ELECTRONICS AND ELECTRICAL ENGINEERING
ISSN 1392 – 1215 2010. No. 6(102)
ELEKTRONIKA IR ELEKTROTECHNIKA

ELECTRONICS SCIENCES
ELEKTRONIKOS MOKSLAI

On Teaching Hardware/Software Co-design using FPGA

N. Bencheva
Telecommunications Department, Ruse University,
Studentska str. 8, 7017 Ruse, phone +359 82 888677, e-mail: nbencheva@ecs.ru.acad.bg
N. Kostadinov
Computing Department, Ruse University,
Studentska str. 8, 7017 Ruse, phone +359 82 888674, e-mail: nkostadynov@ecs.ru.acad.bg
Y. Ruseva
Telecommunications Department, Ruse University,
Studentska str. 8, 7017 Ruse, phone +359 82 888841, e-mail: iruseva@ecs.ru.acad.bg

Introduction

Computer and electrical engineers are involved in the design of computer-based systems to address highly-specialized and specific applications in computer, aerospace, telecommunication, power-production, manufacturing, defense and electronic industries. Modern real-time embedded systems need to perform tasks that are very demanding and require a great degree of sophistication. They need to be able to handle a mixture of data-intensive and control-oriented tasks. They need to have architectural support for multi-tasking, multi-threading, concurrency, polling, interrupts and preemption, as well as user and supervisor modes of operation [1].

At Ruse University in the Faculty of Electrical Engineering, Electronics and Automation the students are offered several introductory courses that focus on microcontroller-based systems, embedded programming and Programmable Logic Design in the bachelor level. Even at the master level, in the process of design, configuration and programming of microprocessors it is hard for the students to optimize the division of functions between hardware and software. In order to improve the students' knowledge and skills in the process of embedded systems design, the Hardware/Software Co-design is used in the teaching process.

The complexities in designing embedded systems motivate the need to use more efficient tools and design methodologies. System Level Design is a methodology to help address these complexities, and enable SoC designs. There are three main system level design approaches: hardware/software co-design, platform-based design and component-based design [2, 3]. Hardware/Software co-design (also referred to as system synthesis) is a top-down approach. It starts with system behavior, and generates the architecture from the behavior. It is performed by gradually adding implementation details to the design [2]. Generally, Hardware/Software co-design consists of the following activities (Fig. 1): specification and modeling, design and validation [4].

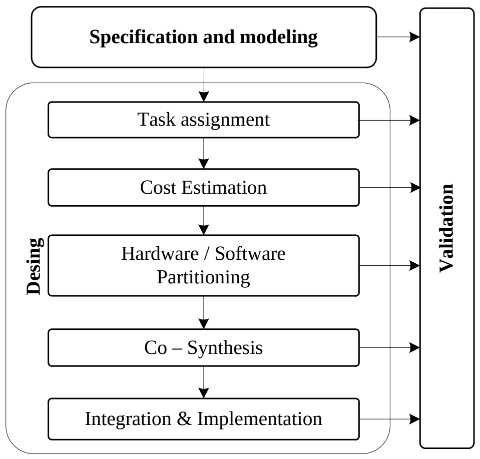
Specification and modeling
Task assignment
Cost Estimation
Hardware / Software Partitioning
Co – Synthesis
Integration & Implementation
Desing
Validation

Fig. 1. Activities of Hardware/Software co-design

Specification is the starting point of the co-design process, where the designer determines the system's specification without defining the implementations. Different languages are used to capture system specifications. Modeling is the process of conceptualizing and refining the specifications. Two approaches are used for system specification, homogeneous modeling and heterogeneous modeling [4, 5]. The goal of a specification language is to describe the intended functionality of a system non-ambiguously. A large number of specification languages are currently being used in embedded system design since there is no language that is the best for all applications [2, 6]. Examples of formal languages are CSP, LOTOS and SDL.

