Modular Video Streaming on a Reconfigurable Platform

Christophe Bobda, Ali Ahmadinia, Mateusz Majer, Ding Ji, Jürgen Teich
University of Erlangen-Nuremberg, Department of computer sciences 12
Am Weichselgarten 3, 91058 Erlangen, Germany
{bobda, ahmadinia, majer, teich}@cs.fau.de

Abstract

We present an efficient implementation of a video streaming application on a well suited reconfigurable platform, the Erlangen Slot Machine (ESM) for partial reconfiguration. The architecture of the video streaming is investigated to derive the requirements that should be fulfilled by a viable partial reconfigurable platform. Various edge detection methods are implemented on the same structure based upon three modules: frame capture, frame processing and frame rendering. The frame processing modules perform different edge detection operations.

1. Introduction

Video streaming can be defined as the process of performing some computations on video frames, streamed on a system picture by picture. The computation on the frames can be done in different steps and therefore presents a nice structure for a pipelined computation. Most of the video streaming systems are built on a chain structure made upon a set of modules. The first module on the chain is in charge of capturing the video frames, while the last module usually deals with the rendering of the frames. Between the first and the last modules, several computational modules can be used according to the algorithm implemented. A module can be implemented in hardware or in software according to the goals. While software provides a great degree of flexibility, it is usually not fast enough to carry the challenging computation required in video application. ASICs can be used to implement the computational demanding parts, however ASIC does not provide the flexibility needed in many systems. On a reconfigurable platform, the flexibility of a processor can be combined with the efficiency of ASIC in order to build a high performance flexible system. Field Programmable Gate Arrays (FPGA) belong to the class of existing reconfigurable devices available to buy. Today's FPGAs offer enough logic to implement many modules of a video streaming chain. Moreover, many FPGAs can be partially reconfigurable, thus providing the possibility to replace a given module on the chain. Several video application have been implemented in the FPGA. However, none of them use the reconfiguration to increase the flexibility of the system. This is in part due to the architecture of the boards and systems used. In this paper, we present a new reconfigurable platform designed for multimedia computation as well as some implementations of a video streaming application. The platform is organized in exchangeable slots able to accommodate some modules in a video streaming chain. Each slot can be reconfigured to perform a different computation. The communication infrastructure available on the platform provides an unlimited access of module to their data, no matter on which slot they are placed. The rest of the paper is organized as follows. Section 2 presents a short overview on the existing work. In section 3, we present the structure of a modular image processing system and explain how reconfiguration can be used to increase the flexibility. Section 5 presents the Erlangen Slot Machine.

2. Related work

Video processing applications involve different processes like video capturing, image enhancement, and object detection [8]. Implementing such applications on a general purpose computer can be easier but not very efficient in term of speed. The reason being the additional constraints put on memory and other peripheral device management. FPGA offers much greater speed than a software implementation. Fahad Alzahrani et al. [3] present a high performance edge detection VLSI architecture for real time image processing applications, the architecture is fully pipelined. Richard G.S [13] discusses the idea of parameterized program generation of convolution filters in an FPGA. A 2-D filter is assembled from a set of multipliers and adders, which are in turn generated from a canonical serial-parallel multiplier stage. Atmel application notes [5] discuss 3x3 convolver with run-time reconfigurable vector multiplier in Atmel FPGA. To overcome the difficulties of pragramming devices with classic Hardware Description Languages(HDL) like VHDL and

Verliog, Celoxica [1] has offered a tool for its multimedia platform, which supports a simple language Handel-C. Handel-C is essentially an extended subset of the standard ANSI-C language, specifically designed for use in a hardware environment. The Handel-C compiler comes packaged with the Celoxica DK1 development environment. The Sonic architecture [10] uses configurable computing to accelerate offline and real-time video image processing. The design consists of plug-in processing elements (PIPEs) connected by the PIPE bus and PIPEflow buses. Sonics architecture exploits the spatial and temporal parallelism in video image processing algorithms. The Splash/Splash 2 [4] is an attached Processor System using Xilinx XC4010 FPGAs as processing elements, implemented as a linear, systolic array. It is well suited to image processing due to high parallelism and specialization (filtering).

