A Flexible Reconfiguration Manager for the Erlangen Slot Machine

Mateusz Majer[1], Ali Ahmadinia[1], Christophe Bobda[2], and Jürgen Teich[1]

[1] University of Erlangen-Nuremberg
Hardware-Software-Co-Design
Am Weichselgarten 3, 91058 Erlangen, Germany
[2] Kaiserslautern University of Technology
Self Organizing Embedded Systems Group
Gottlieb-Daimler-Strasse 48, 67663 Kaiserslautern, Germany

Abstract. We present a new concept as well as the implementation of a reconfiguration manager for a 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 of its location through a programmable crossbar, allowing an unrestricted run-time relocation of modules on the device. To aid the reconfiguration process we present a flexible plugin architecture for a hardware reconfiguration manager. Its advantage is fast preprocessing of bitstream data by different plugins such as a decompression and a relocation plugin. The plugin order is arbitrary and determined at run-time. Moreover, our architecture does not suffer from performance degradation if several plugins are cascaded.

1 Introduction

Growing capacities provided by FPGAs as well as their partial reconfiguration capabilities have made them the ultimate choice for reconfigurable computing platforms. 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 by 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, these algorithms are limited by two main factors. First, very few FPGA platforms exist on which these 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.

Until now, no FPGA platform on the market provides a solution to the problems of design automation for dynamically reconfigurable modules and the efficient and their flexible relocation. 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 accessing 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)* [3, 4] is to overcome the deficiency of existing FPGA platforms by providing:

- A new flexible FPGA platform in which each component is not required to be fixed all the time at a given chip location.
- Tool support for the development of run-time reconfigurable computation and communication modules.
- A powerful reconfiguration manager which enables various preprocessing stages for fast bitstream manipulation. We call the preprocessing stages plugins. For example, a decompression or relocation plugin can be selectively activated before a bitstream is uploaded to the FPGA.

Reconfiguration times in the range of seconds [5] are not enough for applications that require a fast reaction to external events. Our hardware reconfiguration manager, described in this paper, is the foundation for reconfiguration times in the range of milliseconds. For example, these fast reconfiguration times will allow a seamless switching of video filters in a video pipeline architecture.

This paper is organized as follows: Section 2 provides an overview of the Erlangen Slot Machine. Moreover, we depict the advantages of our platform in comparison to related work in this area and give an outlook in the application domain for this platform. In Section 3, we present the concept of our reconfiguration manager and present different workload scenarios. These scenarios show the enhanced flexibility of our concept for different application domains. In Section 4, we conclude our work and provide an outlook of future work.

2 The Erlangen Slot Machine

The main idea of the Erlangen Slot Machine (ESM) architecture is to accelerate the application development as well as the research in the area of partially reconfigurable hardware. The advantage of the ESM platform is its unique slot based architecture which allows the slots to be used independently of each other by delivering peripheral data through a separate crossbar switch as shown in Figure 1. The ESM architecture is based on the flexible decoupling of the FPGA I/O pins from a direct connection to an interface chip. This flexibility allows to place application modules at run-time in any available slot independently. Thereby, run-time placement is not constrained through physical I/O pin locations as the assignment of the I/Os is done in the crossbar.

2.1 Architecture overview

The ESM platform (see Figure 1 is centered around an FPGA serving the main reconfigurable engine and an FPGA realizing the crossbar switch. They were separated into two physical boards and are implemented using a Xilinx Virtex-II 6000 and a Xilinx Spartan-2E 600 FPGAs. Figure 1 shows the slot based architecture of the ESM platform consisting of the Virtex-II FPGA, local SRAM memories, configuration memory and the reconfiguration manager. The top pins of the FPGA connect to local SRAM banks and the bottom edge pins connect to the crossbar switch. Therefore, a module can be placed in any free slot and have its own peripheral I/O links together with dedicated local memory.

