A Processor Design Course Project: Creating Soft-Core MIPS Processor Using Step-by-Step Components' Integration Approach

Ali Elkateeb

Abstract—**Design and implementation of a soft-core MIPS processor using field-programmable gate array (FPGA) technology will be addressed in this paper. Teaching processor architecture and design is considered the key element of the student learning in the undergraduate Computer Engineering program; as such, this project was developed to enrich students' experience in this field. This paper presents a practical introduction to soft-core processor design through the use of step-by-step integrating of the processor's components. Students implemented their processors using Xilinx ISE design tools and downloaded their designs to Xilinx ML 501 FPGA boards, which used Xilinx Virtex 5 chips. A designed and developed soft-core processor provided students with a starting point for applying their designed processors in follow-up courses such as Embedded Systems and Senior Design Project. Students' assessment of the soft-core processor design was analyzed, which indicated that they are confident in confronting the next step of constructing advanced processor architecture.**

Index Terms—**MIPS processor, field-programmable gate array (FPGA), Xilinx design tools, VHDL.**

I. Introduction

Teaching processor design is considered the key element in student learning for the undergraduate computer engineering program. It is essential that students obtain strong knowledge in processor architecture concepts, design methodology, design tools, and technology during their processor design course.

The concepts of complex instruction set computer (CISC) and reduced instruction set computer (RISC) have been taught in the processor design course for many years. However, the design simplicity feature of the RISC-based processors, specifically MIPS architecture, makes such architecture very attractive in teaching processor design courses at the undergraduate level [1].

The processor design approach using components integration can serve as a model for teaching processor design and development [2]. In this approach, the processor's components that students designed, simulated, and tested during the processor design course will be used in their final processor design project course. For example, students can start their course with developing the processor's components such as arithmetic logic unit (ALU), register file (RF), control unit (CU), and program counter (PC). Students will then integrate these components in designing their processor for the final course project. Such design approach will aid students in gradually gaining experience in processor design.

Student learning of top-down and bottom-up design methodologies, in addition to the structural, behavioral, and mixed modeling using hardware description language (HDL), are essential to achieve processor design and development successfully. These design methods and HDL modeling constitute the core of the teaching materials for processor design where students will not only streamline the design but also shorten the processor implementation time [3], [4]. For example, using the VHDL structural and behavior modeling are required for learning bottom-up and top-down design methodology. Also, learning mixed modeling is important to connect and integrate all processor's components to achieve the final design.

The use of reconfigurable technology such as the field programmable gate array (FPGA) has become very effective and suitable for teaching processor design. Students can implement their design on the FPGA chip; if the processor fails to function according to the design specifications, students have the option to modify their designs. In addition, students can take advantage of the FPGA technology feature of providing fast design updates in which they can reprogram their designs until they become satisfied with the design outcomes.

Using the design tools such as ISE from Xilinx will help students simulate and verify their design functionality and performance before implementing their final processor. Also, the design tools can be used to produce the design's downloadable file, also called 'bit' file, that can be downloaded to program the FPGA chip. Furthermore, exposing students to the use of logic analyzers devices will be useful for design testing and debugging.

The processor design approach using components integration will help students to integrate the processor components in a step by step manner until they completed their final processor design. MIPS non-pipelining processor designs. Students' perception of using the step by step approach in developing their final processors and using the FPGA board for their design implementations will be obtained during the end of the term course.

This paper is organized as follows: in Section II we examine previous work related to teaching processor design. The course structure and teaching methodology will be discussed in Section III. In Section IV, the laboratory assignments will be discussed, while the details of step

Manuscript received November 8, 2011; revised December 25, 2011.

A. Elkateeb is with the Department of Electrical and Computer Engineering at University of Michigan, Dearborn, USA (e-mail: elkateeb@umd.umich.edu).

bystep development of the soft-core MIPS processor will be presented in Section V. Students' feedback is addressed in Section VI. The paper is concluded in Section VII.

II. Previous Work

Typically, all computer engineering undergraduate programs offer one or more courses on computer architecture and design. However, the teaching depth of these courses materials is varied from one university to the other. While some universities are focused on the theoretical concepts of processor design, other universities cover all the key elements required for teaching processor design that are mentioned in the previous section [5]-[8].

The flexibility and reprogramming capability of the field programmable gate array (FPGA) has made such technology widely used in academic environments for teaching processors design. Students can download their designs on the same FPGA chip numerous times until they are assured that their designs are operating according to the specifications. Hence, students can gain the essential hands-on-experience on designing, implementing, testing, and debugging processors using actual FPGA-based hardware [9].

The students' knowledge on HDL languages, such as VHDL and Verilog HDL, is an essential requirement to learn processor design [10]. However, insufficient training in HDL for undergraduate students was reported in many U.S. Universities [11]. To cope with this problem, instructors need to provide more intensive HDL learning techniques within the processor design course timeframe. This task has been achieved through the systematic use of auxiliary materials, such as processor components' templates, tutorials, and a library of design templates that are synthesized with industrial standard programmable design tools such as Xilinx ISE [12].

The availability of the inexpensive FPGA boards, low-cost PC-based logic analyzers, and free FPGA design tools have made processor design affordable as well as easily and efficiently achieved by students. The use of FPGA in teaching processor design course in which the methodology and design tools of computer design is outlined and discussed in some works [13]. Also, providing students with hands-on experience in developing processors by porting a standard processor design, such as POWEPC, to FPGA boards and building users logic around the processor have been achieved [14], [15]. The processor implementation using FPGA chip to support specific applications has also been addressed [16].

