An Image Processor for Digital Film Processing

Amilcar do Carmo Lucas, Rolf Ernst
Institute of Computer and Communication Network Engineering
Technical University Braunschweig, Germany
{lucas | ernst}@ida.ing.tu-bs.de

Abstract

This paper presents an FPGA based hardware architecture, named FlexWAFE, for high resolution, high troughput real-time digital film processing. Complex algorithms require several hundred arithmetic operations per pixel which is beyond the scope of current DSP processors. To simplify programming and yet achieve high clock rates, the architecture combines component configuration with weak programmability. It alleviates the memory bottleneck by an efficient use of internal memory blocks and a multi-stream SDRAM memory scheduler with tightly bounded latency. Several examples of a Discrete Wavelet Transform and a complex noise reducer demonstrate the architecture efficiency.

Keywords: weak-programing, stream-based architechture, digital film, reconfigurable, FPGA

1. Introduction

In high-end applications, such as HDTV or electronic motion pictures, bandwidth is critical. High resolution applications, widely used in motion picture and advertising industries, require up to 2K resolutions (2048x1556 pixels per frame at 30 bit/pixel and 24 pictures/s resulting in an image size of 91 MBytes and a data rate of 274 MBytes per second) resolutions that translate to a data-rate of 2.1 Gbit per second and channel [1], [2]. The very high end of digital cinema (D-Cinema) applications have grown in importance over the last couple of years with a brilliant resolution of 4K per frame [3] and even higher resolutions are expected in the future. Real-time processing, such as filtering for up- and down-scaling, color keying, compression or trick effects at this data rate and precision is far beyond the scope of today's workstations and DSP processors.

The market volume for such systems is very small, so ASICs are not economically viable and, therefore, not an option.

In the FlexFilm project [4], a cooperation between two universities and industry, a multi-board, extendible FPGA based system is under development that will implement even complex video algorithms, such as multi-frame (3D) noise reduction requiring several 100 arithmetic operations per pixel. Main challenges of this architecture are the large required memory space holding several consecutive frames that is realized with DDR-SDRAM using high throughput flow controlled memory schedulers [5], and the high communication bandwidth provided by a PCIexpress [6] communication network.

Only large FPGAs, such as the Xilinx *Virtex-II* and *IV* families [7] provide sufficient computing resources, but the relatively small internal memories create a serious memory bandwidth problem. The solution is a configurable combination of parameterized and weakly programmable local memories with controllers (LMC) that support sophisticated memory pattern transformations and of data stream processing units (DPUs). Their sizes fit the typical FPGA blocks and form macro blocks that can be easily laid out reaching clock rates of 125 MHz on a Xilinx *Virtex-II*. These blocks, their configuration and their programming are the topics of this paper.

Section 2 contains an overview of the *FlexWAFE* (Flexible Weakly programmable Advanced Film Engine) architechture. A more detailed explanation of its building blocks is given in section 3. Section 4 describes an application example and section 5 concludes the paper.

1.1. Related Work

The Imagine stream processor [8] 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. [9] is focused on application level stream optimizations and ignore archi-

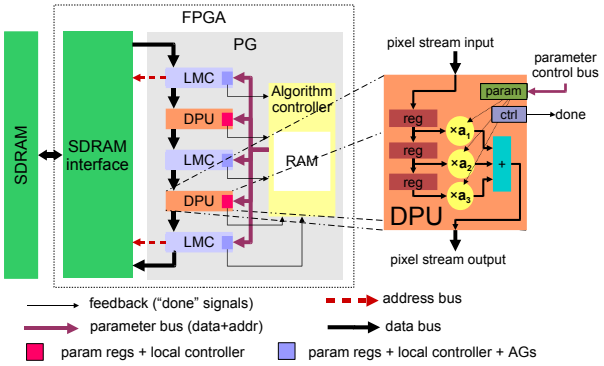


Figure 1. FlexWAFE architecture (left) with a FIR filter DPU example (right)

tecture 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* [7]). Together with the huge amount of distributed logic in the chip (up to 100K CLBs [7]) it is possible to build circuits that compete with ASICs regarding performance, but have the advantage of their configurability and therefore reuse.

