THE ERLANGEN SLOT MACHINE: INCREASING FLEXIBILITY IN FPGA-BASED RECONFIGURABLE PLATFORMS

Christophe Bobda, Mateusz Majer, Ali Ahmadinia, Thomas Haller, André Linarth, Jürgen Teich

Department of Computer Science 12
University of Erlangen-Nuremberg, Germany
{bobda, ahmadinia, majer, teich}@cs.fau.de

ABSTRACT

We present a new concept as well as the implementation of an FPGA-based reconfigurable platform, the *Erlangen Slot Machine (ESM)*. One main advantage of this platform is the possibility for each module to access its periphery independent from its location through a programmable crossbar, allowing an unrestricted relocation of modules on the device. Furthermore, we propose different intermodule communication structures.

1. INTRODUCTION

Despite the announcement made by several companies in the last couple of years about the design and production of new and mostly coarse-grained reconfigurable chips, today's reconfigurable computing platforms are still FPGA-based. The growing capacities provided by FPGAs as well as their partial reconfiguration capabilities have made them the ultimate choice. The Xilinx FPGAs are the only devices on the market with large capacity and the ability to support partial reconfiguration. The Virtex series offers enough logic for efficiently implementing applications with high demand of resources, e.g., arising in video, audio and signal processing as well as in other fields like automotive applications. Partial reconfiguration is useful to increase the flexibility in computation and for efficiency reasons time-sharing the device space. It requires run-time loadable modules to be compiled and stored as bitstreams, which will then be used to reconfigure the device, i.e., allocate space for the computation of the module on the device. Several algorithms for on-line placement were developed in the past [1, 2]. However, their implementation is limited by two main factors. First, very few FPGA platforms exist on which those algorithms can be implemented. Second, the development process of modules is subject to many restrictions that make a systematic development process for partial reconfiguration difficult. Each module placed at a given location on the device is implicitly assigned all the resources in that area. This includes the pins, the clock managers and other hard macros like multipliers and BlockRAM.

SUPPORTED IN PART BY THE GERMAN RESEARCH FOUNDATION (DFG), SPP 1148 (REKONFIGURIERBARE RECHENSYSTEME)

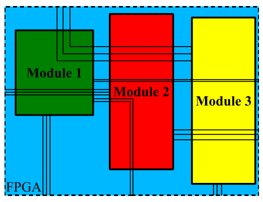

Fig. 1. The feed-through problem with reconfigurable modules

As illustrated in Figure 1, a module using resources outside its placement area must use connections running trough other modules in order to access its resources. We call those signal used by a given module and crossing other modules **feed-trough signals**. Using feed-through lines to access resources has two negative consequences:

- Difficulty of design automation: Each module must be implemented with all possible feed-trough channels needed by other modules. Because we only know at run-time which module needs to feed through the signal, many channels reserved for a possible feed-through become redundant.
- Relocation of modules: Modules accessing pins are no more relocatable, because they are compiled for fixed locations where a direct access to their pins is possible.

Until now, no FPGA platform on the market provides a solution to the problems previously mentioned. Many systems on the market offer various interfaces for audio and video capturing and rendering, for communication and so forth. However, each interface is connected to the FPGA using dedicated pins in a fixed location. Modules willing to access a given interface like the VGA must be placed in the area of the chip where the FPGA signals are connected, thus making a relocation impossible. The purpose of the Erlangen Slot Machine (ESM) is to overcome the deficiency of existing FPGA platforms by providing:

- A new highly flexible FPGA platform in which each component must not be fixed all the time at a given chip location.
- A suitable tool support for the development process of modules for run-time reconfiguration and communication, and for making an efficient use of the architecture.

This paper is organized as follows: Section 2 provides an overview of existing platforms and shows their deficiencies. Due to the large number of existing platforms, we focus on a small representative subset of systems that are actually available. In Section 3, we present the concept of our ESM platform. Section 4 describes all available intermodule communication support for the ESM. In Section 5, we present a first application in video streaming implemented on the ESM. The application makes use of dynamic reconfiguration to enhance the flexibility. The concluding Section 6 provides a look at future work.