However, to the best of the authors' knowledge, the development of MIPS soft-core processor using a step-by-step design approach has not been currently addressed in any published paper. In this paper, we discuss how students gradually integrate processor components until they complete the final processor design. This step-by-step processor development approach has enriched student knowledge not just with MIPS architecture, but with VHDL programming and industrial standard Xilinx ISE design tools (in addition to processor implementations using virtex 5 FPGA boards) and exposes students to the use of logic analyzed devices in design testing.

III. Course Structure and Teaching Methodology

ECE 475 is the processor design course that must be taken by all senior computer engineering students at University of Michigan-Dearborn. The course materials include teaching students the theoretical aspects of the MIPS computer architecture in addition to the hardware description language, specifically VHDL. All students attending ECE 475 course have already learned the basic concepts of computer organization during their computer organization course (ECE 375), which is the prerequisite course for ECE 475. Also, the ECE 375 course provides students with hands-on experience in using ISE Xilinx FPGA design tools and specifically the schematic capture.

The ECE 475L is a laboratory section of the ECE 475 processor design course, where students are introduced to VHDL design flow using Xilinx ISE design tools, and learn processor design implementations and testing using the FPGA technology where students use ML501 Virtex 5 FPGA boards in their laboratory work. Also, students learn the use of the logic analyzed device in testing their designs.

Students learn, during their first three weeks of the ECE 475L laboratory section, the VHDL design flow, using logic analyzer device, and implementing a simple VHDL code using ML 501 Virtex 5 FPGA boards. The instructor conducts demonstrations to help students learn logic analyzer and the ML501 board. These demonstrations are essential to accelerate student learning and refresh students' memory with using ISE and schematic capture design tools which was taught in the previous computer organization course ECE 375. During this period, students learn the VHDL structural modeling at their ECE 475 course. We believe that start teaching VHDL language with VHDL structural modeling will provide students with smooth transition from what they learned at ECE 375 of schematic capture to VHDL design using structural modeling at ECE 475 course. For instance, students work on design and developing MIPS Arithmetic Logic Unit (ALU) that supports few simple instructions using VHDL structural modeling with bottom-up design methodology.

Moreover, students learn the top-down design methodology using VHDL behavioral modeling in ECE 475 course and they applied that they learned on ECE 475L by design and developing other processor components such as register file, instruction and data memory, control unit, and clock divider. The instructor provides students with tutorials and design templates for some of these components to ensure students can complete the development of these components in the time allocated to the laboratory sessions.

The last step in processor design course is to teach students the MIPS processor design through the final course project. Students are introduced to components integration using VHDL mixed modeling where they learn how to integrate their developed processor components to achieve the design of the MIPS RISC processor. The step-by-step design approach has been used by students to achieve their processors, as will be shown in Section V.

The clock divider component is not part of the processor architecture but is a necessary component to help students reduce the clock rate of 100MHZ provided by the ML501 FPGA board to 1Hz clock rate. This very low clock rate can be used by students to run their MIPS processor design and help to visibly observe the processor operations. Also, students can utilize the board's LEDs and switches to examine the functionality of instructions and their execution results. In addition, students can use logic analyzer devices to test the processor operations at 100MHz speed. However, the setup time of the device fly leads is usually a time consuming process making some students more comfortable with using a clock divider approach and the boards' LEDs and switches over the use of the logic analyzer device.

IV. Laboratory Assignment

During the laboratory section ECE 475L of the computer design course, students gain hands-on experience in processor design using VHDL. Different processor components have been designed during the laboratory sessions where their operations are evaluated through functional and timing simulation. Their designs are then downloaded to the FPGA boards where the LEDs and switches that are available on these boards are used to test the components' design. There are six laboratory sessions in ECE 474L and these are as follows:

1) VHDL design flow: The first lab is designed to help students understand VHDL design flow process and the process of downloading designs to FPGA boards.
2) Using logic analyzer devices: Students are introduced to the use of logic analyzer devices that are essential for their design testing where their design runs at the 100MHz clock rates of the FPGA boards.
3) ALU design using VHDL structural modeling: Students design an ALU for MIPS architecture using VHDL structural modeling where they learn the bottom-up design approach. Components from Xilinx library are instantiated to develop ALU.
4) ALU design using VHDL behavior modeling: This lab is similar to lab 3 where the same ALU will be designed but with using VHDL behavioral modeling. Using behavior modeling will expose students to the top-down design approach. As students work with both modeling, they observe the simplicity of developing designs using behavior over structural modeling.
5) Register file design using VHDL behavior modeling: A dual-port register file of four registers is designed in lab 5 where register writing occurs on the positive edge of the clock where the register reading occurs on the negative clock edge.
6) Moore state machine: Students design a Moore state machine to encode the instructions operation codes to control the processor data path.

As students complete the development of the MIPS components, they are directed in the final course project to integrate their designed components and develop other required components to produce the final soft-core MIPS processor.

V. Step-by-Step Approach for Teaching Soft-Core MIPS Design and Development

Teaching MIPS processor design is a challenge when students have just started to learn the VHDL language and have little/no experience in processor and systems design. Therefore, a step-by-step design method is appropriate where students progressively develop their learning experience in this important computer engineering field.

