Open access • Journal Article • DOI:10.1109/TSP.2005.843704

Analysis and VLSI architecture for 1-D and 2-D discrete wavelet transform

— Source link

Chao-Tsung Huang, Po-Chih Tseng, Liang-Gee Chen

Institutions: National Taiwan University

Published on: 01 Apr 2005 - IEEE Transactions on Signal Processing (IEEE)

Topics: Auxiliary memory, Discrete wavelet transform, Lifting scheme and Very-large-scale integration

Analysis and VLSI Architecture for 1-D and 2-D Discrete Wavelet Transform

Chao-Tsung Huang, Po-Chih Tseng, and Liang-Gee Chen, *Fellow, IEEE*

Abstract—**In this paper, a detailed analysis of very large scale integration (VLSI) architectures for the one-dimensional (1-D) and two-dimensional (2-D) discrete wavelet transform (DWT) is presented in many aspects, and three related architectures are proposed as well. The 1-D DWT and inverse DWT (IDWT) architectures are classified into three categories: convolution-based, lifting-based, and B-spline-based. They are discussed in terms of hardware complexity, critical path, and registers. As for the 2-D DWT, the large amount of the frame memory access and the die area occupied by the embedded internal buffer become the most critical issues. The 2-D DWT architectures are categorized and analyzed by different external memory scan methods. The implementation issues of the internal buffer are also discussed, and some real-life experiments are given to show that the area and power for the internal buffer are highly related to memory technology and working frequency, instead of the required memory size only. Besides the analysis, the B-spline-based IDWT architecture and the overlapped stripe-based scan method are also proposed. Last, we propose a flexible and efficient architecture for a one-level 2-D DWT that exploits many advantages of the presented analysis.**

Index Terms—**B-spline factorization, discrete wavelet transform, lifting scheme, line-based method, VLSI architecture.**

I. Introduction

The DISCRETE wavelet transform (DWT) has been developed as an efficient DSP tool for signal analysis, image compression, and even video compression [1]. There are many architectures proposed for the implementation of DWT. For the 1-D DWT, the architectures can be categorized into the convolution-based [2], lifting-based [3], [4], and B-spline-based [5]. The first one is to implement two-channel filterbanks directly. The second one is to exploit the relationship of lowpass and highpass filters for saving multipliers and adders [6], [7]. The third one can reduce the multipliers based on the B-spline factorization [5]. The B-spline-based architectures could provide fewer multipliers while the lifting scheme fails to reduce the complexity.

When extending the 1-D DWT module to the 2-D DWT architecture, the memory issue is the most important design consideration because the 2-D DWT requires a large amount of data access and storage. The design tradeoff mainly comes from the frame memory access bandwidth and the internal buffer size. In [8], the design alternatives are evaluated in the aspects of power and memory requirements. However, the evaluation only covers three specified 2-D architectures. The frame memory is usually off-chip so that the external frame memory access would consume the most power and waste much system memory bandwidth. As the cache is used to reduce the main memory access in the general processor architectures, the internal buffer is used to reduce the frame memory access for 2-D DWT. However, the internal buffer would occupy much die area. Many 2-D DWT architectures using different memory structures have been proposed [9]–[13].

In this paper, we discuss the performance of different 1-D and 2-D DWT/IDWT architectures. The 1-D DWT and IDWT architectures are analyzed in terms of hardware complexity, speed, and register number. Then two independent issues of the 2-D DWT architectures are discussed. The first one is the tradeoff between the external frame memory access and the internal buffer size. The second one is the real-life implementation methods for the internal buffer. Furthermore, the B-spline-based IDWT architecture and one efficient frame memory scan method are also proposed. Last of all, a flexible and efficient 2-D DWT architecture is proposed.

The organization of this paper is as follows. The 1-D DWT and IDWT architectures are analyzed in Section II in which the B-spline-based IDWT architecture is also proposed. Then, we categorize previous 2-D DWT architectures and propose an efficient scan method in Section III. In Section IV, we propose a flexible and efficient architecture for one-level 2-D DWT using the performance analysis. A summary is given to conclude this paper in Section V.

