A Simple Computer Design with Monitor Interface: Integrating Hardware and Software in Early Research in Computer Science

Hassan Farhat
University of Nebraska at Omaha
farhat@unomaha.edu

Abstract

In computer science, the introduction of CAD tools and programmable logic devices makes it feasible to complete the design of a functioning computer during a second course in the hardware track.

In earlier we presented a simple instruction set and gave a complete single cycle design as part of grant for early research in computer science. For this we used the Altera CAD package and the UP2 board.

The contribution of this paper is to expand on previous work. We adopt a similar instruction set as done in earlier paper. Here, however, we interface the design to include a computer monitor. The complete computer design with monitor interface at an early stage in computer science education helps in introducing research early in the curricula.

1. Introduction

The CPU design is a topic of interest that spans the disciplines of computer science as well as computer engineering. The span includes basic electrical circuits, Very Large Scale Integration (VLSI) [3], digital design [4] to [12], computer organization [13] to [15], and computer architecture [16] to [18].

In [1] we presented a computer design that can be introduced early in computer science curricula. The design is used to complement digital design and computer organization textbooks. CPU design is normally studied during junior year in computer science. In fact, the complete computer design is missing from many textbooks; only segments of the design are presented.

In this paper we expand the design of the simple single-cycle computer to include interface to a monitor. This makes the outcome a more attractive instructional tool to use in early research in computer science. In establishing the monitor interface we use a set of VHDL modules as presented in [19]. As in [1] the instruction set chosen is minimal and conforms to instruction set completeness.

The paper is organized as follows. In section 2 we review the instruction set and discuss instruction set completeness as it applies to the set. In addition in the section we briefly look at the design of the single-cycle computer. In section 3 we develop the monitor interface aspects of the design. Section 4 includes experimental results with the design downloaded to the ALTERA up2 board and connected to a monitor. Section 5 contains a brief discussion of the use of the computer. Section 6 contains the conclusion.

2. The Instruction Set of the Single-Cycle Computer

For the computer architecture we use a simple accumulator-based architecture and a simple instruction set that includes one addressing mode only, the direct addressing mode. In addition, a minimal instruction set is used. The set has seven instructions and is chosen to conform 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, [1], is given in Table 1.

Table 1: Instruction set of computer

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

Expanding the functionality of the computer can be accomplished by adding macros that perform arithmetic and logic functions. Examples: 1) logical operations such as AND, OR, NOT and XOR; 2) additional arithmetic instructions such as subtraction and multiplication; and 3) additional conditional branches. Other expansions include clearing register contents. When studying performance of computers, these concepts can be applied to actual designs for comparative purposes.

Following instruction set design and assembly writing the design of the single cycle computer is presented. The block diagram of the single cycle computer is shown in Figure 1.

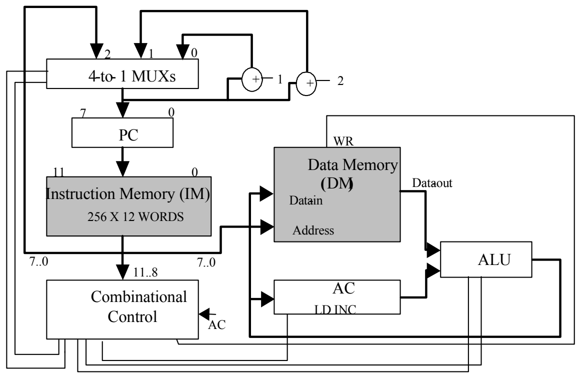


Figure 1: Schematic of simple computer

Table 2 describes the binary code given in each field and the corresponding function to be performed. The needed ROM contents are given in Table 3.

The single-cycle computer design was completed using MAX+PLUS II. The design was mostly at the behavioral level in VHDL. The control unit is one exception; it was completed using schematic capture. The memory elements are another exception; the memory units were generated using the Mega-Wizard Plug-in Manager.

Table 2: Used in determining ROM contents

PC MUX Code, Function	ALU Code, ALU Output	AC.L, AC.INC, DM.WR
00, PC $\leftarrow$ PC + 1	00, ALU $\leftarrow$ DM[x]	0: do not perform corresponding function
01, PC $\leftarrow$ PC + 2	01, ALU $\leftarrow$ AC	
10, PC $\leftarrow$ x	10, ALU $\leftarrow$ AC + DM[x]	1:perform corresponding function
11, Not used	11, ALU $\leftarrow$ AC NAND DM[x]	

Table 3: ROM contents as control words

	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

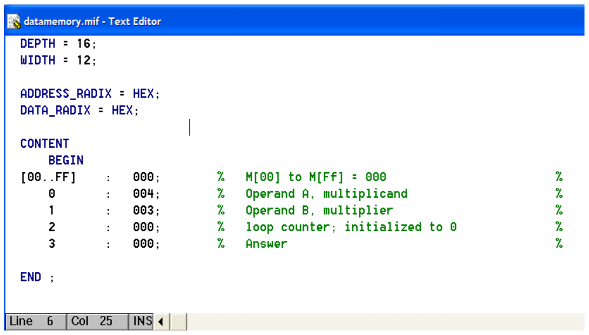
In the design, duplication of units such as adders and memory is covered as part of drawback in single cycle design.

