Hindawi Publishing Corporation
EURASIP Journal on Embedded Systems
Volume 2007, Article ID 71794, 10 pages
doi:10.1155/2007/71794

Research Article

Design Considerations for Scalable High-Performance Vision Systems Embedded in Industrial Print Inspection Machines

Johannes Fürtler,[1] Peter Rössler,[2] Jörg Brodersen,[1] Herbert Nachtnebel,[3] Konrad J. Mayer,[1] Gerhard Cadek,[4] and Christian Eckel[4]

[1] *Business Unit of High Performance Image Processing, Austrian Research Centers Gmbh (ARC), 2444 Seibersdorf, Austria*
[2] *Department of Embedded Systems, University of Applied Sciences, Höchstädtplatz 5, 1200 Vienna, Austria*
[3] *Institute of Computer Technology, Vienna University of Technology, Gußhausstraße 27-29/E384, 1040 Vienna, Austria*
[4] *Oregano Systems – Design and Consulting GesmbH, Phorusgasse 8, 1040 Vienna, Austria*

Received 1 May 2006; Revised 21 September 2006; Accepted 9 October 2006

Recommended by Udo Kebschull

This paper describes the design of a scalable high-performance vision system which is used in the application area of optical print inspection. The system is able to process hundreds of megabytes of image data per second coming from several high-speed/high-resolution cameras. Due to performance requirements, some functionality has been implemented on dedicated hardware based on a field programmable gate array (FPGA), which is coupled to a high-end digital signal processor (DSP). The paper discusses design considerations like partitioning of image processing algorithms between hardware and software. The main chapters focus on functionality implemented on the FPGA, including low-level image processing algorithms (flat-field correction, image pyramid generation, neighborhood operations) and advanced processing units (programmable arithmetic unit, geometry unit). Verification issues for the complex system are also addressed. The paper concludes with a summary of the FPGA resource usage and some performance results.

1. INTRODUCTION

Industrial printing houses, especially companies producing prints which include techniques against counterfeiting (for, e.g., banknote or postal stamps), strive to emit flawless products. 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, for example, 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 machine for quality inspection of printed sheets [1]. The mechanical part consists of a loading station (A), a separator (B), several conveyor belts (C), a switch for sorting (D), as well as trays for sheets which have passed the inspection system (E) and sheets which have been rejected (F). 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 (several meters per second), each camera station is made up of several high-speed line-scan cameras, operating at line rates above 50 kHz and resolutions of at least 1024 pixels, which is necessary to identify the fine details of the print. The cameras differ in spectral sensitivity and 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 control unit (I), a processing system (J), and a machine service server (K) with some clients for user interaction attached to it. The machine control unit serves as an interface to sensors and actuators of the machine, for example, camera triggers, and keeps track of each sheet in the system. During operation of the machine, the server continuously 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.

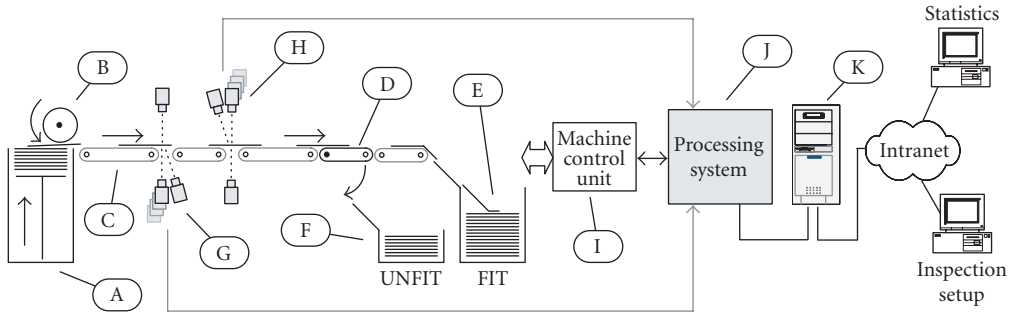

Figure 1: Print inspection system example.

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 which must be handled by the real-time processing system:

(i) tens of sheets simultaneously processed by the machine at different stages,
(ii) feeding rates up to 50 sheets per second,
(iii) more than a gigabyte of input data per second,
(iv) computation of complex image processing tasks, including neighborhood operations, generation of image pyramids, affine transformations, point correlations, and projections.

A vision system for this task has been developed by the ARC Seibersdorf Research GmbH (ARCsr). The system design was significantly influenced by a new generation of high-end field programmable gate arrays (FPGA), which enable implementation of complex system on programmable chip solutions. For this reason, the ARCsr was supported by the Institute of Computer Technology at the Vienna University of Technology and by Oregano Systems - Design and Consulting GmbH who contributed their long-term experience in the design of complex electronic systems and their expert knowledge in VLSI (very large scale integration) circuits design. This paper deals with design considerations for the image processing system and mainly focuses on system parts which have been implemented on FPGAs.

2. SYSTEM DESIGN CONSIDERATIONS

The problem of embedding vision in real-time processing systems has been solved for many times. Typically, these solutions are tailored to the specific application needs. Probably, there are hundreds of architectures which have been considered for this purpose, all having some degree of parallelism [2]. The considered application requires rather complex image processing algorithms to implement a wide range of inspection capabilities. The inspected features include, among others, the detection of pale smears, dirt, fine soiling by splashes of ink, and misalignment of printing phases.