Tasks assignment is a process in which the system specification is divided into a set of tasks/basic blocks that define the system's functionality. The next step is estimation of the cost parameters for implementing the system's basic blocks (output of the task assignment) in hardware or software [2]. Hardware/Software partitioning divides the specification into two parts: a part that will be implemented in hardware and a part that will be implemented in software. Co-synthesis consists of several design steps: communication synthesis, specification refinement, hardware synthesis and software synthesis. Validation is defined as a process of determining that the design, at different levels of abstraction, is correct [2, 7].

Case study: traffic light controller (TLC)

The TLC controls the traffic of a crossroads of one major road and one minor. The traffic on the major road is stopped only when a signal from a sensor is received that there is a car on the minor road.

As it is shown in Fig. 2, the TLC can be composed of two subsystems working in parallel – Control block and Timer module.

The CSP for the *Timer* is simple sequence of events:

```
Timer = ((tmr?start.long → tmr!timeout → Timer)
            □
            (tmr?start.short →  tmr!timeout →  Timer)))
```

Once the CSP description is defined, it can be evaluated in the environment of a CSP model-checking tool, for example *Process Analysis Toolkit* (PAT) [8].

For instance, executed by PAT assertion

```
#assert TLC() deadlockfree;
```

asks whether the process *TLC* is deadlock-free or not, and the verification of assertion

```
#assert TLC() |= car.present →  <> car.passed;
```

validates that if there is a car on the secondary road, it will eventually pass the crossroad.

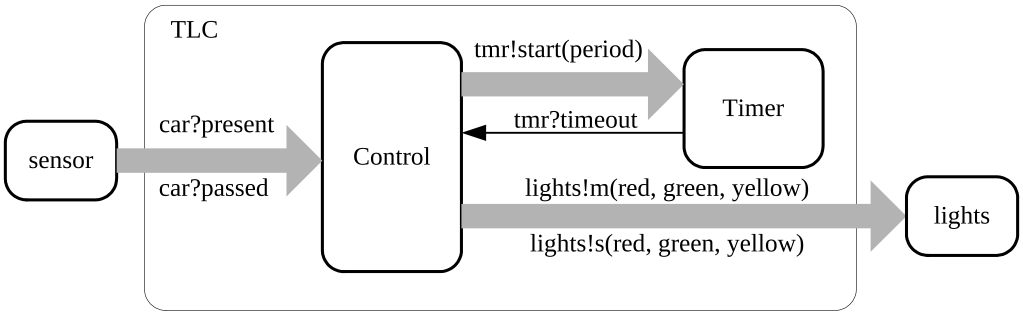

Fig. 2. TLC Block diagram

One of the prominent formalisms for designing concurrent systems at the specification level is the process algebra CSP (Communicating Sequential Processes) [9].

The CSP description of the TLC from the system view is:

```
TLC = (Control || Timer)
```

This states that the system TLC is defined as a parallel composition of two processes – Control and Timer.

The Control process is also defined as a number of sequential subprocesses, corresponding to its main working states:

```
Control = (S0; S1; S2; S3; S4; Control)
S0 = (lights.m.green → lights.s.red → car?present →
SKIP)
S1 = (tmr!start.short → lights.m.yellow →
lights.s.red_yellow → tmr?timeout → SKIP)
S2 = (tmr!start.long → lights.m.red → lights.s.green →
                        ((tmr?timeout → SKIP)
                        □
                        (car?passed → SKIP)))
S3 = (tmr!start.short → lights.m.red_yellow →
lights.s.yellow → tmr?timeout → SKIP)
S4 = (tmr!start.long → lights.m.green → lights.s.red →
tmr?timeout → SKIP)
```

The TLC we are currently developing uses Field Programmable Gate Array (FPGA) device. This is because in recent years FPGAs have been included in the electrical engineering curriculum at Ruse University.

