Remote Reconfigurable Systems for Electronic Engineering and Education

Valery Sklyarov, Iouliia Skliarova, Bruno Pimentel, Manuel Almeida
Department of Electronics, Telecommunications and Informatics/IEETA,
University of Aveiro
3810-193 Aveiro, Portugal
skl@ua.pt, iouliia@ua.pt, pimentel@ieeta.pt, manuel.almeida@ieeta.pt

***Abstract*—Nowadays, reconfigurable systems, in general, and FPGA (field-programmable gate array) based devices, in particular, constitute an essential part of engineering practice. The paper argues the importance of reconfigurable systems in engineering and especially in education and proposes effective learning and design methods and tools, which include laboratory templates, animated tutorials, problem-oriented examples, a remotely reconfigurable prototyping system, and a virtual software/reconfigurable hardware co-simulation environment. The primary objective of these methods and tools is design space exploration, engineering training, and education.**

Keywords-reconfiguration; FPGA; prototyping systems; engineering; education

I. INTRODUCTION

Tremendous progress in the scope of field-programmable gate array (FPGA) technology has made it possible to advance configurable microchips from simple gate arrays that appeared on the market in the mid-1980s to multi-platform FPGAs containing more than 10 million system gates and targeted to the design of very complicated engineering systems. Today, the way to evolve higher performance computing from a general-purpose computer, proposed more than 50 years ago [1], has been finally implemented in reality. The market for FPGAs and other programmable logic devices is expected to grow from $3.2 billion in 2005 to $6.7 billion in 2010, according to Gartner Dataquest [2].

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 [3]. The emphasis is on an extensive use of computer-aided design (CAD) and computer-aided education tools. In fact, the electronic design automation business has profoundly influenced the integrated circuit business and vice-versa [4]. Traditionally, FPGA-targeted CAD systems support schematic and hardware description language-based design flows involving model-specific tools (such as synthesis of finite state machines from a graphical specification) and IP (intellectual property) core generators based on parameterization or templates. Recently, commercial CAD tools allowing digital circuits to be synthesized from system level specification languages (such as Handel-C and SystemC) as well as high-level programming languages (such as C++) have appeared on the market. Thus, the domain of reconfigurable systems design is very dynamic and many-sided. This requires numerous supplementary tools simplifying design space exploration, co-simulation in hardware and software, and computer-aided education targeted to reconfigurable system design and verification.

The remainder of the paper is organized in five sections. Section II presents further arguments to the importance of reconfigurable systems. Section III describes the proposed e-learning tools. Section IV characterizes the developed remotely reconfigurable prototyping system. Section V discusses a virtual visual environment which is very helpful for design space exploration and hardware/software co-simulation. The conclusion is given in Section VI.

II. IMPORTANCE OF RECONFIGURABLE SYSTEMS

Reconfigurable systems possess a number of particularities both in implemented architectures and in functional capabilities. The most important of them are listed below:

- Configurability that makes it possible to consider such systems as soft application-specific integrated circuits, i.e. soft ASICs.
- Reconfigurable devices are hardware circuits that can be designed, physically implemented and tested remotely.
- They introduce a new distinguishing computing paradigm and eliminate the necessity for traditional von Neumann architecture although such architecture can be used if required.
- Parallelism becomes the most important technique allowing system performance to be increased.
- Reconfigurable systems open practically unlimited opportunities for design space exploration and comparison of alternative and competitive solutions.
- They permit any required external interface to be easily established.
- Systems can be composed of novel and pre-designed components and also temporarily embedded circuits for testing and verification. This significantly

simplifies the design process and increases importance of such tools as libraries, templates, IP-cores, and embedded testers.

As a rule, traditional education does not provide sufficient knowledge in the areas listed above. In other words, students after graduating from universities are not capable to apply their engineering acquirements to the design of well structured and optimally organized reconfigurable systems. Due to the great significance of reconfigurable systems for many industrial areas, tending to be significantly increased in the future, the importance of the relevant topics in education is evident.

III. E-LEARNING TOOLS

The developed e-learning tools are organized as a set of laboratory templates; tutorials, problem-oriented examples and supplementary materials.

