

Tutorial on Macro Design for Dynamic and Partially Reconfigurable Systems

Michael Hübner, Jürgen Becker
Universität Karlsruhe (TH), Germany
<http://www.itiv.uni-karlsruhe.de/>
{huebner, becker}@itiv.uni-karlsruhe.de

Abstract

Xilinx Virtex FPGAs offer the possibility of dynamic and partial run-time reconfiguration. This feature enables the substitution of an architecture within a configuration area on the chip. The benefit is, that architecture can be adapted to the actual demand of an application while run-time. High performance, flexibility and adaptivity of these devices raise the interest in academic research and also in industrial fields of application. This new method for designing systems isn't supported very well by tools until now. This tutorial should help designers as well as researchers in developing dynamic and partial reconfigurable systems and increase the number of area of applications with these very promising technology.

Keywords: Dynamic Reconfiguraion, Virtex, Macro

1. Introduction

Xilinx Virtex FPGAs offer the possibility of dynamic and partial run-time reconfiguration. This feature opens a wide field for designing systems exploiting this method. In [5] an approach of such a system for usage in automotive applications is described. New approaches using this feature by outsourcing configuration data which makes it possible, to use FPGAs with smaller configuration memory and consequently smaller chip size. Thus it is possible to save costs and reduce power consumption because not actually used modules of a complete system do not allocate configuration memory and corresponding power consuming hardware [1]. Nevertheless power dissipation during reconfiguration has to be considered. [2] raise this issue with the main focus on energy saving and basis for new design methodology. The designer of a dynamic and partial reconfigurable system on an FPGA must be sure, that no signal lines of a module cross the border to another functional block. During reconfiguration such a signal line might cause a malfunction or a short-circuit, which destroys the FPGA. Because of this it is necessary to implement interfaces which are used as fixed routing resources. These interfaces, called BUS Macros, are placed in the same position for each functional block. Connecting the modules with signal lines on the same position grants the option, to substitute a module by tools until now.

Figure 1.1 shows the hardware reconfigurable system described in [5]. Another approach with a macro based communication structure is described in [4]. The design flow for generating the communication architecture isn't supported in existing tools at the moment. Interface between the reconfigurable areas have to be designed with routing tools like Xilinx FPGA Editor.

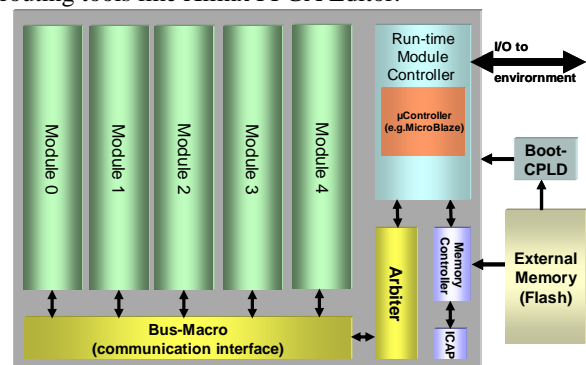


Figure 1.1 Dynamic reconfigurable system

A new approach for designing these interfaces is described in [3] and [9]. There pre-programmed LUTs within a CLB are used to transfer information to the receiving units. The more complicated design process using CLBs as communication interfaces instead of using TBUF elements as described in [7] leads to more save result regarding forbidden connections between reconfigurable areas. This tutorial presents the methods for designing these macros and implementing them into a dynamic and partial reconfigurable system. The paper is organized as follows: Section 2 motivates the idea of dynamic and partial reconfiguration. Section 3 describes the basic principles of an FPGA. Section 4 describes the necessity of bus macros. In section 5 the LUT based approach is presented. The paper is closed with the conclusions in section 6.

