Low-Complexity, Efficient 9/7 Wavelet Filters VLSI Implementation

Low-Complexity, Efficient 9/7 Wavelet Filters VLSI implementation

Maurizio Martina, *Member IEEE*, Guido Masera, *Member IEEE*

***Abstract*— This paper proposes a novel low-complexity, efficient 9/7 wavelet filters VLSI architecture for image compression applications. The 9/7 wavelet filters are widely used in different image compression schemes, such as the JPEG2000 image coding standard. Thus the implementation of efficient codecs is of great concern. The performance of a hardware implementation of the 9/7 filter bank depends on the accuracy of coefficients representation. However the greatest part of current implementations does not consider 9/7 wavelet filters starting from their original derivation, but as taps to be implemented. The aim of this work is to show that great complexity reduction with excellent performance can be achieved going through the derivation of the 9/7 taps values.**

***Index Terms*— low-complexity, 9/7 filter-bank, JPEG2000**

I. Introduction

The Discrete Wavelet Transform (DWT) has gained wide popularity due to its excellent decorrelation property [1]: many modern image and video compression systems embody the DWT as the transform stage (e.g. [2]). It is widely recognized that the 9/7 filters [3] are among the best filters for DWT based image compression [4]. In fact the JPEG2000 image coding standard [5], [6] employs the 9/7 filters as the default wavelet filters for lossy compression.

The performance of a hardware implementation of the 9/7 filter bank depends on the accuracy with which filter coefficients are represented. However high precision representation increases hardware resources and processing time. To reduce the complexity of the 9/7 filters the lifting scheme [7] can be adopted. Unfortunately the lifting scheme increases hardware timing accumulation due to its serial nature [8], so that for certain applications it cannot be employed. The flipping structure [8] is an attractive alternative to the standard lifting scheme DWT, since it reduces timing accumulation, however it still requires multiplications.

Complexity reduction can be achieved resorting to a filter bank implementation, in particular very good results can be obtained with the cascaded method proposed in [9] and exploited in [10]. The basic idea described in [9] is to minimize the number of bits required to represent the 9/7 coefficients. Since this operation would move filters zeros from their original position, the authors modify some terms to account for zeros compensation. Other techniques to reduce the complexity of filter banks implementations are based on distributed arithmetic (e.g. [11]) where only adders are employed.

The authors are with CERCOM (Center for Multimedia Radio Communications) - Dipartimento di Elettronica - Politecnico di Torino.

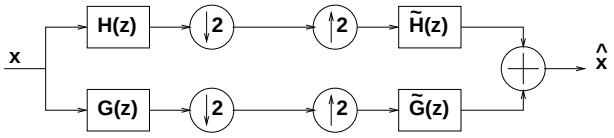

Fig. 1. Filter bank block scheme

Currently the compatibility of low complexity 9/7 filters implementation into standard image/video coding systems has not been stressed yet. The aim of this paper is to show that great complexity reduction can be achieved analyzing the 9/7 filters directly from their analytical derivation [3]. In particular, employing the proposed solution into a JPEG2000 encoder and decoding with a standard JPEG2000 decoder, the image quality loss is negligible. Moreover the complexity and the power consumption with respect to a standard implementation are nearly halved. It is worth noticing that the proposed methodology can be extended to other filters belonging to the 9/7 family as it will be discussed in the following. Compared to the best solution proposed in [9] our solution shows that T, the total number of non zero terms used when writing all the coefficients in sum or difference of powers of two (SPT), is the same: $T = 32$. Moreover our implementation shows that T can be nearly halved exploiting filters symmetry without any loss in terms of performance.

The paper is organized as follows: in section II the 9/7 filters theoretical derivation [3] is analyzed. In section III the proposed low-complexity architecture is derived and described. Experimental results, shown in section IV, prove that the proposed low-complexity solution grants excellent performance together with very interesting complexity and power consumption figures. Finally in section V some conclusions are drawn.

