3

VLSI Architectures of Lifting-Based Discrete Wavelet Transform

Sayed Ahmad Salehi and Rasoul Amirfattahi
Isfahan University of Technology, Department of Electrical and Computer Engineering, Digital Signal Processing Research Lab., Isfahan Iran

1. Introduction

The advantages of the wavelet transform over conventional transforms, such as the Fourier transform, are now well recognized. Because of its excellent locality in time-frequency domain, wavelet transform is remarkable and extensively used for signal analysis, compressing and denoising. Defining DWT by Mallat [1] provided possibility of its digitally hardware or software implementation. The discrete wavelet transform (DWT) performs a multiresolution signal analysis which has adjustable locality in both the space (time) and frequency domains [1]. Unlike the Fourier transform, the wavelet transform has many possible sets of basis functions. A trade-off can be made between the choice of basis functions and the complexity of the corresponding hardware implementations. Using finite impulse response (FIR) filters and then subsampling is the classical method for implementing the DWT. Due to the large amount of computations required, there have been many research efforts to develop new rapid algorithms [2]. In 1996, Sweldens presented a lifting scheme for a fast DWT, which can be easily implemented by hardware due to significantly reduced computations [3]. This method is entirely based on a spatial interpretation of the wavelet transform. Moreover, it provides the capability of producing new mother wavelets for the wavelet transform, based on space domain features. Due to recent advances in the technology, implementation of the DWT on field programmable gate array (FPGA) and digital signal processing (DSP) chips has been widely developed. As described in Sect. 3, in the lifting scheme the structural processing elements, including multipliers, are arranged serially; hence, the number of multipliers in each pipeline stage determines the clock speed of the structure. Based on [4], the main challenges in the hardware architectures for 1-D DWT are the processing speed and the number of multipliers, while for 2-D DWT it is the memory issue that dominates the hardware cost and the architectural complexity. The reason is the limitation of the on-chip memory and the power consumption [4,5].

2. DWT structures

The wavelet transform provides a time-frequency domain representation for the analysis of signals. Therefore, there are two main methods to produce and implement wavelet transforms. These methods are based on time domain or frequency domain features. The frequency based method is Filter Banks (FB) and the time based one is called Lifting Scheme (LS). We will describe them in following sections.

2.1 Filter banks structure

In the FB method, for one level of wavelet decomposition, the input signal is divided into two separate frequency parts by passing it simultaneously through a pair of low pass, $H(z)$, and high pass, $G(z)$, filters, as shown in Fig. 1. Then, subsampling the filter's output to produce the low pass and high pass outputs (s,d). Therefore, the FB method performs the DWT based on convolving filter taps and samples of the input signal. $H(z)$ and $G(z)$ can be written in this form:

$$H(z) = h_0 + h_1 z^{-1} + h_2 z^{-2} + ... + h_N z^{-N}$$
$$G(z) = g_0 + g_1 z^{-1} + g_2 z^{-2} + ... + g_M z^{-M}.$$

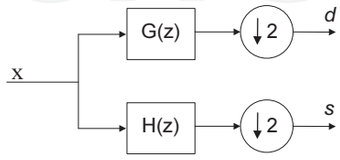


Fig. 1. *Filter Banks Block diagram*

As an example consider the CDF(2,2) wavelet. $H(z)$ and $G(z)$ for this transform are

$$H(z) = \frac{-1}{4\sqrt{2}} z^2 + \frac{1}{2\sqrt{2}} z + \frac{3}{2\sqrt{2}} + \frac{1}{2\sqrt{2}} z^{-1} + \frac{-1}{4\sqrt{2}} z^{-2}.$$

$$G(z) = \frac{-1}{2\sqrt{2}} z^2 + \frac{1}{\sqrt{2}} z + \frac{-1}{2\sqrt{2}}$$

The low pass filter has 5 taps and the high pass has 3 taps, so we call it 5/3 wavelet. Although FB structure is the prior one but it is only capable of providing wavelet transforms in the frequency domain and not in the time domain. Moreover, in general, the FB filter coefficients are not integer numbers; hence, they are not appropriate for hardware implementation. In addition, the number of arithmetic computations in the FB method is very large.

2.2 Lifting structure

The LS method is a new method for constructing and performing wavelets based on the time (space) domain [3].

As shown in Fig. 2, at first the LS structure splits the input signal samples into even and odd samples. Then P function is applied on even samples as a prediction function. The word prediction is used here because P function predicts odd samples using even samples.The difference between this prediction and the actual value of odd sample, creates the high frequency part of the signal which is called "detail" coefficients (d). Then applying the U function on detail signal and combining the result with even samples update them so that the output coefficients (s) have the desired properties. Usually the desired properties of s is the same as the properties of input signal (x) but with half size. So the s signal is an approximation for x and is called approximation coefficient.

Note that the details and approximation coefficients (d,s) in lifting scheme, respectively, are the same as high pass and low pass outputs in FB.

Based on the above description we have

$$d = x_{odd} - P(x_{even}),$$

for prediction block and

$$s = x_{even} + U(d)$$

for update block.
Equations for P and U functions are determined based on the implemented wavelet, also the number and arrangement of P and U blocks in the lifting structure are different for various types of wavelets.