2. Why Dynamic and Partial Reconfiguration

As described in section 1, a lot of advantages can be exploited by using the method of dynamic and partial reconfiguration. Certainly reconfiguration can be compared with the method of changing memory content

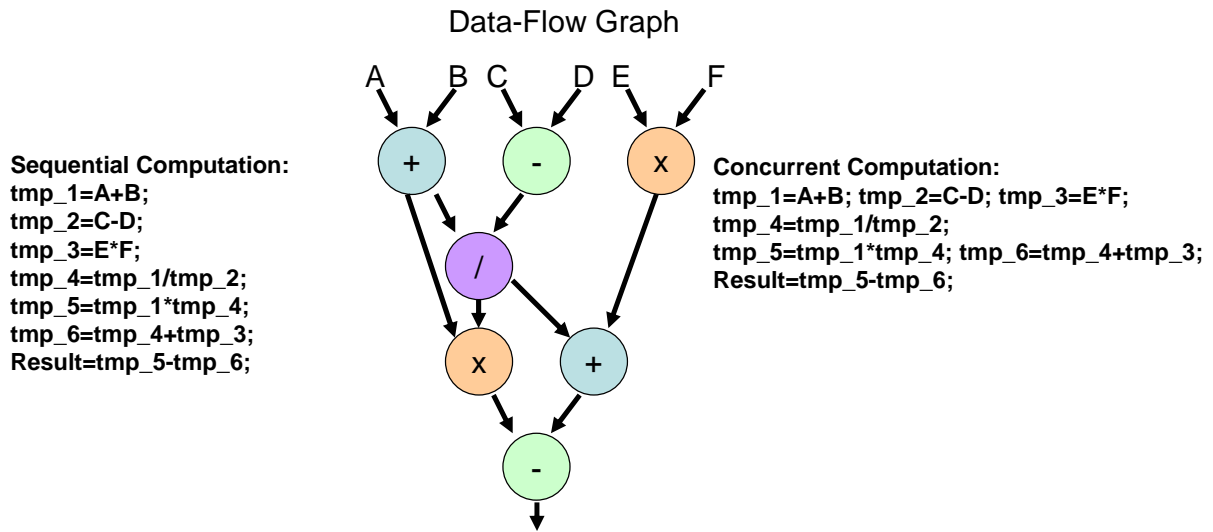


Figure 2. Dataflow Graph with concurrent and sequential code example

using a traditional processor based design. The advantage using hardware reconfiguration is the fact, that parallelization of tasks running concurrently on chip is only possible by using real hardware architecture. While processors regularly only run one thread, parallel hardware architectures have one thread for each application integrated on the chip area. The disadvantage of ASIC design with fixed integrated functional blocks on the silicon can be eliminated by using reconfigurable hardware. Apart from the high risk and costs for developing ASICs, concurrent but fixed architecture isn't adaptable while run-time. Therefore new approaches introducing run-time adaptive systems with the ability to reconfigure parts of the architecture on-demand were developed.

Figure 2 shows an example of a data-flow for a simple calculation. It is obvious, that sequential calculation which has to be done in traditional processors leads to a higher number of intermediate steps before the final result is available.

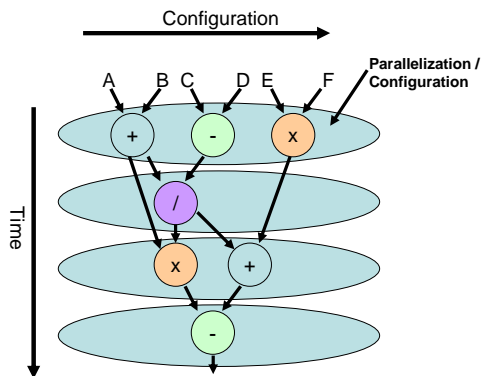


Figure 3. Parallelization of functions

There is no surprise that parallel computation leads to a reduced number of steps for calculating the final result. Much more interesting is the analysis of methods for integrating this small function to reconfigurable hardware. Figure 3 shows the analysis of the data-flow graph of figure 2. The axis describes the time dimension for scheduling the different operations. The configuration axis shows the necessary configuration of the architecture in each time step. It can be seen, that a maximum number of three operations is necessary to run the complete calculation if these operations are re-used for calculation in a succeeding step.

