one or more boards to the integration of these blocks onto a single device. Although the majority of functional blocks in an integrated system are digital, analogue circuits are still required to interface to the real world and this drives the requirement to integrate both analogue and digital circuits onto the same chip.

In such mixed signal environments, the area occupied by analogue circuits is often small compared to the area of their digital counterparts. However, the design of the analogue sections can be complex and can cause a bottleneck in the overall design flow [1]. The demand for a reduction in design time has led to the development of analog automation tools. The development of such tools has resulted in a transition from manual design approaches to simulation based methodologies utilizing numerous optimization methods. As a result of the increase in device model complexity and the difficulty in evaluating large numbers of specifications, circuit simulators have become an essential part of the analogue design flow [2].

It is common in engineering problems to optimize more than one objective function, a process known as multi-objective optimization (MMO). In multi-objective optimization, it is often impossible to find a single solution that satisfies all objectives. Instead, a tradeoff usually exists between the competing objectives and this result in a set of optimal solutions called the Pareto-front [3].

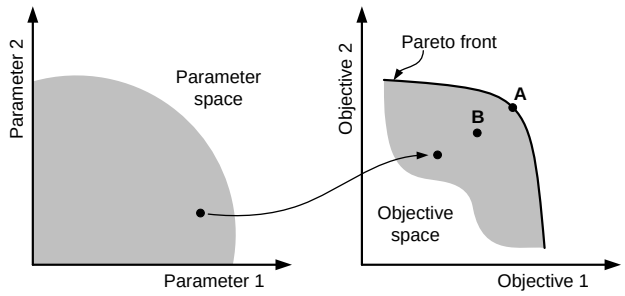


Fig. 1. Design space and objective space

A multi objective optimization problem has a number of objective functions which are to be minimized or maximized. Usually, the optimization has a number of constraints which the solution must satisfy. The multi objective optimization formulation can be generally stated as follows:

$$\begin{aligned} &\textit{Minimise / Maximise } f_m(x), m = 1,2...M \\ &\textit{Subject to } g_j(x) \geq 0,\ j = 1,2...J; \\ &h_k(x) = 0, k = 1,2,...K; \end{aligned} \quad (1)$$

$$x_i^{(L)} \le x_i \le x_i^{(U)}, i = 1,2,...n$$

Where $f_m(x)$ is the set of M performance functions or objective functions that constitute a multi-dimensional space called the objective space. In equation (1), there are several constraints associated with the problem formulation. J and K are inequality and equality constraints define by the $g_j(x)$ and $h_k(x)$ functions whereas x_i is a set of boundary constraints that restrict each decision variables to take a value between a lower and upper bound. These boundary constraints represent the decision space of the design problem.

A solution that satisfies none of the constraints is called an infeasible solution as opposed to a feasible solution that satisfies all constraints. For each point in the decision space, there exists a point in the objective space. Figure 1 shows these two spaces and a mapping between them.

In the objective space, the entire feasible region can be divided into two sets of solutions: non-dominated solutions and dominated solutions. None of the non-dominated solutions can be said to be better than the other with respect to the objective functions, and these are referred to as the Pareto-optimal solutions. If any solution in the objective space is dominated by at least one solution from the Pareto-optimal set, then it is referred to as a non-Pareto-optimal solution. In figure 1, solution B is dominated by solution A. Solution A is said to be non-dominated (Pareto-optimal solution). The thick black line in this figure that represents all the Pareto-optimal solutions is called the Pareto-front. There are several evolutionary based algorithms that have been developed for multi objective optimization in order to generate the optimal Pareto-front. One of the algorithms often used for this purpose is the Non-dominated Sorting Genetic Algorithm - II (NSGA-II) [4].

One of the challenges in downscaling transistor size is to design and verify integrated circuits for a high yield. A major challenge associated with DSM technologies is the increase in process variations which influences the quality and yield of designed and manufactured circuits [5]. With a high correlation between circuit yield and profit, yield maximization has become a major issue in deep sub-micron design. Yield must therefore be considered alongside conventional performance parameters as early as possible in the design process. This consideration is often referred to as Design for Yield or Design for Manufacturability (DFY/DFM) [6].

2 Pareto Based Optimization

The method proposed in this paper is based on Pareto optimization. The Pareto-front resulting from Multi-Objective Optimization is used for the performance and variation model that is developed. This idea is based on the method proposed in [7]. In the proposed Pareto-based yield optimization method, Monte Carlo simulation is used to estimate the yield of the design. In order to reduce the number of Monte Carlo runs, the simulation is only applied to a small feasible region defined by the performance specification boundaries. Because there are only a small number of solutions in the feasible region, far fewer Monte Carlo analyses are required, mitigating the computational overhead. Figure 2 illustrated the Pareto-based yield optimization methodology.

It can be seen from this figure that there are 10 solution points on the Pareto-front that are within the specification boundaries. Therefore, the Monte Carlo simulation is only applied to these solutions and the solution with the highest yield result is used in the design.

