

Teaching FPGA-based systems and their influence on mechatronics

Valery Sklyarov, Iouliia Skliarova

Department of Electronics and Telecommunications/IEETA, University of Aveiro, Portugal
skl@det.ua.pt; iouliia@det.ua.pt

ABSTRACT

The paper discusses teaching reconfigurable digital systems on the basis of Xilinx field programmable gate arrays (FPGAs). This direction is very important for undergraduate curricula including mechatronics. The relevant topics have been taught during more than 5 years at the Department of Electronics and Telecommunications of Aveiro University (Portugal) for students specializing in electronics and software engineering. For software engineering curriculum there are some difficulties because the students do not have sufficient background. The paper discusses teaching methodology allowing this problem to be solved.

1. INTRODUCTION

Field programmable gate arrays (FPGAs) enable engineers to shorten the design time significantly. They have already been efficiently used for developing digital systems, rapid prototyping, design space exploration, experiments, etc. The scope of potential applications of FPGAs is extremely wide and includes such areas as: chipsets for microprocessors, problem-oriented co-processors, embedded systems, systems-on-chip, networks-on-chip, and many others. The rapid evolution of FPGA technology necessitates a large number of well-prepared engineers in the relevant areas. Thus, new trends must be reflected in the respective pedagogical activity.

Mechatronics establishes links between computers and external equipment and covers such areas as robotics, industrial automation and manufacturing systems that need many application-specific customizable devices, which can be very efficiently designed on the basis of FPGAs. Thus, including the relevant topics in curricula for mechatronics is very important. Note that teaching reconfigurable systems requires knowledge in many new areas, such as hardware and system-level description languages, FPGA-targeted computer-aided design (CAD) systems, intellectual property (IP) cores, advanced synchronization techniques, design of virtual systems, dynamic reconfiguration, and many others. Majority of these topics have not been included in traditional curricula especially for specialties, which are not directly devoted to electronics (for example, software engineering, mechatronics, etc.). Thus, FPGA-targeted disciplines have to cope with many problems appeared due to non sufficient background of the students. However, there are also some other difficulties. Tremendous progress in the scope of reconfigurable digital systems has made it possible to advance customizable microchips from simple gate arrays that appeared on the market in the mid-1980s [1] to platform FPGAs containing more than 10 million system gates and incorporating complex heterogeneous structures. Such incredible evolution of FPGAs is continuing now and each year new FPGA families with more advanced capabilities are appearing on the market. They incorporate many important architectural and technological innovations. Indeed, just two years ago Xilinx Spartan-IIE family of FPGAs was recognized as a set of very advanced devices for the design of digital systems and was recommended for future applications. Today, Xilinx Spartan-3, Spartan-3L, Spartan-3E families of FPGAs are much more advanced and less expensive. Such rapid evolution demands a frequent renewal of the pedagogical plans and laboratory equipment and this process is expensive and time consuming. Today, advanced research is being intensively performed in the areas of system-on-chip and networks-on-chip [2] and this requires including the relevant topics in the respective curricula. Thus, the scope of FPGA-based systems is very dynamic and many sided.

Some recent results in teaching reconfigurable systems are discussed in [3] where models, methods and pedagogical innovations (such as using the developed student-oriented design templates and evaluation through mini-projects) are considered. This paper continues to discuss the importance

of reconfigurable systems in education and shows how FPGA-based systems have been taught for a specialty that is not directly targeted to electronics.

2. DESIGN FLOW AND DESIGN PROBLEMS

Design flow for FPGA-based circuits provides for conversion of a given specification to a bit-stream for a digital system that has to function in accordance with the given specification. This flow invokes tools for:

- Problem specification with the aid of different methods and tools allowing a system to be described with the desired details at either behavioral or structural level. Very often a composition of top-down and bottom-up design strategies is used. Most frequently the system is described in hardware description languages - HDL (such as VHDL or Verilog). Recently system-level specification languages (SLSL), such as Handel-C and SystemC have been developed. They enable the design time to be reduced significantly although the resulting circuits typically are less optimized than the circuits synthesized from HDL-based specifications.
- Modeling permitting to validate the specification, to examine the functional correctness of the circuits and to verify the timing characteristics.
- Synthesis, producing the net-list of primary FPGA components.
- Supplying details about input/output pins, input/output standards, positioning constraints, etc. This is achieved through an implementation constraints file.
- Mapping, placement, routing and generating the bit-stream. The latter is a file that can be loaded to the proper FPGA in order to configure all the necessary interconnections and functions of the FPGA components described in the net-list.

