RAPID PROTOTYPING PLATFORM FOR RECONFIGURABLE IMAGE PROCESSING

B.Kovář [1], *J. Kloub* [1], *J. Schier* [1], *A. Heřmánek* [1], *P. Zemčík* [2], *A. Herout* [2]

(1) Institute of Information Theory and Automation
Academy of Sciences of the Czech Republic
Pod vodárenskou věží 4, Prague 8, Czech Republic
e-mail: {kovar, kloub, schier, hermanek}@utia.cas.cz

(2) Graph@FIT, Brno University of Technology
Božetěchova 2, Brno, Czech Republic
e-mail: {zemcik, herout}@fit.vutbr.cz

Abstract

Image processing and computer vision is being used in a variety of applications including applications in embedded devices (surveillance, quality control, traffic monitoring, etc.). In these applications, the data are used in real-time (in the traffic monitoring and decision-making sensors for example). Real-time video processing is computationally demanding, making microprocessor-based processing impossible. Moreover, microprocessor DSPs is energy inefficient, which can be a problem in industrial power-limited embedded systems. The trend of exploiting embedded low-power and low-cost devices for image processing and computer vision is becoming stronger. Using reconfigurable computing, it is possible to achieve high computational performance while the target system remains relatively inexpensive. This paper outlines a proposed architecture for an integrated partial rapid prototyping dynamic reconfigurable system-on-a-chip, targeted at embedded real-time video and image processing.

1 Introduction

Image processing and computer vision algorithms are very computationally demanding tasks which fact increases the importance of well selected hardware platform and carefull and fine-tuned design. At the same time the application market is quickly spreading which reduces the time available for design of individual applications. These facts increase the demand for acceleration hardware and for embedded systems and/or systems-on-chip processing video signal and reasonable design tools. In response to that, different acceleration platforms and specialized chips exist today. The scale ranges from fast, low-power signal processors [1], through processors equipped with specialized video input/output and acceleration modules, to specialized single-purpose chips [2].

The combination of general-purpose, sequentially programmed low-power processors and combinational single-purpose chips (though reconfigurable, as in the case of Field-Programmable Gate Arrays – FPGA's), was also proven useful, allowing optimal distribution of tasks with different nature of computational load [3]. Such platforms include different computational environments, firmware interconnecting them, different development and simulation tools etc., making the development process complex, requiring long training times for the engineers and long development cycles. As an answer to this situation, advanced tools for the MATLAB/Simulink environment from the MathWorks, Inc. (www.mathworks.com), or 3rd party products, compatible with Mathlab (e.g. Synplify DSP), that support DSP development, rapid hardware prototyping and hardware in the loop simulation, have become available in the recent years. A number of extensions are available for DSP, image processing, fixed-point computations (natively, the MATLAB/Simulink environment uses double-precision floating-point operations). The hardware-oriented extensions include most notably the System Generator for DSP, AccelDSP, both from Xilinx. Inc and the Synplify DSP. Naturely, all these tools provide support for FPGA devices.

These tools represent an important step towards simplification of application development for FPGA. Let us recall at this point that our goal is to develop rapid prototyping support for image processing applications, using specialized platform, as will be described in the next section. While the above mentioned tools can be conveniently used to develop the necessary computational blocks, they lack built-in dedicated support for image and video processing.

This paper addresses two important parts of the system: hardware architecture for embedded FPGA/DSP based image processing system and methodology of dynamic reconfiguration allowing efficient usage of the hardware platform under consideration.

2 Reconfiguration in Image processing Applications

Computer vision and image processing applications consist of several phases with different computer architectural needs. As already mentioned, image processing is an area where there is much interest in parallel architectures. The motivation for this is the computationally-intensive nature of processing algorithms. Filtering and features extraction from a PAL resolution image at 25 frames per second requires throughput around 700 MOPS. Pre-processing stage is usually necessary. Computation at this stage (low-level processing) is mostly deterministic so the data parallelism can be exploited. An example of low-level processing is the task to find edges within an image, which might be done by passing a set of Sobel spatial masks over every pixel in image. This operation is localized to the nine pixel neighborhood of the processing pixel. Many others image processing tasks can be interpreted in a same sense. An example of high-level processing is to identify objects in the image using edges found by Sobel operators.

