A Computer Architecture Educational System

Dimitris Mandalidis, Panagiotis Kenterlis, Panagiotis Drosinopoulos, John N. Ellinas
Dept. of Electronic Computer Systems, Technological Educational Institute of Piraeus, 122 44 Egaleo, Greece
E-mail: {D.Mandalidis, P.Kenterlis, pdros, J.Ellinas}@mprolab.teipir.gr

Abstract

This paper describes the implementation of a system-on-a-programmable-chip (SOPC) development board to support computer architecture laboratories at a low cost. A commercial field-programmable-gate-array (FPGA) was employed to develop our reduced-instruction-set-computer (RISC) soft processor core that may be programmed through a user-friendly environment accompanied by an assembler. Our approach aims to support a wide variety of student projects in our engineering curriculum, increase students productivity and decrease the development time. Through our implementation, students are introduced to RISC architecture concepts, SOPC design and structure of assemblers. The reusability of the hardware permits the materialization of future projects according to our educational needs. The proposed inexpensive solution is a complete educational environment suitable for undergraduate use.

1. Introduction

The undergraduate curriculum of our department consists of two courses in computer architecture, familiarizing students more familiar to microprocessor notions and operations, memory and port interfacing, interrupt logic, direct-memory-access (DMA) operation, assembly language programming, computer arithmetic and computer hardware and software in general. The first course is introductory to computer architecture involving assembly language programming of 8-bit microprocessors. The second course involves programming more high-end microprocessors like the Intel's 80X86 family. The laboratory courses are based on special purpose software simulators to aid in modeling the targeted microprocessor. Of course, a number of other courses like logic design and microelectronics assist students to integrate their knowledge in this field. Our intention is to make the systems hardware and software more appealing to students by creating meaningful real time laboratory applications. Field programmable devices contribute to this objective considerably as they are flexible and of low cost. Nowadays, FPGA designs are becoming increasingly popular due to their maintainability, their flexibility and their cost. Designers are able to re-program and reuse the same FPGA for many applications, while the embedded nature of a FPGA enables designers to implement large applications which otherwise would require large portions of discrete integrated circuits. Furthermore, the time between the conception of an idea and its implementation is decreased dramatically. On the other hand, application-specific-integratedcircuit (ASIC) design is generally faster, but its permanent nature is a significant drawback for student designs which require frequent changes. Taking advantage of the flexibility that a FPGA provides, several FPGA vendors have implemented processor

cores in such devices [1]; complete processors which support a wide variety of applications. That approach influenced universities which until now designed or used existing processor cores to support laboratories of their engineering curriculums. [2] Our approach involves designing a processor core based on the MIPS R2000, a 32-bit RISC processor which was originally introduced in 1984 by professor John Hennesy. We modified the original MIPS R2000 architecture [3] to adapt it to our needs and embedded several peripheral functions to enable students monitor their projects. Thus, together with our implemented assembler a complete development environment is formed, ideal for undergraduate students in computer architecture laboratories. In this paper, we will overview the commercial processor cores and the software tools for the development of SOPC applications, the proposed system based on a 32-bit RISC processor and how this system is adapted to our curriculum.

2. Overview

2.1. Hardware of SOPC

Nowadays, FPGA devices are no longer used only for prototyping and debugging purposes, but can also be found in commercial applications due to the low development cost, high level of logic available and their flexible in-system reprogramming capability. Therefore, many popular FPGA vendors have introduced commercial microprocessors which may be used in various designs. Such processors can be ranked in two categories; hard and soft processor cores. Hard processor cores involve a combination of an embedded processor core in dedicated silicon and FPGA logic, while soft processor cores are designed using FPGA logic elements exclusively. Both approaches have advantages and disadvantages some of which are presented in Table I. Notably, Altera [4] and Xilinx [5] have developed both hard and soft

Feature	Hard cores	Soft cores
Performance	+	-
Cost	-	+
Flexibility	-	+
Power consumption	+	-

Table 1. Pros and cons of processor cores

processor cores to support designers implementations. Altera is marketting a hard processor core, named Excalibur [6], in its APEX FPGA family [7]. Excalibur is a 32-bit RISC processor core based on ARM architecture providing clock frequencies up to 200MHz, an external memory bus and various I/O capabilities. In the field of soft processor cores, Altera Nios II [8] is a 32-bit RISC soft processor core providing clock frequencies around 150MHz depending on the FPGA product.

On the other hand, Xilinx supplies hard processor cores [9] based on IBM PowerPC architecture providing 32-bit RISC cores embedded in VirtexII-Pro and Virtex4-FX FPGA families that run at frequencies up to 450MHz. Furthermore, Xilinx is marketting Microblaze [10], a 32-bit RISC soft processor core, capable of running at frequencies around 150MHz.