The control unit design is among the most complex aspect of overall design. When single cycle design is used, the control is a combinational circuit. As a result, the simplest realization of the control lines is as a look-up-table stored in ROM. The ROM control word has the form

PC.MUX	ALU	AC.L	AC.INC	WR
2	2	1	1	1

Control Word Format

Screen-captures of memory initialization files, data and instruction memory, are shown Figures 3 and 4, respectively. The files were used in simulation of multiplication as repeated addition. The multiplicand (4) is added to itself three times as indicated by the multiplier value (memory location 1).

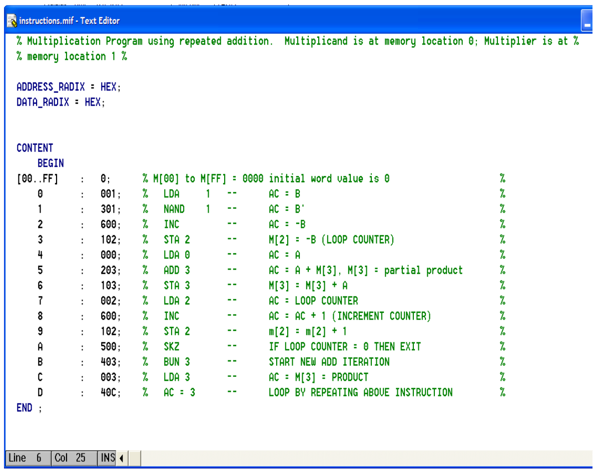


datamemory.mif - Text Editor

```
DEPTH = 16;
WIDTH = 12;

ADDRESS_RADIX = HEX;
DATA_RADIX = HEX;

CONTENT
    BEGIN
[00..FF]    :    000;      %   M[00] to M[FF] = 000                    %
    0       :    004;      %   Operand A, multiplicand                 %
    1       :    003;      %   Operand B, multiplier                   %
    2       :    000;      %   loop counter; initialized to 0          %
    3       :    000;      %   Answer                                  %

END ;
```

Line 6 | Col 25 | INS

Figure 3: Initial memory data contents



instructions.mif - Text Editor

```
% Multiplication Program using repeated addition.  Multiplicand is at memory location 0; Multiplier is at %
% memory location 1 %

ADDRESS_RADIX = HEX;
DATA_RADIX = HEX;


CONTENT
    BEGIN
[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 %
END ;
```

Line 6 | Col 25 | INS

Figure 4: Example multiplication program

A screen capture of the computer simulation is shown in Figure 5. As can be seen from the figure, the computer correctly computes the product of words 0 and 1 of data memory.

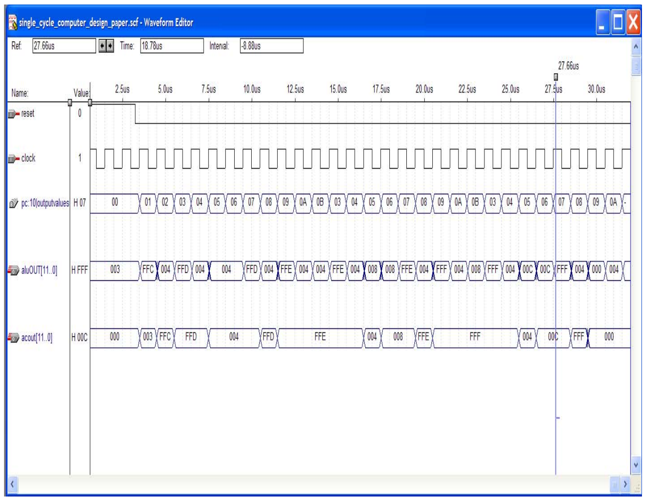


Figure 5: Timing Simulation

3. Adding Monitor Interface, the Modified Design

The design introduced in [1] allows for simulation of the computer functionality as given in the previous section. We expanded the design to include interface to a monitor. This allows the student to perform real time testing. The results can be seen using the monitor and two seven-segments displays. In addition the user can control the clock transitions using pushbuttons.
To interface with the monitor we adopt modules as given in [19]. The complete schematic capture design is given in Figure 6.

