Wavelet Transform Architectures: A System Level Review

M. Ferretti, D. Rizzo

Dip. Informatica e Sistemistica
Univ. of Pavia, Italy
e-mail : {ferretti,rizzo}@ipvmvl.unipv.it

Abstract. In this paper we review the architectures designed for wavelet transforms, with the purpose to highlight their suitability for inclusion in codec systems. Indeed, common VLSI cost functions (such as AT^2) are insufficient to evaluate architectures for compression. At the system level, quantization and coding have processing requirements that must be taken into account when designing the transform engine. The hierarchical structure of wavelet transform allows to use "pyramid" algorithms that optimize latency and processor utilization; on-line solutions try to minimize buffering memory. Such approaches can be substituted with more standard ones, if data reordering is mandatory to apply a good quantization strategy. An upcoming commercial solution offers a sound comparison paradigm.

1. Introduction

In this work, we analyze the architectures that have been proposed, designed and (sometimes) built for a fast implementation of the Discrete Wavelet Transform (DWT). The aim of this analysis is an evaluation of the system level requirements for compression applications. We mainly consider architectures suitable for a VLSI implementation, since an on-line, real-time DWT can become a valid alternative to DCT-based compression systems, especially when one aims at high compression ratios and at block distortion free decoded images. Furthermore, the multi-resolution structure of the DWT allows for flexible compression approaches, among which progressive image transmission, visual sensitivity based quantization and coding. Yet, the designs so far proposed for an hardware implementation of DWT have largely neglected the overall requirements of a compression-decompression system based on such a transform. In the following, we highlight such requirements and analyze to what degree existing proposals comply with them.

Indeed, a forward compression chain requires a pipeline of a few tasks: signal de-correlation, usually following a transform-based approach, coefficients pruning and quantization, samples coding. The backward transformation reverses the steps. This scheme copes only with static image compression.

From a system level point of view, the efficiency of a dedicated architecture for the transform phase of the compression pipeline can only be assessed when one consider all the stages: there is no "best" architecture for DWT, and the existing proposals, among which commercial ones [1], must be ranked according to criteria other than typical VLSI measures, such as AT^2. As an instance of this, we note a peculiarity of DWT-based compression systems which makes them so different from DCT-based ones: the hierarchical nature of the wavelet decomposition privileges the analysis of the whole image. It is not worth splitting the image into sub-images to be transformed, and this holds true even if the transform is computed up to an intermediate band. Usually, a 2-D separable DWT based on convolution is realized with a sequence of two 1-D steps (along the rows and along the columns); so, a buffering of the semi-transformed image is required. Depending on the specific approach, the amount of buffering memory varies from NxN down to O(Nk), N being the linear dimension of the image and k the support of the chosen wavelet. In contrast to this, DCT-based systems partition the image into blocks (whereby the block-effect distortion) and dimension the sub-systems on the basis of the 8x8 dimension of the block. With a DWT transform compression engine, resources for pruning, quantization and so forth must be dimensioned to accommodate for the whole transmitted image.

At the system level, one must consider the actual goal of the compression algorithm: a basic distinction is between a real-time implementation and an off-line compression and/or decompression scheme for storage and retrieval purposes. In the first case, the structure of the incoming signal is dictated by the TV standards and the primary goal is to keep pace with the frame frequency; as long as the three stages (transformation, quantization and coding) allow to sustain such a frequency, any solution for the transform is adequate. The trade off is in the amount of buffering memory used and the accuracy of quantization and coding. In the second scenario, time can be traded for space during coding: there is no limit on the latency and the coded signal can be best organized for a speedy decoding phase.

