VLSI Architecture for Lossless Compression of Medical Images Using the Discrete Wavelet Transform

Isidro Urriza, José I. Artigas, José I. García , Luis A. Barragán and Denis Navarro

University of Zaragoza
María de Luna, 3. 50015 Zaragoza, Spain
email: urriza@posta.unizar.es

Abstract

This paper presents a VLSI Architecture to implement the forward and inverse 2-D Discrete Wavelet Transform (FDWT/IDWT), to compress medical images for storage and retrieval. Lossless compression is usually required in the medical image field. The word length required for lossless compression makes too expensive the area cost of the architectures that appear in the literature. Thus, there is a clear need for designing an architecture to implement the lossless compression of medical images using DWT.

The datapath word-length has been selected to ensure the lossless accuracy criteria leading a high speed implementation with small chip area. The result is a pipelined architecture that supports single chip implementation in VLSI technology. The architecture has been simulated in VHDL and has a hardware utilization efficiency greater than 99%. It can compute the FDWT/IDWT at a rate of 3.5 512· 512 12 bit images/s corresponding to a clock speed of 33MHz.

1: Introduction

In the medical imaging field, digitized images are replacing conventional analog images as photograph or X-rays. However, the vast volume of data required to describe such images considerably slows transmission and makes storage prohibitively costly. Images from Computer Tomography (CT) or Magnetic Resonance (MR) are 512x512 pixels and 12 bit in depth [1]. The quantity of data must be reduced while maintaining the reconstructed image an acceptable fidelity with the original image. Usually this fidelity must be total and lossless compression is required.

Currently, block DCT based compression methods produce block-like image artifacts that could mask or be mistaken for pathology in medical images. In contrast with the block DCT, DWT [2], [3] does not partition the image into blocks for coding, thereby significantly reducing artifact generation in the reconstructed image. Compression is achieved by quantizing and then coding the wavelet data.

The computation of the DWT is a very time consuming task and parallel processing in the form of specialized hardware is therefore necessary. VLSI architectures for DWT computation are reported in the literature [4]- [14]. These VLSI implementations of the forward and inverse wavelet transform use multiply and accumulate (MAC) units, and memory modules to store intermediate results. The implementation cost depends on the number of multipliers and size of storage units. Further, these parameters will depend on the image size, wavelet filters, number of scales and word length.

This paper concentrates on the filters best suited to image compression according to Villasenor [15]. For them, we have obtained the minimum word length required to obtain a reconstructed image numerically identical to the original one. This word length has been obtained in a previous work [16].

In [14], it is surveyed the number of multipliers and memory elements required for the existing implementations of the FDWT and the IDWT that appear in the literature. These architectures have been designed to work with 8 bit resolution images and to achieve a performance of 50-60 dB SNR. If lossless compression is required, high word length is needed and it makes their area implementation cost too high. Therefore, there is a clear need for designing an architecture to perform lossless compression of medical images using the DWT.

The paper is organized as follows. Section 2 reviews wavelet concepts and presents the filters that will be used in this paper. The effect of finite precision in DWT computation is presented in Section 3. The VLSI architecture for computing the DWT is presented in Section 4. Finally, conclusions are reported in Section 5.

2: The Discrete Wavelet Transform

The discrete wavelet transform (DWT) decomposes any arbitrary function *d* into different scale levels, where

each level is then further decomposed with a resolution adapted to the level. The image decomposition method used in this paper was developed by Mallat [2], and is implemented by a pair of Quadrature Mirror Filters (QMFs) using the pyramid algorithm. Each QMF pair consists of a low pass filter (H) and a high pass filter (G) that splits a signal's bandwidth in half.

The two dimensional filtering decomposes an image into an average signal (d_{HH}^{j}), and three detail signals which are directional sensitives (Fig. 1). The average signal is the initial image for the next scale. Then, for a *S*-scale DWT, the algorithm runs from j = 1 to S being d_{HH}^{0} the original image. The output is an average signal d_{HH}^{S}, as well as the detail signals d_{HG}^{j}, d_{GH}^{j}, d_{GG}^{j} from j = 1 to S. The inverse wavelet transform is calculated in the reverse manner, i.e., starting from the lowest resolution subimages, the higher resolution images are calculated recursively.

