Asian J. Adv. Basic Sci.: **2015**, *4(1)*, *5*-11
ISSN (Print): 2454 – 7492
ISSN (Online): 2347 – 4114
www.ajabs.org

Teaching Computer Organization Using Field Programmable Gate Array: An Incremental Approach

Mohammad Awedh[*] and Ahmed Mueen

King Abdulaziz University, Jeddah, *SAUDI ARABIA*
[*] *Correspondance: E-mail: mhawedh,@au.edu.sa and mueen@kau.edu.sa*

(Received 31 July, 2015; Accepted 03 Aug, 2015; Published 08 Aug, 2015)

ABSTRACT: This paper describes a novel approach to teach computer organization using an incremental approach of working models on FPGA. It presents the design concepts and realization of MIPS-based architecture into the teaching tool. Our new approach has improved the student learning and understanding of the major concepts in computer organization. Our main goal of this study is to compare our method (model) of teaching to the one that was used prior to this study in our institute.

Keywords: FPGA; MIPS processor; RTL; Verilog; Hardware Description Language; Synthesis; CAD tools; Computer Organization; Assembly Language; Incremental approach and working models.

INTRODUCTION: The course "computer organization and architecture" is a common course for computer engineering students, which plays a central role in the computer engineering curriculum offered at universities throughout the world (Holland & Hauck, 2003). Teaching this course to computer engineering and computer science students solely on textbook materials is inadequate (Hatfield & Rieker, 2005) and (Kasim et al., 2005). Students have to depend on their imaginations to understand the fundamental hardware-related concepts. Traditionally, most courses in computer architecture and organization include practical work in the laboratory. It is not possible to build a laboratory that can offer several computer architectures for teaching computer organization and architecture. Furthermore, having up-to-date information needs to be in touch with the rapid evolution of the computer technology and industry. Therefore, searching for an effective way of teaching computer organization and architecture is a continuing task (Hatfield & Rieker, 2005).

Literature Review: There are some modern teaching approaches that combined effective hardware design and hardware implementation to help students understanding computer architecture concepts and encourage students to further work on the field (Hala, 2012). One of these approaches is teaching with software simulators, which have several advantages over real microcomputer platforms; they are not costly and more flexible. In addition, graphical presentation and animation support students to better understand various design issues (Rodriguez et al., 1998). By simulating important features of a processor, students can obtain a better understanding of the internal operation of a processor (Yurcik and Wolffe, 2001). Different simulators oriented to help in teaching specific concepts in modern computer architectures, such as cache memory, pipelining, and superscalar organizations (Grnbacher, 1998). In another works, researcher (Li & Chu, 1932). and (Sugawara & Hiraki, 2006) use active tool for teaching computer organization and architecture by taking benefit of simulation and Field Programmable Gate Array (FPGA) technology. FPGA technology helps designing high performance systems at low cost(Sugawara & Hiraki, 2006). This paper presents novel approach to teach computer organization using an incremental approach of working models on FPGA. Our methodology of teaching digital computer organization course differs from the previous (traditional) one in two aspects. The first aspect is the use of the incremental approaches of working models. This teaching tool helps computer engineering students to be familiarized practically with computer organization through development, design, and implement of a functional and synthesizable processor. The second aspect is the use of Field programmable gate arrays to implement a real processor. Hence, the students feel the joy of designing and running a functional processor in real hardware. The students also have the chance of adding more instructions and improve the performance. The previous teaching method for the course was based on theory of computer construction. This method teaches students how computers work by studying their inner details (e.g.,