Moreover, system design is a rather complex task, because a lot of optimization parameters (accuracy, robustness, reliability, speed, etc.) and interdependencies between many of these parameters have to be considered and optimized. Additionally, an important constraint for economically relevant solutions is the cost of the system components. Therefore, the algorithms have to be selected with respect to the required constraints in the multidimensional parameter space.

A dedicated image processing system based on DSPs (digital signal processors) would require very complex data sharing mechanisms among many DSPs, because a single DSP cannot manage the enormous data volume in real time [3]. Common parallel architectures based on DSPs and/or dedicated hardware components are often either limited to a special application or they are implemented in a general way, which means a large overhead on functionality. Therefore, the system cannot be implemented economically. On the other hand, FPGA-based systems promise to enable suitable solutions for the particular application [4]. However, from the author's viewpoint, many attempts did not optimally utilize the FPGA potentials due to generality of the approach, or the solutions are too specialized that they, again, could only be used for a single application.

The analysis of the requirements led to the conclusion that it was not possible to build an image processing system based on off-the-shelf components. Consequently, a new architecture, which can be fine-tuned for different applications, had to be developed [5, 6]. The key issue for the design of high-performance real-time image processing systems is to match algorithms and architecture [2]. Consequently, it is essential to use common hardware/software codesign methodologies to find a balance between algorithms implemented in hardware and algorithms running as software tasks. This principle is not new, however, because of today's high-end FPGAs, featuring thousands of logic elements, reasonable on-chip memory, and a lot more on-chip resources which speed up signal processing tasks, former paradigms for the design of embedded vision systems have been changed.

For image processing FPGAs offer several essential advantages as follows.

(i) Dedicated hardware resources on the FPGA, for example, wide multiplication units, support high-speed execution for common operations.

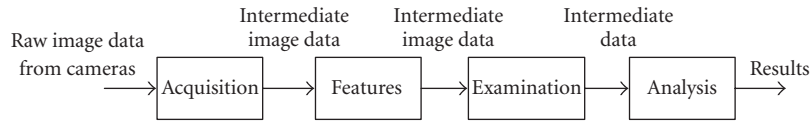

Figure 2: Typical processing sequence, which is well adapted to being implemented as a pipelined image processing system.

(ii) Numerous logic elements available on high-end devices enable multiple instances of complex processing units to be implemented on the same chip.
(iii) Due to parallel hardware structures, FPGAs can handle enormous data transfer rates.
(iv) The possibility of FPGA reconfiguration, even at runtime, is the basis for systems which can be adapted to different needs. Consequently, one hardware platform can be used for several, basically different, applications.

The disadvantages include the following statements.

(i) Compared to high-end DSPs in mass production, high-end FPGAs are a lot more expensive.
(ii) The design flow is typically more time-consuming.
(iii) Poor processing power for sequential (one-dimensional) computations. Due to the general architecture of FPGAs, they are considerably slower for such tasks than dedicated and optimized processor cores (as long as single execution threads are considered). On-chip CPU hardcores and softcores cannot compete with dedicated DSPs. High-end DSPs, like the TMS320C6400 (C64x) 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 [7] at low cost. In addition, the DSP has advantages concerning large portions of fast SRAM-based memory which is available on-chip.

The basic approach presented herein makes use of the benefits of both FPGAs and DSPs while reducing the deficiencies. However, there are several difficulties for the partitioning of tasks between the FPGA and the DSP, which must be overcome.

Some important design questions are the following.

(i) Which unit does control the processing flow?
(ii) How could one balance processing load?
(iii) Where could one partition processing tasks between execution on dedicated FPGA units and software processes running on the DSP?
(iv) What kind of coupling between DSP and FPGA is necessary?

The goal for the proposed hardware driven image processing (HDIP) architecture was a flexible and economically reasonable solution for these problems. Enabled by contemporary FPGA devices, the original contribution of the HDIP approach is the practical application of design principles for high-speed real-time image processing systems like (i) parallel processing, (ii) pipelining, and (iii) multiport memory concepts (see [8]) to build flexible inspection systems based on simple building blocks implemented on FPGAs. Resulting systems should be scalable in terms of the number of attached cameras (20 or more) and scalable to arbitrary processing power. Thereby, a wide range of applications can be covered.

2.1. Parallel processing

Parallelization is the most promising keyword for boosting processing performance in context of image processing. There are two main approaches to parallel processing [2]: (i) data is split up into multiple streams, which are processed by several processing units, (ii) the computational task (functionality) is split up to be processed by several units in parallel. The first approach is referred to as *data parallelism*, which can be utilized for many image processing tasks. Data parallelism is heavily used in the HDIP FPGA design (see Section 4). For example, as shown in Figure 6, camera data fed into the HDIP module is split up into three paths, each going through three identical acquisition units (ACQ). The second approach is also known as *algorithmic parallelism*. Algorithmic parallelism can be successfully exploited in the form of pipelined processing systems. As described later in Section 4, the concept of algorithmic parallelism (pipelining) is applied to the HDIP design as well.

In complex image processing systems, several levels of parallel processing have to be considered. For example, fine grain parallelism can be exploited by multiple processing units on a DSP, while coarse grain parallelism involves multiple modules at a higher level of processing (e.g., two identical systems, one for each side of the sheet, considering the print inspection system described in Section 1).