Considering reconfiguration, three forms of reconfiguration can be discerned:

- Prosessor is reconfigured at load time – static reconfiguration
- The algorithm topology is changed at pre-determined points or phases of a program execution – quasi-static reconfiguration.
- Reconfiguration can occur at arbitrary time during program execution.

An advantage of using reconfigurability is that the same hardware can be reused for several tasks, increasing thus the efficiency of its utilization. It is particularly well suited for low and intermediate image processing tasks where several parallel computational cores can be swapped in the same hardware.

The algorithms used in this paper were not selected because they were especially suited to reconfiguration but because they are a part of a typical computer vision or image processing software. We do not make a case that dynamic reconfiguration is essential but we try to show that it can be used end would make older (low-cost) image processing architecture more attractive.

3 Overall Platform Design

The reconfiguration experiments with FPGA programmable circuits require not only the reconfigurable hardware (FPGA) itself but also some suitable environment that controls the reconfiguration and is able to exploit it. For the purpose of RIPAC project, we have selected a system based on an FPGA connected to a compact CPU that controls the configuration and stores the configuration data in its memory. The CPU should be interconnected with the FPGA with fast data connection that ensures that the reconfiguration is performed rapidly.

For our experiments, we have chosen an existing DX64 board (section 3.1) that was available and up to date at the beginning of the project knowing that while the design itself can become (and will become) obsolete, the above mentioned idea is fulfilled. The TI C6416 DSP used along with the Xilinx Virtex II on the DX64 board is well suitable for our purposes. The

DSP is using its 16bit data bus to connect to the FPGA and the 64 bit bus remains free for connection of the main memory, the FPGA is seen by the DSP as a component in its memory space and the data connection between the DSP and FPGA can be done in SSRAM mode which fact ensures data transfer speeds of up to 200 MByte/sec which is well suitable for reconfiguration. The reconfiguration itself can be done under the control of the DSP itself (through external DX64 pins) or from a host system which can be also DX64 or other host system, such as a host PC.

The platform chosen for experiments does not introduce any loss of generality as the architecture can be preserved even if recent components are used, e.g. newer members of the TI C64 series along with e.g. the Xilinx Virtex IV reconfigurable chips.

The original intention regarding the proposed platform was to create an environment for embedded image processing in such form that would allow for rapid development of image processing applications on embedded systems. Today, the features of the system include:

- Programming models on the embedded system (DSP/FPGA) and the PC are compatible (C/C++ or scripting language).
- Embedded scripting language allows for development of the applications on a host PC without a need to learn the embedded system development software.
- The scripting language allows to access libraries of functions implemented alternatively on FPGA, embedded processor, or host PC processor.
- The environment allows for tracking of values of named variables and allows for multithreading/multiprocessing supported with the scripting language.

3.1 Uni1P/DX64 platform

The hardware used for experimental work and evaluation of the above presented ideas is UNI1-P/DX64 platform by CAMEA, spol. s r o. which is shown in Figure 1. The UNI1-P board

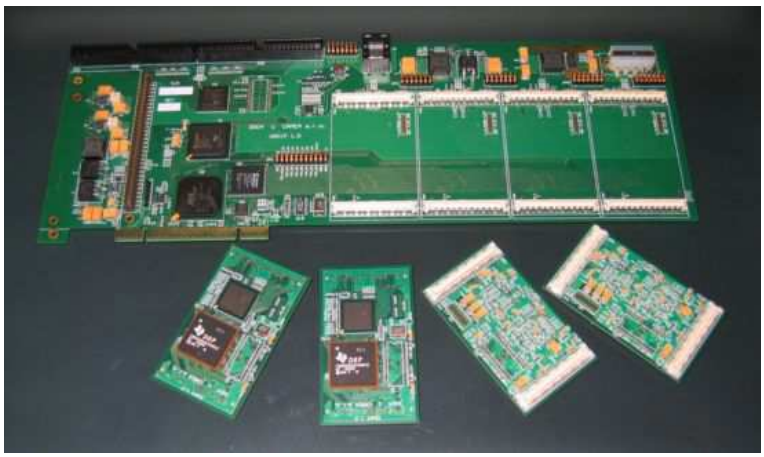


Figure 1: The UNI 1-P/DX64 platform

presents an interface between the host PC and up to four DX64 computational units. It also provides temporary storage for signal/image data in DRAM and communication channels to outside world data. Its block diagram is shown in Figure 2 below.