A laboratory template is a Web page, which enables the students/engineer to select properly all the required components of a project and additional materials that might be useful. An example of such template will be considered at the end of this section.

Tutorials are divided into the following groups:

- Design scenarios;
- Specification, synthesis and implementation of FPGA-based circuits from hardware description languages;
- Standard interfaces with typical peripheral devices;
- Design methods targeted to reconfigurable systems.

Problem-oriented examples demonstrate how to construct reconfigurable circuits based on:

- Reusable hardware description language fragments (RHDLF);
- Templates;
- Demonstration resources organized as education-targeted libraries.

RHDLFs are pieces of easily customized hardware description language (HDL) code that can be inserted into more complicated HDL-based projects. For example, RHDLFs might be specifications of typical operations, such as interface with a keyboard or binary to BCD converters.

Templates are complete customizable specifications for such circuits as finite state machines (FSMs), hierarchical FSMs, parallel FSMs, etc.

Demonstration resources are composed of software programs (that enable the student to understand easier and better different tasks), typical projects and scenarios (illustrating the required sequence of steps and the expectable output or results). The primary objective of problem-oriented examples is to explain how to develop real-world projects composed of design cores worked out by the students and supplied components simplifying the project and shortening the development lead-time. The examples are organized in such a way that enables to provide sufficient knowledge within the limited time common, for example, for laboratory classes in universities.

Let us consider a complete example showing how the developed e-learning tools can be used for the design of hardware circuits allowing recursive sorting based on binary trees to be implemented. Suppose, initial data have to be supplied from a keyboard and the results have to be displayed on a VGA monitor (see Fig. 1). Obviously, other suitable peripheral devices might be chosen.

At the top level the project is divided in two parts:

1) Hardware circuits for interaction with peripheral devices;
2) Hardware circuits for recursive sorting.

For the first part, RHDLFs from electronic library can be applied directly. However, this is not just a library, because it includes:

- RHDLFs that can be used if and only if they are properly understood (see the next point);
- Tutorials, explaining how to interact with the chosen peripheral devices and enabling the students to understand RHDLFs;
- Typical examples, such as reading data from a keyboard and displaying data on a VGA monitor. Thus, the students can simulate HDL code and execute projects avoiding any doubt and misunderstanding.

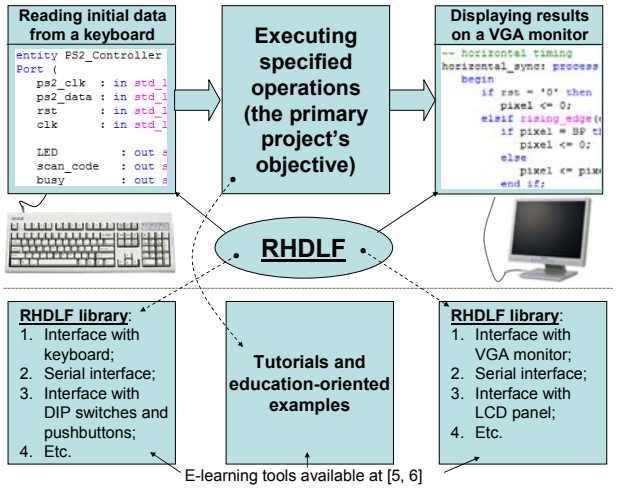

Figure 1. An example of a project demonstrating which e-learning tools might be used

The second part is more complicated and it requires:

1) Studying the proposed recursive algorithm;
2) Understanding how to implement recursion in hardware;
3) Developing hardware-oriented recursive algorithm;
4) Describing the project in HDL (in VHDL);
5) Providing proper interactions with the peripheral input/output devices.

