A WAVELET-TREE IMAGE CODING SYSTEM WITH EFFICIENT MEMORY UTILIZATION

Y. Andreopoulos[1,2], P. Schelkens[1], N. D. Zervas[2], T. Stouraitis[2], C. E. Goutis[2], J. Cornelis[1]

[1] Vrije Universiteit Brussel/IMEC - Dept. ETRO
Pleinlaan 2 - B-1050 Brussel - Belgium
{yandreop,pschelke,jpcornel}@etro.vub.ac.be

[2] University of Patras - Dept. ECE
VLSI Design Laboratory - Rio 26500 - Greece
{zervas,thanos,goutis}@ee.upatras.gr

ABSTRACT

This paper describes an efficient implementation of an image coding system based on the independent wavelet-tree coding concept. The system consists of a transform and a (de)coding engine that operate in a pipelined fashion. The main focus of this paper will be on the encoding part since, due to the system architecture, the decoder has identical memory utilization. Experimental results prove that the proposed system achieves comparable coding performance to the state-of-the-art, while it localizes the memory accesses to small memory modules and uses minimal computational resources.

1 INTRODUCTION

In recent years, wavelet-based compression techniques have matured into versatile tools for the compression and handling of the multimedia content of information. The inclusion of wavelet transforms into the new multimedia standards for video and image coding, namely MPEG-4 [2] and JPEG 2000 [4], has led to exponentially-increased interest about coding techniques for the wavelet-transform coefficients. The wavelet-based coding techniques adopted by the two standards are the inter-subband, wavelet-tree coding of wavelet coefficients [12] and the intra-subband, wavelet-block coding [14]. In this paper, the main focus will be on the wavelet-tree coding approach, which exploits the parent-children dependencies inherent in the dyadic tree organization of a wavelet transform (Figure 1). Various authors, for example [8] [11] [16], propose efficient implementations to coding the parent-children tree (PCT) relations with Embedded Zerotree Wavelet coding (EZW) [10] and its successor, the Set Partitioning In Hierarchical Trees algorithm (SPIHT) [7].

In this paper, we focus on the implementation aspects of the design of an entire image coding system based on the SPIHT coding. The development of our system is twofold; first an efficient wavelet-transform engine is implemented and then, for the coding engine of the proposed compression system, a new compression method is presented, which is coupled with the wavelet-coefficient production, so as to minimize the overall latency. The produced bit-stream can be transmitted to the decoder after the completion of the processing of all the PCTs (burst mode) or the transmission of the compressed data of each tree can occur during the compression of the next PCT (band bursty). Depending on the chosen mode of operation, different memory needs occur for the compression and decompression engines. It must be noted however that the memory needs for the coding and decoding system are perfectly equivalent, both for the transform and for the (de)coding engine. In addition, the memory accesses for the two systems follow approximately the same schedule.

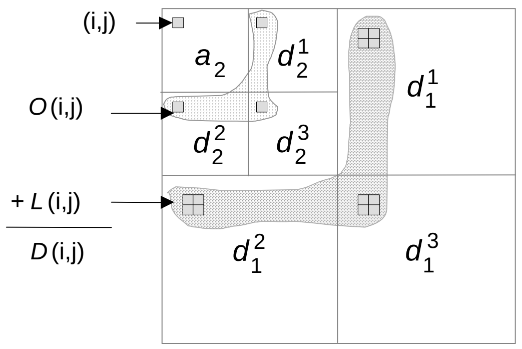

Figure 1 — Dyadic-tree decomposition in 2 levels with an example of a wavelet tree. The definitions from [7] are shown.

Summarized, this paper presents the following novelties:

- For the transform engine, based on the theoretical concepts of [5] and [6], an *implementation* is proposed, which, apart from presenting for the first time actual results with respect to memory size and accesses, gives an illuminating description of the initialization patterns so that the transform engine can independently produce each PCT with a near optimum utilization of the memory and computational resources.
- A pipelined functionality between transform and coding is proposed, *both for the encoding and the decoding*, with the use of a novel compression scheme that is based on the SPIHT coding.

