An Interactive Remote Laboratory on Basic Computer Architecture Using Altera DE2 Board

http://dx.doi.org/10.3991/ijoe.v9i5.2675

A. Kalantzopoulos, E. Galetakis, C. Katsenos, and E. Zigouris
University of Patras, Patras, Greece

Abstract—This paper proposes an interactive Remote Laboratory (RL) on basic computer architecture. The students through this RL are able to understand the architecture, the organization and the operation of two 8-bit Central Processing Units (CPUs). The simple architectures of these CPUs which are also presented in this paper are ideal for introductory courses on computer architecture. The hardware implementations of two micro-computers based on the above CPUs were achieved using the Field Programmable Gate Array (FPGA) based Altera DE2 board. The implemented systems around these two micro-computers allow both local and remote control of the corresponding CPUs. The students using the proposed RL, select the desired micro-computer which is automatically downloaded to the FPGA. They are also able to program the selected micro-computer in assembly language and to observe the step by step execution of the downloaded code. This procedure is achieved using the virtual push buttons and toggle switches of the RL Graphical User Interface (GUI) which is accessible by a common web browser. The values of the registers, internal buses and the DE2 LEDs are also displayed in the GUI during the execution of the students' code. Through this process, the students become familiar with the assembly programming language and they also understand in depth the internal micro-operations of the implemented CPUs.

***Index Terms*— Computer architecture, Field programmable gate array (FPGA), Remote laboratory (RL).**

I. INTRODUCTION

Computer architecture courses are very popular in computer science and computer engineering among the universities all over the world. In most cases, the offered courses cover basic and advanced theoretical topics using lectures and simulators [1, 2].

The available simulators provide students with the ability to interact with the CPU (Central Processing Unit) and to observe the values of internal components such as registers and buses. Advanced simulators integrate compilers and assemblers in order to allow students to write and test their programs. In many cases the simulators have user-friendly GUIs (Graphical User Interfaces) which display the memory contents and the CPU states during the program execution. These simulators are important educational tools because they support the observation and the understanding of the CPU micro-operations [3-6].

For the first steps of students in the area of computer architecture and organization, simple architectures and their simulators have been proposed. Among others, the Very Simple (VS) CPU, the Relatively Simple (RS) CPU and the Machine Architecture that is Really Intuitive and Easy (MARIE), have been discussed in [7, 8]. Their architectures are very simple and they are ideal for the familiarization of students with low theoretical background, in basic topics on computer architecture.

The basic disadvantage of the above educational approach is the lack of hardware experience which is offered by the hands-on laboratories. Therefore, the students are not able to verify their theoretical knowledge by performing experiments on real systems and to improve their practical skills which are very important for their professional careers. In order to fulfill these obstacles many universities are offering hardware oriented courses in computer architecture and organization. Their hands-on laboratories are recently equipped with development boards based on Field Programmable Gate Arrays (FPGAs). The selection of these development boards allows the implementation of digital logic circuits and simple or advanced CPU architectures in real hardware, using Hardware Description Languages (HDLs) [9-14].

The educational approaches which are based on hands-on laboratory courses are the most suitable for the engineering education. However the hands-on laboratories are available to students only specific hours and require the physical presence of one or more instructors. An approach which was proposed by many universities in order to overcome the disadvantages of the hands-on laboratories is the Remote Laboratories (RLs) one [15]. The RLs allow students to perform their experiments by accessing the laboratory equipment through internet without time and place restrictions. They also reduce the cost of the corresponding hands-on laboratories because they enable the sharing of hardware between institutions. Nowadays, RLs are developed in many cognitive fields of science and engineering. In the literature have been proposed RLs on digital electronics, digital signal processing etc. [16-20].

The interactive RL on basic computer architecture which is proposed in this paper allows the students to perform remote experiments with one of the two available 8-bit micro-computers. The corresponding hardware implementation is automatically downloaded to the FPGA of the Altera DE2 board. The students are also able to program in assembly language the selected micro-computer and to remote control the DE2 board in order to verify their program. In addition, they are able to understand the internal micro-operations of the selected CPU by observing the values of its internal registers and buses during the step by step program execution.

The available micro-computers which are based on the RS and the Enhanced Relatively Simple (ERS) CPUs respectively, were implemented in Very High Speed Integrated Circuit HDL (VHDL) and Verilog using the Quartus II Integrated Development Environment (IDE) of Altera. The ERS CPU which is based on the RS CPU integrates the interfacing of isolated Inputs/Outputs (I/Os) and an interrupt handling mechanism. The simple architecture of the above CPUs which are ideal for introductory courses on computer architecture and the hardware implementations of the corresponding micro-computers are also discussed.

This paper is organized as follows: the architecture of the ERS CPU is presented in Section II while the hardware implementations of the two micro-computers are discussed in Section III. Finally, the development and operation of the proposed RL are presented in Section IV.

II. THE ARCHITECTURE OF THE ERS CPU

In this section, the architecture of the ERS CPU which is based on the RS CPU, is described [1, 21]. The ERS CPU is an 8-bit processor with 64 KBytes address space. It accesses the memory and the I/O devices through a 16-bit address and an 8-bit data buses.

The ERS CPU is designed as a multi-cycle processor and supports an Instruction Set Architecture (ISA) with 36 instructions. For the instruction encoding, a 6-bit operation code (opcode) is required but the ERS CPU utilizes an 8-bit opcode which allows its future expansion. The instruction sets of both the RS and the ERS CPUs are presented in Table I.

The ERS CPU uses the load/store architecture and never operates directly with memory operands. The data for an operation must be loaded into registers. Following the operation, the result is stored back into the memory.