The reprogrammable nature of FPGA allows the students to experiment different design solutions at the stage of hardware/software partitioning. With the hardware/software partitioning of the design system an attempt for combining the advantages of microcontrollers and FPGAs is done. In this project a microcontroller PicoBlaze fulfils the functions of the Control block. PicoBlaze is a compact 8-bits microcontroller with RISC architecture, optimized for the families Spartan™-3, Virtex™-II и Virtex-II Pro™ [10]. A part of the integrated RAM (Block RAM) memory in FPGAs is used as a program memory of the microcontroller. Every instruction of the microcontroller is executed in two cycles as the performance of the PicoBlaze varies from 44 to 100 MIPS depending on the target FPGA family and speed grade. Xilinx provides the PicoBlaze as a source-level VHDL file.

PicoBlaze doesn't have a timer and that's why the module Timer of the project is hardware design using FPGA resources.

The control program module of the PicoBlaze is coded in accordance with the algorithm on Fig.3. Assembling the program, a VHDL description of the ROM memory is generated.

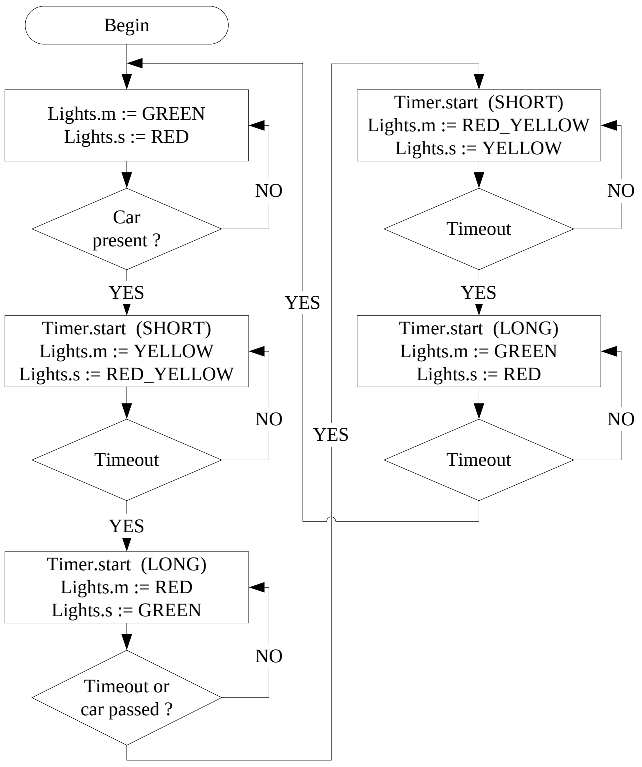

Fig. 3. TLC Flow chart

The project is developed using the Xilinx's WebPack ISE development environment [11]. As it is shown in the hierarchy structure of the project (Fig. 4), the VHDL modules are combined and presented at the highest level with a schematic [12].

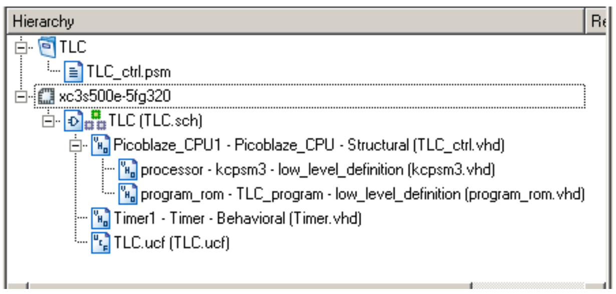
```
Hierarchy
TLC
  TLC_ctrl.psm
xc3s500e-5fg320
  TLC (TLC.sch)
    Picoblaze_CPU1 - Picoblaze_CPU - Structural (TLC_ctrl.vhd)
      processor - kcpsm3 - low_level_definition (kcpsm3.vhd)
      program_rom - TLC_program - low_level_definition (program_rom.vhd)
    Timer1 - Timer - Behavioral (Timer.vhd)
    TLC.ucf (TLC.ucf)
```

Fig. 4. Project hierarchy

For the project implementation the Spartan 3E Starter Kit, including FPGA circuit XC3S500E is used [13].