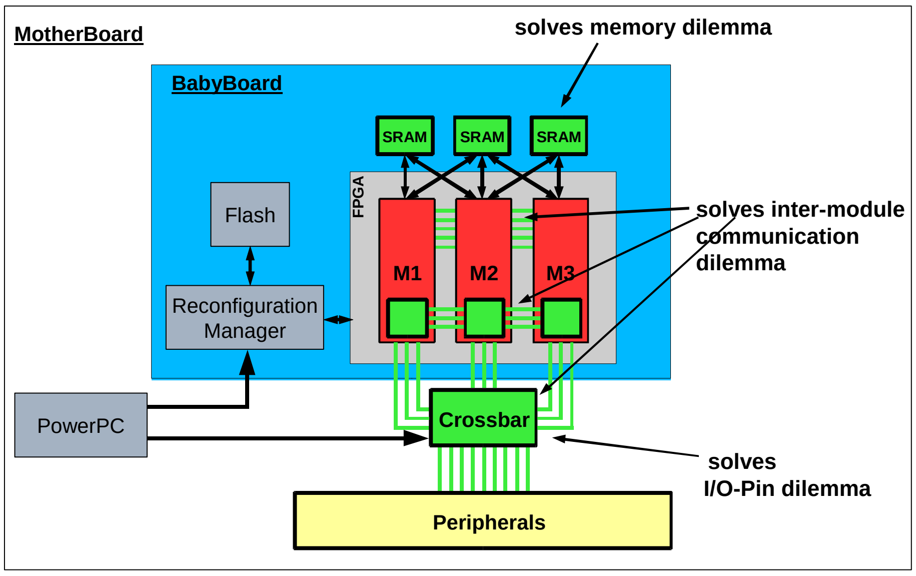

Fig. 1. ESM Architecture overview

The Virtex-II 6000 FPGA consists of 33792 slices arranged 96 CLB rows and 88 CLB columns. We logically divide the 88 columns into 22 micro-slots (A to V) that provide the module and reconfiguration granularity [6, 7]. Three consecutive micro-slots define a macro-slot. Each macro-slot (S1 to S6) can access one full external SRAM bank. Therefore, modules design for a macro-slot can be placed in free macro-slot at run-time. In terms of slice count, a micro slot occupies 1536 slices (4 CLB columns) on the FPGA. Micro-slots A, K, L and V are special as slots A and V interface external pins and slot K, L contain BlockRAM. Application modules can use several micro-slots to implement its logic.

2.2 Advantages over 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. 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 if not special care is taken.
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 loaded exactly in a given region where it will have access to its pins. Another problem related to the pin dilemma 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 accessing to use a device will have to feed many lines through many different components. Run-time relocation of such modules is only possible if all placement combinations of provided modules are considered and feed through lines are inserted at design time. This situation is not only present on Celoxica boards [8]. XESS boards [9], Nallatech boards [10], and Alpha boards [11] face the same limitation. On the XF-Board [12, 13] 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 [14], Celoxica RC1000 and RC2000 [8] 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.

With these limitations in mind, we designed a new and FPGA-based platform, called the Erlangen Slot Machine (ESM) for reconfigurable computing. The depicted architecture in Figure 1 addresses exactly these problems. A tool framework (which will not be addressed in this paper) is currently in development for easing the implementation of modular applications on the ESM.

2.3 Video and audio streaming

Video streaming can be defined as the process of performing computations on streaming video data. Many video algorithms process the data stream picture-by-picture. Usually a picture frame is transmitted pixel by pixel and therefore

can be processed on a pixel-by-pixel basis. However, since most algorithms need the neighborhood of a pixel to compute its new value, often at least one complete frame must be stored and processed before the next one can be accessed. Capturing the neighborhood of a pixel is often done using a sliding window the size of which varies according to the size of a neighbor region. A given set of buffers (FIFO) is used to update the window. The number of FIFOs varies according to the size of the window. In each step, a pixel is read from the memory and placed in the lower left cell of the window. Up to the upper right pixel which is disposed, i.e. output, all the pixels in the right part of the window are placed at the queue of the FIFO one level higher.
In the field of video compression, the processing is usually done in a block-by-block basis, different from the sliding window. However, the overall structure is almost the same.
As shown in Figure 2 the architecture of a video streaming system is usually built on a modular basis. The first module deals with the image captured from an image source. This can be a camera or a network module which collects the picture data through a network channel, or any other source. The frames are alternately written to the SRAM banks RAM1 and RAM2 by the capture module. The second module collects the picture from RAM1 or RAM2 if this RAM module is not in use by the first module, builds the sliding windows and passes it to the third module, which processes the pixel and saves it in its own memory or directly passes it to the next module. This architecture presents a pipelined computation in which the computational blocks are the modules that process the image frames. RAMs are used to temporally store frames between two modules, thus allowing a frame to stream from RAM to RAM and the processed pictures to the output.