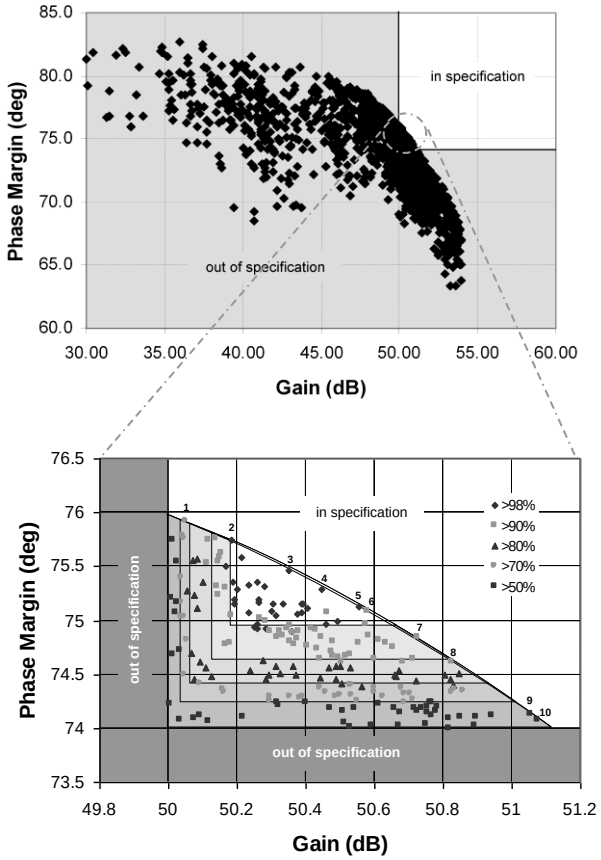


Fig. 2. Pareto-based yield optimization

To demonstrate the advantage of Pareto based optimization over conventional simulation-based approaches, a comparison has been made with NeoCircuit, a commercial optimization tool that optimizes circuit performance and yield. The tool is based on a global optimization approach that combines evolutionary and simulated annealing algorithms. The optimization is performed at the transistor level design. The approach starts with performance optimization to meet a given specification and is followed by yield maximization to push the design far from the specification boundaries. Since there is

no Pareto type exploration in the algorithm, a penalty scheme is used to reduce instances of excessive performance that may occur during yield maximization in order to maximize overall yield. This involves several iterations during the yield maximization. For example, during the first iteration, a performance f1 might be overdesigned and cause the optimization on performance f2 to be limited. In order to increase the yield of f2, the performance of f1 must be reduced. Several stages of iteration are required in order to maximize the overall design yield.

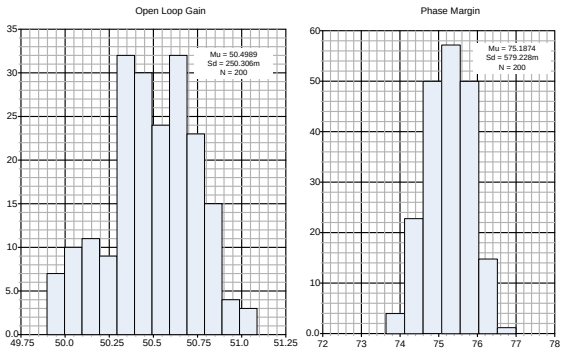
Pareto-based optimization uses a different approach where all the design performances are represented as a trade-off to make it easier to select a more balanced solution and maximize the yield.

To demonstrate the advantage of Pareto-based optimization, a comparison has been made using a symmetrical OTA design. Table I summarizes the comparison with the Monte Carlo histogram shown in Figure 3. It can be clearly seen that the Pareto-front yield optimization method performs significantly faster and produces better results than the NeoCircuit optimization. In this comparison, the Pareto-front technique completed the optimization in 48 minutes and produced a 98% overall yield whilst NeoCircuit took 1hr 29 minutes and produced a 96.5% overall yield. Even though the advantage shown in term of CPU time is not significant, but the main contribution of Pareto-based optimization is the ability to be used at system level design through hierarchical-based optimization. With such divide and conquer approach, huge saving can be capitalized through the proposed technique.

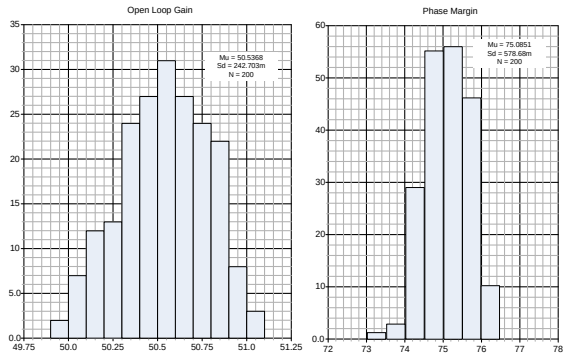
TABLE I
YIELD OPTIMIZATION COMPARISON

Parameters:	Pareto-based optimization:	NeoCircuit
Gain	50.58 dB	50.54 dB
Gain Yield	99%	96.5%
PM	75.14 deg	75.24 deg
PM Yield	98%	99%
Overall Yield	98%	96.5%
CPU Time	48 minutes	1hr 29 minutes

Pareto-front optimization is the basis of the work proposed in this paper and is used together with Monte Carlo analysis to create the behavioral performance and variation model for the design.



a)



b)

Fig. 3. Monte Carlo histogram for gain and phase margin. a) NeoCircuit. b) Pareto-based optimization

3 Experiment Examples

This section presents a complete design example using two different topologies for an operational transconductance amplifier (OTA) circuit. The OTA is a fundamental building block, often employed in analogue circuit applications such as filters. In this section, the performance and variation model for two OTA topologies are created: a symmetrical OTA, depicted in figure 4, and a Miller-compensated OTA, depicted in figure 5. All the simulations were performed using the industry standard Cadence Spectre simulator with foundry level BSim3v3 transistor models from a standard CMOS process technology.

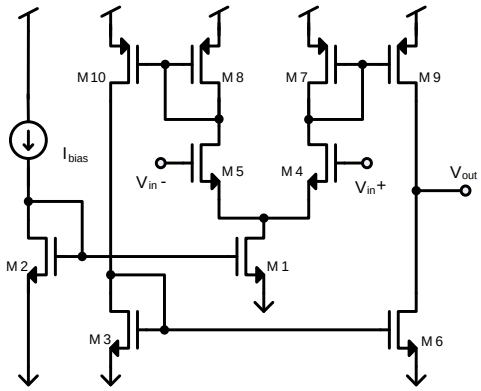


Fig. 4. Symmetrical OTA topology

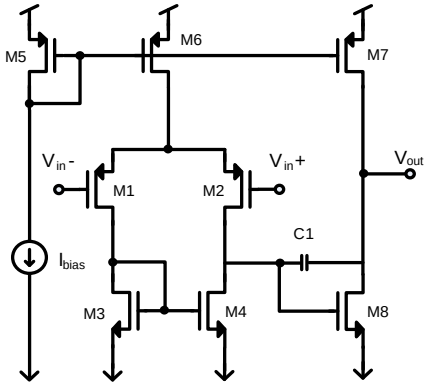


Fig. 5. Miller-OTA topology

3.1 Design Setup

The first step in the algorithm is to create a circuit netlist for the two topologies. The next step is to determine the designable parameters for the circuit topology and specify their objective functions. All transistor lengths and widths for the circuits are the designable parameters and two objective functions were chosen for this example: open loop gain and phase margin. For the purpose of performance evaluations, a test-bench netlist must be created for each of the objective functions.