With reference to the first application case, a further constraint comes from the hierarchical structure of the DWT. Although it is possible to obtain a 2-D DWT using matrix multiplication or Fourier transform, the computations involved (quadratic and log proportional) outnumber by far those of the pyramidal algorithm proposed by Mallat [2], which has a cost linear in the number of samples. Therefore, all implementations are based on the hierarchical approach of that algorithm. Independently of the chosen implementation, the algorithm introduces strong time dependencies among the transformed values: in the straightforward case (Mallat), the bands are produced in sequence and the uppermost band is the last one to be computed; in the refined "recursive" implementation [3], the production of the values of the successive bands is scheduled as soon as the necessary intermediate data are available: upper bands are intermingled with the first band. So far for the forward transform. If one considers the reverse transform, the computations must be carried out starting from the uppermost band and proceeding downwards; so, the relative position of the data of the bands is very important. In the straightforward Mallat's algorithm, the first band to be used is the last produced; with the recursive algorithm, this constraint is somewhat relaxed but it persists. Therefore, a reordering/buffering of the output of the forward transform is mandatory. Even if the kernel of the wavelet is a symmetric one, it is very difficult to obtain a single device to be used both for the forward and for the inverse transform. The two filters (low pass and high pass) are used on the same input stream during the forward phase and produce separate outputs; in the reverse phase, they must be used on separate inputs and the corresponding outputs must be merged. The architecture can reuse the hardware for the convolution kernel, albeit with a rather different I/O and timing structure.

If one considers the application of compression and decompression for purposes other than real-time transmission, it becomes possible to use different I/O structures. As an instance, when the Haar basis is used, one can built a very simple pipeline of convolution processors [4], conceptually one processor per band, that compute a non-separable version of the transform. The kernel of this non-separable version is 2x2. Input data are read in an order that is optimal for the scheduling of the processors; no memory is required for intermediate storage, since the four pipeline registers at each stage are enough to sustain the whole computation. This approach results in a very simple architecture because the support of the kernel and the bi-dimensional subsampling intrinsic to the DWT match perfectly; extending the pipelined, non-separable DWT implementation to larger kernel requires a careful analysis; we are currently checking the area-time trade off.

In the following, we will briefly review existing proposals with the purpose to highlight *latency* , *output scheduling*, *buffering* for quantization, coding and *data reordering* for backward transform.

2. Architectures

The implementations on general purpose SIMD and MIMD systems are outside the scope of this analysis. Preliminary dedicated solutions [6,7] have been superseded by more advanced VLSI oriented ones, based on systolic convolution, parallel filters and/or mixed structures.

2.1 Systolic 1-D

The standard systolic convolution for 1-D signals is adapted to wavelets in two implementations [8,9]. Both apply the "recursive" pyramid algorithm. This algorithm assumes that the coefficients of the wavelet filters, both low pass and high pass, are applied to the input sequence after the usual convolution mirroring; moreover, border values are computed by extending the input with zeros. With these two hypotheses, the first output value can be computed at each band as soon as a single intermediate value is available. In the usual systolic convolution, the k PEs are used every other clock (50% utilization); this allows to produce both the low pass and the high pass outputs (the first band) from the same input, as if working on two independents streams. The subsampling by two due to the wavelet structure makes one out of two computations useless; the $k/2$ PEs thus available can be scheduled to perform the convolutions of higher bands. This is due to the fact that the processing steps required for higher bands are as many as those of the first band. Both proposals generate the output according the schedule of the "recursive" pyramid.

The proposals differ in the management of the storage required for upper bands. In [8] every PE has a local memory of size -1+logN. A systolic control flow generates the proper addressing sequence at each PE and makes data extracted from the local memory also travel to the correct nearby memory element (at distance 2). The activation sequences can be easily derived from the changes in the bits of a gray-code counter having logN bits. In [9] a few alternatives are proposed: i) a systolic routing network which accepts sideways outputs of the low pass convolution and delivers them in proper order to the sideways inputs of PEs when working for higher bands (a semi-systolic version is also possible); ii) a complex RAM, capable of sustaining concurrently a single write and k/4 read operations.

A systolic network is designed in [10] by casting the wavelet computation into recursive equations and by deriving the Dependency Graph. The hierarchical data-flow dependencies due to the subsampling make the derivation of an admissible systolic schedule a non trivial task. By considering the computation of three bands only, the authors obtain a *k* PEs systolic network for the low-pass filter chain; it does not require additional external storage, a considerable improvement over the other solutions. Furthermore, the overall latency is 3/2N, to be compared with the 2N of the "recursive" algorithm. However, processor exploitation drops to 58% and latency scales badly with the depth of computation across bands, since it becomes 2N again if one computes 4 bands and gets even worse for more octaves. A second disadvantage of this otherwise attractive solution is the output schedule it generates, which is less regular than what was available with the recursive based implementations.

