A reconfigurable platform for high-end real-time digital film processing *

Amilcar do Carmo Lucas *and* Sven Heithecker, Rolf Ernst
{lucas | heithecker | ernst}@ida.ing.tu-bs.de

Abstract

Digital film processing is characterized by a resolution of at least 2K (2048x1536 pixels per frame at 30 bit/pixel and 24 pictures/s, data rate of 2.2 GBit/s); higher resolutions of 4K (8.8 GBit/s) and even 8K (35.2 GBit/s) are on their way. Real-time processing at this data rate is beyond the scope of today's standard and DSP processors, and ASICs are not economically viable due to the small market volume. Therefore, an FPGA-based approach was followed in the FlexFilm project. Different applications are supported on a single hardware platform by using different FPGA configurations.

The multi-board, multi-FPGA hardware/software architecture is based on Xilinx Virtex-II Pro FPGAs which contain the reconfigurable image stream processing data path, large SDRAM memories for multiple frame storage and a PCI express communication backbone network. The FPGA-embedded CPU is used for control and less computation intensive tasks.

This paper will focus on three key aspects: a) the used design methodology which combines macro component configuration and macro-level floorplanning with weak programmability using distributed microcoding, b) the global communication framework with communication scheduling and c) the configurable, multi-stream scheduling SDRAM controller with QoS support by access prioritization and traffic shaping.

As an example, a complex noise reduction algorithm including a 2.5 dimensions DWT and a full 16x16 motion estimation at 24 fps requiring a total of 203 Gops/s net computing performance and a total of 28 Gbit/s DDR-SDRAM frame memory bandwidth will be shown.

Keywords: digital film, FPGA, reconfigurable, stream-based architechture, weak programming, SDRAM-controller, QoS, communication centric, communication scheduling, PCI-Express

*funded in part by the German Federal Ministry of Education and Research (BMBF).

1. Introduction

Digital film post processing (also called *electronic film post processing*) at resolutions of 2Kx2K (2048x2048 pixels per frame at 30 bit/pixel and 24 pictures/s resulting in an image size of 15 MBytes and a data rate of 360 MBytes per second) [1], [2] and up are used in the motion picture and advertisement industries. There is a growing market segment that requires real-time or close to real-time processing to get immediate feedback in interactive film processing. Some of those algorithms are highly computation demanding, far beyond current DSP or processor performance. Typical state-of-the art products in this low-volume, high-price market use FPGA based hardware systems with fixed configurations.

Upcoming products face several challenges, such as increasing computing demands and algorithm complexity, larger FPGA architectures with floorplanning requirements, and increasing demands to product customization.

Another key challenge is the global communication infrastructure and communication scheduling which is required due to the high bandwidth and usage of multiple FPGAs and boards to implement the algorithm.

Furthermore, large external memory space holding several frames is of major importance since the embedded FPGA memories are too small. If not carefully designed, external memory access will become a bottleneck. Since modern FPGAs contain both the image processing data path and embedded processors which access the same memory, different access patterns have to be considered.

This paper presents an answer to these challenges in the form of the FlexFilm [3] hardware platform in section 2.1 and its software counterpart FlexWAFE [4] (Flexible Weakly-programable Advanced Film Engine) in section 2.2. Section 2.3 will discuss the global communication architecture. Since external memory and the required memory access scheduling play a key role, chapter 2.4 will explain the memory controller architecture.

An example of a 2.5 dimensions noise reduction application using bidirectional motion estimation/compensation and wavelet transformation is presented in section 3. Section 4 will show some example results about the quality of service features of the memory controller. Finally, section 5

concludes this paper.

1.1. Related Work

The Imagine stream processor [5] uses a three level hierarchical memory structure: small registers between processing units, one 128KB stream register file and external SDRAM. It has eight arithmetic clusters each with six 32-bit FPUs (floating point units) that execute VLIW instructions. Although it is a stream oriented processor, it does not achieve the theoretical maximum performance due to stream controller and kernel overhead.

The methodology presented by Park et al. [6] is focused on application level stream optimizations and ignore architecture optimizations and memory prefetching.

In the recent years, the paradigms "Communication Centric Design" and "Communication Scheduling" have grown in importance for SoCs and especially MPSoCs as well as for complete systems [7, 8]. A great amount of work and research is currently active in the field of network architectures and communication scheduling ([9, 10, 11, 12, 13]). Allthough similar to NoC, none of this works covers a heterogeneous network of a multi-board / multi-FPGA system with a PCI-Express based backbone.

Regarding external memory, the Prophid architecture[14] [15] by Meerbergen et al. describes a dynamic RAM scheduler for the Prophid DSP platform that is focused on streams using large FIFO buffers and round-robin scheduling. The memory scheduler of the previously mentioned Imagine processeor is optimized for the application algorithms, however it does not distinguish different stream types. Closest to the FlexFilm memory controller architecture is a memory scheduler IP offered by Sonics [16] that also handles different access patterns and service levels at high average memory bandwidth. The bandwidth is similar to that of the scheduler presented here and is close to the maximum possible bandwidth. However, their complex 7-stage architecture is designed towards ASICs and not FPGAs.

1.2. Technology Status

Current FPGAs achieve over 300 MHz, have up to 1 MByte distributed RAM and up to 512 18-bit MAC units (source Xilinx *Virtex-IV* [17]). Together with the huge amount of distributed logic in the chip (up to 100K CLBs [17]) it is possible to build circuits that compete with ASICs regarding performance, but have the advantage of their configurability and therefore reuse.

