A Teaching Methodology with Frame Buffer and Monitor Design in the Study of Hardware Track in Computer Science

Hassan Farhat
Computer Science
University of Nebraska at Omaha

Abstract- *While the study of computer design is covered extensively in the hardware track of computer science, minimal actual design of the remaining units required in image display is covered. Designs of simple CPUs with needed memory can be accomplished towards the end of a first course in digital design. Alternatively it can be covered during the early part of a second course on digital and computer architecture. Due to time constraints, the study of video controllers and frame buffers needed in the interface to a monitor is normally covered in advanced classes in computer engineering and is often skipped in computer science. Instead, computer simulation of designs or use of seven-segment-displays is often used to verify computer design correctness.*

In this paper we provide pedagogy for including frame buffer design topics in conjunction with CPU design. The proposed work integrated with previous work on computer design provides a practical sequence of topics to cover from design of a CPU to monitor display. To meet the time constraints imposed by the number of classes in hardware track, unlike discussion of design found using hardware descriptions languages, we provide a methodology of design at the schematic (gate) level. The proposed designs form a complete educational system of a computer with a frame buffer that can be covered in a two 3-credit hour course sequence.

The contribution of the paper is educational in nature. The paper provides a pedagogy of teaching a somewhat advanced design of topics during a first and a second course in hardware in computer science. We provide methodology of teaching designs of a simple buffer in a first course in the hardware track. These topics can then be used in conjunction with CPU designs in a second course on computer design.

(KEYORDS: Digital Design, Instruction Set, Video Controller, Frame Buffer, Computer Organization)

1 Introduction

The study of digital circuits from simple gates to central processing units and to complete computer design is found in computer science as well as computer and electrical engineering. To reduce the complexity of design and analysis, the digital design aspects are studied at several levels. It is covered at the lowest level, the switch and chip design level [1]. It is covered as well at the next higher level, the digital logic level [2] to [7]. At this level, gates are used as building blocks that are realized as several switches from the chip or switch level. Gates are then combined with registers and Arithmetic Logic Units (ALUs) and studied at the computer organization level [8, 9]. The computer architecture [10] to [13] level combines units from the organization level into processors, multi-level memories, and input/output interface units.

The study of hardware track in standard computer science curricula may include the design of a microprocessor. While this design can be accomplished towards the end of a first course or early part of a second course in digital and computer design, the designs of frame buffers and video controllers are normally skipped. The design of such units is normally covered in advanced undergraduate courses in computer and electrical engineering.

In earlier work we incorporated two design packages, Altera [14] and Multisim [15]. For both packages we looked at the design of a simple computer to include under curricula constraints. In [16] we showed the design with monitor interface using units provided by [18]. With the popularity of programmable logic devices and packages such as Multisim and Altera, we propose a teaching methodology that includes designs of sample frame buffers as part of the curricula. While these topics were outside the scope of teaching in a two 3-credit hour courses in digital design, we feel they can be included as part of a first course on digital design. To accomplish this, we show the design of both can be accomplished at the gate schematic level with no need for the overhead required in learning hardware description languages[1]. The frame buffer design is normally presented as a memory unit. Based on the desired interface with the simple computer design, we show the design can be built from components covered in a digital design course with two small ROMs (Read Only Memories). As a result, the design of the frame buffer and the video controller can be accomplished toward the end of a first course on digital

[1] A proposed video controller design at the gate level is found in [16].

design. Figure 1 shows the proposed contribution of this paper.

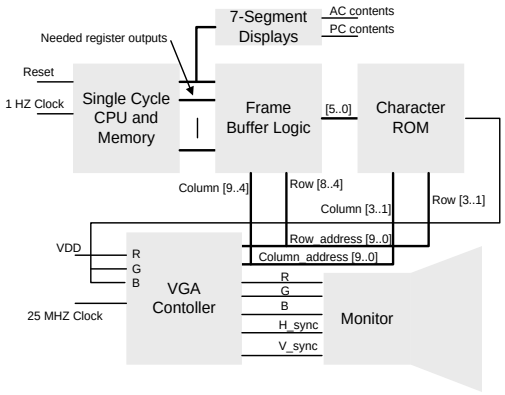


Figure 1: A schematic of the Complete Computer Design with Program Entered using Memory Initialization Files in Altera

The contribution is in proposing schematic realization of the frame buffer logic at the gate schematic level and providing a list of needed topics. Unlike previous work, we realize the design using simple schematic logic captures without the need to learn hardware description languages.