II. One-Dimensional DWT and IDWT Architectures

The multiresolution DWT and IDWT can be viewed as cascades of several two-channel analysis and synthesis filterbanks, respectively [14]. The analysis and synthesis filterbanks are shown in Fig. 1, where $H(z)$ and $G(z)$ are the analysis lowpass and highpass filters. For perfect reconstruction, the synthesis lowpass and highpass filters can be defined as

$$\begin{aligned}\widetilde{H}(z) &= G(-z)\\ \widetilde{G}(z) &= -H(-z)\end{aligned} \tag{1}$$

Many very large scale intergration (VLSI) architectures have been proposed to implement DWT and IDWT, and they can be categorized into three categories, as shown in Fig. 2, including the convolution-based, lifting-based, and B-spline-based architectures. In the following, these three categories of architectures

Manuscript received November 14, 2003; revised April 22, 2004. This work was supported in part by the MOE Program for Promoting Academic Excellence of Universities under Grant 89E-FA06-2-4-8, in part by National Science Council, Republic of China, under Grant 91-2215-E-002-035, and in part by the MediaTek Fellowship. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Chaitali Chakrabarti.

The authors are with Graduate Institute of Electronics Engineering and Department of Electrical Engineering, National Taiwan University, Taipei 106, Taiwan, R.O.C. (e-mail: lgchen@video.ee.ntu.edu.tw).

Digital Object Identifier 10.1109/TSP.2005.843704

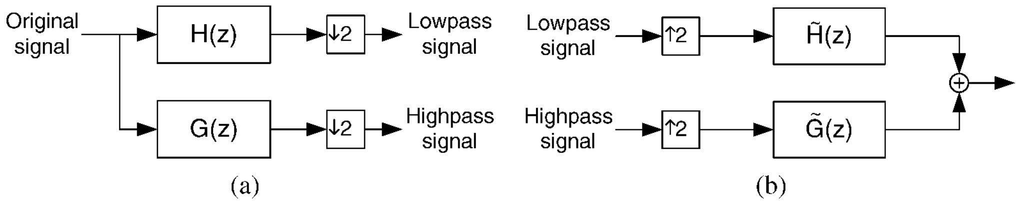


Fig. 1. Two-channel filterbank for DWT and IDWT. (a) Analysis filterbank for forward DWT. (b) Synthesis filterbank for inverse DWT.

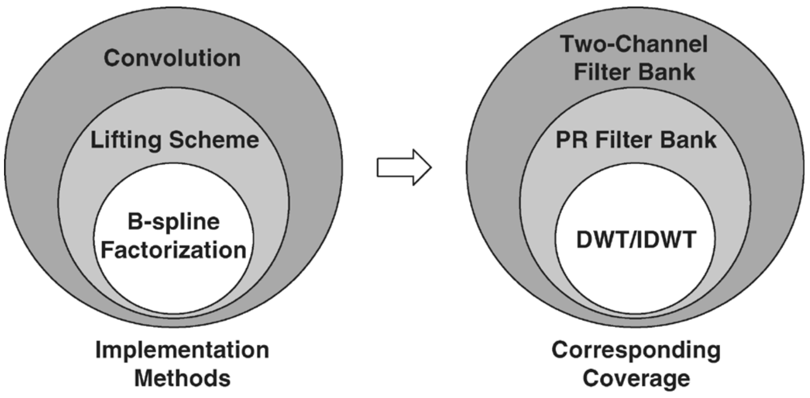


Fig. 2. Categories of DWT and IDWT architectures and the corresponding coverage.

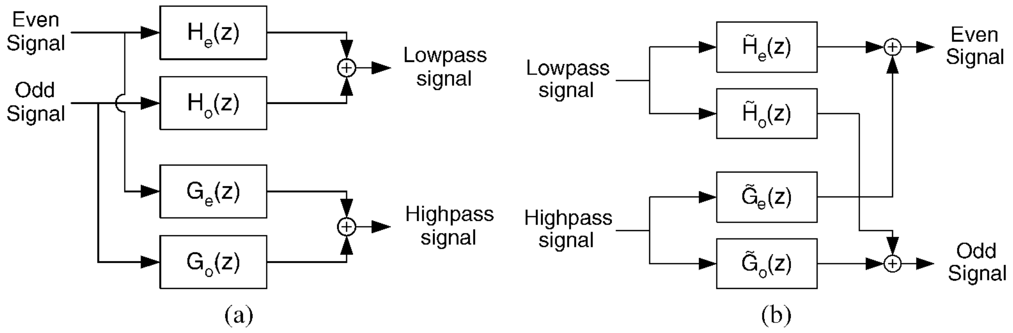


Fig. 3. Polyphase decomposition of two-channel filterbanks. (a) Analysis filterbank. (b) Synthesis filterbank.

are introduced first, and then, the general performance analysis is given.