In using such method, students will design the MIPS processor by gradually integrating the processor components developed during their laboratory course and develop other components required for their final soft-core MIPS processor design. Whenever students complete each of these six steps of the MIPS processor design, they use their Xilinx ML501 Virtex 5 FPGA boards to perform design testing for that step. When students completed all these design steps, the architecture for the non-pipeline soft-core MIPS processor design, as described in Hennessy and Patterson book [1], will be achieved. To help students complete their design work during the ECE 475 course period, processor components have been simplified. For instance, the ALU is designed to support the basic instructions as will be mentioned in the following processor design steps, four registers are the size of the register file, the instruction memory is limited to 16 locations, and the processor data path is designed to be a 16-bit architecture. The three types of the instruction format, i.e., the R-type, I-type, and J-type, for the 16-bit MIPS architecture are as following:

OP_Code (15-12)	Rs (11-10)	Rt (9-8)	Rd (7-0)

R-Type Instruction format

OP_Code (15-12)	Rs (11-10)	Rt (9-8)	Label (7-0)

I- Type Instruction format

Note: *Op Code* field is the Operation Code field,
Rs is the source register address field,
Rt and Rd represents the target and destination registers fields respectively,
Label is representing the target memory address.

As the size of the memory is small, the J-type instruction format is not used in this course project and the I-type is used to serve both I and J-type instructions.

Students are asked to work in design groups of two students and to conduct presentation of their designs at the end of each design step. Using such teaching approach will help students to share design experience through discussion and exchanging ideas. The detail of each design step is as following:

Step 1: Core Processor Design

In the first part of the MIPS soft-core processor course project, students are requested to design simple processor core that includes the register file, ALU, and control unit [Fig. 1]. These components are already designed by students during their lab assignments and students will need to integrate them to produce the processor core. Students will use the structural model of the ALU, the behavioral model of the register file, and the control unit in this step. The mixed VHDL modeling is used in connecting components developed with VHDL structural modeling with other components developed using behavioral modeling. As the students used a positional association technique for ports mapping of the VHDL code for structural modeling of the ALU design, they asked to use named association technique for port mapping to generate the processor core. This has benefited students to understand the use of these two different types of port mapping in developing different components and integrating them to produce the processor core.

The register file has two reading ports and one writing port. Each port size is 16-bits. Also, the register file has a “reg-write” signal to control writing data on one of its four registers. The register file is designed to perform register reading operation on every rising edge clock cycle. The register writing occurs when the reg-write signal is asserted by the control unit at the falling edge of the clock.

Store data in the register file can be achieved through loading data from the ALU output or from the external data bus that is represented by external switches. The ALU has been designed to support the following instructions at this design step: add, subtract, logical and, logical or, add immediate, exclusive-or, complement, and load external data. The control unit is a Moore type state machine which is designed to execute these instructions and produce signals that control the operations of this simple processor core. The instructions are provided to the control unit from the FPGA on-board switches that represent the instruction register, where students can use these switches to provide their instructions manually one at a time to the control unit for decoding purpose. The load external data is the instruction that a student uses to load the register file with the data that can be used by other students in the program. The “src-sel” signal will be generated by the control unit to select the data source from either the output of the ALU or from the external switches that will be saved in one of the registers in the register file.

