Architecture for Hardware Driven Image Inspection based on FPGAs

Johannes Fürtler[a], Jörg Brodersen[a], Peter Rössler[b], Konrad J. Mayer[a],
Gerhard Cadek[c], Christian Eckel[c], Herbert Nachtnebel[d]

[a]ARC Seibersdorf research GmbH, High Performance Image Processing,
A-2444 Seibersdorf, Austria
[b]University of Applied Sciences, Department of Embedded Systems*, Höchstädtplatz 5,
A-1200 Vienna, Austria
[c]Oregano Systems - Design & Consulting GmbH, Phorusgasse 8, A-1040 Vienna, Austria
[d]Institute of Computer Technology, Vienna University of Technology, Gußhausstraße 27-29/E384,
A-1040 Vienna, Austria

ABSTRACT

Requirements for contemporary print inspection systems for industrial applications include, among others, high throughput, examination of fine details of the print, and inspection from various perspectives and different spectral sensitivity. Therefore, an optical inspection system for such tasks has to be equipped with several high-speed/high-resolution cameras, each acquiring hundreds of megabytes of data per second. This paper presents an inspection system which meets the given requirements by exploiting data parallelism and algorithmic parallelism. This is achieved by using complex field-programmable gate arrays (FPGA) for image processing. The scalable system consists of several processing modules, each representing a pair of a FPGA and a digital signal processor. The main chapters of the paper focus on the functionality implemented in the FPGA. The image processing algorithms include flat-field correction, lens distortion correction, image pyramid generation, neighborhood operations, a programmable arithmetic unit, and a geometry unit. Due to shortage of on-chip memory, a multi-port memory concept for buffering streams of data between off-chip and on-chip memories is used. Furthermore, performance measurements of the processing module are presented.

Keywords: Real-Time Image Processing, System-On-Chip, FPGA, HW/SW Co-Design

1. INTRODUCTION

In the field of industrial print inspection, contemporary requirements include, among others, examination of fine details of the print, high throughput, and image acquisition from different views and in different spectral bands, e.g., color, infrared, and ultraviolet. Therefore, an optical inspection system for such tasks has to be equipped with several high-speed/high-resolution cameras, each producing megabytes of data. Figure 1 shows a symbolic facility for quality inspection of printed sheets[1]. The mechanical part consists of a loading station (A), a separator (B), a conveyor belt consisting of multiple parts (C), a switch (D) and trays for unfit (E) and fit (F) sheets. Along the conveyor belt there are two camera stations (G) and (H) to inspect the front side and the back side of the sheets. With regard to high-speed transportation of the sheets (some meters per second) each camera station is made up of several high-speed line-scan cameras, operating at line rates above 50 kHz and resolution of at least 1024 pixels, which is necessary to identify the fine details. The cameras differ in spectral sensitivity or they are arranged to observe the same scene from distinctive viewpoints†. Typical camera stations contain six to nine cameras. The information processing part consists of a machine

* partly funded by the Austrian FHplus research initiative in context to the DECS project (Design Methods for Embedded Control Systems)

† For acquisition of transmitted light the camera has to be arranged opposite to the lighting, therefore capturing light shining through the sheet. The sheet must not be touched by any transporting system part in this area, because this would prevent the light shining through the sheet. Additionally, for high quality image acquisition, the sheets have to pass the camera stations smooth and free of

Real-Time Image Processing 2006, edited by Nasser Kehtarnavaz
Phillip A. Laplante Proc. of SPIE-IS&T Electronic Imaging
SPIE Vol. 6063, 60630D, © 2006 SPIE-IS&T · 0277-786X/06/$15

control unit (I), a processing system (J), and a server (K) with some clients for user interaction attached to it. The machine control unit serves as interface to sensors and actuators of the machine, e.g., camera triggers, and keeps track of each sheet in the system. During operation of the machine the server continually downloads measurement results and raw image data from the processing system, stores the data, and provides them for the clients. On the other hand, the server offers additional services for controlling the processing system. The processing system collects and provides data, computes a quality decision, and triggers the switch accordingly. This paper deals with implementation aspects of the processing system.