Moreover, the above mentioned vendors as well as third-party companies offer development boards to implement a wide variety of design applications. Such boards are usually shipped

with a FPGA for general purpose use or with ready processor cores which can be modified according to the developer's needs. These boards also offer various monitoring and communication facilities such as liquid-crystal-display (LCD), light-emitting-diodes (LEDs), switches, USB, EIA-232 and joint-test-action-group (JTAG) interfaces. Our application was developed in a Memec [11] board which is shown in Figure 1.

2.2. Software tools

The development of a SOPC is achieved by employing specific software tools that may configure processor hardware options for each soft processor core. The design is integrated by combining the system logic needed with the processor core using a standard FPGA synthesis computer-aided-design (CAD) tool.
Vendors also offer fully functional CAD tools. Altera Quartus II [12] and Xilinx ISE [13] are tools intended for general use. In addition to this, Altera SOPC Builder [14] and Xilinx Embedded Development Kit (EDK) [15] help developpers design and evaluate embedded system solutions. Such tools, in conjuction with powerful simulation environment, like Mentor Graphics ModelSim [16], offer FPGA designers the opportunity to exploit FPGA technology capabilities over ASIC by building fully operational systems at no time.

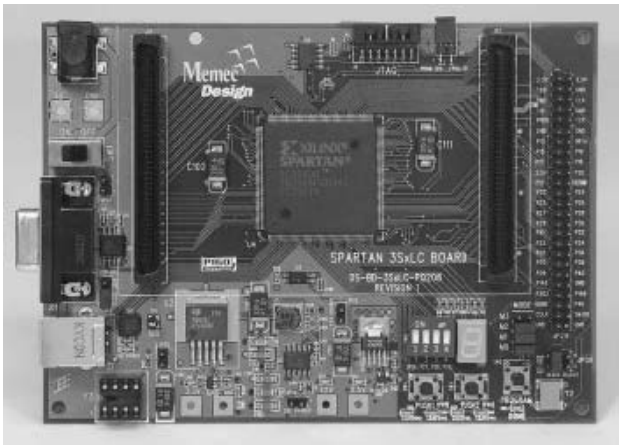

Figure. 1. Memec DS-KIT-3SLC400 development board

3. The Proposer 32-BIT RISC Processor

3.1. Processor core development

The proposed system was implemented on a Xilinx xc3s400 FPGA [17], a 400,000-gate member of the Spartan 3 family, shipped with Memec's DS-KIT-3SLC400 development board [18] as shown in Fig. 1. The board is offering EIA-232 and USB interfaces, JTAG and Platform Flash configuration support, System ACE connector and 109 user I/O pins which can be manipulated through an optional P160 expansion module. It also contains on-board a clock oscillator, a user clock socket, and voltage regulators at 3.3V, 2.5V and 1.2V. The software tools employed were VHDL under Xilinx ISE 6.2 environment and ModelSim 5.7g simulator in order to achieve the desired functionality for the resulting application.

3.2. Processor core assembler

An assembler to support our application was also developed under the Linux operating system due to the batch and remote facilities that the latter offers. In this case, the GNU Assembler (GAS) [19] could be modified to suit our needs. However, its difficult installation as a cross-assembler coupled with the fact that an extra level of indirection would be needed (due to the OS-specific symbols) led us to develop our own assembler. Consequently, our assembler was written in C from scratch using the GNU Compiler Collection (GCC) [19] and the GNU Debugger (GDB) [20]. Finally, it is not architectural dependent and its performance was tested under Solaris and PPC architectures.

3.3. The proposed system

1) Internal architecture: The implemented processor core contains various modifications on the original MIPS R2000 processor. More precisely, our core provides a solution to the memory alignment problem of the MIPS R2000, offers an interrupt controller and extra I/O capabilities. Furthermore, it lacks a floating point unit and pipelining as these concepts were considered of minor importance with respect to our curriculum. As far as performance is concerned, the resulting processor runs at frequencies up to 63MHz and the total estimated power consumption is 125mW. The xc3s400 utilization is shown in Table II where the space availability for further improvements and add-ons is indicated.