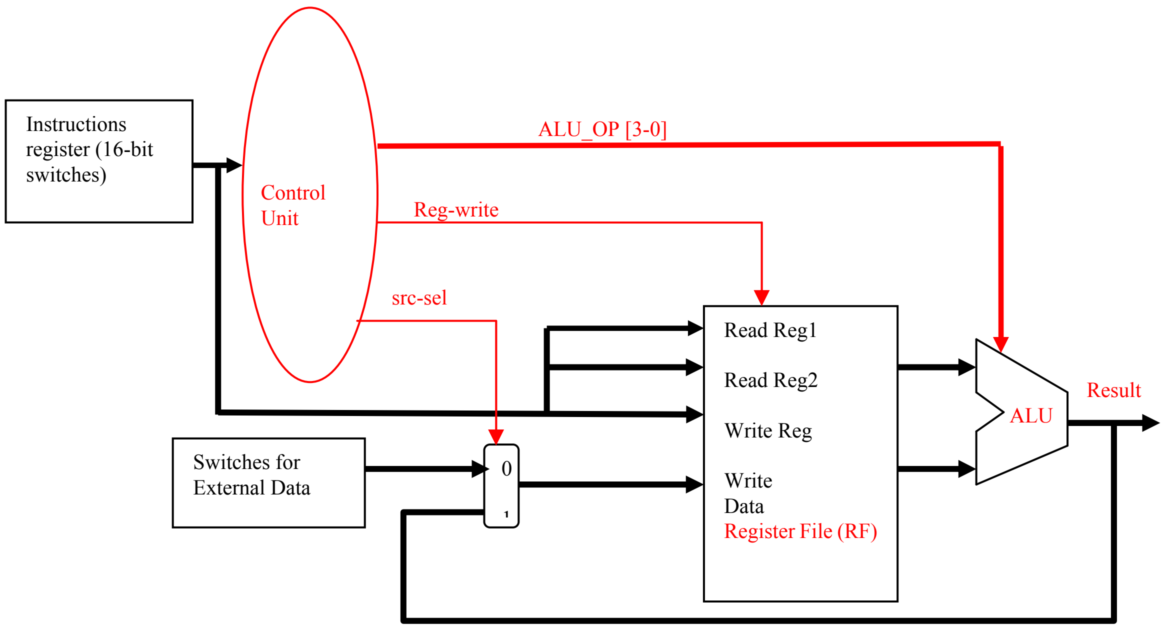

Fig. 1. Step 1: Simple Processor Core

Step 2: Instruction Memory and Program Counter

Using the switches of the FPGA board for implementing the processor’s instructions register during the first part of the course project has helped students to test the functionality of each instruction. Students have used the switches to enter each instruction manually and test its operation. In this part of the project, students have replaced the 16-bit switches that represent the instruction register with small instruction memory (IMEM), where they performed their programs testing without entering each instruction using the switches that exist on the FPGA board. Also, a program counter (PC) is required to track the processed instructions [Fig. 2]. The IMEM size contains 16 locations, which can hold 16 instructions. Students generate a small program that includes all the processor’s instructions and store that program inside the IMEM. The program counter will be incremented with the execution of each instruction, which provides the testing for all implemented instructions. To support the processor architecture with IMEM and PC, the processor’s control unit is designed to provide signals such as “PC-INC” to increment the program counter, “PC-LD” to load the program counter, and “IM-READ” for reading the IMEM. Fig. 2 shows the processor designed by students in step 2 where the dotted lines represent the processor section that designed for the step 2 of the course final project.