The paper is organized as follows. In section 2 we review the instruction set of the single-cycle computer. In section 3 we look at functionality of the frame buffer. Section 4 includes proposed designs and the experimental results. The conclusions are given in section 5.

2. Review of the Instruction Set of the Single-Cycle Computer

Designs of computers can vary depending on the size and type of the instruction set chosen. Popular texts include the design of a simple version of a MIPS Reduced Instruction Set Computer [8]. [13] is a classical text that discussed designs of accumulator based computers. We work with a similar simple design of an accumulator based computer. We also choose a single cycle design with a single addressing mode (direct addressing). Finally we choose a small set of instructions that conforms to instruction set completeness. A 4-bit opcode field is chosen for possible expansion. A small memory is chosen, 256 words. The rationale is to complete design process without having to keep track of all the needed details when compared to larger instruction sets. The complete instruction set of the computer, [16], is given in Table 1.

In [16] we gave simple examples of expanding the functionality of the computer by writing macros for additional logic and arithmetic operations. The instructions given above form a small subset from a popular earlier AC-based instruction set architecture (PDP 8).

Table 1: Simple AC based instruction set of computer design

Instruction	Meaning	Opcode
LDA XX	AC ← M[XX]	0XX
STA XX	M[XX] ← AC	1XX
ADD XX	AC ← AC+M[XX]	2XX
NAND XX	AC ← NOT (AC AND M[XX])	3XX
BUN XX	PC ← XX	4XX
SKZ XX	If AC = 0 then PC ← PC + 1	5XX
INC	AC ←AC + 1	6XX

Since the design is a single-cycle design, we need to complete instruction execution during the same cycle. Since the instruction needs to be fetched from memory, the instruction fetch and instruction execution must be completed in the same cycle. We accomplish this by having two memories, the instruction memory and the data memory. The presence of the data memory eliminates the need for a memory buffer register. A schematic of the simple design is shown in Figure 2, [16].

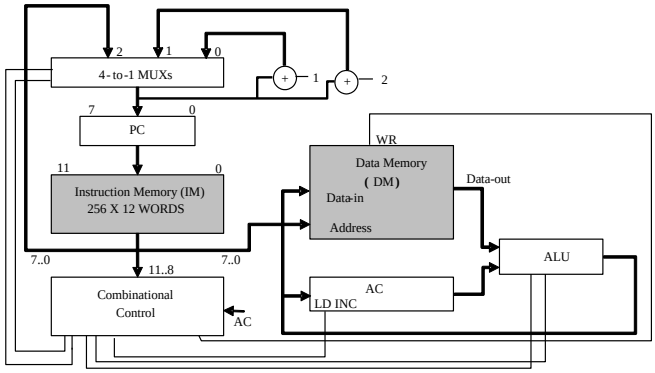


Figure 2: Schematic of the single-cycle computer

Table 2 shows how the control unit of the simple computer can be designed. Since the computer is a single cycle and since the instruction set is a small instruction set, the control unit can be realized as a small read only memory. The outputs of the ROM are the different control signals sent to the different units of the computer.

In the table, the PC_{MUX} is used to adjust the program counter (PC) value in case of branch instructions. From the instruction set, the branch unconditional (BUN X) and the increment and skip if zero (ISZ) instructions are the only two instructions that affect the value of the program counter. The multiplexer input to the program counter chooses from three options, the default increment PC and the previous two options.

3 The Frame Buffer and Needed Topics

We explore the design of the frame buffer interface as a

Table 2: Contents of ROM for simple computer design

	Control Address								Control Word Contents				
Instr.	AC_{11}	AC_{10}	.....	AC_0	IM_{11}	IM_{10}	IM_9	IM_8	PC_{MUX}	ALU	AC_L	AC_{INC}	WR
LDA X	x	x	.....	x	0	0	0	0	00	00	1	0	0
STA X	x	x	.....	x	0	0	0	1	00	01	0	0	1
ADD X	x	x	.....	x	0	0	1	0	00	10	1	0	0
NAND X	x	x	.....	x	0	0	1	1	00	11	1	0	0
BUN X	x	x	.....	x	0	1	0	0	10	xx	0	0	0
SKZ	0	0	.....	0	0	1	0	1	01	xx	0	0	0
SKZ	1	x	.....	x	0	1	0	1	00	xx	0	0	0
SKZ	x	1	.....	x	0	1	0	1	00	xx	0	0	0
.....	.....	.....	1	.....	0	1	0	1	00	xx	0	0	0
SKZ	x	x	.....	1	0	1	0	1	00	xx	0	0	0
INC	x	x	.....	x	0	1	1	0	00	xx	0	1	0
Not used	x	x	.....	x	x	1	1	1	xx	xx	x	x	x

small circuit realized at the schematic level from units covered in a digital design course. The intent is to: a) introduce the students to the concepts of frame buffers early in the hardware curricula, and b) explore the concepts of significant saving in hardware based on the application.