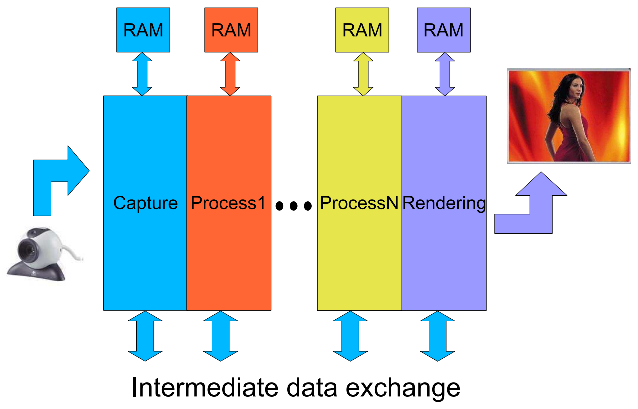

Fig. 2. A modular architecture for video streaming on the ESM

An adaptive video streaming system is characterized by its ability to optimize the computation according to changing environmental condition. In most cases, only one module on the computation chain must be changed while the other keep on running. The video capture module, for example, can be changed if we want to optimize the conversion of pixel to match the current brightness or the current landscape. It is also possible to change the video source from camera to a new one with different characteristics. In an adaptive system, the functionality of a module on the computation path should be changed very fast without affecting the rest of the system. This can be done by providing some parameters to the module to instruct it to switch from one algorithm to the next one. However, the structures of the basic algorithms are not always the same. A Sobel filter can not change into a Laplace filter by just changing the parameters. This is also true for a Median-operator which cannot be replaced by a Gauss-operator by just changing parameters. Network and camera require two different algorithms for capturing the pixels. In many cases, the complete module should be replaced by a module of the same size, but different in its structure while the rest of the system keeps running.

Our architecture fulfills the prerequisites for a modular pipelined and adaptive system for video streaming. In the system architecture presented before, we divided the device in slots, which each of them can implement a given module. RAMs are provided on one the upper part of the device while the lower part can be used by modules to communicate with the rest of the world.

3 Reconfiguration manager

The ESM platform requires an operating system for the initialization of executable application modules and their supervision. The tasks of this operating system are a) scheduling of application modules, b) management of free slots, c) loading, unloading and relocation of application modules, d) configuration of peripheral devices, e) configuration of the crossbar, and f) bitstream management.

In our view, the most-time critical operations must be executed in hardware in order to keep the reconfiguration time to a minimum. We consider the loading, unloading and relocation of modules to be the most time-critical tasks which will be therefore implemented in a dedicated hardware reconfiguration manager. All other system tasks can implemented in C and executed on the PowerPC embedded processor (see Figure 1) belonging to the ESM MotherBoard. These two parts of the operating system are linked via a simple communication bus as shown in Figure 1.

The reconfiguration manager must perform the following three elementary tasks:

- LOAD (load bitstreams to their precompiled position)
- UNLOAD (unload bitstreams to deactivate a running module)
- RELOCATE_AND_LOAD (relocate bitstreams to a different slot position before loading)

in the fastest possible fashion. Therefore, the reconfiguration manager will be a hardware module located in the small Spartan-2E 100 FPGA which is connected to the main FPGA via the SelectMAP interface.