pipelining and cache control) through assembly programming of real life processors and organization simulators. However, this method lacks teaching the students computer organization through the process of building a real functional processor from scratch. Recent technological advances in FPGA and its availability in low cost allow our approach to become a reality. FPGA is a powerful hardware prototyping platforms based on reconfigurable hardware. FPGAs have been efficiently used by engineers for developing digital systems, prototyping, design exploration, and experiments (Aws et al., 2011). Designers use an HDL to describe the behavior and structure of system and circuit designs (Hana et al., 2013). FPGAs have shorten the design time significantly. Potential applications using FPGAs are extremely wide (Jong et al., 2012). Examples are application-oriented soft processors, embedded systems, and systems on programmable chip, programmable network chips, and many others. One of the early uses of FPGA to implement a processor is presented in (Li & Chu, 1996). In this paper, a hardware design and implementation of pipelined RISC processor in FPGA chip is presented. Some recent results in teaching reconfigurable systems are discussed in (Sklyarov & Skliarova, 2005) (Jean et al., 2015). Where models, methods and pedagogical innovations (such as using the developed student-oriented design templates and evaluation through mini-projects) are considered. Our teaching is based on the textbook by Patterson and Hennessy (Patterson & Hennessy, 2014). This paper is structured as follows. We first briefly introduce the proposed methodology.

Designers use an HDL to describe the behavior and structure of system and circuit designs. Understanding FPGA architecture allows you to create HDL code that effectively uses FPGA system features.

Hardware Platform: Our hardware platform is based on a simple RISC-based MIPS architecture (Patterson & Hennessy, 2014) implemented in Digilent Basys2 Spartan3E FPGA Board (Digilent, 2015). The FPGA board, Figure 2, contains I/O devices (peripherals) among which, 8 LEDs, 4-digit seven-segment display, 4 pushbuttons, and 8 slide switches are used, Figure 1.

For MIPS processor to access the I/O devices on the FPGA board, we use memory mapped I/O technique. Three special memory addresses are used to access these I/O devices. Table 1 lists these I/O devices and their addresses. We use the least significant byte (8 bits) of an internal register for switches and LEDs; we use the least significant half word (16 bits) of an internal registers for the 4-digit seven-segment display.

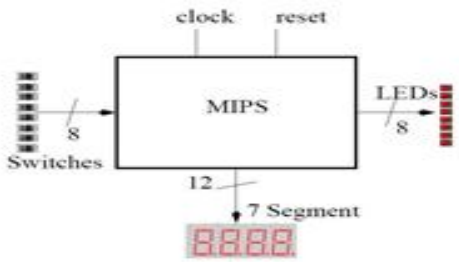

Figure 1: Block diagram.

Figure 2: Digilent Basys2 FPGA Board.

Table 1: Memory-mapped I/O addresses.

I/O	Address
7 Seg	0xD000
Leds	0xE000
Switches	0xF000

MATERIAL AND METHODS: This paper describes an incremental approach of teaching computer organization based on the idea of "working models" (Hermie, 2010). These "working models" are based on pedagogical designs presented in Patterson & Hennessy's text Computer Organization and Design (Patterson & Hennessy, 2014).

In our approach, we present to the students fully-functional processor in different levels of design abstraction. In each level of design abstraction, students modify and extend these designs to enhance their understanding of digital computer design; instruction set design, performance optimization as well as developing design validation skills. Two level of design abstraction are used: Behavioral modeling and register transfer level (RTL) modeling. Behavioral modeling is often used to explore high-level features and functions; RTL modeling is often used to describe block diagram view of a circuit at the level of registers, flip-flops and functional blocks that constitute the design.

Behavioral Description of a Simple MIPS Processor: The first phase in our teaching methodology is to help our students understand and realize the detailed operation (function) of processors. A behavioral description (in Verilog) of a simple single-cycle MIPS processor was presented to the students. This Verilog module of the single-cycle MIPS processor is fully synthesizable and implementable in FPGA. The processor implements the following subset of MIPS instructions: or, ori, and, andi, beq, sub, add, slt, lw, sw, j.