3.2 Multi-Objective Optimization

The designable parameters are constrained within a reasonable range. All transistor lengths were specified to be between 0.12um and 4um and transistor widths were specified to be between 10um and 40um. These ranges were chosen so that the design area will not exceed the targeted transistor active area of 200um. Once the parameters have been determined, a GA string can be constructed. The string is shown in figure 6. Each individual generated by the evolutionary algorithm will consist of a set of designable parameters as defined by the string and these will be used to replace the parameters in the circuit netlist for the circuit simulation. A total of 100 generations each with a population size of 100 were used for the optimization giving a total number of samples of 10,000. The process of parameter generation by evolutionary algorithm, updating spice parameters and circuit simulation is continued until the total number of generations is reached. The end result of the optimization is a Pareto-front consisting of all the optimal solutions for the circuit.

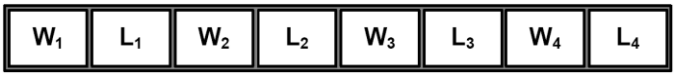


Fig. 6. Evolutionary algorithm string

3.3 Performance and Variation Modeling

To illustrate the results of the optimization, Figure 7 and 8 show a plot of the objective space for both of the circuit topologies. The Pareto front for symmetrical OTA and Miller-OTA can be clearly seen and contains 1022 and 987 optimum solutions respectively. All the solution points on these Pareto-fronts and their design parameters are stored in data files which define the performance model for each topology.

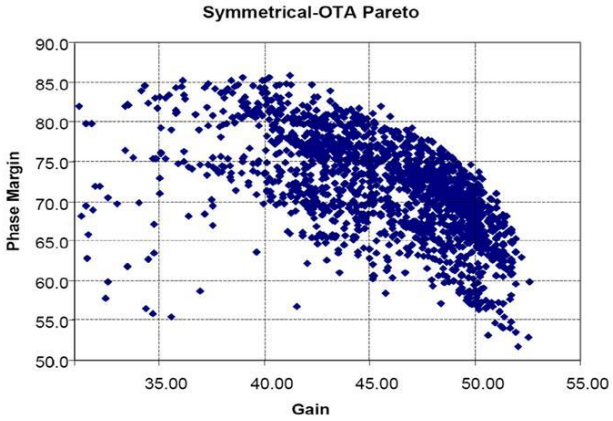


Fig. 7. Symmetrical OTA pareto plot

The next step is to create the variation model for the Pareto-points. Every optimal solution on the Pareto-front undergoes a Monte Carlo simulation using process variation and mismatch models. 200 samples were chosen for the MC simulation and from these the variation for each performance is calculated. This information is stored in a data file and represents the variation model for the circuit.

At this point, a combined performance and variation model for the OTA is developed. Table II shows some selection samples from the performance and variation table for the symmetrical OTA topology. This table is defined as a look-up table for a table model function with the resulting Verilog-A [8] model given in listing 1.

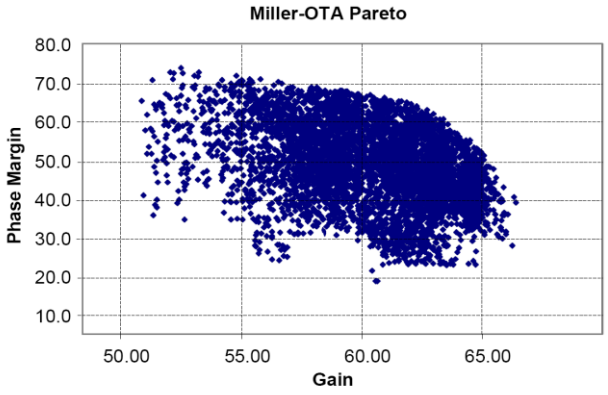


Fig. 8. Miller-OTA pareto plot

TABLE II
PERFORMANCE AND VARIATION VALUES

Design:	Gain (dB):	ΔGain (%):	PM (deg):	ΔPM (%):
21	49.78	0.52	76.3	1.50
22	49.90	0.52	76.1	1.51
24	49.98	0.51	76.0	1.51
25	50.17	0.51	75.8	1.52
26	50.35	0.50	75.5	1.56
27	50.45	0.49	75.3	1.57
34	51.06	0.44	74.1	1.69
35	51.14	0.51	74.0	1.71
37	51.24	0.42	73.8	1.69
38	51.62	0.42	73.2	1.68

3.4 Interpolation Example and verification

The performance and variation model can be used to find a circuit solution for a given performance specification. This avoids the need to re-run the simulation-based optimization and will significantly reduce the design cycle time.

Listing 1. Performance and Variation Model

```
analogue begin
 gm_delta = $table_model (gain,"gm_delta.tbl",”3E");
 ro_delta = $table_model (ro, "pm_delta.tbl", "3E");
 pm_delta = $table_model (pm, "pm_delta.tbl", "3E");
 gm_prop = ((gm_delta/100)*gm)+gm;
 ro_prop = ((ro_delta/100)*ro)+ro;
 pm_prop = ((pm_delta/100)*pm)+pm;
 p1 = $table_model (gm_prop,ro_prop,pm_prop
 "p1_data.tbl","3E,3E,3E");
 p2 = $table_model (gm_prop,ro_prop,pm_prop
 "p2_data.tbl","3E,3E,3E");
 p3 = $table_model (gm_prop,ro_prop,pm_prop
 "p3_data.tbl","3E,3E,3E");
 p4 = $table_model (gm_prop,ro_prop,pm_prop
 "p4_data.tbl","3E,3E,3E");
 fptr=$fopen("params.dat");
 $fwrite(fptr, "\n Generated Design Parameters\n ");
 $fwrite(fptr, "%e %e %e %e", p1,p2,p3,p4);
 $fclose(fptr);
 $display ("params: = %e %e %e %e", p1, p2, p3, p4);
End
```

The model will interpolate a new performance value from a given specification that can produce the highest yield based on the variation model. From this new performance value, a new set of designable parameters is then interpolated based on the performance model. Table III shows an example where the required performance is a gain greater than 50dB and a phase margin of greater than 74 degrees.

