A Multimedia Tool for Teaching Reconfigurable Computing

Iouliia Skliarova
Department of Electronics, Telecommunications and Informatics / IEETA
University of Aveiro
3810-193 Aveiro, Portugal
e-mail: iouliia@ua.pt

Abstract—**This paper proposes a method for efficient teaching of reconfigurable computing. Nowadays, reconfigurable systems, in general, and FPGA (Field-Programmable Gate Array) based systems, in particular, constitute an essential part of engineering practice. The paper argues importance of reconfigurable systems in education and suggests a multimedia tool augmented with an FPGA-based prototyping system, which could contribute to productive teaching of reconfigurable computing.**

Keywords- reconfigurable computing; computer architecture, FPGA, prototyping board, teaching

I. INTRODUCTION

In the recent years we witnessed a tremendous progress in the scope of reconfigurable systems. FPGA (Field-Programmable Gate Arrays) appeared on the market in 1985 [1] and now they are widely used in numerous applications. For example, FPGAs, because of their processing throughput and inherent reconfigurability, provide great value as processing platforms in the area of software-defined radios and cognitive radios [2]. FPGA have proven to be effective for algorithm acceleration in high-performance computing in the areas of biological sequencing, real-time video processing, and gravity simulation, where significant speed and power savings over traditional microprocessors and DSP were achieved [3-4]. Besides, FPGA have been shown to be very efficient in performing such functions as searching, sorting, coding/decoding, signal processing, audio/video/image manipulation, encryption, error correction, random number generation, packet processing, and many others. These functions are of great demand in applications such as seismic processing, acoustics, astrophysics FFT (Fast Fourier Transform), adaptive optics, cryptography, graphics acceleration, HDTV (High Definition Television), mobile radio, car multimedia systems, image recognition, speech recognition, security, video format translation, biotech applications, vehicular traffic simulation, financial modeling, orbit, space, and extra-terrestrial applications, and so on. Many of the mentioned tasks have to be performed in real-time.

A wide variety of industries relies on programmable logic to fuel quick innovation and product differentiation [5]. For instance, the automotive market uses a programmable platform to address different market requirements in automotive graphics systems [6]. The trend in the medical industry is to use programmable logic as a consistent hardware platform for a wide range of portable devices, simplifying recertification and preventing obsolescence issues [6]

Developing engineering systems on the basis of high capacity FPGAs involves vast variety of design tools and requires a large number of well-prepared engineers in the relevant areas [7]. Hence new trends must be appropriately reflected in the respective pedagogical activity and this strongly demands an ongoing review of university curricula. What happens in reality is that reconfigurable computing education does not exist at all at many universities even within electrical engineering and computer engineering curricula [8]. Therefore, an enforcement should be done to promote a positive image of reconfigurable systems by both illustrating the real achievements of reconfigurable computing and highlighting the links with other disciplines illuminating the advantages that such links could provide. We hope that these measures would make reconfigurable computing discipline more attractive to students.

In particular, we would like to find new and better ways of teaching that have greater visual appeal than the current methods and allow engaging more students in *active* learning. We have reported the results of our previous work in this direction in [7] where we presented animated tutorials, mini-projects, hardware templates, and course-oriented libraries that are successfully employed in University of Aveiro to assist the educational process. This paper continues and extends that work by proposing a multimedia tool and an FPGA-based prototyping board that could additionally contribute to effective teaching of reconfigurable systems.

The remainder of the paper is organized in four sections. Section II makes a revision of available design specification methods and discusses the future of reconfigurable systems. Section III provides a brief overview of courses dedicated to reconfigurable systems. In Section IV the proposed teaching method is described. The conclusion is given in Section V.

II. THE PRESENT AND THE FUTURE OF RECONFIGURABLE SYSTEMS

Designers of FPGA-based systems must wade through several layers of design before programming the actual device. The typical FPGA flow includes five major phases illustrated in Fig. 1, which are design entry, synthesis, mapping, placement and routing, FPGA programming, and verification. The latter may occur at different levels such as behavioral simulation, functional simulation, static timing analysis, post-layout timing simulation and, in-circuit verification. If we focus our attention on the design entry, four different specification methods can be envisioned, which are schematic entry, hardware description languages, system-level specification languages (SLSL) and, finally, general-purpose programming languages (GPPL).