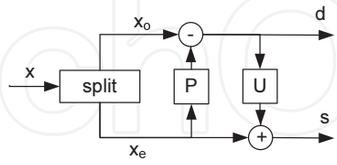


Fig. 2. Block diagram of a lifting stage

We can write matrix equations for P and U blocks respectively as following:

$$\begin{bmatrix} x_{even}(z) \\ d(z) \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 \\ t(z) & 1 \end{bmatrix}}_{\text{P}} \begin{bmatrix} x_{even}(z) \\ x_{odd}(z) \end{bmatrix}$$

$$\begin{bmatrix} s(z) \\ d(z) \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & s(z) \\ 0 & 1 \end{bmatrix}}_{\text{U}} \begin{bmatrix} x_{even}(z) \\ d(z) \end{bmatrix}.$$

Generally speaking, if we have more than one lifting step, the matrix equation is(3):

$$\begin{bmatrix} s(z) \\ d(z) \end{bmatrix} = \begin{bmatrix} k & 0 \\ 0 & 1/k \end{bmatrix} \prod_{i=1}^{m} \begin{bmatrix} 1 & s_i(z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ t_i(z) & 1 \end{bmatrix} \begin{bmatrix} x_{even}(z) \\ x_{odd}(z) \end{bmatrix} \tag{1}$$

In (1), k and $1/k$ are normalization factors. The last matrix is used only for normalization and may be omitted in many applications such as compression. The relation between FB coefficients and LS equations is (3):

$$E(z) = \begin{bmatrix} h_e(z) & h_o(z) \\ g_e(z) & g_o(z) \end{bmatrix} = \begin{bmatrix} k & 0 \\ 0 & 1/k \end{bmatrix} \prod_{i=1}^{m} \begin{bmatrix} 1 & s_i(z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ t_i(z) & 1 \end{bmatrix}$$

Matrix $E(z)$ is called a polyphase matrix, where according to the FB structure, h_e and h_o are even and odd taps of the low pass filter and g_e and g_o are even and odd taps of the high pass filter, respectively. $s_i(z)$ and $t_i(z)$ are related to filter coefficients in FB structure. In other words $s_i(z)$ and $t_i(z)$ can be obtained from FB by factorization algorithm presented in [5].
Example: Let consider the previous example, 5/3 wavelet, in LS. This wavelet consist of one lifting step (one P unit and one U unit together is a lifting step). For this wavelet the prediction of each odd sample in signal is the average of two adjacent even samples. Then P block calculates the difference between the real value of signal sample and its prediction:

$$d(n) = x(2n+1) - \frac{1}{2}[x(2n) + x(2n+2)].$$

U block updates even samples to have the same property as the original signal. It uses two most recently computed differences for update procedure:

$$s(n) = x(2n) + \frac{1}{4}(d(n-1) + d(n)).$$

So the matrix equation for 5/3 wavelet is

$$\begin{bmatrix} s(z) \\ d(z) \end{bmatrix} = \underbrace{\begin{bmatrix} \sqrt{2} & 0 \\ 0 & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 & \frac{1}{4}(1+z^{-1}) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \frac{-1}{2}(1+z) & 1 \end{bmatrix}}_{\mathrm{E(z)}} \begin{bmatrix} x_{even}(z) \\ x_{odd}(z) \end{bmatrix}$$

and the polyphase matrix is

$$E(z) = \begin{bmatrix} \frac{-1}{4\sqrt{2}}z^{-1} + \frac{3}{2\sqrt{2}} - \frac{1}{4\sqrt{2}}z & \frac{1}{2\sqrt{2}}z^{-1} + \frac{1}{2\sqrt{2}} \\ \frac{-1}{2\sqrt{2}} - \frac{1}{2\sqrt{2}}z & \frac{1}{\sqrt{2}} \end{bmatrix}.$$

We propose the following lemma for using in hardware implementation of LS as will be describe in section 3.1.

Lemma1: Factorization can be done so that $s_i(z)$ and $t_i(z)$ are first-order or lower-order polynomials.

Proof sketch: After a polyphase matrix representing a wavelet transform with finite filters is factored into lifting steps, each step becomes a Laurent polynomial. Since the difference between the degrees of the even and odd parts of a polynomial is never greater than 2, it is always possible to find the common divisor of the first-order or lower-order polynomials. Also, the lifting factorization process is non-unique and so there is freedom in the form of factorization. Hence, a classical wavelet filter can always be factored into first-order or lower-order Laurent polynomials (i.e., $s_i(z)$ or $t_i(z)$).□

Compared to the FB method, the LS method has many advantages [6,7]. The most important one is the number of arithmetic computations. In the LS method, the number of arithmetic operations, additions and multiplications, is nearly one-half of that of the FB, which is why the LS structure is more efficient. Even the amount of computations in some types of DWT can be reduced to a quarter of that needed for FB [8]. Furthermore the LS structure has the advantage of implementing the Integer Wavelet Transform (IWT) efficiently. IWT is a wavelet-like transform in which all of the decomposition coefficients are integer [9]. The IWT is appropriate for hardware implementation of the DWT [10]. The practical advantages of using the lifting-based IWT have been described in [11]. Moreover, by using the LS, it is easy to implement the DWT in a fully in-place method, which is memory efficient [2,12].

Regarding the above explanation, the LS structure is used to implement 5/3 and 9/7 wavelets, which are used, respectively, for lossless and lossy compression in the JPEG 2000 standard.

3. 1-DDWT

In this section, some types of lifting-based DWT processing elements and 1-D structures are explained.

3.1 Basic functional units for lifting scheme

As pointed out in Lema1, factorization can be done so that si(z) and ti(z) are first order or lower-order polynomials. Figure 3 shows three possible categories of the basic processing unit and the related polynomials in such a factorization. Different kinds of lifting-based DWT architectures can be constructed by combining the three basic lifting elements. Most of the applicable DWTs like 9/7 and 5/3 wavelets consist of processing units, as shown in Fig. 3(a), which is simplified as Fig. 4. This unit is called the processing element (PE). In Fig. 4, A, B and C are input samples which arrive successively. To implement the P unit, A and C receive

