International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:2, No:4, 2008

Design Techniques and Implementation of Low Power High-Throughput Discrete Wavelet Transform Tilters for JPEG 2000 Standard

Grigorios D. Dimitroulakos, N. D. Zervas, N. Sklavos and Costas E. Goutis

[1]

***Abstract*—**In this paper, the implementation of low power, high throughput convolutional filters for the one dimensional Discrete Wavelet Transform and its inverse are presented. The analysis filters have already been used for the implementation of a high performance DWT encoder [15] with minimum memory requirements for the JPEG 2000 standard. This paper presents the design techniques and the implementation of the convolutional filters included in the JPEG2000 standard for the forward and inverse DWT for achieving low-power operation, high performance and reduced memory accesses. Moreover, they have the ability of performing progressive computations so as to minimize the buffering between the decomposition and reconstruction phases. The experimental results illustrate the filters' low power high throughput characteristics as well as their memory efficient operation.

***Keywords*—**Discrete Wavelet Transform; JPEG2000 standard; VLSI design; Low Power-Throughput-optimized filters

I. INTRODUCTION

THE Discrete Wavelet Transform (DWT) has been introduced as a highly efficient and flexible method for subband decomposition of signals [1]. In digital signal processing, good algorithmic performance especially in the fields of image and video compression has been demonstrated [2], [3], [4]. The inclusion of the DWT in contemporary multimedia compression standards, such as the JPEG2000 [5] and MPEG-4 [6], has lead to intensive research efforts for improving the implementation aspects of the transform.

Our paper is focused on the 1 Dimensional (1D) convolutional DWT which is used for the realization of the seperable 2 Dimensional (2D) DWT. In detail the separable 2D-DWT is realized with two 1D-DWT filterings one along the rows and one along the columns of the input image. A diversity of filters has been proposed for the 1D-DWT [7]. The two extremes of filter types are the serial and parallel ones. Serial filters, which are efficient in terms of area, are characterized by low throughput making their use in streaming applications inefficient. On the other hand, parallel filters dominate in streaming applications as they are characterized by high throughput. However, there are quite many opportunities for improvements that can be made to them.

Many architectures that implement the Two-Dimensional separable Forward (2D-DWT) and Inverse DWT (2D-IDWT) in order to be applied on 2D signals have been presented in the past [7], [8], [9] and [10] . These architectures are consisting of filters for performing the 1D-DWT and memory units for storing the results of the transformation. Due to the fact that streaming multimedia applications - in which the DWT is present - are characterized by high throughput requirements, this imposes the need for optimizing the design of the filters in terms of speed. Moreover, portable multimedia devices require low power consumption for increasing the battery lifetime and this can be achieved by minimizing the storage size and number of memory accesses [11].

In this paper, the design techniques and VLSI realization of parallel Low Power Throughput Optimized (LPTO) convolutional filters that implement the 1D-DWT and the 1D-IDWT of a signal are presented. These filters are based on the DWT as specified in JPEG2000 standard [5], [13]. The filters studied, are the 9/7 and 5/3 for odd taps and the 10/18, 2/10, and 2/6 for even taps. These filters produce two subband samples in every cycle taking as input two signal samples. Also, they don't need separate subsampling and upsampling units for computing the DWT and IDWT, respectively. The proposed synthesis filters employ an adder less than the original Mallat's scheme [12] and the introduced scheme in [10]. Furthermore, exploiting the symmetric nature of the proposed filters the signal extension prior to actual filtering is avoided. Hence, the memory accesses are reduced. Finally, the need for buffering and reordering between the analysis and synthesis filters is eliminated because the samples are consumed in the way they are produced [14]. The aforementioned characteristic makes them useful for streaming applications. The experimental results and the work in [15] illustrate the efficiency of the proposed implementation for the aforementioned filters in terms of performance, power, and memory access efficiency.

The rest of the paper is organized as follows. In section 2 the related work in developing hardware architectures for the DWT is presented. Section 3 depicts the background theory of wavelet transform. In section 4 the architecture of the analysis

[1] Grigorios D. Dimitroulakos is with the Electrical and Computer Engineering Dep. Of the University of Patras, Greece. (corresponding author to provide phone: +30 2610 993421; fax: +30 2610 994798; e-mail: dhmhgre@ ee.upatras.gr)

N. D. Zervas, was with the the Electrical and Computer Engineering Dep. Of the University of Patras, Greece. (e-mail: zervas@alma-tech.com)

N. Sklavos was with the the Electrical and Computer Engineering Dep. Of the University of Patras, Greece. (e-mail: nsklavos@ieee.org)

Costas E. Goutis is a professor in he Electrical and Computer Engineering Dep. Of the University of Patras, Greece. (goutis@ee.upatras.gr)

and synthesis filters is shown. The results for the VLSI synthesis of the analysis and synthesis filters are given in section 5. Finally, section 6 concludes the paper

II. RELATED WORK

Numerous architectures have been proposed for computing the 1D-DWT [7], [8], [9], [10], [16], [17], [18], [19] and [20]. The 1D-DWT architectures can, in principle, be extended to architectures for computing the separable 2D-DWT. This is because the separable 2D-DWT can be computed by one 1D-DWT filtering along the rows followed by a 1D-DWT filtering along the columns. In addition, the non-separable 2D-DWT process 2D data directly. Due to the fact that the minimizations of latency and of storage requirements are important goals for most streaming multimedia applications, mapping 1D-DWT architectures to 2D-DWT architectures is not a trivial issue.