During normal operation the bitstream data will be loaded from the flash memory located on the BabyBoard. However, bitstreams must be first downloaded and stored into the flash memory. Here, two methods are forseen. The first method uses a parallel port interface implemented directly in the reconfiguration manager to download the configuration data from a PC to the flash memory. The second method uses the Ethernet port of the PowerPC processor on the MotherBoard to download bitstreams from a remote host. In order to support these and other reconfiguration scenarios, we developed a very flexible, plugin-based reconfiguration manager.

3.1 Flexible plugin architecture

Our first data block-oriented reconfiguration manager consisted of a simple state machine which controls all interfaces and operates on byte blocks. These data blocks of size 512 bytes correspond to the page size of the flash memory device. For each primitive operation on a data block, an instruction must be processed. When one data block has to be written from flash into the Virtex-II SelectMAP interface two instructions must be processed. First, the data block is read in 512 cycles from the flash device and written to an internal scratch pad. Then, the second instruction is read and the data block from the scratch pad is written to the SelectMAP interface. As all instructions were executed sequentially, the maximum upload speed of a bitstream to the FPGA is slowed down by factor two, due to the exclusive access to the scratch pad.

However, the main problem with this architecture arises when more plugins and extensions were to be added to the reconfiguration manager. If now an error correcting code (ECC) plugin and a decompression plugin are used, then the speed degradation will increase to a factor of six, because four additional instructions are needed for reading and writing the scratch pad. This scenario is illustrated in the second part of Figure 3. An additional maintenance issue is the global finite state machine itself. Its code base has to be changed every time a new plugin is added or removed.

Clearly, this first data block-oriented architecture is not suitable for a high performance solution, since the throughput decreases with every new attached plugin. The main bottleneck is not the flash interface but the scratch pad oriented data flow combined with the sequential execution of the instructions.

Based on these consolidated findings, we developed a second architecture for the reconfiguration manager which can upload the bitstreams into the FPGA at the speed of the flash interface. The central scratch pad was eliminated and replaced by a pipelined data flow architecture. Moreover, the finite state machine was replaced by a PicoBlaze compatible microcontroller [15] and data crossbar switch is employed between the plugins. This new architecture is depicted in Figure 4. The crossbar plugin shown in this figure connects the reconfiguration manager control implemented now in software on the PicoBlaze to the

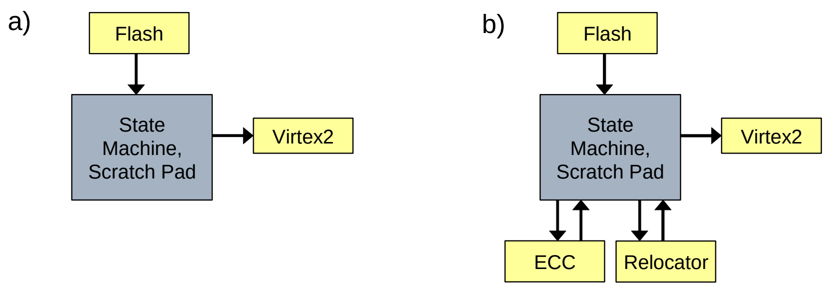

Fig. 3. Simple reconfiguration manager architecture

ESM MotherBoard in order to establish the communication link to the PowerPC shown in Figure 1.

All plugin modules are connected to two communication interfaces. The control bus connects plugins to the PicoBlaze for initialization and control. The data crossbar connects to the data input and output ports of each plugin and its connection setup is also controlled by the PicoBlaze which is programmed in assembly language.

In order to upload a bitstream from the flash to the FPGA, following sequence of steps has to be performed:

- A commando is sent to the PicoBlaze to upload a bitstream to the FPGA without the use of any other plugins.
- The PicoBlaze connects the output of the flash plugin to the Virtex-II plugin input through a write into the configuration register of the data crossbar.
- Next, the PicoBlaze initializes the flash plugin with the start address and length of the bitstream.
- The PicoBlaze then enables the SelectMAP interface in the Virtex-II plugin.
- Then, the PicoBlaze enables the flash plugin to start reading the bitstream.
- The flash plugin sends the bitstream to the Virtex-II plugin byte by byte as long as its ready signal is true (if not, the flash plugin has to wait).
- While the flash and the Virtex-II plugin are running in parallel, the PicoBlaze checks periodically if any of the plugins has finished its operation.
- Only if after finishing one command, the PicoBlaze can execute a new command and reinitialize the plugins and the data crossbar.

