Journal of VLSI Signal Processing 40, 343–353, 2005
© 2005 Springer Science + Business Media, Inc. Manufactured in The Netherlands.

VLSI Architecture for Forward Discrete Wavelet Transform Based on B-spline Factorization

CHAO-TSUNG HUANG, PO-CHIH TSENG AND LIANG-GEE CHEN
DSP/IC Design Lab, Graduate Institute of Electronics Engineering and Department of Electrical Engineering, National Taiwan University, 1, Sec. 4, Roosevelt Road, Taipei 106, Taiwan

Received September 18, 2003; Revised October 16, 2003; Accepted November 13, 2003

Abstract. Based on B-spline factorization, a new category of architectures for Discrete Wavelet Transform (DWT) is proposed in this paper. The B-spline factorization mainly consists of the B-spline part and the distributed part. The former is proposed to be constructed by use of the direct implementation or Pascal implementation. And the latter is the part introducing multipliers and can be implemented with the Type-I or Type-II polyphase decomposition. Since the degree of the distributed part is usually designed as small as possible, the proposed architectures could use fewer multipliers than previous arts, but more adders would be required. However, many adders can be implemented with smaller area and lower speed because only few adders are on the critical path. Three case studies, including the JPEG2000 default (9, 7) filter, the (6, 10) filter, and the (10, 18) filter, are given to demonstrate the efficiency of the proposed architectures.

Keywords: discrete wavelet transform, VLSI architecture, B-spline factorization

1. Introduction

DWT has been developed as an efficient and powerful tool for signal analysis, image compression, and even scalable video coding recently [1]. Because a huge amount of computation would be required, many VLSI architectures have been proposed, which are mainly based on convolution scheme [2–4] and lifting scheme [5–7]. The convolution-based architecture is to implement two-channel filter banks directly, and many VLSI DSP techniques, such as polyphase decomposition [8], pipelining, and retiming [9], have been adopted to enhance the performance. On the other hand, the lifting scheme is used to express the two-channel filter banks in a new way [10]. In [11], a systematic method is proposed to factorize the polyphase matrix into many lifting steps based on the perfect reconstruction property. The lifting scheme usually requires fewer multipliers and adders than the convolution scheme.

However, the intrinsic B-spline property of DWT was not used to construct VLSI architectures in literature. According to [12], any DWT filters can be factorized into the B-spline part and the distributed part. The B-spline part contributes to all important wavelet properties. And the distributed part is used to design DWT FIR filters. Since only the distributed part requires multipliers, the B-spline factorization could use fewer multipliers than the lifting scheme but induce more adders.

In this paper, we propose to implement DWT based on the B-spline factorization. The B-spline part is proposed to be constructed with the direct implementation or Pascal implementation. The latter could reduce the adders, but could be too complex when the filter tap is too long. The distributed part could be implemented with the Type-I or Type-II polyphase decomposition, and conventional filter implementation methods all can be applied. Three case studies are given to examine the efficiency. However, the principal objective of this paper is to motivate a new category of DWT architectures.

The organization of this paper is as follows. Section 2 reviews previous arts of DWT architectures. The

B-spline factorization theory is described in Section 3, and the proposed architectures are presented in Section 4. The case studies of the JPEG2000 default (9, 7) filter, the (6, 10) filter [5], and the (10, 18) filter [13], are given in Section 5. Finally, a summary is given to conclude this paper in Section 6.

2. Previous DWT Architectures

This section introduces previous DWT architectures and classifies DWT architectures into three categories.

2.1. Convolution-Based

The multiresolution DWT analysis can be viewed as a cascade of several two-channel filter banks [14], and the analysis filter bank is shown in Fig. 1, where $H(z)$ and $G(z)$ are the lowpass and highpass filters, respectively. The convolution-based architectures are to implement DWT with the direct structures of two-channel filter banks. Many VLSI DSP design techniques, such as folding, unfolding, and pipelining [9], can be adopted to implement the pair of lowpass and highpass filters. Especially, the convolution-based architecture can be constructed by use of polyphase decomposition [8] as shown in Fig. 2, where $H(z) = H_e(z^2) + z^{-1}H_o(z^2)$ and $G(z) = G_e(z^2) + z^{-1}G_o(z^2)$ if Type-I decomposition is used, and $H(z) = H_e(z^2) + zH_o(z^2)$ and $G(z) = G_e(z^2) + zG_o(z^2)$ if Type-II decomposition

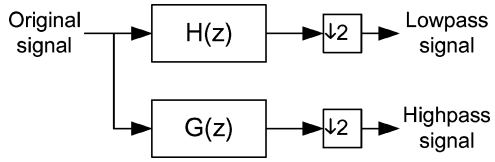


Figure 1. Two-channel analysis filter bank.

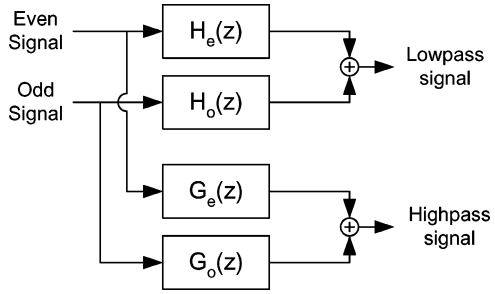


Figure 2. Polyphase decomposition of the analysis filter bank.

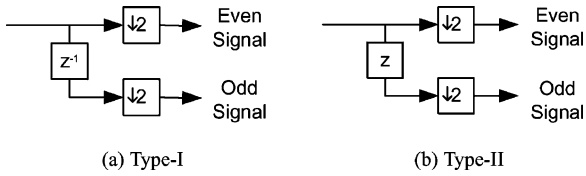


Figure 3. Two polyphase decomposition types.

is used. The Type-I and Type-II decompositions can be illustrated as Fig. 3. Then the four filters in Fig. 2 can be implemented by serial or parallel filters. In this convolution-based scheme, the lowpass and highpass filters are considered independently.

