Bridging the Gap between Relocatability and Available Technology: The Erlangen Slot Machine

— Dagstuhl Seminar 06141 —

Diana Göhringer, Mateusz Majer and Jürgen Teich

Department of Computer Science 12, University of Erlangen-Nuremberg
91058, Erlangen, Am Weichselgarten 3, Germany
{goehringer, majer, teich}@informatik.uni-erlangen.de

Abstract. We present an FPGA-based reconfigurable platform called Erlangen Slot Machine (ESM). The main advantages of this platform are: First, the possibility for each module to access peripherals independent from its location through a programmable crossbar, and local SRAM banks for individual modules. This physical design eases the implementation of run-time reconfigurable partial modules and enables an unrestricted relocation of modules on the device. We present our two-board ESM implementation and demonstrate a partially reconfigurable video filter application as well as a relocatable computer game including a dedicated inter-module communication scheme.

Keywords. FPGA-based reconfigurable platform, inter-module communication, crossbar, video filter demo

1 Introduction

Xilinx FPGAs[1] are one of the few partially reconfigurable devices that provide enough logic resources to efficiently implement applications such as video, audio and signal processing but also applications from other fields such as the automotive sector. Dynamic partial reconfiguration offers the advantage to time-share the resources on the device, which means, that a part of the device gets reconfigured while the rest does not get interrupted and stays fully functional. This means, that instead of implementing a design statically on a big device, a smaller device can be used on which an implemented module, which is actually not in use, can be replaced by a new one. These modules are pre-compiled and stored as bitstreams on-board. *Relocation* offers an advantage with regard to *partial reconfiguration* as here the same bitstream is used for all locations while for partial reconfiguration an individual bitstream is stored for each location. This reduces the amount of required on-chip memory.

Due to many restrictions, which will be explained in detail in Section 2, the development of a dynamically partial reconfigurable design and especially a relocatable one is very difficult. So far no FPGA-based platform on the market

offers sufficient support for the implementation of relocatable modules. Therefore, the idea of the *Erlangen Slot Machine* (ESM)[2,3,4] was born. Its mission is to fully support dynamic reconfigurability. This means to support resource sharing among different modules at run-time placed in fully relocatable module slots. This paper is organized as follows: Section 2 gives an overview of the current deficiencies with respect to dynamic partial reconfiguration and relocation of existing reconfigurable computing platforms. In Section 3 the ESM and its advantages are presented. Section 4 describes the different inter-module communications supported by the ESM. In Section 5 the implementation of the famous Pong computer game on our ESM is presented. Also, we present a dynamically partial reconfigurable video filter application. Finally, in Section 6 our conclusions and ideas for future work are provided.

2 Current deficiencies of existing reconfigurable computing platforms

Dynamic partial reconfiguration and especially dynamic module relocation on current existing FPGA-based reconfigurable computers are restricted by the factors mentioned in the following subsections:

2.1 Tool dilemma:

Very few FPGAs allowing partial reconfiguration exist on the market. Those few FPGAs, like the Virtex series from Xilinx[1], allow only a restricted amount of partial reconfiguration. The reconfiguration is done column-wise meaning that loading a module in a given area will affect all the modules above and below the module.

2.2 I/O-pin dilemma:

Most of the existing platforms offer I/O peripherals such as video, audio, memory devices, etc. which are connected with the FPGA through fixed I/O-pins. Therefore, the relocation of a module using these I/O-pins is impossible or very difficult. Another issue is that I/O-pins belonging to a peripheral, e.g. video, are not situated next to each other. Instead they are spread around the device, which forces modules, that want to access this device, to feed signal lines through several other modules. An example for such a situation is demonstrated in Fig. 1. Here, a static VGA module is connected to several I/O-pins on the right hand side and the bottom of the device. Therefore, the implementation of a relocatable module using no feed-through lines is restricted to the two first columns on the left side of the device. To implement a relocatable module on more than these two columns together with the VGA module a very high design effort is required.

