Learning-by-Gaming in HW/SW Codesign

Vadim Pesonen, Maksim Gorev, Kalle Tammemäe
Department of Computer Engineering
Tallinn University of Technology
Tallinn, Estonia
{vadim.pesonen, maksim.gorev, kalle.tammemae}@ati.ttu.ee

Abstract **— HW/SW Codesign is a part of renewed Computer and Systems Engineering Master Curriculum at Tallinn University of Technology. Already historically, practical works are related to computer game design elements on different FPGA-based prototyping platforms. Current platforms, based on Xilinx Spartan-3 chips, enable complex designs where the only restrictions are the course duration and student imagination. In this paper, the status of course along with the most interesting student design examples are provided and analysed. The students' feedback is used to refresh the course material and complexity of laboratory task set.**

Keywords – Hardware/Software Codesign, system-level design, education, computer engineering curriculum

I. Introduction

With the growing industrial importance of hardware/software codesign the necessity for educating engineers in this field is a regular part of CE curriculum around the world. But purely theoretical knowledge is impractical without hands-on experience with recognised EDA tools and advanced development platforms. The complexity of modern HW/SW design can be effectively overcome with interactive and entertaining laboratory works [1].

Game design projects are widely and successfully used for teaching programming, and it was decided to apply the same method for HW/SW codesign. In a similar work [2] an FPGA was used to implement an object detection algorithm for a robot soccer game. Although probably designed as a student project in the frame of a certain course, the paper focuses mainly on design issues.

HWSW Codesign [3, 4] has been read in Dept. CE of Tallinn University of Technology since 1998. After several redesigns of the curricula and implementation of Bologna system, it is now defined as part of the Master level curriculum of Computer and Systems Engineering at faculty of Information Technology.

The course objectives are defined as follows: After completing the course the student is expected to:

- be able to select the proper modelling languages and tools
- be aware of the design space limits and freedom in case of pure HW or SW design or in case of a blended system
- be able to analyse a model concurrently in the HW and SW contexts
- be able to partition a design on a basis of estimations and analysis
- possess the skills to implement different cross-domain interaction schemes and protocols
- be able to use standard off-the-shelf software implementations as well as IP core (soft, firm, hard) processors
- be able to analyse a design in the frame of dynamic reconfiguration on a basis of temporal quality requirements
- know the contemporary HW/SW mixed system implementation platforms and frameworks
- be able to accomplish team tasks

Currently, the course is provided during the spring semester. Students possess a significant flexibility in planning their curriculum and are free to choose whether to take the course in the first or the second year of their master studies. The course is therefore accessible to them either in the second, or the last semester.

Those students who are taking the course in the first year usually have only a limited experience with digital electronics design, but are likely to have taken the introductory course to hardware description languages (HDL-s). However, the synthesis of digital circuits has been explained at this point only in the broad strokes.

The other students may have taken one or several of the following courses: Digital System Design, Labs on Digital System Design, Computer Engineering Project, Digital Systems – Team Project, Computer Engineering – Team Project, VLSI Synthesis, System-on-a-Chip Design, and several others.

All of the listed courses are optional and as a result students arrive with a notably different background knowledge and experience. This was taken into account when planning the laboratory works for the course at hand and it is reflected in their structure.

II. Structure of the Practical Works in HW/SW Codesign Course

A contemporary laboratory part of the course was introduced in 2008. It consists of four levels in ascending order of complexity:

1. Introductory – first hands-on experience with XESS XSA-3S1000 boards [5], downloading and programming previously implemented tutorial designs.

2. Beginner – simple HW-design using VHDL or Verilog.

3. Intermediate – design of a VGA demo along with PicoBlaze controlled moving element on a screen. VGA generator is reused from one of XESS tutorials. The emphasis is on programming and instantiating of the soft-core processor and VGA IP cores.

4. Advanced – enhanced design where HW/SW trade-offs have to be investigated and results analysed.

The first lab is carried out under direct supervision and is mainly intended for those students who lack experience with digital system design in general or with the FPGA prototyping boards. Students learn the main features and capabilities of Xilinx ISE, the Xilinx Spartan-3 chip and the development board. After that student teams are free to take the development kits home and visiting the laboratory becomes voluntary. Freely available versions of the development software are enough for this course, and about half of the students found it more convenient to work at home. The difficulty level of the introductory designs is that of simple decoders and small state machines.