Fig. 2 depicts the developed e-learning tools available online [5, 6] for such purposes. The student begins with point 1 above. As an example, a fragment of C++ program (taken from [7]) is presented below:

```
// Structure to code a binary tree node
template <class T> struct treenode {
     T val;       //value of an item of type T
     int count;   //number of items with value val
     treenode <T> *lnode; //pointer to left node
     treenode<T> *rnode;  //pointer to right node
   };
//Function template to create a root or add a node
template<class T>
treenode<T> *addnode(treenode<T> *node, T value)
{
  if(node == 0)
  { //constructing a new node
    if((node = new treenode<T>) == 0)
      // exception handling or exit
      node->val = value; //store new value in node
      node->count = 1;   //only one to start
      node->rnode = node->lnode = 0;
  }
  else if(value == node->val)
    node->count++;   //increment the counter
  else if(value < node->val)
    //construct the left node
    node->lnode = addnode(node->lnode,value);
  else
    //construct the right node
    node->rnode = addnode(node->rnode,value);
  //return a pointer to the current node
  return node;
}
```

There are two recursive functions in the program. The first one, shown above, permits to construct a binary tree composed of nodes of type `treenode`. The second function, shown in Fig. 2 (see the right-top corner), sorts data based on the binary tree (all necessary details are considered in [7]).

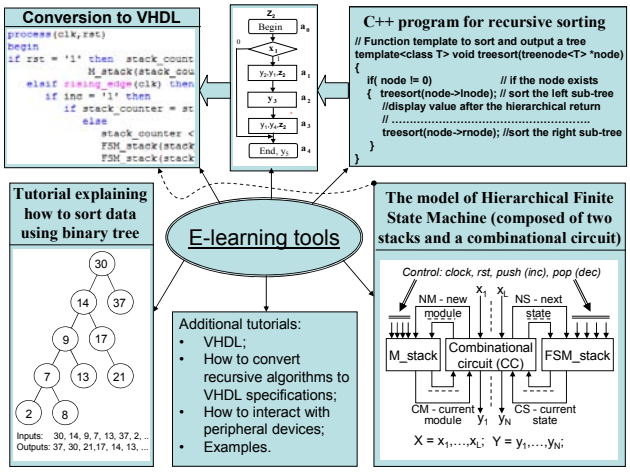

Figure 2. E-learning tools assisting in the development of the project

The student can test various scenarios in software and clearly understand the problem. There is also a tutorial available at [5] which explains the sorting algorithm in an animated PowerPoint presentation (see the bottom-left part of Fig. 2).

At the step 2, indicated above, the students have to understand how to implement recursion in hardware (note that neither hardware description nor system-level specification languages provide support for recursion). There is a tutorial available at [5] which explains how to implement recursive algorithms using the model of a hierarchical finite state machine (HFSM) and gives a VHDL template for HFSM. The latter is VHDL code that can easily be customized to realize any required recursive algorithm.

At the step 3 the students have to develop a hardware-oriented recursive data sorting algorithm. In other words, it is necessary to analyze the given C++ program and to construct flowcharts, which can be formally converted to the relevant control sequence produced by HFSM (see the top-middle and top-left parts of Fig. 2). There are examples available at [5] demonstrating how to perform such conversion. The VHDL template for the combinational circuit of HFSM is given and it looks like the following:

```
process (current_module,current_state,inputs)
begin  -- describing combinational process of HFSM
  case M_stack(stack_pointer) is
    when z0 =>
      case FSM_stack(stack_pointer) is
        -- state transitions in the module z0
        -- generating outputs for the module z0
      end case;
    when z1 =>
      case FSM_stack(stack_pointer) is
         -- state transitions in the module z1
         -- generating outputs for the module z1
      end case;
    -- repeating for all modules
end process;
```

On the one hand, the code above is simple, but, on the other hand, it is sufficient to describe the algorithms and further to synthesize the relevant circuit and to implement it in hardware. Indeed, the students have to describe just state transitions within each module (such as `z0` and `z1` above) and transitions between the modules. *`Stack_pointer`* is a common to both stacks (i.e. the stack of modules and the stack of states) pointer indicating the current hierarchical level (all the required details can be found in [7]). VHDL description of the stacks is fully reusable and it is given. Thus, the students have to customize just the combinational circuit shown above in accordance with the flowchart (see the middle-top part in Fig. 2) constructed from the relevant C++ function (see the right-top part of Fig. 2).

At the next step (4) it is necessary to describe the remaining components of the project in HDL (in VHDL, in particular). There are many e-learning materials, which provide very valuable assistance. These materials are listed below:

- Tutorial (available at [5, 6]) on VHDL constructions and keywords that simplifies the correct use of VHDL;
- Tutorials (available at [5, 6]) on computer-aided design tools demonstrating numerous scenarios, which permit to understand: how to simulate VHDL code; how to

synthesize circuits from VHDL specification; how to use IP core generator, etc.