After loading the bit-stream to the FPGA the designed system can be tested in a physical environment.

In order to be able to practice the considered above design flow the students have to gain experience in a number of specific areas, namely:

- Design methodologies, such as top-down, bottom-up and mixed (top-down + bottom-up), using standard (i.e. predefined within the CAD environment) and application-specific libraries.
- General-purpose languages for modeling and validation of the system and its components in general-purpose computers.
- Coding styles and methodologies for HDLs and/or SLSLs.
- FPGA-specific design issues.
- Numerous supplementary design tools, such as schematic editors, IP core generators, design templates, graphical state machine editors, constraints editors, etc.
- Functional and timing simulation.

Obviously, it is impossible to supply knowledge for all the topics indicated above just within the disciplines dealing with reconfigurable digital systems. Thus, some topics have to be taught within the other disciplines, such as: programming and software engineering; electronics; digital design; signals and systems; digital signal processing; computer architecture; interfaces and peripheral devices. Traditionally, the relevant disciplines are included in curricula for electronics and computer engineering. For example, all these disciplines have been given to students of Aveiro University (Portugal) within the Electronics and Telecommunications engineering program. Although the disciplines indicated above provide just a partial support (for example, HDLs have not been studied) this simplifies many problems in teaching FPGA-based systems. Note that FPGA-based systems have been taught not only within the specialty mentioned above, but also for students of another curriculum, which is more targeted to software engineering and excludes many disciplines from the list above. Thus, due to not sufficient background, teaching FPGA-based systems becomes much more complicated. To cope with this problem the following technique has been applied (see section 3):

1. Rush introduction based on carefully selected examples.
2. Supplying all the necessary materials through the Internet.
3. Sharing experience of previous students through student projects and publications [3].
4. Using methods considered in [3].

3. TEACHING METHODOLOGY

The teaching material is based on the tools available for integrated software environment (ISE) of Xilinx [4] (currently the version 6.3.3 is used), which is a widely known commercial CAD system. Fig. 1 demonstrates how the ISE has been explored by the students. Two types of top-level specifications have been considered, which are schematic and VHDL specifications (see fig. 1). To test the designed circuits in a physical environment the following FPGA-based prototyping boards have been used:

- Trenz Electronic TE-XC2Se [5]. Different boards include either an XC2S300E Spartan-IIE FPGA with 300000 system gates or an XC2S400E Spartan-IIE FPGA with 400000 system gates. The board also contains 256K×16 bit static RAM, 1 MB flash RAM, an LCD (2 lines×16 characters), dip-switches, pushbuttons, and LEDs. The board can be connected to a VGA monitor, supports USB and RS-232 serial interfaces, and has two expansion headers. Power is supplied through USB, and no additional source is required.
- XESS XSA100 [6] with an XC2S100 Spartan-II FPGA (100000 system gates).
- RC100 of Celoxica [7] with an XC2S200 Spartan-II FPGA (200000 system gates).

