A Hardware Environment for Prototyping and Partitioning Based on Multiple FPGAs*

M. Wendling [1,2] W. Rosenstiel [1]

[1] Computer Science Research Center at the University Karlsruhe (FZI)
Haid-und-Neu-Straße 10-14, 76131 Karlsruhe, Germany
[2]Ecole Nationale Supérieure de Physique de Strasbourg (ENSPS)
Parc d'innovation, Bd Sébastien Brandt, 67400 Illkirch, France

Abstract

This paper presents a multiple-FPGA-based experimentation board. The problem to be solved is that of implementing a circuit into a set of FPGAs. This board provides a hardware environment that can be used either as a prototyping board, allowing to test circuit designs, or as a hardware support for practically test and optimize partitioning algorithms. The former use address the problem of rapid prototyping while the latter is a contribution to the domain of hardware/software codesign. The environment consists of the printed circuit board itself, a set of Xilinx FPGAs, static RAM, a parallel and a serial interface. In this paper, the structure of the board will be presented, by pointing out its simplicity, power and flexibility.

1 Introduction

Since the first appearance of FPGAs in the mid-eighties, the interest for this new technology has been continually growing. The main reasons for this are the relatively high complexity and the user-programmability of these devices, allowing a designer to implement a circuit by himself and to do a fast redesign in the case the circuit did not work as expected. For an analysis of FPGA architectures see [1]. The first problem that has been addressed concerning FPGAs was the problem of technology mapping. It consists of transforming a circuit description made up of standard logic gates and RTL elements into a description using the internal elements of an FPGA. There has been a lot of research done on this subject. For more informations, see [2] [3]

After the problem of technology mapping has been very well addressed, a new problem has appeared and is getting more and more into the focus of interest: It concerns the problem of implementing very large circuits into FPGAs. New generations of FPGAs have appeared, faster, more complex, with more and more gates on a chip, thus allowing designers to implement more and more larger circuits. However, despite the degree of sophistication and the number of gates that FPGAs have reached, they are still too small to implement larger circuits, with several ten or hundred thousands of gates. Consequently, the idea has appeared to put a set of FPGAs onto one board, to realize an interconnecting structure between them, to partition a large design into smaller ones that can fit into one single FPGA, and to implement each of these small designs into the different FPGA devices of the board.

This problem can be divided into two parts. On the one side there is the software part, which concerns the partitioning of the circuit. It has the task of dividing the circuit into several pieces according to some criteria, like e.g optimizing the propagation delay or minimizing the number of interconnections between two pieces of circuit. Partitioning algorithms are well konwn [4] but need to be adapted to the technological characteristics of FPGAs. On the other side there must be some hardware support present, for practically implement a circuit into a set of several FPGAs. This hardware must provide enough gates to allow the implementation of large designs, as well as an efficient interconnection structure.

This paper presents an experimental hardware environment, made up of a printed circuit board on which are placed a set of Xilinx FPGAs for the computational and the control part, some memory for the data storage, and a serial and parallel interface through which the board can be driven. Unlike other FPGA

*This work is supported by a research grant from the University of Strasbourg, France

boards [5], this one is not dedicated to a specific application, but has the purpose to be universal, allowing designers on the one side to implement various types of circuits, on the other side to test the efficiency of partitioning algorithms. Section two presents the requirements and specification of the board. The detailed structure of the board is then presented in section three. Section four describes the physical implementation of the board and the software environment used to realize it. Section five gives an overview of the frame within which this work has been done. Finally, section 6 will give a conclusion to this work.

2 Hardware requirements and structural specification of the board

This section will examine the required capabilities of the FPGA hardware environment. Since this environment is made up of several FPGA devices and is therefore closely related to the partitioning problem, the requirements can not only be expressed in terms of hardware, but also must consider the architectural side.