TABLE III
INTERPOLATION EXAMPLE

Performance:	Required Performance:	Variation:	New Performance:
Gain	> 50dB	0.51%	50.26dB
Phase Margin	> 74 deg	1.71%	75.27 deg

The variation for gain and phase margin performance is obtained by interpolation from the table model function. In this case, the relevant look-up table points are those shown in Table II where it can be seen that the gain of 50dB is between design point 24 and 25. The variation interpolation given between this point is 0.51. Using this variation value, it can be said that the actual gain may vary from 49.75dB to 50.26dB and therefore, in order to achieve maximum yield, the specified gain of the design must be at least 50.26dB. If we choose a design point with a 50.26 dB gain value, and 0.51 variation, the gain will vary between 50.01dB to 50.51dB. This will ensure that the required 50dB gain will be achieved within the process extremes. The value of 50.26dB therefore becomes the new targeted performance value and using this new value, the design parameters are interpolated from the performance table. The same strategy is applied for phase margin. Both of the new performance values for gain and phase margin will produce a yield of 100%.

To verify the performance and yield interpolated by the behavioral model, a comparison has been made with transistor level simulation using the design parameters obtained from the table model function. This comparison is shown in table IV. The percentage error in passband gain and phase margin was calculated between the OTA transistor simulation and interpolated values. Figure 9 shows the open loop gain for the Verilog-A model and transistor model. It can be seen from these comparisons that the Verilog-A function matches closely with the transistor level simulation.

Figure 9 shows a divergence in the comparison above 40MHz which is attributed to parasitic poles in the transistor circuit. Although these higher order effects are not modeled in this example, they could be incorporated if required. A Monte Carlo simulation using 500 samples was carried out and verified a yield of 100%.

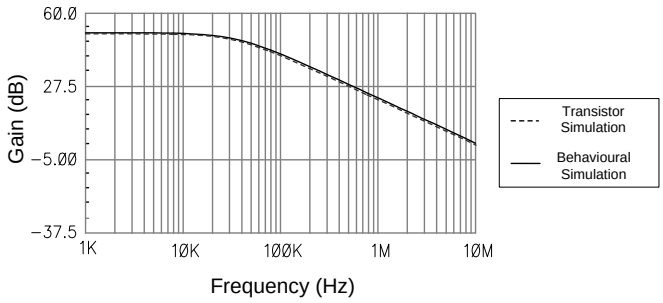


Fig. 9. Open loop gain comparison

TABLE IV
PERFORMANCE COMPARISON

Performance Functions	Transistor Model	Verilog-A Model	% error
Gain	50.73	50.26	0.93%
Phase Margin	76.06	75.27	1.03%

3.5 Topology comparison

The interpolation example shown previously demonstrates how the table model function can be used to search a design solution for a particular circuit topology. However, the model will not find a solution if the new targeted performance is not feasible with the chosen topology. In this case, a search across a different topology could yield the solution. As an example, assume the gain specification is >51dB and Phase Margin is >74 degrees. Looking at table II, the variation for the gain and phase margin is 0.44% and 1.71% respectively. This leads to the new targeted performances of gain greater than 51.22dB and phase margin greater than 75.27 deg. Table II shows that these new targeted performances are not feasible for the symmetrical topology since design points 37 and 27 meet one specification but not the other. This problem can be solved by searching the solution in other topology. As can be seen in figure 10, a Miller-OTA topology satisfies the requirements as shown by the shaded area.

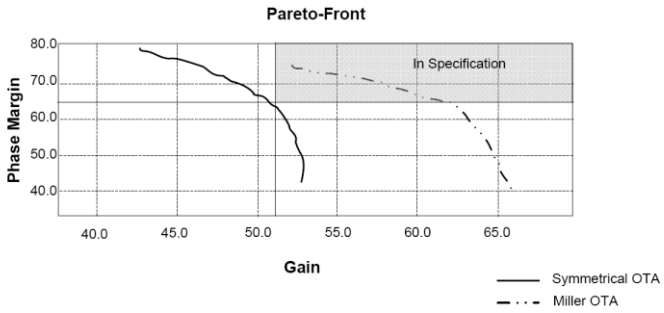


Fig. 10. Pareto comparison between topology

3.6 Summary of Examples

Table V summarizes the parameters associated with the model development. A total of 10,000 simulations were run in the initial MOO step for the performance model for both of the OTA topologies and Monte Carlo analysis was performed on 1022 Pareto Optimal points of symmetrical OTA and 987 points of Miller-OTA for the variation model. The whole model development stage took 4 hours to complete for the symmetrical OTA and 3 hours 40 minutes for the Miller-OTA on a 1.2GHz Ultra Sparc 3 computer system.

The effort involved in developing the performance and variation model can be compared with the transistor level optimization strategy such as that used in NeoCircuit or Pareto-front optimization. Compared to NeoCircuit optimization, which requires 1hr 29 minutes to optimize the OTA, the cost involved for the model development (in terms of CPU time) will be paid off after 3 repeated use.

