Proceedings of the E2LP Workshop
Warsaw, 2014, pp. 9–12

DOI: 10.15439/2014F673
ACSIS, Vol. 4

FPGA Verification Module

Željko Hocenski, Ivan Aleksi
University of Osijek, Faculty of Electrical Engineering
Zeljko.Hocenski@etfos.hr, Ivan.Aleksi@etfos.hr

***Abstract*: This paper addresses verification and debugging tool for development of FPGA modules. Proposed tool is developed for educational purposes in teaching students on Digital Design and VHDL programming language. Main goal of the debugging module is to get/set signal values while the FPGA board is running the module of interest. Two PicoBlaze CPUs are used in order to synchronize the input and output signals between PC and the FPGA. Debugging and verification tool is wrapped around the testing module, and it occupies 14% of the Spartan 3 XC3S200 FPGA device. While using proposed tool, students are getting the knowledge about the PicoBlaze CPU, assembly language, FPGA, VHDL. When using proposed tool, students get deeper understanding of the hardware-software co-design concept. Finally, individual tasks are assigned to student workgroups. Some typical tasks are illustrated in this paper.**

Keywords: FPGA, PicoBlaze CPU, Verification, Debugging.

I. Introduction

FPGA debugging module is proposed in this work. HDL modules debugging is a critical stage in the development of FPGA modules [1]. Verification and validation takes 50% of the product's cost, and it takes 40% or more of the overall design cycle [2]. In addition to shorten design cycle for students on their HDL FPGA project, this paper proposes a debugging platform for FPGA module in the development phase, as it was done in [3]. Proposed platform extends the number of the input and the output ports by using one or more input or output buffers. The platform consists of the FPGA-PC communication and a PC-based graphical user interface (GUI). In this way, students can have an insight to the signals of FPGA module in the development phase. Signal values are placed in buffers on the FPGA and on the PC's GUI. Each buffer can be set as an input or an output. Buffers values on the FPGA and their corresponding copy on the PC are synchronized by the use of two PicoBlaze CPUs and UART communication module {[4],[5]}, c.f. Fig. 1. Usually, development kits have a limited number of I/O devices: buttons, switches, LEDs, etc. On the other hand, proposed debugging tool enables arbitrary large number of I/O devices.

II. RELATED WORK

Similar work is done in [6] where authors presented their Reconfigurable Virtual Instrumentation (RVI). By the cross-university collaboration virtual devices are developed and shared. In this way the RVI becomes an evolvable low-cost hardware-software co-design (HW/SW) educational platform. This platform saves money since it can replace some of the expensive measurement instrumentation, for example: oscilloscope, logic analyzer, etc. In contrast to their grooving variations of virtual devices, we are proposing simple and

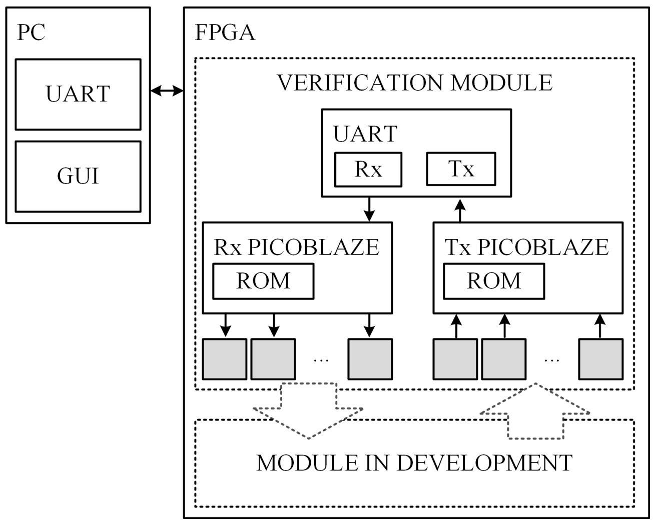


Fig. 1. Virtual devices platform (VDP) with input and output virtual devices (VD)s for functional verification phase of the FPGA modules development.

small VDP that is oriented towards verification and validation of HDL modules via the FPGA-PC interface.