2. Architecture overview

The architecture consists of three kinds of components: datastreams communicators, datastreams processors and image algorithm dependent global control. Datastreams communicators will be referenced as *Local Memories with Controllers* (LMCs), datastreams processors as *Data Processing Units* (DPUs) and image algorithm dependent global control as *Algorithm Controller* (AC) troughout this paper. These units can be combined into a group that will perform a specific image processing algorithm or part of it. Such group is called Processing Group (PG) and figure 1 shows a simple example.

DPU and LMC components are parameterized and optimized for speed. Besides parameters that are configured at design time, there are programmable parameters that are downloaded at run-time to enable processing in multiple passes and simplify component structure. So, instead of introducing complex dynamic reconfiguration, weak programmability is used. Such weak programmability in combination with design time component configuration is heavily used in SoC (System-on-Chip) platforms, such as VIPER [10]. In case of the FlexWAFE architecture, the approach requires little area overhead, as the experiments show.

Weak programming is controlled by a third component type, the central Algorithm Controller, AC. The AC has a micro program store that contains all the run-time code for weakly programmable DPUs and LMCs. The AC uses few component parameter registers to hide programming latency. Loading parameters for the next operations occurs in parallel to the current processing step. Since weak programming requires little information, a small shared bus is sufficient for programming all components. This small global control bus adds little to the global wire cost and can be routed automatically. The local-global controller synchronization uses simple handshaking. It is implemented with a point-to-point connection instead of using the already existing bus. This avoids the latency penalty of a bus while not adding to the routing complexity because of a single bit signal per component.

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.

3. FlexWAFE architecture building blocks

As shown in the previous section, the FlexWAFE architecture is composed by three blocks. Namely, LMC, DPU and AC. This section explains each block in detail.

3.1. LMCs - Local memory controllers

Managing data efficiently is an important issue, not only because big images require high data volume but, also, because image processing algorithms either have a tendency to not reuse data or to reuse data only at a very small scale (i.e. image extension on image boundaries for filtering operations). In the FlexWAFE architecture data is transfered, reorganized and stored using LMCs. To do so, these blocks have a dual ported local memory that is read and written using independent address generators (AG), an ingress AG for the incoming data and an egress AG for the outgoing data as seen on the left LMC in figure 2. These AGs are based on the ones described by Hartenstein et al. in [11] and generate the address sequences using only six parameters. This set of parameters, although small, provides a large number of patterns, allowing the LMC to rotate, flip, decimate, extract a ROI (region-of-interest), scan in different zig-zag patterns, act as a FIFO or implement a combination of any of these operations. Later in this paper, it will be explained how these six parameters work.