PCI-Express, mainly developed by Intel and approved as a PCI-SIG standard in 2002, is the successor of the PCI bus communication architecture. Rather than a shared bus, it is a network framework consisting of a series of bidirectional point-to-point channels connected through switches. Each

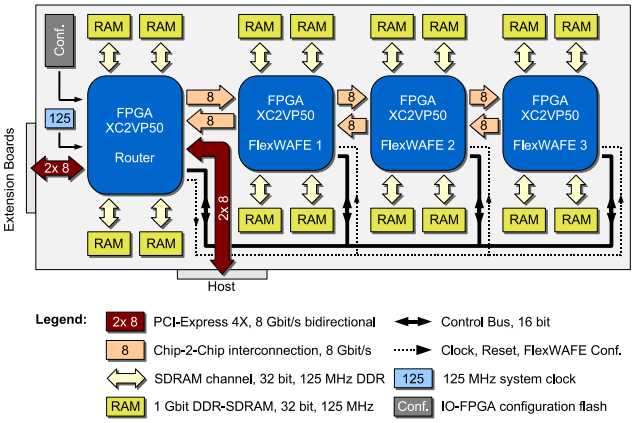

Figure 1. FlexFilm board (block diagram)

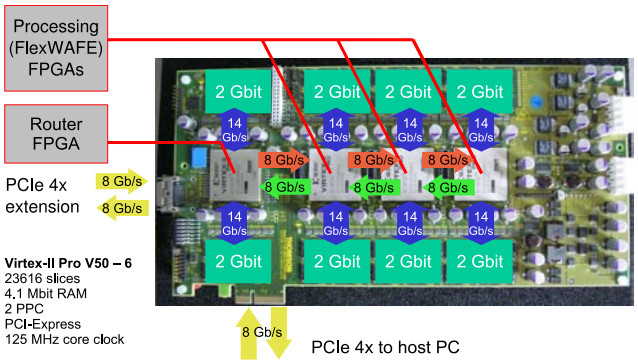

Figure 2. FlexFilm board

channel can operate at the same time without negatively affecting other channels. Depending on the actual implementation, each channel can operate at speeds of 2 (X1-speed), 4, 8, 16 or 32 (X16) Gbit per second (full duplex, both directions each). Furthermore, PCI-Express features a sophisitcated quality of service management to support a variety of end-to-end transmission requirements like minimum guaranteed throughput or maximum latency.

2. FlexFilm Architecture

2.1. System Architecture

In an industry-university collaboration, a multi-board, extendible FPGA based system has been designed. Each FlexFilm board (figures 1 and 2) features 3 Xilinx XC2PV50-6 FPGAs which provide the massive processing power required to implement the image processing algorithms. Another FPGA (also Xilinx XC2PV50-6) acts as a PCI-Express router with two PCI-Express X4 links, enabling 8 Gbit/s net bidirectional communication with the host PC and with other boards (figure 3).

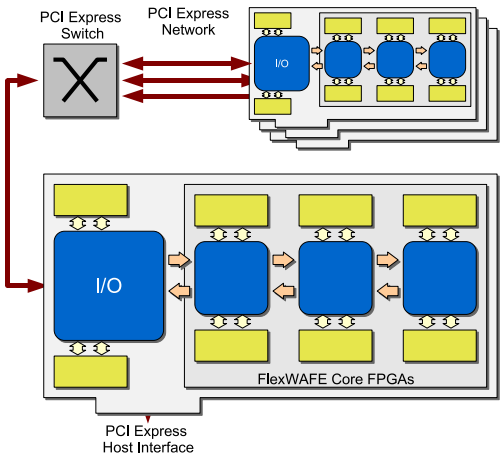

Figure 3. Global system architecture

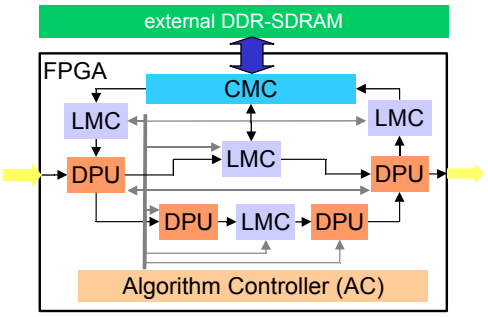

Figure 4. FlexWAFE Reconfigurable architecture

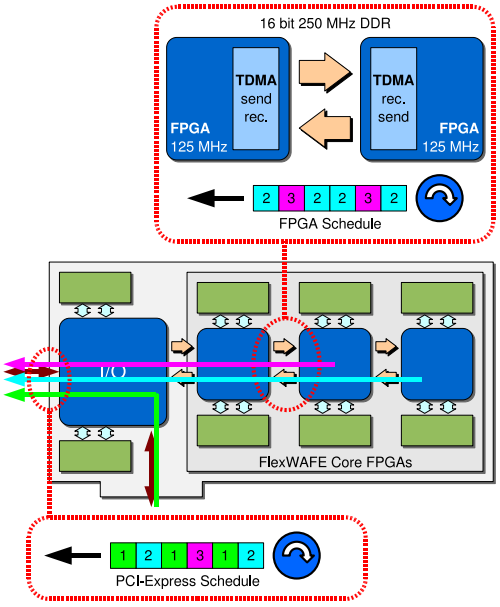

Figure 5. Communication scheduling

The board-internal communication uses multiple 8 Gbit/s FPGA-to-FPGA links, implemented as 16 differential wire pairs operating at 250 MHz DDR (500 Mbit/s per pin), which results in a data rate of one 64-bit word per core clock cycle (125 MHz) or 8 Gbit/s. Additional control lines are available for scheduler synchronization.

As explained in the introduction, digital film applications require huge amounts of memory. However, the used Virtex-II Pro FPGA contains only 4.1 Mbit of dedicated memory ressources (232 RAM blocks of 18 Kbit each). For this reason, each FPGA is equipped with four gigabit of external DDR-SDRAM, organized as four independent, 32 bit wide modules of one gigabit each. Two modules can be combined into one 64-bit module if desired. The RAM is clocked with the FPGA core clock of 125 MHz which results at 80% bandwidth utilizastion in a sustained effective performance of 6.4 Gbit/s per module.