The last step (5) permits to finish the project with the aid of e-learning components, which are very helpful, such as:

- Tutorials (available at [5, 6]) demonstrating how to link the developed circuits with the circuits providing input/output interface (see Fig.1);
- Projects (available at [5]) making it possible to develop many circuits for recursive sorting by analogy.

As can be seen from the previous description there are many components (tutorials, projects, HDL specifications, etc.) that might be used for a particular project. Laboratory templates enable the students to concentrate only on necessary for a particular project components abstracting from the other available components.

Let us consider an example of a laboratory template for the described above project (see Fig. 3). The template is an available through the Internet interactive picture with hyperlinks enabling the students to find quickly all the required components of a particular project. For the project shown in Fig. 1 it is necessary to carry out the following operations:

1) Click on the keyboard image in order to get HDL code for interacting with a keyboard and also all additional materials (tutorials, examples, etc.);

2) Click on the VGA monitor image in order to get HDL code for interacting with a VGA monitor and also all additional materials (tutorials, examples, etc.);

3) Click on the proposed for laboratory work problem;

4) Click on an option to get the required details and assistance (i.e. problem description – to find the description of the laboratory work, software model – to find the program that executes sorting, etc.).

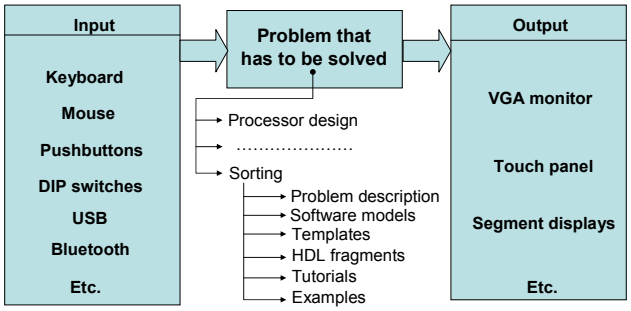

Figure 3. An example of a laboratory template

Such technique is very useful for engineering training and education.

IV. Reconfigurable Systems with Wired and Wireless Interactions

In order to further assist in educational process we have developed an FPGA-based prototyping system [8]. The architecture of the developed prototyping system is illustrated in Fig. 4.

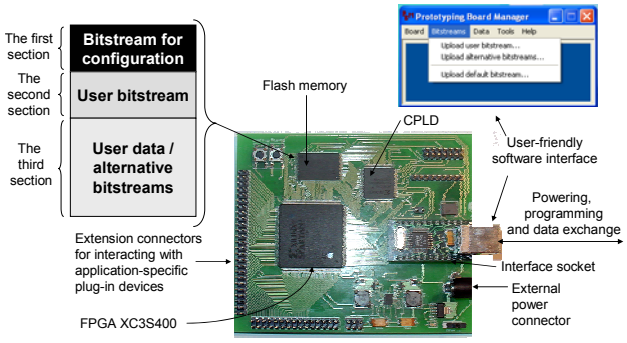

Figure 4. Architecture of DETIUA-S3 prototyping system

The basic hardware/software components of the system are the following [8]:

- XC3S400-4-PQ208 FPGA of Spartan-3 family with 400.000 system gates [9];
- XC9572XL CPLD which is used for FPGA reconfiguration [9];
- 16 Megabit Am29LV160D flash memory, which is divided in three logical sections. 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;
- 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 developed FPGA-based prototyping system [8] possesses the following distinctive features:

- The core FPGA can be configured using wired (USB) and wireless (Bluetooth) interfaces;
- The developed software/hardware components provide support for: 1) wired reconfiguration/interaction through USB interface; 2) autonomous reconfiguration from onboard flash memory; and 3) remote wireless reconfiguration/interaction through Bluetooth interface;
- The design process is supported by numerous developed tools, such as templates, design libraries, IP cores, etc. (see the previous section). Many of them are oriented to remote control and reconfigurability;
- The remote interaction capabilities are supported by the developed software making it possible to verify different projects virtually and remotely in such a way that FPGA-based hardware is considered to be a remotely accessible component and the required functionality is implemented both in physical hardware (FPGA-based prototyping board) and in software supporting visual interface for virtual verification of the developed system (this feature will be considered in the next section in more detail).