TABLE V
SUMMARY OF EXAMPLES

Parameters:	Symmet-OTA:	Miller-OTA:
No. Generations	100	100
Evaluation Samples	10,000	10,000
Pareto Points	1022	987
CPU Time (1.2GHz Sparc 3)	4 hours	3h 40m

4 conclusion

This paper has presented a new approach that combines performance and variation objectives in a behavioral model for analogue circuits. Multi-objective optimization based on an evolutionary algorithm is used to explore tradeoffs between performance and yield, leading to a set of Pareto optimal solutions for the design. Monte Carlo variation analysis is performed on all the Pareto optimal solutions, and a table is constructed for both the performance and variation analysis. A behavioral model developed in Verilog-A is used together with this table to determine the parameters required to achieve the highest yield within a given specification. After the initial time investment to create the model, there are significant improvements in overall simulation time and efficiency compared to conventional simulation based approaches. These benefits are enjoyed without a corresponding drop in accuracy. Two benchmark OTA topologies have been presented to demonstrate the proposed algorithm and the behavior has been verified through transistor level simulations.

5 References

[1] D. L. Wim Kruiskamp, "Darwin: Cmos opamp synthesis by means of a genetic algorithm," *in Design Automation, 1995. DAC '95. 32nd Conference on*, pp. 433-438, 1995.

[2] B.De Smedt and G.Gielen, "Watson: design space boundary exploration and model generation for analogue and rfic design," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transaction on*, vol.22, no.2, pp. 213-224, Feb. 2003.

[3] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*, John Wiley & Sons Ltd, 2001.

[4] G. Gielen, G. Gielen, T. McConaghy, and T. Eeckelaert, "Performance space modeling for hierarchical synthesis of analog integrated circuits," in *Proc. 42nd Design Automation Conference*, T. McConaghy, Ed., 2005,pp. 881–886.

[5] A. Ripp, M. Buhler, J. Koehl, J. Bickford, J. Hibbeler, U. Schlichtmann, R. Sommer, and M. Pronath, "Date 2006 special session: Dfm/dfy design for manufacturability and yield – influence of process variations in digital, analog and mixed-signal circuit design," *in Design, Automation and Test in Europe, 2006. DATE '06. Proceedings*, vol. 1, 6-10 March 2006, pp. 1-6.

[6] B. D. Smedt, G. Gielen, "HOLMES: Capturing the yield-optimized design space boundaries of analogue and RF Integrated Circuits." In *Proc. Of the Design,Automation and Test* in Europe Conference and Exhibition, 2003.

[7] Ali, S., Wilcock, R., Wilson, P. and Brown, A, "Yield model characterization for analog intergrated circuit using a Pareto-optimal surface," *IEEE International Conference on Electronics, Circuits, and Systems*, August 2008, Malta.

[8] Verilog-AMS Language Reference Manual: Analog & Mixed-Signal Extensions to Verilog-HDL, 2nd ed., www.verilog-ams.com.

Leakage - Delay Tradeoff in Wide-Bit Nanoscale CMOS Adders

Claudia Romo, Savithra Eratne, Eugene John, and Byeong Kil Lee
Department of Electrical and Computer Engineering
University of Texas at San Antonio
One UTSA Circle, San Antonio, Texas, 78249, U. S. A.

Abstract*—The scaling of nanometer technology has had a major impact on the power dissipation of CMOS circuits. As transistor size decreases it has become apparent that leakage power is becoming a dominant fighting force against future technology. In this paper the importance of static power consumption on the design of new and advanced CMOS technology is explored with the investigation of leakage power reduction techniques and their implementation on embedded CMOS adder circuits. Four different adder topologies of bit sizes 16, 32 and 64, were implemented using technology nodes 22nm, 32nm, and 45nm. To reduce the leakage power dissipation of these adders three different types of leakage reduction techniques were implemented and simulated using Hspice to determine the leakage and delay. The results were analyzed and the optimal leakage reduction technique(s) for each nanometer adder design was determined.*

Keywords: leakage reduction, CMOS adders, static power

1. Introduction

As CMOS technology advances and is scaled down in size, more transistors are integrated into smaller chip sizes. Technology scaling aims at improving IC performance by reducing power consumption, decreasing gate delay, and increasing transistor density. Unfortunately, as a direct result of the scaling of transistors many design challenges have developed, including the increase of leakage current.

Although scaling can result in higher performance levels for circuits, when the voltage threshold is scaled leakage power increases, up to 5 times each generation [1]. Leakage power consumption has increases from 18% at 130 nm to 54% at 65 nm of the total power consumption [2]. The prediction is that leakage power will eventually account for 50% or more of power dissipation in ICs. Also, it is projected that by the year 2020, leakage power is expected to increase 32 times per device [3]. With these factors involved the performance of a nanoscaled technology having a significantly large percentage of leakage power dissipation will result in a lost in performance. Designing low power VLSI circuits with leakage power in mind has become increasing important in the progression of IC design.

In order to investigate the effectiveness of leakage reduction in various adder topologies that utilize the smallest technology nodes available, for this research, leakage reduction techniques have been applied to four different adders of varying bit sizes, for technology nodes 22nm, 32nm, and 45nm. In Section II of this paper the four adder topologies that were used are described and their equivalent schematics are illustrated. In Section III the three leakage reduction techniques applied to the adder circuits are briefly described. Section IV reviews previous work related to this research and examples of their conclusions are given. Section V describes the approach taken to design and simulate all circuits described in this paper. Section VI gives an evaluation of the results and Section VII presents our conclusions.

2. CMOS Adders

Four common CMOS adders were designed for the simulations in this research, the Ripple Carry Adder (RCA), the Carry Select Adder (CSA), the Carry Bypass Adder (CBA), and the Carry Lookahead Adder (CLA).

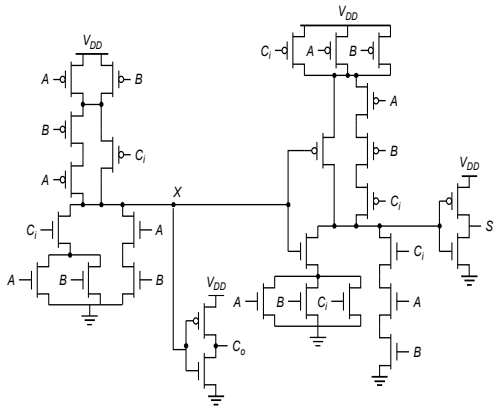


Figure 1. 28T Full Adder

The full adder is the basis for several multi-bit adders. For our research the basic 28 transistor (28T) full adder was used for all the full adder blocks. The schematic of the 28T full adder [5] is given in Fig. 1.