2.2 Systolic 2-D

When the 2-D DWT is realized applying the separable row and column kernels, a systolic structure can be used for processing the rows; intermediate buffers are required to store values for column processing, and this second step cannot be organized with a systolic approach, otherwise latency grows too much. Usually, parallel filters are applied on the columns.

The architecture proposed in [9] consists of a systolic section of two filters for row processing (the same of the 1-D problem), of a buffering memory for data reformatting, of two column parallel filters and of a routing memory to feed the 1-D row section with data computed by column parallel filters. The dimension of this routing network is the same as in the 1-D case. Only one of the two row filters needs to compute higher bands. The additional storage for rows-to-columns reformatting consists of $2Nk$ cells. The overall latency is N^2+N. The output schedule complies with the 2-D "recursive" pyramid algorithm.

A much more complex architecture is proposed in [11]. It embeds the Mallat's algorithm in a time and space parallel structure: for each band there is a systolic filter at each row (space parallelism) that computes the high pass and low pass convolution and four column parallel filters for the sub-bands; bands are activated in pipeline mode. The filters used for row processing are different from the analogous ones of the previous systolic structures, since they read in two samples per clock and produce concurrently both the low pass and high pass outputs. A completely parallel structures uses N such filters for the first band, and $N/2^{i-1}$ for band i ($i\geq2$); parallel filters process the whole column at each band; no staging memory is required. To reduce the hardware complexity of this completely parallel solution, a partitioning scheme processes Q rows ($Q<N$) and uses intermediate buffers to store partial results

for column parallel filters. Each parallel filter in each sub-band requires $N/2^i$ storage cells. This complex structure is best used by continuously pipelining new problem instances, without inter frame latencies. So, using a fully parallel system, the computation time per problem instance is $O(N/2)$. Practical implementations demand a less complex I/O structure: in the limiting case of serial I/O (Q=1), a pipelined sequence of problems takes $O(N^2/2)$ per instance. The output schedule produces parallel outputs at the various bands; values of the sub-bands are generated concurrently.

2.3 Parallel Filters

When VLSI technology is quick enough to support multiple MAC operations between the edges of an external clock cycle, convolution can be implemented with parallel filters rather than with systolic structures. This approach is followed in [5] both for 1-D and for 2-D DWT. The "recursive" pyramid algorithm is modified to accommodate for the latency introduced by the k-tap filters of the low pass and high pass convolution. Inter-octave low pass values are buffered in $\log N$ shift registers, each k-stage long. The k values are extracted in parallel from the proper shift register and forward to the two filters; conflicts in the use of the filters from multiple bands are resolved by delaying the computation of the higher one, according to the "modified recursive" pyramid algorithm. A 2-D DWT can be carried out even for a non-separable kernel by a straightforward extension of this architecture. Intermediate storage grows to $KN/2^{i-1}$, $1 \le i \le \log N$. This solution is very similar to the non-separable Haar transform implementation of [4], even though it accepts data in row major order, thus requiring much more memory.

A somewhat less general solution is described in [12]. The number of bands to be computed is limited to m; the data dependencies implied by the convolutions of 2^m input samples are exploited to derive, using a generalized "life-time" analysis, the minimal storage required to compute higher octaves. The 1-D convolutions are carried out with two k-tap parallel filters. The optimized datapath has been specified for the inverse transform as well. To improve the VLSI feasibility of the architecture, a variable *digit-serial* structure has also been obtained.

A block-based I/O structure is advocated in [13], with n input samples being read in per clock cycle. It is suitable to manage large images and to obtain higher frame rates than the corresponding parallel 2D architectures based on serial input. The throughput rate is n pixels per clock cycle with a latency $N^2/n + 2k+n+3jn$ for j bands. The memory buffering required consists of $j(k-1)(N+n)+ 2n^2$ cells. The order with which data outputs are produced depends on the scheduling that has been done to design the scheduling circuits.

By trading generality and scalability with optimization, the DWT convolution can be implemented exploiting regularities due to the values of the coefficient of the chosen kernel. This approach has been applied in [14] to the separable, 4-coefficient Daubechies wavelet. The data flow graph of the low pass and high pass filters have been optimized for that kernel, both for the forward and the backward transform. An optimized, retimed pipeline produces two outputs (1 low pass and 1 high pass) at every clock cycle, with a latency of 5 cycles, provided that it is fed with two input samples per clock cycle. Nothing else is proposed to manage the row-to-column reformatting.