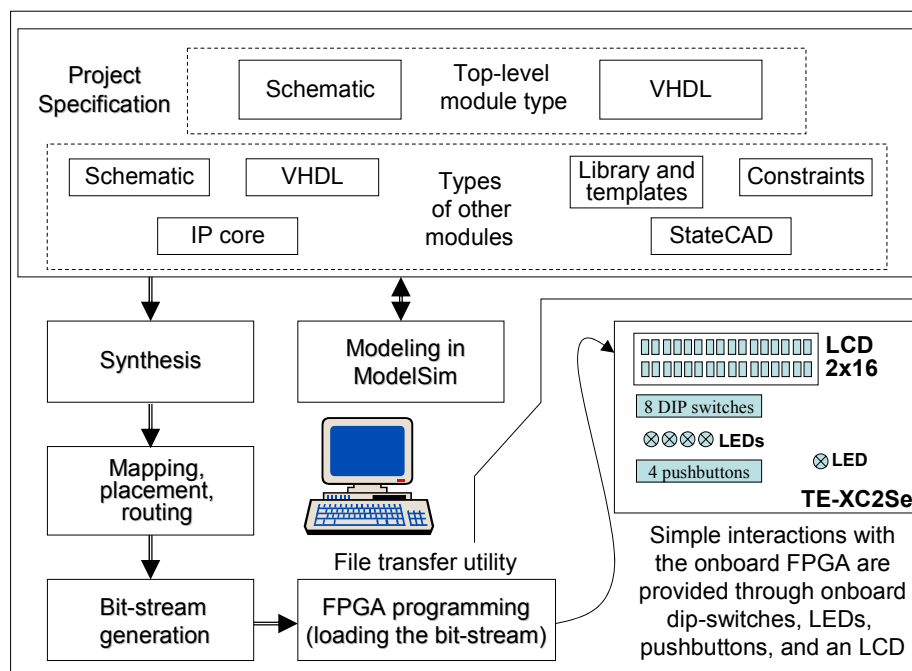


Fig. 1. Using Xilinx ISE and TE-XC2Se prototyping board in educational process

The primary problem encountered was the necessity to start laboratory works from the beginning of the course and the students practically did not have any knowledge of VHDL, i.e. of the language that had to be used for practical design. That is why the course has been organized in such a way that permits a rush introduction to VHDL during just three first theoretical classes. The material was arranged so that the topics of the theoretical classes would be sufficient to apply them for practical design starting from the beginning of the semester. In fact, this task is not simple because VHDL is quite complicated language and, as a rule, its main features should be considered in a proper sequence. The problem becomes more complicated when we want to provide an interaction with peripheral devices available on the prototyping boards, such as an LCD, dip-switches, pushbuttons, LEDs, etc. VHDL can easily be used to describe all the necessary interface circuits. However, relatively complicated language constructions should be employed for such purposes and once again this requires the appropriate knowledge.

In general, the material for the relevant disciplines is divided in four segments. The first segment includes three theoretical and three practical classes and can be characterized as a *rush introduction*. The first theoretical class (given before the first practical class) presents a brief overview of the FPGA technology and FPGA-based design, followed by a simple but complete example. The latter includes a

behavioral VHDL code for a circuit that divides the master clock frequency up to 1 Hz and uses this frequency to switch on/off an individual LED available on the prototyping board [5]. The example is trivial but it enables the students to understand the basic principles of the language and the primary stages of the design in the ISE, which are the following:

- VHDL permits an interface and an implementation to be separated through such constructions as *entity* and *architecture* (see fig. 2(a)(b));
- The interface (*entity*) enables the designer to specify external inputs and outputs to the circuit (see fig. 2(a)). The types *std_logic* and *std_logic_vector* permit to model signals in physical circuits;
- In the implementation (*architecture*) VHDL permits internal signals to be declared;
- Physical circuits are built from components, which function concurrently, i.e. in parallel. There are different ways to describe a similar behavior in VHDL, and two of them are explained in the first example, namely a concurrent signal assignment and a process (see fig. 2(b));
- Any process can be activated concurrently and a sensitivity list, which can be used for such purposes, is explained. The sensitivity list permits for our example to activate the process on either rising or falling edge of the clock (*clk*) and *reset* signals but just the rising edge of the clock is used. The condition *reset = 0* forces initialization of the signal *internal_clock* (see fig. 2(b));
- In order to communicate with the circuit it is necessary to describe connections between the inputs/outputs of the circuit and the proper FPGA pins. This can be provided with the aid of a user-constraint file (see fig. 2(c));
- Finally, a number of steps, such as synthesis, mapping, placement, routing and generating a bit-stream can be sequentially executed in the ISE. As a result the bit-stream will be generated and loaded to FPGA through USB using the available file transfer utility [5] (see fig. 2(d)(e));
- At the end the students can appreciate the circuit functionality on the board (see fig. 2(e));