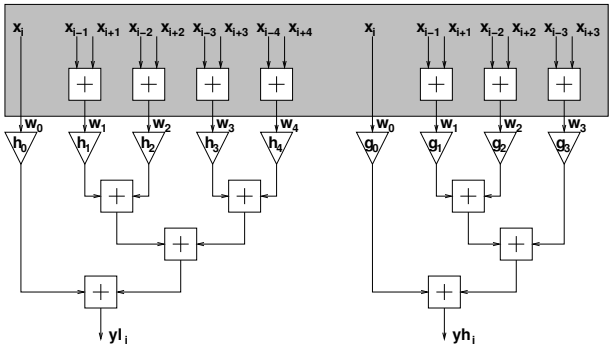

Fig. 2. Fast 9/7 direct implementation

TABLE I

COEFFICIENTS FOR THE 9/7 FILTERS: ORIGINAL, PROPOSED, 9 BITS QUANTIZED

n	$2^{-1/2}h[n]$	$2^{-1/2}\tilde{h}[n]$	$2^{-1/2}h'[n]$	$2^{-1/2}g'[n]$	$2^{-1/2}h^{(9)}_{qnt}[n]$	$2^{-1/2}g^{(9)}_{qnt}[n]$
0	$\frac{6}{8}K_1 - K_2 + \frac{2}{8}K_3$	$\frac{6}{8}J_1 - J_2$	$\frac{1}{2} + \frac{1}{16} + \frac{1}{32} + \frac{1}{128}$	$\frac{1}{2} + \frac{1}{16}$	0.601562500	0.556640625
±1	$\frac{4}{8}K_1 - \frac{7}{8}K_2 + \frac{4}{8}K_3$	$\frac{4}{8}J_1 - \frac{7}{8}J_2$	$\frac{1}{4} + \frac{1}{64}$	$\frac{1}{8} + \frac{1}{16} + \frac{1}{64} - \frac{1}{2}$	0.265625000	−0.296875000
±2	$\frac{1}{8}K_1 - \frac{4}{8}K_2 + \frac{6}{8}K_3$	$\frac{1}{8}J_1 - \frac{4}{8}J_2$	$-\frac{1}{16} - \frac{1}{64}$	$-\frac{1}{32}$	−0.080078125	−0.029296875
±3	$-\frac{1}{8}K_2 + \frac{4}{8}K_3$	$-\frac{1}{8}J_2$	$-\frac{1}{64}$	$\frac{1}{32} + \frac{1}{64}$	−0.017578125	0.044921875
±4	$\frac{1}{8}K_3$	0	$\frac{1}{32} - \frac{1}{256}$	0	0.025390625	0

II. THEORETICAL DERIVATION

Let's consider the filter bank shown in Fig. 1, where $H(z) = \sum_i^k h[i]z^{-i}$ and $G(z) = \sum_i^l g[i]z^{-i}$ are the low pass and high pass analysis filters with length k and l respectively, and $\tilde{H}(z) = \sum_i^{\tilde{k}} \tilde{h}[i]z^{-i}$ and $\tilde{G}(z) = \sum_i^{\tilde{l}} \tilde{g}[i]z^{-i}$ the low pass and high pass synthesis ones with length $\tilde{k}$ and $\tilde{l}$. It is well known that wavelet filter banks ought to satisfy the perfect reconstruction conditions [1]: $H(z)\tilde{H}(z)+G(z)\tilde{G}(z) = 2$ and $H(-z)\tilde{H}(z)+G(-z)\tilde{G}(z) = 0$. Imposing the biorthogonality condition together with filters symmetry ($\tilde{G}(z) = H(-z)$ and $G(z) = \tilde{H}(-z)$) we can rewrite the perfect reconstruction conditions as:

$$H(z)\tilde{H}(z) + \tilde{H}(-z)H(-z) = 2 \tag{1}$$

$$H(-z)\tilde{H}(z) + \tilde{H}(z)H(-z) = 0 \tag{2}$$

As shown in [3], writing the non distortion condition (1) on h and $\tilde{h}$ in terms of trigonometric polynomials, it becomes $H(\xi)\tilde{H}(\xi) + H(\xi+\pi)\tilde{H}(\xi+\pi) = 1$. Moreover, together with divisibility of H and $\tilde{H}$ respectively by $(1+e^{-j\xi})^k$ and $(1+e^{-j\xi})^{\tilde{k}}$ [3] it leads to:

$$H(\xi)\tilde{H}(\xi) = \cos(\xi/2)^{2l}\left[\sum_{p=0}^{l-1}\binom{l-1+p}{p} \cdot \sin(\xi/2)^{2p} + \sin(\xi/2)^{2l}R(\xi)\right] \tag{3}$$

where $R(\xi)$ is an odd polynomial in $\cos(\xi)$ and $2l = k + \tilde{k}$.