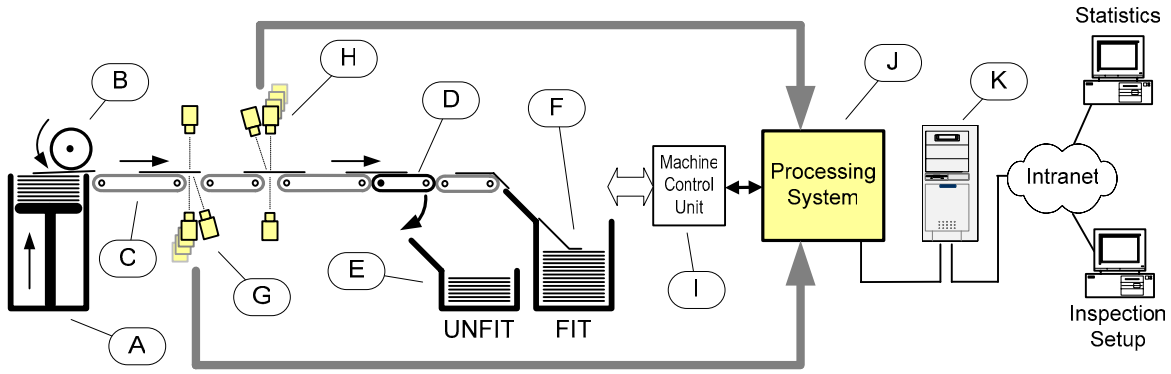

Figure 1: Symbolic example for a print inspection system.

The machine is fed with printed sheets and automatically separates faulty sheets from top-grade products according to user defined rules (inspection setup). During the process of inspection several sheets are simultaneously processed at different positions in the machine. This leads to the following requirements for the real-time processing system which must be handled:

- tens of sheets simultaneously processed by the machine at different stages,
- feeding rates up to 50 sheets per second,
- more than a Gigabyte of input data per second, and,
- computation of complex image processing tasks.

In the next sections we briefly discuss conceptional issues concerning the implementation of processing systems capable to meet the above requirements. However, the main focus is on a specific part of the system, namely, the implementation of image processing functions on a field-programmable gate array (FPGA), which we consider to be the key factor for technically flexible and, not less importantly, marketable solutions.

2. HARDWARE DRIVEN IMAGE PROCESSING

Traditional inspection systems as presented in Section 1 are often based on digital signal processors (DSP). High performance DSPs like the TMS320C6400™ series from Texas Instruments, which exploit fine grain parallelism through very large instruction set architectures and operation frequencies up to 1 GHz, enable timely computation of very complex algorithms[2]. However, memory bandwidth limitations prevent an image processing system solely based on a single DSP to handle the application mentioned above. Common parallel architectures based on multiple DSPs and/or dedicated hardware components are often either limited to one application or they are implemented in a general way which means overkill. Using multiple DSPs requires complex arbitration logic for accessing shared memories. Therefore, systems may not be implemented economically. Another drawback of too general solutions is that, typically, numerous processing units spend most of their time waiting for data from other units[3].

---

tension. These are great challenges regarding the mechanical design of a machine. However, mechanical and optical issues are beyond the scope of this paper.

On the other hand, advanced FPGAs offer a lot of parallel processing power which makes them a suitable platform for image processing tasks. FPGAs offer a great number of user pins (over 1000), including high-speed inputs and outputs, which makes them nearly (with respect to board layout) the optimal target for connection of many peripherals, e.g., memory modules and different external interfaces. Moreover, FPGA systems can be reconfigured, even at runtime, to meet dedicated application demands. Consequently, one hardware platform can be used for several basically different applications. Unfortunately, on-chip memory resources in today's FPGAs are limited, which restricts their practical use in some applications, especially for image processing tasks. Another disadvantage, compared to DSPs, is the poor processing power for common sequential (one-dimensional) computations.