2.2. Lifting-Based

On the other hand, lifting scheme [10] has been widely used to reduce the required multiplications and additions by exploring the relation of lowpass and highpass filters. According to [11], any DWT filter bank of perfect reconstruction can be decomposed into a finite sequence of lifting steps. This decomposition corresponds to a factorization for the polyphase matrix of the target wavelet filter into a sequence of alternating upper and lower triangular matrices and a constant diagonal matrix, which can be expressed as follows:

$$P(z) = \begin{bmatrix} H_e(z) & G_e(z) \\ H_o(z) & G_o(z) \end{bmatrix} = \prod_{i=1}^{m} \begin{bmatrix} 1 & s_i(z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ t_i(z) & 1 \end{bmatrix} \begin{bmatrix} K & 0 \\ 0 & 1/K \end{bmatrix} \quad (1)$$

where $P(z)$ is the polyphase matrix.

Most of the proposed lifting-based architectures in literature are implemented with the above lifting factorization directly [5, 6]. Although the lifting scheme has many advantages, such as fewer arithmetic operations and in-place implementation, the potentially long critical path is a drawback for hardware implementation. In [7], this timing crisis is discussed in detail and addressed by use of the flipping structure, instead of pipelining.

2.3. Classification

As mentioned above, the general two-channel filter banks can be implemented with the convolution scheme. If the two-channel filter bank possesses the

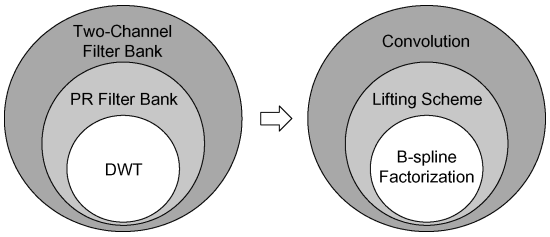


Figure 4. Categories of DWT architectures.

perfect reconstruction (PR) property, it could be implemented with fewer arithmetic operations by use of lifting-based architectures. DWT can be implemented with the above two schemes because it can be viewed as a two-channel filter bank of perfect reconstruction property.

However, the B-spline factorization property of DWT has not been used to construct efficient architectures in literature, which is an important property for DWT and will be described in the next section. Thus, DWT architectures can be categorized as shown in Fig. 4, where DWT is only a subset of convolution- and lifting-based architectures.

3. B-Spline Factorization

According to [12], the lowpass filter, $H(z) = \sum_{i=0}^{P_H-1} h_i z^{-i}$, and the highpass filter, $G(z) = \sum_{i=0}^{P_G-1} g_i z^{-i}$, of any DWT can be factorized as

$$\begin{aligned} H(z) &= \underline{(1+z^{-1})^{\gamma_H}} \cdot \underline{Q(z)} \cdot \underline{h_0} \\ G(z) &= \underline{(1-z^{-1})^{\gamma_G}} \cdot \underline{R(z)} \cdot \underline{g_0} \end{aligned} \tag{2}$$

where the first, second, and third terms of the right-hand side can be called the B-spline part, distributed part, and normalization part, respectively. Based on the B-spline factorization, the output of highpass filter can be viewed as the γ_G-th order difference of the smoothed input signals. There are two differences between the expression (2) and the expression of [12]. The first one is that we treat $1 \pm z^{-1}$ as the B-spline part, instead of $\frac{1+z^{-1}}{2}$. And the second one is the normalization part which is extracted in this paper only for implementation issues.

The B-spline part is responsible for all important properties of DWT, such as order of approximation, reproduction of polynomials, vanishing moments, and multiscale differentiation property. And the distributed part is used to derive efficient FIR DWT filters [12]. Thus, the order of the distributed part is usually designed as small as possible when the order of the B-spline part is given. The normalization part can be implemented independently from the other two parts and further together with the following quantization if image compression is needed. It is very similar to the normalization step in the lifting scheme.

4. Proposed B-Spline Factorized Architecture

We propose to implement DWT by using the B-spline factorization as the Eq. (2). For 100% hardware utilization, the polyphase decomposition is adopted first. After the Type-I or Type-II polyphase decomposition, the general B-spline factorized architecture can be expressed as Fig. 5, where the distributed part, $Q(z)$ and $R(z)$, are decomposed first, and the left is the B-spline part. The distributed part is the only part with multipliers and the four filters can be implemented by serial or parallel filters. Since the normalization part, h_0 and g_0, can be implemented independently from the other two parts, it will be excluded in the following discussion. Below we will introduce two implementation methods for the B-spline part.

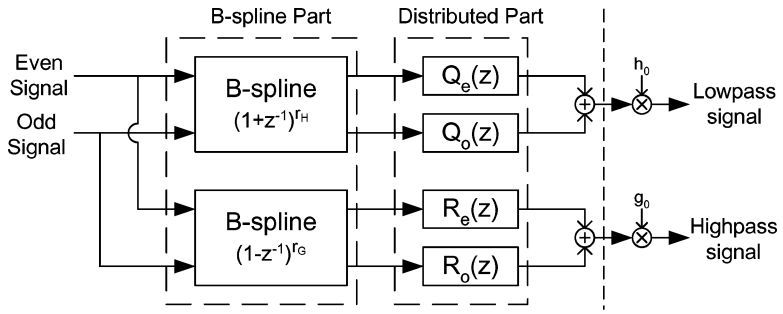


Figure 5. General B-spline-based architecture.

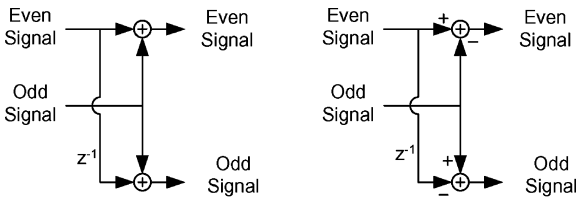


Figure 6. Direct implementation for the B-spline part.