2.2. FlexWAFE Reconfigurable Architecture

The FPGAs are configured using macro components that consist of local memory address generators (LMC) that support sophisticated memory pattern transformations and data stream processing units (DPUs). Their sizes fit the typical FPGA blocks and can be easily laid out as macro blocks reaching a clock rate of 125 MHz. They are parameterized in data word lengths, address lengths and supported address and data functions. The macros are programmed via address registers and function registers and have small local sequencers to create a rich variety of access patterns, including diagonal zig-zagging and rotation. The adapted LMCs are assigned to local FPGA RAMs that serve as buffer and parameter memories. The macros are programmed at run time via a small and, therefore, easy to route control bus. A central algorithm controller sends the control instructions to the macros controlling the global algorithm sequence and synchronization. Programming can be slow compared to processing as the macros run local sequences independently. In effect, the macros operate as weakly programmable coprocessors known from MpSoCs such as VIPER [18]. This way, weak programming separates time critical local control in the components from non time-critical global control. This approach accounts for the large difference in global and local wire timing and routing cost. The result is similar to a local cache that enables the local controllers to run very fast because all critical paths are local. Figure 4 shows an overview.

2.3. Global Data Flow

Even if only operating at the low 2K resolution, one image stream alone comes at a data rate of up to 2.2 Gbit/s.

With the upcoming 4K resolution, one stream requires a net rate of 8.8 Gbit/s. At the processing stage, this bandwidth rises even higher, for example because multiple frames are processed at the same time (motion estimation) or the internal channel bit resolution is increasing to keep the desired accuracy (required by filter stages such as DWT). Given the fact that the complete algorithm has to be mapped to different FPGAs, data streams have to be transported between the FPGAs and boards. These data streams might differ greatly in their characteristics such as bandwidth and latency requirements (e.g. image data and motion vectors), and it will be required to transport multiple streams over one physical communication channel. Minimum bandwidths and maximum possible latencies must be guaranteed, this is accomplished through communication scheduling.

Therefore it is obvious that due to this reasons the communication architecture is therefore a key point of the complete FlexFilm project. The first decision was made to abandon any bus structure communication architecture, since due to their shared nature their available effective bandwidth becomes too limited if many streams need to be transported simultaneously. Furthermore, current bus systems do not provide a quality of service management which is required for a communication scheduling.

For this reason, point-to-point communication channels were used for inter-FPGA communication on one board and PCI-Express was selected for board-to-board communication. Currently, PCI-Express is only used for stream input and output to a single FlexFilm board, however in the future multiple boards will be used.

As explained above, multiple streams have to be transported over one physical channel, and communication scheduling has to guarantee a smooth and deadlock-free operation. Latencies should be kept at a minimum, since large latencies require large buffers which have to be implemented inside the FlexWAFE FPGAs and which are nothing but "dead freight."

Since the streams (currently) are periodic and their bandwidth is known at design time, TDMA (time division multiple access) scheduling is a suitable solution. TDMA means that access to the communication channel is granted to each stream in turn. The bandwidth assigned to each stream can be specified by either a) the access duration per stream – the higher, the more bandwidth is assigned to that stream, or b) by giving access to a stream more often during a complete TDMA schedule.

For example, take 2 streams (1) and (2), whereas the required bandwith for stream (1) is 3 words every 5 clock cycles and stream (2) requires 2 words every clock cycle. In the first solution, stream (1) would be granted access for three clock cycles followed by stream (2) for two cycles, which can be expressed as a TDMA schedule of 1-1-1-2-2. Buffers are required at the sender and receiver for packetizing and depacketizing of the periodic streams. In the second solution, the TDMA schedule would be 1-2-1-2-1, resulting in the same bandwidth assignment, however due to the smaller packet size less buffers are required. For this reason, second solution was choosen for the intra-FPGA communication with a packet size of one word.

In many TDMA implementations, small packet sizes reduce the effective available bandwidth due to packet headers and trailers which carry checksums and scheduling information. This is not required by the FlexFilm project since due to short wire lengths and differential signaling techniques transmission errors are neglectable, and the required scheduling information is transmitted by additional control lines.

Since PCI-Express can be operated as a TDMA bus, the same scheduling techniques apply as for the inter-FPGA communication. The only exception is that PCI-Express requires a larger packet size of currently up to 512 bytes[1]. The required buffers however fit well into the IO-FPGA.

Figure 5 shows an inter-FPGA and a PCI-Express schedule example.

2.4. Memory Controller

As explained in the introduction, memory access patterns from the embedded CPU (PowerPC) and the stream processing data path show different access patterns:

- *Hard real-time periodic access sequences* and *fixed address access patterns* with optimized fixed scheduling [19] generated by the data path. These sequences require a *minimum memory throughput* and can be buffered to increase the maximum allowed memory access time without performance loss. By adapting the buffer size, arbitrary access times are acceptable, but the maximum access time must be bounded to avoid buffer over- or underflows.

- *Random address patterns* generated by *CPU cache misses*, either soft- or hard real-time. Because the processor stalls on a cache miss and since prefetch is of limited use due to the less predictable access patterns, memory access time is the crucial parameter determining performance. On the other hand, (average) memory throughput is less significant. To improve performance, memory access time has to be minimized (soft real-time) or bounded (hard real-time). Such requirements are typically neglected but they are important for the overall system performance, as will be seen in the experiments.

This results in two types of QoS: *guaranteed minimum throughput at guaranteed maximum latency* and *smallest possible latency*.

[1] Limitation by Xilinx PCI-Express IP core

The 125 MHz clocked RAM reaches a peak performance per bank of 8 Gbits per second. To avoid external memory access becoming a bottleneck, a scheduling memory controller (CMC[2]) was developed which implements the QoS types explained above. Other constrains were flexible configurability (see chapter 3) and resource usa ge – since the CMC does not perform stream processing, it can be considered as (required) ballast and thus the area usage should be kept at a minimum.

