eDiViDe: European Digital Virtual Design Lab

A Remote Learning Platform for Digital Design

J. Vandorpe, J. Vliegen, R. Smeets, N. Mentens
KU Leuven @ KHLim
Diepenbeek, Belgium
E-mail: {Jochen.Vandorpe, Nele.Mentens}@khlim.be

M. Drutarovsky, M. Varchola
Technical University of Kosice
Kosice, Slovakia
E-mail: {Milos.Drutarovsky, Michal.Varchola.2}@tuke.sk

K. Lemke-Rust, T. Bartkewitz, P. Samarin, P. Plöger
Bonn-Rhine-Sieg University of Applied Sciences
Sankt Augustin, Germany
E-mail: {Kerstin.Lemke-Rust, Peter.Samarin}@h-brs.de

D. Koch, Y. Hafting, J. Torresen
University of Oslo
Oslo, Norway
E-mail: {koch, yngveha, jimtoer}@ifi.uio.no

Conference Key Areas: Novel education tools for engineering programs, Information and communication technologies, Integration of research in engineering education

Keywords: FPGA, remote lab, digital design, electronics

INTRODUCTION

In European companies and universities, VHDL is the most popular language for the design of digital electronic systems. In many universities, VHDL is contained in electrical engineering courses. The language is used for the development of applications on programmable and non-programmable chips.

The design of digital electronic applications needs a lot of practice. The classical way to practice, is by simulating digital designs using specific software on a PC. This way of verifying a digital design is mostly experienced by students as being boring. Furthermore, a simulation-only verification approach causes the students to lose contact with reality, while the development of a real-life application is a big advantage for a future engineer. To solve the limitations of simulation software, students should be offered the opportunity to configure a programmable chip driving a real-life application. The disadvantage of this kind of learning environment is the overhead in time and money for the developers (i.e. usually the professors or assistants).

eDiViDe is a remote platform that allows students to access several real-life setups whenever they are connected to the internet. These setups are developed by European universities and are made programmable through the internet using VHDL. Each setup will be accompanied by a camera and/or microphone that registers the behaviour of the setup and sends back the result to the student. This way, an additional verification of the design can be done next to simulation. The impact of eDiViDe is two-fold. On the one hand, the platform improves students' skills in digital design in a motivating environment. On the other hand, eDiViDe contributes to the visibility of the research activities in the contributing universities, since each university has the opportunity to focus on its own expertise when adding a setup to the platform. Moreover, this remotely facilitates the integration of research in education. Since eDiViDe is extensible, universities can add their own laboratory setup to the remote platform. At the moment, 8 remote laboratory setups are hosted in Belgium, Germany, Slovakia and Norway.

This paper describes the back-bone of the eDiViDe platform, the remote setups and the learning environment. The paper is organized as follows. Section 1 discusses related work. Section 2 describes the back-bone of the platform. In Section 3 an overview of the currently integrated remote laboratory setups is given. Section 4 concludes the paper and elaborates on future work.

1 RELATED WORK

In 2001, Genoe and Mentens developed a remote laboratory setup that consisted of an FPGA, LEDs, a 7-segment display, an oscilloscope and a webcam [1]. In 2009, Drutarovsky et al. established a hardware working place with powerful measurement equipment (logic analyzer, digital storage oscilloscope, programmable generator) for testing FPGA designs with basic remote acquiring of measured signals and configuration of target FPGA kits [2][3]. In comparison to these two existing platforms, the paper at hand makes the concept of remote laboratories global by providing a distributed solution in which remote laboratories are hosted by institutes in several countries. Further, our platform is not limited to basic setups, but introduces advanced, research-related laboratory setups. At the moment though, the 8 setups that are attached to the platform are not yet research-related. In the future, 8 more research-related setups will be added to the platform.

The work of Morgan et al. also presents an extendable platform together with a case study that introduces the platform into an undergraduate course [4][5]. Nevertheless, our solution distinguishes from their work by providing more varied setups that are distributed over 4 countries.