The 9/7 filters have been proposed in [3] as a particular case of trigonometric polynomials that satisfy (3) with $R \equiv 0$, $k = 4$ and $\tilde{k} = 4$. When $R \equiv 0$, $k = 4$ and $\tilde{k} = 4$, we obtain that (3) becomes $H(\xi)\tilde{H}(\xi) = \cos(\xi/2)^8\left[1 + 4\sin(\xi/2)^2 + 10\sin(\xi/2)^4 + 20\sin(\xi/2)^6\right]$. The term $\cos(\xi/2)^8$ can be split into two equal parts with degree 4. The polynomial in $\sin(\xi/2)$ can be considered as a third order equation and factorized into two polynomials with degree 2 and 4 respectively in order to obtain:

$$H(\xi)\tilde{H}(\xi) = 20\cos(\xi/2)^8\left[(r + \sin(\xi/2)^2) \cdot (a + b\sin(\xi/2)^2 + \sin(\xi/2)^4)\right] \tag{4}$$

where r is the real solution of the third order equation

$$1/20 + 4/20x + 10/20x^2 + x^3 = 0 \tag{5}$$

the product and the sum of the two complex conjugate solutions are respectively a and b. Thus (4) leads to:

$$H(\xi) = \frac{\cos(\xi/2)^4}{a}\left[a + b\sin(\xi/2)^2 + \sin(\xi/2)^4\right] \tag{6}$$

$$\tilde{H}(\xi) = \frac{\cos(\xi/2)^4}{r}\left[r + \sin(\xi/2)^2\right] \tag{7}$$

From (6) and (7) we can build filters coefficients [3] substituting: $\cos(\xi/2) = (e^{j\xi/2} + e^{-j\xi/2})/2$ and $\sin(\xi/2) = (e^{j\xi/2} - e^{-j\xi/2})/2j$.

Thus we obtain the coefficients shown in Table I where $K_1 = (a + b/2 + 3/8)/2a$, $K_2 = (b+1)/8a$, $K_3 = 1/32a$, $J_1 = (r + 1/2)/2r$, $J_2 = 1/8r$. Similar expressions can be found for other filters which satisfy (3).

III. PROPOSED ARCHITECTURE

The standard architecture for a fast 9/7 implementation is the so called, *direct implementation*, where samples that have to be multiplied by the same tap are first added together (w_i with $i \in 0,1,2,3,4$), then multiplied by the proper tap and finally partial results are combined with a tree adder (as depicted in Fig. 2) to obtain the result yl_i (yh_i). Since our application is image compression, the $\sqrt{2}$ term in Table I will appear twice in the computation: once during rows filtering and once during columns filtering. The JPEG2000 image coding standard embeds the $\sqrt{2}$ factors into the quantizer, so that in the following description they won't be considered anymore. In order to reduce the direct implementation architecture (DA) complexity, the analysis described in section II will be employed to derive: **1)** a preliminary architecture (PA), **2)** a low complexity architecture (LCA), **3)** a very low complexity architecture (VLCA).

Considering the two filters h and $\tilde{h}$ as vectors, we can represent them as the product of a matrix M and a vector K (for h) or J (for $\tilde{h}$). Besides h and $\tilde{h}$ symmetry suggests, for the sake of simplicity, to concentrate only on taps with index $n \geq 0$ (see Table I). Thus $h = M \cdot K$ and $\tilde{h} = M^{(5,3)} \cdot J$ where

$$M = \begin{pmatrix} \frac{6}{8} & -\frac{8}{8} & \frac{2}{8} \\ \frac{4}{8} & -\frac{7}{8} & \frac{4}{8} \\ \frac{1}{8} & -\frac{4}{8} & \frac{6}{8} \\ 0 & -\frac{1}{8} & \frac{4}{8} \\ 0 & 0 & \frac{1}{8} \end{pmatrix},\ K = \begin{bmatrix} K_1 \\ K_2 \\ K_3 \end{bmatrix},\ J = \begin{bmatrix} J_1 \\ J_2 \end{bmatrix} \tag{8}$$

and $M^{(5,3)}$ is the sub-matrix obtained from M removing the 5th row and the 3rd column. Being $G(z) = \tilde{H}(-z)$ a similar expression for g can be easily derived: $g = N^{(5,3)} \cdot J$ where

$$N = \begin{pmatrix} \frac{6}{8} & -\frac{8}{8} & \frac{2}{8} \\ -\frac{4}{8} & \frac{7}{8} & -\frac{4}{8} \\ \frac{1}{8} & -\frac{4}{8} & \frac{6}{8} \\ 0 & \frac{1}{8} & -\frac{4}{8} \\ 0 & 0 & \frac{1}{8} \end{pmatrix} \quad N^{(5,3)} = \begin{pmatrix} \frac{6}{8} & -\frac{8}{8} \\ -\frac{4}{8} & \frac{7}{8} \\ \frac{1}{8} & -\frac{4}{8} \\ 0 & \frac{1}{8} \end{pmatrix} \tag{9}$$

The matrix notation emphasizes how we can perform the filtering operations required by the 9/7 filters reducing the number of multiplications, as it will be detailed in next section.