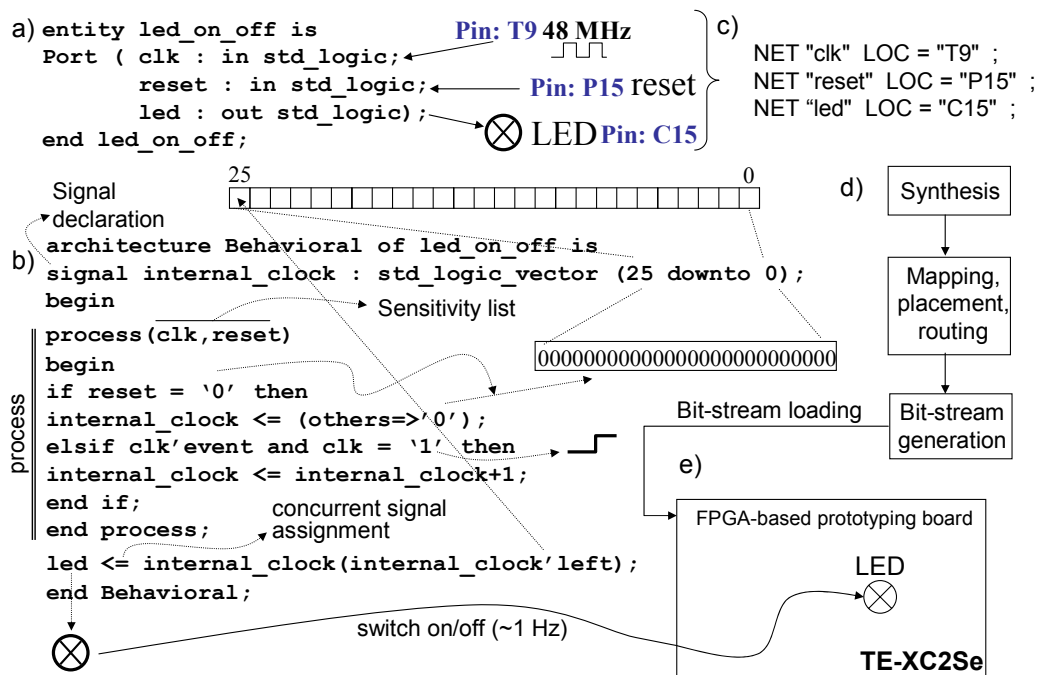


Fig. 2. The first student's project: VHDL entity (a), VHDL architecture (b), part of the user constraint file (c), using Xilinx ISE (d), loading the bit-stream and testing the circuit (e)

Note, that in spite of the simplicity of the circuit, similar steps have to be carried out for the design of complex systems. The considered trivial code with some additional explanations enables the students to understand the future course topics and how they can be used in practice.

The first practical class is devoted to introduction to the ISE and the example considered in the first theoretical class has to be completed, i.e. the circuit has to be described in VHDL, synthesized,

implemented and tested on the prototyping board [5]. After that the students have to provide some trivial changes to the circuit functionality, such as altering the resulting frequency, duty cycle, etc. Our experience shows that such modifications do not give rise to any problem. The subsequent two classes rely on a similar methodology making possible to understand how interface with peripheral devices can be coded in VHDL and how the respective circuits can be synthesized, implemented and tested.

The second example demonstrates a circuit that interacts with pushbuttons, dip-switches and LEDs through a complex programmable logic device (CPLD) available on the prototyping board. The third circuit (example) enables the students to display characters on the LCD screen (available on the board), which has 2 lines with 16 positions (characters) in each line (see fig. 1). These examples involve lots of new VHDL constructions and design principles, such as:

- New VHDL statements, types and instructions: *variable*, *integer*, *string*, *constant*, *array*, *case*, *if-else*, etc. It is very important that from the very beginning the students understand how to utilize formal language constructions for the design of real circuits and how to test them in a physical environment;
- Interaction between multiple concurrent statements and how the concurrent VHDL descriptions relate to actual FPGA hardware;
- Using sequencers to describe multi-step cyclic operations.