In this paper, we will concentrate on 6 filters that are the best suited to image compression according to [15]. The filter coefficients proposed in [15] are shown in Table I. For each filter bank, it is given the low pass filters (H) and inverse low pass filters ($\overline{H}$), the number of taps and the filter coefficients. Origin is the leftmost coefficient. Coefficients for negative indices follow by the symmetry of QMFs, and the high pass transform coefficients can be derived from the low pass QMFs. The fifth column represents the sum of the absolute values of the filter coefficients.

For the FDWT, the number of multiply and accumulate (MAC) operations to compute the jth scale from the (j-1)th scale is:

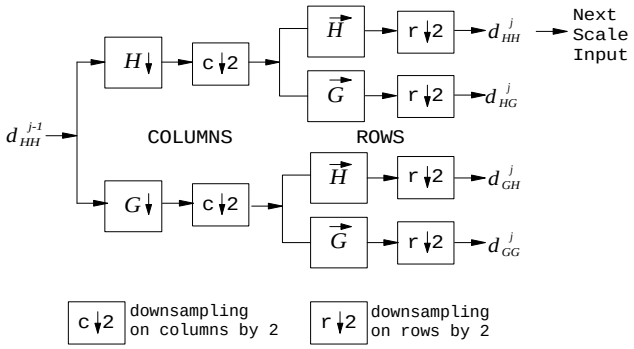

Fig. 1. Basic building block of the 2D forward discrete wavelet transform using pyramid algorithm.

$$4 \cdot N^2 \cdot \big(L(H) + L(G)\big) \cdot \frac{1}{2^{2j}} \tag{1}$$

where N is the number of rows/columns of the image, and $L(H)$, $L(G)$ the length of the QMF filters. If the FDWT algorithm runs from j=1 to S, the total number of MACs operations is:

$$\frac{4}{3} \cdot N^2 \cdot \big(L(H) + L(G)\big) \cdot \left[1 - \left(\frac{1}{4}\right)^{S+1}\right] \tag{2}$$

The same result is valid for the IDWT. Let us assume, N=512, QMF filters length 13, and S=6, then the number of MAC operations to perform the FDWT of an image is $8.99 \cdot 10^6$, and a 133MHz Pentium PC spends 42 seconds to compute it.

		L	Filter coefficients c_n	$\sum \lvert c_n \rvert$
F1	H	9	0.852699, 0.377402, -0.110624, -0.023849, 0.037828	1.952105
	$\overline{H}$	7	0.788486, 0.418092, -0.040689, -0.064539	1.835126
F2	H	13	0.767245, 0.383269, -0.068878, -0.033475, 0.047282, 0.003759, -0.008473	1.857495
	$\overline{H}$	11	0.832848, 0.448109, -0.069163, -0.108737, 0.006292, 0.014182	2.125814
F3	H	6	0.788486, 0.047699, -0.129078	1.930526
	$\overline{H}$	10	0.615051, 0.133389, -0.067237, 0.006989, 0.018914	1.683160
F4	H	5	1.060660, 0.353553, -0.176777	2.121320
	$\overline{H}$	3	0.707107, 0.353553	1.414214
F5	H	2	0.707107, 0.707107	1.414214
	$\overline{H}$	6	0.707107, 0.088388, -0.088388	1.767767
F6	H	9	0.994369, 0.419845, -0.176777, -0.066291, 0.033145	2.386485
	$\overline{H}$	3	0.707107, 0.353553	1.414213

Table I. Best Filters for wavelet image compression [15].

3: Word Length for Lossless Compression

Due to finite precision arithmetic, the reconstructed image might be not numerically identical to the original one, on a pixel-by-pixel basis. That means that lossless compression is not achieved. In the medical image field, that may compromise the diagnostic accuracy.

The adopted numerical system is fixed-point two's complement. For each scale *j*, the FDWT (Fig. 1) causes the magnitude of subimages d_{HH}^{j}, d_{HG}^{j}, d_{GH}^{j}, and d_{GG}^{j} to grow with respect to the initial image d_{HH}^{j-1}, thus the dynamic range increases with scale. The rate of increase is upper bounded by $\left(\sum |c_n|\right)^2$ which is greater than unity (Table I). If the integer part is maintained fixed and equal for all scales *j*, the most significant bits are only fully used in the last scale. Thus, our approach is to increase the integer part with the scale. Table II [16] represents the minimum integer part b_{int} we have to use in order not to exceed the dynamic range for the filters in Table I.