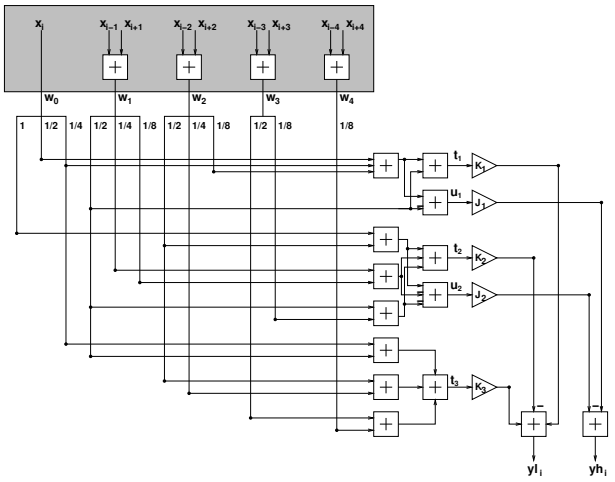

Fig. 3. Fast modified 9/7 direct implementation

A. Preliminary architecture

The PA represents the first modification with respect to the DA shown in Fig. 2. The basic idea is to perform the simple operations described by the matrix M first, then to multiply the results by the K values, and finally to add together the intermediate values to obtain yl_i, as described by (10).

$$\begin{aligned} yl_i &= h_0w_0 + h_1w_1 + h_2w_2 + h_3w_3 + h_4w_4 \\ &= (M_{1,1}K_1 + M_{1,2}K_2 + M_{1,3}K_3)w_0 + \\ &\quad +(M_{2,1}K_1 + M_{2,2}K_2 + M_{2,3}K_3)w_1 + \\ &\quad + \ldots + (M_{5,1}K_1 + M_{5,2}K_2 + M_{5,3}K_3)w_4 \\ &= (M_{1,1}w_0 + M_{2,1}w_1 + \ldots + M_{5,1}w_4)K_1 + \\ &\quad +(M_{1,2}w_0 + M_{2,2}w_1 + \ldots + M_{5,2}w_4)K_2 + \\ &\quad +(M_{1,3}w_0 + M_{2,3}w_1 + \ldots + M_{5,3}w_4)K_3 \quad (10) \end{aligned}$$

where w_i with $i \in 0, 1, 2, 3, 4$ are the values shown in Fig. 2. The same approach can be used for yh_i:

$$\begin{aligned} yh_i &= (N_{1,1}^{(5,3)}w_0 + \ldots + N_{4,1}^{(5,3)}w_3)J_1 + \\ &\quad +(N_{1,2}^{(5,3)}w_0 + \ldots + N_{4,2}^{(5,3)}w_3)J_2 \qquad (11) \end{aligned}$$

In Fig. 3 a block scheme for the PA is shown, where it can be observed that first the additions are applied and then the multiplications by K or J are performed. Given other filters that satisfy (3), a matrix expression similar to the ones shown in (10) and (11) can be obtained. So that an architecture similar to the one depicted in Fig. 3 can be derived for other filters.

B. Low-complexity architecture

The architecture shown in Fig. 3 reduces the number of multiplications from 9 to 5. Concentrating on the values K_1, K_2, K_3, J_1 and J_2 it is possible to further reduce the number of multiplications. Considering the real values obtained solving (5): $r = 0.34238409485837$, $a = 0.14603482098280$ and $b = 0.15761590514163$, we can reduce the number of multiplications approximating K and J values on a very small number of bit. As suggested in [9] better performance can be achieved granting that original filters zeros are kept as much as possible in their original position. Extensive simulations