A fundamental advantage of this approach is exhibited in the two following issues: 1) an opportunity to rapidly acquire from scratch an initial (but on the other hand sufficient for this stage) experience in the FPGA-based design; and 2) supplying the circuits that provide all necessary interactions with peripheral components. Our experience has shown that after just three classes the students understand the basic principles of the VHDL code for the considered examples and they are able to modify and to use the designed circuits in future projects.

The second segment can be characterized as *intensive learning of VHDL*, which is achieved through a rational combination of three following subjects: 1) language constructions; 2) coding styles and methodologies; 3) design and implementation of FPGA-based digital circuits from VHDL specifications. Since the time available for this segment is limited, the emphasis is done on the synthesizable VHDL subset for the ISE [4]. Note that there exists a substantial difference between general-purpose programming languages, such as C/C++, and HDLs, which must be properly understood for hardware design. This issue is a key point, which has been addressed.

The third segment is devoted to *simulation*, which is indispensable for FPGA-based design. It is carried out with the aid of the STARTER version of ModelSIM [8] (see fig. 1). It is important that this version, as well as the WebPACK ISE [4], is a freeware. The relevant practical classes consider simulation issues for the circuits proposed in the previous segment. In fact, the classes for the second and the third segments are given at the same time. The basic principles of the ModelSim environment are explained immediately after the first segment. This enables the students to use simulation for testing different VHDL constructions. As soon as this is needed, more complex simulation scenarios are given to students and the succeeding practical classes permit to invoke these scenarios in the respective design examples.

The fourth segment is devoted to some supplementary CAD tools that are very important and in many cases enable the students to speed up the design process significantly. Note that a typical digital system (circuit) can be decomposed in sub-systems (sub-circuits) in such a way that for the design of each sub-system (sub-circuit) the most efficient and the most appropriate CAD tools can be used. Three types of such tools are lectured to the students (see fig. 1):

- Synthesis and language templates;
- IP core generators;
- Synthesizers from graphical FSM (finite state machine) specifications.

In addition, advanced synchronization techniques are considered based on the blocks DLL (delay locked loop) and DCM (digital clock management).

The synthesis and language templates supply VHDL code for: 1) typical language statements, declarations and constructions; 2) instantiation of FPGA embedded blocks, such as block RAM; and 3) reusable digital circuits, such as counters, shift registers, etc. The basic templates can be taken from the ISE, whereas the others (such as hierarchical FSMs) have been designed by the authors.

An IP core generator is also available for the ISE and it enables the students to construct complex blocks such as those that are used for mathematical computations, application-specific memories, etc. In order to provide for synthesis from FSM state transition diagrams (including their extended forms) the StateCAD software available from the ISE [4] has been used.

Note that when the fourth segment is being considered the students have already acquired sufficient experience and they are able to design rather complicated circuits. In order to simplify the design process several projects (VHDL specifications) are given to the students, namely an interface block with a VGA monitor, a controller of an external static RAM, and a controller of the serial port RS232. These circuits have to be understood, analyzed and used in future work. At this, time two project are proposed to the students.

The first project is the design of a simple processor, which includes a memory, a datapath and a control unit. A set of the processor instructions is given and the memory structure is defined. There are some other requirements: 1) the memory has to be constructed using the IP core generator; 2) the control unit has to be designed in the *StateCAD* environment; and 3) the project has to use language and synthesis templates.

The second project is devoted to the design of a combinatorial processor that is able to solve different optimization problems over simple Boolean and ternary matrices (typically with the size 16x16), such as: finding a minimal row cover for a given matrix, solving the SAT problem, etc. It is allowed to use any available ISE tool.

Note that the projects listed above are quite complicated. However, they can be constructed from blocks many of which have been given to the students. Indeed, the results can be displayed on either a VGA monitor screen or an LCD display; storage can be organized in external static RAM (or alternatively in FPGA block RAM); initial data can be entered from a host computer through the RS232 interface, etc. Such opportunities make possible to simplify the design problem significantly. Our experience shows that the majority of the students are able to complete the listed above projects.

Finally, to conclude the course on FPGA-based systems some recent innovations in the design of reconfigurable systems are exemplified. They include:

- System-level specification languages (on an example of Handel-C language and Celoxica DK3 design suite);
- Dynamic reconfiguration and design of virtual systems.