4.1. Direct Implementation of the B-Spline Part

The direct implementation of the B-spline part is a straightforward one. The concept is to implement $(1+z^{-1})$ and $(1-z^{-1})$ first, and then the B-spline parts can be constructed by serially connecting $(1 + z^{-1})$ and $(1 - z^{-1})$. But two-input-two-output structures of $(1 + z^{-1})$ and $(1 - z^{-1})$ cannot be derived from polyphase decomposition. We propose to implement them by considering the physical connection of signals as shown in Fig. 6, where we assume the Type-I decomposition is used so the even signals are prior to odd signals. Thus, the direct implementation requires $2\gamma_H + 2\gamma_G$ adders for a pair of lowpass and highpass outputs. When connecting the B-spline part to the distributed part, the priority of signals needs to be handled carefully.

Another problem that should be solved is the internal signal wordlength. Since the DC gain of $(1 + z^{-1})$ is 2, the signal magnitude is possible to be double after every $(1+z^{-1})$ stage, and so is after every $(1-z^{-1})$ stage. However, implementing $(1 \pm z^{-1})/2$ instead will lose too much precision. The precision and wordlength issues should be handled carefully as the precision criteria is given. In this paper, a simple method is used to solve it. We scale down the signal by 2 after every two $(1 \pm z^{-1})$ stages for precision preservation and preventing from signal overflow.

4.2. Pascal Implementation of the B-Spline Part

Instead of the direct implementation, we also propose the Pascal implementation that can exploit the similarity of the two B-spline parts to reduce adders. The Pascal implementation expresses the $(1 + z^{-1})^{\gamma_H}$ and $(1 - z^{-1})^{\gamma_G}$ as the Pascal expansion and saves the repeated computation. For example, $1 + 6z^{-2} + z^{-4}$ and $4z^{-1} + 4z^{-3}$ can be computed first for the implementation of $(1 + z^{-1})^4 = 1 + 4z^{-1} + 6z^{-2} + 4z^{-3} + z^{-4}$ and $(1 - z^{-1})^4 = 1 - 4z^{-1} + 6z^{-2} - 4z^{-3} + z^{-4}$. Then the sum of them is $(1 + z^{-1})^4$, and the difference is $(1 - z^{-1})^4$. Furthermore, the integer multiplications of the B-spline part can be implemented with shifters and adders, instead of multipliers. In this example, the Pascal implementation only requires 12 adders, but the direct implementation will need 16 adders. However, the Pascal implementation of long-tap filters will be too complex to be derived, and the complexity reduction is not guaranteed. The precision and wordlength issues are also more complex than those of the direct implementation. In this paper, we preserve as more precision as possible when the internal wordlength is given.

4.3. Performance Discussion

The main advantage of the B-spline factorized architectures is that possibly fewer multipliers are required than the convolution and lifting scheme. This is because the degrees of $Q(z)$ and $R(z)$ (γ_Q and γ_R) are designed as small as possible for given γ_H and γ_G that dominates all wavelet properties.

The below is a general performance discussion. The convolution scheme requires about $\gamma_H + \gamma_G + \gamma_Q + \gamma_R$ multipliers, while the lifting scheme could possibly save a half number of multipliers [11]. But the B-spline factorized architecture only requires $\gamma_Q + \gamma_R$ multipliers which are fewer than $\frac{\gamma_H+\gamma_G+\gamma_Q+\gamma_R}{2}$ if $\gamma_Q + \gamma_R < \gamma_H + \gamma_G$. Daubechies wavelets are optimal in the sense that they have a minimum size support of a given number of vanishing moments [15]. Thus, we can derive the expression as follows:

$$\begin{aligned}&(\gamma_H + \gamma_Q + 1) + (\gamma_G + \gamma_R + 1) \geq 2(\gamma_H + \gamma_G)\\ &\quad \Rightarrow \gamma_Q + \gamma_R \geq \gamma_H + \gamma_G - 2 \end{aligned} \tag{3}$$

The Eq. (3) means that the sum of vanishing moments $(\gamma_H+\gamma_G)$ is always less than or equal to a half of the sum of the lowpass and highpass filter lengths. Thus, the B-spline factorized architectures can always guarantee the complexity reduction of multipliers by 2 relative to the convolution-based ones if Daubechies wavelets are used. But the lifting-based architectures cannot guarantee the performance.

Now we consider the common used linear filters. For the linear DWT filters, the convolution-based architectures can reduce the multipliers by 2 by adopting the linear properties. Since the B-spline part is always

linear, the distributed part is also linear and can reduce the multipliers by 2 as well. However, the lifting-based architectures cannot always adopt the linear properties. Especially for the even length DWT linear filters, the lifting steps are hard to be factorized as linear so that the required multipliers may be even more than convolution-based architectures.

The main disadvantage of the B-spline factorized architectures is that more adders may be required. But the complexity of adders is much less than that of multipliers. And most adders are not on the critical path, so they can be implemented in low speed and small area. In the result, the proposed architectures can provide more reduction of hardware resource than others.

5. Case Studies

In this section, three Daubechies biorthogonal filters are studied and implemented by use of proposed B-spline factorized architectures, including the JPEG2000 default (9, 7) filter, the (6, 10) filter [5], and the (10, 18) filter [13].

5.1. JPEG2000 Default (9, 7) Filter

The B-spline factorization of the (9, 7) filter can be expressed as:

$$H(z) = (1+z^{-1})^4(1+t_1z^{-1}+t_2z^{-2}+t_1z^{-3}+z^{-4})h_0$$
$$G(z) = (1-z^{-1})^4(1+t_3z^{-1}+z^{-2})g_0 \tag{4}$$

where $t_1 = -4.630464$, $t_2 = 9.597484$, and $t_3 = 3.369536$. Thus the B-spline factorized architecture of the (9, 7) filter will only need three multipliers, excluding the normalization part h_0 and g_0. Here, we use the Pascal implementation for the B-spline part, and the Pascal expression of the (9, 7) filter is shown in Fig. 7. The proposed B-spline factorized architecture requires 18 adders, of which 12 adders for the B-spline part and 6 adders for the distributed part. The proposed

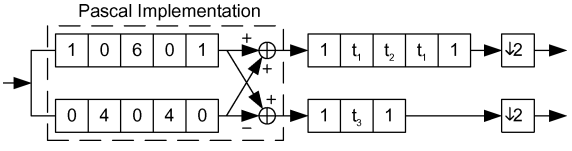


Figure 7. Pascal expression of the (9, 7) filter.

