A reconfigurable HW/SW platform for computation intensive high-resolution real-time digital film applications*

Amilcar do Carmo Lucas, Sven Heithecker, Peter Rüffer, Rolf Ernst
{lucas | heithecker | rueffer | ernst}@ida.ing.tu-bs.de
Technical University of Braunschweig, Germany

Holger Rückert, Gerhard Wischermann, Karin Gebel, Reinhard Fach, Wolfgang Huther
{holger.rueckert | gerhard.wischermann | karin.gebel | reinhard.fach | wolfgang.huther}@thomson.net
Grass Valley Germany GmbH, Thomson group

Stefan Eichner, Gunter Scheller
stefan.eichner@tu-ilmenau.de, scheller@e-technik.tu-ilmenau.de
Technical University Ilmenau, Germany

Abstract

This paper presents a multi-board, multi-FPGA hardware/software architecture, for computation intensive, high resolution (2048x2048 pixels), real-time (24 frames per second) digital film processing. It is based on Xilinx Virtex-II Pro FPGAs, large SDRAM memories for multiple frame storage and a PCI express communication network. The architecture reaches record performance running 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. To increase design productivity and yet achieve high clock rates (125MHz), the architecture combines macro component configuration and macro level floorplanning with weak programmability using distributed microcoding. As an example, the core of the bidirectional motion estimation using 2772 CLBs reaching 155 Gop/s (1538 op/pixel) requiring 7 Gbit/s external memory bandwidth was developed in two men-months.

Keywords: motion-estimation, weak-programming, stream-based architechture, digital film, reconfigurable, FPGA

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) [4], [1] 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 at once, increasing computing demands and algorithm complexity, larger FPGA architectures with floorplanning requirements, large memory space holding several consecutive frames and increasing demands to product customization.

This paper presents an answer to these challenges in the form of the FlexFilm [2] hardware platform in section 2 and it's software counterpart FlexWAFE [10] (Flexible Weakly-programable Advanced Film Engine) in section 3.

An example of a 2.5 dimensions noise reduction application using bidirectional motion estimation/compensation and wavelet transformation is presented in section 4. Section 5 concludes this paper.

1.1. Related Work

The Imagine stream processor [7] 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 instruc-

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

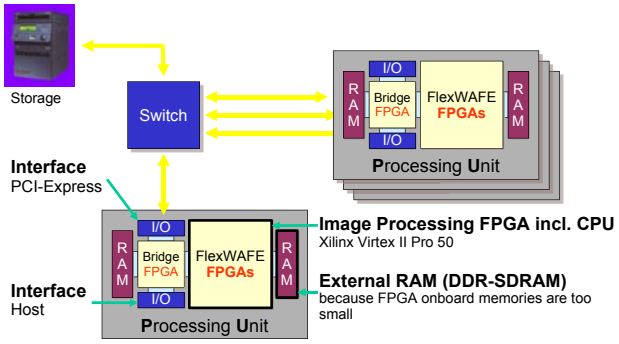

Figure 1. The global system architecture

tions. 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. [15] is focused on application level stream optimizations and ignore architecture optimizations and memory prefetching.

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* [6]). Together with the huge amount of distributed logic in the chip (up to 100K CLBs [6]) it is possible to build circuits that compete with ASICs regarding performance, but have the advantage of their configurability and therefore reuse.

2. FlexFilm System Architecture

In an industry-university collaboration, a multi-board, extendible FPGA based system depicted in figure 1 has been designed. The core processing engines are mounted on PC extension boards and interconnected via PCI-Express [3]

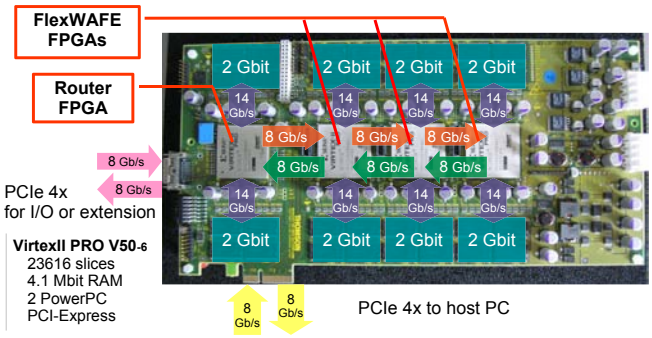

Figure 2. PCI-Express 4x PC extension board

(PCIe) 4x links, shown in Figure 2, each containing four Xilinx XC2PV50-6 FPGAs. Each is equipped with four gigabit of external DDR-SDRAM, organized as four independent, 32 bit wide modules of one gigabit each. The RAM is clocked with the FPGA core clock of 125 MHz which results in a sustained performance of 7 Gbit/s per bank. Three of the FPGAs provide the massive processing power required to implement the image processing algorithms. One FPGA acts as a PCIe router enabling 8 Gbit/s net bidirectional communication with the host PC and 8 Gbit/s net to other boards. The board-internal communication uses 8 Gbit/s FPGA-to-FPGA links.