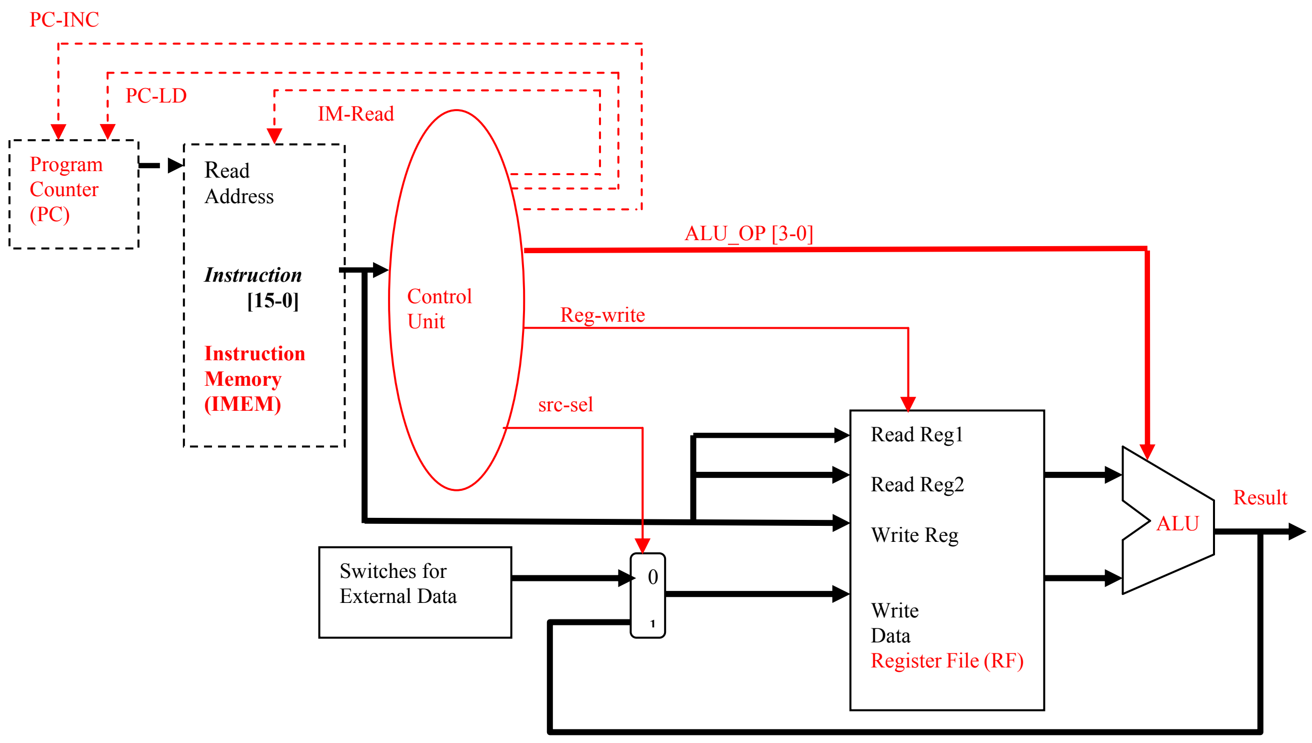

Fig. 2. Step 2: Processor enhanced with instruction memory and program counter

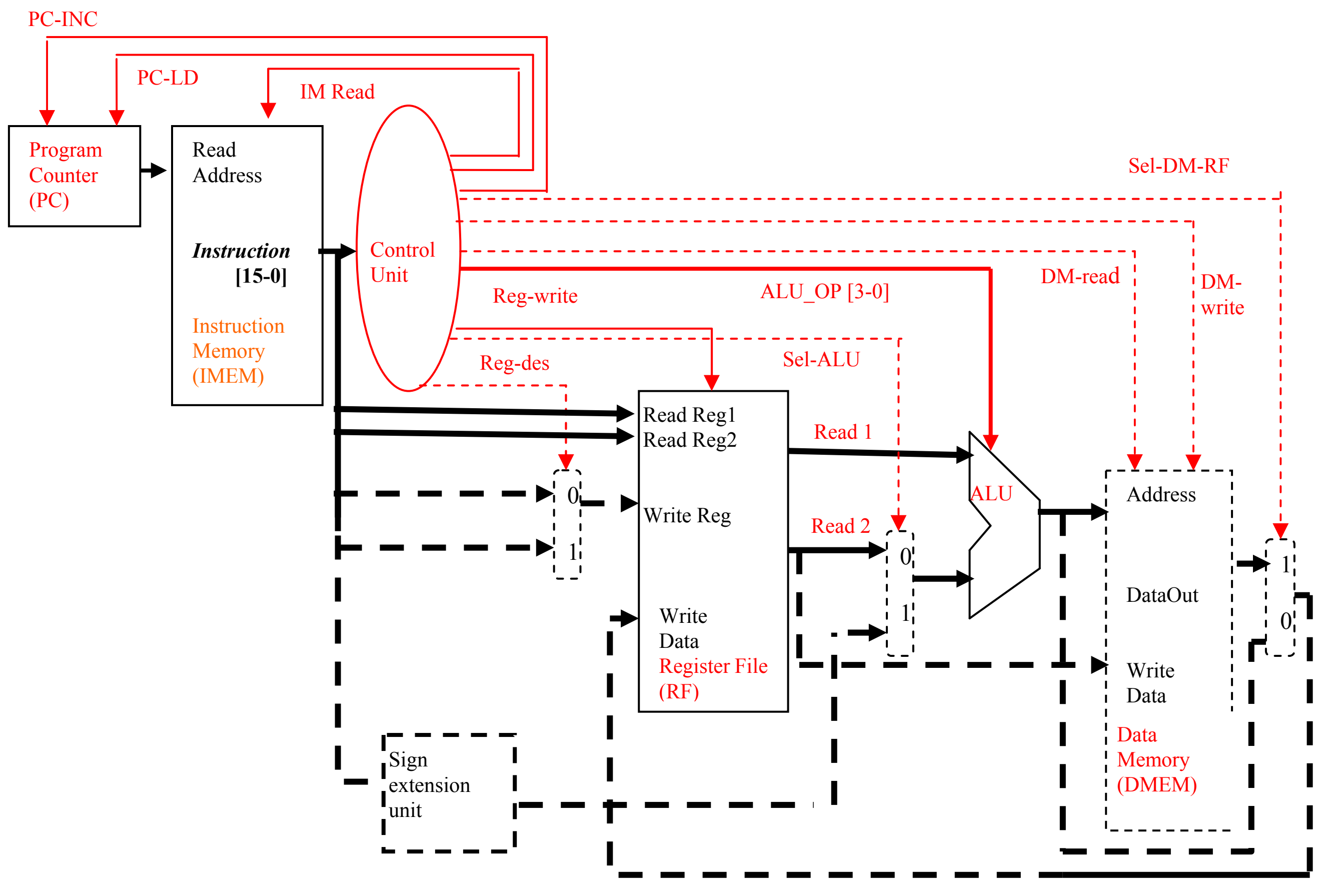

Fig. 3. Step3: Processor enhanced with Data memory and sign extension unit

Step 3: Data Memory and Sign Extend

In the third step, students were asked to upgrade their processor designs to support data memory and sign extension unit. To support these two units, students required to upgrade their instructions set by including two new instructions, the Load Word (LW) and Store Word (SW). Also, the processor data path will be updated to include some multiplexers to manage data transfer between the processor's components [Fig. 3]. One multiplexer is inserted at the output of the data memory where it is used to select between two sources, the output of the ALU and the output of data memory. One of these multiplexer inputs will be connected to the write data port of the register file.

The second multiplexer is located at the read 2 port output of the register file to select between the output of the sign extension unit and the read 2 port of the register file. The output of this multiplexer is connected to the lower input of the ALU. The third multiplexer is located at the Write Reg port of the register file to select the destination register address that is located in different positions in the processor instructions format of R-type or I-type.

The multiplexer at the lower input of the ALU is required to select between either a register to perform arithmetic/logical operation or the output of the sign extension unit. The sign extension of the memory offset for the LW and SW instructions will be added to memory based address, which is provided from one of the registers in the register file. Clearly, the data memory address space is very small for this project, but the use of producing the memoryaddress by adding the offset to the based address will allow students to understand the address generated in real MIPS processor. The control unit will generate the input signals for the multiplexers' select inputs port according to the current processed instructions.