2 THE WAVELET TRANSFORM ENGINE

The implementation presented in this section is based on a parametrical architecture that is capable of independent PCT production with reduced memory and the near minimum number of external (off-chip) accesses [6]. A block-based image traversal is used, originally proposed in [5]. It must be noted that the block-based traversal is not the only traversal algorithm capable of producing independently compressible transform components, such as PCTs. In [5], the authors also present other different traversal algorithms, such as the Recursive Pyramid Algorithm (RPA) [15] that are also known to produce the transform coefficients in a band-bursty manner. In addition, in the block-based coding of JPEG-2000 [3], the authors present a line-based implementation that is capable of coupling the encoder and decoder with delay-line memory components that operate in FIFO manner. In Figure 2 we show graphically the comparisons between the different traversal schedules with respect to memory size and accesses. The numeric values that correspond to Figure 2 can be verified by the theoretical calculations presented in [9]. It can be easily observed that the block-based traversal algorithm that is proposed in [5] achieves good results in terms of memory and especially off-chip memory accesses.

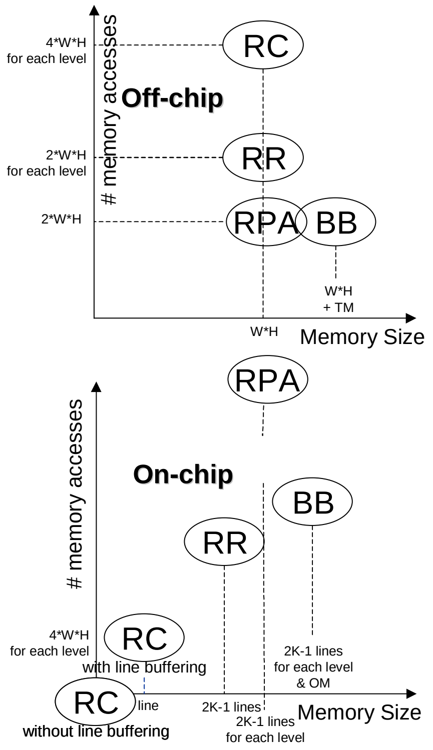

Figure 2 — Indicative graph of off-chip and on-chip memory behavior and requirements for the classical row-column approach (RC) with and without line-buffering, the row-row approach (RR) used in JPEG2000, the recursive pyramid algorithm of Vishwanath (RPA) and the local or block-based implementation (BB). The additional notations used, are explained in Section 2.1. The amount of accesses was obtained assuming a two-level memory hierarchy (registers, on-chip cache, main off-chip memory). The figures were obtained for a lifting scheme implementation, assuming an in-place organization of the wavelet coefficients [13], which explains the increased off-chip memory needs for the BB approach (*WH* factor), where *W* and *H* denote respectively the image width and height.

The wavelet-transform implementation that is presented in this section is parametrical to the number of decomposition levels (*N*) as well as the maximum filter length (2*K*+1), with the only restriction being the assumption of a symmetrical dependence from the image samples, similar to the 9/7 and 5/3 filter pairs used in JPEG 2000 [4]. For every input image, after the completion of the transform, identical results to the classical row-column approach (RC) are produced.

2.1 Description of the Modes of Operation.

The transform production can be divided in three stages. The first stage is the *Initialization Mode*, which deals with initialization phenomena inherent in the mirroring techniques used for the dyadic-tree decomposition. The second and most dominant stage is the *Normal Mode*, where a regular flow of 2^{2N} samples produces 2^{2N} wavelet coefficients and, at each iteration, the transform engine outputs one PCT. The third stage is the *Finalization Mode*, where, by using the residual blocks of input samples, the remaining PCTs are produced.

In the Initialization Mode, the input image is read in sequential non-overlapping blocks of samples in the row direction. An example of the scanning pattern is depicted in Figure 3 for the 5/3 filter pair (*K*=2) for three decomposition levels (*N*=3).

For all the Modes of operation, each block is temporarily stored in a memory component called the *Inter-Pass Memory* (IPM). A pair of low and high-pass filters is applied to the samples contained in the IPM, following the classical row-column fashion. The actual filtering takes place by copying the image samples in FIFO manner from the IPM to the filtering unit, which is called the *Filtering FIFO* (FF), to emphasize the inbuilt functionality. For every new pair of input samples, a pair of low and high-frequency coefficients is produced and stored in-place. In this way, the wavelet-coefficient production proceeds as much as possible for each level. Of course, in order to ensure that the produced results are identical to the row-column implementation of the wavelet transform, during the Initialization Mode, mirroring of the signal edges takes place inside the FF.