Figure 6 shows the CMC block diagram. The controller core, the two-staged memory access scheduler, is capable of increasing the bandwidth utilization by applying bank interleaving (minimizes the bank stall cycles) and read/write request bundling (minimizes bus turnaround cycles). QoS is implemented by prioritizing PowerPC memory accesses (high priority CPU requests are always executed before standard priority data path requests) and by traffic shaping those high priority requests to prevent data path request starvation. The traffic shaper allows patterns such as n requests within a window of T clock cycles (also known as "leaky bucket" principle in networking applications).

The memory accesses the SDRAM using auto precharge mode, requests to the memory controller are always done at full SDRAM bursts. To avoid excessive memory stalls due to SDRAM bank precharge and activation latencies, the address translation is performend in a way that memory requests are evenly distributed across all banks to maximize the bank interleaving effect.

The memory controller is configurable regarding SDRAM timing and layout, application ports (number of ports, different data and address width per port), address translation and QoS settings (prioritization and flow control).

A more detailed description can be found in [20] and [21].

2.5. FPGA Programming

FPGA programming consists of script-based VHDL macro parameterization based on library elements, followed by controller synthesis and FPGA floorplanning. This combination greatly simplifies programming as the resulting modules can easily be placed in the floorplan, avoiding time consuming and ineffective global routing with manual optimization. For system programming, there is a global flow programming tool under development that semiautomatically assigns communication link bandwidth and memory space to the operators of a global data flow graph. So far, these design steps are manual. The design flow is depicted in Figure 7.

[2]Central Memory Controller; historic name, emerged when it was supposed to only have one external memory controller per FPGA.

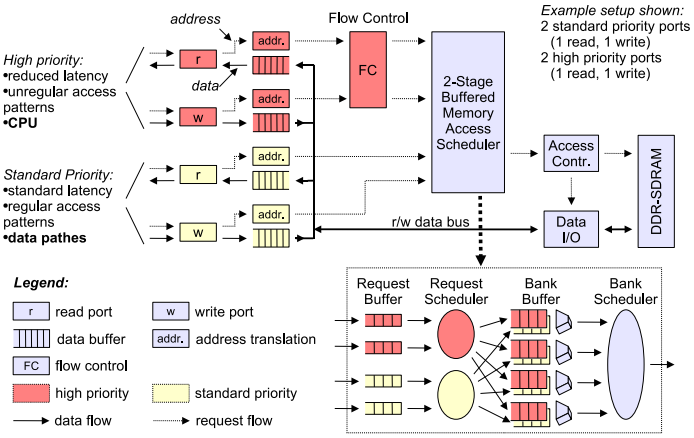

Figure 6. Memory Controller Block Diagram

3. A Sophisticated Noise Reducer

To test this system architecture, a complex noise reduction algorithm depicted in figures 8 and 9 based on 2.5 dimensions discrete wavelet transformation (DWT) between consecutive motion compensated images was implemented at 24 fps. The algorithm starts by creating a motion compensated image using pixels from the previous and from the next image. Then it performs a Haar filter between this image and the current image. The two resulting images are then transformed into the 5/3 wavelet space, filtered with user selectable parameters, transformed back to the normal space and filtered with the inverse Haar filter. The DWT operates only in the 2D space-domain, but due to the motion compensated pixel information, the algorithm also uses information from the time-domain, therefore it is said to be a 2.5D filter. A full 3D filter would also use DWT in the time domain, therefore requiring multiple consecutive images (typically 5). The algorithm is presented in detail in [22].

3.1. Motion estimation

Motion estimation is used in many image processing algorithms and many hardware implementations have been proposed. The majority are based on block matching. Of these, some use content dependent partial search. Others search exhaustively, in a data independent manner. Exhaustive search produces the best block matching results at the expense of an increased number of computations.

A full-search, block matching ME operating in the luminance channel and using the sum of absolute differences (SAD) search metric was developed because it has predictable, content independent memory access patterns and can process one new pixel per clock cycle. The block size is 16x16 pixels and the search vector interval is -8/+7. Its im-

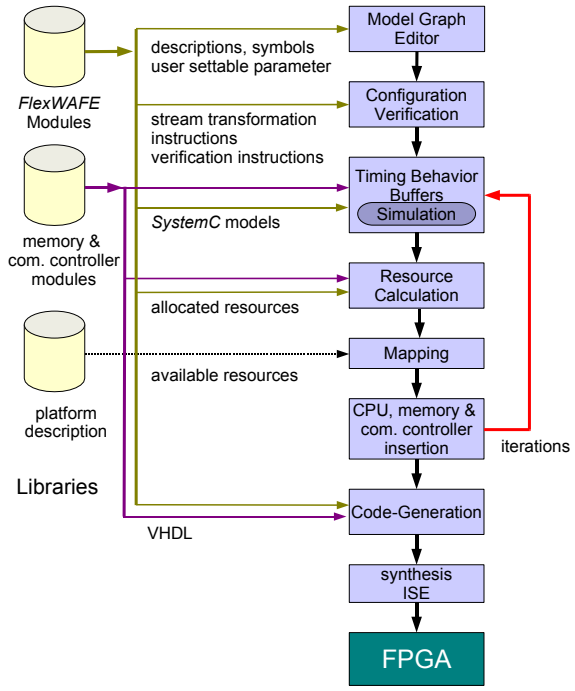

Figure 7. Designflow

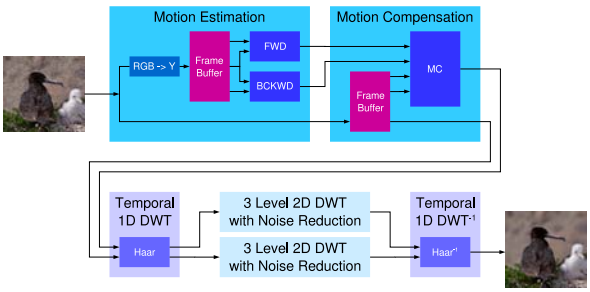

Figure 8. Advanced noise reduction algrithm

plementation is based on [23]. Each of the 256 processing elements (PE) performs a 10 bit difference, a comparasion, and a 19 bit accumulation. These operations and their local control was accommodated in 5 FPGA CLBs as shown in Figure 10. As seen in the rightmost table of that figure, the resource utilization within these 5 CLBs is very high and even 75% of the LUTs use all of its four inputs. This block was used as a Relationally Placed Macro (RPM) and evenly distributed on a rectangular area of the chip. Unfortunately each 5 CLBs only have 10 tri-state buffers which is not enough to multiplex the 19 bit SAD result, therefore the PEs are accommodated in groups of 16 and use 5 extra CLBs per group to multiplex the remaining 9 bits. Given the cell-based nature of the processing elements, the timing is preserved by this placement. To implement the 256 PEs with corresponding SAD bus, 1360 CLBs and 26 extra