Simple FPGA e-Lab is presented in [7]. By using Windows XP Remote Desktop Connection students can remotely connect to a laboratory PC, which is connected with the Spartan 3-E FPGA device. User interface is done over a GUI that is made with National Instrument's LabView software. Authors made control hardware surrounding the FPGA kit. They also connected a web camera for visual feedback, so one can always see if his experiment is running or not. Students are able to upload final project done at home and implement the bit file into the FPGA device. More sophisticated remote learning FPGA facility is presented in [8]. As an approach it is cost effective when compared with classical non-remote laboratories. Authors considered a cluster of FPGA devices that students can use remotely. Students liked that they can connect later and try again if their project didn't work from the first try. However, {[7],[8]} do not offer system verification and validation, as it is presented in this work. Our device is done in MS Visual Basic and does not require the expensive software tools.

According to [9], FPGAs are the key components for making possible to finish complex projects in a one year period. Students were assigned a task to build mobile robot projects. Before taking the FPGA into education courses, students were focused only on a software design of bought professional robotics equipment. The author concluded that undergraduate students are able to construct competitive robots with the use of the FPGA and a basic robot parts.

The authors in [10] prefer teaching real-time digital signal processing by using FPGAs. They believe that the FPGA technology provides more flexibility when compared with the DSP or the MATLAB. The FPGAs are capable for implementing the microprocessor cores, what exclude the need for a certain DSP that is programmable in an embedded C. With the use of Hardware/Software co-design, the FPGAs allow algorithm improvements by using its hardware and/or even parallel implementation. In reference [11], the same author presented the FPGA-based system on programmable chip (SoPC) development platform with NIOS 32-bit RISC soft processor and the µClinux operating system (OS). Students had to implement a VGS and a SRAM interfaces with the SoPC. On the software part of the design, students had to develop routines that control a custom graphical hardware. When combining those two components, the students were exposed to hardware/software co-design, operating systems, reconfigurable hardware and system design concept. The authors concluded that the use of the FPGA-based SoPC in student courses enabled the development of more involved senior design projects.

Authors in [12] presented their work through three consecutive courses in one academic year. Their 1st course deals with logic design using Verilog, followed by 2nd course that deals with logic synthesys and system on chip design, while 3rd course deals with timing and testing of digital design. The topics covered with proposed courses are: digital electronic, Boolean algebra, Karnaugh maps, finite state machine, Verilog HDL, 8-bit RISC SoPC, HW/SW co-design, testing of digital systems, test economics, fault modeling and simulation, sequential test methods, and design-for-test techniques. Since none of the topics deals with the Digital Design Verification, the authors announced a new course with that topic.

Distance laboratory access to the server with dedicated FPGA hardware is presented in [13]. The mainstream of this concept is that students can register, login and reserve a term in order to access the hardware resources. When their term is active, they can send their CAD design and the server's software can implement it on the FPGA. After the execution, the student receives an e-mail with the results. This design concept saves a lot of time to the professors and it enables an individual self learning process.