Lewis and Knowles [21] were the first to propose an architecture for the 2D-DWT. Their architecture was tuned to the Daubechies four-tap filters, so it suffered from scalability since it is strongly dependent on the limited properties of the filters used. Chakrabarti and Vishwanath [7] have proposed a scalable architecture for the encoder based on the non-separable 2D-DWT. Their architecture consists of two parallel computation units of size K^2 and a storage unit of size $\approx NK$. A parallel computation unit of size M consists of M multipliers and a tree of adders to add the M products. Vishwanath *et al.* [9] have proposed an architecture for separable 2D-DWT, which consists of two systolic arrays of size K, two parallel computational units of size K, and a storage unit of size $\approx N(2K+J)$. A drawback of this architecture is that two rows of inputs are fed into the two systolic arrays every two cycles and as a result, an additional data converter is required to convert the raster scan input (one per cycle) into two per two cycles output. Chakrabarti and Mumford [16] introduced an architecture for the analysis (synthesis) filters based on the 2D-DWT, together with two scheduling algorithms for computing the forward (inverse) 2D-DWT. The goal was to minimize the storage requirements and keep the data-flow regular.

III. PRELIMINARIES

The 1D-DWT and the 1D-IDWT of an input signal $x[n]$ is implemented by the filter bank shown in Fig.1. The 1D-DWT is a two-channel subband decomposition of an input signal $x[n]$ that produces two subband coefficients $y_0[n]$ and $y_1[n]$ for one-stage of decomposition [12] according to the following equations.

$$y_1[n] = \sum_k x[k]h[2n-k] \tag{1}$$

$$y_0[n] = \sum_k x[k]w[2n-k] \tag{2}$$

The 1D-IDWT is a two-channel subband reconstruction, taking as inputs the $y_0[n]$ and $y_1[n]$ coefficients and producing the $x'[n]$ signal, where $x'[n]$ has to be equal or a very good approximation of the input signal $x[n]$. The reconstruction of the original is done according to the following equations.

$$x'[n] = \sum_k \left[y_1[k] \cdot h'[n-2 \cdot k] + y_0[k] \cdot w'[n-2 \cdot k] \right] \tag{3}$$

The $h'[n]$ and $w'[n]$ are high-pass filters and low-pass filters respectively, used in the analysis (synthesis) section. After the filtering operations in analysis (synthesis) section the produced sequences are downsampled (upsampled) by a factor of 2.

The four-channel subband decomposition of 2D-DWT is obtained by the successive applications of two-channel decomposition of 1D-DWT in the rows and columns of the input image. Similarly, 2D-IDWT is obtained by two channel reconstruction of 1D-IDWT along the rows and columns. In Fig. 2 the 2D-DWT and 2D-IDWT filter banks are illustrated for two stages of decomposition and reconstruction.

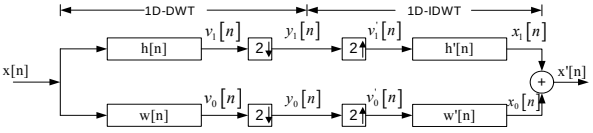

Fig. 1 One-stage of subband decomposition and reconstruction.

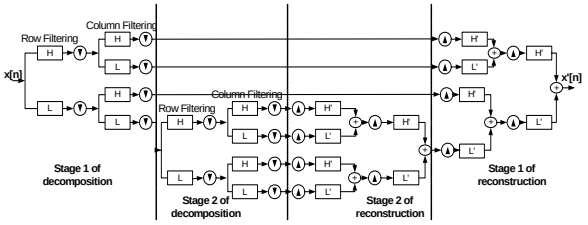

Fig. 2 2D-DWT and IDWT filter banks for J=2 stages.

Throughout this paper, we assume that: a) The signal length is N samples and b) the maximum number of filter taps among the high-pass and low-pass filters is K. Note that K is the maximum number of taps of the high- and low-pass filters. For example, for the 5/3 filter K=5, while for the 9/7 filter K=9.

IV. PROPOSED LPTO FILTERS DESCRIPTION

In this section, the novel LPTO filters for computing the 1D-DWT and 1D-IDWT, are presented. The filters are based on the well-known convolution equations (1), (2) and (3) described in the previous section. These equations were modified for including the subsampling and upsampling operations inside the filters. Also, there is a different filter design depending on the number of filter coefficients and on the number of the input signal samples. The special features of each filter and their design are illustrated in the following paragraphs.

The equations describing the DWT as implemented in the proposed filters are described as follows. Let $x[n]$ denote the 1D sequence of input samples and let $y[n]$ denote the one 1D sequence of interleaved subband samples, where $0 \le n < N$. The low-pass subband corresponds to the even samples $y[2n]$, while the high-pass subband corresponds to the odd samples $y[2n+1]$. The relevant analysis operation in the forward 1D-DWT in terms of inner products is expressed as:

$$y[2n_l] = \sum_k w[k] \cdot x[2n_l + k] \quad (4)$$

$$y[2n_h + 1] = \sum_k h[k] \cdot x[2n_h + k + 1] \quad (5)$$

where, $h[k]$ and $w[k]$ denote the high-pass and low-pass analysis filters respectively and n_l and n_h are the indexes for the low-pass and high-pass coefficients. The synthesis operation in the 1D-IDWT reproduces the original signal $x[n]$ from the interleaved subband sequence $y[n]$ according to the following equation:

$$x[n] = \sum_k [y[2k] \cdot w'[n-2k] + y[2k+1] \cdot h'[n-(2k+1)]] \quad (6)$$

where, $h'[k]$ and $w'[k]$ are the high-pass and low-pass synthesis filters respectively.

Equation (6) has been modified for designing synthesis filters with an adder less than the original Mallat's implementation [12]. With this modification the synthesis filters have the same structure as the analysis ones. Equation (6), describing the synthesis operation, can be modified as:

$$x[2l] = \sum_k \left[y[2k] \cdot w'[2l-2k] + y[2k+1] \cdot h'\left[2l-(2k+1)\right] \right], \quad n = 2l \quad (7)$$

$$x[2l+1] = \sum_k \left[y[2k] \cdot w'[2l+1-2k] + y[2k+1] \cdot h'[2l-2k] \right], n = 2l+1 \quad (8)$$

where $0 \le l \le (N-2)/2$.