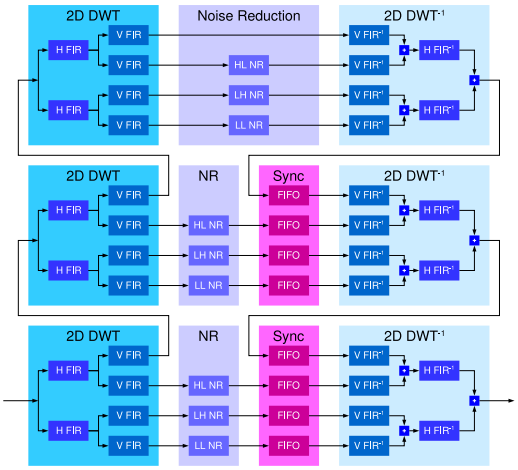

Figure 9. DWT based 2D noise reduction

CLBs are required for finding the minimum SAD including global control. On the edge of the images the motion vectors can only have limited values, a fact that is used to reduce the initial row latency of [23] from 256 to 0.

Bidirectional motion-estimation is achieved using two of these blocks. The ME core processing elements require that the images be presented at its inputs in a column-major way, but the images are transfered between FPGAs and stored in SDRAM in a row-major order. Therefore, each of the PE's three inputs gets data from memory via a group of two LMCs. The first hides the SDRAM latency by performing prefetching as explained in [4] whilhe the second transforms the accesses from row-major to column-major order using a small local blockRAM. When fed with the luminance component of 2048x2048 pixels, 10 bit-per-pixel images at 24 frames per second, the core computational power (ignoring control and data transfers) is 155 Gop/s (tested in post-layout simulation).

The resulting performance is higher than known implementations using NVIDIA GPUs [24], significantly above the 18 Gop/s of the IMAGINE dedicated image processing ASIC [5] running at 400MHz, and far beyond the 0.8 Gop/s of a leading TI fixed-point TMS320C64x DSP running at 1GHz [25].

3.2. Motion Compensation

Motion compensation uses the block motion vectors found by the ME to build an image that is visually similar to the current image, but only contains pixels extracted in a blockwise manner from the previous/next image. The criteria to choose the image block from the previous or next image is the SAD associated to that block, the image block with the smallest SAD (and therefore more similar to the current image block) of the two is chosen. On a scene cut,

Resource	Usage	Percentage
RAMB	44 out of 232	18%
Slices	20,583 out of 23,616	87%
TBUF	5,408 out of 11,808	45%

- bidirectional ME with block size 16x16
- bidirectional MC
- searches -8/+7 vector interval
- 24 fps @ 2048x2048, 10bpp (125 MHz)
- 1024 net add/sub operations/pixel
- 514 net comparations operations/pixel
- 155 net Goperations/s

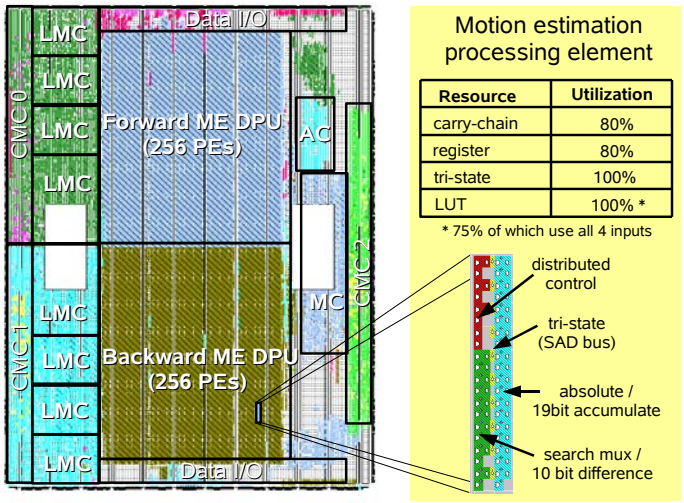

Figure 10. Mapping and resource usage in a Xilinx XC2V50P device

one of the images will produce large SADs (because the contents of it are most probably completely different from the current image) and all blocks will be chosen from the other image, the one that belongs to the same scene. This has the advantage of making the noise reduction algorithm *immune* to scene cuts.

3.3. Discrete Wavelet Transform

The discrete wavelet transform (DWT) transforms a signal into a space where the base functions are wavelets [26], similar to the way Fourier transformation maps signals to a sine-cosine based space. The 5/3 wavelet was chosen for its integer coefficients and invertibility (the property to convert back to the original signal space without data loss). The 2D wavelet transformation is achieved by filtering the row major incoming stream with two FIR filters (one with 5 the other with 3 coefficients) and then filtering the resulting two signals columnwise using the same filter coefficients. The four resulting streams can be transformed back to the original stream by filtering and adding operations. The noise reduction algorithm requires three levels of decomposition, therefore three of these blocks were cascaded and the noise reduction DPUs added. To compensate the latency of the higher decomposition levels, LMCs were used to build FIFOs with the external SDRAM. The resulting system is depicted in figure 9 and was presented in detail in [4] .

