FPGA Applications in Education and Research

Andreas Koch, Ulrich Golze

Technical University Braunschweig
Abteilung Entwurf integrierter Schaltungen (E.I.S.)
Gaußstr. 11, D-38106 Braunschweig, Germany
e-mail: koch@eis.cs.tu-bs.de

Abstract

This paper reports on the application of FPGAs in our education and research. It describes the organization of FPGA-specific lectures and the accompanying labs. Experiences with the hard- and software environment are summarized. A general purpose FPGA-based co-processor expansion card for standard workstations is presented and an overview over its architecture and software interfaces is given. Some possible applications for this expansion card for further research and as a platform for teaching hardware-software co-design are pointed out.

1 Introduction

FPGAs are gaining importance both in commercial as well as research settings. The former appreciate the short turn-around times, the lack of NRE costs for small volume production and the easy prototyping. Since the technology is far more affordable than custom-manufactured ASICs, even smaller companies can take advantage of the capabilities of large-scale integration. Thus, the probability of a student after graduation working on FPGA-based circuits is higher than that of him developing a "real" full- or semi-custom chip.

While researchers also enjoy the aforementioned benefits, they tend to place more emphasis on how to apply FPGAs in novel ways, such as the creation of problem-specific computing architectures.

In order to do this development justice, the theory and practice of working with FPGAs have become a major part of the curriculum as well as the research at the department E.I.S. of the TU Braunschweig. This paper will present some of the experiences gained during the courses and labs. Furthermore, we report on the progress of a hardware project which should have interesting applications both for educational and research purposes.

2 Lectures on FPGAs

In addition to the standard lectures on digital logic and VLSI design, students are offered two week

lectures presenting a general introduction to FPGA design and an overview over different FPGA architectures before a single architecture, the Xilinx LCA, is examined in detail.

2.1 FPGA Overview

The overview begins with a retrospective examining the development of programmable logic from roots such as PROM and PLA to modern FPGAs with a short digression presenting MPGAs. The lecture continues with a presentation of the elements of an abstract FPGA. Afterwards, concrete implementations of these abstract elements are examined. Examples include Mux-, PLD-, transistor- and LUT-based logic blocks. Next, common routing topologies and programming technologies are explained and compared. This general part of the lecture closes with a discussion of the design flow followed when working with FPGAs. The different tools such as partitioning and place and route and their intermediate results are described.

2.2 Xilinx Architecture

The lecture then specializes on the Xilinx LCA. Due to the lack of appropriate CAD tools, only the XC3000 architecture was discussed in the last semester. However, since the situation has been remedied with the availability of more advanced CAD tools through Eurochip, we plan to include the XC4000 architecture the next time.

While a detailed knowledge of the underlying chip architecture might be considered superfluous by some, we feel that it is very important to provide the students with thorough understanding of the chip specifics, enabling them to gain insight into tool algorithms and to consider the implications of changes on the schematic level on the implemented circuit. Furthermore, the experience obtained at this stage will be useful later when different FPGAs have to be compared and evaluated for suitability to a given application.

This part of the lecture uses transparencies from the Xilinx Programmable Gate Array Training Courses, offered to educators by the Xilinx University Program. We found them extremely helpful and suggest, that any site intending to offer Xilinx-based courses establishes firm relations with the program, which provides a wealth of material to educators, ranging from sample project descriptions to lecture notes.

3 FPGA Labs

Students can put their freshly acquired understanding of FPGAs to work in the labs following the lecture. Since this was the first time we offered FPGA labs, only three groups of 2-3 students each participated and were warned, that unexpected glitches of hard- and software might occur during this trial run.

3.1 Work Environment

The labs used PC hardware (486/33-16 MB) under MS-DOS to run the Viewlogic Workview CAD package for schematic entry and simulation and Xilinx XACT for design implementation and LCA-level entry.

While MS-DOS does not provide the multi-user features (file protection, account management) that make Unix easy to administer in a lab environment, it performed adequately for our purposes. The computing power of the hardware was also sufficient for entry and simulation. However, when the designs were to be implemented (mapped, placed and routed), and the tools iterated over a design numerous times to improve chip performance, execution times between eight and sixteen hours were not uncommon. Since MS-DOS is only a single-tasking

system, the computer was unusable for other purposes during such a run.

The Workview software, on the other hand, was a pleasant surprise. Compared to Cadence ES2-EDGE, which we are using for semi-custom designs, it is extremely easy to use while still providing appropriate functionality. The students praised the software's ease of use and flat learning curve. This contrasts quite sharply with the comments regarding ES2-EDGE, which is often described as too complex and user-hostile.

XACT also performed flawlessly, neither the automatic design implementation tools nor the LCA-level editor posed any serious problems for the students.

3.2 Lab Procedures

The semester labs consisted of two smaller problems which had to be entered on the LCA level and one larger problem to be completed using schematic entry and simulation.