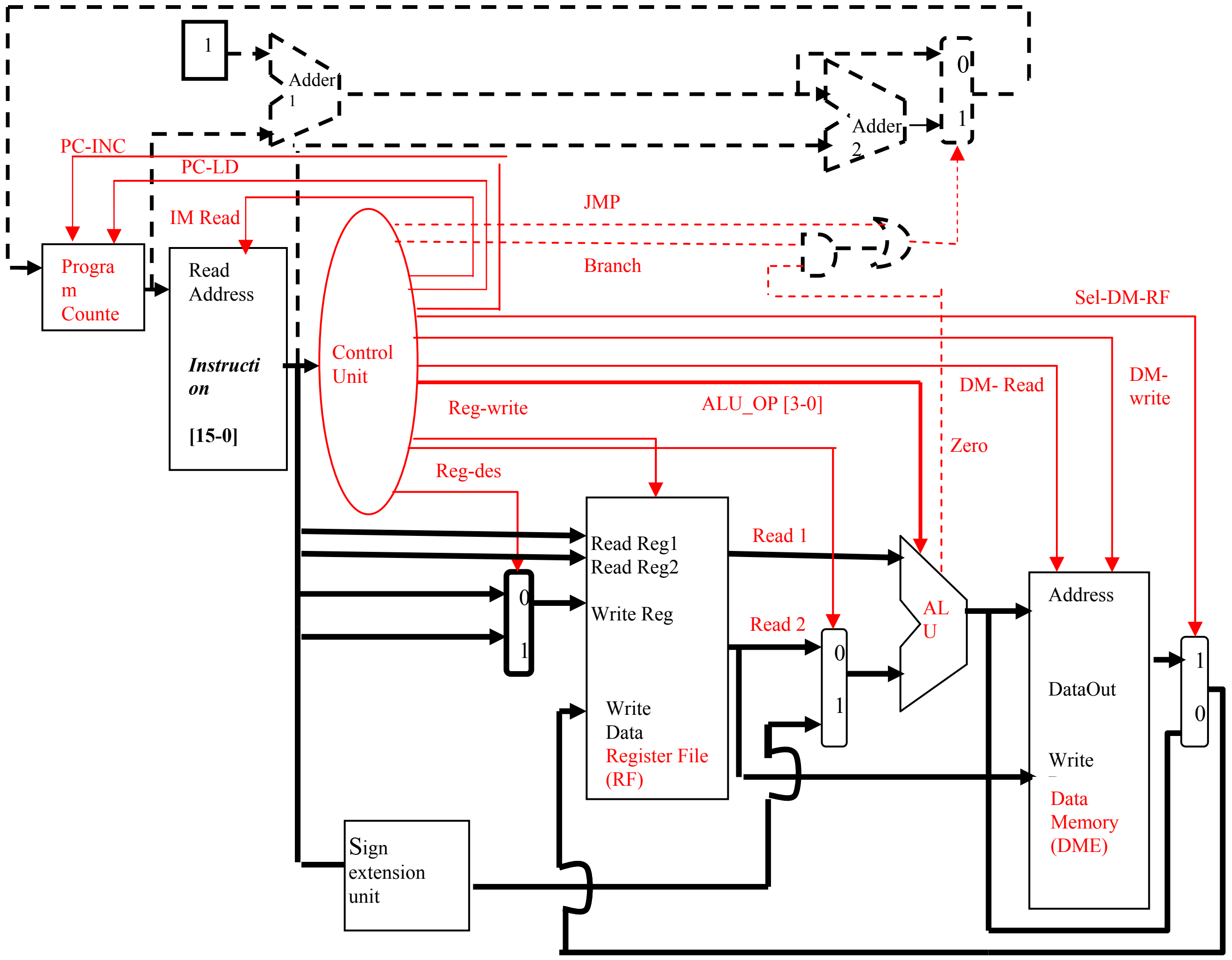

Fig. 4. Step4: Enhance data path with branch equal and unconditional jump instructions

Step 4: Branch Equal and Jump instructions

This step is important to teach students how to enhance their processor architecture to support conditional branch and unconditional jump instructions. When students completed their design and implementation to the processor described in step 3, they asked to update their processor data path to these two instructions, as shown in Fig. 4. The processor data path has been enhanced with two adders. The first adder 'adder 1' is used for address increment for the next program counter value, while the second adder 'adder 2' is to update the program counter with the target address of the jump or the BEQ instructions (when the branch is taken). Incrementing the PC in this step of the project was achieved through updating the processor data path; previously this occurred inside the control unit in the first three steps. Such change is required in handling the PC update since Jump and BEQ instructions might change the program flow to the instructions' target address; otherwise, the PC should be incrementing to point to the next instruction address. The hardware AND/OR logic, shown in Fig. 4 is used to select the multiplexer input in this data path to update the PC. The control unit asserts the JMP or Branch signals whenever Jump or BEQ instructions are executed. The Zero's signal is generated from the ALU whenever the BEQ instruction will require to branch to the target address. Therefore, the PC will always be loaded with the target location specified by the Jump instruction, while it will be loaded with target location specified by the BEQ instruction when it is executed and the Zero signal is generated from the ALU simultaneously.