The addition of plugins to the reconfiguration manager is simple. Any new module must have a fixed control bus interface and a fixed data crossbar interface. With these standard interfaces the plugin can be directly controlled through the PicoBlaze assembly language. The data crossbar uses a parametrized HDL description which can be configured at design time to the number of actually instantiated plugins.

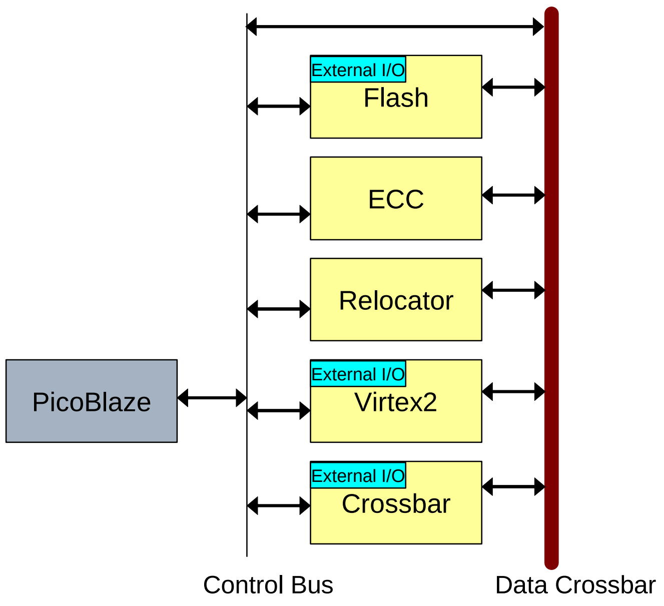

Fig. 4. Architecture of the ESM reconfiguration manager with plugins such as ECC, module relocator and other possible plugins.

3.2 Workload scenarios

Depending on the operating system requirements, different operations can be performed on each bitstream. Before the bitstream is uploaded to the FPGA it can pass through any number of additional plugins. The order of the plugins is configurable at run-time through the setup of the data crossbar switch. This allows a flexible preprocessing of the bitstream prior its execution. However, the number of available plugins in the reconfiguration manager has to be determined at design time.

Based on the introduced reconfiguration manager architecture from Figure 4, then several data flows are possible. Some of these are depicted in Figure 5. In the first scenario, only a basic upload of a bitstream is performed. Therefore, the data flows from the flash plugin output directly through the data crossbar to the Virtex-II plugin input. If an error correction is needed, then the flash output data can be send to the ECC plugin before going to the Virtex-II plugin. This case is shown in Figure 5 b). In the third scenario, the bitstream is read from the flash, error corrected and relocated before being send to the Virtex-II plugin for upload (see Figure 5 c)). Here, the crossbar is configured by the microcontroller in such a way that the output of each plugin is connected to the input of its neighboring plugin. The fourth scenario depicted in Figure Figure 5 d) how the bitstream data is delivered by the PowerPC through the MotherBoard crossbar. The bitstream is error corrected and relocated prior to its upload.