The filter implementation uses polyphase decomposition (horizontal) and coefficient folding (vertical). To maximize throughput, the transformation operates line-by-line instead of level-by-level [27]. This allows for all DPUs to operate in parallel (no DPU is ever idle), minimizes memory requirements and performs all calculations as soon as possible. Because the 2D images are a finite signal some control

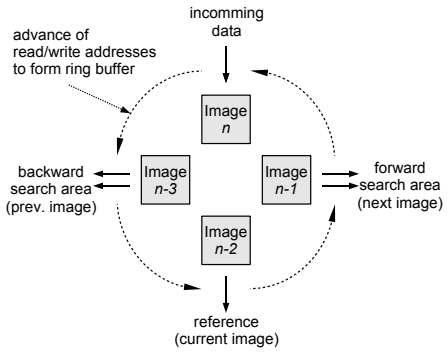

Figure 11. Frame buffer access sequence

was added to achieve the symmetrical periodic extension (SPE) [28] required to achieve invertibility. This creates a dynamic datapath because the operations performed on the stream depend on the data position within the stream. All multiply operations were implemented with shift-add operations because of the simplicity of the coefficients used. One 2D DWT FPGA executes 162 add operations on the direct DWT, 198 add operations on the inverse DWT and 513 extra add operations to support the SPE, all between 10 and 36 bits wide.

3.4. External Memory

Figure 11 shows the required frame buffer access structure of the motion estimation. As can be seen, three images are accessed simultaneously, one image as reference ($n-2$), and two images as backward and forward search area ($n-3$ and $n-1$). The two search areas are read twice with different addresses. Besides that, the current in-

coming image (n) needs to be buffered. Each of the two ME engines contains its own frame buffer to store four full-size images of up to 4Kx4K accessed via its respective CMC0 or CMC1 (Figure 10). Each of the CMCs writes one stream to memory and reads three streams. For ease of implementation each pixel is stored using 16 bits. This translates to 1.5 Gbit/second write and 4.1 Gbit/second read bandwidth to off-chip SDRAM, amounting to a total of 6.1 Gbit/second that is below the maximum practical bandwidth of 7 Gbit/second. The MC block operates in the RGB color space unlike the ME block that uses the luminance only. It stores one RGB pixel in a 32 bit word (10 bits per color component) and uses its own memory controller (CMC2 on Figure 10). It uses a similar ring-buffer scheme as CMC0 and 1 and is also capable of storing four images of up to 4Kx4K resolution, but it groups the two external memory banks and accesses them via a 64 bit bus and is therefore capable of twice the troughput of the ME's CMCs. Due to the nature of SDRAM accesses it is only possible to access blocks of 16 pixels at addresses that are multiples of 16 (memory alignment). This means that in the worst-case two blocks of 16 pixels need to be fetched in order to access a non-aligned group of 16 pixels to build the motion compensated image. The MC block also needs to access the current image in order to do intra-block pixel-by-pixel validation of the results. This leads to a worst case bandwidth of 3.0 Gbit/second write and 9.2 Gbit/second read which is below the practical limit of 14 Gbit/second. The area occupied by these 3 memory controllers is about 12 % of the FPGA area, leaving enough room for the stream processing units.

As explained in chapter 3, the DWTs need synchronization FIFOs to compensate the additional latency of the higher level DWTs. The level 2 FIFOs completely fit into the FPGA internal memory, so only for the level 1 FIFOs external SDRAM was required. Due to layout issues, two memory controllers in a 64-bit configuration were used for separate buffering of read channel and the green/blue channels.

The ME/MC-FPGA requires 3 CMCs in 2 different configurations; the DWTs each require 2 controllers in 2 configurations. Together with the 2 controllers in the router FPGA, 9 memory controllers in 5 configurations were used alltogether.

Since in this application the PowerPC and therefore the QoS features are not (yet) used, these features were separately tested, see chapter 4.

3.5. Mapping and Communication

The complete algorithm was mapped onto the three FlexWAFE image processing FPGAs of a single FlexFilm board. Stream input and output is done via the router FPGA and the PCI-Express host network, the second PCI-Express

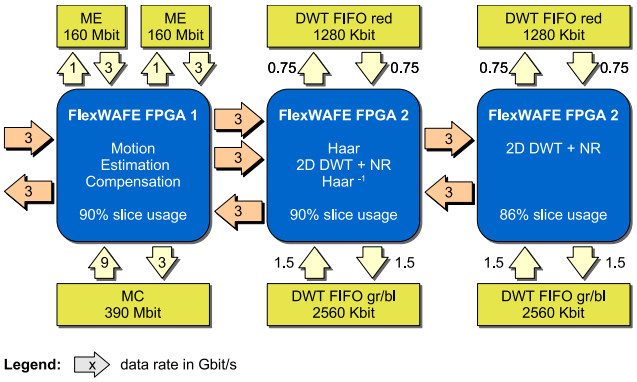

Figure 12. Algorithm mapping

port remains unused. Input and output streams require a net bandwidth of 3 Gbit/s each which can be easily handled by the X4 PCI-Express interface. Since only single streams are transmitted, no TDMA scheduling is necessary. The packetizing and depacketizing of the data, as well as system jitter compensation is done by double-buffering the incomming and outgoing images in the external RAM.

Figure 12 shows the mapping solution (router FPGA omitted). The 1st FlexWAFE FPGA contains the Motion-Estimantion and MotionCompensation, the 2nd FPGA the Haar- and inverse Haar filters and one 2D DWT noise reduction block, the 3rd FPGA contains the other 2D DWT noise reduction block.

As can be seen, between the 1st and the 2nd FPGA two independent image streams – the original and the motion compensated images – with a total bandwidth of 6 Gbit need to be transported over one physical channel. The word size of both streams is 30 bit (10 bit RGB). For transport, always two words of each stream were merged and zero-padded to form a 64 bit word. The TDMA scheduler was programmed with an (in this case simple) sequence of 1-2, which means that the 64 bit words were transmitted in alternating way. Due to the small TDMA packet size of two words no external SDRAM was required for buffers.