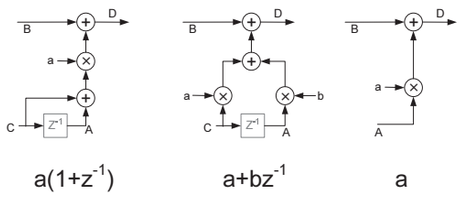


Fig. 3. Basic functional units for LS

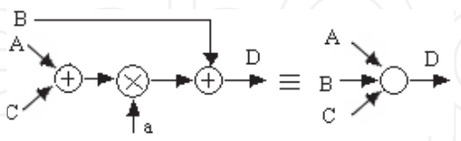


Fig. 4. Most convenient basic PE for LS

even samples while B receives odd samples. On the other hand, for the U unit, A and C are odd samples and B receives even samples. Now, the structure of Fig. 4 can be used to implement 5/3 and 9/7 wavelets. For instance, Fig. 5 and Fig. 6 shows the architecture of the 5/3 and 9/7 wavelets respectively, where each white circle represents a PE. In Fig. 6, the

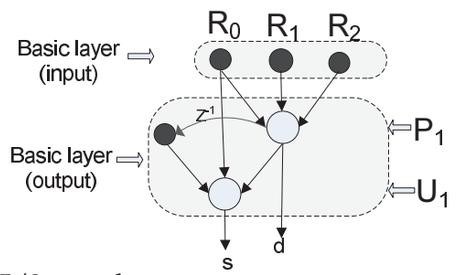


Fig. 5. Lifting Structure for 5/3 wavelet

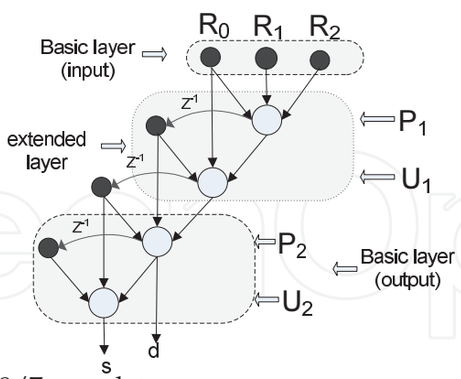


Fig. 6. Lifting Structure for 9/7 wavelet

input and output layers are essential (basic) layers and are fixed for each wavelet type, while by changing the number of extended layers, the type of wavelet can be changed accordingly. For example, omission of a single extended (added) layer in Fig. 6 will change the related architecture from 9/7 type to 5/3 type. The black circles in Fig. 6 represent needed stored data for computing outputs (s,d). R_0, R_1 and R_2, are registers that get their values from new input samples and are called data memory. The other three black circles which store the results of previous computations are known as temporary memory. The number of data memory

registers is constant and is equal to 3, while the number of temporary memory registers is $(2e + 1)$, where e is the number of extended layers [13]. This structure can be implemented by using combinatorial circuits so that, when the input samples are fed to the architecture, outputs are ready to be used after a delay time. Also, the implementation of the structure can be performed via a pipelined structure by adding some registers. The number of pipeline stages depends on the added registers. Increasing the pipeline stages results in increases in the clock frequency, system latency and number of required registers [4]. Note that 2-D DWT architectures are constructed from 1-D DWT units as row-wise and column-wise DWT units. The data of a complete row is saved for each memory in a column-wise unit. So, the sum of the data and temporary memories in the column-wise DWT unit determines the amount of needed internal memory [14,15,16]. The pipeline registers do not affect the required internal memory [17].

3.2 1-D DWT structures

By combining the functional units described in previous section we can construct 1 dimensional DWT. The architectures presented in Fig. 7 can be applied to implement the lifting-based 1-D DWT. The structure shown in Fig. 7(a) processes all input samples concurrently, in parallel form. In Fig. 7(b), input samples arrive in pairs at consecutive clock pulses and the results for each pair are ready after five cycles. However, due to the pipelined structure, the clock frequency of Fig. 7(b) is higher than that of Fig. 7(a). There is a trade-off between the clock speed and the number of pipeline stages.

4. Hardware architectures

4.1 Simple hardware implementation

Figure 7 shows an architecture proposed in [18] for the 9/7 wavelet. Indeed, it is the hardware Implementation of Fig. 6. Accordingly, the architecture presented in Fig. 9 can be used for the 5/3 wavelet.

4.2 Minimizing hardware architectures

In the structure of Fig. 8, there are two similar cascaded blocks which are different only in the multiplier's coefficients. According to [19], one of the similar blocks may be omitted as shown in Fig. 10. Note that in Figs. 7 and 8, the delay unit represented by z^{-1} is implemented by one register, while in Fig. 10 each delay unit contains two consecutive registers. Investigation on the architecture depicted in Fig. 10 shows that this hardware contains one P and one U unit. Note that, as mentioned in Sect. 2, both the P and U units can be implemented with the functional unit depicted in Fig. 4. Therefore, the structure shown in Fig. 10 is implemented by two similar sections which can be reduced to one section. The resulting architecture for the 9/7 wavelet is shown in Fig. 11. In this structure, U1(0) represents the current output of the U1 unit and P1(-1) represents the previous output of the P1 unit, and so on. The control signal, "S", which has four states, selects the inputs of the multiplexers sequentially. In the first state, two consecutive input samples arrive and the P1 function with α coefficient is performed on them. In the second state, the U1 function with β coefficient will be imposed on the result of the previous state (first state's output). Similarly, in the third and fourth states, computations for P2 and U2 units will be performed on the results of the previous states. Thus, P2 and U2 produce final outputs for the structure. The data flow for achieving a pair of wavelet coefficients using the proposed structure is shown in Table 1.