2. DRAWBACKS OF EXISTING SYSTEMS

Due to the following factors, the practical realization and use of partial and dynamic reconfiguration on current existing FPGA-based reconfigurable platforms is highly restricted:

1. **Limitation of reconfiguration on actual FPGAs:** Very few FPGAs allowing partial reconfiguration exist on the market. Those few FPGAs, like the Virtex series, allow only a restricted partial reconfiguration. The reconfiguration is done column wise meaning that loading a module in a given area will affect all the modules on top and below the module.
2. **I/O-pin problems:** Most of the existing platforms have peripheral components like video, RAMs, audio, ADC and DAC connected at fixed locations on the device. This has the consequence that a module must be stuck on a given region where it will have access to its pins, and therefore making a relocation very difficult. Another problem related to the pin connection is that the pins belonging to a given logical group like video, audio, etc. are not situated closely to each other. On many platforms they are spread around the device. A module willing to use a device will have to feed many lines through many different components. This situation is illustrated on figure 2: Two modules (one of which is a VGA module) are implemented. The VGA module uses a large number of pins at the bottom part of the device and also on the right hand side. Implementing a module without feed-through is only possible on the two first columns on the left hand side. The effort needed for implementing a reconfigurable module on more than two columns together with the VGA module is very high. This situation is not only present on Celoxica boards [3]. The XESS boards [4], the Nallatech boards [5], and the Alpha boards [6] face the same limitation. On the XF-Board [7, 8] from ETH Zurich, the peripherals are connected to one side of the device. Each module accesses its peripherals through an operating system (OS) layer implemented on the left and right part of the device. Many other existing platforms like the RAPTOR board [9], Celoxica RC1000 and RC2000 [3] are PCI systems that require a workstation for operation. The use in stand-alone systems as needed in embedded systems is not possible.
3. **Intermodule communication:** Modules placed at run-time on the device typically need to exchange data among each other. This request for communication is a dynamic task due to the on-line module placement. For modules placed far from each other it will not be possible to feed communication lines through several modules in order to establish a connection. New paths are then necessary in order to allow a dynamic communication to take place during run-time.

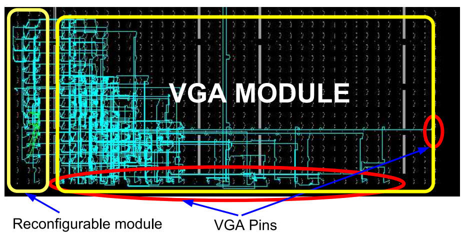

Fig. 2. Pin distribution of the VGA module on the RC200 platform

With these limitations in mind, we designed a new and FPGA-based platform, called the Erlangen Slot Machine (ESM) for reconfigurable computing. A tool (which will not be addressed in this paper) is currently in development for easing the implementation of modular applications on the ESM.

3. THE ESM ARCHITECTURE

The Erlangen Slot machine (ESM) consists of a BabyBoard mounted on a MotherBoard. The clustering of the system into two boards allows the BabyBoard that contains the reconfigurable FPGA to be used on a wide variety of application domains such as image and audio signal processing. For the integration of the ESM into a new domain such as the automotive domain, no redesign of the complete system is necessary. Only a new MotherBoard must be provided according to the computational requirements in the new environment. For example, a multimedia system will provide a MotherBoard with multimedia devices like video and audio. In the following, we provide a detailed description of the board's functionality.