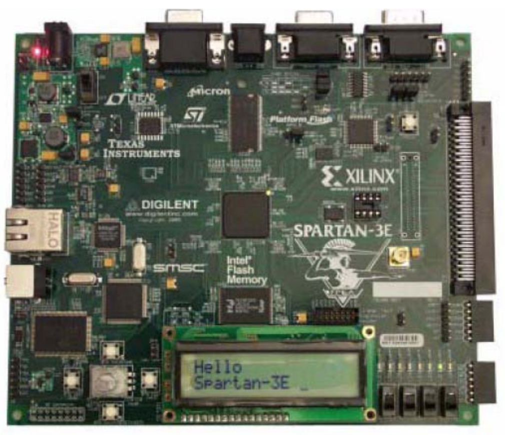

Fig. 5. Spartan 3E Starter Kit

The occupied resources of the FPGA circuit from the project are given in the next table:

Map report			
Number of occupied Slices:	138 out of	4,656	2%
Number of bonded IOBs:	9 out of	232	3%
Number of Block RAMs:	1 out of	20	5%

Conclusions

This approach has been discussed with the students in terms of its usefulness for obtaining practical skills and structured thinking in the process of embedded systems design using the Hardware/Software Co-design.

The reconfigurability of FPGA makes them ideal devices for educational purposes as it allows the students to iterate in their design tasks.

The Hardware/Software Co-design approach in embedded systems design allows the students to implement their knowledge obtained from various subjects and offers more opportunities for course and diploma projects.

References

1. **Nooshabadi S., Garside J.,** Teaching Embedded Systems Design – An International Collaborative Project // 35th ASEE/IEEE Frontiers in Education Conference. – Indianapolis, 2005.
2. **Shaout A., El–Mousa A., Mattar K.** Specification and Modeling of HW/SW CO–Design for Heterogeneous Embedded Systems // Proceedings of the World Congress on Engineering (WCE 2009), 2009. – Vol I.
3. **Cai L.** Estimation and Exploration Automation of System Level Design. – Ph.D. Dissertation, Department of Information and Computer Science. – University of California, Irvine, 2004.
4. **O'Nils M.** Specification, Synthesis and Validation of Hardware/Software Interfaces. – Doctoral thesis, Department of Electronics. – Royal Institute of technology, Stockholm, 1999.
5. **Carreras C., Lopez J., Lopez M., Delgado–cloos C., Martinez N., Sanchez L.** A Co Design Methodology Based on Formal Specification and High Level Estimation // Proceedings of the 4th International Workshop on Hardware/Software Co–Design, 1996. – P. 28–35
6. **Niemann R.** Hardware/software co–design for data flow dominated embedded systems. – Boston: Kluwer Academic Publishers, 1998.
7. **Cesario W., Baghdadi A., Gauthier L., Lyonnard D., Nicolescu G., Paviot Y., Yoo S., Jerraya A., Diaz–Nava M.** Component–Based Design Approach for Multicore SoCs, in Proceedings of 39th Design Automation Conference (DAC02), 2002. – New Orleans, USA. – P. 789 – 794.
8. **Sun J., Yang L., Dong J.,** Model Checking CSP Revisited: Introducing a Process Analysis Toolkit // Proceedings of the Third International Symposium on Leveraging Applications of Formal Methods, Verification and Validation. – 2008. – P. 307–322.
9. **Hoare R.** Communicating Sequential Processes. – PrenticeHall, 1985.
10. PicoBlaze 8–bit Embedded Microcontroller User Guide. – Xilinx, 2005.

11. **Parnell, K., Mehta, N.,** Programmable Logic Design Quick Start Handbook. – USA: Xilinx, 2004.
12. **Perry, D.,** VHDL, 3rd ed. – USA: MacGraw–Hill Inc., 1998.
13. Xilinx CPLD and FPGA devices website. [Online: http://www.xilinx.com/products/devices.htm].