2.2. Pipeline processing

For the particular inspection application, a number of images must be processed for every single sheet. The image data is fed into the processing system, where it passes several processing stages as depicted in Figure 2. This sequential processing can be seen as a pipeline where each stage is related to specific (image) processing tasks. The acquisition stage implements several preprocessing steps, for example, flat field correction, camera calibration, and some other low-level image processing algorithms. Low-level image processing (neighborhood operations like, e.g., Gaussian,

differences of Gaussian or Sobel) is continued in the feature stage. In the examination stage, several high-level image processing algorithms are carried out, including computation of image statistics over arbitrary shaped image regions based on affine backward transformations. The final analysis leads to the quality decision for the processed sheet. The implementation of the pipeline stages as suggested in Figure 2 may involve dedicated units on the FPGA, and/or processing on the DSP. Typically, it is a combination of both.

Pipelining is a very effective strategy to speed up processing. However, the speed of the pipeline is determined by the slowest stage. Therefore, the tasks should be partitioned for evenly distributed processing time. In addition, the pipelined system must be designed in accordance with worst-case timing scenarios. To decouple the tasks, buffer memories can be introduced between the stages. As a matter of fact, pipelining introduces latency of results which is related to the number of the stages.

In this context, several kinds of pipelining have to be distinguished.

(i) Cycle pipelining

That is, pipelining based on the cycle time of the production process which means pipelining as described above. For the aimed application, a minimum cycle time T_c is defined, that is, the feeding rate for the sheets is limited. Consequently, the acquisition with line-scan cameras takes most of the cycle time (minus a small blanking time between successive sheets). Therefore, the maximum time for a pipeline stage is related to the process cycle time.

(ii) Processing pipelining on the FPGA

The same concept can be applied for processing at the pixel level, that is, replacement of the complex pipeline tasks from Figure 2 by simple image processing stages. For example, a pipeline containing a stage for applying a pixel offset and scaling, followed by two stages implementing different neighborhood operations (e.g., Gaussian filter, Sobel filter), and finally a binarization stage. This pipeline can be fed with a stream of pixel data producing an output pixel at every clock cycle. This results in an average processing rate for the sequence of all stages of one clock cycle per pixel. As images typically consist of many pixels, overhead for loading and unloading of the pipeline can be neglected. Obviously, this concept is not limited to data representing pixel values. For this reason we, call such pipelines *feature pipelines*, or *streaming path*.

(iii) Software pipelining

This is a very effective way to speed up loops by exploiting algorithmic parallelism through special utilization of multiple execution units available on the DSP [7].

2.3. Multiport memory concept

In the area of image processing, there is usually a demand for huge amounts of volatile memory (RAM) for storage of image data. Today's RAM chips come in two basic types: SRAM (static random access memory) and DRAM (dynamic random access memory) [9]. 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 reinitialized on changes of the DRAM row address. Moreover, the content of the DRAM memory cells must be refreshed periodically [10].

Complete images, which do not fit into the FPGA internal SRAM memories, have to be stored temporarily, for example, to implement buffer memories between pipeline stages. For this reason, large external DDR-SDRAM (double data rate synchronous dynamic RAM) modules are a reasonable choice. A DDR-SDRAM controller core implemented on the FPGA handles the complex aspects of using the 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.

Common SDRAM controllers usually support an interface which can be accessed by only one unit (e.g., a CPU). However, since several processing units on the FPGA need access to the DDR-SDRAM memories, a multiport memory concept offers a lot of advantages. For example, a multiport memory interface consists of a number of write ports to transfer data from FPGA processing units to the DDR-SDRAM memory (via the SDRAM controller and some kind of arbitration logic), and a number of read ports in order to read data back from the memory. Hence, the multiport memory interface allows multiple data streams to be stored and loaded simultaneously from/to SDRAM memory and can be interpreted as an array of direct memory access (DMA) controllers.

The conceptual idea of an image processing system which implements a multiport memory concept is to stream data from an external memory through a feature pipeline and back to a (possibly different) memory location. The image processing task has to be split up into various runs through different feature pipelines. The advantages are twofold. First, once a single data stream has been set up, there is no need for further interaction. This works similar to a CPU which can continue code execution while the DMA controller transfers data independently from code execution. Second, a number of simultaneous active streaming paths may be implemented. In fact, this number is only limited by hardware resources available on the FPGA.

3. SYSTEM OVERVIEW

With the particular application in mind, the first decision concerned one of the most important characteristics for the processing system—to be scalable to an (almost) arbitrary number of cameras (refer to Figure 1). Starting from this perspective, the typical processing system consists of several

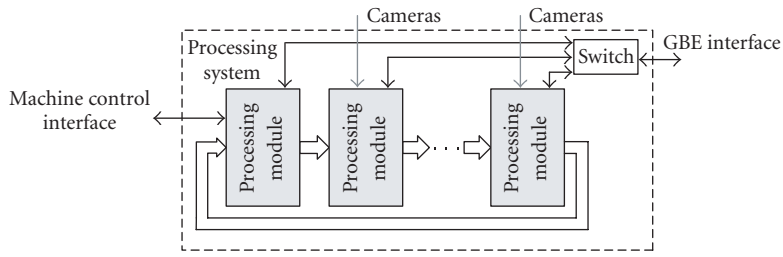

Figure 3: System overview.