- In order to implement large circuits, it is necessary to have sufficent gate complexity as well as a large number of I/Os. There are two ways to meet this requirement: the board must be made up of either many FPGA devices with average gate and I/O capacity or only a few ones with larger gate and I/O capacity. It will be shown in section 3 that newly appeared FPGAs on the market have led to choose the second possibility.
- Since this board is intended to serve as a hardware support for implementing various circuits and test different partitioning algorithms, the FPGAs which it is made up of must be choosen among the re-programmable FPGAs, that is those which are RAM-based.
- The hardware environment must also provide some memory. On the one side this allows FPGAs to take initial values out of it or to store intermediate results. On the other side, it gives the board the ability to work standalone, without any data exchange to the host while computing, which would inevitably lead to a decrease in term of speed.
- The board is intended to be driven by a host computer, therefore there must be an interface providing all necessary signals for configuring the FPGAs, preloading the memory, reading out the results, and controlling the general behaviour of the board from the host.
- Each circuit that is intended to be implemented onto the board has different requirements to the hardware. In order to improve the flexibility and to adapt the board to each circuit, it seems interesting to implement the controller of the board into another FPGA, thus allowing an easy reprogramming.
- Since this board is intended to be an experimental board, its structure has to be simple yet powerful. A structure that is simple will also ensure that the board will be easily expandable.
- With regard to the architecture of the board, it seems reasonable to provide as many intercommunication channels between FPGAs as possible. The wider the datapath, the easier will be the partitioning.
- The memory should be shared so that the FPGA devices can work on the same data. This implies that there will be a common data and address bus. As a consequence of this, since every FPGA will be able to access the same memory, it will be necessary to provide an arbitration logic, which will grant the memory access according to some priorities.
- The interface has to be built so to connect the board either to a PC or to a SUN workstation, thus increasing its flexibility. This specification makes it impossible to use the internal bus (either ISA or SBUS) of these two computer types. The solution to this problem consists in using the interfaces that these two computers have in common, that means the serial and the parallel interfaces.

3 Composition and structure of the board

This section presents the architecture of the board by describing on one side the elements composing it, on the other side the interconnection structure between them. It will be shown how the specification of section two were met. Figure 1 depicts the overall structure of the board. It consists of several functional units: the interface, the memory part, the controller and the FPGA block. Each of them will be described in the following subsections.

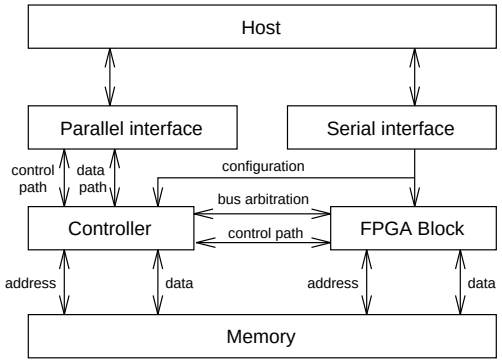

Figure 1: Overall structure of the board

3.1 The interface

One of the specification presented in section two prescribed that the interface must be built to connect the board either to a PC or to a SUN workstation. This requirement prohibits the use of the internal busses of these two computer types. The only way to implement this is to use the parallel and serial interfaces which are common to these two types of computer. As shown in Figure 1 the serial interface is used for the configuration of the FPGA block as well as the controller, since the latter is made up of an another FPGA device. This configuration is done by downloading a bitstream into each of the FPGAs.

Since the serial interface is only used for configuring the FPGAs, the board is controlled by the host through the parallel interface. There is however a counterpart to the flexibility provided by the use of the serial and parallel interfaces: indeed, due to the fact that the parallel interface of the host has a limited number of control and data lines, it would be quite difficult to control every part of the board from the host itself. The solution to this problem consists of using the parallel interface solely for the communication between the host and the controller. It is then the task of the controller to "interpret" the few commands it receives from the host and to "expand" them in commands that are comprehensible by the other parts of the board. The FPGA block and the controller must obviously be configured through the serial interface before any command can be sent by the host to the controller through the parallel interface.

3.2 The memory part

The memory part consists of 64 Kbytes static RAM with a data width of 16 bits. Static RAM has been chosen because of its faster access time and its simpler connection structure than dynamic RAM. As shown in figure 1 the memory can be addressed by both the controller and the FPGA block, that means by every single FPGA device of this block. As it was explained previously, the data exchange between the memory and the host is solely ensured by the controller. It is the task of the controller to preload the memory with initial data and to read out the final results. The memory can be accessed by every FPGA of the FPGA block through common data and address busses with respect to the permissions granted by the arbitration logic of the controller.

