Implementation of a scalable MPEG-4 wavelet-based visual texture compression system

L. Nachtergaele, B. Vanhoof, M. Peón, G. Lafruit, J. Bormans, I. Bolsens

IMEC, Kapeldreef 75, B3000 Leuven, Belgium, Tel. +32/(0)16/281.201

{nachterg, vanhoofb, peon, lafruit, bormans, bolsens}@imec.be

ABSTRACT

The realization of new MPEG-4 functionality, applicable to 3D graphics texture compression and image database access over the Internet, is demonstrated in a PC-based compression system. Applying our system-level design methodologies effectively removes all implementation bottlenecks. A first-of-a-kind ASIC, called Ozone, accelerates the Embedded Zero Tree based encoding and is capable of compressing 30 color CIF images per second.

1. INTRODUCTION

MPEG-4, a standard for multimedia communication, has reached International Standard status since the beginning of 1999 [1]. This newly adopted standard opens high aims in the already very important multi-media market. Consider for example the dissemination via the Internet of the popular "Episode I" trailer of Star Wars [2]. On this web site, ten different versions of the 2.2 minutes trailer movie are offered with file sizes ranging from 5 Mbytes up to 25 Mbytes. Each of the ten file formats presents a trade-off in download time and required computational performance of the decoder. Instead, MPEG-4 aims at providing a solution that scales the communication of the multimedia content over heterogeneous access networks and decoding facilities. The ultimate goal is to reach a "create once, decode everywhere" situation. For that, the key capability that is missing in current audio-visual coding systems is scalability. Employing a wavelet-based compression technique allows for scalable MPEG-4 visual texture coding, the subject of this paper.

When rendering a 3D scene, objects that are near to the horizon only need very coarse textures of perhaps just a few pixels. As objects become closer to the position of the viewer, more detailed textures are required to obtain natural looking scenes. Current 3D applications use de-facto 3D graphics API's such as OpenGL and Direct3D. Hereby the uncompressed textures are read into memory first. Only then, downsampling and filtering generate the required successive lower resolution versions. This technique works well in a PC environment where applications are mostly distributed on CD-ROM. However, imagine for a moment that the 3D content is transmitted via the Internet. It's clear that data volumes typically found on CD-ROM are way too large for this. Hence, compression of the 3D scenes and associated textures is required. The purpose of MPEG-4 wavelet based visual texture coding is exactly meant for that. This paper describes a first-of-a-kind realization [6] that demonstrates the key functionality for visual texture coding.

DAC 99, New Orleans, Louisiana

2. WAVELET-BASED COMPRESSION: IMPLEMENTATION CHALLENGES

Three major algorithmic parts can be distinguished in the MPEG-4 compression algorithm for visual textures (see Figure 1). First, applying the wavelet transform decorrelates the input image. Secondly, trees grouping wavelet coefficients across several subbands are extracted. The Embedded Zero Tree (EZT) algorithm [12] compresses these trees by identifying trees consisting solely of zeroes. One can consider EZT as an intelligent run-length coder that operates on several resolutions of the image at the time. Finally, an arithmetic encoder removes redundancy from the sequence of EZT symbols. The shrinking size of the arrows in Figure 1 between the three algorithmic blocks suggests that the data object size whereupon they operate reduce from image frames, over trees of zeroes to bits.

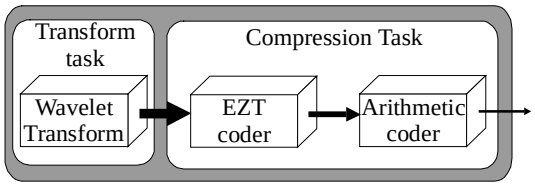

Figure 1: A wavelet-based visual texture compression system contains three major parts

In this paper, the implementation of an end-to-end compression/decompression system implementing this complex compression algorithm is described. The implementation of each identified algorithm represents a challenge on its own. The wavelet transform exhibits a high bandwidth to memory. The EZT coding is a highly data dependent algorithm that is defined in terms of while-loops and dynamic lists. These two concepts prevent a straightforward hardware implementation. And last but not least, the adaptive arithmetic encoder is the end of the processing pipeline that has to cope with all symbols produced. For example, suppose a setup of the compression engine parameters for a compression factor of ten (this corresponds to visually lossless compression). In the case of real-time color CIF images, the arithmetic encoder must then be capable to process (352 columns x 288 lines x 12 bits/pixel x 30 frames/second)/10 = 3.6 10^6 bits/second. This requirement can solely be met by a high throughput application specific datapath. The three implementation problems - high