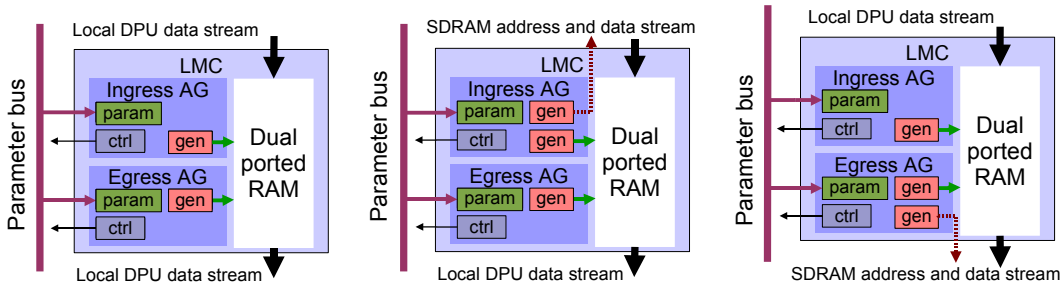


Figure 2. Local Memory Controller (LMC) building blocks

The LMC units can read and write data streams from/to either external memory or from/to local memory. There is an extra address generator to calculate the addresses for the external memory as shown in figure 2 (middle and right blocks). External SDRAM access with small speed penalty requires a specialized bounded delay time memory controller [12], [5] as shown in figure 1. The controller is configurable to different memory sizes, word lengths, and number of parallel access streams. It is bust oriented and the LMCs are responsible for the data de-bursting for read operations and data bursting for write operations.

Image processing streams are known in advance in most applications considered i.e. FIR, IIR and median filtering, exhaustive motion search, color correction, color space transformation, DCT, DWT ... etc, this allows to prefetch the data from external memory and, therefore, effectively hide the latency of most off-chip accesses. In the cases where there is no need to reorder the datastreams, a FIFO can be used instead of the dual ported RAM and its two address generators. The FIFO is based on the one described by Cummings et al. in [13], but gray counters where used instead of binary counters to improve performance. It can be made asynchronous allowing the decoupling of the internal DPU frequency from the external SDRAM bus frequency.

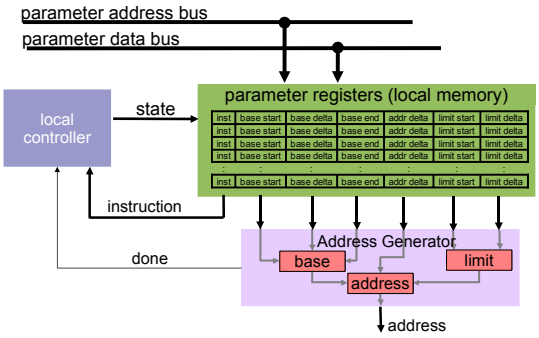


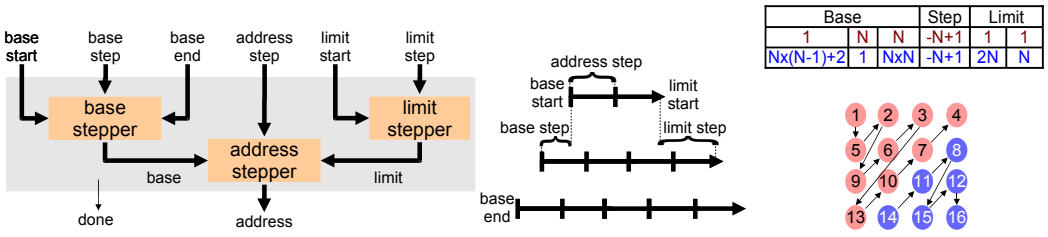
Figure 3. Address generator and local control in detail

In general, the LMCs use different access patterns on their egress and ingress AGs using the local memory as a buffer. The six address parameters operate under the address slider principle [11] depicted in figure 4(a). The *address stepper* generates an address sequence between *base* and *limit* in *address step* steps. The *base stepper* and the *limit stepper* work in an analogous manner. This simple structure allows complicated access patterns, an example of such is the zig-zag stream shown in figure 4(b). To achieve this pattern the ingress AG writes the incoming pixel steam in order (1..16) to the local memory but the egress AG uses the two parameter sets in the table of figure 4(b). Once it finishes the first parameter set (for the upper triangle) it immediately starts the second (for the lower triangle) because the AG can switch the parameters between sequences without introducing idle cycles. As explained in section 2, these address parameters are kept in a small local parameter memory and adddress unit control is also local as seen in figure 3. In all the examples which we investigated, only few register sets are needed per LMC with an address word length of 10 bit or less that can be implemented in CLB registers. Hence, each LMC typically requires only one FPGA memory block.