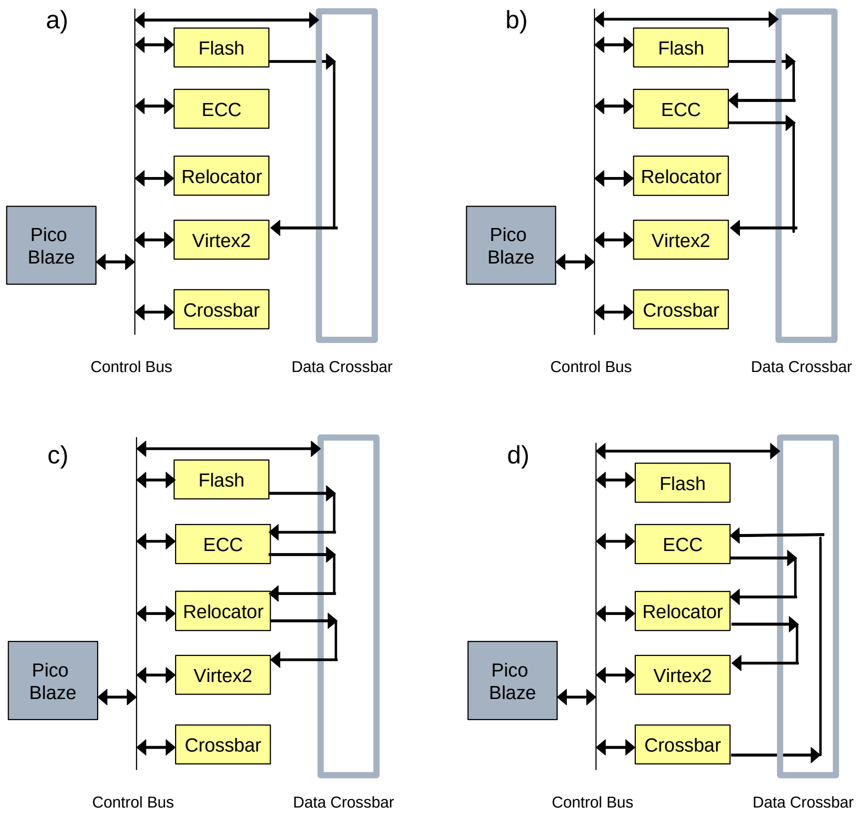

Fig. 5. Four different workload scenarios for the reconfiguration manager

Envisioned plugins for the reconfiguration manager are: ECC plugin, decompression plugin and a relocator plugin which can move the bitstream to any slot location on the FPGA by directly manipulating the address offsets in the bitstream at load-time.

3.3 Implementation results

The reconfiguration manager was implemented and consists of the PicoBlaze compatible microcontroller, parallel port interface plugin, flash memory interface plugin, VirtexII SelectMAP plugin, a Whisbone interface for the control bus and the data crossbar. The control bus is a single 8 bit shared bus, while the data crossbar is a 8 bit full duplex crossbar. The whole design runs at 50 MHz and occupies 360 out of 1200 logic slices plus one BlockRAM of the Spartan2E-100 FPGA.

The flash plugin interface is able to sustain a data rate of 10 Mbyte/s in a conservative and tested timing setup. As the SelectMAP interface can upload bitstreams with 50 MByte/s, an additional decompression plugin would accelerate the reconfiguration time when used on compressed bitstreams.

The final board implementation of the BabyBoard and MotherBoard is shown in Figure 6. The reconfiguration manager is implemented in the Spartan2E-100 FPGA which is located close to the 64 MByte flash device and the main VirtexII-60000 FPGA.

