Teaching FPGA-Based CPU Cores and Microcontrollers

Valentina Rankovska
Faculty of Electrotechnics and Electronics at Technical University of Gabrovo
4 H. Dimitar str.,
Gabrovo 5300
Bulgaria
rankovska@tugab.bg

ABSTRACT: *The method for teaching the security features of the FPGA-based CPU cores and microcontrollers is outlined in this paper basically. To do so, the students face limitations with the complications of the software and hardware of the FPGA and CPU. To obtain the optimum results we in this paper have given illustrations to get better results and suggested tools.*

Keywords: Field-Programmable Gate Arrays (FPGA), CPU (Central Processing Unit) Cores design, Microcontrollers, Teaching methodology

Received: 19 August 2022, Revised 28 September 2022, Accepted 10 October 2022

DOI: 10.6025/isej/2022/9/2/25-31

1. Introduction

In the second half of the PLD (Programmable Logic Devices)-Based Circuits Design course the students have to study implementation of the complex programmable logic arrays to develop embedded systems. They are already familiar with the architecture, features, operation, building blocks modes, etc. of the conventional microcontrollers. This is an actual problem because of the gradual involving of those innovative devices into the scope of digital and microprocessor circuits with the respective advantages [1].

The use of FPGAs also allows the students to study the operation of the microprocessors form the inside, especially when they try to make it themselves [6], [7], [8], [9]. On the other hand it is very difficult for the students to use them, because of the following reasons:

• The architecture complexity of the FPGA integrated circuits in comparison to the conventional ones;

• The more complex digital and microprocessor devices design technology where completely new stages and tools are used together with the traditional ones;

• **Related with the previous point –** new and considerably more complex integrated development environments with a quite many features and possibilities;

• The fact that the enormous logic capacity and the possibility for using a great variety of IP (Intellectual Property) cores allow building into one chip designs with great circuit complexity and multidisciplinary nature;

• **Except the upper featured objective preconditions –** also the lack of students' motivation especially in the present economic conditions, etc.

The aim of the present work is to suggest a methodology for teaching/ study of a CPU (Central Processor Unit) core and peripherals design with simple operation for educational and research purposes.

2. Preliminary Knowledge and Skills of the Students

Preliminary knowledge and skills form from previous courses, connected with:

• Digital elements, devices and basic circuits;

• Architecture and operational principles of a hypothetical microprocessor, microcontroller, embedded system;

• Architecture, building blocks and operational principles of conventional general purpose microcontrollers;

• Assembler and high level programming languages.

Before to begin studying CPU cores design in the current course the students are also familiar with:
• Architecture, features and resources of the Field- Programmable Gate Arrays;

• Stages, hardware and software tools used to design digital circuits in FPGAs;

• Hardware description languages such as VHDL, Verilog, AHDL, etc. (Currently VHDL is used.)

Having in mind the difficulties for the students featured before, they have to be supplied with enough data and documentation:
• FPGA and development boards documentation – handbooks, user manuals, tutorials, example projects, which are useful in studying the features of the programmable logic and development boards;

• Integrated development environment documentation including tutorials helping to study the design sequence;

• Documentation and tutorials concerning the hardware description language used (VHDL);

• Architecture and features of IP cores – CPU and peripherals, which could be used freely for educational purposes;

• Links to Internet resources like:

• Companies producing programmable logic and hardware and software development tools;

• Sites of similar university courses, where often variety of useful information could be found: lecture presentations, labs, course and final projects, etc.

• Projects and forum sites concerning the subjects, etc. Except supplying with documentation the requirements to the students connected with the stated labs and projects problems have to be extremely clear and detailed.

All this is achieved to a greater degree by the means of elearning, part of which is the Moodle course for the discipline [4].

3. Teaching Methodology in Designing FPGA-based CPU Cores and Peripherals

FPGAs of Altera are used in the learning process together with the free version of the software - Quartus II Web Edition and development boards TREX C1 and DE2-70 with FPGA Cyclone and Cyclone II respectively.

There are two possible design approaches, which can be used with the full version of Quartus II:

• **Flat compilation flow with no design partitions –** the entire design is compiled together; it is applied for small and not too complex designs. The software performs the defined logic and placement optimizations on the whole design. This approach is easy to implement and is the only one possible with the free version of Quartus II. It is not convenient for large designs as the compilation time increases considerably.