Some instructions such as the LDAC, STAC, JUMP, JPNZ, LDSP and CALL require as argument a 16-bit memory address which is represented in Table I with the symbol "Γ". The memory is organized into bytes and due to this the above instructions require three bytes of memory. The first byte is the instruction's opcode and the last two bytes indicate the 16-bit memory address. According to the little endian format which is utilized by both CPUs, the second byte represents the least significant byte and the third byte the most significant byte of the required 16-bit memory address.

The INPT and OTPT instructions handle the isolated I/O devices and require an 8-bit argument which indicates the number of the corresponding I/O port. The above instructions require two bytes of memory. The first byte is the instruction's opcode and the second byte indicates the number of the I/O port.

The ISA of the ERS CPU includes six registers which are controlled directly by the programmer. The ACcumulator (AC) is an 8-bit register and receives the result of any arithmetic or logical operation. It also provides one of the operands for the arithmetic and logical instructions which require two operands. The 8-bit general purpose registers, R and B, supply the second operand of all two-operand arithmetic and logical instructions. They are also used to store data temporarily. The 3-bit FLag Register (FLR) contains the Zero (Z), Parity (P) and Sign (S) flags and indicates if the result of an arithmetic or logical instruction is zero, even or negative respectively.

TABLE I.
THE INSTRUCTION SETS OF THE RS AND ERS CPUS

Instruction	Code (Hex)	Operation
NOP	00	No Operation
LDAC	01 Γ	AC ← M[Γ]
STAC	02 Γ	M[Γ] ← AC
MVAC	03	R ← AC
MOVR	04	AC ←R
JUMP	05 Γ	GOTO[Γ]
JMPZ	06 Γ	IF Z=1 THEN GOTO[Γ]
JPNZ	07 Γ	IF Z=0 THEN GOTO[Γ]
ADD	08	AC ← AC+R
SUB	09	AC ← AC−R
INAC	0A	AC ← AC+1
CLAC	0B	AC ← 0
AND	0C	AC ← AC∧R
OR	0D	AC ← AC∨R
XOR	0E	AC ← AC⊕R
NOT	0F	AC ← AC′
MVBAC	1A	B ← AC
MOVB	1B	AC ← B
ADDB	18	AC ← AC+B
SUBB	19	AC ← AC–B
ANDB	1C	AC ← AC∧B
ORB	1D	AC ← AC∨B
XORB	1E	AC ← AC⊕B
LDSP	80 Γ	SP ← [Γ]
CALL	82 Γ	SP ← SP-2, M[SP] ← PC THEN COTO [Γ]
RET	83	PC ← M[SP], SP ← SP+2
PUSHAC	84	SP ← SP-1, M[SP] ← AC
POPAC	85	AC ← M[SP], SP ← SP+1
PUSHR	86	SP ← SP-1, M[SP] ← R
POPR	87	R ← M[SP], SP ← SP+1
INPT	20 Port	AC ← Input Port
OTPT	21 Port	Out Port ← AC
IESET	40	IE ← 1
IERST	41	IE ← 0
IPRST	42	IP ← 0
VIPRST	43	VIP ← 0

Relatively Simple (RS) CPU

Enhanced Relatively Simple (ERS) CPU

The Stack Pointer (SP) which is a 16-bit register holds the memory address of the current top of the stack. The ERS CPU utilizes the SP and the stack in order to handle a subroutine call. This process is achieved using the instructions CALL and RET, the operation of which is given in Table I.

The ERS CPU integrates also an interrupt handling mechanism which supports both Vectored Interrupt Requests (VIRQs) and non-vectored Interrupt Requests (IRQs). The address of the Interrupt Service Routine (ISR) of both interrupt types is generated by the Address Generation (AG) unit. In the case of the VIRQ, the address of the ISR is defined by default to 0x0042 Hex. In the case of the IRQs, the address of the corresponding ISR is calculated dynamically by the AG unit utilizing the 4-bit interrupt vector. This interrupt vector is unique for every external device and it is supplied to the AG when an IRQ is occurred. The micro-operations which are performed by the CPU in order to serve both VIRQs and IRQs are presented in Table II. The control of the interrupt handling mechanism is achieved using the 3-bit Interrupt Status Register which contains three flags, the Enable Interrupts (EI), the Interrupt Pending (IP) and the Vector

TABLE II.
THE MICRO-OPERATIONS OF THE INTERRUPT HANDLING MECHANISM OF THE ERS CPU

Type	Phase	Micro-Operations
IRQ	IRQ 1	AR ← SP
	IRQ 2	DR ← PC[15..8], SP ← SP - 1
	IRQ 3	M ← DR, AR ← AR - 1, SP ← SP – 1
	IRQ 4	DR ← PC[7..0]
	IRQ 5	M ← DR
	IRQ 6	DR ← int. vector
	IRQ 7	PC ← 00 & int. vector & 11
VIRQ	VIRQ 1	AR ← SP
	VIRQ 2	DR ← PC[15..8], SP ← SP – 1
	VIRQ 3	M ← DR, AR ← AR - 1, SP ← SP – 1
	VIRQ 4	DR ← PC[7..0]
	VIRQ 5	M ← DR
	VIRQ 6	PC ← 0x42 (Hex)

Interrupt Pending (VIP). The programmer enables or disables the IRQs using the instructions IESET and IERST, respectively. She/he is also able to clear the interrupt pending flag of both IRQs and VIRQs through the instructions IPRST and VIPRST, respectively.