A. Convolution-Based

The straightforward implementation for DWT and IDWT is to construct the lowpass and highpass filters independently with fundamental VLSI DSP design techniques, such as folding, unfolding, and pipelining [15]. For 100% hardware utilization, the polyphase decomposition [16] is usually adopted, and there are two possible decomposition types as follows:

Type-I decomposition:

$$\begin{aligned} H(z) &= H_e(z^2) + z^{-1}H_o(z^2) \\ G(z) &= G_e(z^2) + z^{-1}G_o(z^2) \\ \widetilde{H}(z) &= \widetilde{H}_e(z^2) + z^{-1}\widetilde{H}_o(z^2) \\ \widetilde{G}(z) &= \widetilde{G}_e(z^2) + z^{-1}\widetilde{G}_o(z^2). \end{aligned} \tag{2}$$

Type-II decomposition:

$$\begin{aligned} &H(z)=H_e(z^2)+zH_o(z^2), \quad G(z)=G_e(z^2)+zG_o(z^2) \\ &\widetilde{H}(z)=\widetilde{H}_e(z^2)+z\widetilde{H}_o(z^2), \quad \widetilde{G}(z)=\widetilde{G}_e(z^2)+z\widetilde{G}_o(z^2). \end{aligned} \tag{3}$$

After polyphase decomposition, the convolution-based architectures for DWT and IDWT can be constructed as Fig. 3, where two-input two-output per cycle is assumed to achieve 100% hardware utilization. Then, the four separate filters of Fig. 3(a) or (b) can be implemented by use of serial or parallel filters.

B. Lifting-Based

The lifting scheme [6] has been widely used to reduce the required multiplications and additions by exploring the relation of lowpass and highpass filters spatially. According to [7], any DWT filterbank 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:

$$\begin{aligned} P(z) &= \begin{bmatrix} H_e(z) & G_e(z) \\ H_o(z) & G_o(z) \end{bmatrix} \\ &= \prod_{i=1}^{m} \begin{bmatrix} 1 & s_i(z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ t_i(z) & 1 \end{bmatrix} \begin{bmatrix} K_n & 0 \\ 0 & \frac{1}{K_n} \end{bmatrix} \end{aligned} \tag{4}$$

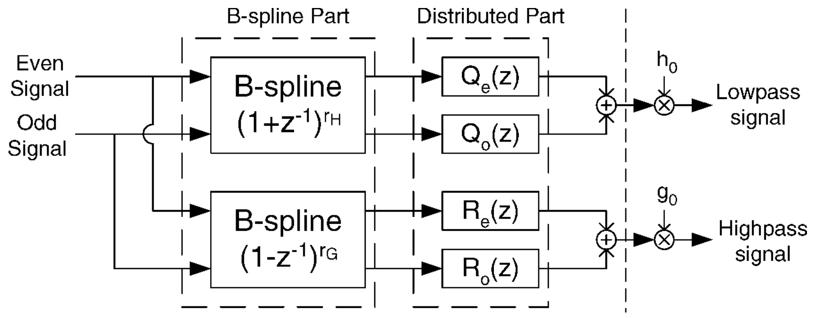


Fig. 4. B-spline-based architecture for DWT.

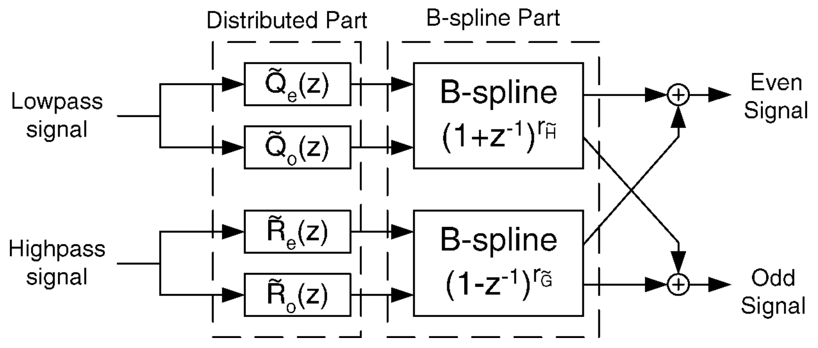


Fig. 5. Proposed B-spline-based architecture for IDWT.

where $P(z)$ is the polyphase matrix. The lifting scheme of IDWT is similar to that of DWT and can be derived from the above equation.