The design of the single-cycle computer, as discussed in the previous section, is shown in the middle lower part of the window. The remaining units are needed to interface with the user when the design is downloaded to the up2 board. The seven-segment displays interfaces are shown in top right corner of the window. The interface to monitor is through the VGA sync module written in VHDL, [19]. The clock division is used to make the computer run at the user speed, the actual onboard clock is 25 MHZ. This clock speed gives 40 ns time to display a 640 by 480 standard VGA image.

The additional units are to generate the monitor frames. The units serve as the frame buffer of the computer. The frame buffer is composed of 3 areas: 1) an area with blank color (background color); 2) an area with constant data; and an area with variable data. The constant area includes the names of the registers used in the computer design. The variable area shows the contents of the registers as instructions are read from memory and displayed on the screen.

4. Downloading the Design

The above design was simulated for correctness and downloaded to the up2 board. The board contains a VGA adapter connector. A monitor cable can be plugged into this adapter. A sample run with the adaptor connected to a monitor is shown in Figure 7. The figure shows the areas of the screen discussed above. The up2 board has two sets of pushbuttons as shown in the figure. The design was downloaded to the FLEX, SRAM based CPLD, the right side of the up2 board.

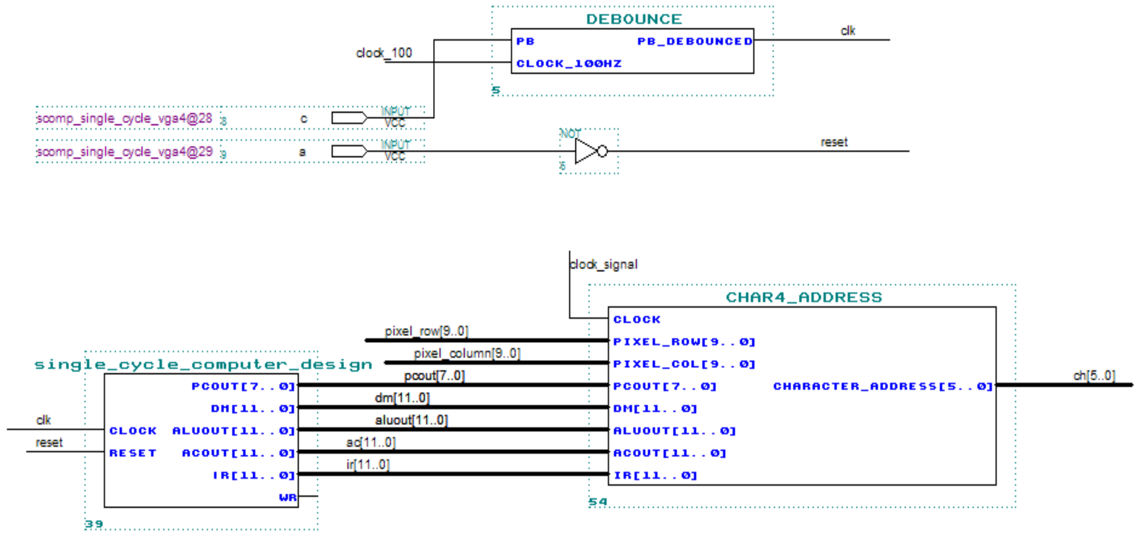


Figure 6: Complete computer design

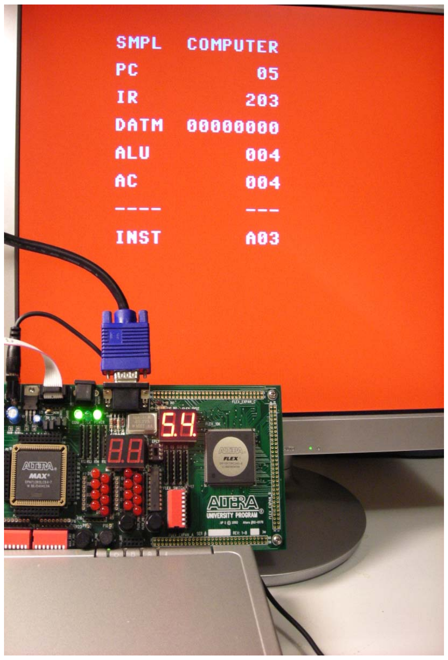


Figure 7: Monitor interface

The data and instruction memory were initialized to the contents shown in Figures 3 and 4, respectively. The figures show an example multiplication program that may serve as initial point of discussing computer performance. The first three instructions for example compute the negative of the operand, B.

As can be seen from Figure 7, the constant area of the monitor (column 1) displays the registers PC, IR and AC. In addition, the area displays the contents of the data memory and an abbreviation of the instruction used (last row). The first row displays SMPL (Simple) computer. The contents of the memory and the registers are displayed in column 2.