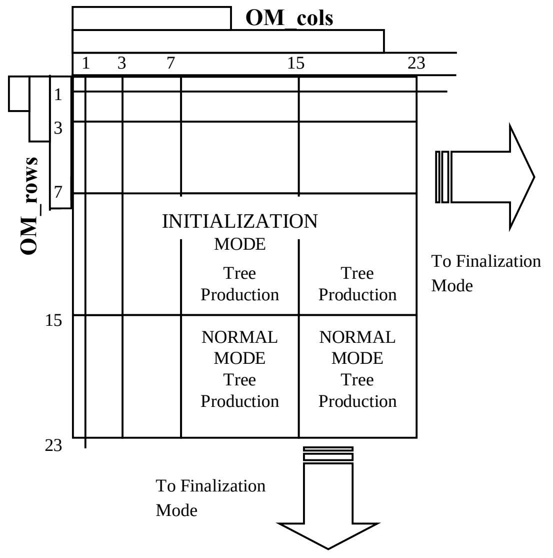

Figure 3 — An example of the proposed initialization pattern.

The intermediate results (low-frequency coefficients) that are produced for the rows and columns of each block are stored in memory components, called the *Overlap Memory for rows* (OM_rows) and *Overlap Memory for columns* (OM_cols), that are depicted in Figure 3. These results are used in the processing of the neighboring blocks and the new intermediate results from each block replace the previous ones that have been used in the current iteration. The high-frequency outputs that are produced during each iteration are pushed into FIFO delay lines, called *Tree Memory* (TM). These delay lines compensate for the skewing in the temporal PCT production with the topological PCT representation, as shown in [5]. This skewing can simply be explained by the fact that no PCTs can be produced, until there exist enough low-frequency coefficients at the highest decomposition level that allow the execution of mirroring and filtering in both directions, rows and columns, so as to produce the quadruplet of coefficients shown in the highest decomposition level of Figure 1. However, until that point, a significant amount of high-frequency coefficients is produced as well and, in order for the first PCT to contain the correct children for each parent, the Tree Memory is utilized for each decomposition level.

2.2 Implementation Benefits.

In Figure 4, the entire architecture used for the transform engine is depicted. It must be noted that for streaming applications, I_Mem, which is the image memory, does not correspond to the entire image but only to one block-scanning line, thus its vertical size is restricted to 2^N samples. In this case, memory reduction *by a factor of magnitude* is achieved in comparison to the classical row-column approach [6], where the entire wavelet-coefficient matrix needs to be buffered.

From the described execution flow, it is clear that for the processing of each block, the image memory (I_Mem) is read, row-column filtering takes place using the depicted architectural components and the outputs are flushed in the Tree Memory. Thus, *the accesses to large memory components*, such as I_Mem and TM, *are restricted to the start and completion of each block-processing iteration*. As a result, if the implementation platform provides enough design-flexibility so that the OM, IPM and FF components are placed internally, *a near optimal minimization of the memory accesses to external (off-chip) memory components is achieved*. This will have a very positive impact on the execution speed as well as on the reduction of the system power-consumption [6].

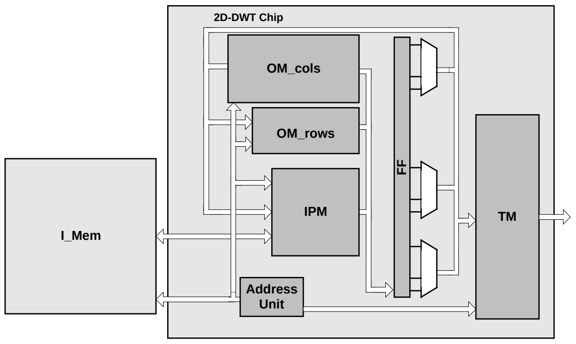

Figure 4 ─ The architecture used for the wavelet-transform production.

3 THE CODING ENGINE

Each PCT that is produced by the transform engine is independently coded by a low-complexity zero-tree algorithm that exploits the parent-children dependencies similar to the SPIHT variations presented in [11]. The main differences lay in the fact that the proposed coding scheme can exploit the pipelined PCT production-consumption and that competitive coding results can be obtained by selecting the truncation points for each PCT coding with a mathematical optimization tool, the Lagrangian multiplier method.

The coding of each PCT occurs in sequential bit-plane passes with a decreasing Quantization Step [7]. For each bit-plane, three sequential sub-passes occur, following a breadth-first traversal and beginning from the highest decomposition level. The partitioning rules are identical to the ones used in the SPIHT algorithm [7], but the linked list functionality is avoided, since the scanning order is fixed. Apart from the memory necessary to store each PCT, two buffers are used that correspond to the status information for each coefficient:

- Buffer of Coefficient Significance (BCS). Holds quaternary values that show if each corresponding coefficient is significant or insignificant for the current Quantization Step.
- Buffer of Subtree Partitioning (BSP). Holds quaternary values that show if the subtree that is rooted to each corresponding coefficient is valid or not, and if true, if it should follow the type A or type B partition rule [7].