3.3 The controller

The task of the controller is twofold: on the one side it is responsible for the communication between the host and the board by getting the commands from the host, interpreting them according to its programming and dispatching commands to the other parts of the board. On the other side, once the FPGAs are all configured and the data loaded into memory, the board is given the control to itself and is then working standalone; the task of the controller is then mostly to arbitrate the memory access requests coming from the FPGA block.

The controller is implemented into a Xilinx XC 3090 FPGA, thus respecting the specification. This allows the designer to adapt the controller to the application being implemented into the FPGA block. It could seem that the counterpart of this flexibility results in additional work for the designer who has to implement a new controller for every new application. But this is actually not the case. Despite the fact that the controller is programmable, it however has an overall structure, which is shown in figure 2.

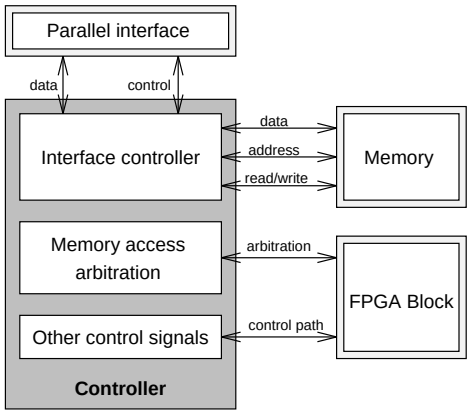

Figure 2: Controller

3.3.1 The interface controller

The interface controller is responsible for tranfering data from the host to the board memory and vice versa. Since the parallel interface has an eight bits wide datapath while the memory has sixteen bits wide address and data busses, some multiplexing and demultiplexing of these data must be performed. To achieve this, the interface controller is simply made up of a sixteen bit counter which is responsible for counting up the addresses, as well as some buffers and latches which are in charge of the multiplexing / demultiplexing part.

As the board and the host do not work at the same clock rate, all the communication between them is asynchronous, which means that for each data exchange there must be some handshake. At first glance, this seems to be a factor slowing down the performance of the board. But the data exchange between the host and the board is only occuring before the computation, for the purpose of preloading the memory, and after the computation, for reading out the results. Since there is no data exchange with the host while the board is computing, the asynchronism of the host-board communication is not a problem.

3.3.2 The memory access arbitration

As it was said previously, the second main task of the controller is the arbitration of the memory access. Indeed, since the FPGAs of the FPGA block share the same data and address busses, there must an arbitration logic which grants the access to the memory according to some priority schedule. The method that has been chosen to achieve this is a local bus arbitration with central arbiter [6]. The functioning mode is very simple: each FPGA has a priority which is shifted on a circular way each time one of the FPGA got access to the memory. If an FPGA needs memory access, it sends a bus request to the controller. If this FPGA is the only one requiring the memory, no arbitration is necessary and the memory access is granted to it. If more than one FPGA require memory access, the one with the highest priority obtains it. The only problem that could arise with this simple method is the possibility for an FPGA to keep the memory access as long as desired once it has been granted to it, preventing the other ones to get it also. If this should occur, then it would be necessary to implement a more complicated arbitration method avoiding this problem. Since the FPGA serving as controller is a Xilinx XC 3090, with a gate capacity of 5000, this should be possible without further complications.

3.4 The FPGA block

The FPGA block constitutes the computational core of the board. Providing that the application to be run on the board has been correctly partitioned, place and routed, it is then downloaded into the FPGA block which has the task to execute it. In accordance with the specifications, the board must have a comfortable gate capacity and a large number of I/Os. To achieve this, it has been decided to build the FPGA block from Xilinx XC 3195 FPGAs of the new 3100 serie. This family offers optimized performance relatively to the 3000 serie. The most interesting improvements of the XC 3195 are a higher clock rate and additional gate and I/O capacity. As depicted in figure 3, the FPGA block consists of a 2x2 array of XC 3195. The only direct connection of the FPGA block to the host is through the serial interface, for the purpose of configuration. Each FPGA device is connected to his two neighbours by thirty-two bit wide datapaths, thus respecting the specification of having a datapath as large as possible. The lower right FPGA on figure 3 is representative for each FPGA and shows the connections to the other parts of the board. The connection to the memory consists of the common data and address busses, each sixteen bits wide, and the read/write signals. The connection to the controller consists of the bus arbitration signals, which are specific to each FPGA, and a common control path.