• **Incremental compilation with design partitions -** the design is split into partitions, on which different designers can work independently. This can simplify the design process and reduce compilation time. It is preferable in large and complex designs.

The first approach is used in the laboratory classes as on one hand it is free and on the other hand it is easier to implement with the comparatively simple students' projects. The CPU core design is a complicated and quite difficult process. That is why the students wouldn't be able to acquire the design of a core for a real application for the short time of the course. And also the course is not dedicated only to microprocessor systems design, but to more general object.

We must have in mind that during the Microprocessor Circuits course (held in the previous semester) they have studied the architecture and the operation of a hypothetical microprocessor on the level of building blocks. That is why it is necessary to pass through the following stages in studying the design of microprocessors and microcontrollers in order to be easier for the students:

1. Embedded system design using software module for its generation and library components, defined from the designer (student):
• Studying the library features and resources of the software module for system-on-a-programmable chip synthesis SOPC Builder (included in Quartus II);

• Studying the architecture, features and configuring options for the three versions of the software core Nios II of Altera;
• Designing an embedded system based on Nios II with reduced set of peripherals.

2. Design of a CPU core with simple set of operations and peripherals:
• Planning the CPU core architecture with a reduced set of operations;
• Designing definite blocks of the CPU core, beginning with the Arithmetic-Logic Unit (ALU);
• Expanding the microprocessor core to achieve a working variant for learning purposes;
• Designing memory blocks and peripherals with reduced complexity;
• Joining together and testing the operation of the CPU core and the peripheral blocks.

3. Analysis and study of a ready CPU core functionalities with simple architecture
Different design approaches are used at the different teaching stages. Firstly the students learn to develop digital projects by inputting the design using Block/ Symbol Editor and library components of Quartus II (they synthesize the circuit (circuits) in a form of a block/functional diagram). The approach is used in the first stage. The modeling in the second stage is performed in VHDL.

It must be noticed here that software development tools for the student processor are not created having in mind the defined course purposes and the limited course workload.

Developing an embedded system using library components
The software module SOPC Builder is used for system in a chip generation. A microprocessor circuit, based on Altera's software processor Nios II is developed (Figure 1) [3].

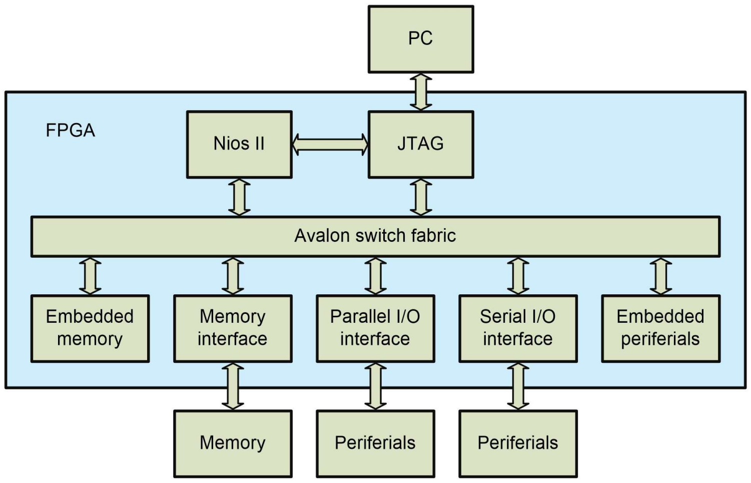


Figure 1. SOPC based on Nios II and DE2-70

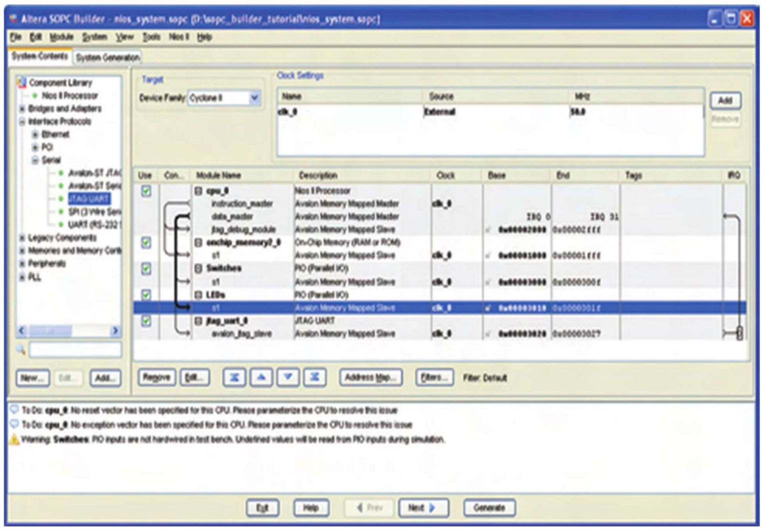
SOPC Builder enables to define and generate entire system in a programmable chip in an easy way. The designed system could be based on a processor or not and if there is a processor in it, it may be Nios II or other. The module connects together the components of the system automatically. For that purpose it makes the routing and (if it is necessary) generates interconnect logic. That is why it is very suitable for the lowest level of learning.