Handel-C permits to describe digital circuits in C-based style, to synthesize and to implement them in hardware with the aid of the design tools developed and supplied by Celoxica [7]. The design flow includes the following basic steps: specification in Handel-C; verification and validation of the description in software; synthesis of hardware system. The result is an EDIF file (electronic design interchange format) that can be used for mapping, placement, routing and generation of FPGA bit-streams with the aid of commercially available CAD systems, such as Xilinx ISE. At the final step the bit-stream can be loaded to the FPGA. A few complete examples (such as that are discussed in [3]) are demonstrated. Handel-C code for the examples can be downloaded from [9,10]. Thus, the most inquisitive students are able to understand the idea of system-level specification languages and, if required, to learn them afterwards without assistance for using in future projects. It is very important that Handel-C enables the students designing very complex circuits within a limited time slot. Thus, the complexity of future projects can be increased considerably (some Handel-C student projects are available online at [9]).

In order to demonstrate capabilities of dynamic reconfiguration the following example is discussed. There are many digital systems that require executing operations over Boolean and ternary vectors of arbitrary size [11]. On the one hand, the number of feasible operations on such vectors is practically infinite. On the other hand, each particular application will typically only require a very limited number of such operations. Thus, it is rational to construct a reusable circuit that can perform a restricted number of operations over the vectors being considered, but where these operations can be customized for an unlimited (or at least very large) number of specific applications. This circuit can be designed using either a static technique (i.e. the set of operations is defined through loading the FPGA bit-stream) or dynamic reconfiguration (i.e. the set of operations can be changed during run-time with the aid of a reconfiguration handler). If an FPGA-based circuit is able to provide changes during the execution of the algorithm then a virtual FPGA-based system can be constructed. This, in particular,

enables a system to be implemented on an FPGA that does not have sufficient hardware resources to accommodate all the necessary functionality.

Finally, all the course topics are presented. Note that from the beginning the students are not so successful and they have many difficulties. However, at the end the majority of the students are well prepared for practical design of FPGA-based systems. Additional assistance is provided with the aid of the tools considered in the next section.

In section 1 it was mentioned that there exists another problem in teaching FPGA-based systems, which is the necessity of periodic update of laboratory equipment. This problem can be simplified through the construction of a simple extendable FPGA-based board with a reusable FPGA socket in such a way that the board can be updated by replacing the currently installed FPGA with a more advanced FPGA potentially available in the future. Such a board should include an FPGA, a flash memory for storing (one or more) FPGA bit-stream(s) and a USB interface providing downloading FPGA bit-streams. Extensibility can be achieved through expansion headers enabling the students to connect any required peripheral device.

4. CAL TOOLS

Computer assisted learning (CAL) is very important because it permits to shorten the time of understanding and mastering many difficult topics. The developed CAL tools for disciplines on FPGA-based systems include (all these tools are available online at [9,10]):

- Animated tutorials and reference guides;
- Simple projects, which explain different typical constructions and coding styles;
- Course-oriented templates and libraries;
- A set of projects and publications of students, which permit the previous experience to be accumulated and reused in future projects.

The designed tutorials [3] cover the following topics:

- An introduction to Xilinx ISE explaining all the steps for the first example considered in section 3;
- Interaction with external devices such as pushbuttons, switches and LEDs available on the TE-XC2Se board [5];
- Xilinx ISE (project hierarchy, schematic editor, IP core generator, VHDL);
- FSM specification, synthesis and simulation in the StateCAD environment;
- VHDL topics in alphabetical order;
- Simulation with ModelSim;
- Interaction with textual LCDs through the controllers HD44780U and KS0073;
- Design of reprogrammable finite state machines;
- Design of hierarchical finite state machines and demonstrating their advantages on examples of recursive and modular algorithms;
- DKx design suite of Celoxica; ISE 6.x of Xilinx for mapping, placement and routing; modeling examples in C++; RC100 prototyping board of Celoxica for experiments.

Simple projects are provided for all the tutorials and they demonstrate how to use all the explained language constructions and the considered ISE and DK scenarios.