Two new functions are defined such that:

$wt'(m) = w'(m)$ and $ht'(m) = h'(m)$, if m is even

$wt'(m) = h'(m)$ and $ht'(m) = w'(m)$, if m is odd (9)

Then (7) and (8) are transformed to:

$$x[2l] = \sum_k \left[y[2k] \cdot wt'[2l-2k] + y[2k+1] \cdot wt'\left[2l-(2k+1)\right] \right], \; n = 2l$$

$$x[2l+1] = \sum_k \left[y[2k] \cdot ht'[2l+1-2k] + y[2k+1] \cdot ht'[2l-2k] \right], n = 2l+1 \Rightarrow$$

$$x[n] = \sum_k \left[y[2k] \cdot wt'[n-2k] + y[2k+1] \cdot wt'\left[n-(2k+1)\right] \right], \; n = 2l$$

$$x[n] = \sum_k \left[y[2k] \cdot ht'[n-2k] + y[n] \cdot ht'\left[n-(2k+1)\right] \right], \; n = 2l+1 \Rightarrow$$

$$x[n] = \sum_k \left[y[k] \cdot wt'[n-k] \right] \quad n = 2l \quad (10)$$

$$x[n] = \sum_k \left[y[k] \cdot ht'[n-k] \right] \quad n = 2l+1 \quad (11)$$

From (10) and (11) it is concluded that the synthesis operation can be done as a convolution of the interleaved sequence of subband samples without the necessity of the addition operation of (6), and with the impulse response of the filters defined by (9). Also, there is no need for buffering because sequence $y[k]$ is consumed in the way it is produced.

The equations (4), (5) for the analysis operation and (10), (11) for the synthesis operation, can be implemented mainly with two hardware designs using parallel filters. The first design [8], called to hereafter as conventional architecture, consists of an input delay line equal to the number N_{taps} of filter coefficients (taps). The same N_{taps} multipliers are used for computing the low- and high-pass coefficients. The conventional architecture implements analysis and synthesis operations in an interleaved manner. Specifically, for the even clock cycles the multipliers are fed with the taps of the low-pass filter, while for the odd cycles the same multipliers are fed with the constant coefficients of the high-pass filter. In this way, a pair of low- and high-pass coefficients is produced every two clock cycles.

The proposed architecture, called hereafter LPTO, consist of a modified delay line that receives two input samples per clock cycle and a separate data path for high-pass and low-pass filtering. So, LPTO architecture produces a pair of low- and high-pass coefficient every clock cycle, resulting in greater speed than the conventional architecture. Also, although it is expected that the conventional architecture occupies less area than the LPTO, this is not always the case. This is because with the LPTO architecture, an additional optimization is enabled called strength reduction [23]. Specifically, multiplication among a variable (input sample) and a constant can be easily reduced to a number of shift-and-add operations, resulting this way in a much smaller implementation. For example, a multiplication times 3 is reduced to a left shift by one and an increment by one. Fig. 3 illustrates the conventional architecture, while in Fig. 4 the LPTO architecture is given, both for the case of 4/3 filter.

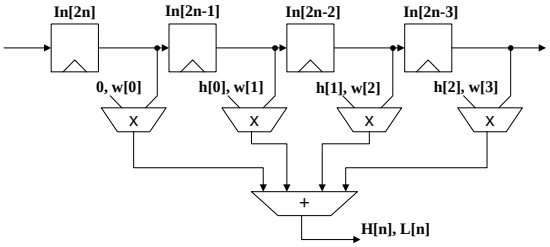

Fig. 3 Conventional filter architecture.

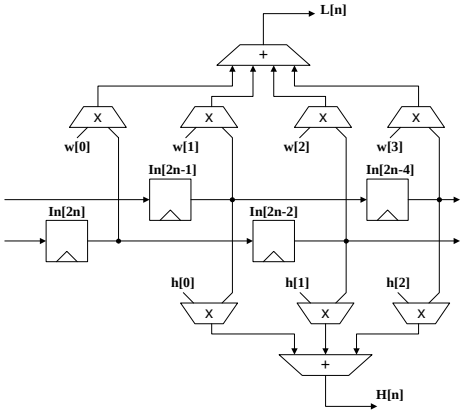

Fig. 4 LPTO filter architecture.

A. Symmetric Extension

Due to the finite length of the input signal sequences, problems arise when the filter processes the signal's boundaries. This problem can be solved by extending the signal at the boundaries as much as needed to complete the filtering operation. In the proposed filter's design the symmetric extension is adopted.

The symmetric extension is a simple method for extending a finite length signal [22]. In this method the signal is extended so as it becomes periodic and symmetric. The type of the symmetric extension is denoted by (x, y), where x and y is the number of times the first and the last sample of the signal is repeated, respectively. Two special cases of

extension are the Whole Sample (WS) symmetry and the Half Sample (HS) symmetry. The WS extends the signal having as center of symmetry the last sample of the signal sequence, so it is a (1, 1) symmetry. The HS extends the signal having as center of symmetry the half of the distance between the last sample and the first sample of the extension, so it is a (2, 2) symmetry. The two basic kinds of symmetric extension are illustrated in Fig. 5.