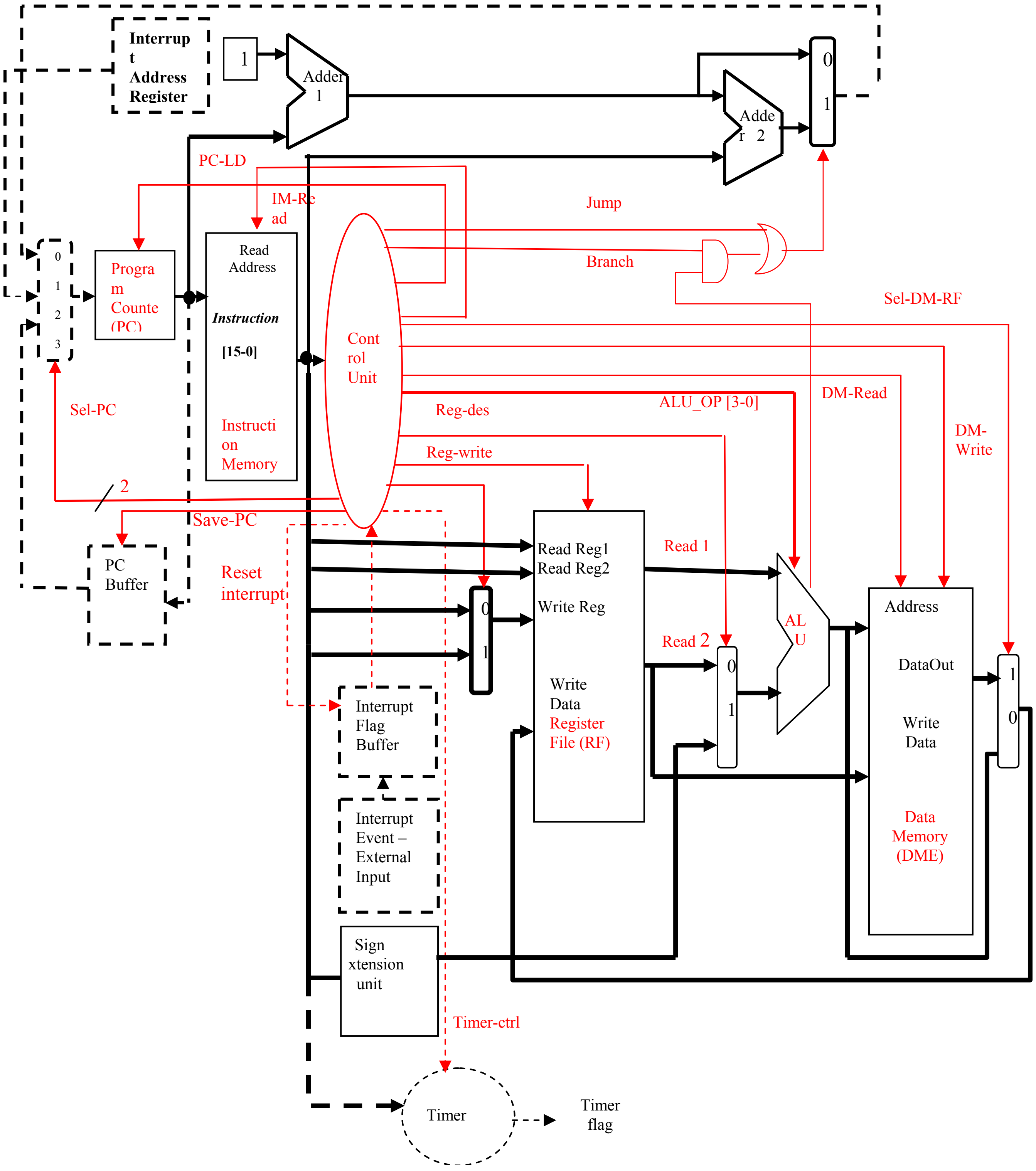

Fig. 5. Data path supported with interrupt and timer units

Step 5: Interrupt support

During the previous four steps of the processor design project, students obtained hands-on experience in design, implementation, debugging, and testing of the conventional MIPS processor. Support the processor design with interrupt and timer units, as discussed in this step and step 6 respectively, will help students to achieve their learning of full processor design cycle [Fig. 5].

In this step, students are asked to support their designed processor with an interrupt mechanism that has an interrupt service routine executed when a rising edge signal is asserted on the processor's interrupt request pin. A very simple interrupt service routine is used for testing the interrupt functionality. Also, a return from interrupt was developed where a return to the main program and specifically to the next instruction that located after the interrupted instruction position. Students have learned how the program counter is saved when the interrupt request occurs and how to restore it at the return from the service routine. Also, students learned how to revise their VHDL control unit design to support interrupt mechanism. To simplify the interrupt structure, a fixed location is used for the interrupt service routine where the interrupt address is saved into specific register called interrupt address register [Fig. 5]. A PC buffer is used to save the return address to the main program after the interrupt service routing is completed. Also, as the signal is asserted at the interrupt request pin, the interrupt flag buffer is used to latch the request until the control unit completed the execution of the current instruction. The interrupt flag buffer will be cleared when the control unit acknowledged receiving the interrupt request from the interrupt external pin.

Step 6: Timer support

During the last step of this course project, students supported their MIPS processor with a timer unit. The timer is 16-bit, decrements with every clock cycle and will raise a flag when counting reached the zero value. The timer is programmable where the start value of the timer can be set by a special "load timer" instruction.

The signal 'timer-ctrl' is produced by the control unit to reset the timer. The timer is designed as an independent component that adds to the data path of the MIPS architecture, as shown in Fig. 5.

VI. Student Feedback

A questionnaire was provided at the end of the processor design course to evaluate students' opinion on their course and laboratory sections. All 46 students that attended the computer design course were involved and responded to this evaluation study.

TABLE I: STUDENTS FEEDBACK

Question	Average rate (%)
1. Learning VHDL	
a) I am confident in using VHDL structural modeling in processor's components design.	94
b) I am confident in using VHDL behavioral modeling in processor's components design.	80
c) I am confident in using VHDL mixed modeling for components integration to implement the final processor.	80
d) The course material helped me understand the processor components design and how to integrate them to produce the final processor.	90
e) I am confident in developing processor components using VHDL.	81
f) The VHDL templates provided throughout the course material were very useful for processor's components and final processor design.	88
g) The course materials increased my confidence in digital systems design.	88
2. Laboratory work	
a) The lab work was very well organized and synchronized with course lectures.	90
b) The processor design and implementation was challenging.	96
c) I was able to develop an effective strategy for the processor testing.	92
d) I have learned how to use logic analyzer device for the debugging of the processor design.	70
e) I have gained very good experience in using the Virtex 5 FPGA board.	96
f) I have mastered the use of FPGA design tools.	88
g) I can implement the processor design on the FPGA board successfully.	90
h) Is the time allocated to complete each lab works is reasonable? If not, explain why?	70
3. Course learning impact	
a) I will use the designed processor in other courses such as senior design project and embedded systems courses.	86
b) The course materials will be useful in my graduate studies.	90
c) This project prepares me for future industrial work.	100