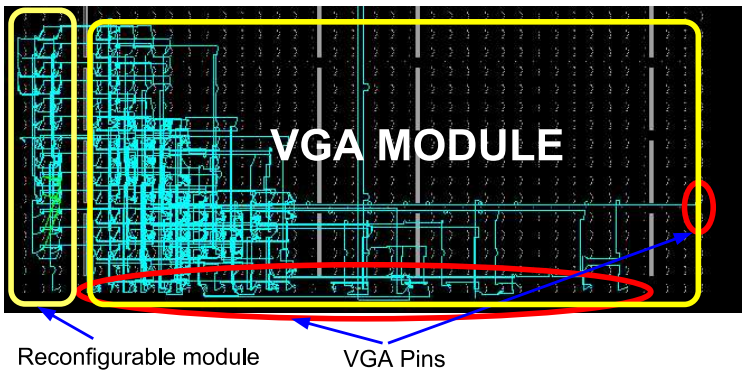

Fig. 1. Used I/O-pins of the VGA module on a RC2000 platform[5].

2.3 Inter-module communication dilemma:

As the position of run-time placed modules is not known at compile-time a dynamic inter-module communication scheme is necessary. To route such signal lines dynamically is difficult. Therefore, new communication schemes such as packet-based DyNoCs[6] or a switch-based Reconfigurable Multiple Bus (RMB)[7] must be investigated.

2.4 Memory dilemma:

Some applications such as image or video processing require large amounts of local memory. The internal memory available in a physical slot area is often not enough or this would require a bigger slot and therefore wasting resources. So using external memory is a good solution, but most reconfigurable devices only offer few external memories, whose I/O-pins are spread over the device, which results again in the I/O-pin dilemma mentioned above. To solve these issues and to ease the implementation of dynamically relocatable designs a new FPGA-based platform called Erlangen Slot Machine (ESM) was designed.

3 Architecture of the Erlangen Slot Machine

The Erlangen Slot Machine[2,3,4,8] is physically divided into two boards called *BabyBoard* and *MotherBoard*. The main reconfigurable engine is implemented on a Xilinx Virtex-II 6000 on the BabyBoard. The application-specific devices together with a Crossbar FPGA are implemented on the MotherBoard. The connection between the two boards is realized through the *Crossbar* (see Fig. 2). This separation offers more modularity as the BabyBoard can be connected with different MotherBoards depending on its application requirement, e.g. Multimedia, Automotive. The actual developed MotherBoard is designed for multimedia applications. Therefore, peripherals needed for audio, image and video applications were integrated into this board.

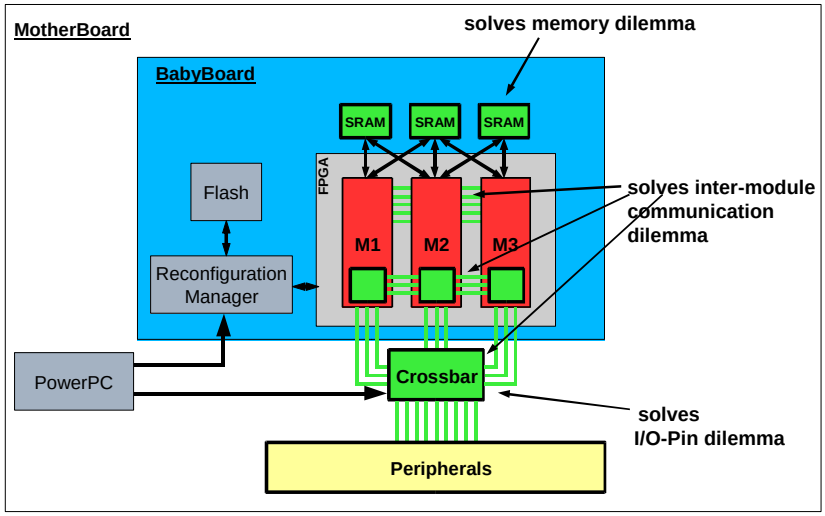

Fig. 2. Architecture of the ESM.