The aim of the second lab is to apply the theoretical knowledge of HDL-s. The designs are still simple at this level, but the students are now to pass through all the development stages – from specification to chip configuration – independently, and they are free to match or challenge their HDL and hardware knowledge by choosing an appropriate design. By doing so students usually find this work practical regardless of their previous digital design experience. However, the selected designs significantly vary and their range extends from simple arithmetic-logic units to more complex ones, such as interface controllers.

The previous two laboratory works form a solid ground for the next steps and by the third laboratory work all students are prepared well enough to combining software control with hardware components. At this point the hardware part of the design is still accomplished by putting together predefined IP blocks. The aim is to learn how to accommodate both software and hardware within a single design, and to get a feel of using programmable soft-core processors, their instruction set, features and particularities.

For better student motivation the tasks in this laboratory work had to be made interactive and with a visual feedback. An FPGA development board with an attached keyboard and a monitor would make a good platform for that – this way students get a feel of a real system, capable a complex interactive behaviour.

Since it is not the aim of this work to familiarize just with the keyboard and screen interfaces, the corresponding IP cores were provided. To further abstract the students away from the interface details, the keyboard and video controllers were accordingly adjusted for the task of this lab.

All the complexity of image manipulation is covered by the provided VGA generator, which maps the pixel data to the corresponding memory cells. Additional multi-port memory controller significantly simplifies memory access. Both video and memory controllers are freely available from the board manufacturer's website.

The configurable keyboard controller disguises the process of key recognition and, depending on the current functioning mode, issues out a short code for a pressed key set. For example, if only the “arrow” keys are used in the design, then the keyboard controller is configured to generate an encoded 2-bit word.

Generally, students are free to choose any soft-core processor they fancy, but the majority prefers the Xilinx PicoBlaze, as it is very stable, easy to use, well documented and its instruction set is rich enough for most simple control oriented tasks. PicoBlaze also performs very well in terms of die area, as it is specifically designed for the FPGA implementation.

All major components are therefore provided, but any additional functional units and glue logic are the responsibility of the students. Most of the student designs in this laboratory work are relatively simple implementations of well-known video scenarios, such as a “snake” game, a text editor or a calculator. The reason behind such selection is simple application logic, few required colours (black-and-white in many cases) and limited amount of data manipulation.

In the final laboratory work students have to design the whole system from the ground up, including HW/SW partitioning trade-off considerations. The aim is to learn how to correctly partition hardware and software according to certain constraints, such as logic area, power consumption or development time. Such analysis turned out to be very difficult for some of the students to perform, as they had little experience in digital design, and practically all failed to make exact estimations.

The required several different HW/SW partitions together with limited digital design experience represented another difficulty - now part of the application logic, which would normally be done in software, had to be implemented in hardware. In addition, the previously used IP blocks may have had to be modified to suite the application. Many found it easier to completely redesign a functional block or an interface controller than adapting an existing one. However, examining and understanding the interface specifications had to become part of the work, if that were the case.

III. An Example of a Game Design

One of the proposed laboratory works is a simple implementation of the Sokoban game [5]. “Sokoban” is a Japanese word for warehouse keeper. Basically, the job of a

warehouse keeper is to place boxes within the warehouse in an organized manner, and this is the key idea in the Sokoban puzzle. The rules are simple and yet give rise to challenging puzzles ranging from simple to extraordinary complex ones. The game consists of a warehouse made up of walls that form passages. Within the warehouse are the pusher and an equal number of boxes and storage locations. The pusher can only push a box, never pull, and only one box can be pushed at a time. The goal is to push all the boxes into the storage locations.

This application is well suited for the task in several ways. Firstly, the design is comprised of several functional modules and a fixed communication schema between them. This allowed to implement certain modules in either hardware or software without affecting the rest of the design. Secondly, the game is visually comprised of a very limited number of different objects and its logic is carried out by repositioning them on the screen. The application simply honours keyboard activity and maintains a map of the game objects. This is quite easily achieved in both software and hardware. Finally, the custom VGA engine is capable of displaying one hundred (ten by ten) predefined images on the screen.