The students were asked to modify the Verilog behavioral description of the MIPS machine to run the following MIPS code. The code has some MIPS instructions that are not implemented in the given Verilog behavioral description.

```
        .text
        ori     $s0 ,   $0 , 0xD000     # Address of 7−segment
        ori     $s1 ,   $0 , 0xE000     # Address of LEDs
        ori     $s2 ,   $0 , 0xF000     # Address of SWs
        lw      $s6 ,   0( $s2 )        # read switches
loop    srl     $s7 ,   $s 6 ,2
        xor     $s7 ,   $s6 , $s7
        sw      $s7 ,   0( $s1 )        # Display onto LEDs
        sw      $s6 ,   0( $s0 )        # Display result in 7−Seg
        addi    $s6 ,   $s6 , −1
        bne     $s6 ,   $0 ,loop
here    j       here
```

RTL Description of a Simple MIPS Processor: The goal of this phase is to teach the students the design of a processor on a structural level. We use the principles and techniques to design and implement a single-cycle processor as explained in Chapter 4 of (Hermie, 2010). Figure 3 depicts the simple single-cycle MIPS processor as described in (Hermie, 2010). The design implements a subset of the core MIPS instruction set:

- The memory-reference instructions load word (lw) and store word (sw)
- The arithmetic-logical instructions add, sub, and, or, and slt
- The instructions branch equal (beq) and jump (j)

A complete functional and synthesizable Verilog description of a single-cycle processor (Figure 4) was presented to the students and they were asked to modify the Verilog code to implement procedures (Call and Return) in MIPS. Section 2.8 of (Hermie, 2010) (Supporting Procedures in Computer Hardware) talks about supporting procedures in MIPS. This section was explained in detail prior to the assignment. To support such function, students have to implement two new instructions: **jal** and **jr**. The students first have to modify the single-cycle design of MIPS (Figure 3) and show how the basic data path and control unit can be extended to handle these new instructions. This required an addition of the proper control signals and hardware (e.g., multiplexers). Then, they implement their modifications into the provided Verilog code. Figure 4 shows the complete top-level system design using Xilinix ISE CAD tool the amount of FPGA resources used.

Pipelined MIPS Processor: The third phase of our teaching is to implement the piplined MIPS. The goal is for our students to gain a deeper understanding of pipelined processor implementation.

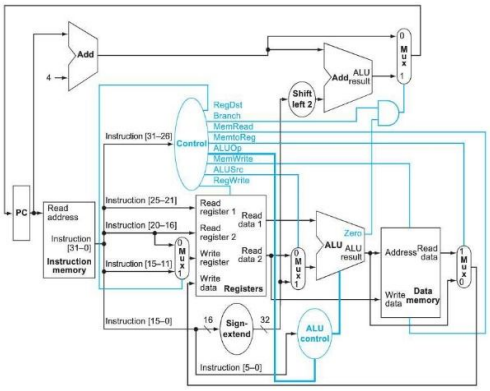

Figure 3: Single Cycle MIPS Processor.

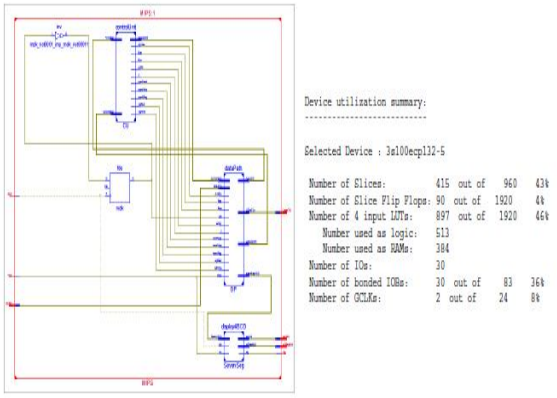

Figure 4: Top-level Design of Single-Cycle MIPS processor.

A fully synthesizable Verilog module of the data path of the pipelined MIPS (Hermie, 2010), Figure 5, is given to the students. The students were asked to implement the pipelined control unit on the datapath implementation.