	Scale					
Filter	*s*=1	*s*=2	*s*=3	*s*=4	*s*=5	*s*=6
F1	15	17	19	21	23	25
F2	16	17	19	21	23	25
F3	15	17	19	21	23	25
F4	16	18	20	22	24	27
F5	15	16	17	18	19	20
F6	16	19	21	24	26	29

Table II. $b_{\text{int}}(s)$ **per scale**

However, in the case of the IDWT, the dynamic range of the reconstructed image at scale *j*-1 decreases with respect to the reconstructed image and detail images at scale *j* due to the perfect reconstruction property of wavelet transform. Using a word length of 32 bits with integer part variable, the lossless accuracy criteria is fulfilled [16].

To sum up, the used precision is 13 bits (including sign) for input images, 32 bits for wavelet filter, and 32 bits with variable integer part with scale for intermediate results.

The VLSI architectures for DWT computation reported in the literature [4]- [14], can be grouped into four types:

A.- *Serial-Parallel* architecture [14]: it consist of two serial filters to compute rows and two parallel filters to compute columns. This circuit is fed in with two rows at the same time.

B.- *Parallel* architecture [14]: is a modification of the previous one, in that all the filters are parallel filters. In this case the input of the circuit is only one row.

C.- *Block based* filtering proposal [13]: Image is splitted into blocks, usually with the same size as the filter length, each block can be processed following *Serial-Parallel* or *Parallel* approach.

D.- *Recursive* 1-D WT [11]. The 1-D WT is computed for all scales in row order, the resulting image is transposed and finally the 1-D WT is computed again over rows.

The above architectures have been designed to work with 8 bit resolution images and to achieve a performance of 50-60dB SNR. Table III shows the hardware requirements for this architectures in number of arithmetic blocks (multipliers and adders) and memory elements.

	Multipliers	**Memory Elements**	**Area (mm²)**
Serial-Parallel	$4 \cdot L$	$2 \cdot L \cdot N + N$	254.36
Parallel	$4 \cdot L$	$2 \cdot L \cdot N + N$	254.36
Block Filtering	$2 \cdot L \cdot S$	$S \cdot N \cdot 3/4 + (21 \cdot S + 1) \cdot S/2 + 27$	246.64
Recursive 1D	$2 \cdot L$	$S \cdot 28/6 + 2 \cdot L \cdot N$	173.72

Table III. Requirements for architectures, *L*: Filter length, *S*: Number of scales, *N*: number of Rows/Columns

Assuming a 0.7μm CMOS technology [18], the multipliers and RAM areas have been obtained generating the cells using the ES2 megacell compiler. The fourth column of Table III represent the area that is occupied by the multipliers and memory blocks for *L*=13, *S*=6, *N*=512 and word length of 32 bits. The results show that the implementation cost is unaffordable for lossless accuracy.

Our goal is to design a specific hardware to speed up the FDWT/IDWT computation with respect to a desktop PC, while maintaining an affordable implementation cost.

4: Proposed Architecture

The datapath block diagram is shown in Fig. 3. We have chosen a implementation for the filters that uses one 32×32 bits multiplier, one 64 bits accumulator and $(N/2) + 2 \cdot 32$ memory elements of 32 bits, in contrast to the hardware requirements shown in Table III.

This architecture uses an external DRAM memory to store initial, intermediate and final convolution results. Only one image size memory is necessary to compute the FDWT/IDWT transform. DRAM accesses are slow and so that, the proposed architecture minimizes them. Each data is read and written only once from/to the DRAM. An input buffer is needed to hold read data from the external memory and an output buffer arranged like a FIFO is