2 BACK-BONE OF THE PLATFORM

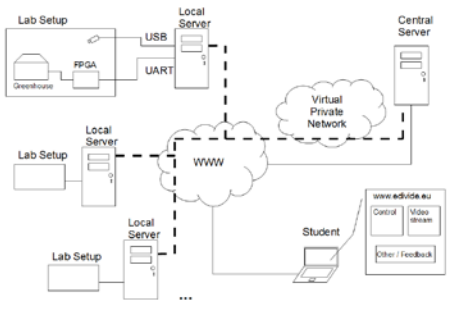


Fig. 1. Distributed server setup

FPGA boards with peripherals are accessible over the internet together with a system that processes the student’s design and programs the FPGA. The back-bone was built as a distributed platform of servers, consisting of one central and multiple local servers. The central server is an interface to and coordinator for the local servers, which are physically connected to the FPGAs and setups.

In summary, the user will browse to http://www.edivide.eu. He selects a specific setup and is presented with an interface that allows control of that setup. *Fig. 1* gives a schematic representation of the eDiViDe platform.

2.1 The web-based interface

The eDiViDe site is more than just a control interface for the setups. It is also a portal to information on digital design using vhdl and has a forum for registered users to discuss their designs and issues. After login, the student is presented with a personal dashboard from which he has access to all features of the platform. He will be able to browse through and read about the available setups, upload his designs and take control of a setup after he has made a reservation.

2.2 The central server

The central server functions as platform controller and access point for all external communication and is thus publicly available. It hosts the webserver at http://www.edivide.eu and provides the eDiViDe interface for students. It is also home to scripts that handle all intersystem communication and coordination with all of the connected local servers. When a reservation is started it will program the FPGA with the student's bitstream and initiate a communication channel and video stream from the setup to the student his browser. The video stream however is processed directly from the local server to an external streaming server to offload the central server.

2.3 The local server

The local server connects with the central server through a VPN tunnel and resides in a laboratory where it directly connects with the FPGA boards and other peripherals over USB and network cables. It also contains the tools for the tool chain processing of vhdl designs and the scripts that will handle all interaction with the connected devices. When the student submits his design it is the local server that will do the actual design processing. This way the distributed system should be able to handle the multitude of classes at different universities using this platform.

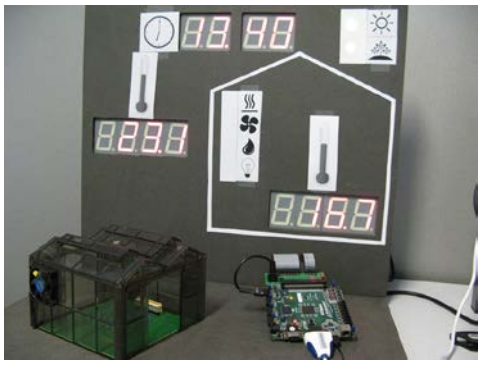
3 REMOTE LABORATORY SETUPS

3.1 Lab setups hosted in Belgium

Greenhouse

The greenhouse setup is a basic setup available on the eDiViDe platform and is hosted by KHLim in Belgium. This setup allows the student to learn the basics of vhdl programming and will gradually increase in difficulty with each exercise. The setup consists of an FPGA board that is connected with the greenhouse infrastructure and a visualisation board with 7-segment displays and LEDs. The FPGA board that is used in this setup is the Nexys 2 from Digilent with a Xilinx Spartan 3E FPGA. The greenhouse setup aims to introduce intermediate level vhdl development to the student. The greenhouse presented in this particular setup is equipped with the following infrastructure that should allow to keep good care of the present plants: illumination, heating, irrigation and ventilation. The setup is depicted in *Fig. 2*.

The greenhouse setup is built around a plant model that requires you to implement necessary control circuitry for the greenhouse infrastructure. The model includes the time of day, the outdoor temperature, the indoor temperature, the amount of daylight (seasons) and a visual representation of the plant's state. The description of the exercises will explain the exact aim and the instructions on how to take good care of the plants in the greenhouse.



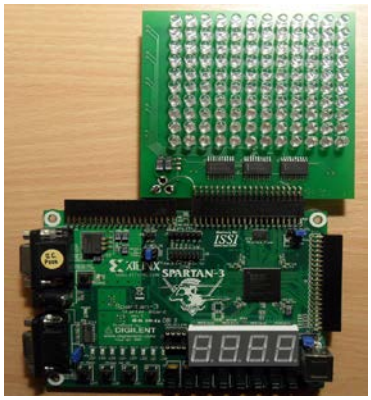
When the vhdl design is successfully synthesized and a bitstream is generated by the system the student will be able to control the FPGA and greenhouse through a control panel in the web interface. Visual feedback of the setup is provided through a webcam live stream. It is also possible to develop and include debug circuitry if the student needs to visualize internal signals or if the design is not reacting as expected.