memory bandwidth for the wavelet transform, a complex irregular highly datadepent EZT coder and the non pipelinable adaptive arithmetic coder - make the implementation of such a compression chain a real implementation challenge.
Traditionally, in order to tackle the design time problem, a divide and conquer approach is followed. The system design would then be split into three separate "processor" designs that each tackle the aforementioned implementation challenges. The result of the independently locally optimized implementations is then integrated as they become available. For data intensive applications, like the compression chain under consideration, huge buffers in-between the "processors" become unavoidable for this integration. Those buffers can jeopardize the goals of the system since they may represent a large part of the total system cost budget. In the worst case, a redesign of one or more "processors" is needed, leading to unexpected delay in the system design. Such delay may be dramatic if they cause a miss of the market opportunity window. In the next section, we present a global system-level design methodology that allows for a global system optimization.

3. GLOBAL SYSTEM-LEVEL DESIGN METHODOLOGY

The applied multi-stage system-level design methodology [3] consists out of two major stages called the task level Data Transfer and Storage exploration (DTSE) [4] and the processor level DTSE [4], as shown in Figure 2.

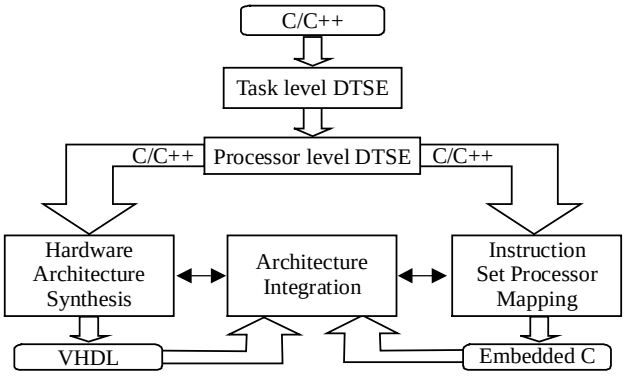

Figure 2: Applied system-level design methodology

Tasks are subsystems that can operate concurrently. The criterion to decide what functionality is considered in one task, is based on a data-dependency analysis. The main reason for splitting the DTSE in two stages is the significant reduction of the system design complexity. This reduction can be achieved without sacrificing system cost in data transfer and storage overhead.
In the still texture compression chain, clusters of data dependencies are found in the wavelet transform on one hand and in the combination of the EZT and adaptive arithmetic encoder on the other hand. Hence, the global system is partitioned into two main tasks as depicted in Figure 1. The resulting two tasks are called the transform task and the compression task respectively. After task partitioning, buffer sizes in-between tasks must be minimized. This is the objective in the so-called "Inter-task" or "task-level" optimizations of the next section.

4. LEVEL OPTIMIZATIONS: THE LOCAL WAVELET TRANSFORM

In order to perform the wavelet transformation and the EZT coding simultaneously, a ping-pong buffer must be provided. While a wavelet-transformed image is produced, the EZT compresses the previous transformed image as shown in Figure 3. As the wavelet-transformed image has the same size as the input image, the size of the ping-pong buffer is twice the input size. For example, with an input image of 1024×1024, this results in 2 mega sample buffer.

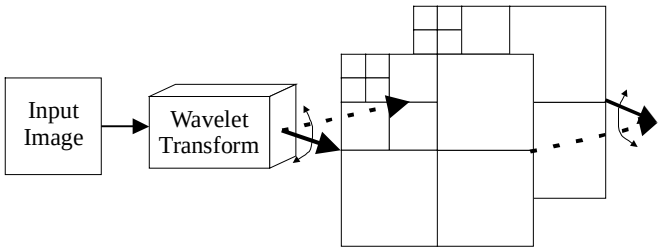

Figure 3: Between the transform and the compression tasks, a buffer of twice the size of the input image is required

This huge inter-task buffer is clearly not desirable in a high-throughput processing pipeline. Therefore it is important to match the data production of the wavelet transform task to the data consumption in the compression task.
Vishwanath [15] was able to develop a single systolic array algorithm, called Recursive Pyramid Algorithm (RPA), for computing the wavelet transform that requires a minimal amount of data-word storage. A modified RPA (MRPA) [5] allowed to reduce the latency of the overall wavelet transform processing by "filling the gaps" in the execution schedule. Although the (M)RPA technique effectively reduces the required intermediate memory, still a high number of transfers to and from the memory are needed. In [8] the optimized result of an intra-task DTSE, called the Local Wavelet Transform (LWT) is described. The LWT drastically reduces the required memory access cost and its associated bandwidth. This is very important in order to achieve low power operation. Furthermore, the LWT needs a slightly smaller memory compared to the MRPA technique. In [9], a memory efficient architecture for implementation of the LWT is presented. In Table 1, the memory size and the number of memory accesses needed when performing a 6 level discrete wavelet transform with a 9/7-taps wavelet filter are given.