The schematic-based approach is nowadays not very appropriate for specifying the functionality of modern systems because instead of thinking in terms of algorithms and data structures it forces the designer to deal directly with the hardware components and their interconnections. Contrariwise, the hardware description languages - HDLs (such as VHDL and Verilog) are widely used for design specification since they typically include facilities for describing structure and functionality at a number of levels, from the more abstract algorithmic level down to the gate level. Recently, commercial tools allowing digital circuits to be synthesized from system-level specification languages, such as Handel-C and SystemC, have appeared on the market. In this area, C and C++ with application-specific class libraries and with the addition of inherent parallelism are emerging as the dominant languages in which system descriptions are provided. This fact allows the designer to work at a very high level of abstraction, virtually without worrying about how the underlying computations are executed. Consequently, even computer engineers with a limited knowledge of the targeted FPGA architecture are capable of producing rapidly functional, algorithmically optimized designs.

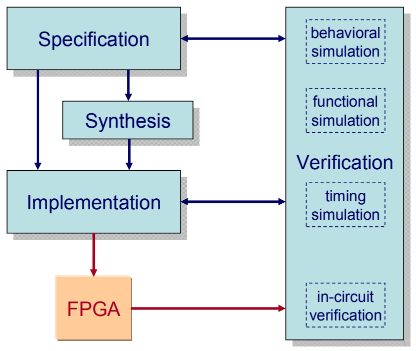


Figure 1. Typical FPGA design flow.

Even higher level of abstraction is achieved with general-purpose programming languages, such as C++ or Java. During the last years commercial tools (for instance, Catapult Synthesis from Mentor Graphics and CoDeveloper from Impulse) started appearing on the market allowing the respective high-level descriptions to be transformed automatically to an HDL, which is further used for synthesis. In this case the code portions that can be executed in parallel are identified automatically by the design tools. Besides of the mentioned design specification methods there exist other opportunities such as vendor libraries, graphical finite state machine editors, parameterizable IP cores and so on.

If we try to assess different design specification methods according to such criteria as performance, FPGA resource usage, portability, ease to learn, ease to change and maintenance and, finally, the development time, we would come to the following conclusions. Schematic-based approach allows circuits with very good performance and efficient resource usage to be created. However, when we speak about portability and ease to learn, change and maintenance and the associated development time, schematic entry is an obvious outsider. As it was noticed in an EE Times survey “The days of designing FPGA with schematics are gone.” [9]. Hardware description languages are currently the golden mean of the design entry methods. They allow creating high-performance circuits, optimized from the resource usage point of view, the development time is not so long and providing changes to the design is not very difficult. The only weak point is that HDLs are not so easy to learn. Obviously, system-level and high-level languages possess the highest portability and the highest level of abstraction. Of course, the higher level of abstraction leads to some performance degradation and not very efficient resource usage. On the other hand SLSLs and GPPLs have many advantages such as ease to learn, ease of change and maintenance, and a very short development time. Therefore, we can expect that as the tools responsible for generating hardware (more specifically, either an EDIF – electronic design interchange format file or an HDL file) from high-level source code advance, the SLSLs and GPPLs may become the predominant hardware description methodology, in the same way as general-purpose high-level programming languages have already supplanted microprocessor assembly languages.

According to Moore’s law, chip complexity grows exponentially with time. But what is important is that the number of available transistors grows faster than the ability to meaningfully design with them. This situation is a well known design productivity gap, which was inherited by FPGA from ASIC and which is increasing continuously. Therefore the design productivity will be the real challenge for future systems. It is believed that platform FPGA could alleviate this problem since they offer the flexibility, time to market and the bandwidth requirements to rapidly bring electronic systems to market. With such highly programmable platforms that include one or more programmable processor(s) and/or reconfigurable logic, derivative designs may be created without fabricating a new system-on-chip (SOC) [10]. Platform customization for a particular SOC derivative then becomes a constrained form of design space exploration: the basic communications architecture and platform processor choices are fixed, and the design team is restricted to choosing certain customization parameters and optional IP from a library [10].