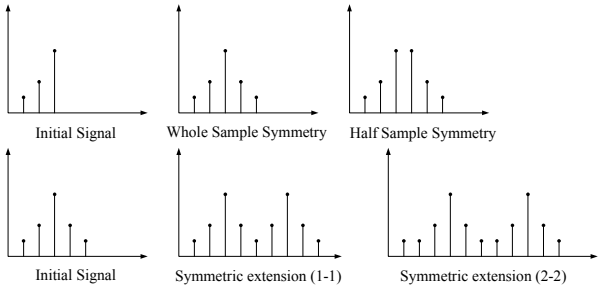

Fig. 5 Types of symmetric extension

The filter's operation consists of three phases: a) *Initialization phase*: initially, the filter loads its delay line with the appropriate number of samples in order start the filtering. Since these samples are present in the delay line, symmetric extension takes place. b) *Filtering phase*: the filter processes the input signal samples present in the delay line. c) *Finalization phase*: at the end of filter's operation, when the input samples are fully consumed, the input signal must be extended to perform the remaining filtering operations imposed by the algorithm.

The symmetric extension during phases (a) and (c) for each type of filter is described in the following.

For the case of odd-tap filters the input signal $x[n]$ is extended using the (1,1) symmetry, described by the equations:

$$x[-n] = x[n] \;\; x[N-1+1] = x[N-1-n] \tag{12}$$

It can be easily shown that the produced sequence of coefficients (low- and high-pass) $y[n]$ has also (1, 1) symmetry, as in the case of the input signal $x[n]$. So, the time relations are:

$$y[n] = y[-n], \;\; y[N-1+1] = y[N-1-n] \tag{13}$$

Thus, we need N subband coefficients when the number of samples of the input signal is even, and N-1 subband coefficients when the number of samples is odd.

For the case of even-tap filters the input signal $x[n]$ is extended using the (2, 2) symmetry, described by the equations:

$$x[-n] = x[n-1] \;\; x[N-1+n] = x[N-n] \tag{14}$$

It can be easily shown that the produced sequence of coefficients (low- and high-pass) $y[n]$ has a symmetry which, is described in the following:

For the case of $x[n]$ having even number of samples, in the left edge of the $y[n]$ sequence the time relations are:

$$y[-2n] = y[2n-2], \;\; y[-2n+1] = -y[2n-1] \tag{15}$$

while in the right edge the relations are:

$$y[N+2n] = y[N-2n-2], \;\; y[N+2n+1] = -y[N-2n-1] \tag{16}$$

For the case of $x[n]$ having odd number of samples, in the right edge of the $y[n]$ sequence the time relations are:

$$y[N+2n-1] = y[N-2n-1], \;\; y[N+2n] = -y[N-2n] \tag{17}$$

Thus, we need N subband coefficients when the number of samples of the input signal is even, and N-1 subband coefficients when the number of samples is odd.

B. Analysis Filters

For the implementation of the analysis operation, described by equations (4) and (5), two types of filters are proposed. The first one is for the case of odd number of taps and it is called Odd Tap Analysis Filter (OTAF). This type of filters can also be represented as $2M_l+1/2M_l-1$ based on the number of taps of the lowpass and highpass filter respectively. The implementation refers to the 5/3 ($M_l = 2$) and 9/7 ($M_l = 4$) analysis filters of the JPEG2000. The second one is for the case of even number of taps and it is called Even Tap Analysis Filters (ETAF). The implementation refers to the 10/18, 2/10 and 2/6 analysis filters of the JPEG2000. Their detailed implementations are described in the following.

In OTAFs the delay lines consist totally of $2M_l+1$ registers, which is equal to the number of taps of the low-pass analysis filter. The delay line holding the odd index samples has M_l registers, while the delay line holding the even index samples has M_l+1 registers. For the 5/3 filter $M_l=2$, while for the 9/7 filter $M_l=4$. Assuming that the registers are counted from left to right, the M_l register will feed the central coefficient of the low-pass filter. For the first filtering operation, M_l+1 samples are needed. The rest of the samples are image copies of the existing samples in reference to the first sample of the input sequence. The register transfer relation of the (1, 1) symmetric extension is:

$$R_{Ml+n} <= R_{Ml-n-2}, \text{ for } n = [1, M_l] \tag{18}$$

where R_{-1} : comes directly from input $x[2n-1]$, and
R_{-2} : comes directly from input $x[2n]$

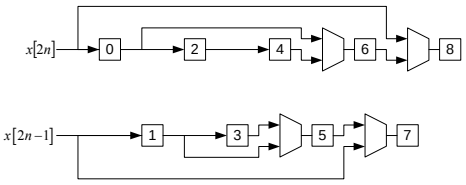

Fig. 6 Structure for the (1, 1) symmetric extension in 9/7 analysis filter.

The four multiplexers in Fig.6 implement the (1, 1) symmetric extension for the 9/7 filter. For Fig.6 and the similar figures that show the structure of the filter's delay line, the numbers in the small rectangles indicate the enumeration of the delay line's registers. For example, number 6 corresponds to register R_6.

When the input signal's samples are consumed the filtering must continue for some more cycles until the last sample of the input signal reaches the last register of its respective delay line. This stage of the filtering is called final filtering stage and is a part of the filtering process for both even- and odd-tap

filters of both analysis and synthesis operations. The required signal samples to complete the filtering process are generated according to [5]. Depending on whether the number of input samples is even or odd, the mathematical relations of symmetric extension at the final stage of filtering is:

for even samples: $R_0 <= R_{4n-4}$, for $n = [1, M_l/2]$
$R_1 <= R_{4n-5}$, for $n = [2, M_l/2]$ (19)
for odd samples: $R_0 <= R_{4n-2}$, for $n = [1, M_l/2]$
$R_1 <= R_{4n-3}$, for $n = [1, M_l/2]$ (20)

For the case of the 9/7 filter, the circuit realization of the symmetric extension at the final filtering stage for even number of input samples is shown in Fig.7, while for the odd number of input samples is shown in Fig. 8.