Most of the lifting-based architectures in the literature are implemented with the above lifting factorization directly [3], [11]. Although the lifting scheme can provide many advantages, such as fewer arithmetic operations and in-place implementation, the potentially long critical path is a drawback for hardware implementation. In [4], this timing crisis is discussed in detail and addressed by use of the flipping structure instead of pipelining.

C. B-Spline-Based

1) Previous Work: According to [17], the lowpass and highpass filters of any DWT can be factorized as

$$\begin{aligned} H(z) &= \underline{(1+z^{-1})^{\gamma_H}} \cdot \underline{Q(z)} \cdot \underline{h_0} \\ G(z) &= \underline{(1-z^{-1})^{\gamma_G}} \cdot \underline{R(z)} \cdot \underline{g_0} \end{aligned} \tag{5}$$

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. 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. The distributed part is used to derive efficient finite impulse response DWT filters [17]. 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 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. The B-spline-based architecture for DWT has been proposed in [18], as shown in Fig. 4.

2) Proposed B-Spline-Based Architecture for IDWT: Using (1) and (5), we can derive the similar B-spline factorization for IDWT as follows:

$$\begin{aligned} \tilde{H}(z) &= (1+z^{-1})^{\gamma_{\tilde{H}}} \cdot \tilde{Q}(z) \\ \tilde{G}(z) &= (1-z^{-1})^{\gamma_{\tilde{G}}} \cdot \tilde{R}(z). \end{aligned} \tag{6}$$

Then, substituting this equation into Fig. 1(b), the general architecture can be derived by use of polyphase decomposition [16], as shown in Fig. 5, where $\tilde{Q}(z) = \tilde{Q}_e(z^2) + z^{-1}\tilde{Q}_o(z^2)$ and $\tilde{R}(z) = \tilde{R}_e(z^2) + z^{-1}\tilde{R}_o(z^2)$.

Similar to the B-spline-based DWT architectures, the IDWT architecture in Fig. 5 comprises the distributed part and the B-spline part. The normalization part is not extracted because usually, no other operations need to be performed before IDWT. The four filters of the distributed part ($\tilde{Q}_e(z)$, $\tilde{Q}_o(z)$, $\tilde{R}_e(z)$, and $\tilde{R}_o(z)$) can be implemented by use of many DSP VLSI techniques, such as serial filter, parallel filter, pipelining, and retiming. The B-spline part can be constructed with the direct method or the Pascal method [18]. The Pascal implementation exploits the similarity of the two B-spline parts to reduce adders. However, the Pascal implementation of long-tap filters will be too complex to be derived, and the complexity reduction is not guaranteed. Then, the direct implementation of the B-spline parts can be used instead.

The concept of the direct method 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 $(1-z^{-1})$, respectively. However, two-input-two-output structures of $(1+z^{-1})$ and $(1-z^{-1})$ cannot be derived from polyphase decomposition directly. Here, we propose to implement them by considering the physical connection of signals, as shown in Fig. 6, where the even signals are assumed to be prior to the odd signals. When

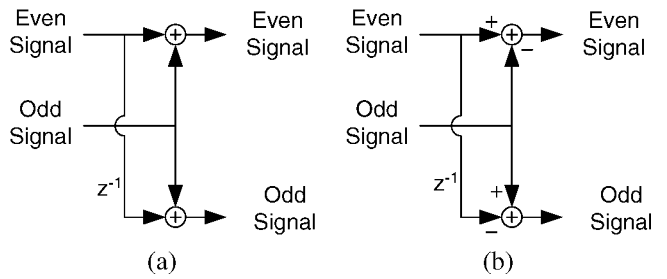


Fig. 6. Direct implementation for the B-spline part. (a) For $(1+z^{-1})$. (b) For $(1-z^{-1})$.

connecting the B-spline part to the distributed part, the order of even and odd signals needs to be handled carefully.