The two initial labs concerned the design and implementation of simple combinatorial circuits directly on the LCA with the students performing both the circuit partitioning, placement and routing manually. Apart from making the students more familiar with the Xilinx architecture, these labs served to demonstrate the flexibility of FPGAs in general, since the designs could be downloaded into the Xilinx XC3000 demonstration board directly after making modifications.

After this first practical exposure to FPGAs, the students began to work on the larger problem, the design of a four digit octal calculator. For this design, we emphasized careful planning and a structured, well documented design style accompanied by intensive simulation to avoid the "trial and error" design methodology commonly encouraged by the use of reprogrammable FPGAs. At this stage of the labs, the aspect of the subsequent FPGA realization was completely disregarded and the students had to work under the assumption that they were designing a real ASIC and serious design errors would cost their companies ten thousands of dollars and delay their market entry by months. No FPGAs were programmed while the students were designing and simulating. However, the prospect that all designs would be "fabricated", as demonstrated during the first phase, motivated the students considerably: To see a design performing as expected in the simulator is one thing, to play with a working calculator quite another. This contrasts with the traditional design labs, where only a few selected designs are submitted for fabrication and most students never see a working chip.

At the end of the semester, the moment of truth came, when the implemented designs were downloaded into a special calculator test board. This board makes a keyboard and five digit display available to the student chips. The interface and communication protocols between chip and board were specified as part of the initial problem description, thus modelling the real world situation that no chip works stand-alone but is always integrated into a larger system. To the students' delight, two of the three designs submitted worked immediately after download, the third one had minor problems when certain operation sequences were entered.

4 Research and Development

In addition to the rewarding application of FPGAs for teaching chip design, they are also used in research and development projects at the department. We are currently working on an FPGA-based co-processor card for SBus workstations (such as the SUN SPARCstation series of machines).

Our proposed architecture will be suitable as a customizable co-processor for the acceleration of specific applications as well as an evaluation

platform for solutions obtained using hardware-software co-design. In a secondary function, it will be able to act as a flexible I/O subsystem. The architectures we considered are based on a small number of FPGAs. While it is certainly possible to obtain more impressive results by employing dozens of large FPGAs in parallel, this approach is not economically feasible for general use. Our aim is to demonstrate that even a small co-processor can be used with good results when added to a standard workstation.

4.1 The SUN SBus

The SBus was developed by SUN Microsystems as an I/O bus to replace the VME bus in their desktop workstations. While it is predominantly used in SUN SPARC-based workstations and their compatibles, the bus is processor independent. It is a high bandwidth/low latency design supporting multiple bus masters and automatic virtual-physical address translation as well as fast burst data transfers. Further features include flow control and retry mechanisms for slave devices, dynamic bus sizing, central bus arbitration with geographic card addressing and flexible interrupt management. The bus has a CMOS-compatible interface and is based on simple, synchronous protocols.

4.2 Co-Processor Architecture

Our proposed architecture is defined by three major design decisions.

First, we decided to add on-board RAM to the expansion card. While it would have been easily possible to directly access the host memory, the bandwidth constraints of the SBus compared to those of a custom processor-memory bus make accesses to the on-board memory far more efficient. This on-board RAM can then be accessed by the host using fast-burst data transfers.

Second, in order to overcome the capacity restrictions of a single LCA, we decided to assign the tasks of data and address management to different but closely coupled LCAs. This allows us to implement a 32 bit data path in the main FPGA (termed user FPGA) and shift all addressing logic to one or more FPGAs containing data paths for address manipulation (termed service FPGAs). Since we now have dedicated LCAs for address operations, we can easily implement complex address generators offering, for example, fast indirect addressing and address arithmetic capable of increment/decrement and scaled indexed addressing as well as the generation of non-linear address sequences. The latter can be especially helpful for the efficient implementation of certain matrix operations. We feel that this is a very natural way of partitioning the target algorithms we envision. The communication between user and service FPGAs will be facilitated by allowing both chips access to the shared data bus and a relatively small (≈10) number of dedicated bi-directional control lines (see Fig. 1).

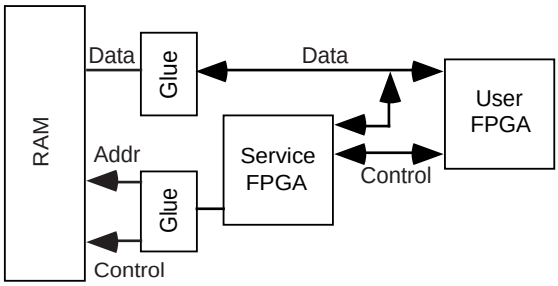

Figure 1 FPGA/RAM Interface

Third, the basic architecture was expanded by the addition of a second RAM subsystem. The gains offered by this approach far outweigh the added complexity. Apart from the additional gate capacity afforded by the second service FPGA, this co-processor can access data in two different RAM banks in parallel. This makes the architecture far

more interesting to experiment with, since it offers a distinct advantage over conventional CPUs, which can only use one RAM access path at a time. This expanded architecture is sketched in Fig. 2. Note that an additional FPGA is used as a master controller for the whole card, managing SBus transactions and the configuration of the user-programmable data path.