3. General structure of video streaming applications

Video streaming can be defined as the process of performing some computation on video data while on their streaming through a given system. Most of the algorithms process on the stream picture by picture. The frames are usually transmitted pixel by pixel and therefore can be processed on a pixel by pixel basis. However, since most algorithms need the neighbourhood of a pixel to compute its new value, a complete frame must be stored and processed before the next one can be accessed. Capturing the neighbourhood of a pixel is often done using a sliding window which its size varies according to the size of a neighbour region. As shown in Figure 1, a sliding window is a data structure used to sequentially capture the neighbourhood of pixels in a given image. A given set of buffers (FIFO) are

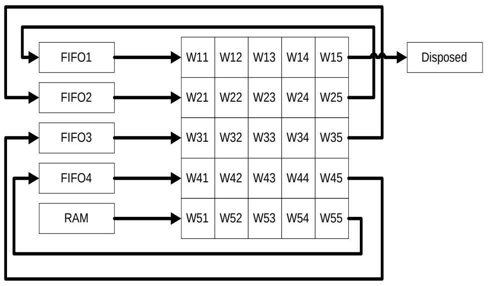

Figure 1. Implementation of a 5x5 sliding windows

used to update the windows. The number of FIFOs vary 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 pixel in the right part of the window are placed at the queue of the FIFO one level higher.

The processing part of the video is a normal image-processing algorithm using the basic operators like:

- Median filtering
- Basic Morphological operations
- Convolution
- Edge detection

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 for 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 through a network channel, or any other source. The frames are alternately written to the RAM1 and RAM2 by the capture module. The second module collects the picture from the RAM1 or RAM2 if this 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 data 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.

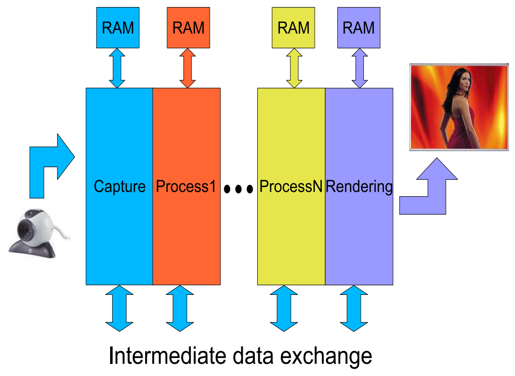

Figure 2. A modular architecture for video streaming on the ESM

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 rest keep 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 or the source can be for instance a network channel. 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 structure of the basic algorithms are not always the same. A Sobel filter can not change in 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 the parameter. The network and the camera requires two different algorithms for capturing the pixels. In many cases, the complete module should be replaced by a module with the same size, but different in its structure while the rest of the system keep running. Reconfigurable devices in general and FPGAs in particular fulfill this requirements. Many available FPGAs can be partly configured while the rest of the system is running. Moreover many FPGAs provide a rich set of small size on-chip memory, able to hold part of a frame (the so called region of interest). It is therefore possible to perform many computations in parallel on different regions of interest, thus increasing the performance of video applications.
Almost no existing FPGA platforms does provide the prerequisites required in an adaptive and modular video streaming system. The main problems are the following:

- **Unlimited external access of module:** Each module on an intermediate level on the pipeline chain should have access to the external word for a data exchange. For example to send the result of a computation done at its level or to collect more data required in the computation at its level. The connection of the module to the external world should not depend on the position of the module in the chain.

- **Efficient storage of frame between the modules:** The pipelined processing requires that frames must be efficiently stored and managed between the modules. The placement of the RAM between the modules as well as their exclusive access to the RAM have to be considered.

- **Logical and physical grouping of interconnection pins:** In many platforms, the connection of the pins are spread around the device without any logical grouping. This makes it difficult to implement partial bitstreams which can be used to reconfigure part of the device at run-time. The main problem here is that module using some pins which are not in their placement area must feed some lines through other modules, thus making the development of such modules difficult.

To overcome the limitations of existing system, we developed a new platform for partial reconfiguration called the Erlangen Slot Machine (ESM). Our architecture fulfils 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 th rest of the world.