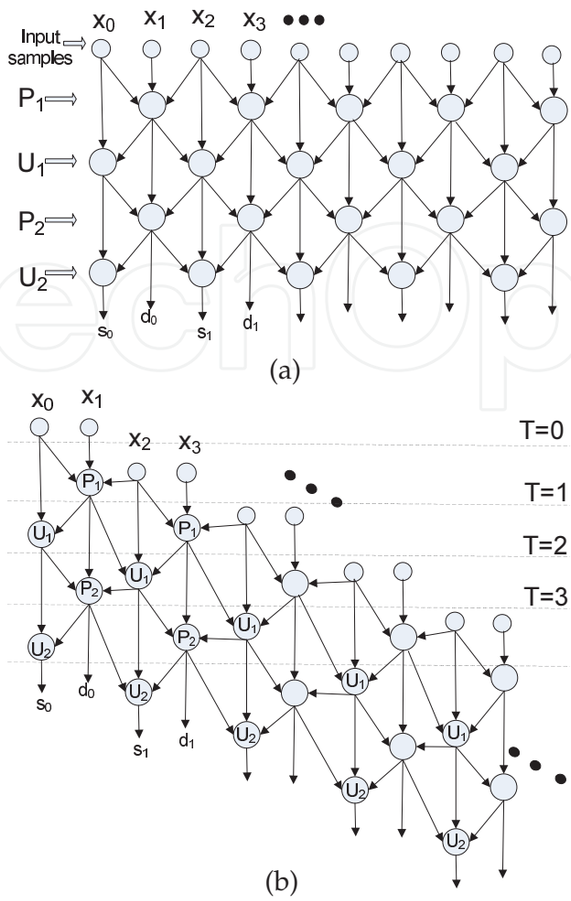


Fig. 7. 1-D DWT structures based on lifting a)parallel architecture b) sequential-pipelined architecture

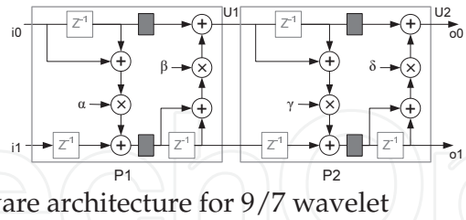


Fig. 8. Lifting-based hardware architecture for 9/7 wavelet

S	in1	in2	out	F(factor)
0	i1	i0	P_1	α
1	i0(-1)	P_1	U_1	β
2	$P_1(-1)$	U_2	P_2	γ
3	$U_1(-1)$	P_2	U_2	ζ

Table 1. Time sequence for structure of Fig. 11

The calculation of consecutive wavelet coefficients is periodic and continuous; therefore, the sequence of control signal "S" for data flow can be easily generated by a simple logic circuit. Figure 11 shows the hardware architecture for Fig. 11. The 5/3 wavelet implementation of the proposed architecture is depicted in Fig. 13. It is clear that only the number of coefficients

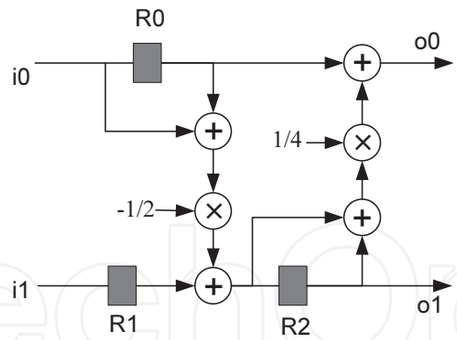


Fig. 9. Lifting-based hardware architecture for 5/3 wavelet

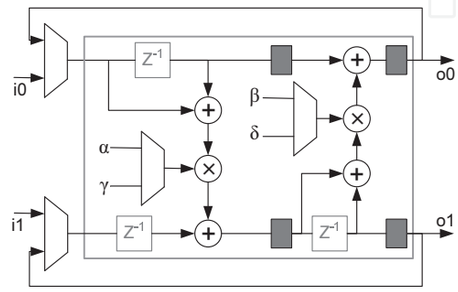


Fig. 10. Propose architecture in [19]

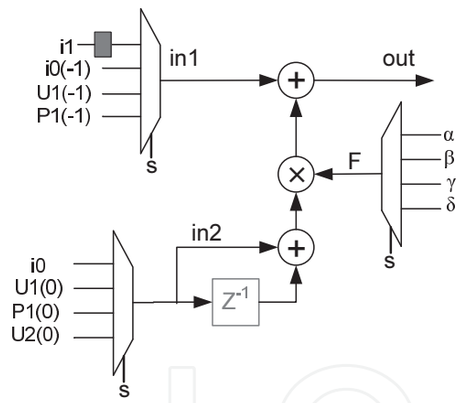


Fig. 11. Minimized structure

and delay block registers, that is, the z^{-1} blocks, have been modified from four to two. So, changing the wavelet type changes these two quantities, coefficients and registers, only. Both P and U units in LS can be implemented by means of the PE shown in Fig. 4. We explored this feature in the previous section and implemented a 1-D DWT structure containing only one PE. We call this method the "folded method". The folded structure is an alternative for the proposed method in [12] by which the lifting-based structures can be designed systematically. As shown in Fig. 14 for 9/7 wavelet, the method in [12] produces systolic architecture, but folded method produces folded architecture. In folded structure, the output of the PE unit is fed back through the delay registers to the PE's input. By incorporating different numbers of delay registers and coefficients with PE, the structure for different wavelets can be designed. For example the folded structure for 5/3 and 9/7 wavelets has two and four delay registers, respectively. Also the coefficients for 5/3 wavelet are $\frac{-1}{2}$ and $\frac{1}{4}$while for 9/7 they are $\alpha, \beta, \gamma, \delta$.

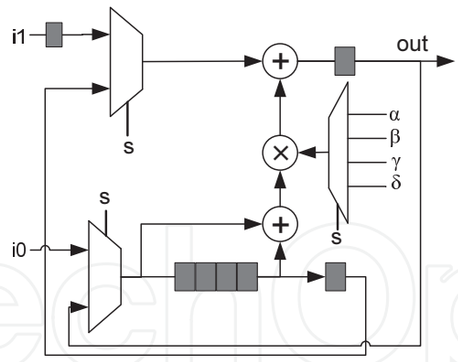


Fig. 12. Hardware implementation for Fig.10

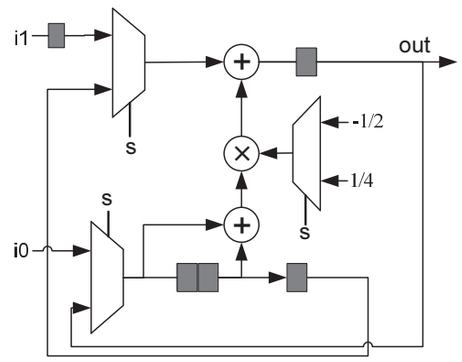
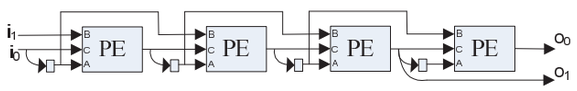
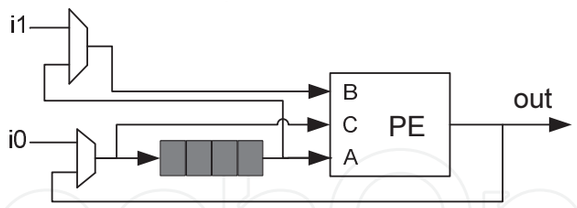