The system consists of the game logic processor, the memories, the video controller, and the keyboard controller. The employed video standard is VESA 800x600 @ 72Hz. The reason behind the selection was the required horizontal clock frequency (50MHz) that matches the frequency, supplied by the development board. The video picture is comprised of 100 (10 x 10) image locations which form the game map. The map is stored in the map RAM, which can be accessed by the processing unit and the video adapter. Each location on the map contains an image, which are stored in image ROM and are accessed only by the video adapter. To form the right VGA signal, the video adapter keeps track of the imaginary cathode ray position on the screen and then, in order to retrieve the pixel colour value, turns first to the map RAM and then to the image ROM. This procedure is pipelined.

The memories are implemented using on-chip block RAM (BRAM). Each object image is 80 x 60 pixels in size and there are up to 16 images in the set. With a 9-bit colour depth the set would not fit into BRAM and so the images are stored shrunk in a 20 x 15 pixel aspect ratio. They are later enlarged by the video adapter by simply repeating each rows and columns of the image four times during display.

The PS/2 keyboard controller accepts the data sent by the input device, extracts the key information, encodes it for convenient processing and sends an interrupt to the processing unit. Figure 1 illustrates the apparatus setup – the development board in the centre, the VGA screen and the keyboard.

Figure 1. Apparatus setup.

Several designs with different hardware/software ratio were implemented. While the VGA adapter was reused in all configurations, the keyboard controller and game logic processor were done in both software and hardware. The Xilinx PicoBlaze processor was employed to run the software. The block diagram of the system is depicted in Figure 2.

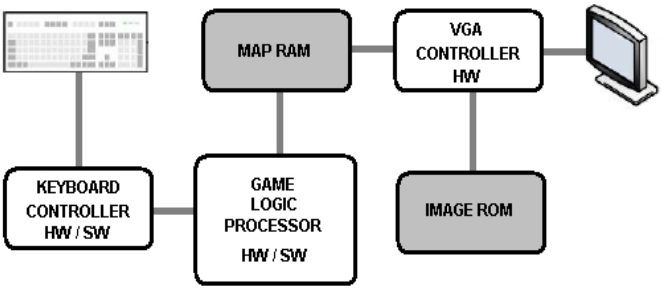

Figure 2. Block diagram of the a game system.

Students reported the following development times of various game components:

Table 1. Development times (in hours) of the game components.

	HW	SW
Game logic controller	8	16
Keyboard controller	1	3
VGA controller	32	n/a

Three different solutions, one of which was pure hardware, were then compared for development time, FPGA logic utilization and power consumption. The results showed, that the pure hardware implementation, due to a complex FSM, was not the most compact in terms of the number of used CLB-s, but required the least power and was faster to design and debug (41 hours [32+1+8] versus 43 [32+3+8] and 49

[32+1+16] for other solutions). The latter fact can be explained by larger digital design experience compared to software development of that group of students. Also, these students practised a mixed language design (VHDL + Verilog), depending on their preferences and readiness.

IV. International Concerns

English was selected as the primary course language as it is the language of a large portion of technical literature and documentation, and is planned to be part of new INTELS international Master curriculum [7] at Tallinn University of Technology. This makes the course suitable in other countries where English as a study language is in use. Generally, the required development boards or the FPGA vendors are not limited to the ones used in our university. But in any case the employed software can be downloaded from the internet at no charge and the boards, which are among the most affordable, can be purchased from the manufacturer web-site.

In addition, this course is a good candidate for blended e-learning because most of study activities can be done outside of classroom using free-ware development tools and lent prototyping kits. Most of the course materials are planned to be open course-ware available through the Estonian e-Learning Development Centre repository.

V. Course Results and Conclusions

The proposed set of laboratory works embraces the whole range of the course goals, and their accomplishment is a good indicator of whether these goals are achieved or not. Unfortunately there is not enough data to evaluate the situations when students gave up after solving the first couple of laboratory works and dropped the course. Although the majority did manage to fulfil the tasks successfully, about 30% of students failed to solve the most complicated laboratory task but a good grade is still possible when essay is presented and assessed by peer students positively.

Another demonstrative fact is that students, who successfully solved all laboratory tasks, didn't have difficulties in solving additional course tasks – analysis of HW/SW codesign related paper or thesis and a presentation of a written essay openly to extend the overall knowledge about the domain for all course participants.