The computational units DX64 themselves are generally useable computational units consisting of a DSP (Texas Instruments C6416) with dual bus interface used for connection of DRAM and an FPGA (Xilinx Virtex II), and a host interface used for communication with the

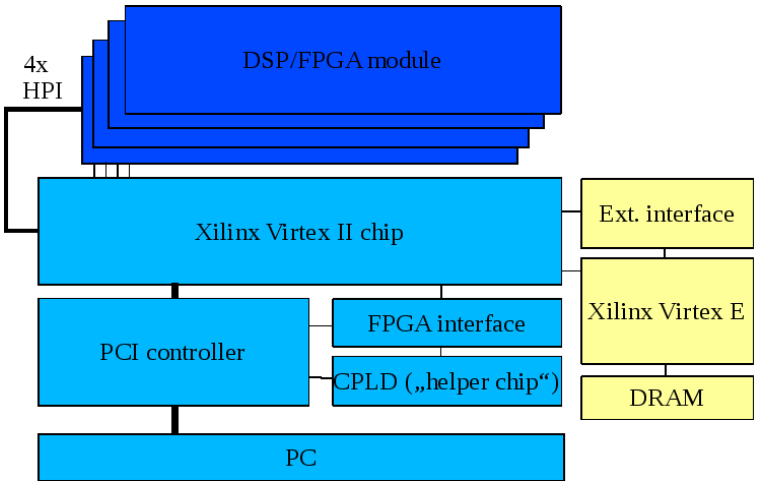


Figure 2: UNI1-P block diagram (the yellow parts are optional)

host system. The subset of the FPGA pins is connected to an external connector of the DX64 in order to provide communication means. In the presented case, these FPGA pins are used for high-speed communication with the other DX64 units and the host UNI1-P board. See the block diagram of DX64 module in Figure 3.

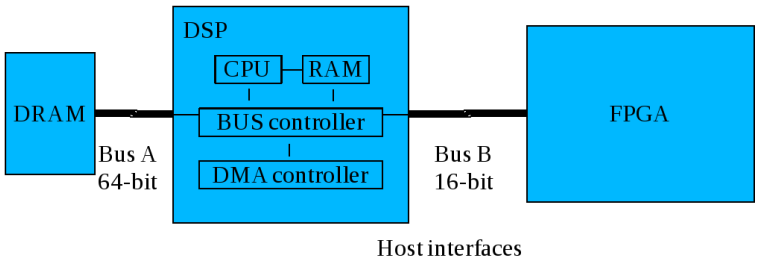


Figure 3: DX64 block diagram

The general scheme of the DX64 board can be seen as a generic and repeatable design useable in visually any design with the given class of DSPs and thus presents virtually no loss of generality from the point for view of usage of different hardware.

4 FPGA Reconfiguration

Implementations of image processing functions, such as de-noising, edge detection etc., on FPGA (Field Programmable Gate Array) [4] benefit from the parallelism of the device. FPGA can hold several computational blocks running in parallel, each processing part of an image. but hardware resources of the circuit are often limited. The Xilinx chip, used in our device, provides an option of dynamic reconfiguration of the circuit, that is, to change circuit configuration and interconnection during application runtime. FPGA can be reconfigured fully or partially – with the later called the partial dynamic reconfiguration. The part of the programmable array which is not changing during runtime is called the static part and the part which is reconfigured is called the dynamic part.