Several registers which are not part of the ISA are fairly standard and are included in many CPU architectures. The CPU using these registers is able to perform the necessary internal operations such as fetch, decode, and execute. In this sense the ERS CPU contains the following registers:

- The 16-bit Address Register (AR) which supplies an address to memory or I/O port utilizing the address bus pins A[15..0] and A[7..0], respectively.
- The 16-bit Program Counter (PC) which contains the address of the next instruction that will be executed or the address of the next instruction's operand.
- The 8-bit Data Register (DR) which temporarily stores data being transferred to or from the memory or I/O port using the data bus pins D[7..0].
- The 8-bit Instruction Register (IR) which stores the opcode of the currently fetched instruction.
- The 8-bit temporary registers TR and TR2 which store temporarily data during the instruction execution phase.

According to the multi-cycle implementation, each instruction is broken down into multiple steps, each step designed to take one clock cycle. The first three steps are identical for all instructions and contain a set of micro-operations which deploy the fetch and decode phases. During these phases, the corresponding instruction is retrieved from the memory, utilizing the AR and the DR and the PC is increased by one. Subsequently, the opcode of the retrieved instruction is stored into the IR (decode phase) and the value of the PC is transferred to the AR in order to start the execution phase of the instruction. The execution phase of the retrieved instruction contains a set of micro-operations which are taking place in one or more cycle steps. The micro-operations of each one of the 36 instructions are analytically discussed in [21].

The hardware design of this ERS CPU can be broken down into two main components. The first component is the datapath and the second one is the control unit.

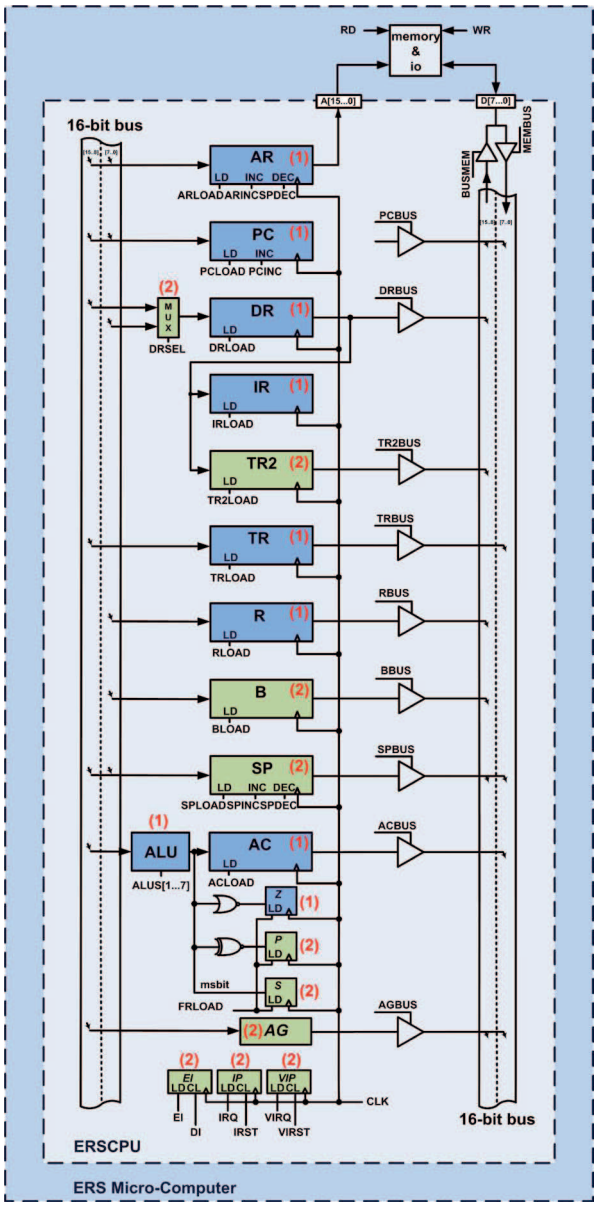

Figure 1. The ERS CPU datapath: (1) common elements in both RS and ERS CPUs, (2) elements included only in ERS CPU.

The block diagram of the ERS CPU datapath which is presented in Fig.1, contains all the individual modules (Arithmetic Logic Unit – ALU, registers, multiplexers, etc.) as well as the interconnections among them. These modules are able to be used more than once, on different clock cycles, during the fetch, decode and execute phases of each instruction. The registers within the processor's datapath are connected via a 16-bit internal bus. In addition, the internal simultaneous transfers of data are supported using direct connections between the CPU components.

The control unit acts as a finite state machine and generates the control signals of the datapath. Due to the rather limited number of instructions, the design of the ERS CPU's control unit either as microprogrammed or as hardwired is a rather affordable educational project. A more detailed description of both the ERS CPU architecture and its VHDL implementation is given in [21].

III. THE MICRO-COMPUTER BASED HARDWARE IMPLEMENTED SYSTEMS

The hardware implementation of the two micro-computers which are based on the RS or the ERS CPUs respectively, is achieved with VHDL and Verilog using the Quartus II IDE and the DE2 board of Atlera.

The DE2 board is one of the most widely accepted development FPGA boards and allows the implementation of a broad range of digital circuits [12, 13, 19, 21, 22]. It is based on the Altera Cyclone II EP2C35 FPGA which includes 33216 logic elements, 105 Random Access Memory (RAM) blocks of 4 Kbits, 35 embedded multipliers of 18-bits and 4 Phase-Locked Loops (PLLs). The DE2 board is also equipped with a series of peripheral devices such as toggle switches, push buttons, seven-segment displays, LEDs, a Liquid Crystal Display (LCD), a serial port, a USB port etc. The block diagram of this board which also indicates the interconnections between the FPGA and the peripheral devices is given in Fig. 2.

The main purpose of the implemented micro-computers is to allow the students to understand the micro-operations of the corresponding CPU by observing the values of its internal registers and buses. Consequently, the hardware implemented systems support the display of these values in hexadecimal format on a computer monitor utilizing the Video Graphic Array (VGA) port of the DE2 board [23]. The advantage of these systems is that they allow the control of the corresponding micro-computer locally or remotely according to the selected operational mode. The default operational mode is the local and these systems switch into the remote operational mode when they receive the appropriate command by the host computer.