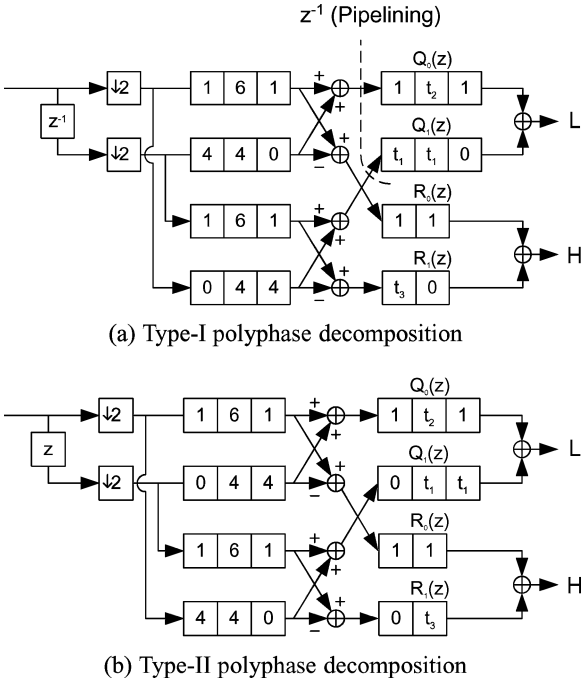


Figure 8. Proposed B-spline factorized architectures for (9, 7) filter.

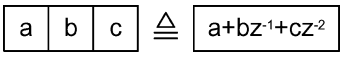


Figure 9. Notation for filters.

architectures are shown in Fig. 8, where Fig. 8(a) and (b) represent Type-I and Type-II polyphase decompositions, respectively. And the notation that we use for FIR filters can be described in Fig. 9.

The original Type-I architecture requires eight registers, and the critical path is $T_m + 5T_a$, where T_m is the time taken for a multiplication operation, and T_a is the time needed for an addition operation. On the other hand, if pipelining is performed through the upside dot line, the critical path can be shortened to $T_m + 2T_a$ with totally 10 registers. However, the critical path of the Type-II architecture is $T_m + 2T_a$ with only 10 registers.

5.1.1. Comparison. By extracting the normalization part h_0 and g_0 and utilizing the symmetric property, the convolution-based architecture of the (9, 7) filter can be implemented by use of 7 multipliers, 14 adders, and 7 registers. And the critical path is $T_m + 3T_a$ if adder tree is used to connect adders.

The lifting scheme of the (9, 7) filter can be factorized as:

$$P(z) = \begin{bmatrix} 1 & a(1+z^{-1}) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ b(1+z) & 1 \end{bmatrix} \times \begin{bmatrix} 1 & c(1+z^{-1}) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ d(1+z) & 1 \end{bmatrix} \begin{bmatrix} K & 0 \\ 0 & 1/K \end{bmatrix} \quad (5)$$

where $P(z)$ is the polyphase matrix, and the coefficients are given as $a = -1.586134342$, $b = -0.052980118$, $c = 0.882911076$, $d = 0.443506852$, and $K = 1.149604398$. The corresponding signal flow graph is shown in Fig. 10. Thus, the lifting-based architecture would require 4 multipliers and 8 adders if the normalization steps K and $1/K$ are excluded. The critical path $4T_m + 8T_a$ is quite long with only 4 registers and can be reduced to $T_m + 2T_a$ by pipelining through the dot lines with totally 10 registers. On the other hand, the flipping structure of the (9, 7) filter is proposed to flip Fig. 10 to reduce the critical path [7] as shown in Fig. 11, where the critical path is $T_m + 5T_a$ without any more hardware overhead than Fig. 10. The critical path can be further reduced to $T_m + 1T_a$ with three additional pipelining registers.

The proposed B-spline factorized architectures as well as the aforementioned convolution-based and lifting-based ones have been verified by use of Verilog-XL and synthesized into gate-level netlists by Synopsys Design Compiler with standard cells from Artisan 0.25-μm cell library. The comparison and synthesis results are shown in Table 1, where the internal bit-widths are all 16-bit, the multipliers are all 16-by-16 multiplications, and the adders are also 16-bit for comparison. The gate counts are given with combinational and non-combinational gate counts separately. The former contributes to the multipliers and adders while the latter is responsible to the registers. For circuit synthesis, the timing constraints are set as tight as possible.

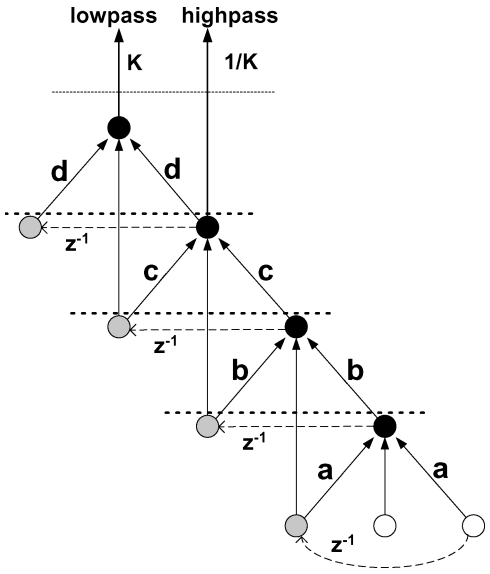


Figure 10. Lifting scheme for the (9, 7) filter.