In order to increase the design productivity the following challenges must be met. First of all, design reuse must be encouraged. Reusable, high-level functional blocks (such as IP blocks) offer great potential for productivity gains because design effort for the reused logic is only a portion of the effort needed for newly designed logic. According to ITRS, it is expected that reuse rate for system-level design will increase from 38% in 2007 to 58% in 2022 [10].

The second point is that design abstraction levels must be raised. Higher levels of abstraction allow many forms of verification to be performed much earlier in the design process, reducing time to market and lowering cost by discovering problems earlier [11]. We have already mentioned that tools are currently emerging allowing for hardware design at a very high level of abstraction.

And finally, the third point is to increase the level of automation which inevitably will allow the number of design iterations to be reduced. In case of platform-based design, further improvements in automated software/hardware partitioning tools are strongly required.

Now it is clear, that reconfigurability will certainly be the key aspect of future systems since it will be required for fault tolerance, for example, for molecular-scale systems, and for development of adaptive and self-correcting or self-repairing circuits. In addition, reconfigurability increases reuse, since existing devices can be reprogrammed to fulfill new tasks. According to ITRS forecast more and more SOC functionality will become reconfigurable [11].

III. Overview of Courses on Reconfigurable Systems

In Aveiro University, disciplines on reconfigurable computing are offered to students of two curricula: MIECT – Integrated Master Degree in Computer Engineering and MIEET – Integrated Master Degree in Electronics and Telecommunications Engineering. Both curricula were recently restructured due to Bologna process.

For MIECT students an obligatory "Reconfigurable Computing" discipline is taught during the 4th year and for MIEET students two optional courses ("Reconfigurable Digital Systems" and "Hardware Description Languages") are offered during the 5th year of study. Within these disciplines the following set of topics is considered:

- Overview of hardware description languages and system level specification languages;
- Introduction to FPGAs (different Xilinx FPGA architectures are analyzed, such as Spartan-II, Spartan-IIE, Spartan-3, Spartan-3E, Virtex-EM, Virtex-II, Virtex-II Pro, Virtex-4, Virtex-5);
- Synthesizable VHDL in detail;
- Statically and dynamically reprogrammable circuits;
- Methods and tools for generating parameterizable circuits;
- Methods and tools for functional simulation;
- Methods and tools for synthesis, implementation, and test of reprogrammable systems;
- Specification, design and implementation of reprogrammable control circuits, which support hierarchy, recursion, and parallelism;
- Design of practical applications in the areas of computing, electronics, telecommunications, etc.

IV. Multimedia Tool for Teaching Reconfigurable Computing

A. *Prototyping System*

The first course on reconfigurable computing in Aveiro University was introduced into pedagogical plans in 1997/1998. Since that time until present different prototyping boards have been used for laboratory works (see Fig. 2).

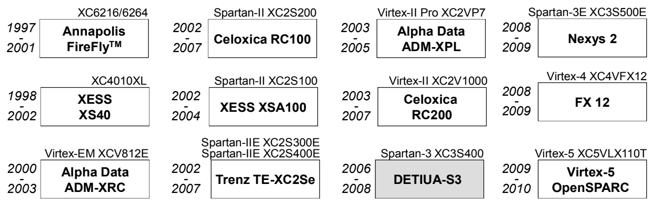


Figure 2. Prototyping boards used within the courses on reconfigurable computing in Aveiro University.

In 2005 we came up with a necessity to create our own board [12] that was fully designed and tested by post-graduate students [13]. Starting from 2006/2007 academic year the board DETIUA-S3 is intensively used in educational process.

The architecture of the developed prototyping system is illustrated in Fig. 3. The basic hardware/software components of the system are the following [13]:

- XC3S400-4-PQ208 FPGA of Spartan-3 family with 400.000 system gates [14].
- XC9572XL CPLD which is used for FPGA reconfiguration.
- 16 Megabit Am29LV160D flash memory, which is divided in three logical sections (see Fig. 3). The first section stores the default configuration bitstream that is downloaded to FPGA when data have to be transferred to/from flash memory. The first section is programmed just once during the board manufacturing and after that is not accessible to the user. The second logical section stores the user bitstream. Finally, the third section allows keeping up to 7 additional user bitstreams or any other user data. As a result, the flash memory permits to use the board as an autonomous device (which does not have any connection to the host computer) and provides support for implementing virtual systems.
- Power supply is provided either through USB interface or from an external dedicated power source.
- DLP-USB245M USB module which is used for powering, programming and data exchange between the host computer and the prototyping board. This module can be replaced with a Bluetooth module [13].
- 80 MHz clock oscillator.
- JTAG connector for external programming of CPLD and FPGA (this is needed in particular when the first

logical section of flash memory becomes damaged and needs to be reprogrammed).