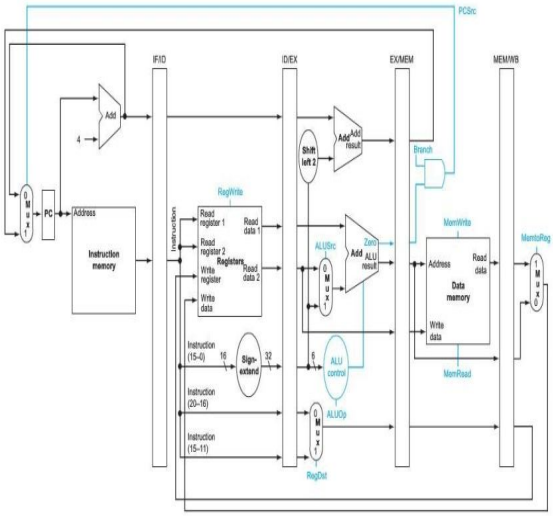

Figure 5: The pipelined datapath with the control signals identified.

Final Projects: The final project is to extend the pipelined implementation to perform data forwarding and data hazard detection. After the students have successfully implemented their pipelined processors, they were given real programs that failed to run because of the data hazard. These programs are control-hazard free.

Even though, data forwarding and data hazard are well explained in (Hermie, 2010), some students struggled to have their processors correctly implemented in FPGA. Hence, we decided to make the final project a group project of two students.

RESULTS AND DISCUSSION: Our new approach had been applied for three consecutive semesters. The number of students in each semester is 10 on the average. Therefore, we decide to evaluate our methodology after the third semester. Any new enhancement and improvement to the new approach will be applied for a future teaching of the course. However, surveys are conducted in each semester. To prepare for applying our teaching approach, a survey was conducted at the first day of the class. The goal is to evaluate students' background and readiness. This survey helped in preparing labs, extra materials and tutorials. A set of questions are related to the profile data of the students attending the courses. Information collected includes the background materials in Digital Design – related to combinational and sequential design of digital systems. Another set of questions are related to using Hardware Description Language (HDL) for design, debugging and functional simulation of digital systems; and the implementation of digital circuits using FPGAs. Table 2 shows the average results of some of the questions in the survey for the three semesters.

Table 2: Students Background in Using HDL.

Which HDL do you know (choose one)?	Verilog	VHDL	Others	None
	85%	10%	0%	5%
Which design abstraction level do you use HDL?	Behavioral	RTL	Logic	Physical
	100%	40%	30%	0%
Have you ever used HDL to design FPGA devices?	Yes		No	
	5%		95%	
I used HDL for	Only Simulation		Simulation and Synthesis	
	95%		5%	

To evaluate our teaching approach, two surveys were conducted; one was after the second phase and the second at the end of the semester. The goal of the first survey is to evaluate the students' progress as well as their feedback to improve the course. Among the questions of this survey were a set that evaluates our method so far and gains students feedback. The five-point Likert scale items (Likert, 1932) were used in these surveys and the average students' responses for the three semesters are presented.

Table 3 shows the average students responses on our teaching methodology for the three semesters. The results show the success of our new approach. However, pipelined processor design had not been included yet. We believe, complex design (like pipelined process design) will be a major and better factor to evaluate the success of our teaching methodology. Hence, we decide to include the same set of questions in the final survey (at the end of the semester).

We also include in this survey the evaluation of the tools that we used in our teaching: Verilog as a hardware description language and FGPA as an implementing hardware. Table 4 shows students' feedback on using these teaching tools. Another set of questions are related to the course contents and students concern. Table 5 shows the students feedback.

The final survey was conducted at the end of each semester. The first set of questions is dedicated to evaluate the overall quality of the course and compared to other courses in the Computer Engineering curriculum. Table 6 shows students feedback. Table 7 shows the overall quality of the teaching tools.

We also included in the final survey the two questions that were asked in the second survey. The goal is to

evaluate our methodology after students finish their implementation of pipelined design and final projects. In addition, a couple of questions were added to evaluate the overall methodology. Table 8 depicts the average students' responses for the three semesters.

Table 3: Evaluation of Our Teaching Methodology: First Feedback.

Question: Modifying working models and adding new instructions helped me realized and understand computer design				
Strongly Agree	Agree	Undecided	Disagree	Strongly Disagree
77%	16%	7%	0%	0%
Question: Incrementally increase the complexity of a design helped me understand process design				
Strongly Agree	Agree	Undecided	Disagree	Strongly Disagree
81%	18%	1%	0%	0%

Table 4: Student Evaluation of Using HDL and FPGA.