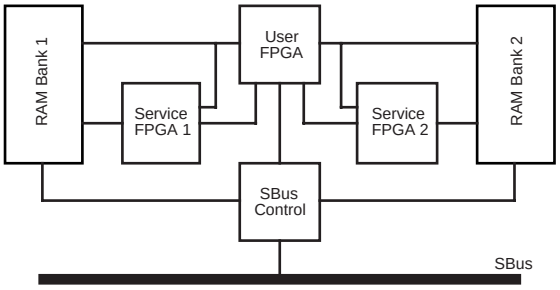

Figure 2 Co-Processor Architecture

4.3 Special Hardware Features

In order to run each FPGA configuration at its optimum speed, the board also includes a programmable clock generator capable of generating frequencies from a few hundred kHz up to 80 MHz with fine tuning to almost any target frequency in between. Thus, FPGA applications can be clocked independently of the SBus clock, slower for more complex circuits, faster for simpler designs.

Configuration information for the complete board is stored in a dedicated RAM bank. Since this bank has sufficient capacity to hold multiple complete configurations for all three FPGAs plus appropriate parameters for the programmable clock, later reconfigurations using the stored data place no load on the SBus at all. While the current configuration can naturally be selected by user software, an important feature of the architecture is the ability of the co-processor to reconfigure itself at run-time without intervention of software or the host CPU. This capability exemplifies our approach of sharing the same hardware not only between different programs and algorithms, but also between phases of a single algorithm, thus maximizing the efficiency of co-processors with limited gate capacity.

4.4 Software Interface

User software communicates with the co-processor through a device driver, which is integrated into the Unix kernel. Only this driver may directly access the card, thus conflicts caused by multiple processes trying to use the same card concurrently can be avoided. Furthermore, the driver supervises the configuration process, ensuring that only complete bit streams are downloaded into the configuration memory, and the driver allows user software and the co-processor to execute in parallel, with the co-processor notifying user programs of the end of calculations through a software interrupt signal.

User programs transfer data to and from the card by advising the host memory management unit (MMU) to map the card on-board RAM into the virtual memory space of the user process, allowing the user to access card RAM regardless of its physical location. Since all accesses are still supervised by the MMU, the memory protection remains inviolate. Thus, algorithms executing on the co-processor cannot bypass system security measures.

4.5 Applications and Future Directions

The card is intended as a test platform for hardware-software co-design and will eventually be used to establish lectures and labs on the subject. The FPGA co-processor satisfactorily merges both VLSI and computer science aspects, a combination which fits extremely well in the agenda of our department (being a branch of the computer science

faculty). Student labs could consist of a general problem specification which had to be implemented initially as a software-only solution to validate the algorithms proposed, which is then profiled to determine bottlenecks and finally partially shifted onto a problem-specific co-processor. This procedure should be interesting for computer science as well as electrical engineering students, exposing them to the intricacies of an advanced multi-user, multi-tasking operating system and a non-standard processor architecture.

Sample applications currently being investigated for successful implementation on our proposed architecture include general purpose algorithms such as data compression as well as more specialized circuits dealing with computer vision and system security. In the current absence of CAD tools tailored to our co-processor architecture, we are using the Xilinx X-BLOX synthesis tool to implement the designs, but further research efforts will concentrate on the seamless integration of hardware facilities such as those provided by the card into augmented high-level languages and synthesis optimized for the proposed architecture.

5 Conclusions

We feel that FPGAs are an extremely valuable tool in teaching VLSI design. While the traditional techniques of full- and semi-custom design certainly have their places for analog, high-performance or complex applications, the prospect of putting "their" chip to the acid test of a real hardware environment motivates students tremendously. Other aspects of FPGAs also make them attractive for use in teaching, such as their low cost and reusability compared to "real silicon", but this increased motivation, created by the guarantee that all chips designed will be "fabricated" at the end of the semester, should not be underestimated.

The dynamic nature of the FPGAs also offers many opportunities in the exploration of novel computing structures and problem-specific processors. The expansion card design presented is only a first step in this direction. Apart from providing a broad variety of areas for further research, ranging from the design and implementation of application specific co-processors to the development of specialized CAD tools, we are also looking forward to the educational possibilities offered by a working configurable co-processor: It could be a good teaching vehicle for hardware-software co-design, featuring sophisticated hard- and software interfaces and thus exposing students to the best of both worlds when designing interacting programs and circuits while still considering the constraints of an existing environment.