Fig. 13. Minimized architecture for 5/3 wavelet



(a) Systolic method



(b) Folded method

Fig. 14. 1-D Lifting-Based DWT for 9/7 wavelet by Systematic Design Method

In order to show the efficiency of our architecture, several architectures are chosen for comparison. Ignoring the pipeline registers, the results of comparison for the 9/7 wavelet are given in Table 2. It is obvious that compared to other architectures, the number of processing units is reduced in the folded architecture, thus requiring less area to implement the DWT. Having smaller 1-D DWT units is very effective in multidimensional architectures or in 2-D DWT, where it is needed to increase the number of 1-D DWT units to achieve a higher performance [20]. The cost is that, in the proposed architecture, the clock pulses required to compute outputs are more than those in the previous architectures. This requirement is due to the sequential states required to complete the computation of each output.

Architecture	Multiplier	Adder	Register
Lifting[18]	4	8	6
Proposed in [19]	2	4	6
Systolic[12]	4	8	4
Folded [30]	1	2	4

Table 2. Comparison of some different 1-D DWT architectures for 9/7 wavelet ($J \longrightarrow \infty$)

5. 2-D DWT structures

In this section, we review four convenient structures for 2-D DWT. It is assumed that the 2-D wavelets reviewed in the following structures are separable, so that the 2-D wavelet transform can be reduced to a 1-D wavelet transform performed on rows and columns, respectively.
In the direct method (step-by-step method), shown in Fig. 15, the input frame, stored in external memory, arrives at the 1-D DWT, row by row. The primary outputs are wavelet coefficients in the row direction and are stored in the external memory. After scanning all the rows of the frame, again the coefficients are transferred from the external memory to the 1-D DWT block, but this time in the column-wise direction. The secondary outputs of the 1-D DWT block are 2-D DWT coefficients of the input frame, which are stored in the external memory again. If computation of coefficients for one more decomposition level is needed, this procedure must be repeated for the LL part of the previous level, whose size is a quarter of the input frame size. This routine will be repeated for higher levels. The direct method hardware is simple, but its latency and the number of external memory accesses are large. The number of external memory accesses for computing a J-level 2-D DWT of an $N \times N$ input image can be calculated by the expression below, where half of the sum is related to the external memory reads and the other half is related to the external memory writes

$$4 \times (1 + \frac{1}{4} + \frac{1}{16} + \cdots + (\frac{1}{4})^{J-1}) \times N^2 \tag{2}$$

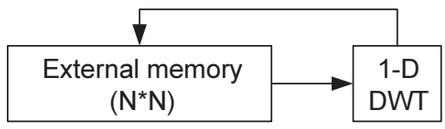


Fig. 15. Direct method

The line-based method can be implemented in two forms: single level and multilevel. In the line-based single level method, which is shown in Fig. 16, each level of DWT is performed by a 2-D DWT block. In this method, only internal memory is used to compute one level DWT for both the row and column directions, hence, there is no external memory access during the computation of one level 2-D DWT (except for reading rudimentary inputs and writing final results for that level). The required internal memory is the sum of the data memory and the temporal memory (black circles shown in Fig. 6) for each line. So the amount of needed internal memory is 6N for 9/7 wavelet and 4N for 5/3 wavelet [13,21]. But the results of the DWT in the row direction for even rows can be used for the computation of the DWT in the column direction without storing them. Hence, the required internal memory for 9/7 and 5/3 2-D wavelets is reduced to 5N and 3N, respectively. Recently, some new modifications have been made for the 2-D DWT block. For example, in [20] the number of data entrances has been increased by using more 1-D DWT units in the 2-D DWT block. Although this modification

increases the speed, it requires more internal memory and the size of the circuit is increased. In [22], by replacing registers with line buffers and controlling data flow in a structure like Fig. 13, with 5 more registers, a 2-D DWT block for 5/3 wavelet has been proposed.
Obviously, for a higher-level 2-D DWT, only LL coefficients of the previous level are used, so the total number of external memory access for a J-level 2-D DWT on an $N \times N$ image is

$$2 \times (1 + \frac{1}{4} + \frac{1}{16} + \cdots + (\frac{1}{4})^{J-1}) \times N^2 \tag{3}$$

The structure of Fig. 17 performs all levels of 2-D DWT, using only internal memory. So,

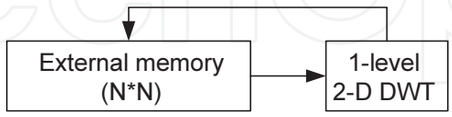


Fig. 16. Single level line-based method

the total number of external memory accesses for a J-level 2-D DWT is limited to $2N^2$, which corresponds to reading the input image for the first level and writing the final DWT results. The line-based multilevel structure, shown in Fig. 17, is much faster than the previous structures, but it needs a larger amount of hardware and so its hardware utilization (i.e., the average value of the area of working parts versus the whole area of the hardware) is low [23], but the 2-D recursive architecture proposed in [24] improves the hardware utilization for the J-level 2-D DWT. In Fig. 17, the required internal memory for the 9/7 wavelet is obtained from equation (4).