Fig. 2. Greenhouse setup

Stepper motor

The stepper motor control setup is another basic setup available on the eDiViDe platform. The setup consist of an FPGA board with LEDs, switches and push buttons and a stepper motor. The FPGA board that is used in this setup is the Nexys 2 from Digilent with the Xilinx Spartan3E FPGA. When the student's vhdl design is successfully synthesized and a bitstream is generated by the system they will be able to control the FPGA and stepper motor through a control panel in the web interface. Visual feedback of the setup is provided through a webcam live stream.

3.2 Lab setups hosted in Germany



Two basic setups that are hosted at the Bonn-Rhein-Sieg university use the same hardware. Each setup consists of an FPGA board from Digilent and a custom-made LED matrix. The board contains a Spartan3 XC3S200 FPGA from Xilinx that has 200K gates, 216Kbit of block RAM, and can be driven by a clock of up to 500 MHz. The user can interact with the board by using 4 push buttons and 8 slide switches. The FPGA is connected to a set of I/O devices such as a 1Mbyte SRAM, 8-bit VGA interface, and a serial port. In addition, three 40-pin expansion connectors are available.

Fig. 3. Basic setups hosted in Germany

One of the expansion connectors is used to connect the board to an LED matrix that contains 120 LEDs arranged in 12 rows and 10 columns. The matrix is time-multiplexed, such that only 10 LEDs can be set at once. The LEDs can be mapped one to one to the web interface, so that the students can set and unset individual LEDs directly. The setup is shown in Fig. 3.

Setup 1: Games on the LED Matrix

The purpose of this basic setup is to enable the students to develop two simple games: Pong and the Conway's Game of Life. Using this setup, the students can learn the basics of programming in VHDL through a set of exercises involving the LED matrix. The difficulty of the exercises gradually increases. In the beginning, the students learn how to set individual LEDs of the matrix, then how to perform time

multiplexing and be able to use the whole LED matrix. After learning the basics, the students will be able to implement Pong and the Game of Life.

Pong is a two-dimensional tennis sports game. The players move rackets vertically in order to keep a ball moving. A player earns points if the other player fails to return the ball. The current score is displayed on the 7-segment display. The game starts with a straight ball in horizontal direction. The first racket that hits the ball at the upper or lower part will make the ball moving also in vertical direction. Every time a racket hits the ball in the middle, the speed of the ball movement is increased until some specified value. When a player reaches five points, the game ends.

Conway's game of life is a zero-player game. Its evolution is completely determined by its initial state. Every cell interacts with its eight cell neighbors. In this setup the neighbors are LED cells that are horizontally, vertically, or diagonally adjacent, i.e. in total 120 cells are possible with this setup. At each step in time, the following transitions occur: 1) Any live cell with fewer than two live neighbors dies, as if caused by under-population; 2) Any live cell with two or three live neighbors lives on to the next generation; 3) Any live cell with more than three live neighbors dies, as if by overcrowding; 4) Any dead cell with exactly three live neighbors becomes a live cell, as if by reproduction.

Setup 2: Crypto Lab

The purpose of this basic setup is to enable the user to develop simple cryptographic algorithms that are well suited to hardware implementations. The output of the algorithms is visualized using a 7-segment display in combination with the LED matrix. The feedback channel to the user is the video stream that shows the visible outputs of the FPGA and the LED matrix.

The Crypto Lab offers a broad range of exercises ranging from simple tasks such as implementing and testing a Linear Feedback Shift Register (LFSR) to the full implementation of a modern hardware-oriented stream cipher such as Trivium. For all crypto implementations, the LED matrix can be used to display the current internal state of the crypto algorithm.

3.3 Lab setups hosted in Slovakia

By these setups we currently provide support for Altera FPGA by using Altera Quartus II CAE tool [1]. We use Altera Nios II Cyclone III evaluation kit (AKIT) [2] as a common hardware platform for all our developed setups. The AKIT includes full-featured FPGA starter board based on Cyclone III EP3C25F324 FPGA and additional LCD multimedia daughter-card as shown in Fig. 4. The Altera EP3C25 contains 24,624 Logic Elements (LEs), 608,256 RAM bits, 66 18x18 bit multipliers and 4 PLLs and all these resources are available for our designs.