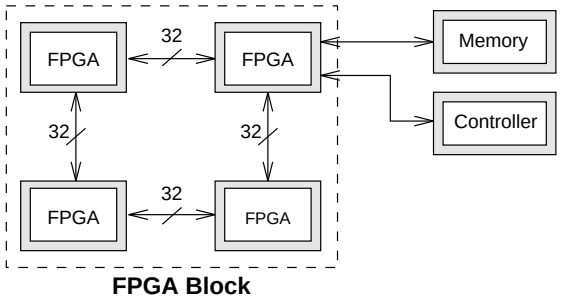

Figure 3: FPGA block

4 Realization and characteristics of the board

4.1 Design considerations

The realization of the printed circuit board, from the schematic entry to the layout, has been fully achieved with the Mentor Graphics Board Station software environment, running on a SUN workstation.

The configuration of the FPGAs is made in a slave serial mode. To achieve this, all the FPGAs have been daisy chained as recommended in [7], getting their configuration bitstream from the host through the serial interface.

One of the very interesting characteristic of the Xilinx XC 3000 / 3100 series is their power-down mode, allowing them to stop their normal operation and retain their configuration data. In order to implement this feature, the power part has been designed so the board can be connected to an external power supply as well as a battery that is fixed on the board itself. The power supply provides the necessary current for normal operation. If it is turned off, the battery automatically takes over and the FPGAs enter the power-down mode. When the power supply is again turned on, the FPGAs return to the normal operation mode, and are ready for a new computation, without the necessity to configure them again.

4.2 Physical characteristics of the board

Since the board is intended to be an experimental board, its size is not a critical element. So it has been decided to make the board large enough to allow a confortable placement and routing of the board elements. The result was a 200x160 mm 4-layers board.

The two external layers, on both sides of the board, are signal layers. Although the board is relatively large, the routing on these signal layers is very dense. This is due to the fact that each FPGA of the FPGA block is implemented into a Xilinx XC 3195, which has 176 I/O pins, almost all of them being used.

The two internal layers are used for the power supply. The Mentor Graphics Board Station offers the possibility to split one layer into two or more area fills. This feature was used to put the two power supplies of the board (the external one and the battery) on one layer, by splitting one of the internal layers into two power fills. All the components having a power-down mode are connected to the battery power fill, the other ones are connected to the power fill corresponding to the external power supply.

4.3 Implementation of the controller

The controller, as described previously, is made up of several parts, which have been implemented with different tools. The following will give a short description of the way the controller was realized.

The controller consists of two parts: the operation unit and the control unit. The former has been described as a schematic with the Mentor Design Architect tool and then simulated with Mentor Quicksim II. The latter has been implemented and simulated as a finite state machine with the Log/IC tool.

The two descriptions has then been converted into the Xilinx format and then merged together with help of the the Xilinx software. Once the description of the controller has been converted into a unique netlist, it is then mapped into the appropriate device target, in this case a Xilinx XC 3090.

The last step consists of place and route the circuit, which is then ready to be downloaded into the controller on the board.

5 System integration

As it was said previously, this hardware environment is intended to serve as a support for prototyping and partitioning. Moreover the result of this work is one part of a set of tools that has been developed at the FZI over the last few years. This section will briefly describe how these tools are linked together and which is their contribution in the design flow.

As shown in figure 4, the design entry is realized with a VHDL description. This description is then passed on to CADDY, a high level synthesis system. CADDY takes a VHDL description and synthesizes it into a description on the RT level. In 1993 Gutberlet et al. [8] described some optimization that had been done on CADDY.

The next step in the design flow is the technology mapping. It consists of transforming the RT description into a description using the internal structure of an FPGA. At the FZI, Weinmann et al. [9] has addressed this problem by building mapping tools for Lookup-Table-Based FPGAs.

Provided that the circuit being mapped is too large to fit into a single FPGA, it has to be partitioned. A partitioning tool is under developement at the FZI. It allows the designer to choose the number of pieces into which the circuit has to be partitioned, as well as the maximum number of connections between two pieces. It further partition the circuit according to some criteria like e.g. optimizing the propagation delay or not breaking the critical path. First results are about to be published.

Once the circuit has been partitioned into subcircuits, each of them has to be placed and routed into its corresponding FPGA device (in this case a XC 3195 part). This is done with help of the Xilinx software.