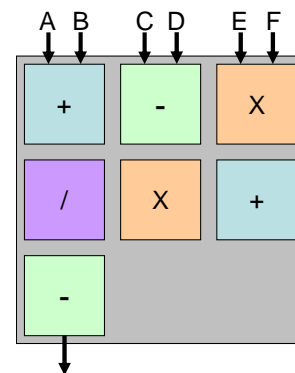


Figure 4. hardware integration of all operations

In figure 4, all required operations are integrated on chip area. This could be an example of a static ASIC approach. Figure 5 shows an approach with a data flow scheduler which switches the path of internal and external data while run-time. This flowware [10] (in opposite to the traditional von Neumann approach) and its scheduler, doesn't fetch instructions. In opposite to the program

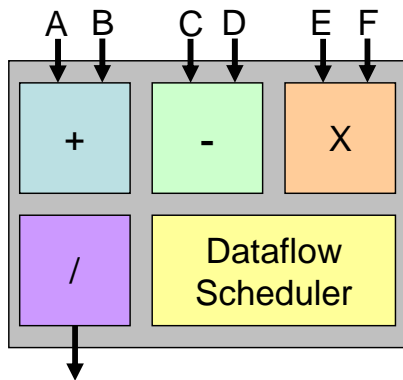


Figure 5. Calculation with dataflow scheduler

counter of von Neumann machines, it is seen as a data counter as described in [10].

This approach is more obvious visualized in figure 6 where dynamic and partial reconfiguration is exploited to integrate the required function while run-time. In this approach, the chip size is definitely smaller in comparison to the integration in figure 4. The computation is done in time and space (area) on the reconfigurable architecture.

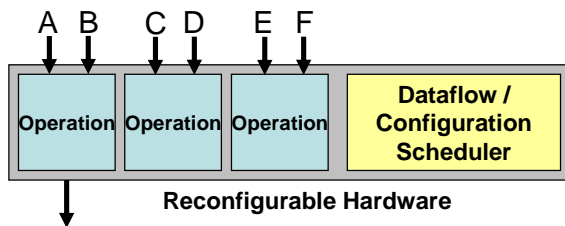


Figure 6. Integrated system exploiting dynamic and partial reconfiguration

3. Basic Internal FPGA Structure

Figure 8 shows the very basic elements of an FPGA. The structure of the FPGA consists of two layers. The configuration layer with its memory includes the information of the actual configuration of the second layer, the hardware layer. The memory is programmed after synthesis, place and route procedure. The configurable logic is connected to the configuration memory. As an easy example figure 8 shows a gate with three inputs and one output which is programmed as logic AND operation. Internal wires of the hardware layer, select one address of the configuration memory. Dependently to the content of the memory, a buffer is switched to 0 or one. This element, responsible for logic operations is called: Configurable Logic Block (CLB). Additional to logic resources, the hardware layer provides routing resources for connecting the logic elements. With

switch matrixes, horizontal and vertical signal lines can be connected. The switches (CMOS-Transistors) are controlled by the configuration memory. Of course this is a very easy example but it can be used to explain colorful how dynamic and partial reconfiguration works. While run-time, different configurations were sent via the configuration access port to the configuration memory. It is clear what happens: Both logic elements and routing resources can be influenced and adapted to a new functionality and routing. It becomes clear that changes influence the behavior of functions in the neighborhood if signal lines cross the area where partial reconfiguration occurs.

In the next sections exemplarily with Xilinx Virtex FPGAs the usage of communication structures necessary for developing reconfigurable systems were described.

4. Bus Macro

For a faultless data communication it's necessary to implement a structure with fixed signal lines. Implementing modules on the FPGA's reconfigurable area without specification of connection points between the functionalities, leads to problems after place and route. Signal lines may be open or even two output ports of a module are connected together. This definitely causes a damage of the chip and therefore communication elements with fixed connection points and wiring is necessary. Figure 7 shows a system with generalized

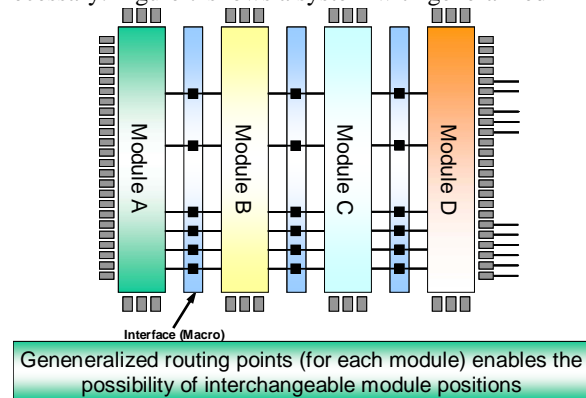


Figure 7. System with communication interfaces

communication interfaces. As mentioned, now fixed connection points for each module are established and while designflow the router connects all signals on a defined position. Generalizing those interfaces enables also to interchange modules and their positions. Not only the fixed connection points in a reconfigurable system are important for an undisturbed data transfer. It is also necessary to fix the used physical signal lines in each reconfigurable area. While reconfiguration, other modules run in parallel and might communicate via the bus. A change of routing can cause a glitch or even a loss of