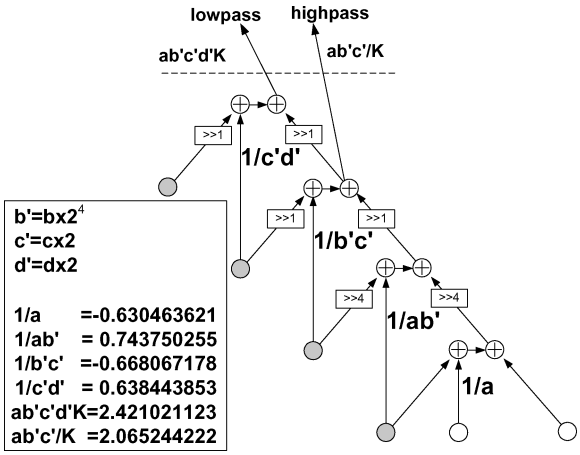


Figure 11. Flipping structure for the (9, 7) filter.

According to Table 1, the proposed architectures could require fewer gate counts under the same timing constraints. Furthermore, the saving of gate counts will be more significant if the multipliers are required to have higher precision.

5.2. *The (6, 10) Filter*

The B-spline factorization of the (6, 10) filter [5] can be expressed as:

$$H(z) = (1+z^{-1})^3(1+s_3z^{-1}+z^{-2})h_0$$

$$G(z) = (1-z^{-1})^3(1-z^{-1})^2(1+s_1z^{-1} + s_2z^{-2}+s_1z^{-3}+z^{-4})g_0 \quad (6)$$

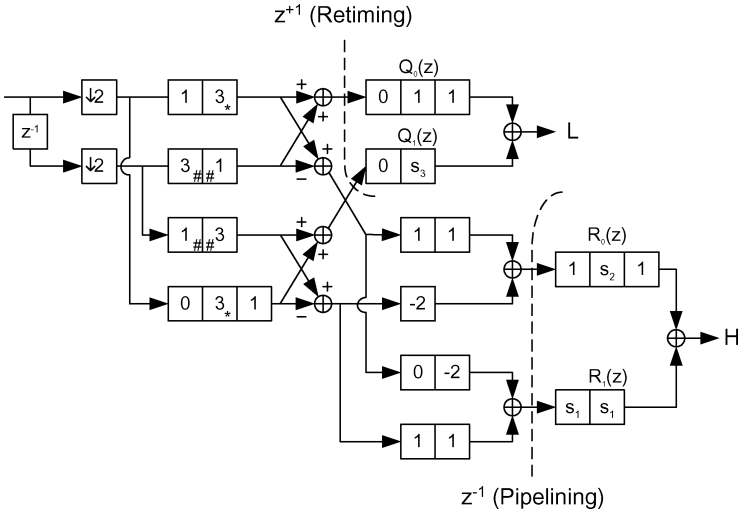
$$= (1-z^{-1})^3(1+r_1z^{-1}+r_2z^{-2} + r_3z^{-3}+r_2z^{-4}+r_1z^{-5}+z^{-6})g_0 \quad (7)$$

where $s_1 = -t_1$, $s_2 = t_2$, $s_3 = -t_3$, $r_1 = 2.630464$, $r_2 = 1.336557$, and $r_3 = -9.934042$. However, the Pascal implementation can only cover $(1 \pm z^{-1})^3$, and

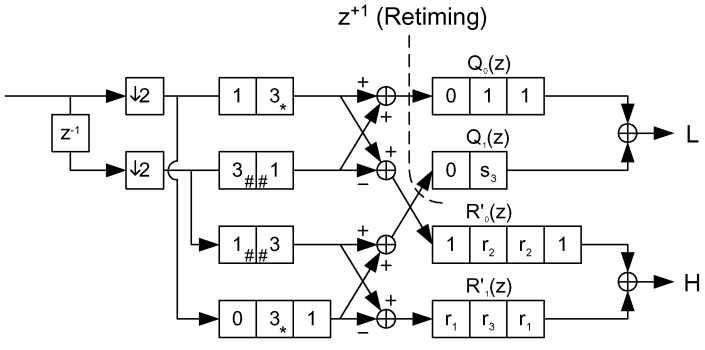
Table 1. Comparisons for DWT architectures of the (9, 7) filter.

Architecture	Multiplier	Adder	Critical path	Register	Timing (ns)	Comb. gate count	Non-comb. gate count
Lifting + no pipelining	4	8	4Tm + 8Ta	4	34	15418.4	796.0
Lifting + 4 pipelining stages	4	8	Tm + 2Ta	10	9.8	15152.0	1495.3
Flipping + no pipelining	4	8	Tm + 5Ta	4	14.1	13326.5	763.7
Flipping + 3 pipelining stages	4	8	Tm + 1Ta	7	7.7	12089.4	1197.7
Convolution + no pipelining	7	14	Tm + 3Ta	7	10.8	17830.5	1266.7
B-spline Type-1	3	18	Tm + 5Ta	8	13.6	9670.4	1271.3
B-splineType-II	3	18	Tm + 2Ta	10	10.3	9419.4	1523.3

there are two solutions for the left part $(1 - z^{-1})^2$ of $G(z)$, Solution-1 and Solution-2, which are corresponding to Eqs. (6) and (7), respectively. The proposed architectures are shown in Fig. 12, where the parts marked with '*' and '##' can be shared. Thus, the Solution-1 of the B-spline factorized architecture would require 3 multipliers and 20 adders while the Solution-2 would need 4 multipliers and 18 adders.



(a) Solution-1 with Type-I polyphase decomposition [Eq. (6)]



(b) Solution-2 with Type-I polyphase decomposition [Eq. (7)]

Figure 12. Proposed B-spline factorized architectures for the (6, 10) filter.

The critical path of the Solution-1 architecture could be T_m+6T_a, T_m+4T_a, or T_m+2T_a by retiming, pipelining, or retiming and pipelining together, respectively. The corresponding numbers of registers are 9, 11, and 13. On the other hand, the Solution-2 architecture can be retimed to obtain a critical path of $T_m + 5T_a$ with totally 9 registers.

5.2.1. Comparison. By extracting the normalization part h_0 and g_0 and utilizing both symmetric and anti-symmetric properties, the convolution-based architecture of the (6, 10) filter can be implemented by use of 6 multipliers, 14 adders, and 8 registers. And the critical path is $T_m + 4T_a$ if the adder tree is used.