In the local operational mode, the students are able to control the program execution utilizing the clock and reset signals which are connected to the push buttons KEY0 and KEY1 of the DE2 board. They are also able to control the 8-bit isolated input using the toggle switches 0-7 and observe the 8-bit isolated output through the red LEDs 0-7 of the DE2 board. An IRQ signal is generated every time the value of the 8-bit isolated input is changed.

In the remote operational mode, the above systems are controlled by a host computer utilizing the USB Blaster interface of the DE2 board. In this mode the hardware implemented system receives the appropriate commands from the host computer and generates the clock and reset signals for the control of the corresponding micro-computer. The data of the 8-bit isolated input are also sent to the implemented system by the host computer. In addition, the values of the internal registers, buses and the 8-bit isolated output are also returned to the host computer in every clock cycle.

A detailed block diagram of these two systems is presented in Fig. 3. The common elements of both systems are indicated with the index (1) and the additional elements of the implementation which is based on the ERS CPU are indicated with the index (2). The presented implementations besides the corresponding CPU and the Memory contain also a VGA Video Generation unit and an In-System Probes and Sources unit [24]. They also contain two multiplexers which undertake to select the clock and reset inputs of the corresponding micro-computer, according to the selected operational mode. The micro-computer based on the ERS CPU is also equipped with an I/O Port. The 8-bit input of the isolated I/O Port is selected by an additional multiplexer. All the above components were implemented using both the library of Quartus II megafunctions and generic HDL models. The megafunctions are vendor specific Intellectual Property (IP) blocks that are parameterizable and optimized for Altera device architecture. The megafunctions library which includes the Library of Parameterized Modules (LPM) and other parameterized functions, offers more efficient logic synthesis and device implementation.

The Memory is a common element in both hardware implementations and is used to store user programs and data. It is declared as RAM utilizing the altsyncram component of the LPM library. The width of the data port is set to 8-bits and the memory length is set to 4096 bytes. The Memory is connected to the correspo-nding CPU through the address, the data and the control buses. The altsyncram megafunction in combination with the In-System Memory Content Editor of Quartus II allows users to view and update the memory contents during run time through the USB Blaster interface [24].

The I/O port is an 8-bit isolated I/O interface which is designed as generic HDL model and is available only in the micro-computer based on the ERS CPU. It is connected to the ERS CPU through the address, data and control buses. The I/O port generates an IRQ signal every time the value of the 8-bit input is changed. This feature allows the students to understand the micro-operations of the ERS CPU during the execution of an interrupt driven program. The 8-bit isolated input is connected to a multiplexer. In local operational mode, the toggle switches 0-7 of DE2 board are connected to the 8-bit isolated input. In the

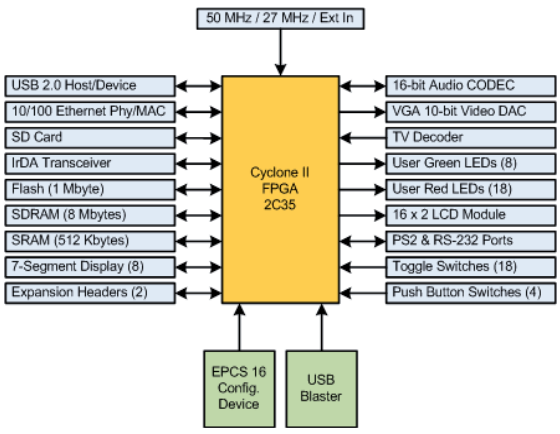

Figure 2. The block diagram of the DE2 board.

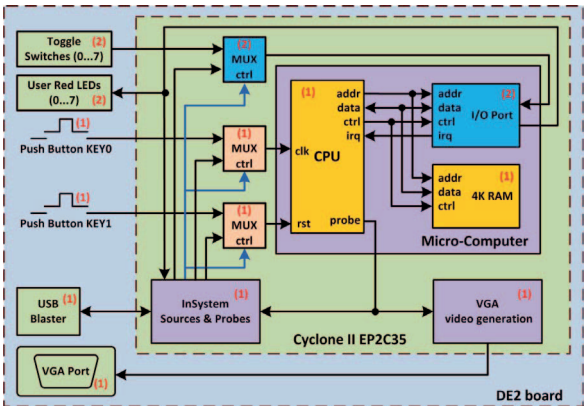

Figure 3. The block diagram of the implemented systems: (1) common components, (2) additional components of the ERS CPU based system

remote operational mode, the 8-bit isolated input is driven by the host computer through the In-System Sources and Probes unit. The data of the 8-bit isolated output which is connected to the red LEDs 0-7 of DE2 board, are sent to the host computer through the In-System Sources and Probes unit.

The hardware implemented systems utilizing the VGA Video Generation unit are able to display on a computer monitor in both local and remote operational modes the values of their internal registers and buses. The resolution of the display area is defined to 640x480 pixels and it is divided to thirty lines of forty characters per line. The representation of each character which is based on a fixed pixel pattern requires a block of 16x16 pixels. The display area is also divided in two sections with two columns per section. The left column of each section displays the registers and buses names and the right column their values. Since left column never changes, the data are retrieved from an internal Read Only Memory (ROM) which is initialized through a memory initialization file (".mif"). The values of the right column are produced dynamically in each clock cycle of the corresponding CPU by the VGA Video Generation unit. The hardware implemented system which is based on the RS CPU utilizes only one section of the display area as it is presented in Fig. 4a. In the case of the system which is based on the ERS CPU, the representation of internal registers and buses requires both sections of the display area (Fig 4b).