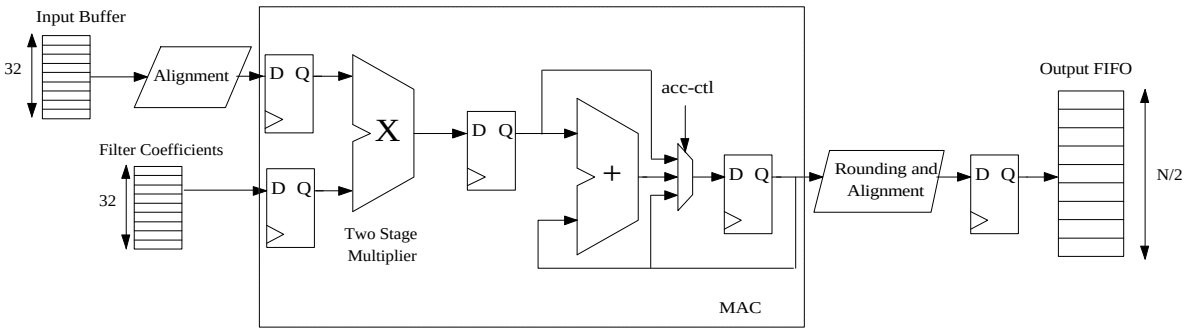

Fig. 3. Data path Block Diagram

employed to avoid data depedences between DRAM reads and writes.

The design of the architecture is based on the computation schedule shown in Fig. 2, which in his turn is the result of applying the pyramid algorithm. A MAC operations to compute one data convolution for 13 tap filters is performed on a macrocycle of 13 cycles (0 to 12). Cycles 13 to 18 are used to extend the macrocycle when a refresh is required by the external DRAM. Every macrocycle, a read and a write operation are performed from the DRAM, and 13 reads from the filter coefficients RAM.

The architecture achieves a maximum utilization factor (busy_cyles/total_cycles) of the filter unit. The multiplier is idle only during the refresh operation of the external DRAMs.

$$Utilization(Multiplier) = \frac{Busy\_cycles}{Total\_cycles} = 99.04\%$$

The architecture has been modeled in fully synthesizable VHDL and simulated on data taken from random images and gave the same output as a software implementation.

4.1: Input Buffer Organization

To minimize DRAM accesses we need an input buffer to hold input data while they are alive in such a way that each data is read only once from the memory. Once a data is read, it must remain live in the input buffer in order to produce the subsequent scale output data during L cycles. That is not valid for the data near the edges of the image due to the border effects. We use a "so called" circular convolution, i.e., an NxN image is extended periodically on both rows and columns. So border data are alive while the row/column is being computed.

If the filter length is $L = 2 \cdot l + 1$, $2 \cdot l$ data are the number of rows/columns extended beyond the image edges, l on the left/top and l on the right/bottom. And $2 \cdot l + 1$ data are needed to compute a new result. Then, the minimum size for this buffer will be:

$$Bsize = 2 \cdot l + 2 \cdot l + 1 = 4 \cdot l + 1$$

Cycle	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
DRAM Manager	DRAM rd						DRAM wr			
Input Buffer	rd_cf4	rd_cf5	rd_cf6	rd_cf7	rd_cf8	rd_cf9	rd_cf10	rd_cf11	rd_cf12	rd_cf13
Acc_ctl	load	acc	acc	acc	acc	acc	acc	acc	acc	acc
Output FIFO	wr					rd				

Cycle	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**
DRAM Manager	DRAM wr		branch	DRAM refresh					
Input Buffer	rd_cf1	rd_cf2	rd_cf3	idle	idle	dec. ptr.	rd_cf1	rd_cf2	rd_cf3
Acc_ctl	acc	acc	acc	hold	hold	hold	hold	hold	hold
Output FIFO									

Fig. 2. Operation scheduling

To simplify the control, this minimum size is rounded up to the nearest power of two. For instance, for L = 13 tap, *Bsize* = 4·6+1 = 25, and the taken buffer size will be 32.

This buffer is folded in two banks (Fig. 4). For even rows/columns, the top of the first bank stores the data used to extend the image and the second bank holds the following *Bsize*/2 data. Depending on the row/column length Bank2 is reused *#rounds* times (Table IV). The remaining data is stored in the bottom of Bank1. The behavior of these banks switches for odd rows/columns.

4.2: MAC Unit

Table V shows part of the 32 x 32 bit multiplier data sheet generated by means of the ES2 Megacell Compiler. We have considered the worst-case industrial conditions for the timing parameters. This 32 x 32 bit multiplier is too slow (access time of 50.88ns) for our purposes. So, we have designed a Wallace tree 2-stage pipelined multiplier. This block is larger than the compiled one, but has a propagation delay that allow us work with a clock period of 25ns. The accumulation is performed in 64 bits to increase the accuracy.

4.3: Alignment and Rounding

As we have said in Section 3, the integer part of the intermediate wavelet data changes with scale. The alignment unit is in charge of this task. The increment (FDWT) or decrement (IDWT) of the integer part at each scale is stored in a configuration memory since it depends on the filters (Table II).