The GUI (Graphical User Interface) of SOPC Builder with library components added is shown in Figure 2.

It was mentioned that at this stage the students are already familiar with a hardwired conventional microcontroller from the Microprocessor Circuits course. Now they have the opportunity to design their own microcontroller with flexible reconfigurable architecture and to test it. And moreover – on one hand they do not need still to examine the inside structure of the used blocks and on the other hand they may configure them according to the assignment in the laboratory class.

Design of a CPU core and peripherals

• Planning of a CPU core architecture



Target

Device Family: Cyclone II

Clock Settings

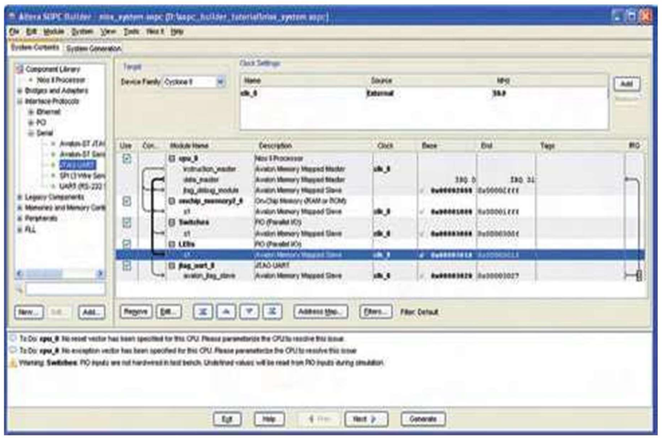
Name	Source	MHz
clk_0	External	50.0

Use	Con...	Module Name	Description	Clock	Base	End	Tags	IRQ
✓		cpu_0	Nios II Processor					
		instruction_master	Avalon Memory Mapped Master	clk_0				
		data_master	Avalon Memory Mapped Master		IRQ 0	IRQ 31		
		jtag_debug_module	Avalon Memory Mapped Slave		0x00002800	0x00002fff		
✓		onchip_memory2_0	On-Chip Memory (RAM or ROM)					
		s1	Avalon Memory Mapped Slave	clk_0	0x00001000	0x00001fff		
✓		Switches	PIO (Parallel I/O)					
		s1	Avalon Memory Mapped Slave	clk_0	0x00003000	0x0000300f		
✓		LEDs	PIO (Parallel I/O)					
		s1	Avalon Memory Mapped Slave	clk_0	0x00003010	0x0000301f		
✓		jtag_uart_0	JTAG UART					
		avalon_jtag_slave	Avalon Memory Mapped Slave	clk_0	0x00003020	0x00003027		

Filter: Default

```
To Do: cpu_0: No reset vector has been specified for this CPU. Please parameterize the CPU to resolve this issue
To Do: cpu_0: No exception vector has been specified for this CPU. Please parameterize the CPU to resolve this issue
Warning: Switches: PIO inputs are not hardwired in test bench. Undefined values will be read from PIO inputs during simulation.
```

Figure 2. Components of SOPC system based on Nios II



Use	Con...	Module Name	Description	Clock	Base	End	Tags	IRQ
✓		cpu_0	Nios II Processor					
		instruction_master	Avalon Memory Mapped Master	clk_0				
		data_master	Avalon Memory Mapped Master		IRQ 0	IRQ 31		
		jtag_debug_module	Avalon Memory Mapped Slave		0x00002800	0x00002fff		
✓		onchip_memory2_0	On-Chip Memory (RAM or ROM)					
		s1	Avalon Memory Mapped Slave	clk_0	0x00001000	0x00001fff		
✓		Switches	PIO (Parallel I/O)					
		s1	Avalon Memory Mapped Slave	clk_0	0x00003000	0x0000300f		
✓		LEDs	PIO (Parallel I/O)					
		s1	Avalon Memory Mapped Slave	clk_0	0x00003010	0x0000301f		
✓		jtag_uart_0	JTAG UART					
		avalon_jtag_slave	Avalon Memory Mapped Slave	clk_0	0x00003020	0x00003027		