The first implementation of the multi-bit adder is in the ripple carry adder (RCA). An RCA is an adder consisting of cascading or 'rippling' of full adders in which the output carry bit of one full adder becomes the input carry bit for the next full adder [5][6]. The schematic of the 4-bit RCA utilizing the full adder cell is given in Fig. 2.

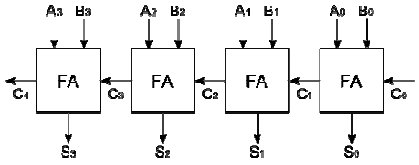


Figure 2. 4-bit Ripple Carry Adder

The CSA is an adder designed to improve the delay of the RCA. It uses two sets of full adder chains and multiplexers to determine the sum and to decrease delay of the circuit. The two full adder chains compute the addition of the inputs simultaneously, one for a carry input of 'one' and the other for a carry input of 'zero' [6]. Once the carry in is inputted it becomes the select signal and the sums and carry out are chosen from different multiplexers. An illustration of a typical 4-bit CSA setup is shown in Fig. 3.

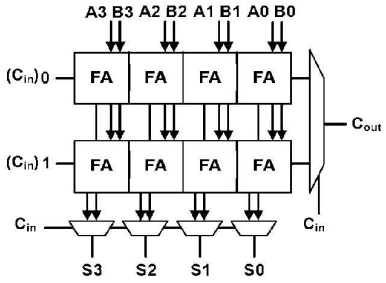


Figure3. 4-bit Carry Select Adder

The third adder design utilizing the full adder block is the CBA and also aims at improving the delay of the RCA. It contains added circuitry, which includes set-up circuitry to calculate the *propagate* and *generate* of each input bit and a 2:1 multiplexer to determine the final carry-out [5]. The 2:1 MUX has two inputs, the original carry-in that was bypassed, and the carry-out from the final full adder chain. The select is dependent on the *propagates* of each input bit. The speed is gained when the carry-out is generated and cascaded to the next CBA chain. The schematic for a 4-bit CBA is shown in Fig. 4.

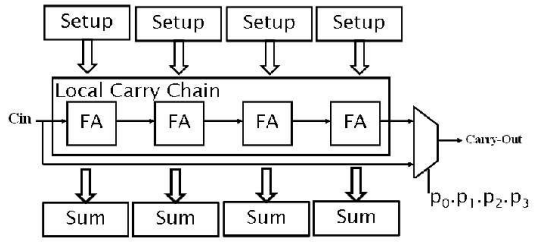


Figure 4. 4-Bit Carry Bypass Adder

One of the most common and faster types of multi-bit adders is the carry-lookahead adder, through the use of Boolean functions called *generate* and *propagate*, a block of circuitry consisting of AND and XOR gates is designed to calculate the final carry out bit. The carry *propagate* determines if the carry input bit can propagate to the next stage and the carry *generate* determines if the carry output bit is generated in the current stage. Propagates and generates are then used to calculate all the carry outputs simultaneously by using the carry out functions specified in [5]. The sums for this adder circuit are calculated by XOR of the *propagates* and the corresponding carry outs.

3. Leakage Reduction Techniques

The increasing effect of leakage power on the advancement of technology has resulted in an extensive effort to discover methods to reduce or minimize leakage in CMOS circuits. In this research, we investigate the impact of three widely known leakage reduction techniques, dual voltage threshold, transistor stacking, and power gating, on CMOS wide-bit adders.

3.1 Dual Voltage Threshold

Dual-V_{th} technology is applied by dividing the data paths of a digital circuit into critical and non-critical paths and assigning low and high V_{th} values, respectively, to the corresponding transistors. By decreasing the V_{th} of the transistors along the critical path, speed is improved and by increasing in the V_{th} of the non-critical path transistors, leakage power decreases in the standby and active mode [7].

3.2 Transistor Stacking

Transistor stacking method exploits the stacking effect that results from two or more turned off transistors in series to reduce leakage [8]. In the transistor stacking method, also known as 'forced stacking' a transistor is added in series to transistors along the path with the largest current leakage [9].

3.3 Power Gating

Power gating is a method in which blocks of circuitry are 'cutoff' from the power supplies while not in use, to improve overall power consumption. Power gating can have PMOS and/or NMOS switching transistors to cut off the supply voltages [10].

4. Related Studies

Many leakage reduction techniques have been previously applied to CMOS adder circuits, resulting in reduced power consumption. In [11], an enhanced MTCMOS scheme, a type of power gating technique, was applied to an 8-bit Brent-Kung adder. The results were an improvement on leakage that was dependent on the device widths being optimized. In the analysis conducted in [3], a 28 transistor 1-bit full adder was the basis for the testing of a power gating, input vector control, forced stacking, and sleepy stack method of reducing leakage. In all cases, leakage was improved, with power gating of both source terminals having the most improvement and sleepy stack having the least improvement. In the experimental research of the sleepy stack method in [8], a 4-bit adder is constructed from cascading 28-transitor full adders. The sleepy stack achieves 2490x leakage reduction over the base case, and achieves 190x leakage power reduction over the forced stack, while increasing delay by 6% and increasing area by 113%. In [12], the dual voltage threshold method was applied to the 28 transistor full adder and simulated

against various threshold values; concluding that a combination of low and high voltage values resulted in the least amount of power consumption.