- Expansion connectors for interacting with extension boards.
- Prototyping Board Manager (PBM) which provides user-friendly interface to the board through either USB or Bluetooth interfaces. In particular, PBM allows to replace the default bitstream in the first section of flash memory, to download user bitstreams to the second and the third sections of flash memory, to erase selected memory sectors, to exchange data with the board through USB (this is very useful for testing/debugging user circuits). Besides, PBM is also able to detect and report a number of problems with the board.

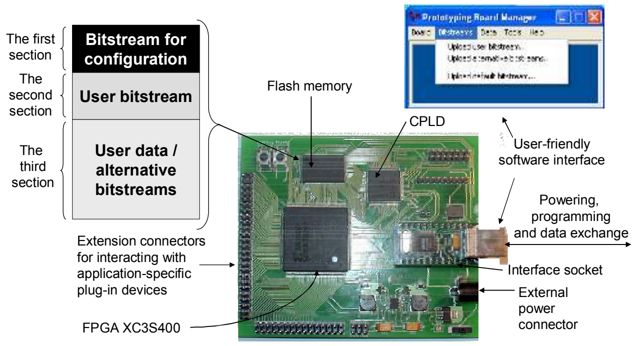

The first section
Bitstream for configuration
The second section
User bitstream
The third section
User data / alternative bitstreams
Flash memory
CPLD
Prototyping Board Manager
User-friendly software interface
Powering, programming and data exchange
Interface socket
External power connector
Extension connectors for interacting with application-specific plug-in devices
FPGA XC3S400

Figure 3. Architecture of DETIUA-S3 prototyping system.

B. *Multimedia Tool*

During laboratory classes within the courses on reconfigurable computing, the students have to design, synthesize, simulate, implement, and test a number of proposed to them systems of medium complexity. Examples of such systems are:

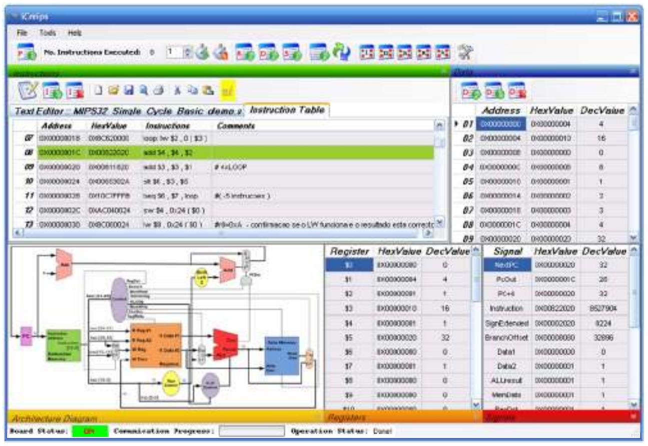
- A VGA controller;
- Interface with a mouse;
- Interface with a keyboard;
- An LCD controller;
- A simple calculator (allowing data to be entered from a keyboard and visualizing the results on a VGA monitor screen);
- A simple processor.

During the first weeks of the courses, when the students have to become proficient in VHDL and the target FPGA architecture, we propose to reuse the knowledge of computer architecture to speed up the educational process. This is possible, since the students of the courses on reconfigurable computing have a good knowledge of computer architecture. In particular, within "Computer Architecture I" discipline they acquired a solid understanding of MIPS architecture [15].

Therefore a fully synthesizable VHDL model of an instruction subset of MIPS architecture [15] was created. This work was done by a final year student within his M.Sc. dissertation in 2007/2008. With the aid of the proposed multimedia tool, the teacher has a possibility to generate a VHDL model of the processor in which some components (selected by the teacher) are missing (for example, an ALU or a register file). It is possible to exclude different components for different groups of students to diversify their work and to eliminate the possibility of copying. Then the students are asked to describe in VHDL just the omitted components (all the interfaces between the processor components are fixed and may not be altered), to functionally simulate these components, and, finally, to synthesize and implement in an FPGA the whole processor.