The In-System Sources and Probes unit is a hardware component which enables the remote control of the implemented systems through the USB Blaster interface. This unit utilizing altsource_probe megafunction of the LPM Library provides source and probe ports which act as inputs and outputs of the implemented micro-computer, respectively. In remote operational mode the In-System Sources and Probes unit initially generates the selection signal of multiplexers. Subsequently, it catches the virtual push buttons events from the host computer and triggers the clock and reset signals of the corresponding micro-computer. In case of the system which is based on the ERS CPU, the In-System Sources and Probes unit also generates the data of the 8-bit isolated input according to the virtual toggle switches event. In addition, using the probe port of this unit the values of the internal registers and buses are returned to the host computer. The data of the 8-bit isolated output which drives the red LEDs 0-7 of the DE2 board are also transferred to the host computer through the corresponding probe port. The In-System Sources and Probes Editor of Quartus II can be used to shift data to and from the source and probe ports of the design through the USB Blaster interface [24]. Consequently, the control of the presented hardware implemented systems and their feedback are available to the students locally or remotely through a host computer.

(a) (b)

Figure 4. The video representation of internal registers and buses of (a) the RS CPU and (b) the ERS CPU.

IV. THE RL ON COMPUTER ARCHITECTURE

This section presents the architecture and the operation of the proposed RL on computer architecture. In addition, the verification process of an example code from the student's point of view is also given in order to demonstrate the features of the proposed RL. The results of an early evaluation of this RL are also discussed.

A. The RL Architecture

According to the RL structure which is presented in Fig. 5, the RL Server consists of two components, the LabVIEW Server and the Tool Command Language (TCL) Server. The communication between the two components is achieved through Transmission Control Protocol/Internet Protocol (TCP/IP) messages.

The LabVIEW server allows the students to remotely program, control and verify the hardware implemented system based on the selected CPU through a user friendly GUI. This GUI is accessible to students through a common web browser utilizing the features of the integrated LabVIEW web server and the remote panel technology of National Instruments. The LabVIEW server also supports the syntax error check of student's assembly programs and the translation of these programs in machine code using the integrated cross-assembler [6]. Following the successful completion of the syntax error check and the assemble process, the LabVIEW server produces a ".mif" file which contains the generated machine code.

The TCL server is a socket server based on the TCL Application Program Interface (API) of Quartus II. It acts as an intermediate tier between the LabVIEW server and the DE2 board. The TCL server handles the Quartus II Programmer in order to program the FPGA of DE2 board by downloading the corresponding static random access memory object file (".sof"). It also updates the memory

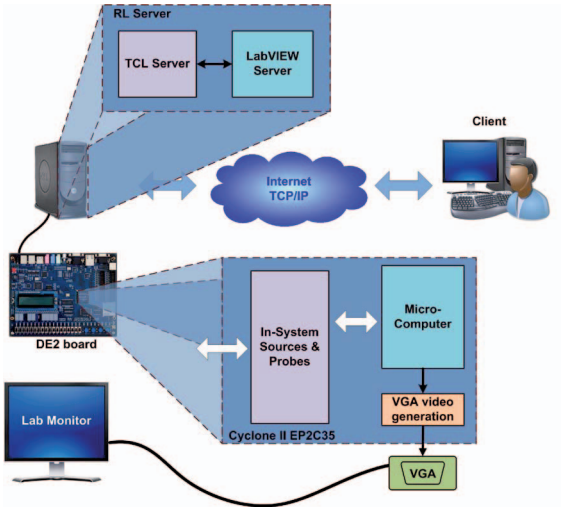

Figure 5. The structure of the RL on computer architecture.

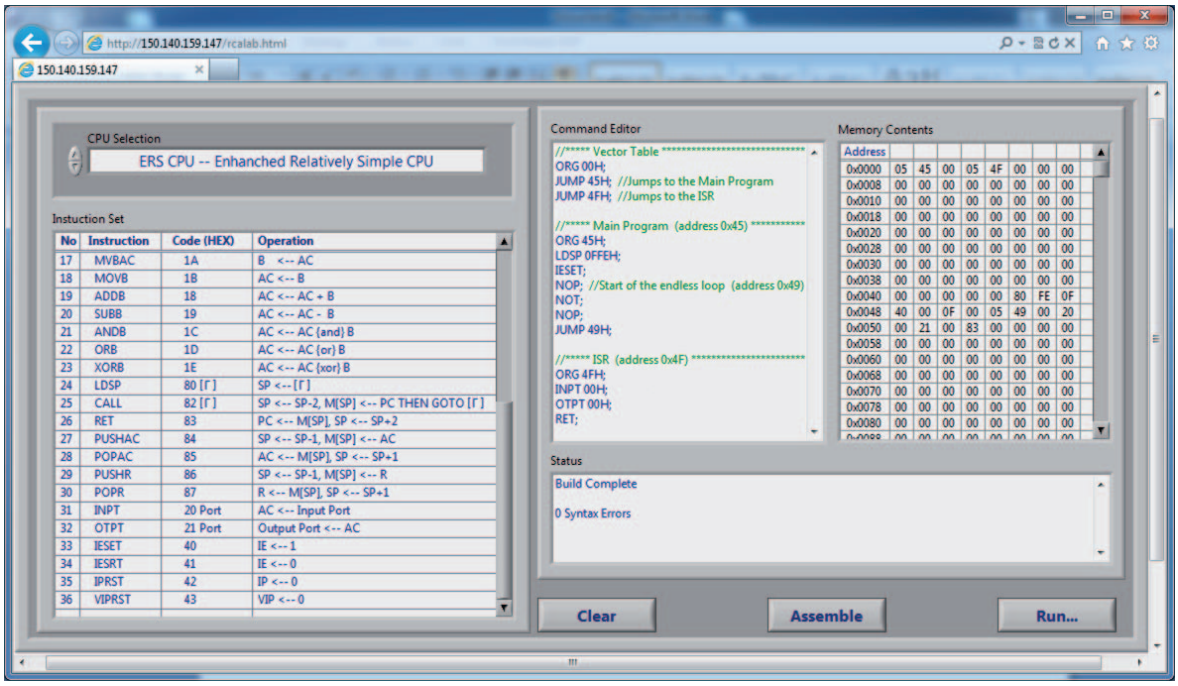

Figure 6. The web page of the Programming Environment.