An image processing application typically uses several operations over an image. Since the operations are often not needed in the same time and/or the hardware resources may lack due to limited capacity of the device, dynamic reconfiguration can be an interesting option: the operations that are not used in the same time, can be swapped in the dynamic part. Reconfiguration time could be overlapped by using another functional block in FPGA. Second approach for using dynamic reconfiguration in image processing is to decompose complex general purpose

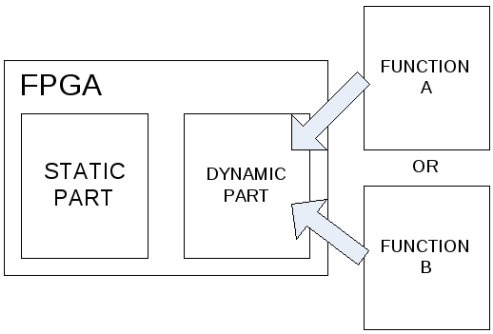


Figure 4: Partitioning of an FPGA area to the static and dynamic part

functional block to several smaller specialized functional blocks. Decomposition increases circuit utilization and performance due to optimization of specialized functional blocks.

5 Experiments and Results

In our experiments with the Uni1P/DX64 system, the system FPGA drives communication between PCI controller interface, DX64 Host Port Interface (HPI) and configuration interfaces (Slave SelectMAP) of all FPGAs on the Uni1p board. Configuration interface data throughput is around 24MB/s (Table 1).

Table 1: The system overall edge detection benchmark. The throughput is in MB/s

Img. Size	RunTime [s]	ComTime [s]	DX64 Throughput	System Throughput
512 × 384	0.008314	0.236824	22.55	0.792
752 × 564	0.016507	0.400683	24.50	1.009
1024 × 768	0.03062	0.633103	24.49	1.185

An image detector is implemented in the FPGA on the DX64 module. The image detector is implemented using partial dynamic reconfiguration: part of implementation is without changing during runtime, the detector can be replaced with different functionality. The Table 2 summarizes partial dynamic reconfiguration memory and configuration latency saving.

In Figure 6, an FPGA Editor Tool view of static and dynamic part of FPGA is shown. There is the communication interface implemented in the static part, which is never changed during runtime and in the dynamic part, the image edge detector is implemented which can be replaced with another function.

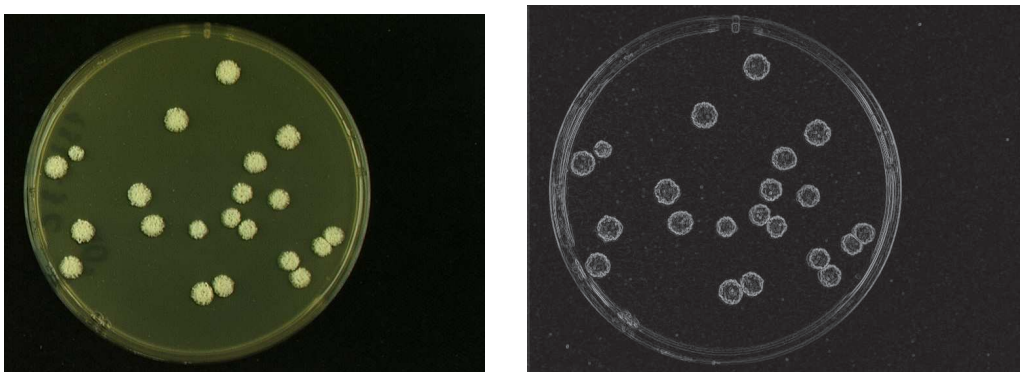
Using uniform interface between static and dynamic part of FPGA enables developing of another function with easy way for example with Synplify DSP ad-on to the Simulik product from The MathWorks. Synplify DSP provides designers to implement DSP circuits in FPGA and ASIC devices. Image processing functions including mentioned edge detector is designed using Synplify DSP 3.6 development tool (Figure 5).