The preliminary specification of a commercial device [1] allows to evaluate the system level choices done to embed into a single chip the complete chain required for compression, namely wavelet transformation, coefficient quantization and coding; the same device can be used for the whole reverse chain. The ADV601 device interfaces with the most common video formats (PAL and NTSC), handles color, and uses the (7,9) bi-orthogonal wavelet. Furthermore, it applies sub-band adaptive quantization, performs run-length encoding as a preliminary step to Huffman coding. Since this device has been designed to support the whole compression decompression chain, it embeds resources to execute the various steps with a minimum of external logic. In the following, we review what can be deduced by the obviously short description of the preliminary specification.

The device applies a modified Mallat algorithm: the first pass consists of the high pass and low pass filters along the rows only. The decimated high pass output is "discarded" (actually output), and the decimated low pass filter is used as the basis of the subsequent steps, according to the standard LL,LH,HL,HH decomposition. The likely reason of this choice is the gain in internal SRAM buffer required to support the row-to-column reshuffling; this buffer is halved with respect to what necessary to complete the LL, LH, HL and HH decomposition on the whole image. The complex output scheduling of the transform phase privileges a coherent processing of the sub-bands for the subsequent quantization; a set of statistics on the coefficients are collected on-the-fly, and transformed values must be stored in an external DRAM for reordering and subsequent quantization. In order to produce a constant bit-rate for the output stream, external cooperation from a DSP is required to compute the "bin width" to be used for the intra sub-band adaptive quantization. Run length encoding and Huffman coding are executed on sub-bands. The complete encoded sequence is made up of a hierarchy of fields, containing the blocks of the luminance and chrominance components: each block contains a sequence of encoded data, with the proper information for the inverse quantization step.

3. System Level Considerations

The brief analysis of the proposed architectures for DWT is summarized in Table I. The items quoted are those pertinent to evaluating the architecture from the system level perspective. *Latency* gives a measure of the execution time of a single problem instance, *output schedule* effects the ordering of transformed data as generated by the transform phase, *buffering* indicates the amount of storage required for the transform, *IDWT* lists the proposals that analyze in detail the architectural requirements of the inverse transform, *N.Bands* is the number of levels for which the transform can be computed. The table has no entry for [14], which describes just an enhanced filtering sub-unit, and for the commercial ADV601, which is a complete codec chip.

The table shows that the inverse transform has been largely neglected. This is surprising, since the data dependencies of the IDWT are considerably different from those of the DWT. As a consequence of this fact, while most proposals aim at minimizing memory requirements for the internal buffering necessary to the DWT, no consideration is placed on the required data storage and reformatting for matching the output from the DWT stream with the input to IDWT.

No effort is directed towards an easy adaptive quantization. A small latency is usually obtained at the cost of a rather irregular output schedule. Unless quantization is carried out in static, point-wise way, there is a need to collect statistics, most likely to be computed within sub-bands. Output schedules should therefore produce time contiguity among transformed values within sub-bands, so minimizing the unavoidable extra memory required to store outgoing values while statistics are collected. The "recursive" pyramid algorithm has a 2^i dependency on the level i; other solutions are even worse. The system level consequence of the need of a proper output schedule is that latencies are only required not to exceed the amount of time necessary for storing and retrieving transformed values to be quantized.

It is worth noting that some architectures are only specified for 3 bands at the most; by profiting from this, certain gains are possible and the specialized architectures do not scale well. However, one should keep into account the fact that the transform process is seldom carried out completely, especially when considering compression applications. Thus, such specialized solutions can offer interesting engineering results.

Progressive image transmission should also be easier with hierarchical transforms such as wavelet. Again, output schedule heavily influences this system requirement. Proper and easy identification of upper bands time slots is mandatory, but not sufficient, if data reordering and grouping is not possible.

References

[1] ADV601 Preliminary Data Sheet, ANALOG DEVICES, 1996. Available at http://www.analog.com/pdf/adv601.pdf

[2] S. Mallat, " A Theory for Multiresolution Signal Decomposition: The Wavelet Representation," *IEEE Trans. on Pattern Analysis and Machine Intell.*, vol. 11, n. 7, pp. 674-693, July 1989.

[3] M. Vishwanath, "The Recursive Pyramid Algorithm for the Discrete Wavelet Transform," *IEEE Trans. on Signal Processing*, vol. 42, n. 3, pp. 673-677, March 1994.