Our basic approach was to make use of the benefits of FPGAs and DSPs while reducing the deficiencies. However, there are several difficulties that must be overcome. From our viewpoint, many previous attempts did not optimally utilize the FPGA features or they were so specialized that they, again, could only be used for one application. The key issue for designing high performance real-time image processing systems is to match algorithms and architecture. It is essential to use common hardware/software co-design methodologies to find a balance between algorithms implemented in hardware (FPGA) and algorithms running as software tasks on a DSP. With the proposed hardware driven image processing (HDIP) architecture a flexible solution for this problem is presented. For the HDIP architecture, we adhered to common design principles[2] for high-speed real-time image processing systems like parallel processing, pipelining, and multi-port memory concepts. Enabled by contemporary FPGA devices, the original contribution of our approach is the practical application of these principles to build flexible inspection systems based on simple building blocks. Resulting systems should be scalable in terms of the number of attached cameras (20 or more) and scalable for arbitrary processing power. Thereby, a wide range of applications may be covered.

Starting from this perspective, the typical inspection system consists of several processing modules (PM) which are interconnected in a ring topology. The actual number of PMs depends on the application. For example, the input provided by three different cameras may be processed by one PM. Other special features which need additional processing power, e.g., character recognition, may require an additional PM. So the inspection system is scalable by using an appropriate number of PMs. One PM implements the physical connection to the machine control unit and controls the whole inspection system. Therefore, this PM serves as a master to the other modules.

Figure 2 shows the main modules of a PM. It typically consists of a FPGA and a DSP. A PM may be equipped with only one of them. For example, the master PM contains only the DSP part and, instead of the FPGA, it is equipped with additional interfacing capabilities. The FPGA and the DSP are interconnected via a high-speed bus providing bi-directional data transmission. For application reasons, it is necessary to store a number of raw images acquired by the cameras for later usage. The server, connected to each PM via Gigabit-Ethernet (GBE), may download data at any time. Therefore, the download must not interfere with real-time behavior of the system.

The DSP serves as a master to the FPGA and controls the data flow. The DSP is also heavily involved in computing complex image analysis algorithms. In order to minimize amount of data to be transferred between the DSP and the FPGA, advanced data reduction based on image analysis takes place within the FPGA.

3. FUNCTIONALITY IMPLEMENTED IN THE FPGA

The FPGA design (also referenced as the HDIP design) has been implemented using VHDL (Very High-Speed Integrated Circuits Hardware Description Language) and is therefore independent from the target technology, e.g., FPGA or application specific integrated circuit (ASIC). However, some technology dependent FPGA resources have been used, e.g., memory blocks and DSP blocks. The same applies for intellectual property (IP) cores supplied by Altera. These modules have to be adapted according to the underlying technology.

A main module of the HDIP design is a multi-port memory interface to fast off-chip memory modules. It allows multiple data-streams to be stored and loaded simultaneously. The conceptual idea is to stream data along a streaming path from an external memory through some processing elements and back to a (possibly different) memory. The image processing task has to be split into various runs through different streaming paths. The advantages are twofold. First,

once a single data stream has been set up, there is no need for further interaction. Second, a number of simultaneous active streaming paths may be implemented. In fact, this number is only limited by hardware resources contained in the FPGA.

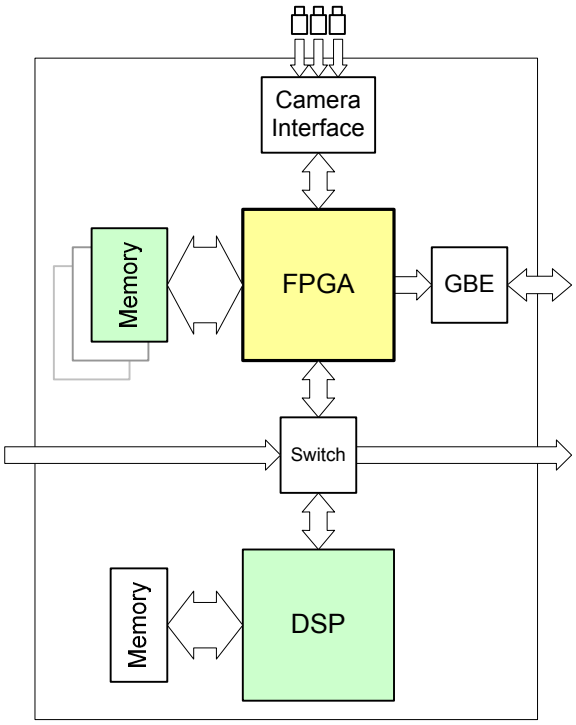

Figure 2: Processing module.