To Do: cpu_0: No reset vector has been specified for this CPU. Please parameterize the CPU to resolve this issue.
To Do: cpu_0: No exception vector has been specified for this CPU. Please parameterize the CPU to resolve this issue.
Warning: Switches: PIO inputs are not hardwired in test bench. Undefined values will be read from PIO inputs during simulation.

Figure 3. Functional circuit of a general purpose CPU

It is a considerably complicated stage which is difficult for the students at most degree. That is because till the moment they have studied a ready submitted structure and an operation of a microprocessor and a microcontroller without explaining the reasons for being that. So the processes passing in the blocks and the ways of their interaction are still hidden for the students.

At the CPU core design, although it is not obvious, the process begins not from the mechanical "collecting" of blocks building its architecture (though at the end it will looks like a known one), but from the answers of the following general questions:

- What kind of operations have to perform the designed processor;
- How it will access the various types of memory blocks, etc.

A functional diagram of a general purpose microprocessor is shown in Figure 3 [7].

After defining the instruction set the ways of their decoding and execution have to be determined. Several questions must be answered: how many and what kind of operations must be able to execute the processor; what will be the mnemonic code of the instructions; what operation code will be assigned to each instruction and what will be their length.

After that the design of the executable unit follows. A datapath has to be made on this step, so the following questions have to be answered: what kind of operation unit we need; how many registers; how will they be organized; how will be connected the two units – the control and the executable ones.

At creating the datapath we have to determine how the processor will fetch and execute the instructions from the program memory. In this connection there are additional operations and registers, for instance the program counter (PC), the instruction register (IR), etc.

The control unit cyclic pass through three major steps, called usually instruction cycle: 1) instruction fetch; 2) instruction decoding and 3) execution. Every step is executed during one state of the state machine.

The simplest variant of a completely synchronous programmable logic-based device is a RISC processor with a two-stage pipelined execution of the instructions.

• Arithmetic-Logic Unit
We use a minimum number of instructions executed by the ALU for the example training processor – addition, subtraction, increment, decrement, logic AND, OR, NOT and accumulator output. Hence three bits are enough for the operation codes (S). Two general purpose registers are used – A and B. The operands are 4-bits long in order to implement the project using the 18 switches of the development board. We have to input the operands and the machine code of the instruction tested. The process, in which body the instructions are modeled (in VHDL), is the following:

```
PROCESS(S, A, B)
BEGIN
    CASE S IS
        WHEN "000" => F <= A;
        WHEN "001" => F <= A AND B;
        WHEN "010" => F <= A OR B;
        WHEN "011" => F <= NOT A;
        WHEN "100" => F <= A + B;
        WHEN "101" => F <= A - B;
        WHEN "110" => F <= A + 1;
        WHEN OTHERS => F <= A - 1;
    END CASE;
END PROCESS;
```

The students perform a functional simulation to test all the operations and for that purpose they first make a vector waveform file. Practically the design is tested on the DE2-70 development board. A test of the "adding" operation is shown in Fig. 4.