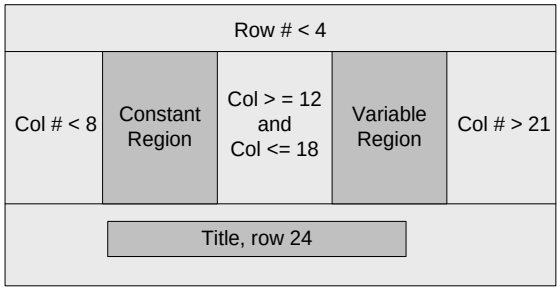


Figure 3: VGA display divided into four regions, background region colored as

The VGA monitor for our application is used to display the current instruction and the computer register contents. We refer to Figure 3 for illustration. To design the frame buffer we divide the monitor area into four regions: 1) the background region, 2) the constant region, 3) the title region, and 4) the variable display region. The background region display the background color. The title region is used to display the title of the design. The constant region displays 4 character words representing registers, arithmetic logic unit, the current instruction and data memory labels. The variable region displays the contents of the corresponding registers, the ALU, and instruction and data memory. The variable values are the only part of the display that changes.

In a traditional design, the frame buffer is composed of a memory unit with contents that mirror the contents displayed on the screen. Here, however, based on the application, we choose a combination of a small memory and standard logic gates to realize the frame buffer design. The frame buffer produces the needed pixel values for a 640×480 standard VGA monitor. The character sizes displayed are 16 by 16 pixels. As a result the display is composed of 30 rows (480/16); each row is 40 characters. By incorporating a small character ROM using concepts of lookup-tables from early designs, one can reduce the memory requirements.

We reduce the memory requirements further by: a) using two lookup tables, and b) determining the needed logic gates design for region determination. A partial schematic of the frame buffer design is shown in Figure 4. The design is done in the Altera MAX+PLUS II package.

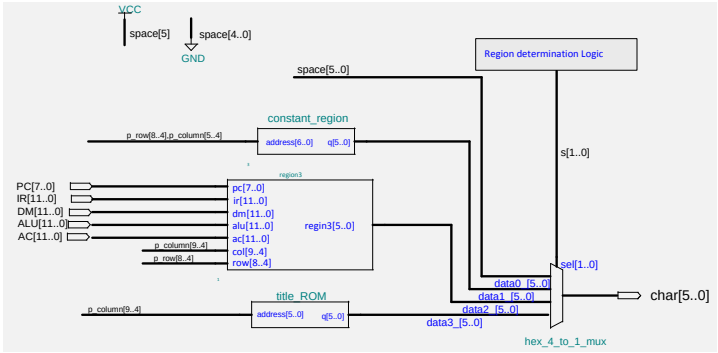


Figure 4: Partial Schematic of the Frame Buffer Design

As can be seen the traditional memory realization of a frame buffer is replaced with gate schematic and two small ROMs (constant region includes a ROM as well). Only region 3 receives inputs from the simple computer design. The hex_4_to_1_mux selects which of 4 data busses to route to the output. The select lines values are computed by the region determination logic block diagram. The logic determination block uses a set of comparators with column and row values as inputs obtained from a standard VGA controller. The title ROM and the constant region ROMs are small memory units used to display the proper characters within each of the regions. The space[5..0] data input displays the space character (ASCII 40_8). The memory initialization files for the constant and title regions are shown in Figure 5 (a) and 5 (b) respectively.

```
Depth = 128;
Width = 6;
Address_radix = bin;
Data_radix = oct;
% Constant Region %
Content
  Begin
[0000000..1111111] : 40; % space    %
0010100  : 20 03 40 40;  %  PC      %
0011100  : 11 22 40 40;  %  IR      %
0100100  : 04 01 24 15;  %  DATM    %
0101100  : 01 14 25 40;  %  ALU     %
0110100  : 01 03 40 40;  %  AC      %
0111100  : 55 55 55 55;  %  ---     %
1000100  : 11 16 23 24;  %  INST    %
End
```