The parameters are sequentially transmitted in advance over the low bandwidth control bus from the AC. Each parameter register has its own unique address. In effect, this local memory provides shadow registers for the working parameters that currently control the address generator. Once an address sequence is finished, the local controller sends an acknowledge signal to the AC and that feeds the next set of parameters to the address generator. This way, the AG can work uninterruptedly and the AC can control and synchronize multiple address and operation sequences.

3.2. DPUs - Data processing units

DPUs process the streams provided by the LMCs. DPUs can have multiple data input and output ports. This covers operations with two operands or more (i.e. addition) and/or two or more results (i.e. division with quotient and rest). For the applications considered so far, they have one or two



(a) slider principle

(b) Zig-zag address sequence example for n=4

Figure 4. Address generator based on the slider principle

data inputs and one or two data outputs. The complexity of the operations they perform can vary, a simple example is a truncation DPU, a complex one is a decimated vertical filter with internal SPE [14] calculation (see section 4). The simpler DPUs do not need an interface to the AC because they are not weakly programmable (i.e. an adder) more complex DPUs use the same shadow memory technique as the LMCs (see section 3.1) to decouple their local controller from the AC. The right side of figure 1 presents an example of a three tap FIR filter DPU in which the filter coefficients are programmable.

3.3. AC - Algorithm controller

In the current implementation, the AC consists of a micro code memory and a simple micro code sequencer implemented as an FSM (finite state machine). Simple sequencing is possible since all operation sequences need a fixed execution time. Synchronization is used to account for the buffered DDR-SDRAM memory access. The AC reacts to the DPUs and LMCs via the point-to-point connections and controls them trough the shared parameter by programming their local shadow registers as seen in figure 5.

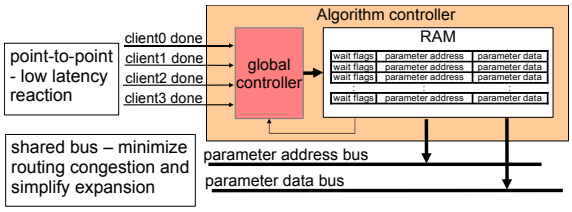


Figure 5. Algorithm controller block diagram

4. DWT - An application example

The discrete wavelet transform (DWT) allows one to transform a signal into a space where the base functions are

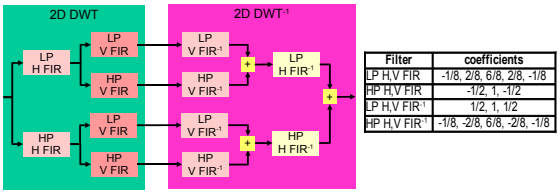
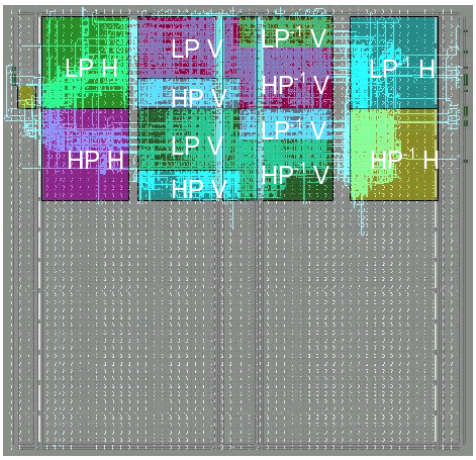


Figure 6. One DWT level (analysis and synthesis)

wavelets [15], 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 as seen on figure 6.