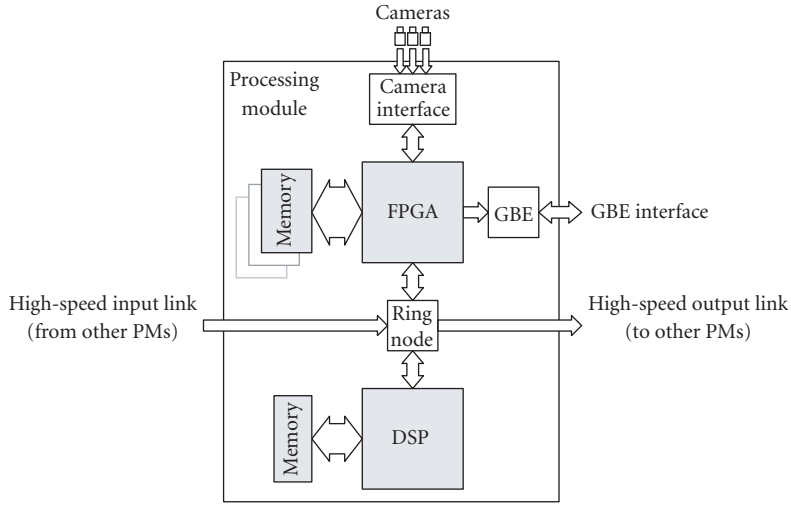

Figure 4: Processing module.

processing modules (PM), which are interconnected in a ring topology as shown in Figure 3. This arrangement is backed up by successful applications of the ring topology for multisensor image processing systems [11, 12]. The ring topology allows a simple extension of the system, where 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 image processing features which need additional processing power, for example, character recognition, may require an additional PM. The system has been designed to match worst case scenarios, therefore, no dynamic load balancing is implemented at the moment. However, static load balancing can be fine tuned by choosing a number of PMs with appropriate capabilities as will be outlined below. Typically, one PM implements the physical interface to the machine control unit and, therefore, it controls the whole processing flow. In this context, this particular PM serves as a master to the other modules. (Direct) communication between the machine service server and the PMs is established via a Gigabit Ethernet (GBE) interface.

Figure 4 shows the main modules implemented on a single PM. According to the HDIP approach, a PM basically consists of an FPGA module and a DSP module. However, a PM can be equipped either with a standalone FPGA, or with a standalone DSP, respectively. Additionally, the printed circuit board can be equipped with different devices, for example, varying speed grade or complexity for the FPGA, and different clock speeds for the DSP. This introduces a flexible way for selecting the appropriate processing power needed for the specific task. For example, the master PM contains the DSP part only and, instead of the FPGA, it is equipped with additional interfacing capabilities for communication to the machine control unit. However, usually a PM is equipped as shown in Figure 4. Both processing units, the FPGA and the DSP, are interconnected to a switching fabric. The high-speed link provides bidirectional data transmission between the FPGA and the DSP. The input link and the output link used to build the ring topology are also connected to a switching fabric (ring node). Consequently, data transmission between any PM in the overall system is possible. For specific application reasons, it is necessary to store a number of raw images acquired by the cameras for later usage. Hence, the machine service server may download data (e.g., the raw images) at any time via the high-speed link. This download must not interfere with real-time behavior of the system.

For the particular application, the DSP serves as a master to the FPGA and controls the processing flow. However, this is not a general rule, rather it is the result from the hardware/software codesign process for the specific application. Other strategies may be implemented as well without any

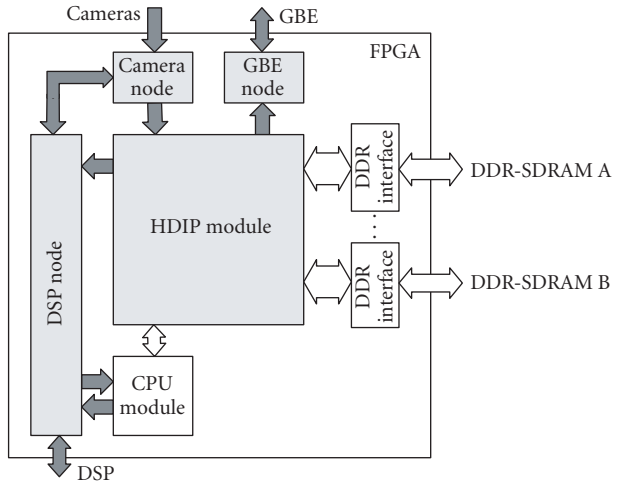

Figure 5: Modules implemented on the FPGA.

changes of the hardware, which is an important advantage of the approach. Here, the DSP is also heavily involved in computing complex high-level image analysis algorithms. Therefore, the analysis stage (Figure 2) is implemented as software task on the DSP. Low order image processing and intermediate order image processing is done by the FPGA (including acquisition stage, feature stage, and examination stage). In order to minimize the amount of data to be transferred between the DSP and the FPGA, advanced data reduction based on image analysis takes place on the FPGA.

4. IMPLEMENTATION DETAILS

The FPGA design (also referenced as the HDIP FPGA design) has been implemented using VHDL (very high-speed integrated circuits hardware description language) and is therefore independent from the target technology, for example, FPGA or application specific integrated circuit (ASIC). However, some technology-dependent resources available on the chosen Altera Stratix™ device have been used, for example, memory blocks and DSP blocks [13]. The same applies for intellectual property (IP) cores supplied by Altera (Nios™ softcore CPU, DDR-SDRAM controller). These modules have to be adapted according to the underlying technology.