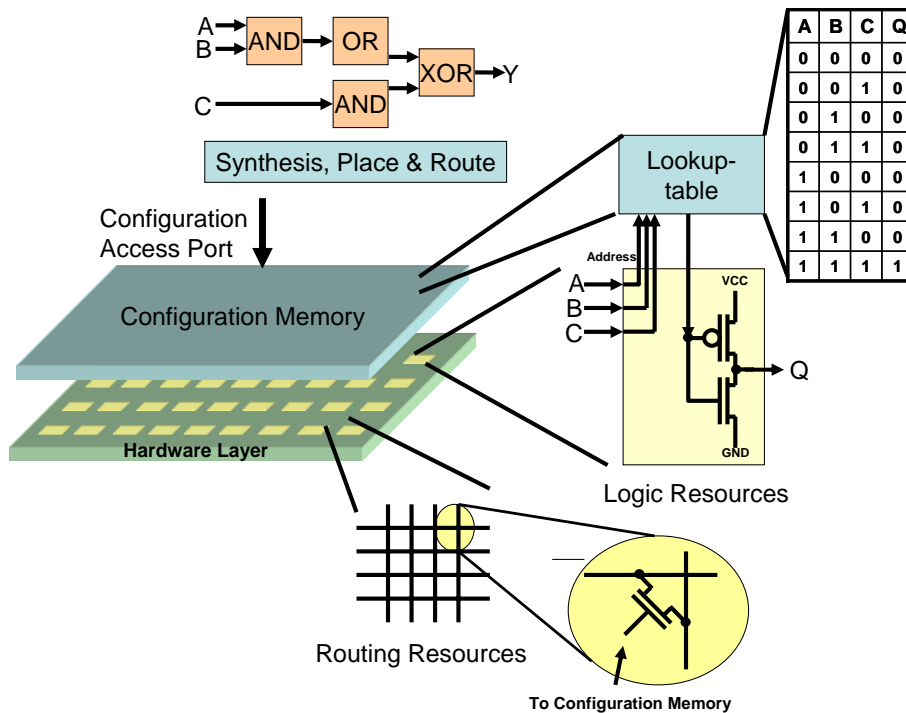


Figure 8. Basic internal FPGA structure

information while reconfiguration. A variety of tests were realized to placing in secure the save and undisturbed data transfer. The architecture which makes this possible are bus-macros. Bus-macros are a combination of communication interfaces and fixed signal lines.

5. CLB Based Macro Approach

Figure 9 shows a schematic for an input macro for four reconfigurable areas (slots). The usage of CLBs introducing macro external pins by designing macros with Xilinx FPGA editor, enable the autorouter to connect signal lines to fixed and defined points. Using the traditional approach with TBUF elements as shown in figure 10, often leads to problems after routing phase. Because signal names are identical, the autorouter can connect signals from the left module to resources of the right module. By introducing macro external pins to dedicated sites of a macro, this problem can be solved. Therefore the CLB based approach provides a secure design of reconfigurable systems without time consuming

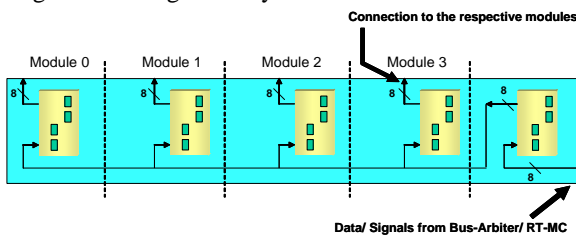


Figure 9. Input Macro for parallel data transfer

and uncomfortable check for sources of failure. The FPGA editor enables the designer to program the content of CLB internal look up tables which then can be used to transfer input signals to the respective modules within the design. The way from specification over the design of CLB based macros to implementation on FPGA will be described and practically demonstrated in this tutorial.