In order to be able to verify the correct functionality of the processor, a multimedia tool iCmips [16] was developed whose general interface (depicted in Fig. 4) is similar to SPIM simulator [17], but the great difference is that assembly language instructions are actually executed in real hardware created by the students. The contents of the registers as well as data memory are visible to the user and updated in real time. It is possible to either execute assembly language instructions step by step, run the user program until a specified breakpoint, or execute the whole user code at once. Currently, MIPS co-processors are not supported.

This teaching method has a number of advantages. First of all, it allows to design and test in an FPGA a simple circuit during the first laboratory classes, when the students do not have yet enough knowledge of VHDL. Besides, with the aid of the proposed multimedia tool the students are able to appreciate the results of the work immediately through a user-friendly interface. And, finally, it is well known that students learn in different ways therefore a number of learning activities should be provided. The suggested multimedia tool enriches the set of teaching methods employed within the courses on reconfigurable computing.



	Address	HexValue	Instructions	Comments
07	0X00000018	0X8C620000	loop: lw $2 , 0 ($3)	
08	0X0000001C	0X00822020	add $4 , $4 , $2	
09	0X00000020	0X00611820	add $3 , $3 , $1	# 4xLOOP
10	0X00000024	0X0065302A	slt $6 , $3 , $5	
11	0X00000028	0X10C7FFFB	beq $6 , $7 , loop	#(-5 instrucoes)
12	0X0000002C	0XAC040024	sw $4 , 0x24 ($0)	
13	0X00000030	0X8C080024	lw $8 , 0x24 ($0)	#r8=0xA - confirmacao se o LW funciona e o resultado esta correcto

	Address	HexValue	DecValue
01	0X00000000	0X00000004	4
02	0X00000004	0X00000010	16
03	0X00000008	0X00000000	0
04	0X0000000C	0X00000008	8
05	0X00000010	0X00000001	1
06	0X00000014	0X00000002	2
07	0X00000018	0X00000003	3
08	0X0000001C	0X00000004	4
09	0X00000020	0X00000020	32

Register	HexValue	DecValue
$0	0X00000000	0
$1	0X00000004	4
$2	0X00000001	1
$3	0X00000010	16
$4	0X00000001	1
$5	0X00000020	32
$6	0X00000000	0
$7	0X00000001	1
$8	0X00000000	0
$9	0X00000000	0

Signal	HexValue	DecValue
NextPC	0X00000020	32
PcOut	0X0000001C	28
PC+4	0X00000020	32
Instruction	0X00822020	8527904
SignExtended	0X00002020	8224
BranchOffset	0X00008080	32896
Data1	0X00000000	0
Data2	0X00000001	1
ALUresult	0X00000001	1
MemData	0X00000001	1

Figure 4. User interface of a multimedia tool iCmips.

C. *Implementation Details*

Single-cycle datapath have been implemented consisting of the following major components: instruction memory, program counter, ALU, data memory, control blocks, multiplexers, adders, shifters, and sign extension blocks [15]. It is known that systems based on MIPS processors typically divide memory into three parts: text segment, data segment, and stack segment [15]. We followed this convention and constructed the memory on the basis of Spartan-3 embedded RAM blocks. Spartan-3 embedded memory blocks have a capacity of 16384 bits and their depth/width aspect ratios are programmable. We employed a block *RAMB16_S36*, which is a single-port synchronous Block RAM with 32-bit data bus (not counting parity bits) and 9-bit address bus.

The memory block was divided as follows. The text segment, which holds the user program's instructions, starts at address *0x000*. The data segment resides above the text segment starting at address *0x100* and contains static data followed by dynamic data. The stack segment occupies the top of the address space (starting at address *0x1ff* - the stack grows in the direction of lower addresses). With this memory organization, user programs can contain at most 256 assembly language instructions and exploit at most 256 32-bit words of static and dynamic data. Of course, such an implementation does not allow real-world programs to be executed but, on the other hand, the available memory is more than sufficient to check students' knowledge of VHDL and to test their projects. Besides, memory can easily be expanded by using multiple embedded RAM blocks and this could constitute one of the tasks given to the students.