After the accumulation in 64 bits and the bit alignment, rounding narrows the datapath word length to 32 bits. If the MSB of the truncated bits is 0, truncation is performed; if the MSB is 1, then round-up by one is performed.

4.4: FIFO Organization

When calculating the FDWT from scale *S1* to *S2*, the output of a convolution is used as input for the next one. Then, when computing the horizontal convolution on a column, we must use the old values generated by the last convolution. Therefore, there are a write-after-read dependence. For each row *j* in the same processing column, we must ensure that the number of cycles since this memory position *j* is read till the same memory position *j* is written must be greater than zero.

For a given column *c*, the memory position of row *j*, is read and write at the following cycles:

$$\mathrm{Re}ad\_cycle(j) = l+1+j,\ from\ j=0\ to\ \frac{N}{2^{S1}}-l-1$$

$$Write\_cycle(j) = \begin{cases} 2\cdot l+1+2\cdot j, \\ \quad \text{from j} = 0\ \text{ to } \left(\mathrm{N}/2^{\mathrm{S2}}\right)-1 \\ 2\cdot l+2+2\cdot\left(j-N/2^{S2}\right), \\ \text{from j} = \left(N/2^{S2}\right) \text{ to } \left(\mathrm{N}/2^{\mathrm{S1}}\right)-1 \end{cases}$$

where $L = 2 \cdot l+1$ is the filter length. Let us define the distance of the data dependence as the difference between the cycle at which *j* is written and the cycle at which it is read:

$$Distance(j) = Write\_cycle(j) - \mathrm{Re}ad\_cycle(j)$$

When the new value is available before the old value has been read, the new value write operation is delayed D

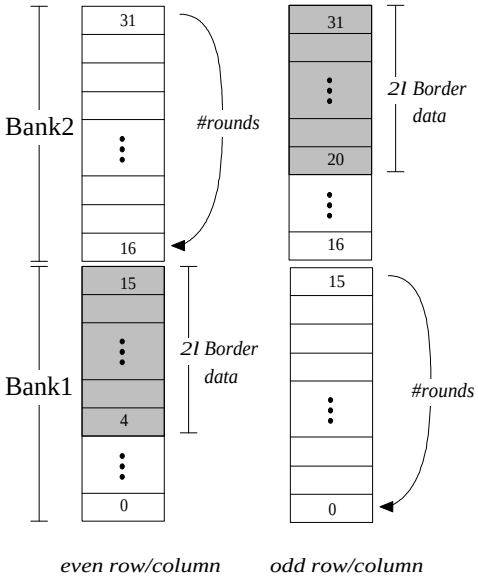

Fig. 4 Input buffer organization

Scale	row/column size	#rounds
1	512	31
2	256	15
3	128	7
4	64	3
5	32	1
6	16	0

Table IV. Bank2 utilization for a 512x512 image

	Access time to (ns)	Cell Area (mm^2)
ES2	50.88	2.92
Pipelined	23.45	8.03

Table V. DWT Multiplier Design

cycles in order to satisfy the order implied by the dependence such that:

$$MIN\left[Distance(j)\right] + D > 0 .$$

D delays are generated with a *D* deep FIFO. It can be seen that *D* depends on the scale. Thus, the variable deep FIFO is implemented in an intermediate RAM that is accessed as a FIFO.

The above calculations give a minimum value to *D*. But *D* is also upper bounded. In the change between vertical and horizontal convolution appear read-after-write data dependences that impose a maximum value to *D*. The IDWT also imposes restrictions on *D*. Table VI sums up the imposed bounds on *D* for $N = 512$ and $L = 13$.

Scale	1	2	3	4	5	6
MIN(*D*)	250	122	58	26	10	2
MAX(*D*)	504	248	120	56	24	8

Table VI. Bounds on FIFO deep

5. Conclusions

A VLSI architecture for computing the forward and inverse 2-D Discrete Wavelet Transform has been presented. The targeted applications aim at the compression for storage and retrieval of Medical Images, thus real-time is not required. The datapath word length has been selected to achieve lossless compression for the filters best suited to image compression that appear in the literature [15]. The implementation employs only one multiplier and N/2 + 32 memory elements to compute all scales what results in a considerable smaller chip area (11.2 mm^2) than former implementations (Table III). The hardware design has been captured by means of the VHDL language and simulated on data taken from random images. Running at 33MHz, this architecture computes 3.5 512×512 with 12 bit resolution images/s. Thus, our architecture is 154 times faster than a desktop Pentium 133MHz PC. We are currently working on the design of a chip based on the proposed architecture, with a PCI Bus interface. This chip is the core of a PCI board that will speedup the DWT computation on desktop PCs.