$$5N \times (1 + \frac{1}{2} + \frac{1}{4} + \cdots + (\frac{1}{2})^{J-1}) \tag{4}$$

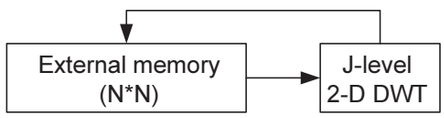


Fig. 17. Multilevel line-based method

There is a trade-off between the size of the internal memory and the number of external memory accesses in the 2-D DWT structures mentioned previously. Now, a block-based structure that parameterizes the aforementioned trade-off is introduced. The block-based structure is similar to the line-based method, but instead of considering the total length of a row for DWT in the row direction, only a part of it with length M pixels is considered (Fig. 18). It means that the first M columns of the main frame (the gray area in Fig. 18) are used as the input frame and 2-D DWT coefficients are computed for them. So the required internal memory, which is determined by the length of the rows, is decreased. As an example, for the 9/7 wavelet the internal memory size will be decreased from 5N to 5M (where M is a fraction of N). It is possible to consider a block of image by partitioning the image in both the row and column directions. In this method, the block (or window) slides across the image and both the row- and column-wise 1-D DWT will be performed on them [25]. The size of tile windows may be reduced to 2×2 pixels [26].

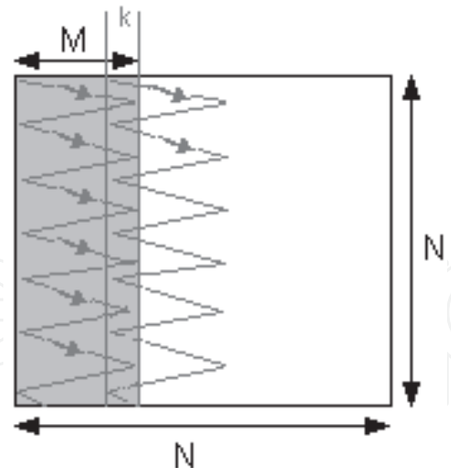


Fig. 18. Scan method in block-based structure

6. Scan methods for block-based structure

However, in the above-mentioned method, there is a problem in the boundary region between two M-pixel sections. To compute the DWT for the beginning pixel of the nextM-pixel section, values of K previous pixels are needed. These K pixels produce values of temporary memory (black circles shown in Fig. 19). K is equal to $n_t - 2$, where nt is the number of filter taps corresponding to the desired DWT. For the 9/7 wavelet, as shown in Fig. 19, K is equal to 7 (shaded circles). To solve the boundary problem, the overlapped scan method has been

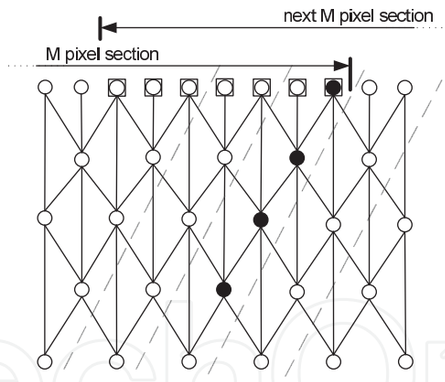


Fig. 19. Boundary region for 9/7 wavelet

proposed in [27]. A new M-pixel section begins from the last K pixels in the previous section. So two sections are overlapped in K pixels, and this causes the number of external memory reads to be $\frac{N^2M}{(M-K)}$ instead of N^2. The number of external memory writes is limited to writing the output results and is equal to N^2.

We can use an alternate scan method for the overlapped region at the boundary between two M-pixel sections. The relationships for the new scan method are different from the previous method. The temporary memory at the boundary of two M-pixel sections (black circles in Fig. 19) can be saved for the computation of the next section. These saved values will be used for the computation of the next M pixels. Hence, a new M-pixel section begins without any overlap with the previous section. The required memory to save temporary data is $L \times N$, where L is the value of the temporary memory in the related DWT core. For example, in Fig.

	Direct	Single level line-based	Block-Based	
Internal Memory Size	0	$5N$	$5M$	$5M$
External Memory Reads	$2N^2$	N^2	$\frac{N^2M}{M-K}$	$N^2(1+\frac{4}{M})-4N$
External Memory Write	$2N^2$	N^2	N^2	$N^2(1+\frac{4}{M})-4N$
Control complexity	Low	Medium	Medium	Medium

Table 3. Comparison of different 2-D DWT structures for one level 9/7 wavelet

19, L is 4 for the 9/7 wavelet without pipelining. The storage of temporary memory may be fulfilled in internal or external memory. If internal memory is used to save temporary data, the number of external memory accesses is equal to N^2 read and N^2 write operations. However, if external memory is used to save temporary memory, both of the external memory reads and writes are increased by an amount of $(\frac{N}{M}-1)N \times L$. Hence, the number of external memory reads as well as the number of external memory writes is equal to $N^2+(\frac{N}{M}-1)N \times L$. In this expression, the $(\frac{N}{M}-1)$ coefficient is the number of M-pixel sections. Due to hardware limitations (the limit size of internal memory on FPGA ICs), we select the second case for implementation. Comparisons of the aforementioned methods for one level 2-D DWT are given in Table 3. The table shows the values of the internal memory size and external memory accesses for the three algorithms. It is shown that in our proposed algorithm the internal memory size is between those of two other algorithms (the direct method and the line-based method). The same conclusion is true for the external memory size. The table also shows the order of complexity for the control circuits of these methods based on [28]. Similar comparisons for J-level ($J \longrightarrow \infty$) 2-D DWT are given in Table 4. Note that M has been considered to be fixed for all levels of 2-D DWT. It means that the width of the M-pixel section for the current level is the same as the one in the previous level. The J-level structure can be implemented either in the form of a single level (Fig. 15) or multilevel structure (Fig. 16), and the relevant values are listed in Table 4. We observe that the parameter M can be determined according to hardware limitations, such as internal memory. The conclusion from the two tables is that the block-based structure with the new scan method, in comparison with the direct method, needs more internal memory, but needs only about one-half of the external memory accesses. Due to the shorter access time for internal memory, the clock pulse frequency will increase, and based on the energy model in [5] the power consumption will decrease. In comparison with other methods, the new method remarkably decreases the needed internal memory at the cost of a soft increase in the number of external memory accesses.