Course-oriented templates and libraries enable the students to use for their projects some preliminary designed and verified blocks. As a rule, a template is a piece of VHDL code that can be inserted into any project to supply the required functionality, such as conversion of binary codes to BCD codes, traversing binary trees, etc. The course-oriented library contains components that are required to input and output data and provide support for:

- Output data to a VGA monitor;
- Interaction with a mouse;
- Input data from a keyboard;
- Output data to textual LCDs, externally connected through an expansion header;
- Interaction with a touch panel and a graphical LCD;
- Data exchange through the RS232 interface;
- Data exchange with external static RAM.

This library is periodically updated. For example, the components, which provide for USB interface and data exchange with flash RAM will be available soon.

Finally, the disciplines on FPGA-based design give foundations for a number of other courses and final year projects. Just in 2002/2003 and 2003/2004 two following final year projects have been finished in this direction:

- Combinatorial processor allowing the exact minimal row cover of binary matrices to be discovered (ADM-XPL PCI board containing Xilinx Virtex-II Pro XC2VP7 FPGA);
- FPGA-based accelerator for solving 3-SAT problem.

The following two final year projects are under development in 2004/2005 pedagogical year:

- FPGA-based image processor interacting with an external GPS (global positioning system) device.
- Extended library for data exchange between FPGA-based systems and peripheral hardware (the TE-XC2Se board).

Two of the projects considered above were proposed to the students of the Electronics and Telecommunications curriculum and the remaining two projects were given to the students of Software Engineering curriculum.

5. CONCLUSION

The paper describes the methodology, which has been used in the Department of Electronics and Telecommunications of Aveiro University (Portugal) for teaching FPGA-based systems. The emphasis is done on the course for a specialty that is not directly dedicated to electronics. This is important because the scope of potential applications of FPGAs is very broad and engineers specializing in different directions should be familiar with the respective design methods and tools. Indeed, FPGA technology makes possible to reduce design and implementation time and to shorten significantly time-to-market for systems from numerous application areas. It provides also very helpful prototyping and many other useful facilities. For example, our experience has shown that even undergraduate students are capable to design, implement and test (in hardware) embedded blocks with relatively complex functionality within a few weeks. However, initially these students have to be trained and they have to acquire experience in a number of relevant fields indicated above. The paper has focused on the systems oriented to Xilinx FPGAs, which hold the leading place in the world FPGA production. Obviously, the other commercially-available reconfigurable microchips are also very important but an experience acquired in using Xilinx FPGAs can easily be retargeted to other platforms in the future.

REFERENCES

1. Hauck, S., "The Roles of FPGA's in Reprogrammable Systems", Proceedings of the IEEE, vol. 86, no. 4, pp. 615-638, Apr. 1998.
2. Benini, L. and De Micheli, G., "Networks on Chip: A New SoC paradigm", IEEE Computer, pp. 70-78, Jan. 2002.
3. Sklyarov, V., Skliarova, I., Teaching Reconfigurable Systems: Methods, Tools, Tutorials and Projects, IEEE Transactions on Education, vol. 48, no. 2, May 2005.
4. Xilinx, Inc., Products and Services, Available: <http://www.xilinx.com/>.
5. Spartan-IIIE Development Platform, Available: <http://www.trenz-electronic.de>.
6. XESS Corp., Products, Available: <http://www.xess.com/>.
7. Celoxica, Software Tools and Development Boards, Available: <http://www.celoxica.com/products/default.asp>.
8. ModelSim, Products and Solutions, Available: <http://www.model.com/products/default.asp>.
9. Department of Electronics and Telecommunications, Semester 1, the discipline Sistemas digitais reconfiguráveis, Available at: <http://elearning.ua.pt>, login: *alunosdr*, password: *sistemas*.
10. <http://www.ieeta.pt/~iouliia>.
11. Sklyarov, V., Skliarova, I., Oliveira, A., Ferrari, A., "A Dynamically Reconfigurable Accelerator for Operations over Boolean and Ternary Vectors", Proceedings of EUROMICRO Symposium on Digital Systems Design – DSD'2003, Turkey, Sept. 2003, pp. 222-229.