The possibility of algorithmic parallelism enables pipelined operation. If, for example, a sequence of three streaming paths, A to C, has to be completed for a given algorithm, the streaming operation of path A may be instantly restarted processing data of the second item to inspect, while path B is started for processing data of the first item (path C is not used at this time). In the next step, path C may be started for the first item, path B for the second item, and path A for the third item. All three paths are busy at this time, processing data from different items. In addition to pipelining, data parallelism may be achieved by multiple streaming paths of the same kind, i.e., the same operation is applied to different data streams simultaneously. In this context, the term processing unit is defined as any combination of a sequence of processing elements. Processing elements can be arbitrarily interconnected by a generalized streaming data interface. Processing units are either linked to memory ports of any multi-port memory interface in the system, or to input interfaces or output interfaces, respectively.

Figure 3 shows the main units implemented in the FPGA. All external interfaces (camera interface, DSP interface, and GBE interface) are based on the bus concept already mentioned in Section 2. The camera interface connects to the camera node, which is, in turn, connected to the DSP node. In addition, image data from the camera data path are fed into the actual image processing module through an interface which converts them from camera data format to the data format of the streaming data interface. The DSP interface is connected to the DSP node, which is linked to both multi-port memories included in the system. The GBE interface is only connected to multi-port memory A.

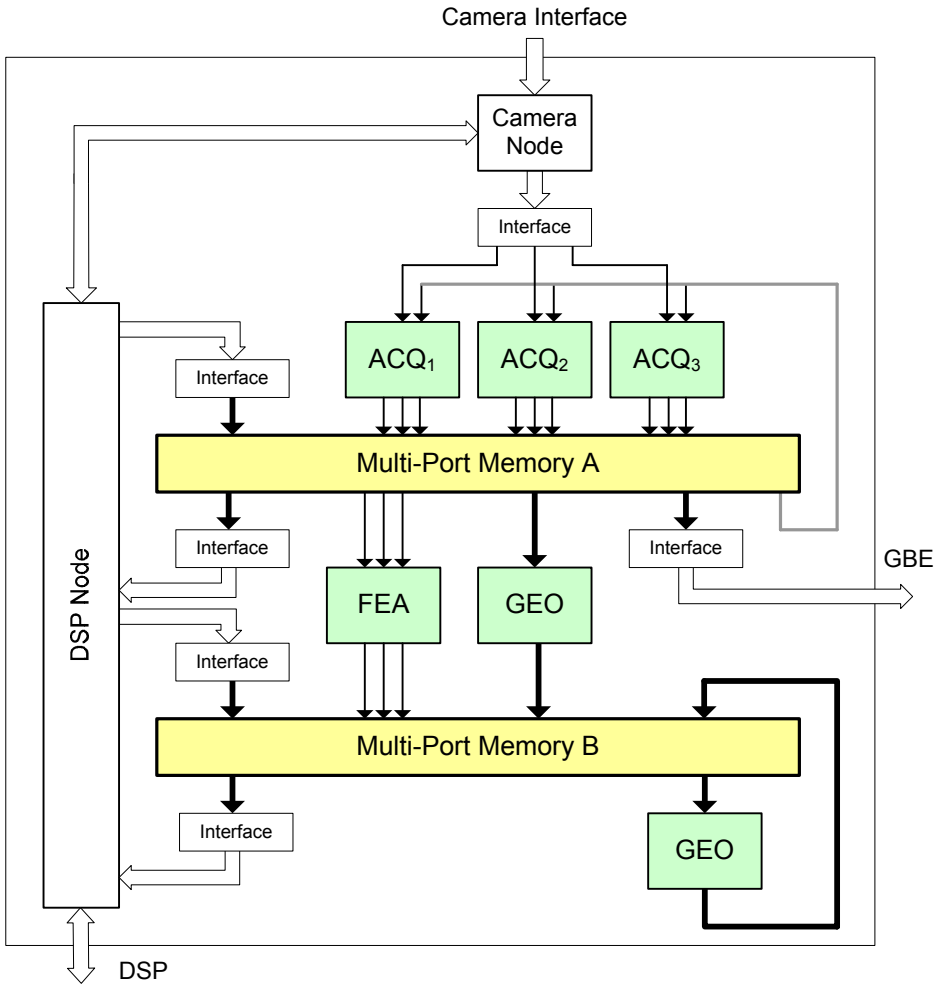

Figure 3: Processing units implemented in the FPGA.

