EXPERIENCES IN TEACHING RECONFIGURABLE COMPUTING AT ERLANGEN UNIVERSITY

Christophe Bobda
Departement of Computer Science 12
University of Erlangen-Nuremberg
Am Weichselgarten 3, 91058 Erlangen, Germany
bobda@cs.fau.de

Abstract: The last decade has experienced an increase interest on reconfigurable computing (RC). This evolution were boosted by FPGAs which have grown from simple glue logic elements to complex devices able to implement several complex hardware applications and be partially and dynamic reconfigured at run-time. Despite the high number of courses offered in the last years in reconfigurable computing, we could not find a course covering all aspects of reconfiguration. The large majority of reconfigurable computing course are limited to FPGA programming. This is in part due to the fact that no textbook actually exists in this area. In this paper, we present our experience in providing a course in RC from the scratch. Our goal in designing this course is to provide a strong theoretically and practical background to students by covering all aspects of RC as usually reflected in conferences and industry. Therefore, our course is a mixture of theory and practical exercises to be done in lab.

Keywords: Reconfigurable Computing, FPGA, Teaching

1. INTRODUCTION

General purpose computing is based on the traditional Von Neumann Computing model. A given program is computed by a sequential execution of its instructions. Instructions as well as data are stored in a memory. An instruction is fetched from the memory and decoded. The required operands are then collected from the memory before the instruction can be executed. After execution, the result is written back in the memory. This computation paradigm involves five steps in general: Instruction Read, Decoding, Reading of Operands, Execution, and

Write Result. The main advantage of the Von Neumann computing paradigm is that it can be used to program almost all existing algorithms. The architecture has a high degree of flexibility. Nevertheless, each Algorithm to be implemented on a Von Neumann computer has to be coded into its sequential computational behavior. I.e. *the algorithm has to be adapted to the hardware* to be sequentially executed even if it is inherent parallel. The Drawback of the Von Neumann computers can be overcame by directly implementing the best computational paradigm adapted to a given application in hardware. In this case, parallelism can be exploited to increase the performance. For each function to be implemented, the optimal hardware, usually called an Application Specific Processor (ASP), will be built. We said that *the hardware is adapted to the algorithm.* While the main advantage of ASPs is their speed, ASPs are not flexible at all. A function implemented in an ASP cannot be changed anymore. Ideally, we would like to have the advantages of the General Purpose Processors (GPP) and ASPs combined in one device. That means we will like to have a device which is flexible enough to implement any kind of algorithm and efficient enough to run very fast. This degree of performance and flexibility can be reached only if *the device can dynamically adapted to algorithms.* Devices, which are able to be dynamically adapted to match the computation paradigm of a given application are called reconfigurable devices, adaptive devices or Reconfigurable Processing Units (RPU). The adaptation of the hardware to new algorithms is done by changing the device configuration.

2. COURSE PURPOSE

With the rapid changing in the area of reconfigurable computing. the purpose of the course "reconfigurable computing" is to provide to students, the necessary knowledge for understanding and designing reconfigurable systems. The course provides a strong theoretical and practical background to students by covering all aspects of RC as usually reflected in conferences and industry. This include the architectures of reconfigurable systems, the algorithms and the applications. A considerable part of the course is devoted to lab, which are done on the basis of the Xilinx FPGAs, one of the few FPGAs on the market to support partial reconfiguration.

3. COURSE CONTENT

The course is intended for graduate students in computational engineering with focus on microelectronics, information technology or sensor technology. However, the course is also open to computer science

students who have completed introductory digital logic design. It is organized in two parts: In the first part the readings are done while in the second part exercises and lab alternate. The reading as well as the exercises are done in a frequency of two hours per week each. Exercises are offered either as theoretical exercises for a better understanding as the theoretical materials or as lab for dealing with reconfiguration in the praxis. The parts covered are:

- **Architecture of reconfigurable systems**: In this section, technology as well as the coupling possibilities of reconfigurable systems is considered, from the fine grained look up table (LUT) based reconfigurable systems like the field programmable gate arrays (FPGA) to the new coarse grained technology.
- **Design and implementation**: This section considers the implementation of reconfigurable system. It covers the steps needed (design entry, functional simulation, logic synthesis, technology mapping, place and route and bit stream generation) to implement today's FPGAs. We focus deeply in logic synthesis for FPGAs, in particular LUT technology mapping.
- **Temporal partitioning**: This section considers the high level synthesis for reconfigurable system. It covers the implementation of large functions which cannot fit in one FPGA. Several temporal partitioning techniques are explained.
- **Temporal placement**: In this section, stand alone reconfigurable systems are considered. We assume that a kind of OS for reconfigurable system is in charge of managing the resources of the system and allocate space on a device for the computation of incoming tasks. We therefore present several temporal placement approaches for on-line placement.
- **On-line and Dynamic Interconnection**: Modules dynamically placed at run-time on a given device need to communicate together and also exchange data with ff-chip devices. Therefore, they dynamically create a need of communication on the chip. This chapter reviews and explains the different approaches to solve this dynamic intercommunication need.
- **Designing a reconfigurable application on Xilinx Virtex FPGA**: This section considers the implementation of reconfigurable system. Apart from the steps needed (design entry, functional synthesis, technology mapping, place and route and bitstream generation) to implemented today's FPGAs, the generation

of partial bitstreams for component to be placed at run-time on the FPGA is considered. This is done using the modular design flow.

- **System on programmable chip**: System on programmable chip is a hot topic in reconfigurable computing. This is mainly the integration of a system made upon some peripheral (UART, Ethernet, VGA, etc.) but also computational (Coding, filter, etc.) hardware modules on one programmable chip. We present the current usable solutions: The Xilinx EDK, the Altera Excalibur and the Atmel System designer
- **Applications**: This section presents applications of reconfigurable systems. It covers the use of reconfigurable system in computer architecture (rapid prototyping, reconfigurable supercomputer, reconfigurable massively parallel computers) and algorithm better adapted for reconfigurable systems (distributed arithmetic, network packet processing, etc...)

4. LABORATORY SUPPORT

The laboratory includes:

- Capturing a design using VHDL: This is done using the Synopsys FC2 design environment. The produced EDIF-file is used as entry for the Xilinx ISE-tool
- Implementation of a design using Xilinx ISE. Here we use the Xilinx ISE 4 for synthesizing the EDIF-design produced in the previous step. The result is a bitstream used for configuring the FPGA.
- The Modular deign. Here we use the Xilinx ISE 6, which includes the tool for the modular design. Students learn how to design a large project in a team of engineers. They also learn how to constraint a components to a given location, how to place bus-macros among component to insure a signal integrity on reconfiguration, and how to produce the partial bitstream for each reconfigurable module.

Due to the goal seeked, we use three different platforms.

1 The first one is a Digilab XLA, featuring a Spartan XLA. Because this device is not supported in the current Xilinx CAD tool (the ISE), we must first provide an EDIF implementation of our design. That is why we first use the Synopsis FC2 to produce the EDIF-files.

2 The second platform is the Xilinx-Digilab XCR featuring a Spartan 3 FPGA. Since the device on this is supported in the newest Xilinx tool, the complete design (VHDL entry, Synthesis, download) is done in just one environment.

3 The third platform used is the RC200 board of Celoxica. This board is used here for pure demonstration. It is the only board available in our Lab which support partial reconfiguration. Since only one board is available. Students have to design separately, and sequentially implement on the board.

The software used as well as part of the hardware are fully or partly provided by the company Xilinx as part of its university programs. Apart from the first lab done on Unix workstations, all the remaining labs were done on Windows PCs.

5. COURSE MATERIAL

All the course material (slides, assignments and tutorials) are disseminated via the world-wide web [1]. Students generally utilize a split-screen approach with one window containing tutorial instructions and a second window exercising the appropriate CAD tool needed for the design. A script is also provided to students as a preliminary version of a textbook, thus the web access to the script is restricted. All the remaining materials are free for access.
The lab consists of two designs. The first one is the implementation of a digital alarm clock. This lab helps the student to understand the relation between a circuit running in FPGA and the peripherals on the board. The second design is a traffic light control (TLC). This design is divided in three blocks. The first one is a finite state machine (FSM) for interpreting the pedestrians need, the second one is a VGA module in charge of displaying a traffic light infrastructure on the screen. The third module is a light visual (LV) module to construct the infrastructure by computing the colors to be display at a given location by the VGA module. The FSM sends the command on the behavior to the LV which then compute the correct color according to the pedestrian need.

6. EXPERIENCES

In the first lab, students found it difficult to exchange the files from one tool (the Synopsis FC2 to another (the Xilinx ISE). Also, this process was time consuming. With the tools remotely installed, the designs could not be done in the foreseen time. Once we move to the integrated

Windows design in the remaining labs, the students felt more comfortable and all the designs could be done on time with many iterations

REFERENCES

[1] Christophe, Bobda, "Reconfigurable Computing", http://www12.informatik.uni-erlangen.de/edu/rc/