Image Size	Method	Memory Size (#samples)	Memory Access (#reads/sample)
1024×1024	Mallat	1 M	2.67
	RPA	327 k	11.3
	LWT	281 k	3.07
4096×4096	Mallat	16 M	2.67
	RPA	1.3 M	11.3
	LWT	1.1M	3.14

Table 1: Memory size and access for Mallat, RPA and LWT for a 9/7-tap, 6 level wavelet transform

After the buffer in-between the transform and the compression tasks is largely removed, the tasks on their own are still to be optimized. This is the next stage in our system-level design methodology and is called "intra-task" or "processor-level" optimization and is the subject of the next section.

5. PROCESSOR-LEVEL OPTIMIZATIONS

The two tasks resulting from the partitioning operate on different basic datatypes. The transform task is composed out of filter and downsampling operations that process samples. Hence, a wavelet transform is well suited to be implemented on a DSP processor or a processor with media extensions. In contrast, the compression

task works on symbols and bits. Bit-oriented algorithms are typically less suited for implementation on processors. Therefore, we have opted for an application specific solution (ASIC). We now discuss efficient implementations by applying the processor-level optimizations on the transform and the compression task.

5.1 Discrete Wavelet Transform Implementations

A good survey of research until 1996 concerning implementation issues of the discrete wavelet transform is found in [10]. The most recent wavelet zero tree processor, presented in [7], implements a wavelet codec with a four taps Daubechies orthogonal wavelet filters and compresses the wavelet coefficients by applying a single quantization coding scheme that generates zero tree symbols in one single wavelet band. The latter is not conformant to Shapiro's [12] inter-band Zero-Tree coding.

The wavelet compression system used in this paper differs in two fundamental points compared with the above mentioned implementations. A first significant difference is in the way that the wavelet coefficients are calculated. In [13] the Lifting scheme is proposed. This technique reduces the number of multiply-accumulate operations, but a far more important feature of the Lifting scheme is that the wavelet transform becomes reversible and can be realized in hardware using integer arithmetic operators. Hence unlike all above surveyed solutions, our system also allows for a lossless compression. Although the reduction of the processing requirements is a nice property of the Lifting scheme, we have explained in the previous section that the main implementation bottleneck is the memory architecture and not the arithmetic related aspects. The second major difference is the used compression technique. The proposed compression task implements EZT coding corresponding to the bi-level quantization in MPEG-4 standard. In contrast with the above surveyed architectures, this EZT coding also exploits inter-subband correlation.

At the time the chip for the compression task became available (Section 5.2), no functional hardware for the transform task was ready. Therefore, we resorted to an earlier started effort in speeding up a Pentium based implementation and developed an MMX optimized lossless wavelet transform, resulting from a constrained processor-level optimization. This optimized wavelet transform performs 30 wavelet transforms in a second on a color CIF image when executed on a Pentium II PC running at 350 MHz. This illustrates how the inter-task DTSE is independent from the implementation style of the tasks. In this case, we have demonstrated how the multi-stage system design methodology can cope with real-life situations such as fast time-to-market requirements and the corresponding constrained design time.

5.2 "Ozone": ASIC for Embedded Zero Tree and Adaptive Arithmetic Encoding

In this section, processor-level implementation issues of the compression task are discussed. Next, we describe the programmability of the Ozone, followed by an architecture description. In the last subsection, a summary is given of the design tools used for designing the Ozone.

5.2.1 EZT Coder

The coding scheme we present in the Ozone exploits the correlation between the wavelet coefficients in different bands, corresponding to the bi-level quantization scheme in MPEG-4. A general EZT coding scheme is proposed in [12]. To obtain a memory efficient implementation of the EZT coder, 5 transformations to the original algorithm were applied of both algorithmic nature and partly along the approach formalized in [4]. The memory size is reduced with a factor 450 for a 5-level transformed CIF image, the number of accesses with an average factor of 4.8. A detailed description of the optimizations performed on the EZT encoder can be found in [14].