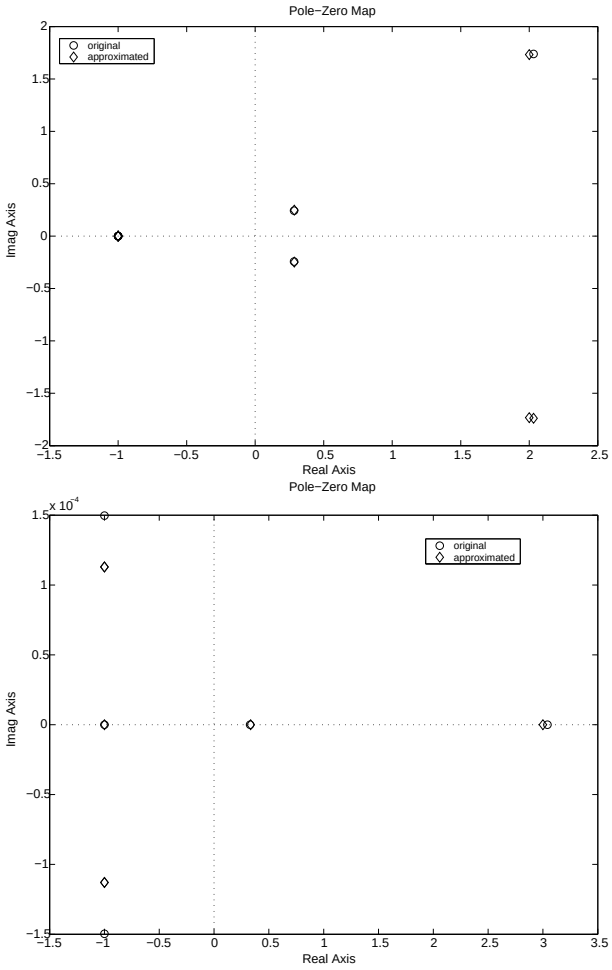

Fig. 4. Original and approximated h and $\tilde{h}$ filter zeros

show that K and J values can be approximated as: $K_1 = (a+b/2+3/8)/2a \simeq 2+1/16$, $K_2 = (b+1)/8a \simeq 1$, $K_3 = 1/32a \simeq 1/8 + 1/16 + 1/32$, $J_1 = (r+1/2)/2r \simeq 1 + 1/4$ and $J_2 = 1/8r \simeq 1/4 + 1/8$. These values can be obtained starting from K_1, K_2, K_3, J_1 and J_2 binary representation on 16 bits and then trying to approximate them on a small number of bits while granting that the zeros position is almost the same of the original filters. In Fig. 4 zeros positions for the original and the approximated filters are shown. As it can be observed zeros of the approximated filter are very close to the original ones. This approximation has a positive impact from the architectural point of view. In fact with the proposed approximation we can modify the architecture shown in Fig. 3 to obtain a low-complexity architecture. In Fig. 5 the multiplierless, low-complexity proposed architecture is depicted, where the multiplications have been substituted with additions.

C. Very low-complexity architecture

From Fig. 5 it is possible to further reduce the architecture complexity collapsing together some of the partial results. This operation can be obtained from Fig. 5 writing yl_i and yh_i as functions of w_i with $i \in 0, 1, 2, 3, 4$. More precisely considering $K_1^{'} = 2+1/16$, $K_2^{'} = 1$, $K_3^{'} = 1/8+1/16+1/32$,

$J_1^{'} = 1+1/4$ and $J_2^{'} = 1/4+1/8$, we can build the equivalent filters $h^{'}$ and $g^{'}$ as $h^{'} = M \cdot K^{'}$ or $g^{'} = N^{(5,3)} \cdot J^{'}$ (see Table I). The implementation of $h^{'}$ and $g^{'}$ leads to the architecture shown in Fig. 6 where multipliers are not employed and the number of adders is reduced with respect to the architecture shown in Fig. 5. It is worth noticing that $h^{'}$ and $g^{'}$ can be represented on 9 bits as 2 complement numbers, with 1 bit for the sign and the integer part and 8 bits for the fractional part. However $h^{'}$ and $g^{'}$ are slightly different from the filters we would obtain quantizing the original h and g on 9 bits ($h_{qnt}^{(9)}$ and $g_{qnt}^{(9)}$). This difference impacts on filters performance, as it will be detailed in section IV, since the position of $h^{'}$ and $g^{'}$ zeros is near h and g zeros, whereas the position of $h_{qnt}^{(9)}$ and $g_{qnt}^{(9)}$ zeros is rather far from h and g ones. Compared with the best solution proposed in [9] our solution exhibits $SPT_{h'} = 19$ and $SPT_{g'} = 13$. However, exploiting filters symmetry, the proposed architecture's complexity depends only on the SPT of taps with non negative index ($h_0^{'}$, $h_1^{'}$, $h_2^{'}$, $h_3^{'}$ $h_4^{'}$ and $g_0^{'}$, $g_1^{'}$, $g_2^{'}$, $g_3^{'}$). Thus the proposed very low-complexity architecture needs only $SPT_{h'} = 12$ and $SPT_{g'} = 8$. Moreover the architecture depicted in Fig. 6, exploiting some SPT terms that are common both to $h^{'}$ and $g^{'}$, further reduces the amount of hardware required. In fact as it is shown in Table I many terms are common to both $h^{'}$ and $g^{'}$, namely both $h_0^{'}$ and $g_0^{'}$ need the terms $1/2 + 1/16$.