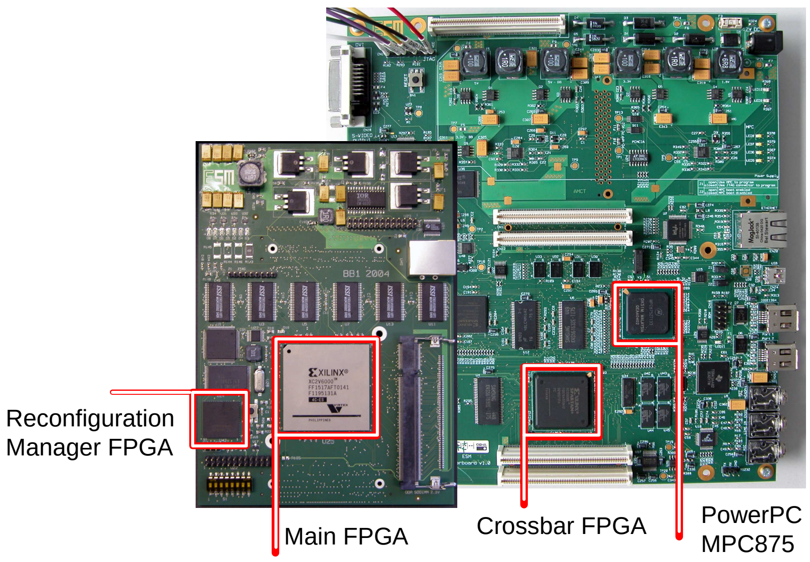

Fig. 6. Implementation of the ESM BabyBoard and MotherBoard

4 Conclusions

We have presented a flexible and fast hardware reconfiguration manager architecture concept for the ESM platform. After a short introduction of the ESM architecture, we have shown why it is more suited for partial and dynamic reconfiguration than existing platforms. We presented four different workload scenarios of a flexible reconfiguration manager based on plugins and motivated its constant high performance in different scenarios. The results were presented and comparison to a previous implementation was also made. Future work includes the addition of the relocation plugin to the reconfiguration manager. This will allow us to finally reallocate any bitstream at run-time without any time penalty. The ESM platform will then be used to demonstrate the partial reconfiguration of application modules which actively use the input and output peripherals of the ESM MotherBoard.

References

1. Ahmadinia, A., Bobda, C., Fekete, S., Teich, J., van der Veen, J.: Optimal routing-conscious dynamic placement for reconfigurable devices. In: Proceedings of International Conference on Field-Programmable Logic and Applications. Volume 3203 of Lecture Notes in Computer Science (LNCS)., Antwerp, Belgium, Springer (2004) 847–851
2. Bazargan, K., Kastner, R., Sarrafzadeh, M.: Fast template placement for reconfigurable computing systems. IEEE Design and Test of Computers **17** (2000) 68–83
3. Majer, M., Teich, J., Bobda, C.: ESM - the Erlangen Slot Machine. (http://www.r-space.de)
4. Bobda, C., Majer, M., Ahmadinia, A., Haller, T., Linarth, A., Teich, J., Fekete, S.P., van der Veen, J.: The Erlangen Slot Machine: A highly flexible FPGA-based reconfigurable platform. In: Proceeding IEEE Symposium on Field-Programmable Custom Computing Machines. (2005) 319–320
5. Krasteva, Y., Jimeno, A., Torre, E., Riesgo, T.: Straight method for reallocation of complex cores by dynamic reconfiguration in Virtex II FPGAs. In: Proceedings of the 16th IEEE International Workshop on Rapid System Prototyping, Montreal, Canada (2005) 77–83
6. Bobda, C., Majer, M., Ahmadinia, A., Haller, T., Linarth, A., Teich, J.: Increasing the flexibility in FPGA-based reconfigurable platforms: The Erlangen Slot Machine. In: Proceedings of the IEEE Conference on Field-Programmable Technology, Singapore, Singapore (2005) 37–42
7. Bobda, C., Ahmadinia, A., Majer, M., Ding, J., Teich, J.: Modular video streaming on a reconfigurable platform. In: Proceedings of the IFIP International Conference on Very Large Scale Integration, Perth, Australia (2005) 103–108
8. Celoxica Ltd.: RC2000 Development Board. (2004) http://www.celoxica.com/products/boards/rc2000.asp.
9. Xess Corp: (FPGA Boards) http://www.xess.com.
10. Nallatech, Inc.: (FPGA Boards)
11. Alpha Data Ltd.: ADM-XRC-II Xilinx Virtex-II PMC. (2002) http://www.alpha-data.com/adm-xrc-ii.html.
12. Platzner, M., Thiele, L.: XFORCES - executives for reconfigurable embedded systems. (http://www.ee.ethz.ch/~platzner)
13. Steiger, C., Walder, H., Platzner, M., Thiele, L.: Online scheduling and placement of real-time tasks to partially reconfigurable devices. In: Proceedings of the 24th International Real-Time Systems Symposium, Cancun, Mexico (2003) 224–235
14. H. Kalte, M. Porrmann, U.R.: A prototyping platform for dynamically reconfigurable system on chip designs. In: Proceedings of the IEEE Workshop Heterogeneous reconfigurable Systems on Chip (SoC), Hamburg, Germany (2002)
15. Xilinx, Inc.: (MicroBlaze CPU) http://www.xilinx.com.