Due to the mapping onto different SDRAM channels as explained in 3.4 the maximum effective bandwidth per SDRAM channel of 7 Gbit/s (14 Gbit/s for a 64 bit combined channel, respectively) was not an issue.

3.6. Implementation

Each buildding block has been programmed in VHDL using an extensive number of *generics* (VHDL language constructs that allow parameterizing at compile-time) to increase the flexibility and reuse. The sequence of run-time programable parameters for the LMCs image transfers (the contents of the AC memory) were described in XML and transformed to VHDL via XSLT. In the future it is planed

to use even more scripts and XML based high-level system descriptions. Each block was behaviorally simulated individually using Modelsim 7.1b and synthesized using Xilinx ISE 7.1i SP4. All blocks except the motion compensation have also been simulated after place-&-route and the desired functionality and speed were achieved. The data I/O blocks and the CMCs have also been tested in hardware.

Currently the ME and MC are being integrated in a single chip (Figure 10), and due to the large resource utilization (87% of the FPGA slices are used) floorplaning is necessary to achieve the required speed. So far the Xilinx Floorplanner has been used, but in the future it is planned to use Xilinx PlanAhead and/or Synplicity Premier with Design Planner. Both softwares are currently being evaluated.

3.7. Outlook

Currently, the 56 filter parameters used by the DWT filter are static. However, in [22] it is shown that the results can be significantly improved by runtime adapting the filter coefficients depending on the image. The required calculations will be done by a PowerPC which will have to access parts of the images in the MotionCompensation FPGA frame memory, thus requiring the QoS service features of the memory controller.

4. SDRAM QoS

In [20] and [21] a complex simulator setup was used before availability of the FlexFilm board to evaluate the CMC QoS architecture. However, due to lack of a cycle-accurate instruction set simulator for the embedded PowerPC 405 core, these results were inaccurate. Therefore, a real test environment was created, consisting of the PowerPC, two SDRAM-controllers, two load generators and the required PowerPC-CMC interfaces (figure 13).

The load generators (one read and one write) created memory access streams with linear address patterns similar to the DWT filters and a programmable period. Requests to the SDRAM were done at 64 bit and a burst length of 8 words, which means the maximum possible period is 8 clock cycles. Since one SDRAM data transfer takes 4 clock cylces (8 words @ 64 bits, 2 words per clock cycle), two load generators running at a period of 8 clocks would have created a theoretical SDRAM load of 100 %. However, due to memory stall cycles (refresh cycles, bus switching stall cycles) a maximum period of 10/9 (or 9/10) was possible, resulting in a bandwidth utilization of 72 %. If the period is too small, the load generators start loosing memory requests (they cannot operate in real-time any more).

The PowerPC was clocked at 250 MHz and executed an adapted version of the JPEG decrompression program from the MiBench benchmark suite [29] (we have chosen a

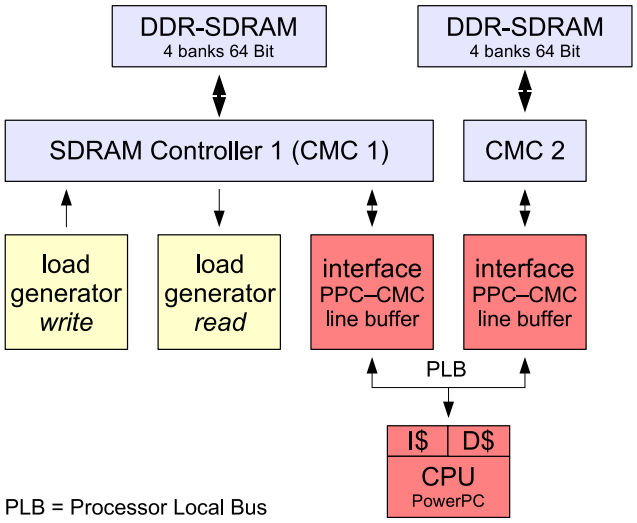

Figure 13. CMC QoS test environmant

real application rather than artificial benchmarks like SPEC for more realistic results). Code and data were completely mapped to the 1st memory controller. Since the original program accessed the harddisk to read and write data, in our environment the Xilinx Memory-FileSystem was used which was mapped to the 2nd memory controller. Both PPC instruction and data caches (16K each) were activated, at a cache miss 4 words @ 64 bits were read and/or written to the memory. Since one memory access burst reads or writes 8 words @ 64 bits, the PowerPC-CMC interface contains an additional line buffer ("1-burst-cache").

CPU accesses to the 1st memory controller were optionally priorized and flow controlled. Table 1 shows the test results.

With the CPU activated, but without prioritization (and flow control) the load generators start loosing requests (that means, missing the deadline) at periods of 11/11 (nr. 1.1 to 1.3). Activating CPU priorities leads to a noticeable CPU speedup, however the load generators start loosing requests very quickly, with results getting worse at periods of 11/11 (nr. 2.1 to 2.3). With flow control enabled, there is still a CPU speedup compared to the non-priorized test, however this time the load generators are fully operational again (nr. 3.1 to 3.6). Moreover, results 3.5 and 3.6 show that with CPU prioritization and traffic shaping enabled load periods of 11/11 are possible which is not the case without any QoS service (1.3). Activating complex traffic shaping patterns shows a positive, albeit very small effect.

5. Conclusion

A record performance reconfigurable HW/SW platform for digital film applications was presented. The combina-

nr.	pri. CPU access	flow control T/n	load gen. period read	load gen. period write	CPU exec. time [ms]	lost requests read	lost requests write
1.1	no	deactivated	12	12	2195	0	0
1.2	no	deactivated	12	11	2217	0	0
1.3	no	deactivated	11	11	2263	5	7
2.1	yes	deactivated	12	12	2119	258	175
2.2	yes	deactivated	12	11	2121	4,324	7,598
2.3	yes	deactivated	11	11	2121	25,872	25,871
3.1	yes	32 / 1	12	12	2144	0	0
3.2	yes	61 / 2	12	12	2123	0	0
3.3	yes	45 / 1	12	11	2169	0	0
3.4	yes	93 / 2	12	11	2163	0	0
3.5	yes	57 / 1	11	11	2209	0	0
3.6	yes	113 / 2	11	11	2193	0	0

Flowcontrol T/n: n requests within T clock cycles.