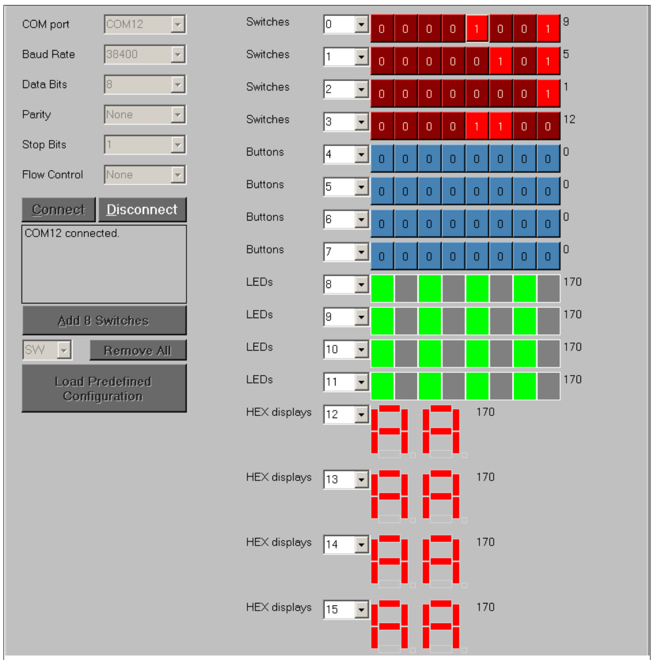
A multimedia tool for teaching reconfigurable computing is presented in [14]. In order to get familiar with the VHDL and FPGA, in the first few weeks, students had to implement a few medium complexity VHDL modules: VGA, PS/2, LCD, calculator and a simple processor. Their knowledge about computer architecture is used to speed up the learning process about the reconfigurable computing. Behavioral VHDL model of a MIPS processor is used in order to instantiate a MIPS microprocessor with some parts missing. Students have to recover the missing VHDL parts (ALU, register file, etc.) and test the functionality of their MIPS using the iCmips simulator. Simulator provides a full view into the registers, data memory, while it executes all of the commands on the real hardware.

Remote laboratory for HW/SW co-design that is proposed in [15] accepts a student's HW and SW design files, implements the HDL file on the real FPGA and executes the software test bench file. As the result, the output file is then sent back to the student. The task was to implement a simple 16-bit RISC processor with Harvard architecture, different memories for machine code and storage data and four general purpose register. The CPU design had to execute nine basic instructions: load, store, move, add, sub, compare, halt, conditional and unconditional jumps.

III. FPGA-BASED HARDWARE/SOFTWARE CO-DESIGN

Common system design approach in embedded systems is a Hardware/Software co-design concept [2]. FPGA devices are capable to synthesize such design concept since they possess a large number of programmable logic cells, which can be configured as combinatorial and sequential, respectively. Hardware/Software co-design commonly consists of programmable devices, microcontrollers, microprocessors or signal processors, which have a memory for a software part of the design, c.f. Fig. 1.

Proposed hardware design uses 14% of the Xilinx's Spartan-3 XC3S200 FPGA device, c.f. Table I. It contains of two of the PicoBlaze CPU's, where each occupies 5%. Remaining 4% are used by UART and some signal interchange. Minimum period of implemented logic is 7.922ns, and the maximum operating frequency of VDP is 126.239(MHz). Proposed design consumes 48(mW). Utilization summary for Xilinx's 3s200ft256-5 device is illustrated in Table I.

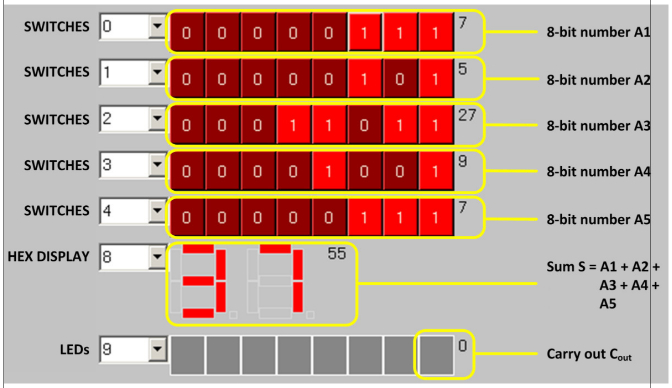


COM port: COM12
Baud Rate: 38400
Data Bits: 8
Parity: None
Stop Bits: 1
Flow Control: None

COM12 connected.