3. 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 e.g. diagonal zig-zagging or 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 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 [11]. 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 3 shows an overview.

The interface to each SDRAM memory is controlled by a central memory controller (CMC) that applies access prioritization and traffic shaping [14] to merge several data streams and cache accesses from the on-chip PowerPC processors without violating the stream processing real-time constraints that would quickly lead to buffer overflows or underflows.

3.1. Programming

FPGA programming consists of script based VHDL macro parameterization based on library elements followed

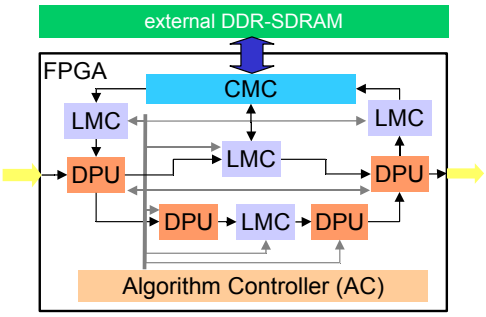

Figure 3. FlexWAFE Reconfigurable architecture

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 on Figure 4.

4. A sophisticated noise reducer

To test this system architecture, a complex noise reduction algorithm 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 an 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 uses also 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 [12].

The algorithm was divided into the three FlexWAFE FPGAs on the flexfilm board. The first one implements ME/MC, the second one implements Haar filtering and 2D DWT and the third 2D DWT.

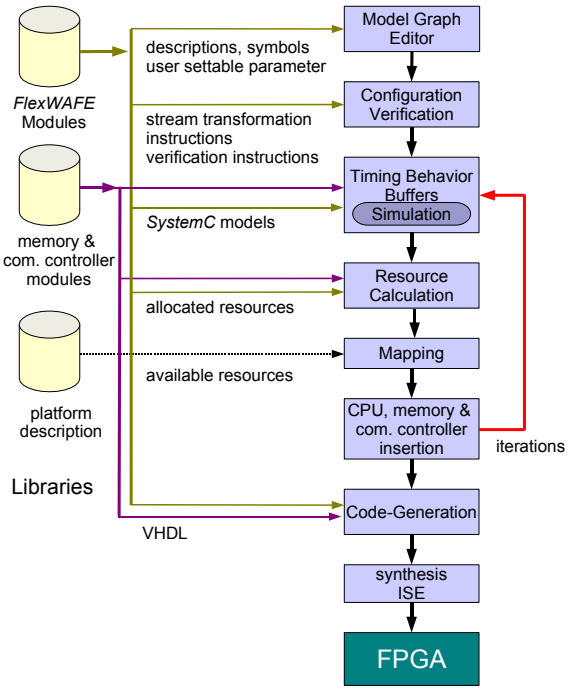

Figure 4. Designflow

4.1. Motion estimation

Motion estimation is used in many image processing algorithms and many hardware implementations have been proposed. The majority is 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 vectors interval is -8/+7. It's implementation is based on [9]. 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 5. 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 and that is not enough to multiplex the 19 bit SAD result, so 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 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, that fact is used to reduce the initial row latency of [9] 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 it's 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 [10], the second one transforms the accesses from row-major to column-major 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 [17], way above the 18 Gop/s of the IMAGINE dedicated image processing ASIC [7] running at 400MHz, and far beyond the 0.8 Gop/s of a leading TI fixed-point TMS320C64x DSP running at 1GHz [5].

4.2. 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, one of the images will produce big 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.

4.3. Discrete Wavelet Transform

The discrete wavelet transform (DWT) transforms a signal into a space where the base functions are wavelets [16], similarly 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 6 and was presented in detail in [10] .

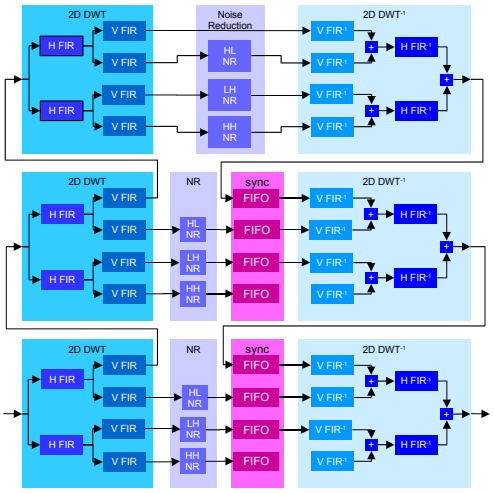

Figure 6. DWT based 2D noise reduction

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 [18]. 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 was added to achieve the symmetrical periodic extension (SPE) [8] 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 where 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.