V. Virtual Visual Environment for Engineering and Education

Fig. 5 demonstrates the basic idea of a virtual visual environment. As a rule, circuits implemented in FPGAs are parts of larger circuits making up the designed system. The latter might be composed of other circuits, sensors, actuators, electromechanical devices, etc.

In general, the virtual visual environment consists of the following three primary parts (see Fig. 5):

1) Virtual devices implemented in a host computer. They are virtual devices because they are implemented in software and provide such functionality that is very similar to physical devices. They are visual because we are able to observe the functionality (such as different motions of mechanical elements, states of electronic components, etc.) in visual mode on a monitor screen (or possibly in some other connected peripheral devices). They are easily controllable because we can carry out numerous functional and timing scenarios, for example, test just the selected fragments of implemented algorithms, execute algorithmic steps faster or slower, etc.

2) Physical devices implemented in FPGA and interacting with virtual devices in such a way that allows to make up the designed system, i.e.: *physical devices + virtual devices = the designed system*. Such system is flexible and extendable, because functionality of both software and reconfigurable hardware can be altered.

3) Hidden from the end user software/reconfigurable hardware interface providing interaction between the virtual and physical devices.

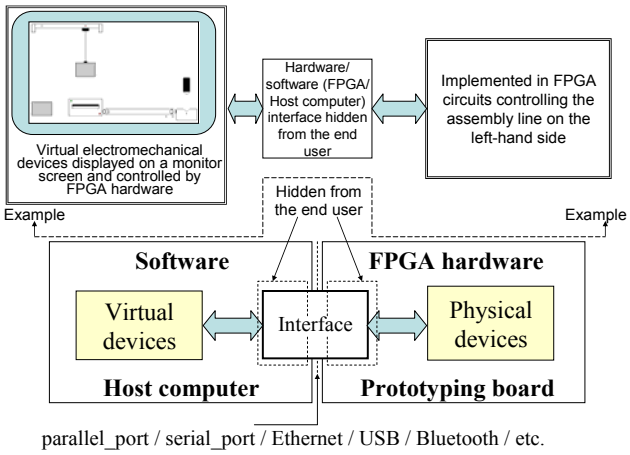

Figure 5. Visual virtual environment with an example

An example is given in the upper part of Fig. 5 where an FPGA on the right-hand side controls an assembly line on the left-hand side and the respective interaction is established through an *FPGA/host computer* interface. Another example can be taken from the area of embedded systems [10]. Suppose it is necessary to design a device controlled by a sequence of instructions such as "switch on/off", "set in a given state", "reset", etc. These instructions affect the behavior of an execution unit via actuators, and the sequence of the instructions depends on the states of the execution unit sensors. The actuators and sensors can be electronic, optical, mechanical, etc. The proposed technique permits:

- To verify the embedded system entirely in software;
- To implement the embedded system partially in software and partially in hardware;
- To carry out hardware/software co-simulation with fuzzy boundary between hardware and software (i.e. to analyze the embedded systems with either more software and less hardware or vice versa).

This task is very interesting and useful for designers dealing with different problems from vast variety of application areas, such as:

- The mentioned above assembly lines (see upper part of Fig. 5) and embedded systems;
- Problem-oriented computations, such as solving combinatorial search problems. In this particular case, the execution unit of the considered processing block can be modeled in hardware and the control algorithms can be implemented in software;

- Design of electromechanical peripheral devices, such as plotters (see examples in [10]). In this case electromechanical part can be modeled in software and electronic (logic) part can be implemented in hardware.

It is important that the virtual visual environment makes it possible to create vast variety of virtual peripheral devices for the FPGA-based prototyping core considered in section IV. Indeed, the presented above basic idea allowing real-world projects to be implemented, is the division of the project into core components, developed by the students, and supplied components, given to the students (see Fig. 1 and 3). The latter are mainly interface circuits for peripheral devices providing data input and output. The proposed technique allows implementing all these devices virtually in such a way that the system displays virtual peripheral devices and communications with such devices are organized much like communications with physical peripheral devices.