Switches	0	0	0	0	0	1	0	0	1	9
Switches	1	0	0	0	0	0	1	0	1	5
Switches	2	0	0	0	0	0	0	0	1	1
Switches	3	0	0	0	0	1	1	0	0	12
Buttons	4	0	0	0	0	0	0	0	0	0
Buttons	5	0	0	0	0	0	0	0	0	0
Buttons	6	0	0	0	0	0	0	0	0	0
Buttons	7	0	0	0	0	0	0	0	0	0
LEDs	8									170
LEDs	9									170
LEDs	10									170
LEDs	11									170
HEX displays	12									170
HEX displays	13									170
HEX displays	14									170
HEX displays	15									170

Fig. 2. PC-based graphical user interface for validation and verification phase of FPGA modules development.



SWITCHES	0	0	0	0	0	0	1	1	1	7	8-bit number A1
SWITCHES	1	0	0	0	0	0	1	0	1	5	8-bit number A2
SWITCHES	2	0	0	0	1	1	0	1	1	27	8-bit number A3
SWITCHES	3	0	0	0	0	1	0	0	1	9	8-bit number A4
SWITCHES	4	0	0	0	0	0	1	1	1	7	8-bit number A5
HEX DISPLAY	8									55	Sum S = A1 + A2 + A3 + A4 + A5
LEDs	9									0	Carry out C_{out}

Fig. 3. Example application for applying functional test on an Array Adder.

TABLE I. UTILIZATION SUMMARY FOR XILINX 3S200FT256-5.

Resource	Number of resources	[%]
Number of Slices	282 out of 1920	14 %
Number of Slice Flip Flops	323 out of 3840	8 %
Number of 4 input LUTs	521 out of 3840	13 %

Verification is done with an arbitrary large set of input and output buffers that enables a real-time insight into internal FPGA signals. Each buffer is 8-bit long with assigned value and unique address. Buffers exist on FPGA and have corresponding copy on the PC. FPGA and PC buffer values are synchronized by using two of the PicoBlaze CPU's which communicate with the UART-based full-duplex RS232 communication protocol @ 38400 baud rate. One PicoBlaze CPU is used for data transmission, while the other one is used for data reception. Software algorithms for buffer array synchronization are described with Alg. 1 and Alg. 2.

Algorithm 1. Reception PicoBlaze (PC to FPGA).

```
1. byte  GetByte()
2. Set data  byte
3. byte  GetByte()
4. Set address  byte
5. Set BufferArray[address]  data
6. Go to Line 1
```

Algorithm 2. Transmission PicoBlaze (FPGA to PC).

```
1. For each address = 8 to address = 16 with step +1
2.     Set data  BufferArray[address]
3.     SendByte(data)
4.     SendByte(address)
5. Go to Line 1
```

Buffer values on the PC are displayed with the GUI, c.f. Fig. 2. PC-based GUI is implemented in the Visual Basic programming language on a standard PC. It represents the software part of the design that communicates with the FPGA over the RS2323 serial port. USB to RS232 adapter is used in this work. The GUI is made by using dynamical programming techniques with the following features.

- Management of serial port communication;
- Dynamic buffer creation and deletion;
- Selection of arbitrary address for a certain buffer;
- Data transmission from the PC to FPGA's input buffers;
- Data reception from FPGA's output buffers to the PC;
- Displaying buffer values on the screen.

Proposed debugging and verification platform is adaptive since it can dynamically create/delete buffers with arbitrary address and buffer's type: switches, buttons, LED and HEX displays, c.f. Fig. 2.

IV. EXPERIMENTAL RESULTS

In this section, one Array Adder is considered as an FPGA module in the development phase. In this work we used the array adder with 5 8-bit inputs, one 8-bit output and one bit output, c.f. Fig. 3. Array adder has a task to sum 5 8-bit numbers and provide a result as an 8-bit sum with 1-bit carry out. Firstly, the VHDL code was successfully tested with simulations. Subsequently, proposed debugging module was used to test the design's functionality. After successful synthesis, students are very happy to be able to see their design doing required task on an FPGA board.

V. CONCLUSION

Proposed debugging platform is the simple device that helps students to accomplish their final project successfully. It is easy to use and will have a lot of applications on future testing of student projects. It provides the student with extended I/O devices via FPGA-PC interface and makes easier to check does the student's product meet the requirements of a given project by using proposed GUI. Proposed approach is quite challenging for students since it includes multidisciplinary tasks: hardware design, embedded assembly software and verification process. Therefore it can be applied in the final years of their study.