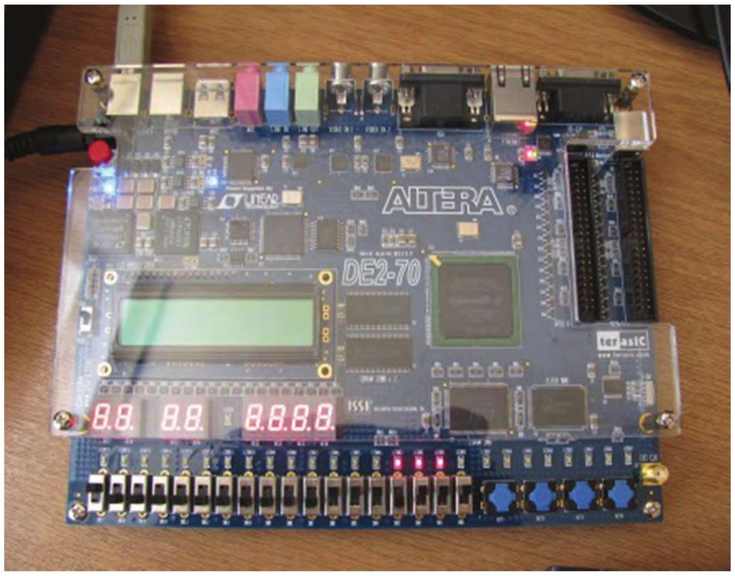


Figure 4. Testing the instruction for addition of two register operands

After that additional blocks and features have to be added at the development process, such as: interrupt management block; reset control block; ways to access the memory blocks, including pointer and data registers, instructions, status flags, etc.

It is also necessary to organize the pipelined execution of the instructions.

These problems are quite complicated to be performed in the laboratory classes and further more - for the part of the semester. That is why the students have to be supplied not only with detailed directions for completing the tasks but also with additional information like ready models of similar blocks. They will have the opportunity and the task to analyze and modify them and also to make course and final projects.

• Program and Data memory

The FPGAs Cyclone II include embedded memory consisting of columns of M4K blocks that can be configured to provide various memory functions such as RAM, first-in first-out (FIFO) buffers, and ROM. M4K memory blocks provide over 1 Mbit of RAM at up to 250-MHz operation. It is not enough volume for the most real applications but it is enough for the designed simple processor.

The M4K blocks which are used as a program memory, are initialized in advance with the operation codes of the instructions of the example test program. For that purpose the embedded in Quartus II memory editor is used.

Analysis and study of a ready CPU core functionalities with simple architecture

The study of a ready software core for a real application is an extremely complicated process, even if it is simple. It is a stage which could be applied in activities like extracurricular unaided work or team-based work at the end of the course, practices, course and final projects. For that purpose it is convenient to use free software core with a comparatively simple functions [2], [5].

4. Conclusion

A methodology for teaching FPGA-based CPU cores and microcontrollers from the very first stages is presented in the paper.

There are many circumstances which make difficult for the students to study the microprocessor and microcontroller development using programmable logic. The suggested approach includes step by step beginning with simple problems design of two kinds of microprocessor systems with minimum functionalities – using library components and modeling a system with VHDL.

The future work addresses enlarging the e-learning means, like animations, multimedia, etc.

Also an archive of custom (students) library components and projects will be made, which will be accessible for the next-year students as useful examples – to learn and expand.

Acknowledgement

The present work is partially supported by the Science Research Fund at the Ministry of Education, Youth and Science.

References

[1] Rankovska, V. & Karailiev, H. (2011). (in Bulgarian). *Digital Devices and Systems Design Using Field-Programmable Gate Arrays*, Vol. 1–2, pp. 8–15.

[2] Rankovska, V. (2011), (in Bulgarian) Microprocessor cores for complex programmable logic arrays. *Unitech'11* [Conference proceedings], Vol. 1. Gabrovo, Bulgaria, pp. I-186–I-191.

[3] Rankovska, V. (2012) FPGA (Field Programmable Gate Arrays) – Based System-on-a-Programmable-Chip Development for Educational Purposes. *ICEST 2012* [Conference proceedings], Vol. 2. Sofia, pp. 489–492.

[4] umis.tugab.bg/moodle/.

[5] www.opencores.org.

[6] Sklyarov, V. & Sklyarova, I. (2006) *Multimedia tools for teaching reconfigurable systems*, MoMM2006 [Conference proceedings]. Yogyakarta, Indonesia, pp. 211–220.

[7] Hwang, E.O. *Digital Logic and Microprocessor Design with VHDL*. La Sierra University: Riverside, 2005.

[8] Weng, T., Zhu, Y. & Chung-Kuan, C. (2008) "*Digital Design and Programmable Logic Boards: Do Students Actually Learn More?", 38th ASEE/IEEE Frontiers in Education [Conference proceedings]. Saratoga Springs,* NY, USA.

[9] Olivares, J., Palomares, J.M., Soto, J.M. & Gámez, J.C. (2010) "Teaching Microprocessors Design Using FPGAs", *IEEE EDUCON education Engeneering* [Conference proceedings]. Sapin: Madrid, pp. 1189–1193.