Both positive and negative feedback was received during the course. Generally students appreciated the opportunity to design something real and challenging. Some of them recognized this course to be the only one during the semester which put them to work in teams with full intensity. Some of the more successful game designs are demonstrated to first years undergraduate students during the Introduction to Speciality course [8], as an example of how laboratory works can be both practical and interesting.

On the other hand a significant complexity gap was mentioned between assignments 3 and 4. Apparently a jump from component based design to fully custom was too big. In addition, complexity of the labs was overwhelming for roughly 30% of the students. The main reason was lack of HDL knowledge and insufficient number of working design examples, especially for the more complex tasks. Currently, these problems are being addressed: the tutorials are being revised and at least basics of HDL-s are compulsory for all arriving to course.

VI. Future Work

Apart from considering all the negative feedback that was previously received, future enhancements include experimenting with more powerful and flexible soft-core processors, rather that the PicoBlaze, for the following reasons:

- The general complexity of games (e.g. large number of game control states, the need for trigonometric calculations) are too often exceeding the PicoBlaze processing power.
- The PicoBlaze can only be programmed in its own assembly language, which limits the use of freely available game codes. Development of translation skills from a high-level code to assembly language is not one of the course goals.
- The PicoBlaze is designed to operate with a singe-cycle access on-chip memory. Although on-chip block memory can be explicitly rewritten using JTAG interface, there is not enough JTAG programmer modules in the laboratory to distribute along with kits. As a result the whole design has to be resynthesised by the student if the software is modified, which is very inconvenient and time-consuming. Fixing and reloading just the program is a better way to debug the system, but to do so the program must be stored in an external memory. In addition, employing only the on-chip memory would mean limiting the experience of using more capable, though slower, external dynamic memories.

Among the more powerful soft-core processor candidates is the Xilinx MicroBlaze. Being widely used in the industry, it represents a significant interest in an academic sense. It also by far outperform the PicoBlaze in the above listed areas. However, the flexibility of MicroBlaze makes it more difficult to configure and manage, demanding a significant portion of students' time during the course. Consequently, a large amount of work has to be prepared beforehand, as the current emphasis is on implementing of a learning object, which is intended to shorten learning time and to offer students a systemic set of IP-s ready to embed into real application. Another drawback of the MicroBlaze, in the frame of this course, is its restricted suitability for distant studies – the trial period of the freely available Xilinx ISE Embedded Edition evaluation suite is limited to 30 days. As the course lasts longer, student have to return to the university laboratory, where computers are equipped with full-licensed EDK tool.

Other soft-core processors, which are considered as alternatives to the PicoBlaze, include the OpenRISC processor [9] and the COFFEE RISC core [10]. A significant advantage of these two processors over MicroBlaze is the access to their source codes.

References

[1]Min He, Ming-Che Tsai, Xiaolong Wu, Fei Wang, Ramzi Nasr, "Hardware/Software Codesign – Pedagogy for the Industry," itng, pp.279-284, Fifth International Conference on Information Technology: New Generations (itng 2008), 2008.

[2]Tim Kaulmann, Marc Strünkmann, and Ulf Witkowski, FPGA-based Object Detection in Robot Soccer Application, Proceedings of the 3rd International Symposium on Autonomous Minirobots for Research and Edutainment (AMiRE 2005).

[3]M. Kruus, P. Ellervee, "Four Years of System-on-Chip Curricula: Experiences at Tallinn University of Technology." The 6th European Workshop on Microelectronics Education (EWME'2006), Stockholm, Sweden, pp.88-91, June 2006.

[4]M. Kruus, K. Tammemäe, P. Ellervee, "SoC Curricula at Tallinn Technical University." The 19th NORCHIP Conference pp.99-104, Stockholm, Sweden, Nov. 2001.

[5]XESS Corporation. http://www.xess.com/

[6]The official Sokoban website. http://www.sokoban.jp/.

[7]The Archimedes Foundation. http://str.archimedes.ee/redirect/381.

[8]Maksim Gorev, Vadim Pesonen, Peeter Ellervee, “Introducing Computer Systems Related Topics in the First Study Semester”. Accepted for 8th European Workshop on Microelectronics Education (EWME), Darmstadt, Germany, May 2010.

[9]The OpenRISC project . http://www.opencores.org/project,or1k.

[10]The Coffee project. http://coffee.tut.fi/.