Figure 5 shows the main units implemented on the FPGA. All external interfaces (camera interface, DSP interface, and GBE interface) are based on the link concept mentioned in Section 3. The camera interface is linked to the camera node, which, in turn, is connected to the DSP node and the actual image processing module (HDIP module). The separation of the DSP node and the camera node is due to the high data volume (data from several cameras), which is passed to the HDIP module. Nevertheless, it is possible to redirect image data to other PMs available in the ring. The DSP interface and the on-chip CPU module are connected to the DSP node, whereas the GBE interface has its own node linked to the HDIP module. Two external DDR-SDRAMs are attached to the HDIP module via an Altera DDR-SDRAM IP core [14].

In order to control the image processing flow, the external DSP sends sequences of command scripts to the on-chip CPU. The CPU executes these scripts and sends results back to the DSP. The execution of the scripts involves a lot of interaction between the CPU and the HDIP module. Keeping these interactions locally on the FPGA reduces communication between FPGA and DSP. Moreover, the local processing does not utilize the DSP to handle the fine details of the image processing task. As a result, more time can be spent on the DSP for number crunching tasks, where its VLIW architecture can be exploited.

Figure 6 shows details concerning the HDIP module. The camera data fed into the module is split into three paths, each going through identical acquisition (ACQ) units, which are linked to the multiport memory A. The geometry (GEO) unit and the feature (FEA) unit reside between the two multiport memories. A second geometry unit is linked to multiport memory B only. The ACQ unit implements the generation of image pyramids [15]. 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, $G_1, \ldots, G_n$). There are 5×5 Gaussian kernels in use, as well as a reduction of width and height by a factor of 1/2. Each acquisition unit has three interfaces which are linked to memory A, referring to three pyramid levels which can be generated and stored to the memory in parallel. The feedback path from memory A enables generation of pyramids of arbitrary height. In addition, the ACQ unit can contain processing elements for flat field correction or lens distortion correction. The geometry unit (for a detailed description refer to [16]) can be used to compute image statistics over arbitrary shaped image regions based on affine backward transformations with interpolation, which are required for operations like point correlation [17] and projections. The feature unit is capable of combining the data from different paths. For the combination operation, a programmable arithmetic and logic unit has been implemented. The paths through the FEA unit can be configured to pass several neighborhood operations, for example, Gaussian, differences of Gaussian, and Sobel. Moreover, images can be shrunk or expanded. All units (except the camera, DSP and GBE interface blocks) are connected to the on-chip CPU. These interfaces are not shown in Figure 6 for clarity reasons. The CPU can also access the external DDR-SDRAM memories via a dual-ported memory (DPM) unit. For high speed data transfers from the memories to the external DSP, both multiport memories are connected to the DSP node. Transfers in the opposite direction (DSP to external memory) require interaction of the CPU, which stores data from the DSP into the external memory via the DPM unit.

Data can be transferred from all read and write ports of the multiport memory in parallel. For this purpose, a scheduler controls transfers between the ports and the (single-ported) DDR-SDRAM memory via the SDRAM controller. In addition, the ports implement a small SRAM-based buffer memory. If, for example, a write port asserts a request for

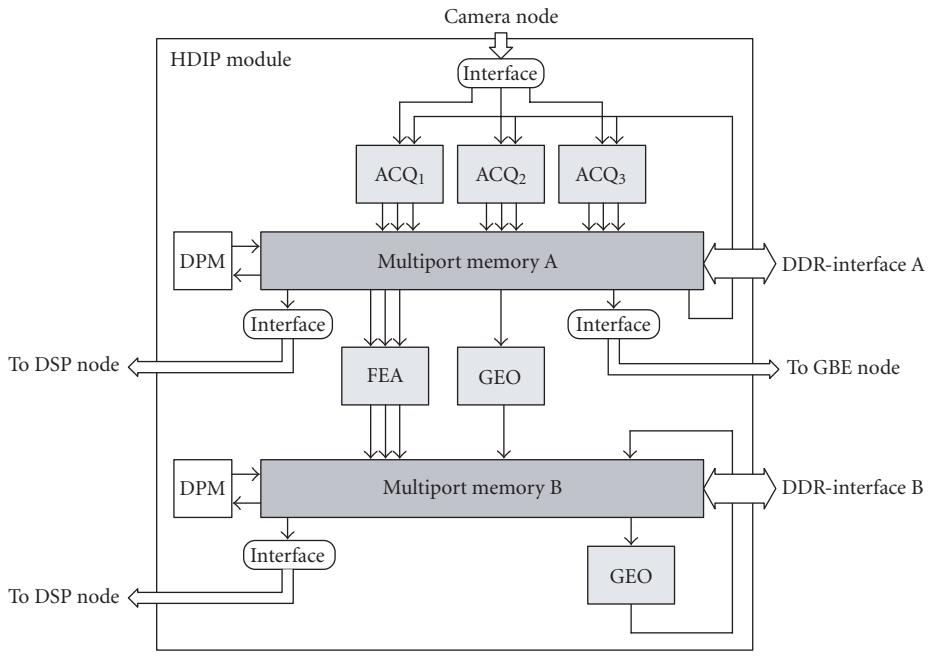

Figure 6: Units of the HDIP module.

a data transfer while another transfer is already in progress, the new transfer is delayed. Data for the write port will then be temporarily stored to the buffer. When the first transfer is finished, the scheduler grants access to the delayed write port, which then transfers its data stored in the buffer to the DDR-SDRAM memory. That way, transfer requests from all read and write ports can be performed concurrently almost without any delays. Like a DMA controller, configurable address generation is part of the multiport memory controller. Transfers are set up by the on-chip CPU, which can also detect their completion.