	Too Easy	Easy	Fair	Hard	Extremely Hard
Using Verilog to describe digital design	5%	60%	20%	10%	5%
Using Verilog for Functional Simulation	82%	13%	3%	2%	0%
Using FPGA to implement digital design	0%	10%	45%	30%	15%

Table 5: Course Contents and Students Concern.

	Too Easy	Easy	Fair	Hard	Extremely Hard
Using Verilog to describe digital design	5%	60%	20%	10%	5%
Using Verilog for Functional Simulation	82%	13%	3%	2%	0%
Using FPGA to implement digital design	0%	10%	45%	30%	15%

Table 6: Overall Quality of the Course.

Overall importance of the courses for Computer Engineering students				
Not important	Slightly important	Fairly important	Important	Very important
0%	2%	11%	24%	63%
Classification among all courses in Computer Engineering				
Too Easy	Easy	Fair	Hard	Extremely Hard
5%	12%	23%	29%	31%
Degree of motivation to pursue hardware studies				
Too Low	Low	Moderate	High	Too High
3%	4%	13%	37%	43%
Overall quality of the contents presented in the course				
Too Easy	Easy	Fair	Hard	Extremely Hard
0%	10%	45%	30%	15%

Table 7: Overall Quality of the Teaching Tools.

The availability of resources				
Too Easy	Easy	Fair	Hard	Extremely Hard
94%	6%	0%	0%	0%
The complexity of the tools				
Too Easy	Easy	Fair	Hard	Extremely Hard
23%	52%	12%	7%	6%
Using Verilog to describe and simulate digital design				
Strongly Agree	Agree	Undecided	Disagree	Strongly Disagree
74%	21%	4%	1%	0%
Using FPGA to implement digital design				
Strongly Agree	Agree	Undecided	Disagree	Strongly Disagree
63%	26%	7%	1%	3%

Table 8: Evaluation of Our Teaching Methodology: Second Feedback.

Modifying working models and adding new instructions helped me realized and understand computer design				
Strongly Agree	Agree	Undecided	Disagree	Strongly Disagree
79%	13%	7%	0%	1%
Incrementally increase the complexity of a design helped me understand process design				
Strongly Agree	Agree	Undecided	Disagree	Strongly Disagree
78%	19%	1%	0%	2%
The Overall teaching method used is useful and helpful in understanding complex digital design				
Strongly Agree	Agree	Undecided	Disagree	Strongly Disagree
76%	24%	0%	0%	0%
The teaching technique used is adequate for other computer engineering courses				
Strongly Agree	Agree	Undecided	Disagree	Strongly Disagree
44%	51%	3%	1%	1%

The outcomes of the surveys will be used to improve teaching the course in the future and suggest improvements to the prerequisite courses as well.

CONCLUSION: This paper has described the use of FPGA to reinforce the concepts of Computer Organization design using an incremental approach in which students extend and modify existing models. It highlights the importance of design process abstraction levels for teaching hardware design to students in a hands-on manner.

A positive feedback we got is from the senior project committee. The committee has stated that a small, however increasing, number of senior projects use soft processor in their senior projects has been noticed. Some students have modified the original design of the processor and include it in their design of embedded systems.

Students' responses to our new teaching methodology have been enthusiastic; they indicate in comments of the final survey that the method has greatly enhanced their understanding of processor design. However, some students suggested to use simpler processor; mostly suggested 8-bit version of the MIPS structure as presented in (Hermie, 2010). In addition, almost all students suggested working in fixed number of groups (two or three) from the beginning of the course.

In future courses, it would be beneficial to include the design of a cache memory in our design methodology; students may implement and improve some cache-related design issues. For instance, improving cache performance, applying different methods of handling cache misses and writes. Furthermore, final projects can be more beneficial by including more design issues. For instance, the implementation of control hazard (specifically, the static and dynamic branch prediction) and improving cache performance.