Design described with graphical representation of dataflow graph with functional block in

Table 2: Partial dynamic reconfiguration memory and latency saving summary

Bit-stream	Size	Latency
Full	199 204 Bytes	7,9 ms
Partial	63 396 Bytes	2,5 ms
Saving	68,18%	68,35%

Table 3: The edge detection on a real image data sets



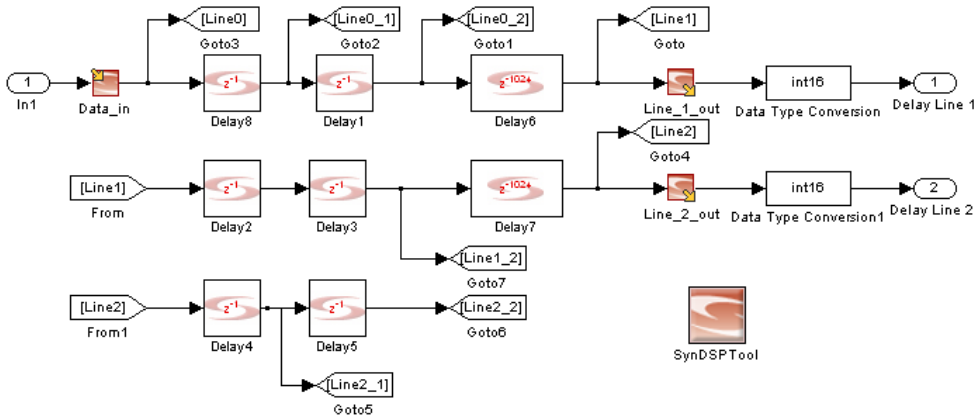


Figure 5: The edge detection implementation using Synplify DSP

Simulik is exported to VHDL format. VHDL format contains behavioral description of designed function which is simply placed as component in partial reconfiguration design flow supported in Xilinx ISE 9.1.02i_PR2 development tools. Partial bitstream is generated for new function only which can be directly used in application without changing another hardware and software resources.

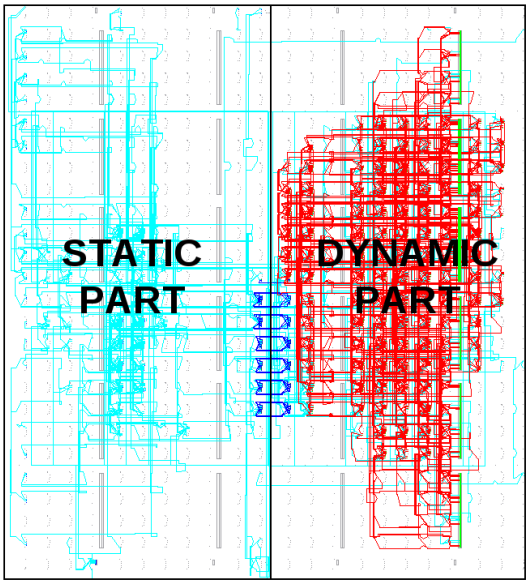


Figure 6: FPGA Editor view of the image edge detection application

6 Conclusions

In our paper, we have focused on application of a runtime FPGA reconfiguration in a video processing system. An example of an FPGA/DSP-based architecture for embedded video processing has been described. The principles of reconfiguration have been outlined and paramerter of implementation of a simple Sobel filer have been presented. It has been shown that using the partial dynamic reconfiguration, interesting savings both in the bitsteam size and in the reconfiguration time can be achieved.

Acknowledgements

This work has been supported by the Rapid prototyping tools for development of HW-accelerated embedded image- and video-processing applications, GA AVCR, 1ET400750408 grant project, and by the Ministry of Education, Youth and Sports of the Czech Republic under the research program LC-06008 (Center for Computer Graphics), by the research project "Security-Oriented Research in Informational Technology" CEZMŠMT, MSM0021630528.

References

[1] TMS3B0C6711, TMS320C6711B Floating point Digital Signal Processors, Texas Instruments, SPRS088B, USA, (available at http://www.ti.com)

[2] Fujitsu MB91680A-T chip, www.fujitsu.com

[3] Fučík O. et al.: Imaging Algorithm Speedup Using Co-Design, In: Summaries Volume Process Control 01, Štrbské Pleso, SK, 2001, p. 96-97, ISBN 80-227-1542-5

[4] Virtex^{TM}-E 1.8 V Extended Memory Field Programmable Gate Arrays, Xilinx, DS025-2 (v2.2), USA, (available at http://www.xilinx.com)