5. Tools and Methodology

The four adder designs, in 16-bit, 32-bit, and 64-bit sizes were implemented using HSPICE. Each of the HSPICE decks were modified to simulate with the PTM BSIM4 model cards for bulk CMOS sizes: 22nm, 32nm, and 45nm [13]. For all transistors the width to length ratio was 2:1. In all cases, leakage current was measured for ten input combinations and the delay of each circuit was measured from the input bit to the most significant output bit. Input combinations varied from all low inputs to all high inputs, to combinations which required the carries to propagate thru to the carry-out. For all sums and carry-outs an inverter buffer was placed after the output.

For the leakage reduction technique of stacking transistors, transistors along the non-critical path were stacked each having a width to length ratio of 1:1. For the Dual-V_{th} method, transistors along the critical path maintained the same low voltage threshold as given in the model card and the voltage threshold of all other transistors were increased by 10% of the original values. For the Power gating method, a sleep transistor was placed at the supply voltage only.

6. Results and Analysis

Leakage power of the four adder designs, in 16-bit, 32-bit, and 64-bit sizes at the 32nm feature size is shown in Fig. 5, 6, and 7, respectively. 'Standard' was the label given to the adder design without any applied leakage control methods. The RCA maintained the least amount of leakage in all instances, across all bit sizes, technology nodes, and leakage techniques, largely due to the minimum amount of circuitry required for that design. All leakage reduction techniques tested were effective in reducing the leakage.

In the case of the RCA the most effective method of reducing leakage was transistor stacking. This resulted in 36%, 42%, and 41% decrease in leakage for all bit sizes in 22nm, 32nm, and 45nm technology, respectively. Transistor stacking was also most effective for the CSA in the technology sizes 32nm and 45nm, resulting in a decrease in leakage power of 48% to 57% for each bit size. The 22nm CSA was most improved with the power gating method which resulted in a leakage power decrease of 67% for all bit sizes. For the CBA all three methods were similarly effective resulting in leakage reductions of 22% to 30%. But in the 22nm and 32nm technologies power gating was the most effective resulting in a decrease of 64% and 33% respectively for all bit sizes. In the case of the CLA the dual-V_{th} and the stacked transistor method reduced leakage by 45% to 54% for all cases. The power gating method was once again the most effective for the technology size 22nm resulting in leakage reductions of up to 67%.

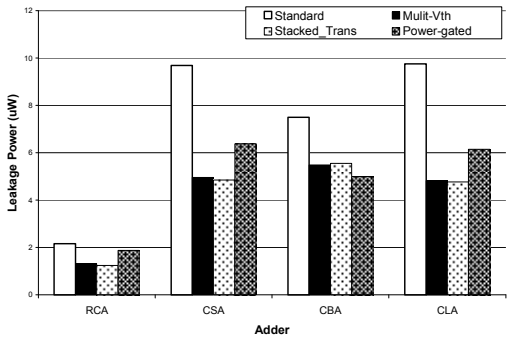


Figure 5. Leakage Power for 16 bit Adders at 32nm

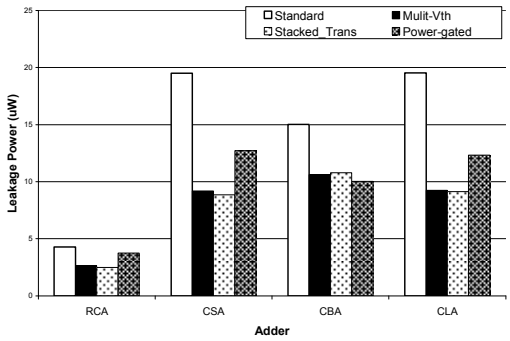


Figure 6. Leakage Power for 32 bit Adders at 32nm

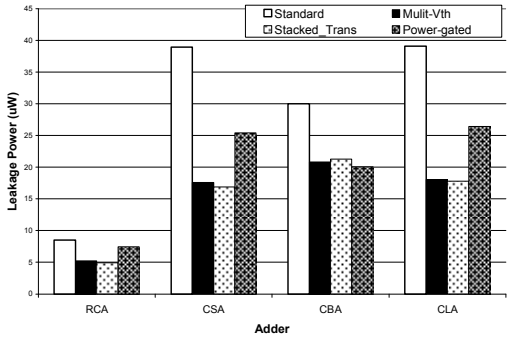


Figure 7. Leakage Power for 64 bit Adders at 32nm

Fig. 8, 9, and 10, show the delay impact on the 16-bit, 32-bit, and 64-bit adders, respectively, for the 32nm feature size. In all cases the power-gated method resulted in the most delay penalty, followed by the stacked transistors and then the dual-V_{th}. For the RCA, the most effective leakage reduction technique, transistor stacking, resulted in a delay penalty of up to 12%. For the CSA, transistor stacking and power-gating, resulted in a delay increase of up to 140% and 250% for the 16-bit, 79% and 184% for the 32-bit, and 41% and 138% for the 64-bit. The leakage reduction method of power-gating for the CBA resulted in delay increases of 90% to 226%, for each case. The CLA with dual-V_{th} had a delay increase of 20% to 59%, for all feature sizes and bit sizes. The stacking method resulted in delay increase of 97% to 173%, for all features sizes and bit sizes. Power gating for the CLA for the 22nm technology size resulted in the most increase delay of all simulations, nearly 4x the original delay for bit sizes 16-bit 32-bit and 64-bit.

The resulting leakage power and delay for the 22nm and 45nm technology nodes of the 32-bit adders that are not shown in the graphs are given in Table 1 and 2, respectively. Due to length limitation only the data of the 32-bit adders are shown.