The proposed technique can be very efficiently used in the scope of design space exploration. For example, there exist a large number of practical applications that require solving combinatorial search problems, such as Boolean satisfiability, binary matrix covering, graph coloring, etc. It is possible to design a reusable circuit that might be customized for solving any problem listed above and perhaps many other problems from the area of combinatorial computations [11]. Such a reusable circuit can be entirely modeled in software. However, it may be beneficial to implement this circuit in FPGA. One of the easiest ways is a sequential conversion of the software model to hardware implementation in such a way that the required hardware is incrementally extended replacing more and more software. This, in particular, significantly simplifies testing and debugging of hardware circuits. It is also very appropriate for student laboratory works because it permits to begin and to finish an entire design step within a laboratory class. Besides, the considered hardware/software model opens practically unlimited capabilities for engineers in the scope of design space exploration. For example, the following opportunities can be used:

- Software debugging facilities allowing to test different operations available for the circuit;
- Verifying different algorithms step by step either in software or in hardware appreciating the results in a convenient visual mode;
- Providing hardware/software partitioning and executing algorithms partially in software and partially in hardware changing the boundary between software and hardware. This is indeed design space exploration because the designers can check if hardware implementation is really capable to improve performance and other characteristics of the system.

VI. Conclusion

The paper argues the importance of reconfigurable systems in engineering design and education and describes the developed e-learning tools, the prototyping system, and the virtual visual environment that make it possible to increase the efficiency of design and productivity of designers. E-learning tools are organized as a set of laboratory templates, tutorials, problem-oriented examples, and supplementary materials. The majority of these tools are available online through the Internet [5,6]. The FPGA-based prototyping system with wired and wireless interfaces has been designed, implemented, tested, and manufactured. Finally, the developed virtual visual environment provides significant assistance for design space exploration and hardware/software co-simulation.

References

[1] G. Estrin, “Organization of Computer Systems – The Fixed Plus Variable Structure Computer”, *Proc. Western Joint Computer Conf.*, New York, 1960, pp. 33-40.

[2] B. Lewis, “Market Trends: ASIC and FPGA, Worldwide, 2004-2010”, Gartner Dataquest, 2006.

[3] V. Sklyarov, I. Skliarova, “Teaching Reconfigurable Systems: Methods, Tools, Tutorials, and Projects”, *IEEE Trans. on Education*, vol. 48, no. 2, 2005, pp. 290-300.

[4] D. MacMillen, M. Butts, R. Camposano, D. Hill, T. W. Williams, “An Industrial View of Electronic Design Automation”, *IEEE Trans. on Computer Aided Design of Integrated Circuits and Systems*, vol. 19, no. 12, Dec. 2000, pp. 1428-1448.

[5] E-learning tools and materials of courses on reconfigurable computing. Online: http://www.ieeta.pt/~skl/.

[6] E-learning tools and materials of courses on reconfigurable computing. Online: http://www.ieeta.pt/~iouliia/.

[7] V. Sklyarov, “FPGA-based implementation of recursive algorithms”, *Microprocessors and Microsystems, Special Issue on FPGAs: Applications and Designs*, vol. 28/5-6, 2004, pp. 197-211.

[8] M. Almeida, B. Pimentel, V. Sklyarov, I. Skliarova, "Design Tools for Rapid Prototyping of Embedded Controllers", Proceedings of the 3rd International Conference on Autonomous Robots and Agents - ICARA'2006, Palmerston North, New Zealand, December 2006, pp. 683-688.

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

[10] V. Sklyarov, “Hardware/Software Modeling of FPGA-based Systems”, *Parallel Algorithms and Applications*, vol. 17, no. 1, 2002, pp. 19-39.

[11] I. Skliarova, V. Sklyarov, "Design Methods for FPGA-based Implementation of Combinatorial Search Algorithms", in *Proc. Int. Workshop on SoC and MCSoC Design - IWSOC'2006*, *4th Int. Conf. on Advances in Mobile Computing and Multimedia - MoMM'2006*, Yogyakarta, Indonesia, December 2006, pp. 359-368.