4 Limitations of existing platforms

Partial reconfiguration on FPGAs is done by downloading a full or partial bitstream to the fPGA. Those bitstreams represent the resource used by a given modules on the device at a given time. Modules to be downloaded at run-rime on a reconfigurable device must first be developed at compiled time according to some rules and stored as partial bitstream to be downloaded at run-time on the FPGA. As illustrated in figure 3, a module using resources outside

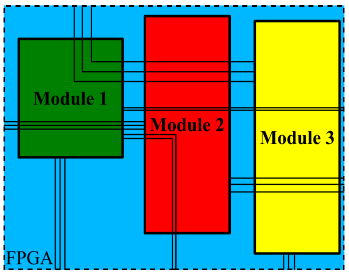

Figure 3. Routing of modules on an FPGA

its placement area must use **feed-through signals** through other modules in order to access its resources. Using feed-through lines to access resources makes the design automation difficult since each module must be implemented with all possible feed-trough channels needed by other modules to reach their resources. Moreover, module accessing external components through device pins are no more relocatable, because they are compiled for fixed locations where a direct access to their pins is possible.
Up to now, none of the most used available FPGA platforms [7], [11], [2], [12], [9], [6] provides solutions 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 our platform is to overcome the deficiency of existing FPGA Platforms by providing a new and highly flexible FPGA Platform in which each component must not be fixed all the time at a given chip location.

5 The Erlangen Slot Machine

The Erlangen Slot machine (ESM) is made upon a BabyBoard mounted on a MotherBoard. The separation of the system in two boards allows the BabyBoard which contains the reconfigurable module to be used on a wide variety of systems. For the integration of the ESM in a new platform, no redesign of the complete system must be done. Only a new MotherBoard must be provided according to the computational requirement in the new environment. A multimedia system for example will provide a MotherBoard with multimedia devices like video and audio. In an automotive system, the MotherBoard will mostly contain sensor and actuators. The BabyBoard can be mounted on a PCI MotherBoard in a computer system or on a stand alone MotherBoard in an embedded system.

5.1 The BabyBoard: Computation and reconfigurable engine

The reconfigurable engine of the ESM platform is a baby board which features a Xilinx Virtex II-6000 FPGA from Xilinx, several SRAMs and a configuration circuit. The Xilinx Virtex FPGAs are partially and column wise reconfigurable. The reconfiguration of a module in a given region will affect all the running module in the corresponding columns. We built our architecture to efficiently use this restriction.
As shown in figure 4, we logically divided the FPGA in vertical columns of 2 CLBs called *micro slots*, therefore providing 22 micro slots. According to its size, a module can be implemented in a given number of micro slots. The ESM platform 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 application like video streaming which is targeted here. A given module exclusively accesses a picture at a time for computation. Because the capacity of the available memory (BlockRAMs) in a module region is limited, external SRAM memory are therefore added to allow the storage of large amount of data by each module. To allow a module to exclusively access its memory, the SRAMs are connected at the top part of the FPGA. In this way, a module will have its connection to its peripheral from the bottom part and the top part will be used to temporally store the computation data. No module will need to feed its lines through other modules for accessing its resources.