Table 1. SDRAM controller test results

tion of programmable and parameterized macros that can easily be handled in floorplaning and decentralized weak programming with non-critical timing was key to a high designer productivity.

We have also shown that the scheduling memory controller with QoS support helps to improve the overall system performance.

The FPGA resource utilization is very satisfactory including memory and routing resources. The FlexWAFE architecture is part of a larger project towards an extendible PCI-Express based real time film processing system.

References

[1] http://www.quantel.com.

[2] http://www.discreet.com.

[3] http://www.flexfilm.org.

[4] Amilcar do Carmo Lucas and Rolf Ernst. An Image Processor for Digital Film. In *IEEE ASAP*, July 2005.

[5] Jung Ho Ahn, William J. Dally, Brucek Khailany, Ujval J. Kapasi, and Abhishek Das. Evaluating the Imagine Stream Architecture. *SIGARCH Comput. Archit. News*, 32(2):14, 2004.

[6] Joonseok Park and Pedro C. Diniz. Syntesis of Pipelined Memory Access Controllers for Streamed Data Applications on FPGA-based Computing Engines. In *ISSS*. ACM, October 2001.

[7] William J. Dally. Computer architecture is all about interconnect (it is now and it will be more so in 2010). In *International Symposium on High Performance Computer Architecture (HCPA)*, February 2002.

[8] Design Issues in Distributed, Communication-Centric Systems. In *ARTIST Workshop at DATE-Conference*, 2006.

[9] Florin Dumitrascu, Iuliana Bacivarov, Lorenzo Pieralisi, Marius Bonaciu, and Ahmed A. Jerraya. Flexible MPSoC Platform with Fast Interconnect Exploration for Optimal System Performance for a Specific Application. In *Design, Automation and Test in Europe (DATE)*, March 2006.

[10] Umit Y. Ogras, Radu Marculescu, Hyung Gyu Lee, and Naehyuck Chang. Communication Architecture Optimization: Making the Shortest Path Shorter in Regular Networks-on-Chip. In *Design, Automation and Test in Europe (DATE)*, March 2006.

[11] Frits Steenhof, Harry Duque, BjÃ¶rn Nilsson, Kees Goossens, and Rafael Peset Llopis. Networks on Chips for High-End Consumer-Electronics TV System Architectures. In *Design, Automation and Test in Europe (DATE)*, March 2006.

[12] Sorin Manolache, Petru Eles, and Zebo Peng. Buffer Space Optimisation with Communication Synthesis and Traffic Shaping for NoCs. In *Design, Automation and Test in Europe (DATE)*, March 2006.

[13] Sudeep Pasricha and Nikil Dutt. COSMECA: Application Specific Co-Synthesis of Memory and Commu-

nication Architectures for MPSoC. In *Design, Automation and Test in Europe (DATE)*, March 2006.

[14] Jeroen A. J. Leitjen, Jef L. van Meerbergen, and Adwin H. Timmer. PROPHID: a Heterogeneous Multi-Processor Architecture for Multimedia. In *International Conference on Computer Design*, pages 164–169, October 1997.

[15] W. Verhaegh, P. Lippens, and E. Aarts. Multi-dimensional periodic scheduling: model and complexity. pages 226–235. Springer Verlag, 1996.

[16] Wolf-Dietrich Weber. Sonics MemMax Memory Scheduler. 2003.

[17] http://www.xilinx.com.

[18] S. Dutta, R. Jensen, and A. Rieckmann. Viper: A multiprocessor SoC for advanced set-top box and digital TV systems. In *IEEE Design and Test of Computers, Sip*, pages 21–31, Oct 2001.

[19] P.R. Panda, F. Catthoor, and N.D. Dutt. Data and Memory Optimization Techniques for Embedded Systems. 6(2):140–206, April 2001.

[20] Sven Heithecker, Amilcar do Carmo Lucas, and Rolf Ernst. A Mixed QoS SDRAM Controller for FPGA-Based High-End Image Processing. In *Workshop on Signal Processing Systems Design and Implementation*, page TP.11. IEEE, 2003.

[21] Sven Heithecker and Rolf Ernst. Traffic Shaping for an FPGA-Based SDRAM Controller with Complex QoS Requirements. In *Design Automation Conference (DAC)*, page 34.5. ACM, June 2005.

[22] Stefan Eichner, Gunter Scheller, and Uwe Wessely. Wavelet-temporal basierende Rauschreduktion von Filmsequenzen. In *21. Jahrestagung der FKTG, Koblenz*, May 2004.

[23] Cesar Sanz and Matias J. Garrido and Juan M. Meneses. VLSI Architecture for Motion Estimation using the Block-Matching Algorithm. In *EDTC '96*, page 310.

[24] Robert Strzodka and Christoph Garbe. Real-time motion estimation and visualization on graphics cards. In *Proceedings IEEE Visualization 2004*, pages 545–552, 2004.

[25] http://www.ti.com/.

[26] Satyabrata Rout. Orthogonal vs. Biorthogonal Wavelets for Image Compression. Master's thesis, Virginia Polytechnic Institute and State University, 2003.

[27] N. Zervas, G. Anagnostopoulos, V. Spiliotopoulos, Y. Andreopoulos, and C. Goutis. Evaluation of Design Alternatives for the 2D-Discrete Wavelet Transform. In *IEEE Transactions on Circuits and Systems for Video Technology*, page to appear.

[28] Christopher M. Brislawn. Classification of nonexpansive symmetric extension transforms for multirate filter banks. In *Applied and Computational Harmonic Analisys*, volume 3, pages 337–357, 1996.

[29] http://www.eecs.umich.edu/mibench.