As standard communication and debug channels for our designs we use 64-bit packets sent via UART in both directions. Our setups can use additional available channels: 800 x 480 LCD screen and audio codec monitored by camera and microphone. We implemented interfaces to these channels as custom logic implemented in VHDL and controlled by a set of Finite State Machines (FSMs).

Clear separation of these interfaces from our setups allows us to develop users' setups independently from hardware (HW) interfaces and/or optimize HW interfaces (e.g. increase driver's resolution) in future upgrades. Our two implemented Sea Noise Emulator and FSM setups share one AKIT HW resources.

Sea Noise Emulator setup

By this setup we demonstrate the basics of VHDL encoding of simple sequential and combinatorial circuits typically used as a part of more complex communication and security devices - Linear Feedback Shift Registers (LFSRs). The sea noise emulator setup aims to introduce basic level of VHDL development and demonstrate potential of simple LFSR circuit to provide more complex functionality. The setup consists of an FPGA board that is connected to the audio input of the server. The sea noise emulator setup is built around a simple combination of LFSR based pseudorandom number generator and a nonlinear counter that control each other. The nonlinear counter provides dynamic clock input for LFSR and pseudorandom LFSR output provides varying input for threshold change in the nonlinear counter. The emulator represents a very compact circuit that provides interesting audio output. The users can start/stop generator, change the LFSR seed value, monitor audio output and buffer generated data to the output UART channel. Users can play with thresholds and even modify its structure.

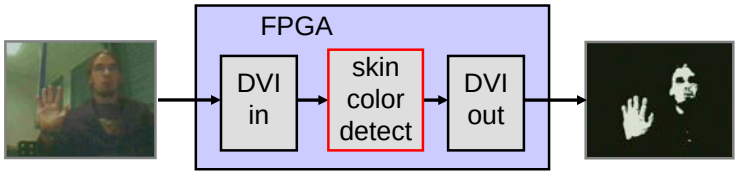
Finite State Machine setup

By this setup we demonstrate the basics of VHDL encoding of simple sequential and combinatorial circuits typically used as a part of more complex digital designs - FSMs. The Finite State Machines setup uses a simple FSM which controls information indicated on the AKIT LCD display. The FSM transactions can be controlled via UART interface. The FSM is used to implement the count-down system. At beginning, the preset value is loaded to the two-digit decimal number. The command send via UART starts the decrementation of indicated number. When zero is reached, the ash blinks eight times. After that, the preset value is loaded to the display. The user can also define the preset value in this state using the UART interface. User can implement FSM by using different FSM encoding techniques as: Mealy, Moore or Registered Next State types of FSMs as well as various coding styles, for example One-Hot, Gray coding, etc.

3.4 Lab setups hosted in Norway

Human Skin-Colour Detection

The goal of this lab is to demonstrate the compute power of an FPGA and how to utilize it using VHDL on detecting human skin colour in a live video stream.



(R,G,B) is classified as skin if:
R>95 and G>40 and B>20 and
max{R,G,B}-*min*{R,G,B}>15 and
|R-G|>15 and R>G and R>B

Fig. 4. Human skin-colour detection setup

The setup illustrated in *Fig. 4* allows students to upstream own webcam data to the FPGA board. There the video is processed by the skin colour detection module (to be written by the students) and returned back for providing live feedback experience.

Skin colour is detected using Peers algorithm [8] (shown right of the figure). By using pipelining, pixel rates of up to 200 MHz are possible on the provided Xilinx Spartan-6 LX45 FPGA. Taking only about 1% of the logic resources and consuming about 1W power, students will sense that their tiny FPGA module provides already >2GOPS processing throughput when they compare this with a software solution. This is a universal setup encouraging student to try out various other image processing

algorithms (e.g., Sobel or Median). In an advanced lab, modules will be swapped using partial reconfiguration, for example, to react on different light conditions.

MIPS CPU

The best way to learn about how a CPU works is to build one! In this lab, students will learn basic principles of CPU design. While computer science students commonly write software and compile it for a given CPU, this lab changes the role and students get a program and they will implement the CPU instead. We chose the MIPS-1 instruction set as a reference instruction set architecture (ISA) for two important reasons: 1) it is easy to understand and 2) it is well supported by many compilers and simulators.