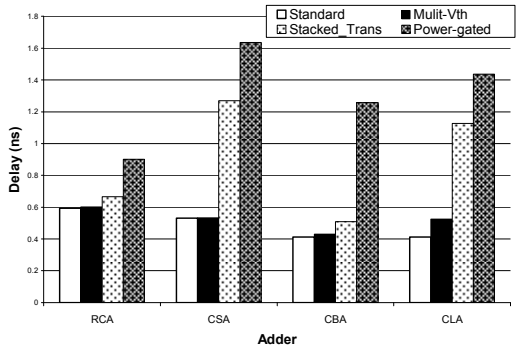


Figure 8. Delay for 16 bit Adders at 32nm

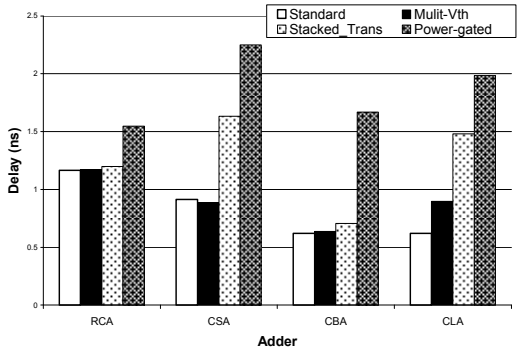


Figure 9. Delay for 32 bit Adders at 32nm

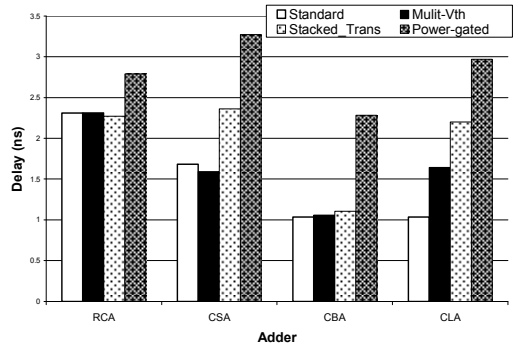


Figure 10. Delay for 64 bit Adders at 32nm

7. Conclusion

Leakage reduction techniques applied to the adders examined in this research resulted in reduced leakage in all cases. The reduction ranged from 8% for the 64-bit 45nm RCA to 68% for the 4-bit 22nm CLA. The most effective method varied based on the adder, the technology size, and the bit size. With the tradeoff of decreased leakage power consumption and increased delay penalty, it is expected that the most effective method results in the most penalty delay. This was the case for many of the simulations results. Tradeoffs could be optimized such as the 64-bit CSA in which the leakage was reduced with the dual-V_{th} by up to 54% with a delay penalty of only 5%. In all cases the power-gated method resulted in the most delay penalty and the dual-V_{th} resulted in the least delay penalty. The delay penalty also varied greatly from no delay to an increase of nearly 4x the original delay.

Table 1. Leakage Power (uW) of 22nm and 45nm 32-Bit Adders

	RCA		CSA		CBA		CLA	
Technique	22nm	45nm	22nm	45nm	22nm	45nm	22nm	45nm
Standard	10.29	3.95	49.22	17.64	37.57	13.38	49.33	17.26
Dual-Vth	6.71	2.45	26.19	8.70	27.92	9.57	26.19	8.53
Stacked	6.56	2.32	25.60	8.45	28.44	9.74	26.33	8.54
Power-gated	6.54	3.65	16.30	13.72	13.63	10.48	15.86	13.11

Table 2. Delay (ns) of 22nm and 45nm 32-Bit Adders

	RCA		CSA		CBA		CLA	
Technique	22nm	45nm	22nm	45nm	22nm	45nm	22nm	45nm
Standard	1.31	1.28	0.99	1.00	0.70	0.67	0.70	0.67
Dual-Vth	1.32	1.29	0.97	0.96	0.74	0.69	1.20	0.92
Stacked	1.29	1.33	1.60	1.70	0.76	0.75	1.70	1.49
Power-gated	1.86	1.59	2.82	2.20	1.94	1.59	3.32	1.83

REFERENCES

[1] S. Borkar, "Design Challenges of Technology Scaling," Micro, IEEE Volume 19, Issue 4, Jul-Aug 1999 Page(s):23 – 29.

[2] P. Gupta, A.B. Kahng, P. Sharma, and D. Sylvester, "Gate-length biasing for runtime-leakage control," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Volume 25, Issue 8, Aug. 2006 Page(s):1475 – 1485.

[3] B. S. Deepaksubramanyan, and A. Nu˜nez, "Analysis of Subthreshold Leakage Reduction in CMOS Digital Circuits," 50th Midwest Symposium on Circuits and Systems, 2007. 5-8, pp1400 – 1404, Aug. 2007.

[4] J.M. Rabaey, A. Chandrakasan, and B. Nikolic, "Digital Integrated Circuits, A Design Perspective," Pearson Education, 2003.

[5] M.M. Vai, "VLSI Design," CRC Press LLC, 2001.

[6] J.T. Kao and A.P. Chandrakasan, "Dual-Threshold Voltage Techniques for Low-Power Digital Circuits," IEEE Journal of Solid-State Circuits, Volume 35, Issue 7, July 2000, p1009 – 1018.

[7] J.C. Park, and V.J. Mooney III, "Sleepy Stack Leakage Reduction," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Volume 14, Issue 11, Nov. 2006 p1250 – 1263.

[8] S. Mukhopadhyay, C. Neau, R.T. Cakici, A. Agarwal, C.H. Kim, and K. Roy, "Gate Leakage Reduction for Scaled Devices Using Transistor Stacking," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Volume 11, Issue 4, Aug. 2003 Page(s):716 – 730.

[9] Y. Tsai, D. Duarte, N. Vijaykrishnan, and M.J. Irwin, "Impact of process scaling on the efficacy of leakage reduction schemes," International Conference on Integrated Circuit Design and Technology, 2004. Page(s):3 – 11.

[10] R.M. Rao, J.L. Burns, and R.B. Brown, "Analysis and Optimization of Enhanced MTCMOS Scheme," 17th International Conference on VLSI Design, 2004. Proceedings. p234 – 239.

[11] A. Ghosh, and D. Ghosh, "Optimization of Static Power, Leakage Power and Delay of Full Adder Circuit Using Dual Threshold MOSFET Based Design and T-Spice Simulation, " International Conference on Advances in Recent Technologies in Communication and Computing, 2009, p 903-905.

[12] Predictive Technology Model (PTM) website, available at:- http://www.eas.asu.edu/~ptm