REFERENCES

[1] M. Pezzé and M. Young, "Software Testing and analysis: Process, Principles, and Techniques," Wiley, ISBN 13: 978-0-471-45593-6, USA, 2007.

[2] R.C. Cofer, B.F. Harding, "Rapid System Prototyping with FPGAs," Newnes, ISBN: 0750678667, September, 2005.

[3] Ž. Hocenski, I. Aleksi, V. Sruk, "Adaptive Virtual Devices Platform for Verification of FPGA Modules in Student Courses on Digital Design," 7th IEEE International Conference on e-Learning in Industrial Electronics (ICELIE), pp. 22-27, ISBN: 978-1-4799-3180-4, Vienna, Austria, 10-13.11.2013.

[4] Xilinx Inc., "PicoBlaze 8-bit Embedded Microcontroller User Guide for Spartan-3, Spartan-6, Virtex-5, and Virtex-6 FPGAs," UG129 (v2.0), January 28, 2010.

[5] Pong P. Chu, "FPGA Prototyping VHDL Examples, Xilinx Spartan-3 Version", Wiley-Interscience, ISBN: 978-0470185315, February, 2008.

[6] A. Cicuttin, M.L. Crespo, A. Shapiro, N. Abdallah, "Building an Evolvable Low-Cost HW/SW Educational Platform--Application to Virtual Instrumentation," IEEE International Conference on Microelectronic Systems Education (MSE'07), pp.77-78, 2007.

[7] R. Hashemian, J. Riddley, "FPGA e-Lab, a Technique to Remote Access a Laboratory to Design and Test," IEEE International Conference on Microelectronic Systems Education (MSE'07), pp.139-140, 2007.

[8] Y. Rajasekhar, W.V. Kritikos, A.G. Schmidt, R. Sass, "Teaching FPGA system design via a remote laboratory facility," International Conference on Field Programmable Logic and Applications, DOI: 10.1109/FPL.2008.4630040, pp.687-690, September 2008.

[9] M.A. Soderstrand, "Role of FPGAs in undergraduate project courses," MSE, pp.0109, 1997 International Conference on Microelectronics Systems Education (MSE '97), 1997.

[10] T.S. Hall, D.V. Anderson, "A Framework for Teaching Real-Time Digital Signal Processing With Field-Programmable Gate Arrays," IEEE Transactions on Education, vol. 48, no. 3, pp. 551-558, August 2005.

[11] T.S. Hall, J.O. Hamblen, "Using an FPGA Processor Core and Embedded Linux for Senior Design Projects," MSE '07 Proceedings of the 2007. IEEE International Conference on Microelectronic Systems Education.

[12] J.D. Lynch, D. Hammerstrom, R. Kravitz, "A cohesive FPGA-based system-on-chip design curriculum," IEEE International Conference on Microelectronic Systems Education, 2005. (MSE '05), DOI: 10.1109/MSE.2005.5, ISBN: 0-7695-2374-9, pp. 17-18, June 2005.

[13] R. Seinauskas, "A distance laboratory for computer-aided design", IEEE International Conference on Microelectronic Systems Education, 1997., ISBN: 0-8186-7996-4, pp. 107 - 108, Jul 1997.

[14] I. Skliarova, "A Multimedia Tool for Teaching Reconfigurable Computing", Computer and Electrical Engineering, 2009., ICCEE '09, vol.1, pp.204-208, 28-30, Dec. 2009.

[15] J.S. Pastor, I. Gonzalez, J.Lopez; F.Gomez-Arribas, J.Martinez, "A remote laboratory for debugging FPGA-based microprocessor prototypes," IEEE International Conference on Advanced Learning Technologies, pp. 86-90, DOI: 10.1109/ICALT.2004.1357380, September 2004.