While the program uses only a subset of the MIPS ISA, (load/store, simple arithmetic, branches), the overall learning outcome of this lab is that only two pages of VHDL code are sufficient to write a tiny 32-bit CPU that provides more performance than many dedicated microcontrollers. Students will learn how machine code is actually executed by a CPU. To limit complexity, students are not asked to implement a pipelined version of this CPU and there is no hazard detection required.

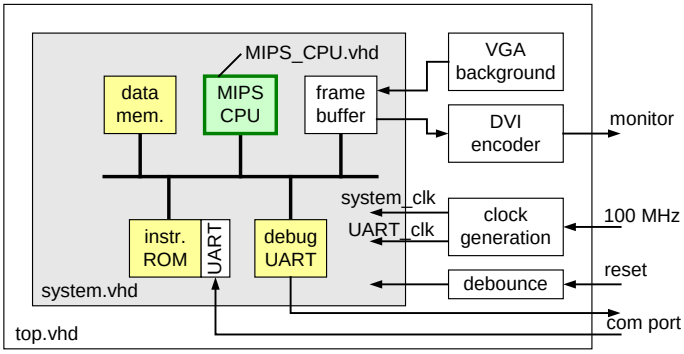


Fig. 5. Top-level view of the MIPS CPU setup

The setup provided to the students comprises a wrapper for the MIPS CPU and a surrounding system with memories and a frame buffer module, as shown in *Fig. 5*. The content of the frame buffer is to be manipulated by the MIPS and the buffer content is displayed on an external monitor output which is send to the student. In another lab, students will extend the ISA with own run-time reconfigurable instructions, hence, defining their own CPU which can still be programmed using C.

4 CONCLUSIONS AND FUTURE WORK

This paper presents a learning platform for digital design based on remote laboratory setups. The remote setups are distributed over 4 countries and the platform is extendable. At the moment, 8 setups are attached to the platform. In the future, the platform will be extended with 8 more research-related setups. Further, the platform will be integrated in undergraduate and graduate courses in the participating institutes. The learning outcomes will be evaluated and analysed and partners will continuously be sought for using and extending the platform.

5 ACKNOWLEDGMENTS

This work is partially funded by the Education, Audiovisual & Culture Executive Agency (EACEA) under the Lifelong Learning Program (LLP) with project number 518565-LLP-1-2011-1-BE-ERASMUS-ESMO and project title eDiViDe. This work is also partially funded by the Onderwijsontwikkelingsfonds (OOF) of KU Leuven.

REFERENCES

[1] Genoe, J. and Mentens, N. (2001), Implementatie in hardware (FPGA) en visualisatie via een webcam en een oscilloscoop van VHDL-code die via het internet wordt ingebracht, http://ontwerpen1.khlim.be/~jgenoe/PUBL/STIHO%20digitaal.pdf.

[2] Drutarovsky, M., Saliga, J., Michael,i L. and Hroncova, I. (2009), Remote Laboratory for FPGA based Reconfigurable System Testing, in the Proceedings of the XIX IMEKO Word Congress, Fundamental and Applied Metrology, pp. 54-58.

[3] Drutarovsky, M., Saliga, J. and Hroncova, I. (2009), Hardware infrastructure of remote laboratory for experimental testing of FPGA based complex reconfigurable systems, In Acta Electrotechnica et Informatica, Vol. 9, No. 1, pp. 44-50.

[4] Morgan, F., Cawley, S., Callaly, F., Agnew S., Rocke, P., O'Halloran, M., Drozd, N., Kępa, K. and McGinley B. (2011), Remote FPGA lab with interactive control and visualisation interface, Proc. of the Int. Conference on Field Programmable Logic and Applications (FPL), IEEE, pp. 496-499.

[5] Morgan, F., Cawley, S. and Newell, D. (2012), Remote FPGA Lab for Enhancing Learning of Digital Systems, In ACM Transactions on Reconfigurable Technology and Systems, Vol. 5, No. 3, pp. 18:1-18:13.

[6] Altera Corporation. (2013) Design software webpage. [Online]. Available: http://www.altera.com/products/software/sfw-index.jsp.

[7] Altera Corporation. (2013) Nios II Embedded Evaluation Kit, Cyclone III Edition User Guide, Altera, July 2010. [Online]. Available: http://www.altera.com/products/devkits/altera/kit-cyc3-embedded.html

[8] Peer, P., Kovac, J. and Solina, F. (2003), Human Skin Colour Clustering for Face Detection, Proc. of the IEEE EUROCON Conference, pp. 144-148.