In contrast to the odd symmetric (9, 7) filter, the polyphase matrix of the even linear (6, 10) filter can be decomposed as follows:

$$P(z) = \begin{bmatrix} 1 & a \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ b + cz^{-1} & 1 \end{bmatrix} \begin{bmatrix} 1 & e + dz \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 \\ -f + gz^{-1} + fz^{-2} & 1 \end{bmatrix} \begin{bmatrix} K_2 & 0 \\ 0 & K_1 \end{bmatrix} \quad (8)$$

where the coefficients are given as $a = -0.369536$, $b = -0.42780$, $c = -0.119532$, $d = -0.090075$, $e = 0.872739$, $g = -0.572909$, $f = 0.224338$, $K_1 = 0.874919$, and $K_2 = 1.142963$ [5]. Thus, the lifting-based architecture can be shown as Fig. 13, where 7 multipliers, 8 adders, and 5 registers are required if K_1 and K_2 are excluded. The critical path is $4T_m + 5T_a$ without pipelining and can be pipelined to $T_m + 2T_a$ with six pipelining registers. The flipping structure can also reduce the critical path to $T_m + 5T_a$ by flipping and can be further pipelined to $T_m + 2T_a$ with four pipelining registers [7].

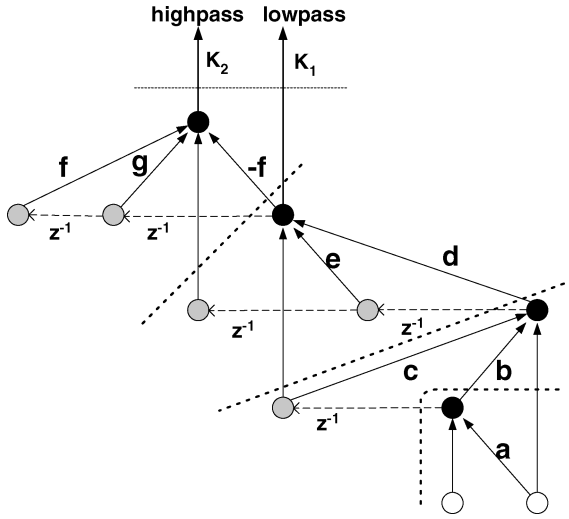


Figure 13. Lifting scheme for the (6, 10) filter.

Similarly, the proposed, convolution-based, and lifting-based architectures have been verified and synthesized. The bit-width is the same as the case of (9, 7) filter. The results are listed in Table 2. In this case, the lifting-based architecture requires even more multipliers than the convolution-based one because the lifting scheme of even-tap linear DWT filters is not as efficient as that of odd symmetric filters. However, the proposed B-spline architecture can still reduce the number of multipliers to three. Table 2 shows that the proposed architectures can achieve the same timing constraints with fewer gate counts than the other three architectures.

Table 2. Comparisons for DWT architectures of the (6, 10) filter.

Architecture	Multiplier	Adder	Critical path	Register	Timing (ns)	Comb. gate count	Non-comb. gate count
Lifting + no pipelining	7	8	4Tm + 5Ta	5	26.6	12664.0	929.33
Lifting + 4 pipelining stages	7	8	Tm + 2Ta	11	9.15	14181.7	1679
Flipping + no pipelining	7	8	Tm + 5Ta	5	13.6	12525.3	1007
Flipping + 3 pipelining stages	7	8	Tm + 2Ta	9	8.7	12304.0	1423.0
Convolution + no pipelining	6	14	Tm + 4Ta	8	12	13685.4	1332
Bspline-1 Type-I + retiming	3	20	Tm + 6Ta	9	14.05	9623.7	1322.3
Bspline-1 Type-I + retiming + pipe.	3	20	Tm + 4Ta	11	11.5	8788.7	1571.0
Bspline-1 Type-I + pipelining	3	20	Tm + 2Ta	13	9.5	8648.7	1782.0
Bspline-2 Type-I + retiming	4	18	Tm + 5Ta	9	13.1	11647.7	1366.7

Table 3. Detailed gate count comparison for the (6, 10) filter.

Architecture	Cell name	Cell number	Ave. gate count	Total gate count
Lifting + 4 pipelining stages	nbw	7	1476.85	10337.95
	cla	5	482.73	3611.97
	bk	3	399.44	
B-spline-1 Type-l + pipelining	nbw	3	1071.11	3213.33
	cla	1	487.67	
	bk	9	290.78	5256.67
	rpl	9	207.59	
	rpcs	1	283.67	

nbw: non-booth-recorded wallace tree multiplier; *cla*: carry-lookahead adder; *bk*: brent-kung adder; *rpl*: ripple carry adder; *rpcs*: ripple carry select adder.

5.2.2. Detailed Gate Count Comparison. The B-spline factorized architecture can provide fewer multipliers but introduce more adders. We compare the gate counts of multipliers and adders in more detail to examine the resulting hardware resource reduction. The lifting-based architecture with four pipelining stages and the B-spline Solution-1 architecture with pipelining are chosen, which are both of critical path $T_m + 2T_a$.

The detailed comparison of the gate counts is listed in Table 3, where the gate counts of different kinds of multipliers and adders are separate. The Synopsys Design Compiler synthesizes all multipliers to non-booth-recorded wallace tree multipliers, which can have trade-offs between the processing speed and the area size. Many kinds of adders are used for circuits synthesis, and the carry-lookahead adders are the fastest but the largest ones.

All multipliers of the lifting-based architecture are on the critical path, so the gate counts of them are quite large and about 1500 gates in average. However, the multipliers of the B-spline factorized architecture are not all on the critical path, so the average gate count is only about 1000 gates. Furthermore, the lifting-based architecture requires 4 more multipliers than the B-spline factorized one. In the result, the total gate counts of multipliers are about 10000 and 3000 gates, respectively.