3.1 BabyBoard:

A Xilinx Virtex-II 6000 FPGA working as a reconfiguration engine is the heart of the BabyBoard. This FPGA is logically divided into 22 so-called *Microslots*, each 4 CLB columns wide. The reconfigurable modules can be placed onto one or more Microslots depending on their requirements. Due to this slot-based architecture the name Erlangen Slot Machine was chosen. The arrangement into slots and the homogeneity of the device eases the relocation as each slot contains an equal amount of resources. To the north of this device 6 SRAM banks, 2 MByte each, are connected. They offer local memory for the modules or can be used for a shared memory communication between modules placed directly next to each other as it is often used in streaming applications for example. The bottom I/O-pins of the main FPGA are connected over the Crossbar implemented on a Xilinx Spartan-II FPGA on the MotherBoard to the peripherals of the MotherBoard. Thereby, each module can access the needed peripherals independent of the slot in which it is actually placed. Another important device on the BabyBoard is the *Reconfiguration Manager* implemented on a Xilinx Spartan-II FPGA, whose task is to dynamically reconfigure the main FPGA with partial or full bitstreams stored in the on-board Flash device. An additional task is the relocation of a partial bitstream. Due to this design two of the deficiencies mentioned in the previous section are solved. These are the memory dilemma which is solved by adding six SRAM banks to the north of the device and preventing their I/O-pins from being spread over the whole FPGA boundary. Also the I/O-pin dilemma is solved by connecting the I/O-pins of the bottom of the device to the programmable Crossbar, which connects these pins with the different peripherals

implemented on the MotherBoard. Several solutions for solving the inter-module communication dilemma will be given in Section 4.

3.2 MotherBoard:

On the MotherBoard, all application specific peripherals such as video, audio, Ethernet, USB, etc. are implemented together with an embedded PowerPC processor (MPC875). Linux is running on this processor offering a shell environment called *ESM-Shell* which can be accessed over the Ethernet by the user to control the board. Using this the developer can send commands for reconfiguration/relocation to the Reconfiguration Manager on the BabyBoard. Also the Crossbar can be dynamically programmed over the PowerPC depending on the peripheral to pin connection needed by the actual running application.

4 Inter-module communication

On the ESM we offer four different paradigms for inter-module communication. The first two are applicable only for adjacently placed modules, while the second two can be used also for modules not placed next to each other (see Fig. 3).

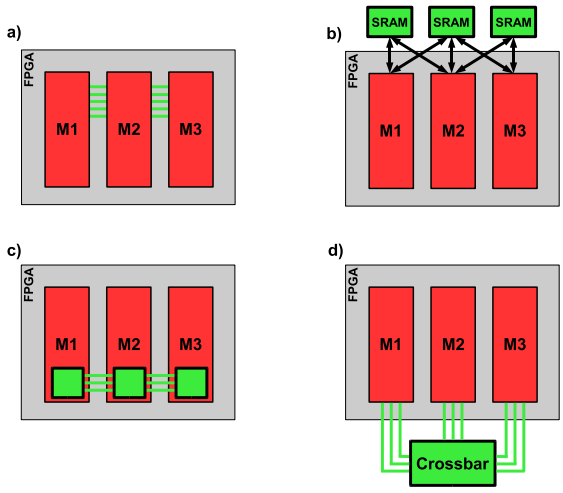

Fig. 3. Available inter-module communication media for the ESM are: a) busmacros, b) SRAMs, c) RMB, d) programmable crossbar.

4.1 Communication via busmacros:

Unidirectional busmacros can be used to build communication channels between adjacently placed modules (see Fig. 3a)). Each busmacro allows to wire 8 single

bit lines together. This means for n signals at least n/8 busmacros have to be instantiated. In Section 5.2 we present a dynamically partial reconfigurable video filter application, which uses busmacros as a communication interface between the static part of the design and the reconfigurable slot.

4.2 Communication via SRAM:

The six external SRAM banks can be used for shared memory communication between adjacently placed modules. This means that each SRAM bank can be accessed by the module placed below it and by the direct neighbors of this module (see Fig. 3b)). The access priority is controlled by a memory controller. This way of communication is very useful in streaming applications such as video processing, where a large amount of data needs to be forwarded to the next module after it has been processed.