Students were requested to answer questions that were related to three groups: learning VHDL, laboratory work, and the course learning impact [TABLE I]. Each statement has a score from 1 to 5, with 1 indicating that they strongly disagreed while a 5 indicated they strongly agreed. We considered students' average score of 80% and above in responding to each question as good and no action would be taken to improve the course material related to that question. For the questions related to learning VHDL, students' learning of different VHDL modeling and use in designing and developing the processor was evaluated; student feedback indicated their satisfaction on the covered teaching materials.

For the laboratory work questions, students displayed concern regarding the use of the logic analyzer device and the time allocated to the structural modeling laboratory session. To address their concern on using logic analyzer device, we constructed a short summary about how to use the device, specifically the formatting of the 32 output channels, triggering channels, and the readings of device measurements. With regards to the time allocated for laboratory session, we tackled this issue by conducting a tutorial on ALU design using structural modeling and assigning more time to complete the laboratory session.

VII. Conclusion

A reference design for a soft-core processor development using a step-by-step methodology for processor's components integration has been presented. Over the course of an academic term, the FPGA technology was utilized to implement and test the developed MIPS processor. Students gained hands-on experience on structural, behavioral, and mixed modeling using a VHDL language. They also learned industrial standard FPGA Xilinx ISE and Modelsim design and simulation tools. Students were divided into groups where each group implemented and tested each design step on the Xilinx ML-501 FPGA Virtex 5 board before proceeding to the next step of their designs.

Student feedback regarding their learning experience was evaluated at the end of the course term. Students expressed their satisfaction on the Soft-core processor design project. Also, students enjoyed the laboratory section of the course even though some struggled with the theoretical aspect of the soft-core processor design course. Processor components integration on the FPGA boards has given students significant confidence in processor design and implementation.

References

[1] D. Patterson and J. Hennessy, *Computer Organization and Design*, Morgan Kaufman.

[2] J. Wing, "Computational Thinking," *Communications of the ACM*, March 2006, Vol. 49, No. 3, pp. 33-35.

[3] J. Bhasker (1999), *VHDL Primer*, Prentice-Hall, third edition.

[4] S. Yalamanchili (2000), *Introductory VHDL from simulation to synthesis*, Prentice-Hall.

[5] G. Brown and N. Vrana, (1995). "A computer architecture laboratory course using programmable logic," *IEEE Transaction on Education*, Vol. 38, No. 2.

[6] V. Sklyarov and I. Skliarova, "Teaching reconfigurable systems: methoda, tools, tutorials, and projects," *IEEE Tran. on Educ.,* Vol. 48, No. 2, pp. 290-300, May 2005.

[7] Computing Curricula, Computer Science [2004, sep. 10]. [On-line]. Avaliable on: http//www.celoxica.com/products/default.asp.

[8] A. Bindal, S. Mann, N. Ahmed, and L. Raimundo, "An undergraduate system-on-chip (soc) course for computer engineering students", *IEEE Trans. Edu.,* vol. 48, no. 2, pp. 279-289, May 2005.

[9] Y. Zhu, T. Weng, and C. Cheng, "Enhancing learning effectiveness in digital design courses through the use of programmable logic boards," *IEEE TRAN. Educ.,* vol. 52, no. 1, pp. 151-156, Feb. 2009.

[10] C. Rowen (2002), "Reducing SOC simulation and development time," *IEEE Computer*, pp. 29-34.

[11] A. Sagahyoon, "From AHDL to VHDL: A course in hardware description languages," *IEEE Trans. Educ*., vol43, no4, pp. 449-454, Nov, 2000.

[12] V. Sklyarov and I. Skliarova, "Teaching reconfigurable systems: method, tools, tutorials, and projects," *IEEE Tran. Educ.,* Vol. 48, No. 2, pp. 290-300, May 2005.

[13] R. D. Williams, R. H. Klenke, and J. H. Aylor, "Teaching computer design using virtual prototyping," *IEEE Trans. Educ.*, vol. 46, no. 2, pp. 296–301, May 2003.

[14] R. Foist, C. Grecu, A. Ivanov, and R. Turner, "*An FPGA design project: creating a powerpc subsystem plus user logic*," *IEEE Trans. Educ,* Vol. 51, no. 3, pp. 312-318, Aug. 2008.

[15] A. Sagahyoon, "From AHDL to VHDL: A course in hardware description languages," *IEEE Trans. Educ*., vol. 43, no. 4, pp. 449-454, Nov. 2000.

[16] J. Hamblen, "Rapid prototyping using field-programmable logic design," *IEEE Micro,* Vol. 20, N0. 3, pp. 29-37, May/Jun. 2000.

El Kateeb received his PhD from Concordia University, Montreal, in 1992 and his MSc from Kent University, UK, in 1980 and his BSc from University of Technology, Iraq, in 1976. He is currently an Associate Professor in the Department of Electrical and Computer Engineering at University of Michigan, Dearborn, USA.
His research interests include high-speed networks, computer architecture, reconfigurable computing, and computer applications. He has over 60 papers published in international journals and conferences.