2) Memory and I/O: The most important modification concerning memory architecture was the switching from Harvard to von Neumann architecture; contrary to Harvard architecture, von Neumann's devotes common signals and units for code or data memory access. This modification enabled the reuse of the universal-asynchronous-receiver-transmitter (UART) controller for both student programs and run-time programming of the processor core. The FPGA internal block RAM of 16KB proved sufficient for our needs

Part	**Total amount**	**Amount used**	**Usage (%)**
Slices	3584	1853	51
Slice Flip-Flops	7168	1041	14
4 input LUTs	7168	3140	43
Bonded IOBs	141	47	33
BRAMs	16	10	62
MULT18X18s	16	4	25
GCLKS	8	1	12

Table 2. FPGA Utilization

and finally a 512x32 portion of the FPGA internal ROM memory was used for firmware purposes. A wait state is introduced through a data-multiplexing-demultiplexing (DMD) unit to avoid access to out of bounds memory address. Thus, the alignment problem is solved as shown in Fig. 2 and 3 where a word write at address 0x00000005 is being attempted. A memory-mapped I/O unit was also implemented to take

Figure. 2. Initial access cycle

Figure. 3. Alignment correction cycle

advantage of our embedded peripherals as well as the development board facilities. It consists of a 32-bit timer, a UART controller running at 14400bps, and two 8-bit bidirectional I/O ports that may drive external devices such as LCD, keyboards, LED, DIP switches, 7-segment displays. Some of these peripheral devices are provided by the development board.

3) Interrupt-driven system: An embedded interrupt controller was also implemented to take full advantage of the I/O functionality instead of supplying an external interrupt device. The mps command was added to the original instruction set to support read or write operations of interrupt registers. The most important interrupt registers are the mask and vector-page registers. The first is used to enable or disable interrupts and the second specifies the address of an interrupt service routine. Furthermore, the interrupt logic serves byte transmission or reception through the UART controller, timer overflow and arithmetic-logic-unit (ALU) overflow. The interrupt controller consists of an internal stack which is capable of holding sixteen return addresses and processor status and therefore servicing sixteen nested interrupts, the four of which are internal requests while the others are external. Apart from the basic interrupt control registers there are also supplementary registers shown in Table III.

Interrupt register	Access method
Masked flags	Read
Unmasked flags	Read
Mask	Read/Write
Vector address	Read
Vector page	Write
Return address	Read

Table 3. Interrupt registers

4) Timer module: The implemented timer module consists of an incremental 32-bit counter which provides up to 1:8 prescaling through a 4-to-1 multiplexer. A multiple-frequency divider is used as input to the prescaler that provides various timing periods according to the values of the respective controlling bits. The functionality of the timer is controlled by two registers. One of them holds the time constant and the other controls other options. The timer may provide delays of up to 45 minutes using a 50MHz clock.

5) UART module: Our UART controller provides an asynchronous full-duplex communications path with a variety of data-terminal-equipment (DTE) or data-connecting-equipment (DCE) devices. It runs at 14400bps and its data frame consists of one start-bit, eight data bits, one stop bit and without parity functionality (8N1). A double buffer is also implemented to prevent bytes overlapping in transmission process. Its functionality is controlled by three registers; two for transmitting or receiving data and one for enabling or disabling the transmission or the reception.

6) Assembler: The assembler developed for our application is a two pass assembler which provides the user with listing information (see Table IV) along with a HEX file used to load the executable program. During the first pass, the assembler tracks down labels with their respective data or address, and afterwards, this performs the second pass using that information. It is also batch-capable allowing performance of other tasks while running in the background. It should also be noted that it is released under the GNU General Public Licence (GPL) [21].

Memory address	Object code	Command
0x00000400	0x34014000	ori $01, $0, 0x4000
0x00000404	0x0c000116	jal 0x116
0x00000408	0x34030038	ori $03, $0, 0x38
0x0000040c	0xa0230004	sb $03, 0x4($01)
0x00000410	0x08000104	j 0x104
0x00000444	0x8024000c	lb $04, 0xc($01)
0x00000448	0xa0240010	sb $04, 0x10($01)
0x0000044c	0xa0240008	sb $04, 0x8($01)
0x00000450	0x0c000116	jal 0x116
0x00000454	0xd4000000	reti
0x00000458	0x3c020001	lui $02, 0x1
0x0000045c	0x34420002	ori $02, $02, 0x2
0x00000460	0xd0021803	mps $03, $0, $02, 3
0x00000464	0x34020444	ori $02, $0, 0x444
0x00000468	0xd0021805	mps $03, $0, $02, 5
0x0000046c	0x03e00008	jr $ra

Table 4. Assembler listing

7) Supplementary commands: A few extra commands were added to the original MIPS R2000 instruction set; apart from **mps** which is used for interrupt control, **reti** command was added to support return from an interrupt service routine, **push, pushd, pop** and **popd** were added to implement stack support.