The implemented single-cycle datapath is displayed to the user on the host computer monitor screen. The user interacts with the processor in the following way:

- Create an assembly program using the supported MIPS instructions;
- Generate machine code with the aid of an embedded into the tool assembler;
- Download the code to the text segment;
- Download static user data to the data segment;
- Optionally set a breakpoint;
- Execute the program using one the following modes: single instruction, multiple instructions until the specified breakpoint is reached, or the whole program at once;
- The tool as well as the FPGA monitoring circuit perform permanent observation of the contents of registers, data memory and main control signals that manage the flow of information in the datapath. All these data are made available to the user through a graphical interface (Fig. 4).

V. Conclusion

In this paper, we suggested a multimedia tool, which could contribute to productive teaching of reconfigurable computing. We hope that the tool is of great value since it provides to the students an opportunity to construct, physically implement, and test a quite complicated system with a very limited knowledge of hardware description languages.

Besides, there exist a lot of possibilities of extensions. For example, it is known that the performance of single-cycle implementation is low since the clock period is determined by the longest possible path in the processor whereas several instruction classes could fit in a shorter clock cycle. Therefore, one possible extension to our work is to implement a multi-cycle or a pipelined processor. Comparing these three alternative MIPS implementations would constitute an excellent exercise for the students. Another opportunity of future work is implementing different co-processors (for example for supporting floating-point operations), that could enrich the experience of the students.

Acknowledgment

The author would like to acknowledge contribution of Bernardo Silva who developed reconfigurable hardware and software for iCmips project.

References

[1] History of Xilinx, Online: http://www.xilinx.com/company/history.htm.

[2] M. Uhm, "Making the adaptivity of SDR and cognitive radio Affordable", DSP Magazine, issue 2, May 2006, pp. 25-27.

[3] M. Devlin, "Multi-FPGA systems for high performance computing applications", IEE Developers Forum, October 2005, Online: http://www2.theiet.org/oncomms/sector/electronics/Articles/Heading/512.

[4] S. Margerm, "Reconfigurable computing in real-world applications", FPGA and Structured ASIC Journal, February 2006, Online: http://www.fpgajournal.com/articles_2006/20060207_cray.htm.

[5] J. Turley, "Survey: Who uses custom chips", Embedded Systems Programming, vol. 18, no. 8, August 2005.

[6] Altera, Programmable platform solutions, August 2006, Online: http://www.altera.com/literature/wp/wp-01004.pdf.

[7] V. Sklyarov, I. Skliarova, "Teaching reconfigurable systems: methods, tools, tutorials, and projects", IEEE Trans. on Education, vol. 48, no. 2, 2005, pp. 290-300.

[8] R. Hartenstein, "Reconfigurable computing (RC) being mainstream: torpedoed by education", keynote talk at Int. Conf. on Microelectronic Systems Education, Anaheim, USA, June 2005.

[9] EE Times (2006, July 31), R. Goering, "FPGA users rank challenges, tasks".

[10] International technology roadmap for semiconductors, Design, 2007.

[11] International technology roadmap for semiconductors, System Drivers, 2007.

[12] M. Almeida, V. Sklyarov, I. Skliarova, B. Pimentel, "Design tools for reconfigurable embedded systems", Proc. of the 2nd Int. Conf. on Embedded Software and Systems - ICESS'2005, IEEE Computer Society, Xi'an, China, December 2005, pp. 254-261.

[13] V. Sklyarov, I. Skliarova, M. Almeida, B. Pimentel, "A prototyping system for mobile devices", Proc. of the ACM 2007 Int. Wireless Communications and Mobile Computing Conf. - IWCMC'2007, Honolulu, USA, August 2007, pp. 505-510.

[14] Xilinx, Inc., products and services, Online: http://www.xilinx.com.

[15] J.L. Hennessy, D.A. Patterson, Computer Organization and Design. The Hardware/Software Interface, Morgan Kaufmann Publishers, Inc., 2004.

[16] iCmips, Online: http://clientes.netvisao.pt/aliencod/iCmips_web/index.htm.

[17] SPIM – A MIPS32 Simulator, Online: http://www.cs.wisc.edu/~larus/spim.html.