The data fed into the processing module is split into three paths each going through identical acquisition units (ACQ) which are linked to the multi-port memory A. The geometry unit (GEO) and the feature unit (FEA) reside between the multi-port memories. A second geometry unit is linked to multi-port memory B only. The acquisition unit implements the generation of image pyramids[4]. Image pyramids result from consecutive application of a Gaussian filter followed by a reduction of resolution which leads to the next pyramid level (denoted as G_0 for the highest level, G_1, …, G_n). There are 5x5 Gaussian kernels in use reducing width and height of the input image by a factor of ½. Each acquisition unit has three interfaces which are linked to memory A, referring to three pyramid levels which can be stored in parallel. The feedback path form memory A enables generation of pyramids of arbitrary height. In addition, the ACQ unit may contain processing elements for flat field correction or correction of lens distortion. The geometry unit can be used to calculate affine transformations and interpolation, as well as, to compute statistics based on image data which are required for operations like point correlation[5] and projections. The feature unit is capable of combining the data of different paths. For the combination operation, a programmable arithmetic and logic unit has been implemented, e.g., weighted sum. The paths through the FEA unit can be configured to pass several neighborhood operations, e.g., Gaussian, differences of Gaussian, and Sobel. Moreover, images can be shrunk or expanded.

4. IMPLEMENTATION OF THE MULTI-PORT MEMORY

In the area of image processing, there is usually a demand in huge amounts of volatile memory (RAM) for temporary storage of image data. Today's RAM chips come in two basic types: SRAM (Static Random Access Memory) and DRAM (Dynamic Random Access Memory)[6]. Large SRAM memories in the range above decades of Megabytes are much more expensive than DRAM memories at equal size. On the other hand, accessing DRAM is much more complicated than accessing SRAM due to the internal physical structure of DRAM memories. In contrast to SRAM, internal DRAM address registers must be initialized prior to read or write accesses. Address registers must be re-initialized on changes of the DRAM row address. Moreover, the content of the DRAM memory cells must be refreshed periodically[7]. For the HDIP design SRAM and DRAM memories have been implemented.

The Altera Stratix EP1S60 FPGA contains several types of distributed internal SRAM-based memory blocks[8], see Figure 4. The so called M4K-SRAM blocks are perfectly tailored to image processing operations like a 3x3 Gaussian filter which operates on a few adjacent rows of an image in parallel. The larger M-SRAM blocks are mainly used as program and data memory for a soft-core CPU (Altera Nios™) implemented on the FPGA. The smallest type of SRAM blocks (M512-SRAM) are used as, e.g., synchronization FIFOs for the various FPGA interface units.

Type of RAM	M512 SRAM	M4K SRAM	M SRAM	DDR SDRAM
Size	64 Byte	512 Byte	64 kByte	1 GByte
# of blocks available	574	292	6	2
Location	FPGA	FPGA	FPGA	external