7. Experimental results

The folded 1-D DWT architecture was described in VHDL code and simulated by Active-HDL6.3 software. Then the relevant VHDL code was synthesized by the Synplify7.5.1 software tool to be implemented on IC XC2V40 (from the VirtexII family of Xilinx FPGAs). The maximum estimated frequency for implementing Fig. 12 on this IC is 122.4 MHz, which is practical for real-time implementation of the 9/7 wavelet for large images. The maximum frequency to implement the 5/3 wavelet on the IC is 163.1 MHz. Also, the block-based architecture with the new scan method was modeled and simulated for the 5/3 wavelet with N = 1024 and 8-bit pixels. The code was synthesized by Synplify7.5.1 for implementation on VirtexII. After post place and route simulation, the clock pulse frequency achieved was 97 MHz. The structure receives one pixel as input per each clock pulse. So, according to the

	Direct	Line-Based		Block-Based			
				Single Level Line-Based		Multilevel Line-Based	
		Single Level	Multi-level	(27)	(30)	(27)	(30)
Internal Memory Size	0	$5N$	$10N$	$5M$	$5M$	$(5M)^J$	$(5M)^J$
External Memory Reads	$\frac{8}{3}N^2$	$\frac{4}{3}N^2$	N^2	$\frac{4}{3}(\frac{N^2M}{M-K})$	$\frac{4}{3}N^2(1+\frac{4}{M})-8N$	$N^2+\frac{4}{3}\frac{N^2K}{M-K}$	$N^2+\frac{16N^2}{M}-8N$
External Memory Write	$\frac{8}{3}N^2$	$\frac{4}{3}N^2$	N^2	$\frac{4}{3}N^2$	$\frac{4}{3}N^2(1+\frac{4}{M})-8N$	N^2	$N^2+\frac{16N^2}{M}-8N$

Table 4. Comparison of different 2-D DWT structures for J level 9/7 wavelet ($J \longrightarrow \infty$)

calculations below, the folded structure can be used to perform 3 levels of 2-D DWT for 70 frames (1024×1024 pixels) per second, and it is suitable for use in real-time hardware video codec.

$$t = \frac{1024 \times 1024 \times (1 + \frac{1}{4} + \frac{1}{16})}{97MHz} = 14.2ms,$$
$$n_f = \frac{1}{t} = \frac{1}{14.2ms} \simeq 70(frame/s).$$

8. Conclusion

Lifting Scheme and some different lifting based architectures for DWT presented in this chapter. Then we focused on the size (area) of the architecture. An architecture to minimize the number of multipliers and adders has been investigated for implementation of 1-D DWTs. All types of 1-D DWTs can be implemented by modifying only the number of registers and coefficients of the architecture. Thus, the folded architecture, which has fixed form units for all DWT types, presents a new folded method for systematic implementation of DWT. It is possible to design a software program to produce the folded architecture for different types of wavelets. What is needed is to define coefficients ($\alpha, \beta, ...$) required for each step of the desired wavelet in the lifting scheme. The folded method can be extended for large and complex structures such as multilevel discrete wavelet packet transforms [29] to reduce the area. Also, we have reviewed the 2-D DWT block-based structure and shown its power to trade off between the internal memory size and the number of external accesses by a controlling parameter.

9. References

[1] S. Mallat, A theory for multiresolution signal decomposition: the wavelet representation. IEEE Trans. Pattern Anal. Mach. Intell. 11, 674–693 (1989)

[2] T. Acharya, C. Chakrabarti, A survey on lifting-based discrete wavelet transform architectures. J. VLSI Signal Process. 42, 321–339 (2006)

[3] I. Daubechies, W. Sweldens, Factoring wavelet transform into lifting steps. J. Fourier Anal. Appl. 4, 247–269 (1998)