On the other hand, only one carry-lookahead adder is used in the proposed architecture while five are used in the lifting-based one. Although more adders are required, most of them are synthesized to the smaller adders in the proposed architecture. The overhead gate count of adders for the proposed architecture is about 1600 gates. By combining the result of multipliers, the net reduction of gate count is about $7000 - 1600 = 5400$. The efficiency of the proposed architecture for reducing multipliers is demonstrated.

5.3. The (10, 18) Filter

The coefficients of the (10, 18) analysis filter bank are given in [13]. The analysis lowpass filter is a symmetric 10-tap filter, and the highpass filter is an anti-symmetric 18-tap filter. The coding efficiency can be better than the well-known (9, 7) filter [13, 16]. The B-spline factorization of the analysis filter bank is as follows:

$$\begin{aligned} H(z) &= (1+z^{-1})^5(u_1 z^{-8} + u_2 z^{-7} + z^{-6} \\ &\quad + u_2 z^{-5} + u_1 z^{-4}) \cdot h_0 \\ G(z) &= (1-z^{-1})^9(u_3 z^{-8} + u_4 z^{-7} + u_5 z^{-6} + u_6 z^{-5} \\ &\quad + z^{-4} + u_6 z^{-3} + u_5 z^{-2} + u_4 z^{-1} + u_3) \cdot g_0 \end{aligned} \tag{9}$$

where $u_1 = 0.1049758$, $u_2 = -0.524577$, $u_3 = 0.0094393$, $u_4 = 0.08498056$, $u_5 = 0.33152476$, $u_6 = 0.74232477$, $h_0 = 0.27485$, and $g_0 = 0.101111$.

For the (10, 18) filter bank, the Pascal implementation will be too complex to derive because the degrees of the B-spline parts are 5 and 9. Thus, we use the direct implementation for the B-spline part. The proposed architecture for the (10, 18) filter is as shown in Fig. 14, where 6 multipliers and 40 adders are used if the normalization part is excluded. If retiming z^{+2} is performed, the critical path will become $T_m + 11T_a$ with totally 23 registers. In concept, we can reduce the critical path to $\frac{T_m + 11T_a}{2}$ by pipelining with 4 additional registers.

5.3.1. Comparison. Here we consider that the convolution-based architecture of the (10, 18) filter is implemented into the parallel filters. If the linear property and the adder tree are adopted, 12 multipliers, 26 adders, and 16 registers are required while the critical path is $T_m + 5T_a$. As the case of (6, 10) filter, the lifting scheme of the (10, 18) cannot be linear and cannot reduce the hardware complexity. Thus, we will not include the lifting scheme into the comparison.

The proposed and convolution-based architectures have been verified and synthesized. The internal bit-width is the same as the case of (9, 7) filter, except the multipliers become 16-by-16 multiplications. The results are listed in Table 4. The pipelining of the proposed architecture is cut before the last two $1 + z^{-1}$ stages as shown in Fig. 14 for this cell library. The

Table 4. Comparisons for DWT architectures of the (10, 18) filter.

Architecture	Multiplier	Adder	Critical path	Register	Timing (ns)	Comb. gate count	Non-comb. gate count
Convolution	12	26	Tm + 5Ta	16	13.5	27173.4	2400.3
Bspline + retiming (z^{+2})	6	40	Tm + 11Ta	23	21.8	19193.9	2617.0
Bspline + retiming (z^{+1}) + pipe.	6	40	∼(Tm + 11Ta)/2	27	12.7	20050.2	3109.7

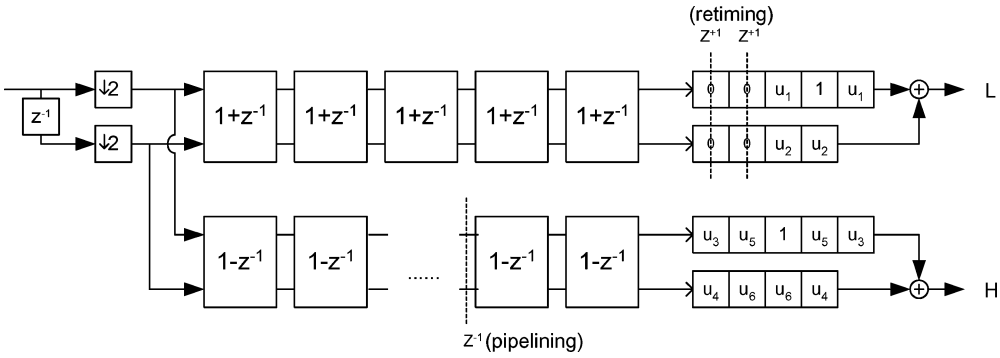


Figure 14. B-spline factorized architecture for the (10, 18) filter.

proposed architectures require only about two-thirds of the gate count of the convolution-based one.

6. Conclusion

In this paper, a new category of DWT architectures is proposed on the basis of B-spline factorization. The B-spline part can be implemented by use of the direct or Pascal implementation. And the distributed part could be implemented with the Type-I or Type-II polyphase decomposition and conventional filter design techniques. For Daubechies wavelets, the proposed B-spline factorized architectures can guarantee the complexity reduction of multipliers by 2 while the lifting scheme cannot. Although more adders are required, many adders can be implemented in small area and low speed because most of them are not on the critical path. Based on three case studies, including the (9, 7), (6, 10), and (10, 18) filters, the required gate counts of the proposed architecture are much smaller than that of the convolution-based and lifting-based ones, which demonstrates the efficiency.

Acknowledgment

This work was supported in part by MOE Program for Promoting Academic Excellence of Universities under the grant number 89E-FA06-2-4-8, in part by National Science Council, Republic of China, under the grant number 91-2215-E-002-035, and in part by MediaTek Fellowship.