For the particular application, four processing cycles (as suggested in Figure 2) have been introduced. The first three cycles are executed on the FPGA. Hence, 75 percent of the total processing time is spent for FPGA processing, which indicates the importance of the proposed approach. For the acquisition cycle, the three ACQ units are used concurrently. The feature cycle is executed on the FEA unit, whereas both GEO units are utilized during the examination cycle. Finally, the DSP is busy during the analysis cycle. Figure 7 shows how the processing units are processing data from different sheets which are fed into the machine. After the pipeline is filled, four different sheets are inspected concurrently. However, the sheets are processed in different stages. The latency introduced by the pipeline processing requires the switch (refer to Figure 1) to be located in an appropriate distance from the last camera, which is provided by the mechanical design of the machine.

5. VERIFICATION

The verification process of such a large system as presented herein containing hardware and software blocks and even mechanical parts (Figure 1) was, of course, a challenge for the whole project team. In the case of ASICs, it is common for verification teams to spend 70 percent and more of their time in verification and debugging [18]. For FPGAs, where design errors are not so penalized as a design respin is a matter of hours not months, there is, nevertheless, still an obvious need for efficient debug methodologies which enable design teams to identify and fix errors early in the design process.

Several approaches for the verification of the inspection system were used to cover different levels of system complexity.

(i) Most important was the usage of hardware/software coverification techniques. For verification of the image processing module (refer to Figure 4), a software library for emulation of functional hardware behavior was implemented. Hence, a set of images acquired for a single sheet can be processed (of course at much slower speeds than on the actual hardware) via software on the PC. Output data at the several processing stages from the hardware implementation as well as the corresponding results from the emulation can be compared automatically which is important for regression testing.

(ii) Verification for all subunits of the FPGA design was done by the use of VHDL testbenches. Some simulation models, for example, the external DDR-SDRAMs and the external interfaces (written in VHDL, Verilog, and ANSI C) were used in order to set up top-level simulations for the whole FPGA design.

(iii) FPGA prototyping was important for two main reasons. First, it helps to speed up verification since a VHDL-based simulation for a single 1024×768 pixel image requires minutes of simulation time, while on the prototyping hardware several images are processed

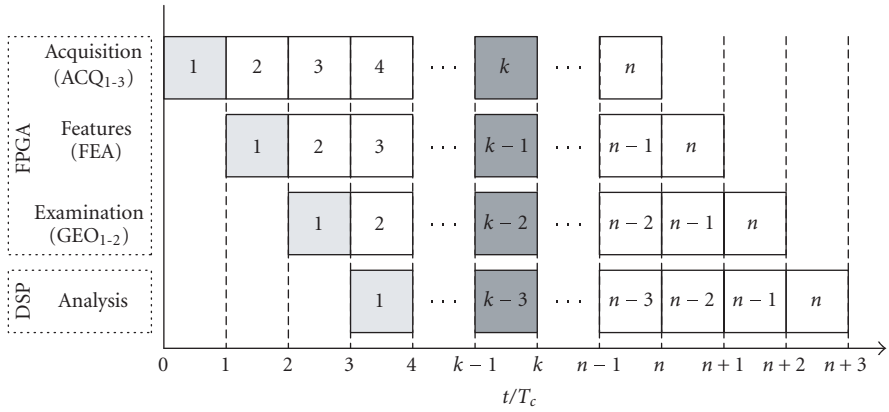

Figure 7: Pipelined operation of FPGA units and DSP software tasks.

Table 1: FPGA resource usage.

Processing unit	Single instance			Instances	All instances		
	Logic resources	SRAM	DSP blocks		Logic resources	SRAM	DSP blocks
ACQ (acquisition) unit	3%	7%	11%	3	9%	21%	33%
FEA (feature) unit	9%	5%	17%	1	9%	5%	17%
GEO (geometry) unit	9%	6%	22%	2	18%	12%	44%
DDR-SDRAM core	2%	—	—	2	4%	—	—
Single port of the multiport memory	1%	1%	—	27	27%	27%	—
Nios subsystem	9%	8%	—	1	9%	8%	—
Interfaces, networking, etc.	27%	18%	—	1	27%	18%	—
Total	—	—	—	—	93%	91%	94%

in fractions of a second. Furthermore, most FPGA processing elements can be verified very easily by observing the processed images in real time on the screen.

(iv) Boundary Scan JTAG (joint test action group)-based testing supported by the EDA backend tool (altera signal tap) was used to observe internal signals of the FPGA design without the need of changing the VHDL code. This was very helpful to detect external timing problems around the DDR-SDRAM memories.

(v) Finally, the FPGA internal CPU turned out to be a valuable resource for setup of complex test cases and to verify the (immediate) results for a large number of verification runs during the verification and design cycle of the system.

6. RESULTS

The HDIP FPGA design shown in Figure 6 has been implemented on an Altera Stratix™ 1S60 FPGA device, which was one of the most complex FPGAs at the time of the design kick-off. The design team spent several man years only on the FPGA design, not including the design of the PCB board, DSP software, and so forth.