4.3 Communication via Reconfigurable Multiple Bus (RMB):

An RMB[7,9,10] is a bundle of bus segment lines that allows to establish individual circuit-switched horizontal interconnections between slot modules (see, e.g., in Fig. 4c)). It consists of a set of processing elements, which are called *crosspoints* (CP). Each slot has its own CP and the CPs are connected with each other through switchable bus segments. To generate a new connection between a sender and a receiver, the sender has to send a request signal in the direction of the receiver. This is done in a wormhole fashion. Each CP forwards the request until it reaches the corresponding receiver CP, who then sends an acknowledgment back to the sender. Each CP, which receives this acknowledgment connects two line segments by activating the corresponding switch and forwards the acknowledgment until it reaches the slot, who had send the original request. When the acknowledgment has reached the origin of the request, the corresponding two slots can start to communicate in a latency-free manner with each other, until the sender issues an explicit destroy signal. To insure a correct operation during reconfiguration, busmacros are inserted at the boundary of the module and the CP. In Section 5.1 we present the famous computer game Pong, whose modules use the RMB to communicate with each other.

4.4 Communication via Crossbar:

The programmable crossbar on the MotherBoard, which is connected to the I/O-pins on the bottom of the slot-based main FPGA on the BabyBoard, can finally also be used as a communication medium for adjacently as well as not adjacently placed modules.

5 Applications on the ESM

In the following subsections, we describe two different applications which we implemented on our ESM platform and which serve to describe a) relocation, and b) high speed partial reconfiguration capabilities.

5.1 Computer game Pong demonstration

To demonstrate the functionality of the ESM and to test the RMB bus structure, we implemented the famous computer game Pong onto the ESM (see Fig. 4a)). This game was divided into four modules that use the RMB bus structure to communicate with each other(see Fig. 4b)).

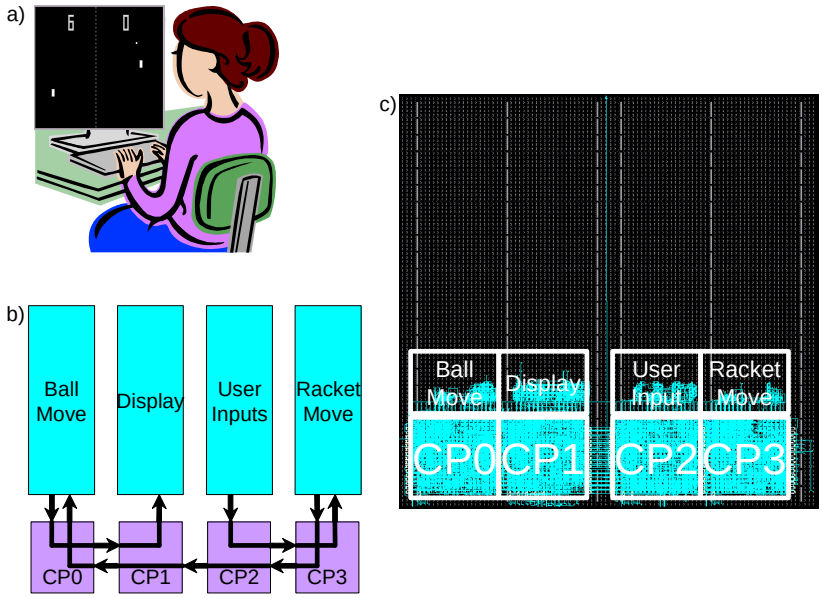

Fig. 4. Implementation of the computer game Pong using an RMB[7,9,10] (reconfigurable multiple bus) structure with four crosspoints (CP0,...,CP3). This allows any of the four application modules (*Ball Move*, *Display*, *User Input* and *Racket Move*) of the video game to be placed in any of the four displayed slots. The figure is divided into 3 different views: a) computer game pong, b) schematic of the implementation and the communication between the modules, c) screenshot of the implementation using the FPGA Editor from Xilinx.

The four modules are:

1. **User Input:** The user sends its steering commands, over RS232 to the embedded PowerPC on the MotherBoard that forwards the commands to the Crossbar. The Crossbar then sends these signals to the corresponding I/O-pins of the User Input module. This module then sends the commands to the module that calculates the movement of the two rackets.
2. **Racket Movement:** After receiving the steering commands from the User Input module the positions of the rackets get calculated and are forwarded to the module that calculates the movement of the ball.