4.4. External Memory

Like 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 of the FlexFilm Board is equipped

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

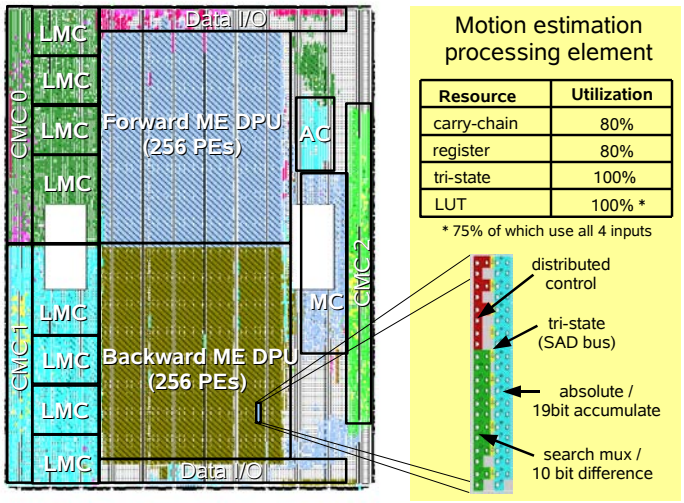

Figure 5. Mapping and resource usage in a Xilinx XC2V50P device

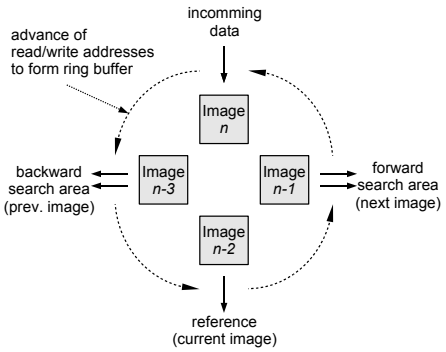

Figure 7. Frame buffer access sequence

with four gigabit of external DDR SDRAM memory, organized as four independent, 32 bit wide modules of one gigabit each (Figure 2). The RAM is clocked with the FPGA core clock of 125 MHz which results in a peak performance per bank of 8 Gbits per second. To access the DDR SDRAM at a high bandwidth a scheduling memory controller (CMC) was developed. Figure 8 shows a simplified 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). The memory accesses the SDRAM using auto precharge mode. To avoid excessive memory stalls due to SDRAM bank precharge and activation latencies, memory accesses are evenly distributed across all banks to maximize the bank interleaving effect. A more detailed description can be found in [13] and [14].

Figure 7 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 incoming image (n) needs to be buffered. Each of the two ME engines contains it's own frame buffer to store four full-size images of up to 4Kx4K accessed via it's respective CMC0 or CMC1 (Figure 5). 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 it's own memory controller (CMC2 on Figure 5). 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 that is below the practical limit of 14 Gbit/second.

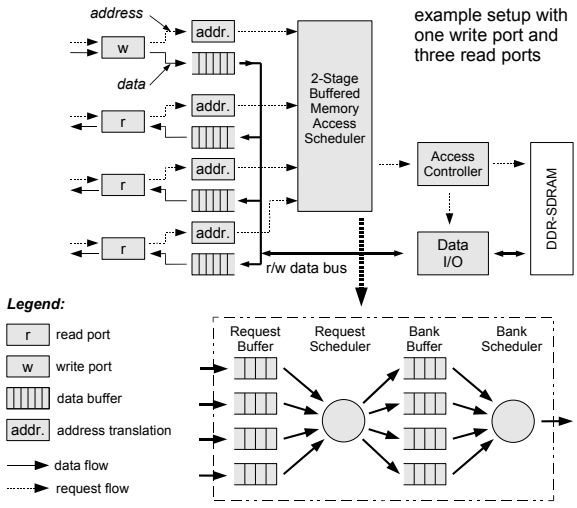

Figure 8. high bandwidth SDRAM controller

4.5. 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 5), 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.

5. Conclusion

A record performance reconfigurable HW/SW platform for digital film applications was presented. The combination 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. The FPGA resource utilization is very satisfactory including memory and routing resources. The FlexWAFE architecture is part of a larger project towards an extendible PCIexpress based real time film processing system.

References

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

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

[3] http://www.pcisig.com/home/.

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

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

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

[7] J. H. Ahn, W. J. Dally, B. Khailany, U. J. Kapasi, and A. Das. Evaluating the Imagine Stream Architecture. *SIGARCH Comput. Archit. News*, 32(2):14, 2004.

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

[9] 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.

[10] A. do Carmo Lucas and R. Ernst. An Image Processor for Digital Film. In *IEEE Application-Specific Systems, Architectures, and Processors (ASAP 2005)*, pages 219–224. IEEE computer Society, 2005.

[11] 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.

[12] S. Eichner, G. Scheller, and U. Wessely. Wavelet-temporal basierende Rauschreduktion von Filmsequenzen. In *21. Jahrestagung der FKTG, Koblenz, Germany*, May 2004.

[13] S. Heithecker, A. do Carmo Lucas, and R. Ernst. A Mixed QoS SDRAM Controller for FPGA-Based High-End Image Processing. In *Proceedings of the 2003 IEEE Workshop for Signal Processing Systems*, 2003.

[14] S. Heithecker and R. Ernst. Traffic Shaping for an FPGA based SDRAM Controller with Complex QoS Requirements. In *Design Automation Conference (DAC)*, page 34.5. ACM, 2005.

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

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

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

[18] 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*, volume 11, pages 1246–1262, 2001.