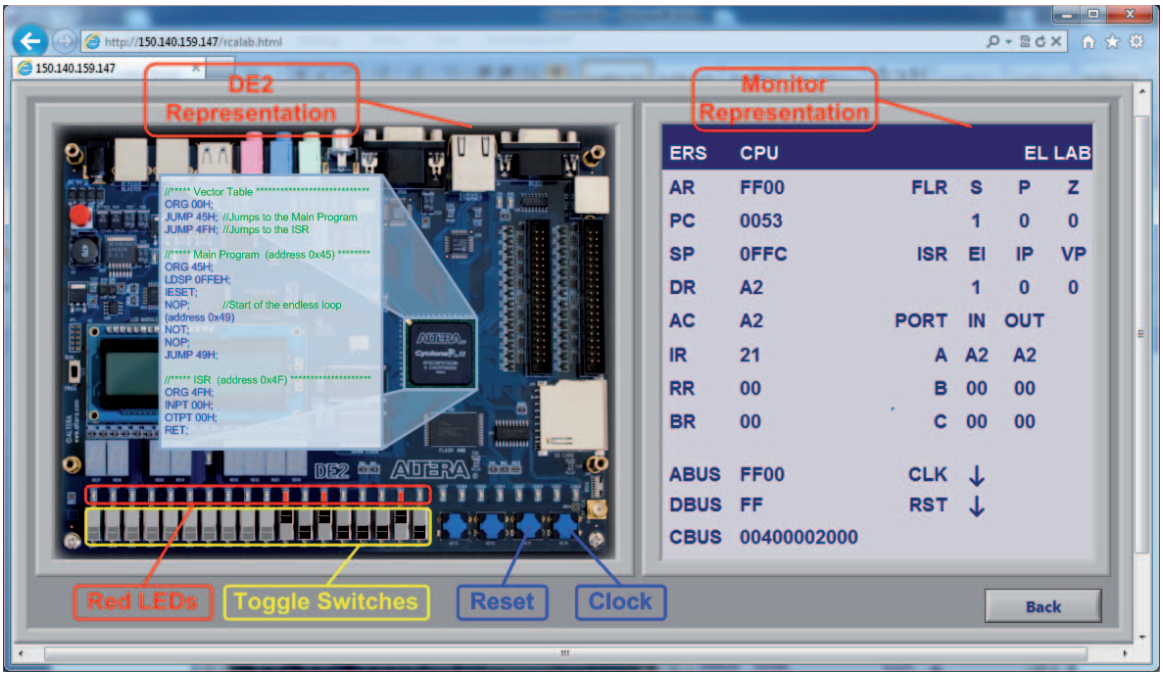

Figure 7. The web page of the Control Environment.

content of the selected micro-computer using a ".mif" file which is generated by the cross-assembler of the LabVIEW server. In addition, utilizing the features of both the In-System Probes and Sources unit of the hardware implementation and the In-System Probes and Sources Editor of Quartus II, the TCL server handles the data of 8-bit isolated input, the clock and the reset signals. It also reads the data of the internal CPU registers, buses and 8-bit isolated output.

B. The RL Operation

The student, who performs remote experiments with the proposed RL, initially selects the desired CPU architecture through the field CPU Selection of the Programming Environment's web page (Fig. 6). The field Instruction Set displays the instructions which are supported by the selected CPU. Initially, the student should write her/his assembly code using the Command Editor of the presented RL.

Subsequently, the student enables the syntax error check procedure by pressing the button Assemble and the possible errors are displayed in the field Status. If the student's program has no syntax errors, the integrated cross-assembler undertakes the translation from assembly to the machine language and generates a ".mif" file which contains the executable code. In the field Memory

Contents the student is able to observe in hexadecimal format the generated code which will be stored in the micro-computer's memory.

Following a successful assembly procedure, the student presses the button Run and enables the download procedure. The selected hardware implemented system is automatically downloaded to the FPGA. Subsequently, the generated executable code is also downloaded to the micro-computer's memory and the student is redirected to the Control Environment's web page (Fig. 7). Through this webpage, she/he remotely controls the DE2 board and the selected micro-computer in order to verify the operation of the corresponding CPU.

C. The Verification Process of an Interrupt Driven Example Code

The verification process of an interrupt driven example code from the student's point of view is presented in this paragraph for demonstration purposes. The example code which is written in assembly language is presented in Fig. 6. This code allows the student to understand the mechanism of the non-vectored interrupts and how the ERS CPU handles the 8-bit isolated I/O device which is referred as Port A. The presented code is divided into three parts, the vector table, the main program and the ISR. The simple vector table defines the addresses of the main program and of the ISR of the supported IRQ. The main program initializes the SP with the instruction LDSP and enables the non-vectored interrupts using the instruction IESET. It also implements a simple endless loop which executes the instructions NOP, NOT and NOP. In addition, the example code contains the ISR of the supported IRQ. The ISR which is executed when the IRQ is occurred, reads the data of the 8-bit isolated input of Port A using the instruction INPT. Subsequently, it writes this data to the 8-bit isolated output of Port A using the instruction OTPT. The 8-bit isolated I/O of port A is associated with the first eight virtual toggle switches and the first eight red LED of the DE2 representation, respectively.