As depicted in figure 4, the FPGA board described in this paper constitutes the last step of the design

flow. Once it has been placed and routed, the circuit is downloaded into the board. The application is then ready to be run and tested.

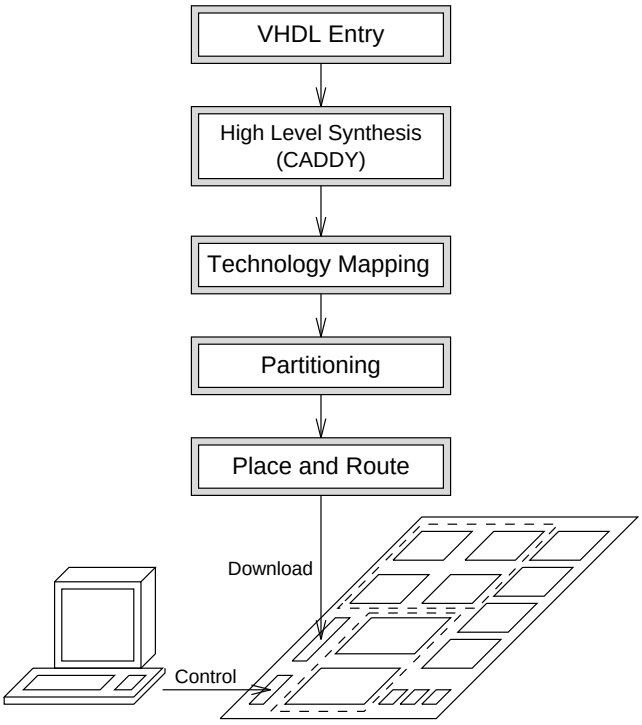

Figure 4: System integration

6 Conclusion and future work

This paper presented a hardware environment intended to address the problem of implementing a circuit into a set of several FPGAs. The main goals of this environment is on one side to serve as a support for prototyping, on the other side to allow the testing of partitioning algorithms.

The structure of this board has been kept simple and flexible. With only 4 FPGA devices for implementing the applications, it is nevertheless powerful, thanks to the use of the new Xilinx 3195, providing a capacity of about 36000 gates. The use of an FPGA as controller makes it also flexible.

The next step in this work consists of practically implement circuits onto the board. One interesting question is to know wether some types of applications are more suitable than others for being partitioned into several FPGAs. The idea is here to do a qualitative study by implementing various types of combinatorial and sequential circuits. The conclusions should allow among others to optimize partitioning algorithms and to make suggestions about possible improvements to a future version of this hardware environment.

Acknowledgment

I would like to thank Mr. Werner Dreher very much who contributed a lot of valuable advices, thanks to his thorough of knowledge of the design tools used for this work.

References

[1] U. Weinmann, A. Kunzmann, U. Strohmeier, "Evaluation of FPGA Architectures", *Int. Workshop on Field Programmable Logic and Applications*, pp. 147-156, Oxford, 1991

[2] K. Karplus, "Amap: a Technology Mapper for Selector-based Field-Programmable Gate Arrays", *28th ACM/IEEE Design Automation Conference*, pp. 244-247, 1991

[3] R.J. Francis, J. Rose, Z. Vranesic, "Chortlecrf: Fast Technology Mapping for Lookup Table-Based FPGAs", *28th ACM/IEEE Design Automation Conference*, pp. 227-233, 1991

[4] B.W. Kerninghan, S. Lin, "An Efficient Heuristic Procedure for Partitioning Graphs", *Bell System technical Journal*, Vol. 49, No2, 1970

[5] D. Ross, O. Vellacott, M. Turner, "An FPGA Based Hardware Accelerator for Image Processing", *Int. Workshop on Field Programmable Logic*, Oxford, 1993

[6] T. Flik, H. Liebig, "Mikroprozessortechnik", pp. 145-151, Springer Verlag, 1990

[7] Xilinx, "The Programmable Logic Data Book", 1993

[8] P. Gutberlet, W. Rosenstiel, "Interface Specification and Synthesis for VHDL Processes", *Proc. of the 2nd EURO-DAC*, Hamburg, 1993

[9] U. Weinmann, W. Rosenstiel, "Technology Mapping for Sequential Circuits based on Retiming Techniques", *Proc. of the 2nd EURO-DAC*, Hamburg, 1993