6: References

[1] H. Lee, Y. Kim, E. A. Riskin, A. H. Rouwert, and M. S. Frank, A Predictive Classified Vector Quantizer and its Subjective Quality Evaluation for X-ray CT images, IEEE Trans. on Medical Imaging, vol. 14, nº 2, pp. 397-406, 1995.

[2] S. Mallat, "A Theory of Multiresolution Signal Decomposition: The Wavelet Representation" in *IEEE Trans. Patt. Analy. Mach. Intell.*, Vol. 11, pp. 674-693, 1989.

[3] M. Antonini, M. Barlaud, P. Mathieu, and Y. Daubechies, "Image Coding using Wavelet Transform" in *IEEE Trans. on Image Processing.*, Vol. 1, No. 2, pp. 205-220, 1992.

[4] A.S. Lewis and G. Knowles, "VLSI architecture for 2D Daubechies Wavelet Transform without multipliers" *Electronic Letters*, Vol. 27, pp. 171-173, 1991.

[5] H.Y.H. Chuang, H.J. Kim, and C.C. Li, "Systolic Architecture for discrete wavelet transform with orthogonal bases" in *Proc. SPIE Conf. on Applications of Artificial Intelligence X: Machine Vision and Robotics*, Vol. 1708, pp. 157-163, April, 1992.

[6] J. D. Hoyt, H. Wechsler. "The Wavelet Transform - A CMOS VLSI ASIC Implementation", *Proc. of 11th IAPR Int. Conf. on Pattern Recognition*, vol. 4, pp. 19-22, IEEE Comput. Soc. Press, 1992.

[7] K. K. Parhi, T. Nishitani, "VLSI architectures for discrete wavelet transforms", *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol 1, pp 191-202, Jun. 1993.

[8] H.Y.H. Chuang,, and L. Chen, "Scalable VLSI parallel pipelined architecture for discrete wavelet transform" in *Proc. SPIE Machine Vision Applications, Architectures, and System Integration II, Boston, MA*, pp. 66-73, September, 1993.

[9] H. H. Szu , C. C. Hsu, P. A. Thaker, M. E. Zaghloul, "Image wavelet transforms implemented by discrete wavelet chips", *Optical Engineering*, vol. 33, pp 2310-2325, Jul. 1994.

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

[11] A. Grzeszczak, M. K. Mandal, S. Panchanathan, T. Yeap, "VLSI Implementation of Discrete Wavelet Transform", *IEEE Trans. on VLSI Systems*, Vol. 4, No. 4, pp. 421-433, Dec., 1996.

[12] A. Grzeszczak, M. Mandal, S. Panchanathan, and T. Yeap, "VLSI Implementation of Discrete Wavelet Transform", *IEEE Trans. on very large scale integration systems*, Vol.4, No. 4, pp 421-433, December. 1996.

[13] T. C. Denk and K. K. Karhi, "Calculation of minimum number of register in 2-D Discrete Wavelet Transform using lapped block processing", *Int. Symposium on Circuits and Systems*, pp 22-81, 1994.

[14] C. Chakrabarti, M. Viswanath, R. M. Owens, "Architectures for Wavelets Transforms: A Survey", Journal of *VLSI Signal Processing*, Vol. 14, pp. 171-192, Dec., 1996.

[15] J. Villasenor, B. Belzer, and J. Liao, "Wavelet Filter Evaluation for Image Compression", *IEEE Trans. on Image Processing*, Vol. 4, pp 1053-1060, August 1995.

[16] I. Urriza, L.A. Barragan, J.I. Artigas, J.I. Garcia, D. Navarro. "Choice of word length in the design of an specialized hardware for lossless wavelet compression of medical images", Optical Engineering, Nov. 1997.

[17] M. L. Hilton, B. D. Jawerth, and A. Sengupta, "Compressing still and moving images with wavelets", *Multimedia Systems*, Vol. 2, pp 218-227, 1994.

[18] ES2 ECPD07 library Databook. European Silicon Structures.