During the verification process of the above example code the student is redirected to the Control Environment web page (Fig. 7). At this point, the student starts the step by step execution of her/his code by pressing the push button Clock. Every time she/he presses the push button Clock a single clock pulse is generated by the In-System Sources and Probes unit of the selected hardware implemented system and the current values of the internal CPU registers and buses are displayed, in hexadecimal format, in the GUI's monitor representation. Through this process which is very important, the student understands the internal CPU micro-operations which are taking place during the fetch, decode and execute phases of each instruction. One major advantage of the proposed RL is that the student is able to understand the interrupt handling mechanism of the ERS CPU. When the student changes the value of one of the first eight virtual toggle switches the isolated I/O device (Port A) generates an IRQ signal and the value of the IP changes to logical '1'. The student by pressing the push button Clock observes the step by step execution of the internal micro-operations which are taken place in order to be served the occurred IRQ by the ERS CPU. She/he initially observes how the ERS CPU pushes the current value of the PC to the stack. Subsequently, she/he understands the generation of the ISR address and how the ERS CPU calls the ISR of the corresponding interrupt. Following the ISR completion the student learns how the ERS CPU retrieves the return address from the stack and resumes the execution of the main program. During the execution of the above ISR, the student is able to understand the step by step operation of the handing mechanism of an 8-bit isolated I/O device. The student is also able to reset the selected CPU by pressing the push button Reset. In this case, the selected CPU resets its internal registers. In the following clock cycle the CPU starts over the program execution.

D. The Students' Evaluation

At this time the proposed RL is in the pilot phase and is going to be integrated into the course "Computer Architecture and Digital System Design with VHDL" in the next academic year. An early evaluation of the proposed RL was already voluntarily held, in the winter semester of the academic year 2012-2013, by ten postgraduate students.

Most of the students were enthusiastic with this RL not only because it has a user-friendly GUI but also because they are able to write assembly programs that can be downloaded onto and executed by the selected hardware implemented system. They are also convinced that this RL provide the students in one hand with a valuable experience in assembly programming language and on the other hand with a deep understanding of the CPU's program execution mechanism. As they commented, this process is very important especially for students with low background knowledge in computer architecture. In addition, the students mentioned that remote experiments which utilize the isolated I/Os and the interrupt handling mechanism of the ERS CPU allow them to understand how a simple CPU serves an IRQ from an external device. They also confirmed that the use of the proposed RL is more interesting and suitable for their education than the corresponding simulators because they are able to interact with a real system. Finally, the students suggested the extension of the proposed RL features in order to support the verification of hardware designs which were implemented by them.

V. CONCLUSIONS AND FUTURE WORK

This paper proposes an interactive RL which allows the students to understand basic theoretical concepts on computer architecture and organization using two rather simple micro-computers. These micro-computers are based on the RS and the ERS CPUs which are also presented in this paper. The simplicity of their architectures renders these CPUs ideal for educational purposes. The ERS CPU is a super set of the RS CPU and due to this, it has an extended ISA. The main advantage of the ERS CPU is the integration of a subroutine calling mechanism which is implemented by exploiting the stack. It also supports the handling of both vectored and non-vectored interrupt requests and an interface of 8-bit isolated I/Os. The presented hardware implemented systems which are based on the above CPUs were designed in both VHDL and Verilog using the Quartus II.

The students through the programming environment web page of the proposed RL select and program in assembly one of the two available micro-computers which are based on the above CPUs. The hardware implementation of the selected micro-computer and the machine code which is generated by the integrated cross-assembler

are downloaded to the FPGA of the DE2 board. During the verification process the students following the step by step execution of the downloaded program, are able to observe the values of the CPU's internal registers and buses using the monitor representation of the RL Control Environment. The required clock and reset signals are produced by pressing the corresponding virtual push buttons. The students through the web page of the proposed RL are also able to test their programs using the virtual toggle switches and LEDs of DE2 board representation.

The integration of new features on the proposed RL which will support the remote control of additional DE2 board peripheral devices, as for example the seven segment displays or the LCD display, belongs to the authors' future plans. These features will increase the level of the interactivity of the proposed RL. In addition, there are plans for the development of remote experiments which are based on other CPU architectures such as the Microprocessor without Interlocked Pipeline Stages (MIPS).

REFERENCES

[1] J. D. Carpinelli, *Computer Systems Organization & Architecture.* Addison Wesley, 2001.

[2] D. A. Patterson and J. L. Hennessy, *Computer Organization and Design: The Hardware/Software Interface*, 4th ed. Morgan Kaufmann Publishers, 2009.

[3] B. Nikolic, Z. Radivojevic, J. Djordjevic, and V. Milutinovic, "A Survey and Evaluation of Simulators Suitable for Teaching Courses in Computer Architecture and Organization", *IEEE Transactions on Education*, vol. 52, no. 4, pp. 449-458, November 2009. http://dx.doi.org/10.1109/TE.2008.930097

[4] M. A. Lusco and C. E. Stroud, "PSIM: A Processor SIMulator for Basic Computer Architecture and Operation Education", *IEEE SoutheastCon 2010 (SoutheastCon)*, Charlotte-Concord, North Carolina pp. 115-118, 18-21 March 2010.

[5] R. R. Peralta Meza, C. A. Vera Bernal, F. J. Guerra Manchego and P. T. Llerena Valdivia, "A CPU Simulator based on FPGA Soft Processor", *9th Latin American and Caribbean Conference for Engineering and Technology (LACCEI)*, Medellin, Colombia, pp. WE1-1-10, 3-5 August 2011.