3.1. The BabyBoard

3.1.1. Computation and Reconfigurable Engine

The reconfigurable engine of the ESM platform is a baby board that features a Xilinx Virtex II-6000 FPGA from Xilinx, several SRAMs and a configuration circuit. Due to the restriction[1] in the reconfiguration of Virtex FPGAs, we adapt our architecture to match the following properties:

- **Free relocation of modules:** On-line placement of modules on a reconfigurable device, in this case the FPGA, is done by downloading a partial bitstream that implements a given task in the FPGA. This requires a relocation that places a module in a location different from the one for which it was compiled. Relocation can be done only if all the resources are available and structured in the same way in the previous location and in the new location. This includes the device pins used by the module. If the connection of a module with its peripheral is done via some pins that fixed at a given chip location, the module must be restricted to that location. A relocation will be possible, but the module must feed signals through all the other modules between the pins and the new location. We solved this problem on the ESM by avoiding a direct connection of peripherals on the FPGA. As shown in Figure 3, all the bottom pins from the FPGA are connected to an interface controller (crossbar). The interface devices are connected to the interface controller as well. This makes it possible to establish any connection from one module to its peripheral via the crossbar.

- **Uniform repartition of resource:** Like any other reconfigurable platform, the ESM is primarily a parallel computing engine in which each module carries its own computation and communicates with other modules. Therefore, a certain amount of resources must be allocated to each module for its computation, at least for a given period of time. Memory is very important in applications like video streaming in which a given module must exclusively access a picture at a time for computation. However, the capacity of the available BlockRAMs in FPGA is limited. External SRAM memory is therefore added to allow storage of large amounts of data by each module. To allow a module to exclusively access its memory, the SRAM is connected at the top part of the FPGA. In this way, a module will have its connection to its peripheral from the bottom part, while the top part will be used for temporally storing the computation data. According to the number of memory banks which can be connected on the top, the device is divided into a set of exchangeable slots A to V, each of which has access to the SRAM on the top part and to the crossbar at the bottom. Our system is called the *Erlangen Slot Machine* due to this arrangement of reconfigurable slots. This modular organization of the device simplifies the relocation, primary condition for a viable reconfigurable computing system. Each module moved from one slot to another will encounter equal resources. The architecture is illustrated in Figure 3.

[1] The reconfi guration can be done only columnwise

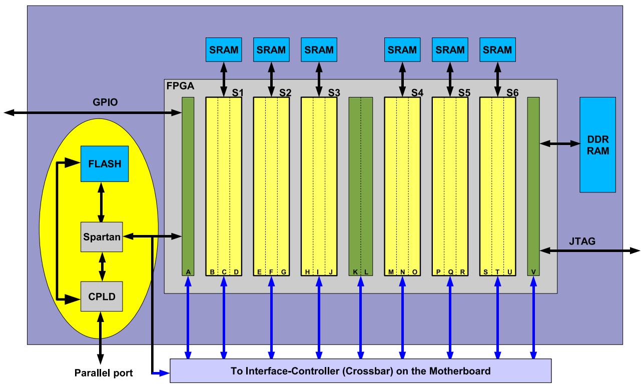

Fig. 3. Architecture of the ESM BabyBoard

3.1.2. The Configuration Manager

Apart from the main FPGA, the BabyBoard also contains the configuration circuitry. This consists of a CPLD, a configuration FPGA (a small Spartan II FPGA) and a Flash.

- The CPLD is used to download the Spartan II configuration from the flash upon power-up. It also contains board initialisation routines for the on-board PLL and the Flash.

- The configuration program for the main FPGA is implemented in the Spartan II. Besides the configuration finite state machine, this device contains a circuit to perform module relocation while downloading the bitstream. Because the Virtex II bitstream format is not yet open, this solution has not been implemented yet.

- The Flash provides a capacity of 64 MBytes, thus enabling the storage of up to 32 full configurations and a few hundred partial bitstreams.

3.1.3. *Memory*

Six SRAM banks of size 2 MBtye each are vertically attached to the board on the top side of the device, thus providing enough memory space to six different super slots for temporal data storage. The SRAMs can be also used for shared memory communication between neighbour modules, e.g., for streaming applications. They are connected to the FPGA in such a way that the reconfiguration of a given module will not affect the access to other modules.

3.1.4. *Debug Lines*

Debugging capabilities are offered through general purpose I/O provided at regular distances between the basic slots. A JTAG port provides debug capabilities for the main FPGA, the CPLD and the Spartan II.