5.2.2 Adaptive Arithmetic Coder

The adaptive arithmetic coder algorithm [10], recursively compresses messages using the interval [0,MaxRange]. The current interval is divided according to the probability distribution of the past symbols in the input string. With the current symbol, one partition is selected as the new interval. The application of three transformations resulted in an efficient hardware implementation. Especially the introduction of an additional Huffman coder, transforming the 4-symbol alphabet into a binary alphabet, has a major impact on the hardware implementation. An in-depth discussion of the arithmetic coder optimizations can be found in [10].

5.2.3 Ozone Programmability

In order to allow a flexible compression operation, the Ozone is made programmable:

1. The number of wavelet transformation levels can be varied between 2 and 5.
2. The image size should be an integer multiple of 2 to the power of the number of wavelet transformation levels. The maximum image size is limited to 1024×1024.
3. The quantization levels, programmable for each wavelet subimage individually, allow fine-tuning of the compression performance.
4. As the statistics for the 4-symbol dominant pass depend on the image content and the compression settings, an efficient translation of the 4-symbol alphabet to the binary alphabet can only be achieved with a programmable Huffman table.
5. A user-programmable header of 16 bytes, inserted in the compressed output stream, allows the transmission of compression settings, image identification, etc.

5.2.4 Ozone Architecture

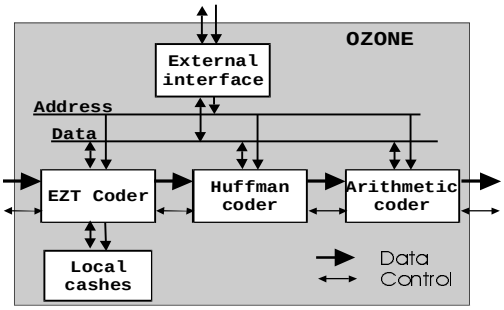

Figure 4: Ozone architecture

The architecture of the Ozone is shown in Figure 4. A dynamic dataflow transfer methodology was selected to perform the communication between the different hardware blocks. The proposed architecture supports parallel operation of Ozones. By carefully programming the quantization levels, each of the Ozones compresses a set of bitplanes. In case of an equal distribution of the load over the Ozones, the throughput increases with a factor equal to the number of parallel Ozones.

5.2.5 Design Methodology

The C++ based hardware design philosophy called OCAPI [11] was used to design the Ozone. The fast high level modeling and simulation of the hardware implementation, allows an extended design space exploration. In the end, the optimal solution is automatically translated from C++ to VHDL. The OCAPI design flow allows an automatic verification of the VHDL description of each

block, followed by an integration test. The SYNOPSYS toolkit has been used to synthesize the VHDL descriptions from within the vendor design environment. Place & route and clock-tree synthesis were performed using Avant!.

6. DEMONSTRATION SETUP

An Ozone-evaluation board (Figure 5) was developed as an add-on card for an FPGA-based PC-board. Using this board, extensive testing of the Ozone was performed. Functionally the board contains an Ozone, 32 MHz clock generation circuitry and FIFO memories allowing burst type communication. Additionally, DC level shifters are necessary for interfacing to the 3.3V Ozone ASIC and the 5V PC power supply. Three separate voltage supplies are foreseen: one for the 5V level shifters and two 3.3V power regions, one for the Ozone and one for the FIFO memories and the clock generation, allowing accurate power dissipation measurements.

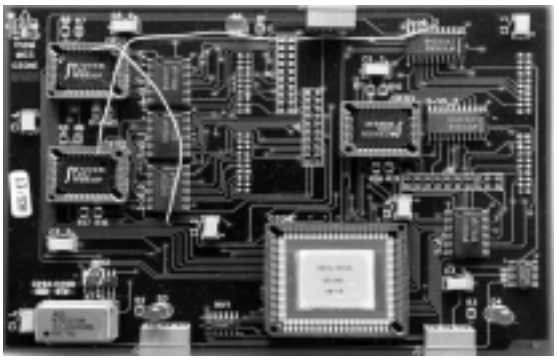

Figure 5: The Ozone evaluation board

A PC-based demonstrator set-up, implementing the complete compression and decompression chain, was build around this Ozone-evaluation board. An average power dissipation of 100 mW was measured for the complete 3.3V power region. This dissipation is distributed evenly over the Ozone and FIFO voltage supply regions. Additionally, the 5V interfacing to the PC dissipates 105 mW.