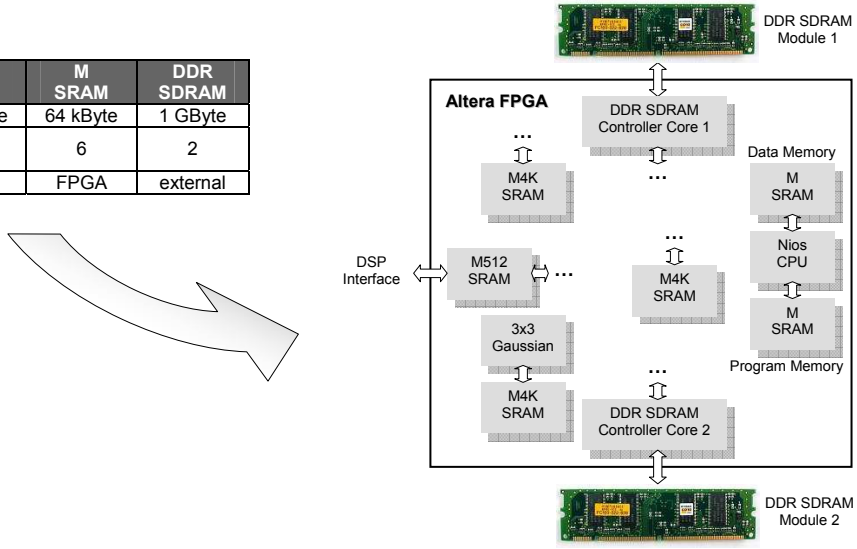

Figure 4: Types of RAM used in the HDIP system.

Complete images, which do not fit into the internal SRAM memories contained in the FPGA, have to be stored temporarily. Hence, two large external DDR-SDRAM (Double Data Rate Synchronous Dynamic RAM) modules were added to the HDIP system. Each external DDR-SDRAM module interfaces to the FPGA via an Altera DDR-SDRAM controller core[9] which resides in the FPGA. The DDR SDRAM controller core handles the complex aspects of using DRAM. It initializes the memory devices, manages SDRAM banks and keeps the device refreshed at appropriate intervals. The core translates read and write requests from the local (FPGA-internal) interface into all necessary SDRAM command signals.

Since several FPGA sub-units need access to the DDR-SDRAM memories, some additional logic was added around the DDR-SDRAM controller core and the external DDR-SDRAM module. From a functional point of view, the additional logic, the DDR-SDRAM controller core, and the external DDR-SDRAM module comprise a 1 GB multi-port memory. The multi-port memory consists of a number of write ports to transfer data from FPGA processing units to the DDR-SDRAM memory, and some read ports in order to read data back from the memory (Figure 5).

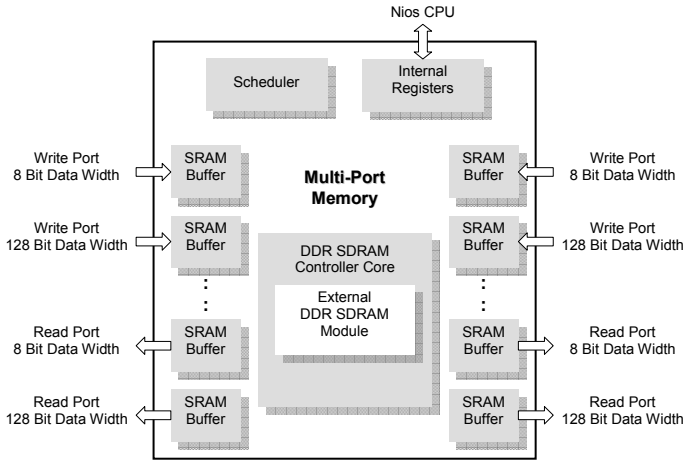

Figure 5: Multi-port memory architecture.

Data can be transferred from all read and write ports in parallel. For this purpose, a scheduler controls access from read and write ports to the (single-port) DDR-SDRAM memory via the DDR-SDRAM controller core. Moreover, each read and write port implements a small SRAM-based buffer. If, for example, a write port asserts a request for data to be transferred to the memory but a data transfer from a read port is already in progress at this time, the data transfer from the write port to the DDR-SDRAM memory will be delayed by the scheduler. Data from the write port will be stored to the local SRAM buffer of this port temporarily. Later, when the data transfer from the read port is finished the scheduler grants access to the write port which then transfers data stored in the local SRAM buffer to the DDR-SDRAM memory. That way, data transfer requests from all read and write ports can be performed concurrently almost without any delays.

Since most of the FPGA sub-modules store or read data always to or from the same DDR-SDRAM memory location, DDR SDRAM address generation is performed by an address logic which is also part of the additional multi-port memory logic that encapsulates the DDR-SDRAM controller core and the external DDR-SDRAM memory. Some address registers of the multi-port memory, which can be accessed over the Nios CPU, define the size of the data-block and its location in DDR-SDRAM memory. Moreover, some registers provide status information for the Nios. If, for example, a complete block was received via a write port, an interrupt can be released in order to notify the Nios CPU about this event.