3.2. The MotherBoard

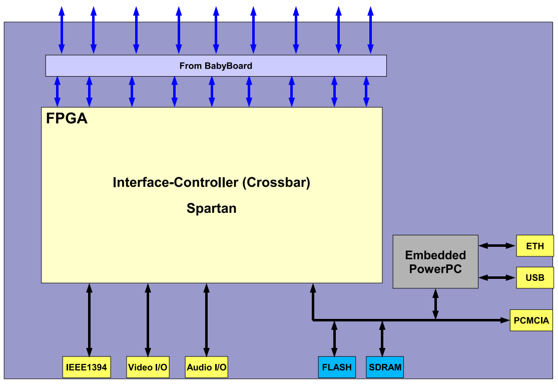

Fig. 4. Architecture of the ESM MotherBoard

The MotherBoard provides programmable links from the FPGA to all peripherals. The physical connections are established at run-time through a programmable crossbar implemented in an a Spartan chip on the MotherBoard. Besides the run-time-programmable crossbar, many peripherals for multimedia and communication are available on the board (Figure 4). Video capture and rendering interfaces as well as high speed communication links are available on the MotherBoard. A Power PC processor is also available for managing the dataflow on the MotherBoard as well as the communication with external systems.

4. INTERMODULE COMMUNICATION

One of the central limiting points for the wide use of partial dynamic reconfiguration is the problem of intermodule communication. Each module that is placed on the device must collects its data from a given source and send its results to a given destination. For the ESM, we provide four main possibilities for realizing the communication among different modules (see Figure 5): The first one is a direct communication using bus macros between adjacent modules. Secondly, shared memory communication using SRAMs or BlockRAMs is possible. However, only adjacent modules can use those two communication modes. For modules placed in non-adjacent slots, we provide a dynamic signal switching communication architecture called reconfigurable multiple bus (RMB) [10]. In case of emergency, the communication between two different modules can also be realized through the external crossbar.

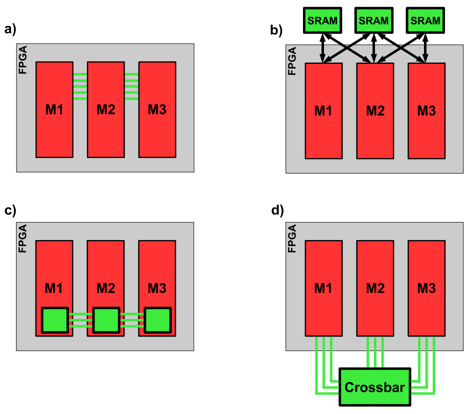

Fig. 5. Intermodule communication possibilities on the ESM: a) bus-macro, b) shared memory, c) reconfigurable multiple bus RMB, d) external crossbar

4.1. Adjacent Communication

Adjacent communication is done between abutting modules. On the ESM, *BusMacros* are used to realize a direct communication between neighbour modules, providing fixed communication channels that help to keep the signal integrity on reconfiguration. Because only four signals can be passed for each bus macro, the number needed for connecting a set of n signals is $n/4$.

4.2. Communication via Shared Memory

Communication between two neighbouring modules can be done in two different ways using shared memory: In the first, the dual ported BlockRAMs are use for implementing communication among two neighbour modules working in two different clock domains. The sender writes on one side, while the receiver reads the data on the other side. The second possibility uses external RAM. This is particular useful in applications in which each module must process a large amount of data and then sends the processed data to the next module, as it is the case in video streaming. On the ESM, each SRAM can be accessed by its corresponding module as well as its right and left neighbours. A controller is used to manage the SRAM access. Depending on the application, the user may set the priority of accessing the SRAM for the three modules.