Instead of updating the LIP, LIS and LSP lists of [7], the appropriate bits in the BCS, BSP should be asserted. For each PCT, the coding is terminated after the Quantization Step is below a predetermined value.

To increase the coding efficiency, the compressed bit stream is not transmitted immediately; an intermediate buffer is used to accumulate all outputted bits for all the PCTs. In addition, after the completion of each step of the pseudocode of [7], one Rate-Distortion (*R-D*) point is accumulated in memory. We used the MSE as the Distortion metric. After the completion of the coding, the optimal truncation point for each PCT was identified using the well-known Lagrangian multiplier λ method [1], in a similar way to the Post-Compression Rate-Distortion Optimization of EBCOT [14]. Thus, given a target bit-rate, compression to this rate is feasible by establishing a monotonically decreasing set of *R-D* slope values [14] and then, based on the λ value, selecting the compression rate for each PCT. Since the proposed algorithm follows the partitioning ideas of SPIHT but applies them directly in the parent-children trees without any list functionality and uses a post-compression optimization method, it is called *Parent-Children Tree Set Partitioning with Optimization* (PCTSPO).

It must be noted that the examination of the PCT-coefficient memory is passive, since only read operations are performed without copying anything into lists. In addition, the proposed algorithm can be modified to compress independently each PCT. This is feasible if one arbitrary selects a value for the λ parameter, thus viewing the Lagrangian multiplier as a quality factor; higher values give better PSNR results and smaller compression ratios. This makes the coding algorithm easily parallelizable for a set of PCTs and reduces to a minimum the memory requirements.

4 EXPERIMENTAL RESULTS

In this section, the proposed system is compared with respect to memory utilization and coding efficiency with two state-of-the-art image-compression algorithms, the binary-uncoded SPIHT implementation from [7] and JPEG 2000 VM 6.0 [3].

Both the transform and the coding engine were developed in standard ANSI-C without any platform-dependent customizations. For the experimental setup, the 9/7 filter pair of [4] was used in a 5-level decomposition. The test images Lena, Goldhill and Bridge were used. All three are of size 512x512. For the SPIHT implementation, 4 decomposition levels were used because a 4-level wavelet tree used in SPIHT coding corresponds to the same number of subband coefficients as a 5-level PCT [7]. The size of the code-blocks in JPEG 2000 VM 6.0 implementation was selected based on the same criterion.

Table 1 — Memory sizes and profiling results for the three modes of operation (100% of the transform execution). The accesses to the FF do not include the filtering operations.

Memory Module	Size (coefficients)	# Accesses (#/pixel)
OM_rows	217	268282 (1.02)
OM_cols	7161	256170 (0.98)
IPM	1024	1475085 (5.63)
TM	93744	262144 (1.00)
IM	16384	262144 (1.00)
FF	9	1171981 (4.47)
TOTAL:	**118539**	**3695806 (14.10)**

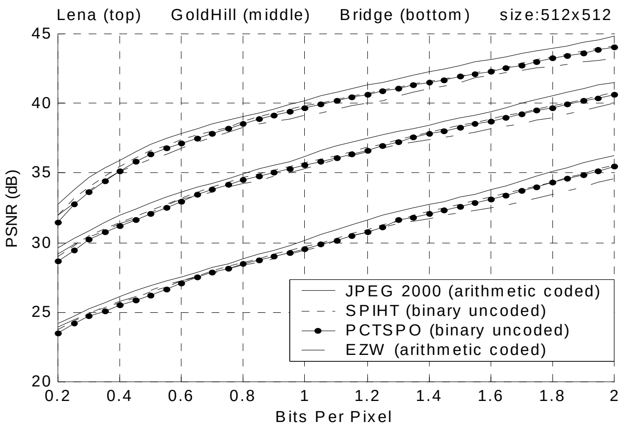

Figure 5 — Coding results for three standard test images.

Profiling results were obtained for the memory accesses to the various components. All the memory modules are considered to have bandwidth 1 coefficient/access. These results, along with the transform-engine memory requirements, are shown in Table 1. Even with the addition of the coding memory, the total memory used for encoding corresponds roughly to 1.3×10^5 coefficients (for a maximum 2.0 bpp compression rate, using the λ parameter as a quality factor), which is at least 5 times less than the minimum memory needs of the SPIHT linked-list implementation used in our experiments. In addition, if one assumes 16-bit quantization for the wavelet coefficients, the proposed system implementation is 100% more memory efficient from line-based implementation of JPEG 2000 [3], with respect to the numbers reported for the '-mem' parameter in the same experimental setup. To improve the memory efficiency of the VM6.0 software, code-blocks with size 8x128 were used. This corresponds to a low-memory setting but comes with a penalty of 0.3 dB degradation in PSNR [14].