6. CONCLUSION

A full end-to-end demonstration of the wavelet based compression chain, the key functionality of MPEG-4 visual texture coding, has been presented. According to our multi-stage system-level design methodology, global as well as local optimizations were applied to tackle three major implementation bottlenecks. First, the local wavelet transform, based on a reversible lifting scheme, reduces the memory requirements with an order of magnitude. Second, an application specific implementation of the Embedded Zero Tree coding algorithm, originally defined in terms of recursive while-loops and dynamic lists, has been developed. Finally, an architecture for a non-pipelineable adaptive arithmetic coder, capable of processing 10 Msymbols/second at a clock speed of 32 MHz, has been designed using the OCAPI C++-based hardware design environment. The Ozone ASIC, consisting of the EZT coder and the adaptive arithmetic coder, is capable of compressing 30 color CIF images per second. The Ozone ASIC functionality was successfully demonstrated in a PC-based demonstrator set-up, implementing the complete compression-decompression chain. The Ozone evaluation board dissipates 205 mW.

7. ACKNOWLEDGMENTS

The authors would like to thank following colleagues for their valued contribution: T. Gijbels, P. Vos, B. Debaillie and A. Van Loenen and R. Braspenning, both students from Delft University, The Netherlands. The Ozone algorithm was developed under the ESA Scades-3 project (ESTEC/contract Nr.10208/92/NL/FM).

REFERENCES

[1] ISO/IEC JTC1/SC29/WG11, Coding of audio-visual objects, ISO/IEC 14496, ‘98.
[2] “Episode I trailer movie”, http://www.starwars.com
[3] Catthoor F., et. al., “Proposal for unified system design meta flow in task-level and instruction-level design technology research for multi-media applications”, ISSS'98, Hsinchu, Taiwan, December 1998.
[4] Catthoor F., et. al., “Custom Memory Management Methodology - Exploration of Memory Organisation for Embedded Multimedia System Design”, Kluwer Academic Publishers, Boston, ‘98.
[5] Chakrabarti C., et. al., “Architectures for Wavelet Transforms: A Survey”, Journal of VLSI Signal Processing Systems for Signal Image and Video Technology, Vol. 14, No. 2, November ’96, 171-192.
[6] Clarke P., “MPEG-4 project in Europe achieves wavelet silicon”, EE Times, 28 November ‘98, http://www.eetimes.com/story/OEG19981125S0008.
[7] Knowles G., “A single chip wavelet zero-tree processor for video compression and decompression”, DATE ’98, February ’98, 61-65.
[8] Lafruit G., et. al., “Optimal memory organisation for scalable texture codecs in MPEG-4”, IEEE Tr. on Circuits and Systems for Video Technologies, in press.
[9] Lafruit G., et. al., "The Local Wavelet Transform: a memory-efficient, high-speed architecture for a Region-Oriented ZeroTree coder," Journal of Integrated Computer-Aided Engineering, ‘99, in press.
[10] R. Lang, “Parallel VLSI architectures for one-, two-, and tree-dimensional discrete wavelet transforms”, PhD thesis, Department of Electrical and Computer Engineering, The University of Newcastle New South Wales, 2308 Australia, March 1996.
[10] Peón M., et. al., “Design of an arithmetic coder for a hardware wavelet compression engine”, IEEE Signal Processing Symposium, March 1998, Leuven, Belgium, 151-154.
[11] Schaumont P., et. al., “A Programming Environment for the Design of Complex High Speed ASICs”, DAC, June ’98, 315-320.
[12] Shapiro J.M., “Embedded image coding using the zerotrees of wavelet coefficients”, IEEE Tr. on Image Processing, Vol. 41, No. 12, , Dec. ’93, 3445-3462.
[13] Sweldens W., “The Lifting Scheme: A new Philosophy in Biorthogonal Wavelets constructions,” Proc. of the SPIE conference, Vol. 2569, 1995, 68-79.
[14] Vanhoof B., et. al., “A Scalable Architecture for MPEG-4 Embedded Zero Tree Coding”, CICC’99, in press.
[15] Vishwanath M., et. al., “VLSI Architectures for the Discrete Wavelet transform”, IEEE Tr. on Circuits and Systems-II, Vol. 42, No. 5,May ’95, 305-316.