[6] C. Katsenos, "*Flexible Web-Based Processor Simulator for Learning and Teaching Basic Computer Architecture*", MSc Thesis, Electronics Laboratory, Physics Department, Patras University, Greece, 2012, http://hdl.handle.net/10889/5402, (in Greek).

[7] J. D. Carpinelli and F. Jaramillo, "Simulation Tools for Digital Design and Computer Organization and Architecture", *31st ASEE/IEEE Frontiers in Education Conference (FIE)*, vol. 3, pp. S3C-1-5, Reno, Nevada, 10-13 October. 2001.

[8] L. Null and J. Lobur, *The Essentials of Computer Organization and Architecture*. Jones and Bartlett Publishers, 3rd ed., 2012.

[9] B. Hatfield, M. Rieker and L. Jin, "Incorporating Simulation and Implementation into Teaching Computer Organization and Architecture", *35th ASEE/IEEE Frontiers in Education Conference (FIE)*, pp. F1G-18-23, Indianapolis, Indiana, 19-22, October 2005.

[10] J. H. Lee, S. E. Lee, H. C. Yu and T. Suh, "Pipelined CPU Design with FPGA in Teaching Computer Architecture", *IEEE Transactions on Education*, vol. 55, no. 3, pp. 341-348, August 2012. http://dx.doi.org/10.1109/TE.2011.2175227

[11] A. Elkateed, "A Processor Design Course Project: Creating Soft-Core MIPS Processor Using Step-by-Step Components' Integration Approach", *International Journal of Information and Education Technology*, vol. 1, no. 5, December 2011.

[12] P. Bulic, V. Gustin, D. Sonc and A. Strancar, "An FPGA-Based Integrated Environment for Computer Architecture", *Computer Applications in Engineering Education*, vol. 21, no. 1, pp. 26-35, March 2013. http://dx.doi.org/10.1002/cae.20448

[13] O. Vainio, E. Salminen and J. Takala, "Teaching Digital Systems Using a Unified FPGA Platform", *12th Biennial Baltic Electronics Conference (BEC)*, pp. 137-140, Tallin, Estonia, 4-6 October 2010. http://dx.doi.org/10.1109/BEC.2010.5629729

[14] H. Oztekin, F. Temurtas and A. Gulbag, "BZK.SAU.FPGA10.1: A Modular Approach to FPGA-Based Micro Computer Architecture Design for Educational Purpose", *Computer Applications in Engineering Education* [online], doi:10.1002/cae.20553 http://dx.doi.org/10.1002/cae.20553

[15] L. Gomes and S. Bogosyan, "Current Trends in Remote Laboratories", *IEEE Transactions on Industrial Electronics*, vol. 56, no. 12, pp. 4744-4756, December 2009. http://dx.doi.org/10.1109/TIE.2009.2033293

[16] R. Hashemian and J. Riddley, "FPGA e-Lab, a Technique to Remote Access a Laboratory to Design and Test", *IEEE International Conference on Microelectronic Systems Education (MSE)*, pp. 139-140, San Diego, California, 3-4 June 2007.

[17] Y. Rajasekhar, W. V. Kritikos, A. G. Schmidt and R. Sass, "Teaching FPGA System Design via a Remote Laboratory Facility", *International Conference on Field Programmable Logic and Applications (FPL)*, pp. 687-690, Heidelberg, Germany, 8-10 September 2008.

[18] F. Morgan, S. Cawley, D. and Newell, "Remote FPGA Lab for Enhancing Learning of Digital Systems", *ACM Transactions on Reconfigurable Technology and Systems*, vol. 5, no. 3, pp. 18:1-13, October 2012.

[19] J. Lobo, "Interactive Demonstration of a Remote Reconfigurable Logic Laboratory for Basic Digital Design", *International Journal of Online Engineering (iJOE)*, vol. 8, special issue 1: exp.at'11, pp. 19-20, February 2012.

[20] A. Kalantzopoulos, D. Karageorgopoulos and E. Zigouris, "A LabVIEW Based Remote DSP Laboratory," *International Journal of Online Engineering (iJOE)*, vol. 4, special issue 1: REV 2008, pp.36-44, July 2008.

[21] E. Galetakis, "*Design and Implementation in VHDL of a Micro-Computer System Based on an Enhanced Relatively Simple CPU*", MSc Thesis, Electronics Laboratory, Physics Department, Patras University, Greece, 2012, http://hdl.handle.net/10889/5401, (in Greek).

[22] Altera, *DE2 Development and Education Board*, User Manual, version 1.42, Altera Corporation, 2008.

[23] J. O. Hamblen, T.S. Hall, and M. D. Furman, *Rapid Prototyping of Digital Systems, SOPC Edition.* Springer, 2008. http://dx.doi.org/10.1007/978-0-387-72671-7

[24] Altera, *Quartus II Handbook*, version 11.1, vol. 3: Verification, section IV: System Debugging Tools, November 2011.

AUTHORS

A. Kalantzopoulos is with the Electronics Laboratory, Electronics and Computers Div., Department of Physics, University of Patras, Rio Patras, GR-26500 (e-mail: kalan@upatras.gr).

E. Galetakis is with the Electronics Laboratory, Electronics and Computers Div., Department of Physics, University of Patras, Rio Patras, GR-26500 (e-mail: mgaletakis@gmail.com).

C. Katsenos is with the Electronics Laboratory, Electronics and Computers Div., Department of Physics, University of Patras, Rio Patras, GR-26500 (e-mail: xkatsenos@upatras.gr).

E. Zigouris is with the Electronics Laboratory, Electronics and Computers Div., Department of Physics, University of Patras, Rio Patras, GR-26500 (e-mail: ez@physics.upatras.gr).

Submitted, April, 12, 2013. Published as resubmitted by the authors on June, 7, 2013.