4. Proposed SOPC In The Undergraduate Curriculum

The students of a computer architecture laboratory are provided with a bootable Linux CD. The CD includes an executable binary file of the assembler as well as extensive documentation of the development roadmap of the system. Then the user could use the assembler to save his work in a USB stick. Several laboratory exercises have been developped such as interfacing of LCD displays, 4x4 keyboard matrices and real-time-clock (RTC) chips. Our laboratory is currently developing more advanced exercises such as driving of dc and stepper motors, interfacing of various sensors (temperature, humidity, pressure, weighing, gas etc).

The students attending the computer architecture laboratory are going through a number of exercises that train them in a variety of issues concerning the FPGA implementation of a RISC processor, its architecture, peripherals, programming, functionality and interfacing. The enthusiasm of the students attending this course motivates our team to enhance the present work. Among others, extended UART capabilities could be implemented such as programmable baud rate, synchronous communication, parity and data width control. Timer facilities could also be extended to support pulse-width-modulation (PWM) and another timer could be implemented to extend maximum delay. Moreover, an external memory interface could be added to enable the design of memory-consuming projects like digital-signal-processing (DSP) applications. As far as the software part is concerned, assembler macro commands could be introduced and a C compiler could be implemented. Fortunately, due to the rapid evolution of FPGA technology, a more advanced development board along with a larger FPGA could help us implement almost everything imaginable.

5. Conclusion

This paper describes the hardware and software implementation of an educational system based on a RISC soft processor core that will support the computer architecture laboratory of our department. Overall, the approach of using FPGA for the support of our computer architecture laboratory proved to be cost-effective; the complexity of assigned essays was increased while the cost of the equipment was greatly decreased. Through inexpensive equipment we found out that the support of a whole laboratory was possible. Furthermore, the time needed for students to finish a report has decreased, and the opportunity for the students to extend the functionality of the system could also be provided through a dissertation. Finally, we found that our students enjoyed our approach and managed to better accomplish the objectives of a computer architecture laboratory.

REFERENCES

[1] R. Lysecky and F. Vahid, “A study of the speedups and competitiveness of fpga soft processor cores using dynamic hardware/software partitioning,” Design, Automation and Test in Europe (DATE’05), vol. 1, pp. 18–23, 2005.

[2] T. S. Hall and J. O. Hamblen, “System-on-a-programmable-chip development platforms in the classroom,” IEEE Trans. Educ., vol. 47, no. 4, pp. 502–507, Nov. 2004.

[3] G. Kane and J. Heinrich, MIPS RISC Architecture, 2nd ed. Prentice Hall PTR, 1991.

[4] Altera corp. [Online]. Available: http://www.altera.com

[5] Xilinx inc. [Online]. Available: http://www.xilinx.com

[6] Excalibur Devices Overview. [Online]. Available: http://www.altera.com/products/devices/arm/overview/arm-overview.html

[7] APEX 20K Devices: System-on-a-Programmable-Chip Solutions. [Online]. Available: http://www.altera.com/products/devices/apex/ apx-index.html

[8] Nios II Processor Cores. [Online]. Available: http://www.altera.com/products/ip/processors/nios2/cores/ni2-processor %cores.html

[9] PowerPC 405 Processor. [Online]. Available: http://www.xilinx.com/products/silicon solutions/fpgas/virtex/virtex ii% pro fpgas/ capabilities/powerpc.htm

[10] Microblaze Soft Processor Core. [Online]. Available: http://www.xilinx.com/xlnx/xebiz/designResources/ip product details.jsp% ?sGlobalNavPick=&sSecondaryNavPick=&category=-1212028&iLanguageID=1&key=micro %blaze

[11] Memec. [Online]. Available: http://www.memec.com

[12] Quartus II Software. [Online]. Available: http://www.altera.com/products/software/products/quartus2/qts-index.htm%l

[13] Xilinx ISE Foundation. [Online]. Available: http://www.xilinx.com/ise/logic design prod/foundation.htm

[14] SOPC Builder. [Online]. Available: http://www.altera.com/products/software/products/sopc/sop-index.html

[15] Platform Studio and the EDK. [Online]. Available: http://www.xilinx.com/ise/embedded design prod/platform studio.htm

[16] Modelsim SE. [Online]. Available: http://www.xilinx.com/ise/embedded design prod/platform studio.htm

[17] xc3s400 FPGA datasheet. [Online]. Available: http://www.xilinx.com/bvdocs/publications/ds099.pdf

[18] Memec Design Spartan-3 LC Development Kit. [Online]. Available: http://www.memec.com/uploaded/Spartan3LC 4.pdf

[19] GCC. [Online]. Available: http://www.gnu.org/software/gcc/

[20] GDB. [Online]. Available: http://www.gnu.org/software/gdb/

[21] GNU General Public Licence. [Online]. Available: http://www.gnu.org/licenses/gpl.html