The coding efficiency of the proposed scheme is illustrated in Figure 5. For all cases, the proposed system has inferior coding results to SPIHT, something that is in fact observed in all the similar compression methods [11] [16]. The average reduction in PSNR in comparison to the SPIHT is limited to 0.16 dB, while our results surpass in most cases the EZW coder.

5 CONCLUSION

In this paper, an efficient implementation of a novel image coding system has been proposed. Apart from the exploitation of the pipelined functionality between the transform and the coding system, the proposed approach combines also other advantages such as reduced memory utilization and inherent minimization of the external memory accesses without significant reduction of the compression efficiency.

6 ACKNOWLEDGEMENTS

The authors would like to acknowledge Dr. G. Lafruit (IMEC) for his technical comments during the implementation of the transform engine. This research was funded by the Belgian IWT (Project IWT-980302).

REFERENCES

[1] H. Everett, "Generalized Lagrange Multiplier Method for Solving Problems of Optimum Allocation of Resources," Operations Research, vol. 11, pp. 399-417, 1963.

[2] ISO/IEC JTC1/SC29/WG11, FCD 14496-1, "Coding of Moving Pictures and Audio," May 1998.

[3] ISO/IEC JTC1/SC29/WG1, WG1N1575, "JPEG 2000 Verification Model 6.0," January 2000.

[4] ISO/IEC SC29/WG1, FCD 15444-1, "JPEG 2000 Image Coding System,", official release expected at March 2001.

[5] G. Lafruit, L. Nachtergaele, J. Bormans, M. Engels and I. Bolsens, "Optimal Memory Organization for Scalable Texture Codecs in MPEG-4," IEEE Trans. CSVT, vol. 9, No. 2, pp. 218-243, March 1999.

[6] G. Lafruit, L. Nachtergaele, B. Vanhoof, F. Catthoor, "The Local Wavelet Transform: a memory-efficient, high-speed architecture optimized to a Region-Oriented Zero-Tree Coder," Integrated Computer-Aided Engineering, Vol. 7, No. 2, pp. 89-103, March 2000.

[7] A. Said, W. A. Pearlman, "A new fast and efficient image codec based on set partitioning in hierarchical trees," IEEE Trans. CSVT, vol. 6, pp. 243-250, June 1996.

[8] P. Schelkens, F. Decroos, G. Lafruit, F. Catthoor, and J. Cornelis, "Efficient Implementation of Embedded Zero-Tree Wavelet Encoding," Proc. of IEEE ICECS, Paphos, Cyprus, Vol. II, pp. 1155-1158, September 5-8, 1999.

[9] P. Schelkens, G. Lafruit, F. Decroos, J. Cornelis, and F. Catthoor, "Power Exploration For Embedded Zero-Tree Wavelet Encoding," Vrije Universiteit Brussel/IMEC, Brussel, ETRO/IRIS Technical Report TR-0061, 1999.

[10] J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," IEEE Trans. SP, vol. 41, pp. 3445-3462, Dec. 1993.

[11] R. Shively, E. Ammicht, P. Davis, "Generalizing SPIHT: A Family of Efficient Image Compression Algorithms," Proc. IEEE ICASSP 2000, Istanbul, Turkey.

[12] I. Sodagar, H. J. Lee, P. Hatrack and Y. Q. Zhang, "Scalable Wavelet Coding for Synthetic/Natural Hybrid Images," IEEE. Trans. CSVT, vol. 9, No. 2, pp. 244-254, March 1999.

[13] W. Sweldens, "Lifting Scheme: A New Philosophy in Biorthogonal Wavelet Constructions," Proc. SPIE, Vol. 2569, pp. 68-79, September 1995.

[14] D. Taubman, "High Performance Scalable Image Compression with EBCOT," IEEE Trans. IP, vol. 9, no. 7, pp. 1158-1170, 2000.

[15] M. Vishwanath, "The recursive pyramid algorithm for the discrete wavelet transform," IEEE Trans. SP., vol 42, pp. 673-676, Mar. 1994.

[16] F. Wheeler, W. A. Pearlman, "Low-Memory Packetized SPIHT Image Compression," Proc. of the 33st Asilomar Conf. On Signals, Systems and Computers, Nov. 1999.