The required resources (logic and DSP blocks, as well as SRAM) for the individual processing units as reported by the EDA tools (FPGA synthesis, place, and route) are summarized in Table 1. The design consumes about 93% of the logic resources, 91% of the internal SRAM memories, and about 94% of the FPGAs DSP blocks.

System clock frequency for the image processing modules is 133 MHz, while the Nios CPU module is clocked at 100 MHz. Both external DDR-SDRAM modules are running at 133 MHz (i.e., 64 bits are transferred on both, the rising and the falling clock edge), which provides a raw memory bandwidth of about 2 GBytes/s ($133\,\text{MHz} * 64\,\text{Bit} * 2 = 1.7 * 10^{10}\,\text{Bit/s} \approx 2\,\text{GByte/s}$) per multiport memory module. Data transfers from all read and write ports are interlaced by the control logic of the multiport memory. Hence, almost the maximum memory bandwidth of 2 GByte/s (minus a few percentage of performance due to DDR-SDRAM address reinitialization on DDR-SDRAM bank or row address changes and refresh cycles) is available for the read and write ports. The multiport memory performance was verified during the test and verification phase by running several dedicated performance test programs on the Nios CPU.

Table 2: Typical FPGA and DSP processing times.

Processing unit	DSP [ns/pixel]	HDIP [ns/pixel]
ACQ (acquisition) unit	0.8	7.5
FEA (feature) unit	9.0	3.0
GEO (geometry) unit	4.0	10.0

In typical applications, high-speed line-scan cameras with resolutions of at least 1024 pixels, operating at line rates from 50 kHz to 100 kHz, are used (here, a pixel is represented by an 8 bit intensity value). For support of three cameras, the PM is equipped with three acquisition units (clocked at 133 MHz). Hence, the PM is able to cope with three input data streams of up to 133 MPixels/s resulting in a total input data rate of about 400 MByte/s. The camera data (pyramid level G0) is directly fed into its ACQ unit, where two new pyramid levels (G1, G2) are generated in parallel (refer to Figure 6) and are continuously stored into the external memory A. Consequently, the feature pipeline (see Section 2.2) for generation of pyramid image G1 processes a new input pixel every clock cycle which is equivalent to an average processing time of 7.5 ns/pixel (note that an output pixel is produced only every fourth clock cycle as width and height of the image are reduced for every pyramid level). A corresponding DSP implementation (C641x)—exclusively running this task requires about 0.8 ns/pixel (using software pipelining in highly optimized assembler code as described in [7]). Additional tasks have to share processing units, as well as memory bandwidth.

Performance analysis for other image processing blocks of the HDIP FPGA module is much more complex, because streaming pipelines in the GEO unit and the FEA unit are typically used in multiple configurations, resulting in different measures for total throughput. Thus, Table 2 summarizes typical processing times measured for practical application of the HDIP units compared to their functional counterparts implemented for C641x DSPs (1 GHz). Detailed performance analysis is beyond the scope of this paper and, therefore, is subject of further publications (e.g., [16]).

The DSP outperforms the FPGA implementation in most situations, except for the complex processing sequence implemented in the FEA unit, where many processing steps can be implemented in parallel. On a subfunction basis (i.e., portions of code where the DSP can operate on its internal memory or cache memory), the advantage of the DSP can be even greater. For example, Table 2 reveals that the calculation of a single pyramid level takes 0.8 ns per pixel on the DSP, compared to 7.5 ns for the FPGA.

However, the FPGA implementation has important advantages compared to a single DSP solution due to the following reasons.

(i) The DSP is limited in handling the enormous input data rate of the considered application, while the FPGA benefits from its parallelism. For example, on the C641x, the external memory interface EMIF-B can be used for camera acquisition, because the external memory (SDRAM) has to be connected to the EMIF-A as image processing algorithms typically need high memory bandwidth. Figure 8 shows that the nominal value for EMIF-B transfer rate is about 266 MBytes/s. Practically, an overhead of up to 20% has to be taken into account (e.g., arbitration overhead, communication overhead, etc.), leading to a bandwidth of not much more than 200 MByte/s available for data transfers. Assuming the same memory interface type, speed and technology for both DSP and FPGA, the FPGA enables implementation of more memory interfaces or wider memory interfaces to overcome bandwidth limitations. For the HDIP design, two 64 bit memory ports have been used, compared to the 16 bit EMIF-B port of the DSP. Moreover, calculation of image pyramids is only a part of the image processing algorithm (refer to Figure 2). Hence, for implementation of the HDIP functionality using DSPs only, several DSPs are necessary. Consequently, less than 100 MByte/s can be used for image acquisition, as data (e.g., pyramid images) has to be transferred to other DSPs (also using EMIF-B). Thus the available bandwidth on a single DSP is only approximately a quarter of the 400 MPixel/s of the HDIP approach.

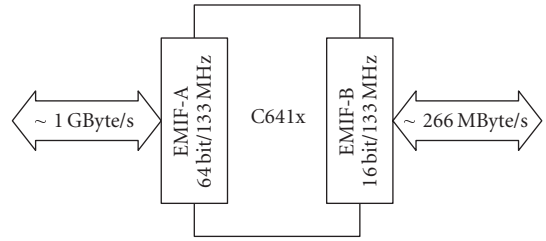

Figure 8: External transfer rates for the C641x DSP.