[4] C.-T. Huang, P.-C. Tseng, L.-G. Chen, Flipping structure: an efficient VLSI architecture for liftingbased discrete wavelet transform, IEEE Trans. Signal Process. 52 (2004), pp. 1080–1089
[5] N.D. Zervas et al., Evaluation of design alternatives for the 2-D-discrete wavelet transform. IEEE Trans. Circuits Syst. Video Technol. 11(12), 1246–1262 (2001)
[6] K. A. Kotteri, S. Barua, A. E. Bell, and J. E. Carletta, "A comparison of hardware implementations of the biorthogonal 9/7 DWT: convolution versus lifting," IEEE Trans. Circuits Syst. II, Expr. Br., vol. 52, no. 5, pp. 256-260, 2005.
[7] M. Maurizio, M. Guido, P. Gianluca, and Z. Maurizio, "Novel JPEG 2000 compliant DWT and IWT VLSI implementations," J. VLSI Signal Process., vol. 35, no. 2, pp. 137-153, Sep. 2003.
[8] J. Reichel, On the arithmetic and bandwidth complexity of the lifting scheme, in Proc. of International Conference on Image Processing (2001), pp. 198–201
[9] R. Calderbank, I. Daubechies, W. Sweldens, B.-L. Yeo, Wavelet transforms that map integers to integers. Appl. Comput. Harmon. Anal. 5(3), 332–369 (1998)
[10] A. Jensen, A. La Cour-Harbo, Ripples in Mathematics: The Discrete Wavelet Transform (Springer, Berlin, 2001)
[11] K.G. Oweiss et al., A scalable wavelet transform VLSI architecture for real-time signal processing in high-density intra-cortical implants. IEEE Trans. Circuits Syst. 54(6), 1266–1278 (2007)
[12] C.-T. Huang, P.-C. Tseng, L.-G. Chen, Efficient VLSI architectures of lifting-based discrete wavelet transform by systematic design method, in IEEE International Symposium on Circuits and Systems, vol. 5 (2002), pp. 565–568
[13] W.-H. Chang, Y.-S. Lee, W.S. Peng, C.-Y. Lee, A line-based, memory efficient and programmable architecture for 2D DWT using lifting scheme, in IEEE International Symposium on Circuits and Systems, vol. 4 (2001), pp. 330–333
[14] K. Andra, C. Chakrabarti, T. Acharya, A VLSI architecture for lifting-based forward and inverse wavelet transform. IEEE Trans. Signal Process. 50(4), 966–977 (2002)
[15] O. Fatemi, S. Bolouki, Pipeline memory-efficient and programmable architecture for 2D discrete wavelet transform using lifting scheme, in Proceedings of IEE Circuits, Devices and Systems, December 2005, pp. 703–708
[16] Lan, N. Zheng, Y. Liu, Low power and high-speed VLSI architecture for lifting-based forward and inverse wavelet transform. IEEE Trans. Consumer Electron. 51(2), 379–385 (2005)
[17] C.Y. Xiong, J.W. Tian, J. Liu, A note on Şflipping structure: an efficient VLSI architecture for liftingbased discrete wavelet transformŤ. IEEE Trans. Signal Process. 54(4), 1910–1916 (2006)
[18] J.M. Jou, Y.H. Shiau, C.C. Lio, Efficient VLSI architectures for the biorthogonal wavelet transform by filter bank and lifting scheme, in Proceedings of IEEE ISCAS 2001, pp. 529–533
[19] C.J. Lian, K.F. Chen, H.H. Chen, L.G. Chen, Lifting based discrete wavelet transform architecture for JPEG2000, in IEEE International Symposium on Circuits and Systems (ISCAS 2001) Sydney, May 2001
[20] C.Y. Xiong, J.W. Tian, J. Liu, Efficient architectures for two-dimensional discrete wavelet transform using lifting scheme. IEEE Trans. Image Process. 16(3), 607–614 (2007)

[21] P.-C. Tseng, C.-T. Huang, L.-G. Chen, Generic RAM-based architecture for two-dimensional discrete wavelet transform with line-based method, in Asia-Pacific Conference on Circuits and Systems (2002), pp. 363–366
[22] H. Varshney, M. Hasan, S. Jain, Energy efficient novel architectures for the lifting-based discrete wavelet transform. IET Image Process. 1(3), 305–310 (2007)
[23] C.-T. Huang, P.-C. Tseng, L.-G. Chen, Hardware implementation of shape-adaptive discrete wavelet transform with the JPEG2000 defaulted 9/7 filter bank, in IEEE International Conference on Image Processing, vol. 3 (2003), pp. 571–574
[24] L.Hongyu, M.K. Mandal, B.F. Cockburn, Efficient architectures for 1-D and 2-D lifting-based wavelet transforms. IEEE Trans. Signal Process. 52(5), 1315–1326 (2004)
[25] C.H. Yang et al., A block-based architecture for lifting scheme discrete wavelet transform. IEICE Trans. Fundam. 90(5), 1062–1071 (2007)
[26] J.W. Kim et al., Tiled interleaving for multi-level 2-D discrete wavelet transform, in IEEE Int. Symp. Circuits Syst., May 2007, pp. 3984–3987
[27] C.-T. Huang, P.-C. Tseng, L.-G. Chen, Memory analysis and architecture for two-dimensional discrete wavelet transform, in IEEE International Symposium on Circuits and Systems (ISCAS 2004)
[28] C.-T. Huang, P.-C. Tseng, L.-G. Chen, Analysis and VLSI architecture for 1-D and 2-D discrete wavelet transform. IEEE Trans. Signal Process. 53(4), 1575–1586 (2005)
[29] C. Wang, W.S. Gan, Efficient VLSI architecture for lifting-based discrete wavelet packet transform. IEEE Trans. Circuits Syst. 54(5), 422–426 (2007)
[30] S.A. Salehi, S. Sadri, "Investigation of Lifting-Based Hardware Architectures for Discrete Wavelet Transform," Journal of Circuits Systems and Signal Processing, vol. 28, N.1, pp1-16, 2009.

Discrete Wavelet Transforms - Algorithms and Applications

Edited by Prof. Hannu Olkkonen

ISBN 978-953-307-482-5
Hard cover, 296 pages
Publisher InTech
Published online 29, August, 2011
Published in print edition August, 2011

The discrete wavelet transform (DWT) algorithms have a firm position in processing of signals in several areas of research and industry. As DWT provides both octave-scale frequency and spatial timing of the analyzed signal, it is constantly used to solve and treat more and more advanced problems. The present book: Discrete Wavelet Transforms: Algorithms and Applications reviews the recent progress in discrete wavelet transform algorithms and applications. The book covers a wide range of methods (e.g. lifting, shift invariance, multi-scale analysis) for constructing DWTs. The book chapters are organized into four major parts. Part I describes the progress in hardware implementations of the DWT algorithms. Applications include multitone modulation for ADSL and equalization techniques, a scalable architecture for FPGA-implementation, lifting based algorithm for VLSI implementation, comparison between DWT and FFT based OFDM and modified SPIHT codec. Part II addresses image processing algorithms such as multiresolution approach for edge detection, low bit rate image compression, low complexity implementation of CQF wavelets and compression of multi-component images. Part III focuses watermaking DWT algorithms. Finally, Part IV describes shift invariant DWTs, DC lossless property, DWT based analysis and estimation of colored noise and an application of the wavelet Galerkin method. The chapters of the present book consist of both tutorial and highly advanced material. Therefore, the book is intended to be a reference text for graduate students and researchers to obtain state-of-the-art knowledge on specific applications.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820

Fax: +385 (51) 686 166
www.intechopen.com

Fax: +86-21-62489821

IntechOpen