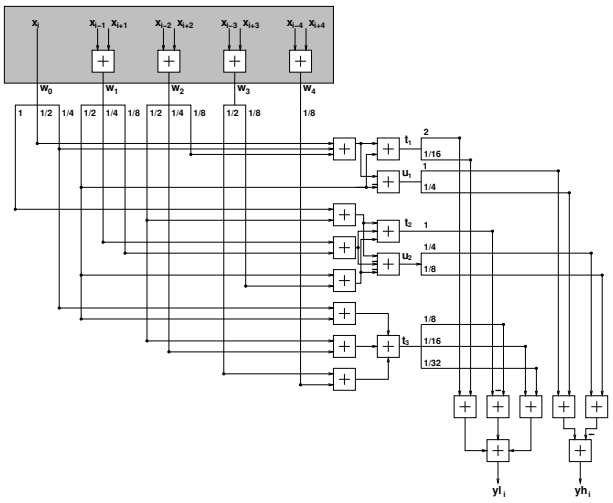

Fig. 5. Fast, low complexity, modified 9/7 direct implementation

IV. Experimental results

The proposed very low-complexity architecture has been tested inside the JPEG2000 image coding standard framework [5]. A free JPEG2000 codec written in C language, *openjpeg* [12] that is Class-1 Profile-1 compliant with the standard, has been employed for our tests. Five standard images have been used: 'Lenna' 256×256 (img1), 'Barbara' 512×512 (img2), 'Boat' 512×512 (img3), 'Golhill' 512×512 (img4) and 'Fingerprint' 512×512 (img5) [13]. The number of DWT decomposition levels (L) has been varied from 1 to 3 for 256×256 images and from 1 to 4 for 512×512 images. This corresponds to $\Lambda = L + 1$, where Λ is the number of DWT

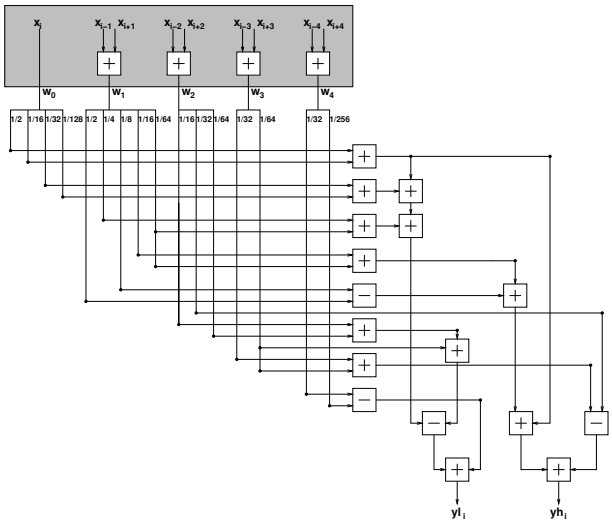

Fig. 6. Fast, very low complexity, 9/7 implementation

resolution levels required by *openjpeg*. Different compression ratios (ρ) have been imposed, namely 1:1, 8:1, 16:1, 32:1 and 64:1, precinct and code-block size are the encoder default values. First we evaluated the original *openjpeg* implementation performance, in terms of peak signal to noise ratio (PSNR), for the different L and ρ values on the aforementioned images. Then we substituted the standard 9/7 lifting scheme implementation of the encoder with proposed very low-complexity filter bank one, leaving the standard 9/7 lifting scheme at the decoder [13] (original *openjpeg* decoder). Finally we employed the $h_{qnt}^{(9)}$ and $g_{qnt}^{(9)}$ filters to show the loss of quality with respect to the proposed solution (see Table I). To obtain the results shown in Table II, a filter bank implementation must be employed. Since the equivalent lifting scheme (obtained converting $h'[n]$ and $g^{'}[n]$) is based on divisions [7], it requires a higher number of fractional bits. This is a critical aspect in fixed point DWT implementations as in *openjpeg*. Results shown in Table II prove that the proposed very low-complexity 9/7 filters are compatible with the JPEG2000 image coding standard. In fact given a JPEG2000 bit-stream generated via the proposed very low-complexity DWT, a standard JPEG2000 decoder can decode it granting high quality in terms of PSNR even at ρ=1:1. This adavantage stems from the position of our filter bank zeros. In fact, as shown in Fig. 4, they are extremely close to the 9/7 original zeros. Moreover the DA and the VLCA have been implemented in VHDL and synthesized on a 0.13 μm standard cells technology. Since all the proposed implementations have in common the first additions (the shaded part in Fig. 2, 3, 5 and 6) the w_i with $i \in 0, 1, 2, 3, 4$ have been considered as the input signals of the architectures. To make the comparison fair the architectures have been implemented as combinational blocks. Even if this choice can not achieve high clock frequencies, we are granted that further complexity in term of sequential elements is not added into the design. In fact with registers the logic synthesizer could perform retiming operations, that would make the comparison not fair. Thus the results obtained with the logic synthesizer `design_compiler` (by Synopsys) actually