To reach to the above mentioned steps of processor and cache design, we are working, as also suggested by the students, in using simpler MIPS processor. Simple processor design helps to expedite and debug design without scarifying in understanding the major concepts of computer organization design.

REFERENCES:

1. Aws Yousif, Fida El-Din and Hasan Krad (2011) Teaching Computer Architecture and Organization using Simulation and FPGAs, IJIET, 1(3), 190 194.
2. Digilent (2015) Digilent Basys 2 Spartan-3E FPGA Board. http://www.digilentinc.com /Products/Detail.cfm?NavPath=2,400,790&Prod= BASYS2.
3. Grnbacher H (1998) Teaching computer architecture/organization using simulators, *Proc. Frontiers Educ. Conf.*, 1107-1112.
4. Hatfield B. and Rieker. M (2005) Incorporating simulation and implementation into teaching

computer organization and architecture. In *35th ASEE/IEEE Frontiers in Education Conf*, pages FIG–18, Indianapolis, USA.

5. Hana Park, Young-Woong Ko, Jungmin So and Jeong-Gun Lee (2013) Synthesizable Manycore Processor Designs with FPGA in Teaching Computer Architecture International Journal of Control and Automation, 6(5), 429-438.

6. Hala ElAarag (2012) Teaching computer organization: a practical approach. Journal of Computing Sciences in Colleges, 28, 210-1017.

7. Hermie. J. M. Voulgarelis (2010) The importance of physically built working models in design teaching of undergraduate architectural students. *2nd international conference on design education*, 28 june - 1 july.

8. Holland J. H. and Hauck. S. (2003) Harnessing fpgas for computer architecture education, *Proceedings of the IEEE Intr. Conf. on Microelectronic Systems Education (MSE03)*.

9. Jean-Luc Dekeyser, Ahmad Shadi Aljendi (2015) Adopting New Learning Strategies for Computer Architecture in Higher Education Case Study: Building the S3 Microprocessor in 24 Hours. Workshop on Computer Architecture Education held in conjunction with the 42nd International Symposium on Computer Architeture, Portland, United States.

10. Jong Hyuk Lee and Seung Eun Lee and Heon Chang Yu and Taeweon Suh (2012) Pipelined CPU Design With FPGA in Teaching Computer Architecture. Education, *IEEE Transactions*, 55(3), 341-348.

11. Kasim. M., Al-Aubidy, and A. A. Samadi (2005) Simulation and fpga implementation of a simple computer. In *The 7th Middle Eastern Simulation Multiconference "MESM2005"*, 151–158.

12. Likert, R. (1932) A technique for the measurement of attitudes. *Archives of Psychology*.

13. Li. Y. and Chu.W. (1996) Using fpga for computer architecture/organization education. In Proceedings *of the 1996 Workshop on Computer Architecture Education*, WCAE-2 '96, New York, NY, USA.

14. Likert. R. (1932) A technique for the measurement of attitudes. *Archives of Psychology*, 22(140).

15. Patterson D. A . and Hennessy J. L. (2014) *Computer Organization and Design*. Elsevier, MA, USA, 5th edition.

16. Rodriguez .L. Pardo, M. Moure, M. Valdes, and E. Mandado. Viscp(1998) a virtual instrumentation and cad tool for electronic engineering learning. In *Frontiers in Education Conference, 1998. FIE '98. 28th Annual*, 3, 1095–1099.

17. Sklyarov V and Skliarova I. (2005) Teaching reconfigurable systems: methods, tools, tutorials, and projects. *Education, IEEE Transactions on*, 48(2), 290–300.

18. Sugawara. Y and Hiraki. K (2006) A computer architecture education curriculum through the design and implementation of original processors using fpgas, In *36th ASEE/IEEE Frontiers in Education Conference*.

19. Yurcik. M. H. W, Wolffe G. D. (2001) A survey of simulators used in computer organization/architecture courses. *Proceedings of the 2001 Summer Computer Simulation Conference (SCSC)*.