4.3. Communication via RMB

In its basic definition, the Reconfigurable Multiple Bus (RMB) architecture [11, 12] consists of a set of processing elements, each posessing a switching bus connection to other processing elements. The switches control connection requests between individual modules. The RMB is a one-dimensional arrangement of switches. In out implementation on an FPGA, the horizontal arrangement allows for the communication among modules placed in the slots. The request for a new connection is done in a wormhole fashion, where the sender (processor P_k) sends a request for communication to its neighbor (processor P_{k+1}) in the direction of the receiver. Processor P_{k+1} sends the request to processor P_{k+2}, etc., until the receiver receives the request and returns an acknowledgment. The acknowledgment is then sent back in the same way to the sender. Each processor that receives an acknowledgment sets its switch to connect the two bus segments. Upon receiving the acknowledgment, the sender can start the communication (circuit routing). The wired connection stays until an explicit release command is issued by the sender. This approach was first presented in [12] and extended later in [11] with a compaction mechanisms for quickly finding a free segment. In our implementation of the RMB on Xilinx

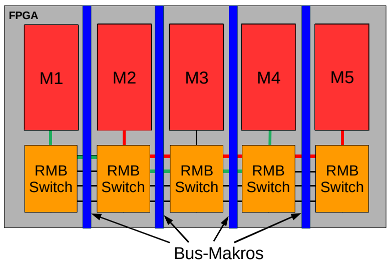

Fig. 6. FPGA implementation of the RMB for partial reconfiguration

Virtex FPGAs, we separated the switching controller from the modules. In this way, we provide a uniform interface to designers for connecting their modules on the bus. The implementation of the RMB structure on an FPGA Virtex II 6000 with four processors and four parallel buses each having a bandwidth of 16 reveals an area overhead of 4% with a frequency of 120 MHz on the controller [10]. As shown on Figure 6, Bus-macros are used at the boundary of modules and controllers to insure a correct operation upon reconfiguration.

4.4. Communication via the Crossbar

The last possibility of establishing a communication among modules is to use the crossbar. Because all the modules are connected to the crossbar via the pins at the bottom of the FPGA, the communication among the modules can be set in the crossbar as well. This possibility should only be used as emergency solution due to the high latency caused by an off-chip communication.

5. APPLICATION

In this section, we explain the implementation of a reconfigurable video streaming application on the ESM. In video

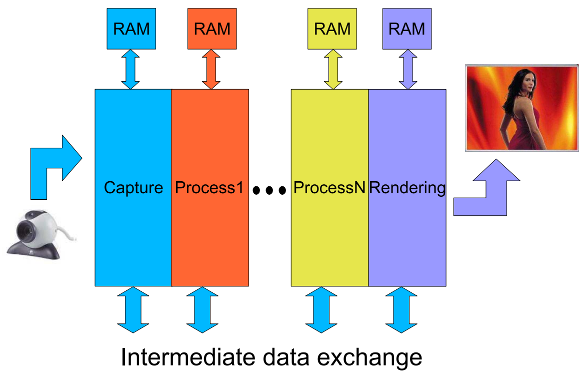

Fig. 7. A modular architecture for video streaming on the ESM.

streaming, a video is transmitted image by image through a given system, with the images being processed in different parts of the system, pixel by pixel. Because many operations on each pixel require the neighbourhood pixels, a sliding window is used for keeping track of the adjacent pixels. Our reconfigurable architecture is shown in Figure 7. The first module deals with the image capture from an image source. This can be a camera or a network module that collect the pictures through a network channel. Images are alternately placed in the SRAM1 and SRAM2. The second module collects the picture from the SRAM1 or SRAM2. If this is not in use by the first module, build the sliding windows and pass it to the third module, which processes the pixel stream and saves it in its own memory or directly passes it to the next module. The images are then streamed from SRAM to SRAM and the processed pictures can be output either by an intermediate module or by the final module (Process N). In this architecture, each module can be reconfigured depending on the application's need. The first module can be reconfigured if the type of camera changes or if the calibration of the camera requires that. The last module may be reconfigured if the type of the outputs changes, e.g., if the processed pictures need to be sent through the network instead of being displayed. An intermediate module might be replaced by a new one with new functionality. In our first implementation, we execute an edge detection algorithm on a video stream using different methods. Each method is implemented as a single module that can be exchanged with other modules. We currently have implemented several versions of the Sobel operator, the Laplace operator and the Prewit operators.