(a)

```
Depth = 60;
Width = 6;
Address_radix = bin;
Data_radix = oct;
% Title ROM %
Content
  Begin
[000000..111011] : 40; % space   %
000100  : 23 11 15 20; %  SIMP   %
001000  : 14 05 40 03; %  LE C   %
001100  : 17 15 20 25; %  OMPU   %
010000  : 24 05 22 40; %  TER    %
010100  : 04 05 23 11; %  DESI   %
011000  : 07 16 40 40; %  GN     %
End
```

(b)

Figure 5: Memory Initialization Files using Altera MAX+PLUS

We illustrate the design of region 3 and show the design can be accomplished from standard topics covered in digital design. Two sets of multiplexers are used; one set is used to determine the proper row of the display and another set is used to display the proper column. Figure 6 show a partial schematic of the column multiplexers. As can be seen, the bulk of the realization is done using multiplexers.

at its inputs. If the region determination logic selects the variable region this code is forwarded to the character ROM unit.

Due to space limitation, we skip further details of the design. Instead, we look at a sample case in the design where the AC contents are to be displayed. By referring to Figure 6, we note that the s[1..0] select lines are such that 00 selects the leftmost column and 11 selects the rightmost column. Hence the AC contents are placed on the multiplexer data inputs from most significant (multiplexer data 0 = ac[11..8]) to least significant (multiplexer data 3 = ac[3..0]). As can be seen from the figure, actual inputs have o[1..0] = 11 appended from the left. This is used so as to display the variable data properly [18, p. 150]. In [18] the character ROM table is such that the hexadecimal character codes 0 through F have the octal code 60 through 77. As a result, to display the proper hexadecimal value of a register we append 11 to the left part of the code. For example, an AC[7..4] = 1010 is displayed as hexadecimal A by appending 11 to the left part of the binary value 111010 is octal 72 (character A).

4 Experimental Results

The computer design and monitor interface was completed in MAX+PLUS II. To experiment with the design, the complete design was downloaded to the UP2 board supplied by Altera as part of university educational program. The board contains two programmable logic devices the MAX[R] 7000 Complex Programmable Logic Device (CPLD) and a FLEX 10K Field Programmable Gate Array Device

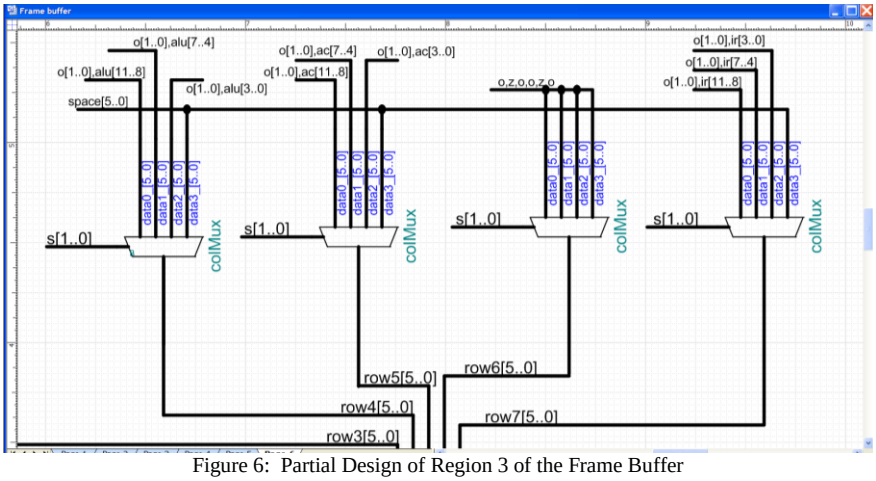


Figure 6: Partial Design of Region 3 of the Frame Buffer

In the above schematic, the S[1..0] selects from among 4 columns to display on the. The outputs of the multiplexers are used as data inputs to a row select multiplexer. The row multiplexer then selects one of the character codes available

(FPGA). The computer design was downloaded to the FPGA part of the UP2 board. This part contains two seven-segment displays and a VGA interface used to control a monitor display when connected to the interface.