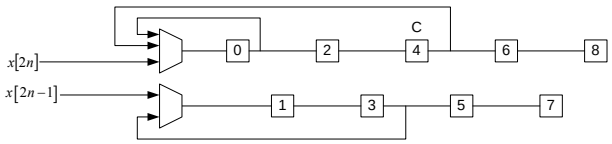

Fig. 7 (1, 1) symmetric extension at the final filtering stage for even number of samples in 9/7 analysis filter.

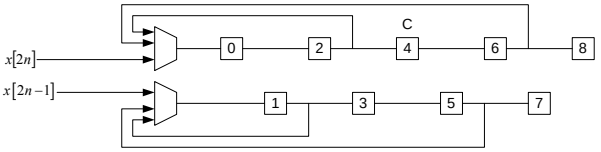

Fig. 8 (1, 1) symmetric extension at the final filtering stage for odd number of samples in 9/7 analysis filter.

According to equations (18),(19) and (20) it is concluded that:

a) (K-1)/2 samples are symmetrically extended into the filter's delay line during the initialization phase.

b) (K-3)/2, (K-1)/2 samples are symmetrically extended into the filter's delay line during the finalization phase, for even or odd samples respectively.

Hence, totally K-2, K-1 samples are symmetrically extended into the filter's delay line, for even or odd samples respectively.

In ETAFs the delay lines consist totally of $2M_h$ registers, which is equal to the number of taps of the high-pass analysis filter. The number of registers in each delay line is M_h. For the 10/18 filter M_h=9, for the 2/10 M_h=5 and for M_h=3. Assuming that the registers are counted from left to right, the center of the low-pass filter will be fed from the M_h register. In this case, M_h samples are needed for the first filtering. The rest ones are produced with a (2, 2) symmetric extension. The register transfer relation of this extension is:

$$R_{Mh+n} <= R_{Mh-n-1}, \text{ for } n = [1, M_h-1] \quad (21)$$

The multiplexers in Fig.9 implement the symmetric extension for the 10/18 analysis filter.

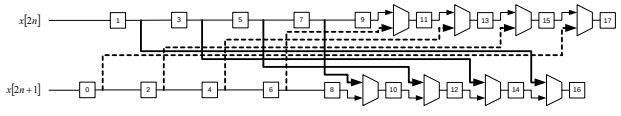

Fig. 9 Structure for the (2, 2) symmetric extension in 10/18 analysis filter.

The symmetric extension at the final stage of filtering is done according to [5]. Depending on whether the number of input samples is even or odd the mathematical relation of the extension at the final filtering stage is:

for even samples: $R_0 <= R_{4n-3}$, for $n = [1, (M_h - 1)/2]$
$R_1 <= R_{4n-4}$, for $n = [1, (M_h - 1)/2]$ (22)
for odd samples: $R_0 <= R_{4n-5}$, for $n = [1, (M_h - 1)/2]$
$R_1 <= R_{4n-6}$, for $n = [2, (M_h - 1)/2]$ (23)

where, R_{-1} comes directly from input $x[2n]$.

The circuit realizations of the symmetric extension at the final filtering stage for even and odd number of input samples are shown in Fig. 10 and Fig. 11, respectively.

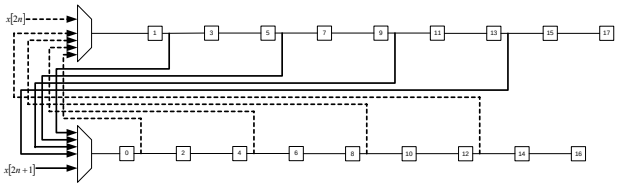

Fig. 10 (2, 2) symmetric extension at the final filtering stage for even number of samples in 10/18 analysis filter.

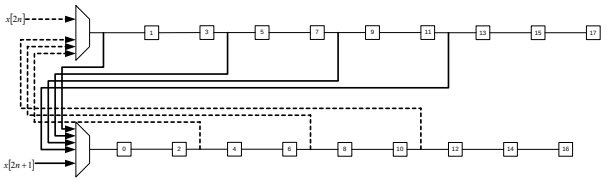

Fig. 11 (2, 2) symmetric extension at the final filtering stage for odd number of samples in 10/18 analysis filter.

According to equations (21),(22) and (23) it is concluded that:

a) $(K-2)/2$ samples are symmetrically extended into the filter's delay line during the initialization phase.

b) $(K-2)/2$, $(K-4)/2$ samples are symmetrically extended into the filter's delay line during the finalization phase, for even or odd samples respectively.

Hence, totally $K-2$, $K-3$ samples are symmetrically extended into the filter's delay line, for even or odd samples respectively.

C. Synthesis Filters

For the implementation of the synthesis operation, described by equations (10) and (11), two types of filters are proposed as in the case of analysis operation. The first one is for the case of odd number of taps and it is called Odd Tap Synthesis Filter (OTSF). It is used for the 3/5 and 7/9 synthesis filters in JPEG2000. The second one is for the case of even number of taps and it is called Even Tap Synthesis Filters (ETSF). It is used for the 18/10, 10/2 and 6/2 synthesis filters in JPEG2000. Their detailed implementations are described in the following two sub-sections.

In OTSFs the delay lines consist totally of $2M_l$ +1 registers, which is equal to the number of taps of the respective low-pass analysis filter. The delay line holding the odd index samples has M_l+1 registers, while the delay line holding the even index samples has M_l registers. For the 3/5 synthesis filter M_l=2, and for the 7/9 filter M_l=4. Assuming that the

registers are counted from left to right, the M_l+1 register will feed the central coefficient of the low-pass filter. Before the first filtering the delay line must be filled with the appropriate samples. Totally M_l+2 samples are needed for the first filtering. The rest of the samples are produce by (1, 1) symmetric extension. The register transfer relation of the extension is:

$$R_{Ml+n+1} <= R_{Ml-n-1}, \text{ for } n = [1, M_l\text{-}1] \quad (24)$$

The three multiplexers in Fig.12 realize the symmetric extension for the 7/9 synthesis filter.