The additional logic around the DDR SDRAM controller core and the DDR-SDRAM memory is written in VHDL. Therefore, many parameters are fully configurable at design time via VHDL constants, like:

- Number of read ports of the multi-port memory.
- Number of write ports of the multi-port memory.
- Data width of each read and write port, which can be either 8-bit or 128-bit. Data widths of 8-bit are used to interface FPGA-internal 8-bit pixel oriented processing sub-units to the DDR-SDRAM memory. 128-bit ports are used to connect dedicated FPGA sub-units which require high-speed data transfers from/to DDR-SDRAM memory.
- Priority of the port. Each read and write port of the multi-port memory is assigned to a unique priority level. Concurrent requests by either read or write ports will be resolved by the scheduler according to the priority levels of the ports in question.
- Size of the local SRAM buffer for each read and write port which will be instantiated in the FPGA. If the multi-port memory consists of, e.g., many read or write ports, larger SRAM buffer sizes must be configured since the delays (port request → start of the actual data transfer from SRAM buffer to DDR-SDRAM memory and vice versa) will increase. However, larger SRAM buffer sizes reduce the total amount of available FPGA SRAM which is heavily used for implementing many of the FPGA-internal processing operations.

- Amount of data for a single uninterruptible data transfer of a read or write port. Internal address registers of the DDR SDRAM module must be re-initialized by the DDR-SDRAM controller core on changes of the SDRAM bank or row address, which causes a performance penalty. Hence, the transfer length for a single uninterruptible data transfer of a read or write port should be configured as large as possible for system performance reasons. However, a larger transfer length increases the latency if a high-priority port asserts a request for a data transfer but a transfer from another port is already in progress.
- Port request SRAM buffer trigger level for each read and write port of the multi-port memory. A write port asserts a request to the scheduler for a data transfer from its local SRAM buffer to DDR-SDRAM memory, if the amount of data currently stored in the SRAM buffer exceeds the port request SRAM buffer trigger level. Similarly, a read port asserts a request to the scheduler for a data transfer from DDR-SDRAM memory to its local SRAM buffer, if the amount of data currently stored in the SRAM buffer falls under the port request SRAM buffer trigger level.

5. RESULTS

The HDIP design presented in Figure 3 has been implemented on an Altera Stratix 1S60 FPGA device. The required logic elements (LE) for the individual processing units are summarized in Table 1. The design consumes approximately 70 % of the logic elements available on the chosen device. As well, a total amount of 5 MBits internal memory resources (60 %) and 70 of the 9-bit DSP block elements (50 %) are used. The system clock frequency for the image processing module is 133 MHz. The Nios module is clocked at 100 MHz. Both external DDR-SDRAM modules are running at 133 MHz, which provides a raw memory bandwidth of about 2 GB of data per second for read and write transfers. Data transfers from all read and write ports are interlaced by the control logic of the multi-port memory. Hence, almost the maximum DDR-SDRAM memory bandwidth (minus a few percentage of performance due to DDR-SDRAM address re-initialization on DDR-SDRAM bank or row address changes and refresh cycles) is available for the read and write ports of the multi-port memory.

Processing unit	Les	Instances	Total LEs
Acquisition unit	1500	3	4500
Feature unit	5000	1	5000
Geometry unit	5000	1	5000
DDR core	1000	2	2000
Single port of the multi-port memory	500	27	13500
Nios subsystem	5000	1	5000
Interfaces, networking, …			5000
Total			**40000**

Table 1: Summary of FPGA logic elements usage.

The configuration of a PM with three acquisition units is able to cope with input data rates up to 400 MB/s. The feature unit may be configured for several processing modes resulting in different measures for total throughput. The application specific average data rate is approximately 300 MB/s, however, the peak rate is 400 MB/s. The geometry unit processes data at over 100 MB/s. The FPGA-to-DSP bandwidth is over 200 MB/s. Table 2 summarizes processing times for some HDIP units compared to their functional counterparts implemented on a DSP (C64x, 1 GHz).