TABLE II

PERFORMANCE COMPARISON OF THE PROPOSED LOW-COMPLEXITY DWT VERSUS THE STANDARD JPEG2000 IMPLEMENTATION AND A JPEG2000 IMPLEMENTATION WITH DWT TAPS REPRESENTED ON 9 BITS, PSNR FOR MULTIPLE DECOMPOSITION LEVELS AND COMPRESSION RATES

Image	L	*Openjpeg* [dB]					*Openjpeg* 9 bit [dB]					Low-complexity [dB]				
		1:1	8:1	16:1	32:1	64:1	1:1	8:1	16:1	32:1	64:1	1:1	8:1	16:1	32:1	64:1
img1	1	49.41	37.08	30.59	25.22	19.36	43.92	36.88	30.50	25.17	19.14	49.35	37.11	30.62	25.22	19.33
	2	49.16	38.50	33.29	28.72	24.61	39.38	36.33	32.56	28.47	24.48	48.79	38.50	33.30	28.74	24.60
	3	49.26	38.75	33.44	29.11	25.82	36.66	34.91	31.97	28.44	25.44	48.48	38.72	33.48	29.13	25.79
img2	1	49.57	35.77	29.48	24.05	20.63	45.16	35.82	29.63	24.12	20.55	49.48	35.75	29.52	24.06	20.61
	2	49.25	37.56	32.16	27.69	24.17	40.88	36.42	31.82	27.74	24.25	49.07	37.56	32.08	27.70	24.18
	3	49.21	37.87	32.78	28.75	25.31	38.15	35.29	31.83	28.36	25.38	48.80	37.78	32.74	28.75	25.32
	4	49.21	37.89	32.78	28.83	25.79	36.34	34.29	31.38	28.22	25.45	48.53	37.77	32.82	28.86	25.79
img3	1	49.42	37.66	32.64	28.22	23.39	44.62	37.52	32.58	28.29	23.49	49.45	37.61	32.63	28.28	23.39
	2	49.11	38.87	33.96	30.14	26.96	40.35	37.10	33.31	30.07	26.85	48.92	38.84	34.01	30.10	26.96
	3	49.06	39.04	34.46	30.88	27.86	37.38	35.54	32.93	30.17	27.57	48.72	39.03	34.47	30.91	27.96
	4	49.06	39.08	34.55	30.99	28.06	35.44	34.14	32.16	29.78	27.41	48.58	38.99	34.55	31.01	28.00
img4	1	49.80	35.77	31.68	27.55	23.14	44.76	35.70	31.70	27.54	22.93	49.77	35.79	31.70	27.57	23.13
	2	49.57	36.32	32.87	30.10	27.37	40.37	35.24	32.35	29.87	27.42	49.30	36.33	32.94	30.16	27.37
	3	49.55	36.45	33.15	30.53	28.43	37.54	34.12	31.94	29.77	28.00	49.12	36.40	33.16	30.52	28.43
	4	49.54	36.45	33.19	30.52	28.48	35.24	32.98	32.23	29.31	27.72	49.15	36.44	33.19	30.52	28.45
img5	1	49.68	35.80	31.73	27.76	17.72	42.89	35.60	31.68	28.02	17.58	49.63	35.78	31.73	27.95	17.72
	2	49.54	36.18	32.36	29.14	25.99	37.96	34.42	31.51	20.89	26.12	49.31	36.17	32.36	29.13	25.98
	3	49.51	36.26	32.45	29.48	26.79	34.82	32.71	30.65	28.44	26.20	49.32	36.24	32.46	29.47	26.78
	4	49.52	36.25	32.48	29.52	26.88	32.58	31.20	29.64	27.85	25.95	49.29	36.24	32.49	29.52	26.88

represent the complexity of the three architectures. In Table III post synthesis results are shown. The proposed architectures produce both a low pass and a high pass coefficient every clock cycle. Therefore for an $R \times C$ image $RC/2$ clock cycles are required to perform the 1D DWT. Considering that *openjpeg* represents the 9/7 taps on 13 bits it can be observed that the proposed low-complexity architecture shows interesting figures both in terms of complexity and power consumption. In fact compared with a direct implementation, the proposed one shows nearly the complexity of a 9 bit direct implementation, with the performance of a 13 bit direct implementation. However for the sake of completeness Table III shows results for different data widths. Namely samples are considered to be represented on $m = 12$, 13, 14, 15 and 16 bits, whereas taps for the DA on $n = 9$, 10, 11, 12 and 13 bits. Finally to compare the proposed VLCA with other architectures the competitive, multiplierless solution proposed in [11] have been implemented: both the former and the latter show the same latency ($RC/2$). Since [11] is derived from a 13 bits filter bank, it shows the same PSNR of the original *openjpeg* model. Table III shows that the proposed VLCA has a reduced complexity even compared with [11].