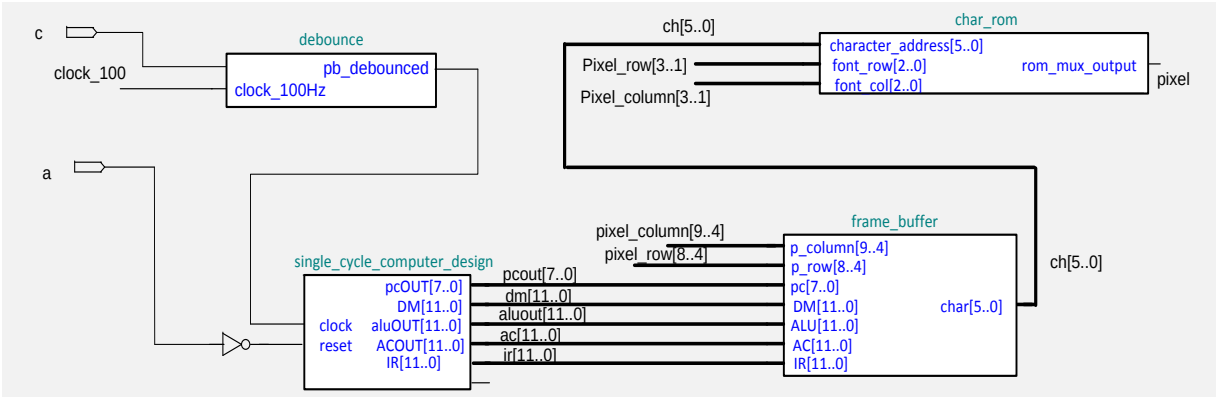


Figure 7: Partial design of computer and frame buffer interface

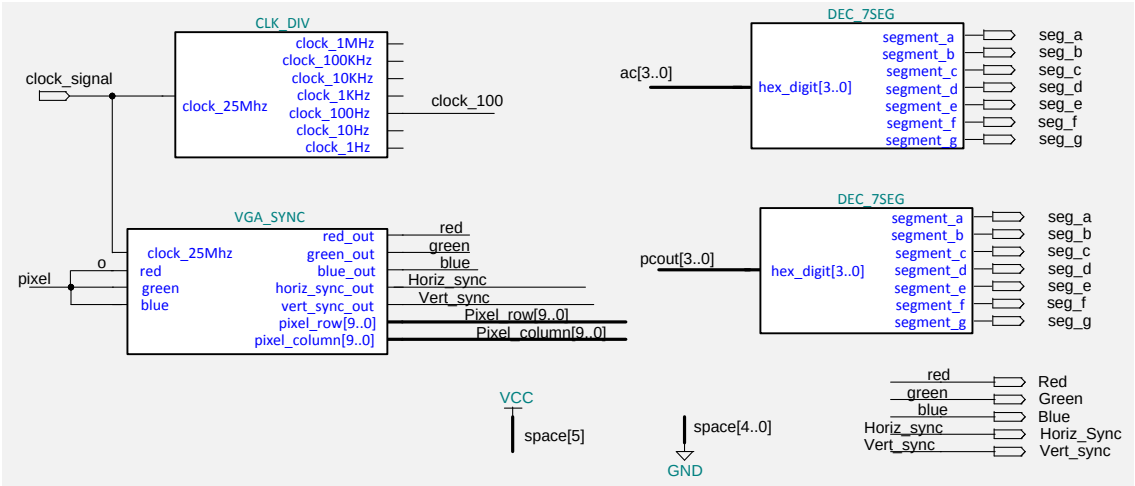
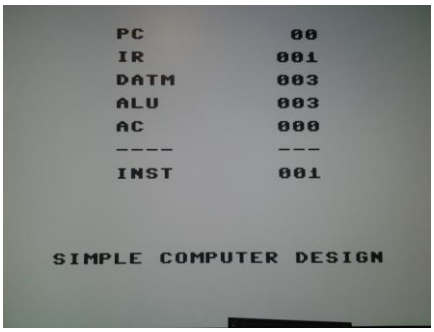


Figure 8: Remaining design of computer and frame buffer interface

Figure 7 and Figure 8 show the high-level block design in MAX+PLUS II.

The design of the auxiliary units is found in [17]. We have implemented the design and downloaded the design to the Altera UP2 board. A partial screen capture of the monitor display is shown in Figure 9(a). The display shows the register contents of a multiplication program shown in Figure 9(b). Pushbuttons on the up2 board are used to clock the design or to reset the program counter. The program counter is reset to the first instruction of the multiplication program. On clocking the design we step through the instruction set of the sample program. The accumulated result is stored in data memory location 3.