Processing unit	DSP [ns/Pixel]	HDIP [ns/Pixel]
Acquisition unit	0.8	2.5
Feature unit	9.0	3.0
Geometry unit	4.0	10.0

Table 2: Normalized processing times.

The DSP outperforms the FPGA implementation in most situations, except for the complex processing sequence implemented in the feature unit. On a sub-function basis, the advantage for the DSP is even greater. For example, the calculation of a single pyramid level takes 0.5 ns per pixel on the DSP, compared to 7.5 ns for the FPGA. However, due to parallelism utilized on the FPGA, the overall performance is superior to a single DSP solution because

- multiple processing units may be pipelined and therefore they work in parallel, in contrast to the sequential execution order on the DSP,
- the FPGA provides more scalability, e.g., on the FPGA an additional processing unit may be included, while the DSP has a fixed architecture,
- the DSP cannot handle the enormous input data rate, and, additionally, heavy data transfers degrade DSP performance even more, and
- combination of several sub-functions linked to a streaming path leads to substantially higher performance, as evident from results for the feature unit.

6. CONCLUSIONS

The proposed hardware driven image processing architecture takes advantage of contemporary FPGA structures. Despite the fact that a DSP is much faster for most single aspects of a complex algorithm, the proposed architecture is superior thanks to the advantage of algorithmic parallelism and data parallelism enabled by the FPGA. The architecture offers flexibility to adapt the actual processing flow to specific application demands by implementing appropriate processing units. A future enhancement may simplify the construction of processing modules by simply choosing appropriate processing elements from a library and linking them together according to the actual image processing algorithm. This ensures design reuse and short development times. Due to image processing on the FPGA, there is no need for an image processing system based on parallel DSP architectures at the processing module level. The parallelism of multiple DSPs is introduced at the processing system level, where the scalable arrangement of multiple processing modules in a ring topology has proven to be suitable for demanding image processing applications. On the other hand, caused by system complexity the implementation of the processing elements was accompanied by high effort for design verification. In addition, some cuts to the original universality of the approach were made to evade FPGA constraints limiting high system frequencies. In the future, a complete processing module may be implemented within a single FPGA, which enables further integration of a processing module into the housing of a camera. Consequently, a prospective image processing system consists only of interconnected camera modules. However, this goal can only be achieved if the performance of CPU cores available on FPGAs is substantially improved.

REFERENCES

1. Fürtler, J., Krattenthaler, W., Mayer, K., Penz, H., Vrabl, A., *SIS-Stamp: An integrated inspection system for sheet prints in stamp printing application*, Computers in Industry, Vol. 56, No. 8-9, pp. 958-974, 2005, ISSN 0166-3615
2. Fürtler, J., Mayer, K., Krattenthaler, W., Bajla I., SPOT - Development Tool for Software Pipeline Optimization for VLIW-DSPs used in Real-time Image Processing, Real-Time Imaging Vol. 9/6, p. 387-399, 12/2003, ISSN 1077-2014
3. Davies, E. R., *Machine Vision*, Morgan Kaufmann Publishers, San Francisco, 2005, ISBN 0-12-206093-8
4. Jähne, B., *Digital Image Processing*, Springer-Verlag New York Berlin Heidelberg, ISBN 0-387-53782-1
5. Penz, H., Bajla, I., Mayer K., Krattenthaler, High-Speed Template Matching with Point Correlation in Image Pyramids, In *Diagnostic Imaging Technologies and Industrial Applications*, SPIE Proceedings Volume 3827, Conference on Diagnostic Imaging Technologies and Industrial Applications, Munich, 14-15 June 1999, pp. 85-94, 1999, ISSN 0277-786X, ISBN 0-8194-3313-6
6. Turley, J., *The Essential Guide to Semiconductors*, Prentice Hall, 2003
7. *128MB, 256MB, 512MB (x64, SR) PC3200 200-PIN DDR SODIMM*, Datasheet, Micron Technology, 2004
8. *Stratix Device Handbook*, Altera Corporation, USA, 2003
9. *DDR SDRAM Controller MegaCore Function User Guide*, Document Version 1.2.0 rev 1, Altera Corporation, USA, March 2003