In Figure 7, the screen capture shows the state of the computer after the fifth clock pulse (fifth time the left pushbutton is pressed). The contents of PC = 05, IR = 203, ALU = 004, AC = 004, INST = A03 (add). These values are confirmed as read from Figure 4 (Instruction 5 in the figure). The value of AC and the ALU are also confirmed by referring to the data memory. On pressing the pushbutton (representing the clock) again one obtains Figure 8, confirming the next instruction execution. The instruction is a store instruction as shown from the contents of IR and referring to Table 1. The contents are stored in the data memory at location 03. This location holds the accumulated partial product computed so far. In this case, it is the value of A as a first iteration in the repeated sum.

5. Use of Computer Design

The computer was done as part of a grant that introduces students to early research in the sciences field. In earlier curricula, the design of a computer was a topic that is tackled only in electrical engineering at the senior or graduate level. In computer science, the design, with the minimal set, can be introduced early after two classes in the hardware track. This can be accomplished with minimal reference to electrical concepts. Introducing the computer science students to hardware realization, as complex as a simple computer, early in the curricula (sophomore level) aids in integrating research in software and hardware. The students can explore the hardware realization of many software algorithms.

6. Conclusion

The contribution of the paper is to expand on the single-cycle computer design presented in [1]. As can be seen from the previous section this makes the interface to the design more attractive instructional tool. Similar to [1], the advantage of the proposed designs is in the simplicity of covering the several design aspects. In the paper we expanded the design so as to allow for downloading to the ALTERA up2 board and to interface with a monitor in standard VGA mode. With the advances in CAD tools, designs of completely functioning computers with monitor interface can be discussed early in computer science curricula with minimal knowledge needed in electronics. The complete computer design with monitor interface at an early stage in computer science education helps in introducing research early in the curricula.

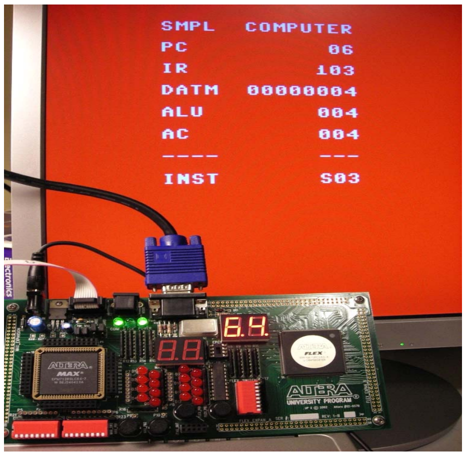


Figure 8: Monitor Display after a store instruction

7. References

[1] H. Farhat, “Integrating Hardware and Software in Early Research in Computer Science”, *Proc. ISCA CAINE,* Honolulu, Hawaii, USA, 2005, pp. 390-394.

[2] H. Farhat, “A comparative study of Max+PLUS II and Quartus schematic capture tools,” International Conference on Systems Engineering (ICSE2006), 2006, pp. 119-124.

[3] Weste N. and Karman E., *Principles of CMOS VLSI design A Systems Perspective 2nd edition*, Addison Wesley, 1993.

[4] Brown S. and Vranesic Z, *Fundamentals of Digital Logic with VHDL Design.* McGraw-Hill, 2000.

[5] Dewey A., *Analysis and Design of Digital Systems with VHDL*. PWS publishing, 1997.

[6] Floyd T., *Digital Fundamentals with VHDL*. Prentice Hall.

[7] Katz, R. and Gaetano B., *Contemporary Logic Design, 2nd edition*. Prentice Hall, 2005.

[8] Kleitz W., *Digital Electronics A Practical Approach*. Prentice Hall, 2008.

[9] Mano M. M. *Digital Design, 3rd Edition*. Prentice Hall, 2003.

[10] Tocci R., Widmer N. and Moss G.,*Digital Systems Principles and Applications*. Prentice Hall, 2007.

[11] Vyemura, J., *A First Course in Digital Systems Design an Integrated approach*. Brooks/Cole, 2000.

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

[13] Carpinelli J. D., *Computer Systems Organization and Architecture*. Addison-Wesley, 2000.

[14] Patterson D. A. and Hennessy J. L., *Computer Organization and Design: The Hardware/Software Interface* 3nd edition. Morgan Kaufmann, 2005.

[15] Stallings W., *Computer Organization and Architecture, 5th Edition.* Prentice Hall, 2000.

[16] Harris, D. and Harris, S., *Digital Design and computer Architecture*. Morgan Kaufman, 2007.

[17] Hayes J. P., *Computer Architecture and Organization* 3rd edition. McGraw-Hill, 1998.

[18] Hennessy J. L. and Patterson D. A., *Computer Architecture: A Quantitative Approach* 3rd edition. Morgan Kaufmann, 1996.

[19] Hamblen J. and Furman M., *Rapid Prototyping of Digital Systems, a tutorial approach.* Kluwer Academic Publishing, 2001.