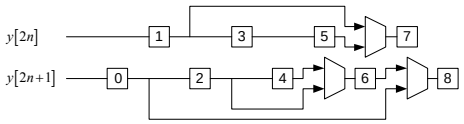

Fig. 12 Structure for the symmetric extension in 7/9 synthesis filter.

At the last stage of filtering, depending whether the number of input samples is even or odd the mathematical relation of symmetric extension at the final filtering stage is:

for even samples: $R_0 <= R_{4n-2}$, for $n = [1, M_l/2]$

$$R_1 <= R_{4n-3}, \text{ for } n = [1, M_l/2] \quad (25)$$

for odd samples: $R_0 <= R_{4n-4}$ for $n = [1, M_l/2 + 1]$

$$R_1 <= R_{4n-5}, \text{ for } n = [2, M_l/2 + 1] \quad (26)$$

For even number of input samples the circuit realizing the symmetric extension at the final filtering stage is shown in Fig. 13, while in Fig.14 the extension for odd number of input samples is shown.

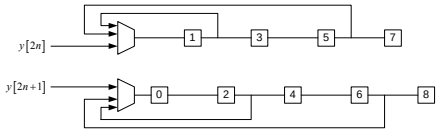

Fig. 13 Symmetric extension at the final filtering stage for even number of samples in 7/9 synthesis filter.

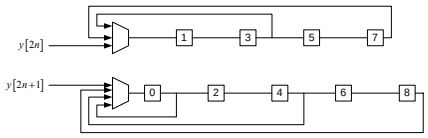

Fig. 14 Symmetric extension at the final filtering stage for odd number of samples in 7/9 synthesis filter.

According to equations (24), (25) and (26) it is concluded that:

a) (*K-3*)/*2* samples are symmetrically extended into the filter's delay line during the initialization phase.

b) (*K-1*)/*2*, (*K-3*)/*2* samples are symmetrically extended into the filter's delay line during the finalization phase, for even or odd samples respectively.

Hence, totally *K-2*, *K-3* samples are symmetrically extended into the filter's delay line, for even or odd samples respectively.

In ETSFs the delay lines consist totally of $2M_h$ registers, which is equal to the number of taps of the high-pass analysis filter. The number of registers in each delay line is M_h. For the 18/10 filter M_h=9, while for the 10/2 M_h=5. Assuming that the registers are counted from left to right, the M_h-1 register will feed the central coefficient of the transformed low-pass filter. For the first filtering, M_h+1 samples are needed, the rest samples are produced by symmetric extension according to [5]. The register transfer relation is:

$R_{Mh+n-1} <= R_{Mh-n-1}$,

$$R_{Mh+2n-2} <= -R_{Mh-2n-2}, \text{ for } n=2k, \text{ where } n = [1, (M_h\text{-}1)/2] \quad (27)$$

The multiplexers in Fig.15 realize the extension operation for the 18/10 synthesis filter.

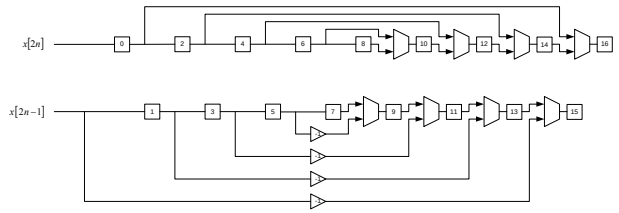

Fig. 15 Structure for the symmetric extension in 18/10 synthesis filter.

The samples that are needed at the final filtering stage are generated by symmetric extension according to the following register transfer relations:

for even samples: $R_0 <= R_{4n-4}$, for $n = [1, (M_h - 1)/2]$

$$R_1 <= -R_{4n-7}, \text{for } n = [2, (M_h - 1)/2] \quad (28)$$

for odd samples: $R_0 <= R_{4n-2}$, for $n = [1, (M_h - 1)/2]$

$$R_1 <= -R_{4n-5}, \text{ for } n = [1,(M_h - 1)/2] \quad (29)$$

with R_{-1} taking the value 0.

The circuit realizations of the symmetric extension at the final filtering stage for even and odd number of input samples are shown in Fig. 16 and Fig. 17, respectively.

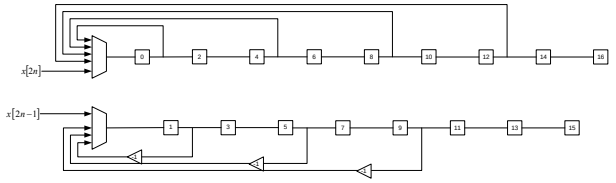

Fig. 16 Symmetric extension at the final filtering stage for even samples in 18/10 synthesis filter.

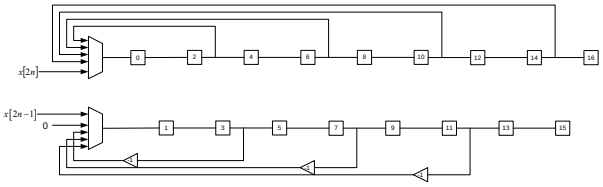

Fig. 17 Symmetric extension at the final filtering stage for odd samples in 18/10 synthesis filter.

According to equations (27), (28) and (29) it is concluded that:

a) (*K-2*)/*2 s*amples are symmetrically extended into the filter's delay line during the initialization phase.

b) (K-4)/2 samples are symmetrically extended into the filter's delay line during the finalization phase, for even and odd samples.

Hence, totally K-3 samples are symmetrically extended into the filter's delay line, for even and odd samples.