References

1. D. Taubman, "Successive Refinement of Video: Fundamental Issues, Past Efforts and New Directions," in *International Symposium on Visual Communications and Image Processing*, 2003.
2. K.K. Parhi and T. Nishitani, "VLSI Architectures for Discrete Wavelet Transforms," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 1, no. 2, pp. 191–202, 1993.
3. M. Vishwanath, R.M. Owens, and M.J. Irwin, "VLSI Architectures for the Discrete Wavelet Transform," *IEEE Transactions on Circuis and Systems-II: Analog and Digital Signal Processing*, vol. 42, no. 5, 1995, pp. 305–316.
4. C. Chakrabarti, M. Vishwanath, and R.M. Owens, "Architectures for Wavelet Transforms: A Survey," *Journal of VLSI Signal Processing*, vol. 14, 1996, pp. 171–192.
5. K. Andra, C. Chakrabarti, and T. Acharya, "A VLSI Architecture for Lifting-Based Forward and Inverse Wavelet Transform," *IEEE Transactions on Signal Processing*, vol. 50, no. 4, 2002, pp. 966–977.
6. W. Jiang and A. Ortega, "Lifting Factorization-Based Discrete Wavelet Transform Architecture Design," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 5, 2001, pp. 651–657.
7. C.-T. Huang, P.-C. Tseng, and L.-G. Chen, "Flipping Structure: An Efficient VLSI Architecture for Lifting-Based Discrete Wavelet Transform," *IEEE Transactions on Signal Processing*, vol. 52, no. 4, 2004, pp. 1080–1089.
8. P.P. Vaidyanathan, *Multirate Systems and Filter Banks*, Prentice Hall, 1993.

9. K.K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*, John Wiley & Sons, 1999.
10. W. Sweldens, "The Lifting Scheme: A Custom-Design Construction of Biorthogonal Wavelets," *Applied and Computaional Harmonic Analysis*, vol. 3, no. 15, 1996, pp. 186–200.
11. I. Daubechies and W. Sweldens, "Factoring Wavelet Transforms Into Lifting Steps," *The Journal of Fourier Analysis and Applications*, vol. 4, 1998, pp. 247–269.
12. M. Unser and T. Blu, "Wavelet Theory Demystified," *IEEE Transactions on Signal Processing*, vol. 51, no. 2, 2003, pp. 470–483.
13. M.J. Tsai, J.D. Villasenor, and F. Chen, "Stack-Run Image Coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, no. 5, 1996, pp. 519–521.
14. S.G. Mallat, "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 7, 1989, pp. 674–693.
15. S. Mallat, *A Wavelet Tour of Signal Processing*, Academic Press, 1998.
16. N. Polyak and W.A. Pearlman, "A New Flexible Bi-Orthogonal Filter Design for Multiresolution Filterbanks with Application to Image Compression," *IEEE Transactions on Signal Processing*, vol. 48, no. 8, 2000, pp. 2279–2288.

Chao-Tsung Huang was born in Kaohsiung, Taiwan, R.O.C., in 1979. He received the B.S. degree from the Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan, R.O.C., in 2001. He currently is working toward the Ph.D. degree at the Graduate Institute of Electronics Engineering, National Taiwan University. His major research interests include VLSI design and implementation for signal processing systems.
cthuang@video.ee.ntu.edu.tw

Po-Chih Tseng was born in Tao-Yuan, Taiwan in 1977. He received the B.S. degree in Electrical and Control Engineering from National Chiao Tung University in 1999 and the M.S. degree in Electrical Engineering from National Taiwan University in 2001. He currently is pursuing the Ph.D. degree at the Graduate Institute of Electronics Engineering, Department of Electrical Engineering, National Taiwan University. His research interests include VLSI design and implementation for signal processing systems, energy-efficient reconfigurable computing for multimedia systems, and power-aware image and video coding systems.
pctseng@video.ee.ntu.edu.tw

Liang-Gee Chen received the B.S., M.S., and Ph.D. degrees in electrical engineering from National Cheng Kung University, Tainan, Taiwan, R.O.C., in 1979, 1981, and 1986, respectively.

In 1988, he joined the Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan, R.O.C. During 1993–1994, he was a Visiting Consultant in the DSP Research Department, AT&T Bell Labs, Murray Hill, NJ. In 1997, he was a Visiting Scholar of the Department of Electrical Engineering, University of Washington, Seattle. Currently, he is Professor at National Taiwan University, Taipei, Taiwan, R.O.C. His current research interests are DSP architecture design, video processor design, and video coding systems.

Dr. Chen has served as an Associate Editor of IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY since 1996, as Associate Editor of the IEEE TRANSACTIONS ON VLSI SYSTEMS since 1999, and as Associate Editor of IEEE TRANSACTIONS CIRCUITS AND SYSTEMS II since 2000. He has been the Associate Editor of the *Journal of Circuits, Systems, and Signal Processing* since 1999, and a Guest Editor for the *Journal of Video Signal Processing Systems.* He is also the Associate Editor of the PROCEEDINGS OF THE IEEE. He was the General Chairman of the 7th VLSI Design/CAD Symposium in 1995 and of the 1999 IEEE Workshop on Signal Processing Systems: Design and Implementation. He is the Past-Chair of Taipei Chapter of IEEE Circuits and Systems (CAS) Society, and is a member of the IEEE CAS Technical Committee of VLSI Systems and Applications, the Technical Committee of Visual Signal Processing and Communications, and the IEEE Signal Processing Technical Committee of Design and Implementation of SP Systems. He is the Chair-Elect of the IEEE CAS Technical Committee on Multimedia Systems and Applications, During 2001-2002, he served as a Distinguished Lecturer of the IEEE CAS Society. He received the Best Paper Award from the R.O.C. Computer Society in 1990 and 1994. Annually from 1991 to 1999, he received Long-Term (Acer) Paper Awards. In 1992, he received the Best Paper Award of the 1992 Asia-Pacific Conference on circuits and systems in the VLSI design track. In 1993, he received the Annual Paper Award of the Chinese Engineer Society. In 1996 and 2000, he received the Outstanding Research Award from the National Science Council, and in 2000, the Dragon Excellence Award from Acer. He is a member of Phi Tan Phi.
lgchen@video.ee.ntu.edu.tw