(ii) Heavy data transfers degrade DSP performance even more because the CPU is interrupted more often (e.g., by the DMA controller) and it is more likely that the CPU has to wait longer for completion of data transfers. This context switching can be implemented more efficient (i.e., resulting in higher throughput) in hardware, as discussed in Section 2.3.

(iii) In contrast to sequential execution order on the DSP, multiple instances of a processing unit can be implemented on the FPGA in parallel. For example, three instances of the ACQ unit result in an average time of 2.5 ns/pixel for the FPGA implementation; two GEO units lead to an average time of 5 ns/pixel. All these functions can be implemented on a single chip! A comparable DSP-based system would require several DSP devices. Not counting the bandwidth limitation, at least three: one for the acquisition, one for the feature calculation, and one for the geometry based tasks. This results in extra hardware costs. For the HDIP approach, data access over shared memories is elegantly implemented as multiport memory interface to external DDR-SDRAMs requiring only a single FPGA.

(iv) The FPGA provides a high degree of scalability while the DSP has a fixed architecture. For a particular application, processing units can be either added to the FPGA or exchanged against processing units which are not used for that specific application.

(v) Combination of several subfunctions increases the amount of exploited parallelism. Linked to a feature pipeline, substantially higher performance can be achieved, as evident from the results of the FEA unit.

7. CONCLUSION

The proposed hardware-driven image processing architecture takes advantage of contemporary high-end FPGA devices. 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 will 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 provides 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. Instead, 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 resulting in slower system frequencies as expected during the specification phase. 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 will be substantially improved in future devices.

ACKNOWLEDGMENT

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

REFERENCES

[1] J. Fürtler, W. Krattenthaler, K. J. Mayer, H. Penz, and A. Vrabl, "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.

[2] E. R. Davies, *Machine Vision*, Morgan Kaufmann, San Francisco, Calif, USA, 2005.

[3] J. Stein, *Digital Signal Processing: A Computer Science Perspective*, John Wiley & Sons, New York, NY, USA, 2000.

[4] Z. Salcic and A. Smailagic, *Digital Systems Design and Prototyping Using Field Programmable Logic and Hardware Description Languages*, Kluwer Academic, Dordrecht, The Netherlands, 2000.

[5] P. Rössler, C. Eckel, H. Nachtnebel, J. Fürtler, and G. Cadek, "FPGA-Design für ein Hochleistungs-bildverarbeitungssystem," in *Proceedings of the Austrochip 2004, The Austrian National Conference on Microelectronics*, pp. 83–88, Villach, Austria, October 2004.

[6] J. Fürtler, J. Brodersen, P. Rössler, et al., "Architecture for hardware driven image inspection based on FPGAs," in *Real-Time Image Processing*, vol. 6063 of *Proceedings of SPIE*, pp. 105–113, San Jose, Calif, USA, January 2006.

[7] J. Fürtler, K. J. Mayer, W. Krattenthaler, and I. Bajla, "SPOT—development tool for software pipeline optimization for VLIW-DSPs used in real-time image processing," *Real-Time Imaging*, vol. 9, no. 6, pp. 387–399, 2003.

[8] S. Siegel, "A unified streaming memory controller and its utility in image processing applications," White Paper Datacube, AIA Machine Vision Online, Ann Arbor, Mich, USA.

[9] J. Turley, *The Essential Guide to Semiconductors*, Prentice-Hall, Upper Saddle River, NJ, USA, 2003.

[10] "128MB, 256MB, 512MB (x64, SR) PC3200 200-PIN DDR SODIMM," Datasheet, Micron Technology, 2004.

[11] J. Brodersen, K. J. Mayer, D. Landl, and I. Bajla, "Novel data acquisition and communication bus architecture for real-time multisensor imaging systems," in *Real-Time Imaging VII*, vol. 5012 of *Proceedings of SPIE*, pp. 122–131, Santa Clara, Calif, USA, January 2003.

[12] J. Brodersen, R. Palkovich, D. Landl, J. Fürtler, and M. Dulovits, "Advanced real-time bus system for concurrent data paths used in high-performance image processing," in *Real-Time Imaging VIII*, vol. 5297 of *Proceedings of SPIE*, pp. 278–286, San Jose, Calif, USA, January 2004.

[13] "Strat ix Device Handbook," S5V1-3.1 and S5V2-3.1, Altera, San Jose, Calif, USA.

[14] *DDR SDRAM Controller MegaCore Function User Guide*, Document Version 1.2.0 rev 1, Altera, San Jose, Calif, USA, March 2003.

[15] B. Jähne, *Digital Image Processing*, Springer, New York, NY, USA, 1991.

[16] J. Fürtler, K. J. Mayer, C. Eckel, J. Brodersen, H. Nachtnebel, and G. Cadek, "Geometry unit for analysis of warped image features on programmable chips," to appear in *EURASIP Journal on Embedded Systems*, special issue on Embedded Vision Systems.

[17] H. Penz, I. Bajla, K. J. Mayer, and W. Krattenthaler, "High-speed template matching with point correlation in image pyramids," in *Diagnostic Imaging Technologies and Industrial Applications*, vol. 3827 of *Proceedings of SPIE*, pp. 85–94, Munich, Germany, June 1999.

[18] J. Bergeron, *Writing Testbenches, Functional Verification of HDL Models*, Kluwer Academic, Dordrecht, The Netherlands, 2nd edition, 2003.