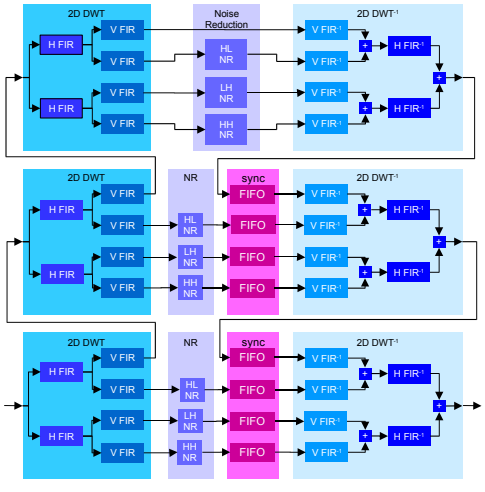
The implementation of the filters uses polyphase decomposition (horizontal) and coefficient folding (vertical) and is based on the work of Po-Cheng Wu et al. [16]. To maximize throughput the transformation operates line-by-line instead of level-by-level [17]. This allows for all DPUs to operate in parallel (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 [14] 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. The simple system on figure 6 executes 18 add operations on the direct DWT, 22 add operations on the inverse DWT



Resource	Usage	Percentage
RAMB	12 out of 56	21%
Slices	1,650 out of 10,752	15%
Flip-Flops	1,149 out of 21,504	5%

- 10 bits per pixel
- 2 pixels/clock cycle
- 91 additions of ten or more bits/clock cycle
- three lines latency between input and output
- no external memory required
- configurable up to 2048x2048 pixel/image
- up to 125 Mhz
- 59 fps @ 2048x2048, 10bpp

Figure 7. Mapping and resource usage in a Xilinx XC2V2000 device



Resource	Usage	Percentage
RAMB	44 out of 56	78%
Slices	7,184 out of 10,752	66%
Flip-Flops	4,729 out of 21,504	21%

- 12 bits per pixel
- 2 pixels/clock cycle
- no external memory required
- configurable up to 2048x2048 pixel/image
- up to 100 Mhz
- 41 fps @ 2048x2048, 12bpp

Figure 8. A Noise reduction application using DWT

and 57 extra add operations to support the SPE, all between 10 and 24 bits wide. It processes two pixels per clock cycle and runs at 125 MHz on a Xilinx *Virtex-II* XC2V2000 chip [7]. The mapping and the resource usage for images up to 2048x2048 pixels, 10 bits per color component (only one component was considered, also known as grayscale images) can be seen on figure 7. This system achieves 59 frames per second with images of 2048x2048 pixels. Based on these DPU blocks a noise reduction algorithm that operates on the wavelet space was developed [18]. This algorithm requires three levels of decomposition, therefore three of the blocks described in figure 6 and 7 were cascaded and the noise reduction DPUs added. To compensate the latency of the higher decomposition levels, FIFO based LMCs were used. The resulting system is depicted in figure 8.

5. Conclusion

The FlexWAFE architecture consists of chains of memory transformations and data paths. A combination of component design time configuration and weak programmability with small local controllers and hidden programming latency requires few basic components yet reaches high clock speed even for complex applications. 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.

6. Acknowledgements

This work was partly founded by German BMBF and Thomson - Grass Valley

References

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

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

[3] http://www.thomsonbroadcast.com.

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

[5] Sven Heithecker and Rolf Ernst. Traffic Shaping for an FPGA based SDRAM Controller with Complex QoS Requirements. In *Design Automation Conference (DAC)*, page (to appear). ACM, 2005.

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

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

[8] 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.

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

[10] 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, October 2001.

[11] R. Hartenstein, A. Hirschbiel, and M. Weber. MOM - Map Oriented Machine. In *Proceedings of the International Workshop on Hardware Accelerators*, 1987.

[12] Sven Heithecker, Amilcar do Carmo Lucas, and Rolf 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.

[13] Clifford E. Cummings and Peter Alfke. Simulation and synthesis Techniques for Asyncronous FIFO Design with Asyncromous Pointer Comparations. In *Synopsys Users Group - San Jose*, 2002.

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

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

[16] Po-Cheng Wu and Liang-Gee Chen. An Efficient Architecture for Two-Dimensional Discrete Wavelet Transform. *IEEE Transactions on circuits and systems for video technology*, 11(4), 2001.

[17] 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.

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