Received 2010 04 29

N. Bencheva, N. Kostadinov, Y. Ruseva. On Teaching Hardware/Software Co-design using FPGA // Electronics and Electrical Engineering. – Kaunas: Technologija, 2010. – No. 6(102). – P. 91–94.

At Ruse University in the Faculty of Electrical Engineering, Electronics and Automation the students are offered several introductory courses that focus on microcontroller-based systems, embedded programming and Programmable Logic Design. In the process of design, configuration, and programming of microprocessors it is hard for the students to optimize the division of functions between hardware and software. One of the reasons is the growing number of advanced programming techniques and design principles, as well as sophisticated Integrated Development Environments. The courses have a fixed number of hours and the teacher cannot extend the content of the lectures and the labs. In order to improve the students' knowledge and skills in the process of embedded systems design, the Hardware/Software Co-design is used in the teaching process. The paper considers one example of implementation of this approach in the design of a traffic light controller. The project is based on the Intellectual Property Core of PicoBlaze microcontroller, embedded within Xilinx Field Programmable Gate Array Families. For the project development Evaluation Board Spartan 3E including XC3S500E FPGA device is used. Ill. 5, bibl. 13 (in English; abstracts in English, Russian and Lithuanian).

Н. Бенчева, Н. Костадинов, Й. Русева. Обучение хардуер-софтуерного совместного проектирования на основе применения ПЛИС // Электроника и электротехника. – Каунас: Технология, 2010. – № 6(102). – С. 91–94.

В университете города Русе, на факультете Электротехники, электроники и автоматики студентам предлагаются несколько базовых курсов о микроконтролерных системах, встроенное программирование и проектирование на основе программируемых логических интегральных схем (ПЛИС). В процессе проектирования, конфигурирования и программирования микропроцессорных систем студентам нелегко оптимизировать распределение функций между программным и аппаратным обеспечением. Одна из причин для этого является нарастающий количество современных методов программирования и принципов проектирования и сложные интегрированные среды разработки. Курсы имеют фиксированное количество учебных часов и преподаватель не может растянуть содержание лекций и практических занятий. Чтобы улучшить знания и умения студентов в процессе проектирования встраимаевых систем используется совместное аппаратно-программное проектирование. В статье приведен пример о приложении этого подхода при проектировании светофора. Проект базируется на Intellectual Property Core микроконтролера PicoBlaze встроенный в ПЛИС фирмы Xilinx. В процессе проектирования использован Evaluation Board Spartan 3E включающий ПЛИС XC3S500E. Ил. 5, библ. 13 (на английском языке; рефераты на английском, русском и литовском яз.).

N. Bencheva, N. Kostadinov, Y. Ruseva. Techninės ir programinės įrangos bendro projektavimo mokymas taikant FPGA // Elektronika ir elektrotechnika. – Kaunas: Technologija, 2010. – Nr. 6(102). – P. 91–94.

Rusės universiteto Elektros inžinerijos, elektronikos ir automatikos fakultete studentams siūlomi keli įvadiniai kursai. Juose daug dėmesio skiriama mikrovaldikliams ir jų sistemoms, įterptinėms sistemoms bei programuojamosios logikos projektavimui. Studentams projektuojant, konfigūruojant ir programuojant mikrovaldiklius sunku optimizuoti funkcijų padalijimą tarp programinės ir techninės įrangos. Taip yra dėl nuolat tobulėjančios pažangios programavimo technikos ir projektavimo principų skaičiaus didėjimo. Studentų žinioms ir įgūdžiams pagerinti mokymo procese bei optimizuoti naudojamas įvadinis kursas „Techninės ir programinės įrangos projektavimas“. Pateikiamas vienas pavyzdys – mikrovaldiklio, skirto šviesoforams reguliuoti, kūrimas. Il. 5, bibl. 13 (anglų kalba; santraukos anglų, rusų ir lietuvių k.).