- Macro external pins belong to left or right module
- Therefore: Output is not on a definite side
- Design not capable for dyn. part. reconfiguration

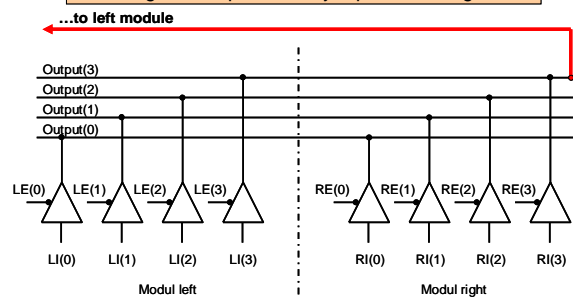


Figure 10. Macro with TBUF elements

6. Tutorial Schedule

The described methodology for developing dynamic reconfigurable systems has the aim to introduce this novel approach into lectures with practical experiments for students and developers from the industry. The degree of freedom which is introduced with this technology enables

to integrate adaptive and high-performance systems. These systems are adaptive in time and space and therefore an ideal platform for academic research and also for real industrial systems. The benefit of re-programming is extended to re-programmable systems while run-time.

The tutorial consists of a theoretical part for introducing the basic knowledge of hardware.

The theoretical part is separated in:

- Graph Theory
- Synthesis of hardware modelled in VHDL / Verilog
- Scheduling
- Technology Mapping

To learn with real hardware, also a hands-on tutorial goes along with the theoretical part and is separated in the following way:

- Introduction: Hardware Design Tools
- Basic systems: Integration to real hardware
- Macro Design exemplarily with Xilinx Spartan-III FPGAs
- Development of a partitioned dynamic reconfigurable system
- Development and integration of a scheduler
- Integration of the complete reconfigurable system

The tutorial can be held within two weeks with alternately theoretical part and practical experiments.

7. Conclusions

This tutorial-paper shows an implementation of a reconfigurable system using slices as connecting resources instead of TBUF elements. Using this design method an automatic design flow without manual debugging is possible [8]. The goal of this tutorial is to describe the designflow for designing dynamic and partial reconfiguration in a save and non time consuming manner. With practical demonstration the functionality of macros designed while tutorial will be presented. The forward looking technology using reconfigurable hardware as an adaptive device should be accessible for academic research and development to improve electronic systems of tomorrow.

8. References

- [1] J. Becker, M. Hübner, M. Ullmann: "Power Estimation and Power Measurement of Xilinx Virtex FPGAs: Trade-offs and Limitations", SBCCI03, Sao Paulo, Sep. 03
- [2] L. Benini, G. De Micheli: "Networks on Chip: A New Paradigm for Systems on Chip Design", Date 02, March 3~7, Paris France
- [3] M. Huebner, T. Becker, J. Becker "Real-Time LUT-Based Network Topologies for Dynamic and Partial FPGA Self-Reconfiguration", SBCCI04, Brasil
- [4] J.C. Palma, A. Vieira de Melo, F. G. Moraes, N. Calazans, "Core Communication Interface for FPGAs", SBCCI02, Porto Alegre Brazil
- [5] M. Ullmann, M. Huebner, B. Grimm, J. Becker: "An FPGA Run-Time System for Dynamical On-Demand Reconfiguration", RAW04, Santa Fee
- [6] http://www.xilinx.com/ise/design_tools/
- [7] XAPP291, Xilinx Application note
- [8] A. Donlin, J. Becker, M. Hübner: "Models and Tools for the Dynamic Reconfiguration of FPGAs", IEEE-SOCC2005, Washington, USA
- [9] M. Hübner, K. Paulsson, M. Stitz, J. Becker: "Novel Seamless Design-Flow for Partial and Dynamic Reconfigurable Systems with Customized Communication Structures Based on Xilinx Virtex-II FPGAs", ARCS05, Innsbruck, Austria
- [10] Reiner Hartenstein: "Reconfigurable Technologies", Seminar at Department of Informatics, Kyushu University, Higashi-ku, Fukuoka City, Kyushu, Japan; July 23, 2004