TABLE III

(A)REA [KGATES] AND (P)OWER [MW] FOR DA, VLCA AND [11]

		DA					VLCA	[11]
	m	n=**9**	n=10	n=11	n=12	n=**13**		
A	12	**2.77**	3.30	3.76	4.22	**4.74**	**1.95**	**3.78**
	13	**2.96**	3.53	4.02	4.50	**5.07**	**2.06**	**4.05**
	14	**3.15**	3.76	4.28	4.80	**5.41**	**2.18**	**4.32**
	15	**3.34**	3.99	4.54	5.09	**5.74**	**2.29**	**4.58**
	16	**3.53**	4.21	4.80	5.38	**6.01**	**2.41**	**4.85**
P	12	**10.85**	12.97	14.88	16.79	**18.87**	**6.89**	**17.75**
	13	**11.57**	13.82	15.90	17.87	**20.14**	**7.28**	**19.02**
	14	**12.31**	14.80	16.93	19.03	**21.46**	**7.74**	**20.29**
	15	**13.09**	15.67	17.94	20.15	**22.74**	**8.20**	**21.54**
	16	**13.78**	16.51	18.91	21.26	**23.98**	**8.66**	**22.80**

V. CONCLUSION

In this paper a very low-complexity, efficient 9/7 wavelet filters implementation, has been derived. A detailed analysis of the proposed solution architectural impact has been shown with performance and comparisons with the direct implementation. The proposed architecture shows to be compatible with the JPEG2000 image coding standard: very high quality can be achieved employing the proposed architecture in a low complexity JPEG2000 encoder and decoding with a standard JPEG2000 decoder. Moreover the proposed very low-complexity architecture shows noteworthy figures in terms of complexity and power consumption.

REFERENCES

[1] G. Strang and T. Q. Nguyen, *Wavelets and Filter Banks*. Wellesley-Cambridge, MA: Wellesley, 1996.

[2] D. Taubman, "High performance scalable image compression with EBCOT," *IEEE Trans. on Image Processing*, vol. 9, no. 7, pp. 1158–1170, Jul. 2000.

[3] M. Antonini et al. "Image coding using the wavelet transform," *IEEE Trans. on Image Processing*, vol. 1, no. 2, pp. 205–220, Apr. 1992.

[4] J. Liao et al. "Wavelet filter evaluation for image compression," *IEEE Trans. on Image Processing*, vol. 4, no. 8, pp. 1053–1060, Aug. 1995.

[5] M. Boliek, "JPEG 2000 Final Committee Draft," 2000.

[6] B. F. Wu and C. F. Lin, "Memory-efficient architecture for JPEG 2000 coprocessor with large tile image," *IEEE Trans. on Circuits and Systems-II*, vol. 53, no. 4, pp. 304–308, Apr. 2006.

[7] I. Daubechies and W. Sweldens, "Factoring Wavelet Transforms into Lifting Steps," Bell Labs, Lucent Technologies, Tech. Rep., 1996.

[8] C. T. Huang et al. "Flipping Structure: an efficient VLSI architecture for lifting-based discrete wavelet transform," *IEEE Trans. on Signal Processing*, vol. 52, no. 4, pp. 1080–1089, Apr. 2004.

[9] K. A. Kotteri et al. "Design of multiplierless, high-performace, wavelet filter banks with image compression applications," *IEEE Trans. on Circuits and Systems-I*, vol. 51, no. 3, pp. 483–494, Mar. 2004.

[10] K. A. Kotteri et al. "A comparison of hardware implementations of the biorthogonal 9/7 DWT: convolution versus lifting," *IEEE Trans. on Circuits and Systems-II*, vol. 52, no. 5, pp. 256–260, May 2005.

[11] M. Alam et al. "Efficient distributed arithmetic based DWT architecture for multimedia applications," in *IEEE International Workshop on System-on-Chip for Real-Time Applications*, 2003.

[12] "http://www.openjpeg.org."

[13] M. Martina, "Low Complexity 9/7 Wavelet: Modified OpenJPEG model," downloadable at www.vlsilab.polito.it/∼martina.