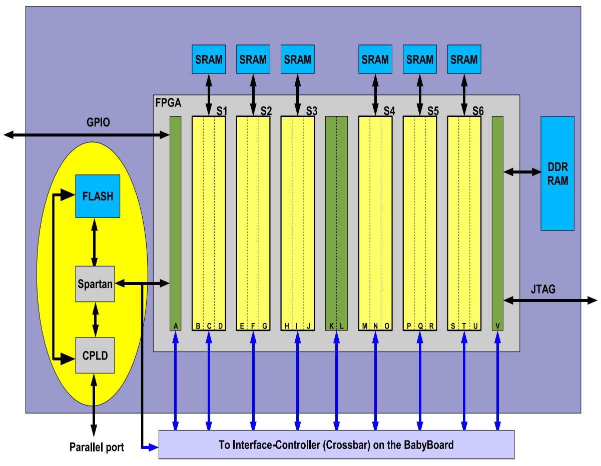

Figure 4. Architecture of the ESM-Baby board

As stated earlier, the device is divided in a set of exchangeable slots, each of which has access to the SRAM on the top part and to the crossbar at the bottom. Due to the number of pins needed to access one external SRAM, each module willing to use an SRAM module must be implemented in a multiple of 3 micro slots. Apart from the main FPGA, the BabyBoard also contains the configuration circuitry. This consists of a CPLD, a configuration FPGA that is a small Spartan II FPGA and a Flash. The CPLD is used to download the Spartan II configuration from the flash on power up. It also contains board initialization routines for the on board PLL and the Flash. The configuration program for the main FPGA is implemented in the Spartan II. Due to its small size, this reconfiguration program could be directly implemented in the CPLD, which could configure the main FPGA at power on. But we choose a much larger device to increase our degree of freedom in the reconfiguration management. The Flash provides a capacity of 64 MByte, thus enabling the storage of up to 32 Full configurations and few hundreds partial bitstreams. 6 SRAM banks with 2 MBtye each are vertically attached to the board on the top side of the device, thus providing enough memory space to six different slots for temporal data storage. The SRAMs can be also used for shared memory communication between neighbour modules in 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. Debugging capabilities are offered through General purpose I/O provided in regular distance between the basic slots. A JTAG port provides debug capabilities for the main FPGA, the CPLD and the Spartan II.
The ESM architecture provides a nice support for dynamic

placement of modules on a reconfigurable device. Placement of modules on a reconfigurable device, in this case the FPGA, is done by downloading a partial bitstream implementing a given task in the FPGA. The full and partial bitstreams represent circuits implemented on a given region of the device at compile time. In other words, to place a module in another location other than the location for which it was compiled, a relocation of the module must be done. The relocation of a given module therefore modifies the coordinates of all the resources used by the module in its previous location (the location in which the module was constrained at compile time) with the coordinates of the module in the new location. 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 include the device pins used by the module. If the connection of a module with its peripheral is done via some pins fixed at a given chip location, the module must be constrained at 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 peripheral on the FPGA.

5.2 The MotherBoard

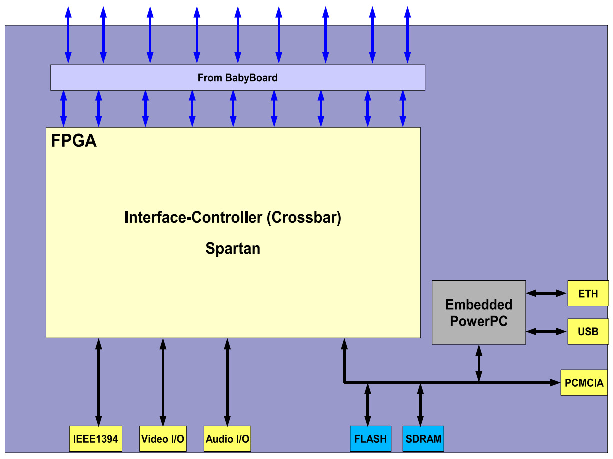

Figure 5. Architecture of the ESM MotherBoard

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 a Spartan chip on the MotherBoard. This crossbar functionality basically solves the I/O pin dilemma of many existing FPGA platforms, thus allowing free relocation of modules requiring I/O pin connectivity. Besides the run-time programmable crossbar, many peripherals for multimedia like video and audio input and output as well as communication interfaces are available on the board (Figure 5). Figure 6 shows a screenshot of the ESM under test.

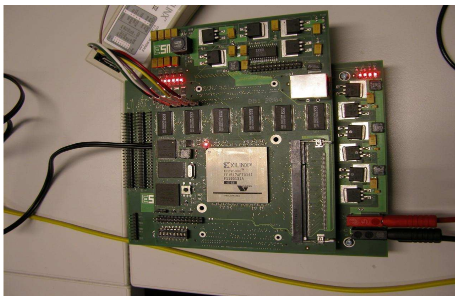

Figure 6. The ESM Platform under test

5.3 Streaming of video frames

As stated earlier, the frames of a video stream must be computed by modules and stored between the modules for further computation by the next module on the chain. In the FPGA, the frames can be stored in the memory blocks called BlockRAM availabe on the chip. Although they provide a nice possibility to implement applications in multiple clock domains, their limited size does not allow for saving large data containing in a complete frame. BlockRAMs can however be used to hold a small part of the frame (the region of interest) as required in many applications. Since we are interested in more general solution, we use the external SRAMs to temporally store the frames between the modules.
On the ESM, each module will be attached to an SRAM that is used for storing the Data. The streaming of data is only possible if modules can access their neighbour-SRAM for reading or writing. As illustrated in figure 7, an SRAM-controller is available in each module. The controller is implemented in a priority basis such that only one module can access the SRAM at a time. The input of the controller is the