3. **Ball Movement:** This module calculates the position and movement of the ball in correspondence to the actual racket position. After finishing with the calculations, the new position of the ball as well as the positions of the rackets are forwarded to the module that is responsible for the visualization.
4. **Display:** As soon as this module receives the new positions of the ball and the rackets it updates the screen output, by sending the corresponding signals to its I/O-pins on the FPGA. These pins are connected with the video output module on the MotherBoard over the programmable crossbar.

Fig. 4c) shows a screenshot of the game implementation on the ESM using the FPGA Editor from Xilinx. As one main reason for this implementation was to test the resource overhead for RMB implementations, the here seen crosspoints are much bigger than required for the actual Pong game. In the shown implementation each CP is implemented to be able to connect around 100 single bit lines to each of its neighbors, and its local module. As can be seen in the picture, the RMB occupies around 10% of the slices. Besides the modules mentioned above, additional modules were designed, such as a *Fast Ball Movement*, a *Slow Racket Movement* and an *Autoplayer*. These partial modules can be reconfigured into a slot on the board within the range of a millisecond at runtime,(partial reconfiguration).

5.2 Dynamically partial reconfigurable video filter application

As another demonstration for partial reconfiguration on the ESM we designed a video filter application with four filter types (*none*, *inversion*, *contrast*, *gray*), which can be dynamically partial reconfigured. These filters communicate with the static part of the design over busmacros. The application is implemented as follows (see Fig. 5a)): First, the video input device on the MotherBoard sends the data received from the camera to the programmable crossbar. The crossbar then forwards this data to the main FPGA on the BabyBoard, where it is deinterlaced before it is send to the reconfigurable filter. After filtering the data is send over the crossbar to the video output device which is connected to a screen in order to display the filtered image. The SRAM banks connected to the north of the main FPGA on the BabyBoard are used to store the video stream frames before and after deinterlacing. Fig. 5b) shows a screenshot from the FGPA Editor from Xilinx of the implementation of the video filter application.

Again for this application, only very few columns need to be reconfigured at run-time. For example, we may exchange the following filters: normal, inversion, contrast and gray within the range of a millisecond.

6 Conclusions and Future Work

In this paper, we presented two applications actually running on our new FPGA-based reconfigurable platform called ESM. The flexibility and homogeneity of this platform supported the design of these applications. Different communication structures were also conceptualized and implemented depending on the

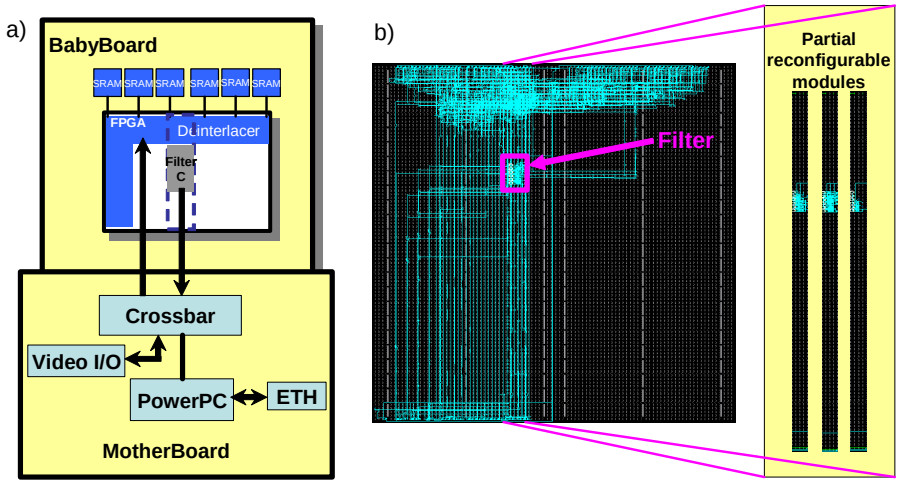

Fig. 5. Partial reconfigurable video filter application: a) schematic view, b) view of the implementation using the FPGA Editor from Xilinx.

needs of the applications. We also demonstrated the support for relocation and reconfiguration offered by our platform, which is visualized in Fig. 6.