In the screen capture shown in Figure 9, column 1




```
[00..FF]  :  0;    % M[00] to M[FF] = 0000 initial word value is 0                %
   0      :  001;  %  LDA     1   --      AC = B                                   %
   1      :  301;  %  NAND    1   --      AC = B'                                  %
   2      :  600;  %  INC         --      AC = -B                                  %
   3      :  102;  %  STA 2       --      M[2] = -B (LOOP COUNTER)                 %
   4      :  000;  %  LDA 0       --      AC = A                                   %
   5      :  203;  %  ADD 3       --      AC = A + M[3], M[3] = partial product    %
   6      :  103;  %  STA 3       --      M[3] = M[3] + A                          %
   7      :  002;  %  LDA 2       --      AC = LOOP COUNTER                        %
   8      :  600;  %  INC         --      AC = AC + 1 (INCREMENT COUNTER)          %
   9      :  102;  %  STA 2       --      m[2] = m[2] + 1                          %
   A      :  500;  %  SKZ         --      IF LOOP COUNTER = 0 THEN EXIT            %
   B      :  403;  %  BUN 3       --      START NEW ADD ITERATION                  %
   C      :  003;  %  LDA 3       --      AC = M[3] = PRODUCT                      %
   D      :  40C;  %  AC = 3      --      LOOP BY REPEATING ABOVE INSTRUCTION      %
```

Figure 9: Screen Capture of Monitor in Standard VGA

contains the constant region. This region contains: the computer registers, DATM where the result of multiplication is stored in data memory, the three dashes, and the current instruction. The title region displays the simple computer design title. The variable region is column 2, where the contents of the program counter, instruction register, arithmetic logic unit, accumulator, current instruction and result in data memory is displayed.

5 Conclusion

While the study of computer design is normally covered in a first or second course in the hardware track of computer science, minimal or no coverage of peripheral units is studied. Topics such as actual VGA or frame buffer designs are normally reserved for upper classes in computer engineering. In this paper we looked at how the study of such units can be incorporated early in digital design classes based on the knowledge needed to accomplish the design. In particular we looked at the design of a frame buffer at the schematic level without the need to employ hardware description languages.

References

[1] N. Weste and D. Harris. "CMOS VLSI Design: A Circuits and Systems Perspective 4th edition". Addison Wesley, 2011.

[2] S. Brown and Z. Vranesic. "Fundamentals of Digital Logic with VHDL Design with CD-ROM, 3rd edition". McGraw-Hill, 2009.

[3] T. Floyd. "Digital Fundamentals: A System Approach". Prentice Hall, 2013.

[4] W. Kleitz. "Digital Electronics A Practical Approach with VHDL, 9th edition". Prentice Hall, 2012.

[5] M. Mano and M. Cilette. "Digital Design, 5th edition". Prentice Hall, 2013.

[6] R. Tocci, N. Widmer and G. Moss. "Digital Systems Principles and Applications, 11th edition". Prentice Hall, 2011.

[7] J. Wakerly. "Digital Design Principles and Practices 4th edition". Prentice Hall, 2006.

[8] D. Patterson and J. Hennessy. "Computer Organization and Design, Revised 4th edition". Morgan Kaufmann, 2011.

[9] W. Stallings. "Computer Organization and Architecture, 9th Edition". Prentice Hall, 2013.

[10] D. Harris and S. Harris. "Digital Design and computer Architecture, 2nd edition". Morgan Kaufman, 2013.

[11] J. Hayes. "Computer Architecture and Organization 3rd edition". McGraw-Hill, 1998.

[12] J. Hennessy and D. Patterson. "Computer Architecture: A Quantitative Approach 5th Edition". Morgan Kaufmann, 2011.

[13] M. Mano. "Computer System Architecture, 3rd Edition". Prentice Hall, 1993.

[14] Altera, www.Altera.com.

[15] Electronics Workbench, www.electronicsworkbench.com.

[16] H. Farhat. "A Simple Computer Design with Monitor Interface: Integrating Hardware and Software in Early Research in Computer Science". Proc. ICSNC, 218-222, 2008.

[17] H. Farhat. "Including Computer Design in Early Hardware Study of Computer Science". Int'l Conf. Frontiers in Education: CS and CE, FECS'13, pp. 263-267, July 2013.

[18] J. Hamblen and M. Furman. "Rapid Prototyping of Digital Systems, a Tutorial Approach". Kluwer Academic Publishing, 2001.