From the results presented, relative to the number of samples symmetrically extended into the filter's delay line, we can estimate the ratio by which the number of memory accesses is reduced. Let e be the number of signal samples symmetrically extended during a filtering over an input signal sequence of size N. Then the reduction of the memory accesses is:

$$r\% = \frac{e}{N+e} \cdot 100\%$$

where $N+e$ is the total number of memory accesses without the symmetric extension taking place in the filter's delay line.
In Tables 1 and 2 the values of the memory accesses reduction are illustrated for the Odd-Tap and Even-Tap filters, respectively. The values for each filter type are considered for four cases of input signal's size N and two cases of filter's size K.

TABLE I
REDUCTION OF THE MEMORY ACCESSES FOR THE ODD-TAP FILTERS

Filter type / Samples		N=16		N=32		N=64		N=128	
		K=5	K=9	K=5	K=9	K=5	K=9	K=5	K=9
OTAF	Even	16%	30%	8.5%	18%	4.8%	9.8%	2.3%	5.2%
	Odd	21%	33.3%	11.1%	20%	5.9%	11.1%	3%	5.9%
OTSF	Even	16%	30%	8.5%	18%	4.8%	9.8%	2.3%	5.2%
	Odd	11.1%	27.2%	5.9%	15.8%	3.0%	8.6%	1.5%	4.5%

TABLE II
REDUCTION OF THE MEMORY ACCESSES FOR THE EVEN-TAP FILTERS

Filter type / Samples		N=32		N=64		N=128		N=256	
		K=18	K=10	K=18	K=10	K=18	K=10	K=18	K=10
ETAF	Even	33.3%	20%	20%	11.1%	11.1%	5.9%	5.9%	3%
	Odd	31.9%	17.9%	19%	9.9%	10.5%	5.2%	5.5%	2.7%
ETSF	Even	31.9%	17.9%	19%	9.9%	10.5%	5.2%	5.5%	2.7%
	Odd	31.9%	17.9%	19%	9.9%	10.5%	5.2%	5.5%	2.7%

As concluded from the Tables 1 and 2 there is a reduction in memory accesses that depend on the length of the input signal sequence and reaches 33.3%. This reduction is due to the symmetric extension occurring internally in the filter's delay line, something that it is not the case in previously published works [7], [8], [9] and [10]. In [11] it is shown that a reduction in memory accesses results in an analogous reduction in memory power consumption. The same reduction is achieved for the case of 2D signals since the filters, in the separable 2D-DWT case, are applied first in the rows and then in the columns of the input image.
A significant reduction is also accomplished by applying the proposed filters for computing the tile based 2D-DWT [15]. In this way of computing, the 2D-DWT is applied in blocks of the input image. These blocks have typical sizes from 16×16 to 64×64 samples in multiples of 2. Thus, according to Tables 1 and 2 a reduction that ranges from 5% to 33.3% is achieved.

V. EXPERIMENTAL RESULTS

The eight filters designs were captured by using VHDL language (VHSIC Hardware Description Language). All of the system components have been described with structural architecture. Two different VLSI implementations are presented.

TABLE III
AREA (FPGA): CONFIGURABLE LOGIC BLOCKS (CLB), FUNCTION GENERATORS (FG), AREA (ASIC): SQMILS, F: OPERATING FREQUENCY (MHZ), P: ESTIMATED POWER (MW)

Filter Type	FPGA (XILINX v50ecs144)			ASIC (0.33 um)		
	Area		Freq	Area	Freq	P
	CLB	FG				
OTAF	183	365	71.4	995	46.8	3.6
OTSF	134	268	72	985	48.8	2.1
ETAF	345	490	57.8	2267	48.5	5.2
ETSF	359	497	57.5	3120	41.1	5.2
COTAF	121	241	57.3	951	48.2	6.2
COTSF	122	244	63.4	960	50.1	5.9
CETAF	271	541	46.9	2278	45.7	12.7
CETSF	336	671	61.7	2787	40.2	13.3

According to the first approach all the introduced filters were synthesized placed and routed using an FPGA device. Additionally experimental results were taken for the ASIC implementation. For the synthesis a 0.33 um CMOS standard cell library was used. In the Table 3 the synthesis results for both FPGA and ASIC implementations are illustrated in terms of covered area and operating frequency.
Finally the area-delay model for the ASIC technology is illustrated in the Fig. 18. Based on the experimental results it is cleared that the proposed filter architectures has almost the same covered area and performance compared with the conventional but with lower power consumption (Fig. 19).

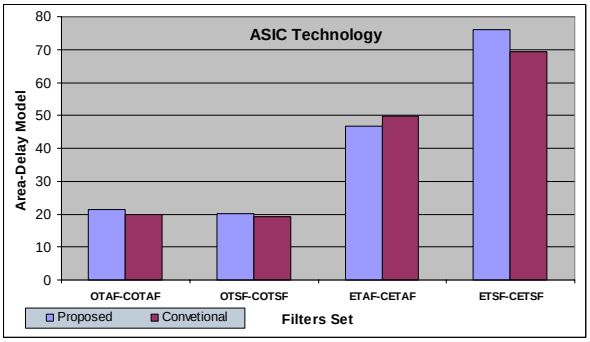

Fig. 18 Area-delay model for the DWT filters.

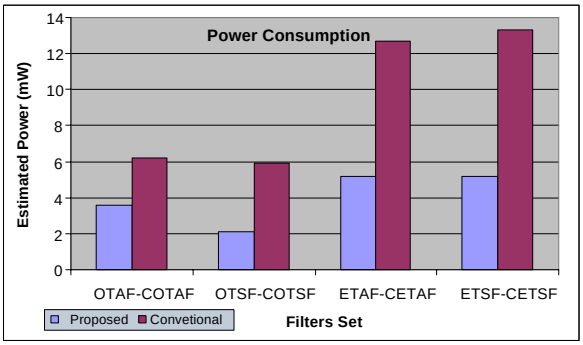

Fig. 19 Power consumption for the DWT filters.