In the shown demonstrations, reconfigurations are still initiated by human interaction. In the future, we are planning to improve the capabilities in such a way that the reconfiguration as well as the relocation will be provided by the Reconfiguration Manager. The Crossbar, which already can be programmed via the ESM-Shell, will be programmed accordingly to where a module is relocated so that this module can communicate with arbitrary external peripherals. Also, the video demonstration will be extended to use the external SDRAMS available on the MotherBoard instead of the SRAMS as used so far. Thus, the static part of the demo, which does the deinterlacing for example, can be processed on the MotherBoard. Thereby, only the real reconfigurable modules will be placed on the main FPGA offering more space for dynamic hardware modules. Also, more complex filters will be added such as a Sobel-Filter and a Laplace-Filter. Finally, an additional tool called *SlotComposer* that supports the design of relocatable modules and especially their communication is in development. In particular, the SlotComposer tool automatically instantiates the busmacros or the RMB-crosspoints in the main design file. Thereby, saving the user a lot of design time. Also, it automatically generates batch-files to run the partial design flow thus generating the final partial bitstreams.

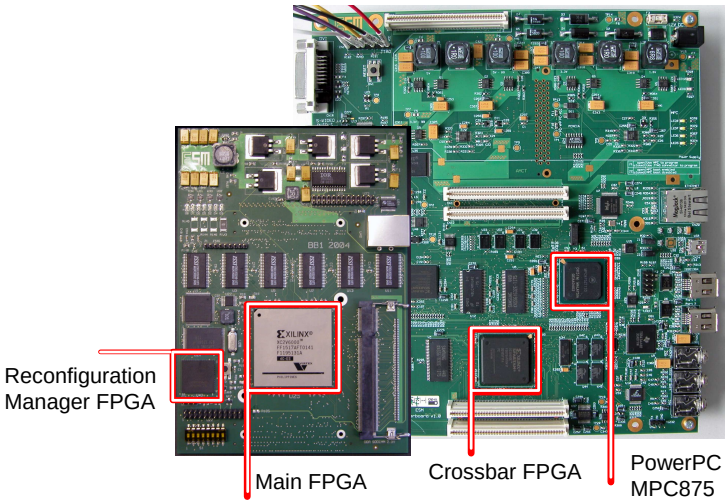

Fig. 6. ESM: BabyBoard and MotherBoard. The BabyBoard is mounted on the application-specific MotherBoard, in this case a multimedia board, using 4 connectors.

Acknowledgments

We would like to thank at this point the German Science Foundation (DFG) for funding this project under title "ReCoNodes" within the priority program SPP 1148, and also Xilinx Research Labs in San Jose for additional support.

References

1. Xilinx, Inc.: FPGAs. (2006) http://www.xilinx.com.
2. Majer, M., Teich, J., Bobda, C.: ESM - the Erlangen Slot Machine. http://www.r-space.de (2005)
3. 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
4. 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
5. Celoxica Ltd.: RC2000 Development Board. (2004) http://www.celoxica.com/products/boards/rc2000.asp.
6. Ahmadinia, A., Bobda, C., Majer, M., Teich, J., Fekete, S., van der Veen, J.: DyNoC: A dynamic infrastructure for communication in dynamically reconfigurable devices. In: Proceedings of the International Conference on Field-Programmable Logic and Applications, Tampere, Finland (2005) 153–158

7. Ahmadinia, A., Ding, J., Bobda, C., Teich, J.: Design and implementation of reconfigurable multiple bus on chip (RMBoC). Technical Report 02-2004, University of Erlangen-Nuremberg, Department of CS 12, Hardware-Software-Co-Design (2004)
8. Majer, M., Teich, J., Ahmadinia, A., Bobda, C.: The Erlangen Slot Machine: A Dynamically Reconfigurable FPGA-Based Computer. Journal of VLSI Signal Processing Systems (to appear) (2006)
9. ElGindy, H.A., Somani, A.K., Schröder, H., Schmeck, H., Spray, A.: RMB - a reconfigurable multiple bus network. In: Proceedings of the Second International Symposium on High-Performance Computer Architecture (HPCA-2), San Jose, California (1996) 108–117
10. Ahmadinia, A., Bobda, C., Ding, J., Majer, M., Teich, J., Fekete, S., van der Veen, J.: A practical approach for circuit routing on dynamic reconfigurable devices. In: Proceedings of the 16th IEEE International Workshop on Rapid System Prototyping, Montreal, Canada (2005) 84–90