6. CONCLUSION

In this paper, we presented a new approach for designing and implementing an FPGA-based reconfigurable platform. This platform provides enough flexibility for implementing one-dimensional on-line placement algorithms in reconfigurable computing. A great focus has been placed on communication, uniformity of resource distribution, and placement freedom by means of relocation. Despite the non-support for two-dimensional reconfigurability, we believe that the flexibility of the board provides the necessary condition for implementing previously developed algorithms. The running system is available at our institute and can be provided on demand to other researchers working in this direction.

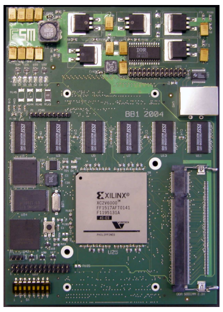

Fig. 8. Implementation of the ESM BabyBoard

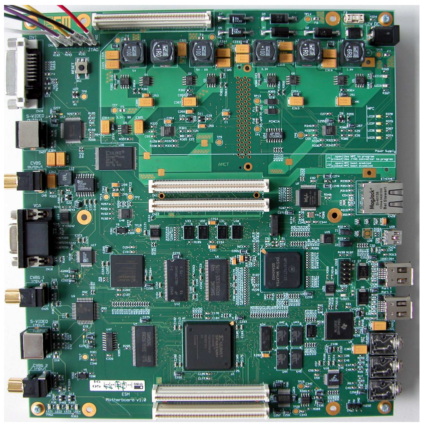

Fig. 9. Implementation of the ESM MotherBoard

7. REFERENCES

[1] A. Ahmadinia, C. Bobda, S. Fekete, J. Teich, and J. van der Veen, “Optimal routing-conscious dynamic placement for reconfigurable devices,” in *Proceedings of International Conference on Field-Programmable Logic and Applications (FPL)*, ser. Lecture Notes in Computer Science (LNCS), vol. 3203. Antwerp, Belgium: Springer, Aug. 2004, pp. 847–851.

[2] K. Bazargan, R. Kastner, and M. Sarrafzadeh, “Fast template placement for reconfigurable computing systems,” *IEEE Design and Test of Computers*, vol. 17, no. 1, pp. 68–83, 2000.

[3] *RC2000 Development Board*, Celoxica Ltd., 2004, http://www.celoxica.com/products/boards/rc2000.asp.

[4] X. Corp, `http://www.xess.com`.

[5] I. Nallatech, `http://www.nallatech.com`.

[6] *ADM-XRC-II Xilinx Virtex-II PMC*, Alpha Data Ltd., 2002, http://www.alpha-data.com/adm-xrc-ii.html.

[7] M. Platzner and L. Thiele, “XFORCES - executives for reconfigurable embedded systems,” http://www.ee.ethz.ch/p̃latzner.

[8] C. Steiger, H. Walder, M. Platzner, and L. Thiele, “Online scheduling and placement of real-time tasks to partially reconfigurable devices,” in *Proceedings of the 24th International Real-Time Systems Symposium (RTSS’03)*, December 2003.

[9] U. R. H. Kalte, M. Porrmann, “A prototyping platform for dynamically reconfigurable system on chip designs,” in *Proceedings of the IEEE Workshop Heterogeneous reconfigurable Systems on Chip (SoC)*, Hamburg, Germany, Sept. 2002.

[10] A. Ahmadinia, J. Ding, C. Bobda, and J. Teich, “Design and implementation of reconfigurable multiple bus on chip (rmboc),” University of Erlangen-Nuremberg, Department of CS 12, Hardware-Software-Co-Design, Tech. Rep. 02-2004, Nov. 2004.

[11] H. A. ElGindy, A. K. Somani, H. Schrder, H. Schmeck, and A. Spray, “Rmb - a reconfigurable multiple bus network,” in *Proceedings of the Second International Symposium on High-Performance Computer Architecture (HPCA-2)*, San Jose, California, Feb. 1996, pp. 108–117.

[12] R. Vaidyanathan and J. L. Trahan, *Dynamic Reconfiguration: Architectures and Algorithms*. IEEE Computer Society, 2003.