VI. CONCLUSIONS

In the design techniques for implementing throughput

optimized 1-D filters for forward and invert wavelet transform was introduced. These filters can be embedded in a 2D-DWT encoder/decoder [15] for implementing the convolutional 2D-DWT according to the JPEG2000 standard. The illustrated filter architectures are based on reduced memory accesses, power and progressive computations. More specifically, the synthesis filters employ an adder less than the original Mallat's scheme and the proposed scheme in [5]. The signal extension prior to actual filtering is avoided and a great reduction in memory accesses is achieved. Finally, the need for buffering and reordering between the analysis and synthesis filters is eliminated. The proposed filter architecture has almost the same covered area and performance compared with the conventional but with low power estimation consumption.

REFERENCES

[1] I. Daubechies, "Ten Lectures on Wavelets," *CBMS-NSF Series in Applied Mathematics*, 61, SIAM, Philadelphia, 1992.
[2] Munteanu, J. Cornelis, G. V. der Auwera, P. Cristea, "Wavelet based lossless compression scheme with progressive transmission capability," *International Journal of Imaging Systems and Technology*, vol. 10, pp. 76-85, January 1999.
[3] Said and W. A. Pearlman, "A new fast and efficient image codec based on set partitioning in hierarchical trees," *IEEE Trans. Circuits and Syst. Video Technol.*, vol. 6, pp. 243-250, June 1996.
[4] J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Trans. On Signal Processing*, vol. 41, pp. 3445-3462, Dec. 1993.
[5] JPEG 2000 Image Coding System, ISO/IEC FCD15444-1, 2000.
[6] MPEG-4, ISO/IEC JTC1/SC29/WG11, FCD 14496, "Coding of Moving Pictures and Audio," May 1998.
[7] C. Chakrabarti and M. Vishwanath, "Efficient realizations of the discrete and continuous wavelet transforms: from single chip implementations to SIMD parallel computers," *IEEE Trans. Signal Processing*, vol. 43, no.3, pp. 759-771, March 1995.
[8] C. Chakrabarti and M. Vishwanath and R. M. Owens, "Architectures for wavelet transforms: A survey," *Journal of VLSI Signal Processing*, vol. 4, no. 2, pp 171-192, 1996.
[9] Vishwanath, R. M. Owens, M. J. Irwin "VLSI architectures for the discrete wavelet transform", *IEEE Trans. Circuits and Syst. II*, vol. 42, no. 5, May 1995.
[10] N. D. Zervas, G. P. Anagnostopoulos, V. Spiliotopoulos, Y. Andreopoulos and C.E. Goutis, "Evaluation of design alternatives for the 2-D-discrete wavelet transform", *IEEE Trans. Circuits and Syst. Video Technol.*, vol. 11, no. 2, pp. 1246-1262, December 2001.
[11] F. Catthoor, S. Wuytack, E. De Greff, F. Balasa, L.Nachtergale, A. Vandecappele, "Custom Memory Management Methodology - Exploration of Memory management Organization for Embedded Multimedia System Design", Kluwer Academic Publishers, 1998.
[12] S. Mallat, "A Wavelet Tour of Signal Processing", 2nd Edition.
[13] A. Skodras, C. Christopoulos, T. Ebrahimi, "The JPEG 2000 still image compression standard", in *IEEE Signal Processing Magazine*, vol. 18, no. 5, pp. 36-58, Sept. 2001.
[14] G. Lafruit, L. Nachtergale, J. Bormans, M. Engels, I. Bolsens, "Optimal memory organizations for scalable texture codecs in MPEG-4", *IEEE Trans. Circuits and Syst. Video Technol.*, vol. 9, no. 2, pp. 218-242, March 1999.
[15] G. Dimitroulakos, M. D. Galanis, A. Milidonis and C.E. Goutis, "A high- throughput memory efficient architecture for computing the tile-based 2D Discrete Wavelet Transform for the JPEG 2000 Standard", Integration the VLSI Journal, Elsevier Publishers, vol. 39, no. 1, pp. 1-11, 2005.
[16] C. Chakrabarti and C. Mumford, "Efficient realizations of analysis and synthesis filters based on the 2-D discrete wavelet transform," in *Proc. Int. Conf. On Acoustics, Speech and Signal processing*, pp. 3256-3259, May 1996.
[17] F. Fridman and E. S. Manolakos, "Distributed memory and control VLSI architectures for the 1-D discrete wavelet transform," in *IEEE VLSI Signal Processing VII*, pp. 388-397**,** 1994.
[18] Grzeszczak, M. K. Mandal, S. Panchanathan, and T. Yeap, "VLSI implementation of discrete wavelet transform," *IEEE Trans. VLSI Syst.*, vol. 4, pp. 421–433, Dec. 1996.
[19] G. Knowles, "VLSI architecture for the discrete wavelet transform", *Electronic Letters*, vol. 26, no.5, pp. 1184-1185, July 1990.
[20] R. Lang, E. Plesner, H. Schroder, and A. Spray, "An efficient systolic architecture for the one-dimensional wavelet transform," in *Proc. SPIE Conf. Wavelet Applicat.*, pp. 925–935, April 1994.
[21] A. S. Lewis and G. Knowles, "VLSI architecture for 2-D Daubechies wavelet transform without multipliers," *Electronic Letters*, vol. 27, no.5, pp. 171-173, Jan 1991.
[22] C.M. Brislawn, "Classification of nonexpansive symmetric extension transforms for multirate filter banks" *Tech. Rep. LA-UR-94-1747*, Los Alamos, Nat. Laboratory, May 1994.
[23] T. Denk and K. Parhi, "Calculation of minimum number of registers in 2-D discrete wavelet transforms using lapped block processing," in *Proc. Int. Symp. Circuits Syst.*, pp. 77–81, 1994.