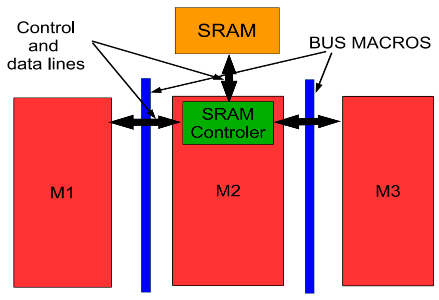

Figure 7. SRAM-based intermodule communication on the ESM

data and address lines of the central module as well as that of its two neighbours. The controller is in charge of passing the correct data to the memory according to the priority. The priorities may vary from one application to another one.
At the boundary of the modules, the interconnection are done by means of so called Bus macros which provide fixed communication channels in FPGAs. The purpose of the Bus Macros is to allow data to flow on secure channels at reconfiguration. With this each module on the streaming chain is reconfigurable and its reconfiguration will not disturb the application.

6 Case study

Our case study consists of a reconfigurable video streaming made upon three blocks. A video capture module that is in charge of collecting video data from the camera and performing some conversion like the YUV to RGB includes a processing module and a VGA rendering module. Each of the module can be reconfigured as explained in the introduction. However, we have focussed here only on the reconfiguration of the processing part. Each processing module implements a filter for edge detection with a given characteristics. The following filter have been implemented: the Sobel, the Prewit and the Laplace. While the Sobel and the Prewit have the same structure, it is not the same for the laplace and the Prewit and therefore, reconfiguration is used to replace the complete structure of a module without affecting the whole application. We are currently working on more performant filters like the Canny edge detector and the Hough transformation operation. Figure 8 shows a screenshot of the edge detection using the Sobel operator. Emulating a region of interest with the quarter size of the VGA screen, we could use BlockRAMs in the modules for processing.

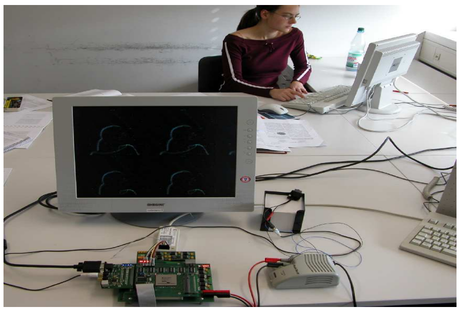
Figure 8. Edge detection on the ESM with 4 ROI

7 Conclusion

In this paper, we presented a video streaming processing on a reconfigurable platform, the Erlangen Slot Machine. We first focussed on the streaming, modular and pipelined structure of video streaming application and derived the requirements that should be fulfilled by a viable computation platform. We then presented the ESM with its advantages for computation in video streaming. We presented the implementation of a case study. Various edge detection methods are implemented on the same structure based upon three modules: frame capture, frame processing and frame rendering. The frame processing modules implement different edge detection operations. In the future, we intend to investigate the use of embedded processors to help performing control intensive operation as it is the case in face detection, object recognition.

References

[1] Celoxica limited. `http://www.celoxica.com`.

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

[3] F. Alzahrani and T. Chen. A real-time high performance edge detector for computer vision applications. In *Proceedings of the 1997 Asia South Pacifi c Design Automation Conference (ASP-DAC)*, pages 671–672, 1997.

[4] J. M. Arnold, D. A. Buell, D. T. Hoang, D. V. Pryor, N. Shirazi, and M. R. Thistle. The splash 2 processor and applications. In *ICCD*, pages 482–485, 1993.

[5] ATMEL Corporation. webpage: . *Coprocessor Field Programmable Gate Arrays*. `www.atmel.com`.

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

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

[8] P. M. Curry, F. Morgan, and L. Kilmartin. Xilinx fpga implementation of a pixel processor for object detection applications. In *Proceeding of the Irish Signals and Systems Conference, UCD, Dublin*, pages 404–411, 2000.

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

[10] S. D. Haynes, J. Stone, P. Y. K. Cheung, and W. Luk. Video image processing with the sonic architecture. *Computer*, 33(4):50–57, 2000.

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

[12] M. Platzner and L. Thiele. XFORCES - executives for reconfi gurable embedded systems. http://www.ee.ethz.ch/p̃latzner.

[13] R. G. Shoup. Parameterized convolution fi ltering in an FPGA. In *Proceedings of International Conference on Field Programmable Logic and Arrays*, pages 274–280, 1993.