[4] M.G. Albanesi, M. Ferretti, "A High Speed Haar Transform Implementation," *J. Circuits, Systems and Comp.*, vol. 2, n. 3, pp. 207-226, 1992.

[5] C.Chakrabarthi, M.Vishwanath "Efficient realizations of the Discrete and Continuous Wavelet Transform: From Single Chip Implementations to Mappings on Simd Array Computers," *IEEE Trans. on Signal Processing*, Vol 43, no.3,March 1995,pp759-771.

[6] G. Knowles "VLSI architecture for the discrete time wavelet transform," *Electronics Letters*, vol. 26, n. 15, pp. 1184-1185, July 1990.

[7] A.S.Lewis, G.Knowles "VLSI architectures for 2-D Daubechies wavelet transform without multipliers," *Electronics Letters*, vol. 27, n. 2, pp. 171-173, Jan. 1991.

[8] R. Lang, E. Plesner, H. Schroder, A. Spray, "An efficient systolic architecture for the one-dimensional wavelet transform," SPIE, Vol 2242, Wavelet Applications, pp.925-935, 1994.

[9] M. Vishwanath, R.M. Owens, M.J. Irwin, "VLSI Architectures for the Discrete Wavelet Transform," *IEEE Trans. On circuits and Systems-II: Analog and Digital Signal Processing*, vol. 42, n. 5, pp. 305-316, May 1995.

[10] Josè Fridman, S.Manolakos "On the synthesis of regular VLSI architectures for the 1-D discrete wavelet transform," Proc. of SPIE Conf. on Mathematical Imaging: Wavelet Application in Signal and Image Processing II, San Diego CA, July 1994.

[11] H.Y.H. Chuang, L. Chen "VLSI Architectures for Fast 2D Discrete Orthonormal Wavelet Transform," *Journal of VLSI Signal Processing*, vol. 10, pp. 225-236, 1995.

[12] K. K. Parhi, T. Nishitani, "VLSI Architectures for Discrete Wavelet Transforms," *IEEE Trans. on VLSI*, vol. 1, n. 2, pp. 191-202, June 1993.

[13] J. Bae, V.K. Prasanna, "Synthesis of VLSI Architectures for Two-Dimensional Discrete Wavelet Transforms," in *Int. Conf. on Appl. Specific Array Proc.*, July 1995.

[14] X. Chen, T. Zhou, Q. Zhang, W. Li and H. Min, "A VLSI Architecture for Discrete Wavelet Transform," *Proc. ICIP96*, vol. 2, pp. 1003-1006, 1996.

Ref.	*Latency*	*Out. sched.*	*Buffering*	*IDWT*	*Bands*	*Structure*
[8]	**2N**	**RPA**	**k (logn-1)**	**no**	**logN**	**1-D syst.**
[10]	**3/2 N**	**irregular**	**no**	**no**	**3**	**1-D syst.**
[9a]	**2N**	**RPA**	**k(logN-1)**	**no**	**logN**	**1-D syst.**
[5a]	**≈N**	**MRPA**	**klogN**	**no**	**logN**	**1-D p.f.**
[12]	**n.a. (*)**	**n.a. (*)**	**minimal(*)**	**yes**	**3**	**1-D p.f.**
[11]	**$N^2/(2Q)+2^{j-1}(2k+3)$**	**M+par.I/O**	**absent**	**no**	**logN**	**2-D mix.**
[9b]	**N^2+N**	**RPA**	**2kN+k(logN-1)**	**no**	**logN**	**2-D mix.**
[5b]	**$\approx N^2$**	**MRPA**	**$kN/2^{i-1}$ (**)**	**no**	**logN**	**2-D p.f.**
[13]	**$N^2/n+2k+n+3nj$**	**n.a. (***)**	**$j(k-1)(N+n)+2n^2$**	**no**	**logN**	**2-D p.f.**

Table I. j=logN, N is the linear dimension of the image. Output schedule **M** refers to Mallat's. **p.f.** stands for parallel filters.(*) Latency and output schedule depend on the length of the filter, on the number of bands but no closed form expression is available. (**) Dimension of the shift register for each band i. (***) Ouput schedule similar to that of [12]. See (**). A similar annotation holds for buffering requirements.