D. Case Study

Since some cases of DWT have been studied in [18], we only study the case of IDWT in this paper. The (10,18) filter is chosen because its hardware complexity is quite high, and the corresponding lifting scheme cannot reduce the complexity as the (6,10) filter.

1) (10,18) Filter: The coefficients of the (10,18) analysis filterbank are given in [19]. The analysis lowpass filter is a symmetric 10-tap filter, and the highpass filter is an antisymmetric 18-tap filter. The coding efficiency can be better than the well-known JPEG 2000 lossy (9,7) filter [19], [20]. The synthesis filterbank can be derived from (1).

By use of polyphase decomposition, the convolution-based architecture of the synthesis (10,18) filter can be constructed as Fig. 7, where h_i's and g_i's are the coefficients of $\underset{\sim}{H}(z)$ and $\underset{\sim}{G}(z)$, respectively, and the filter notation of Fig. 8 is used. $\underset{\sim}{H}_e(z)$ and $\underset{\sim}{H}_o(z)$ are mirror images of each other, and so are $\widetilde{G}_e(z)$ and $\widetilde{G}_o(z)$. Thus, the number of multipliers can be reduced by using mirror images, but it would require more registers. Here, we discuss two possible architectures. The architecture I implements the four filters as parallel filters directly to minimize the number of registers. The required numbers of multipliers, adders, and registers are 26, 26, and 14, respectively. (h_0 and g_0 can be shared because they are parallel filters.) The architecture II minimizes the number of multipliers by use of mirror images. The required numbers of multipliers, adders, and registers are 14, 26, and 24, respectively. The critical path of these two architectures are both $T_m + 4T_a$, where T_m and T_a are the execution time taken for a multiplier and an adder, respectively.

2) Proposed B-Spline-Based Architecture: The B-spline factorization of the synthesis filterbank is as follows:

$$\begin{aligned}\widetilde{H}(z) &= \frac{(1+z^{-1})^9}{8}(v_1z^{-8}+v_2z^{-7}+v_3z^{-6}+v_4z^{-5}+v_5z^{-4}\\ &\qquad +v_4z^{-3}+v_3z^{-2}+v_2z^{-1}+v_1)\\ \widetilde{G}(z) &= \frac{(1-z^{-1})^5}{4}(v_6z^{-8}+v_7z^{-7}+v_8z^{-6}+v_7z^{-5}\\ &\qquad +v_6z^{-4}) \end{aligned} \tag{7}$$

where $v_1 = 0.007\,653\,5$, $v_2 = -0.068\,739\,8$, $v_3 = 0.268\,166\,4$, $v_4 = -0.600\,457\,6$, $v_5 = 0.808\,888$, $v_6 = -0.115\,410\,4$, $v_7 = -0.576\,72$, and $v_8 = -1.0994$.

The proposed architecture of the synthesis (10,18) filterbank is shown in Fig. 9. The four filters of the distributed part are all symmetric such that the number of multipliers can be reduced by half. The denominators (8 and 4) of the (7) are introduced only for the precision issues. These two denominators are implemented by shifting right the signals between the $(1+z^{-1})$ and $(1-z^{-1})$ stages, which are not shown in Fig. 9 for simplicity.

There are two possible implementations for these filters: serial or parallel filters. If the four filters are implemented as serial filters, the critical path will be $T_m + 11T_a$. Because the registers cannot be shared among these filters, the required number of registers is 24. On the other hand, the critical path will be $T_m + 13T_a$, the registers can be shared, and the number is reduced to 20 if parallel filters are adopted. In Fig. 9, retiming can be performed to $\widetilde{R}_e(z)$ and $\widetilde{R}_o(z)$ to decrease the number of registers. Pipelining can also be used to shorten the critical path. In concept, the pipeline can be cut at half the critical path with four additional pipelining registers.

3) Comparison: The proposed B-spline factorized architectures as well as the convolution-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 I, where the internal wordlengths 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 in combinational and noncombinational parts separately. The former contributes to the multipliers and adders, whereas the latter is responsible to the registers. For circuit synthesis, the timing constraints are set as tight as possible. The lifting-based architecture cannot reduce the hardware complexity as the case of the (6,10) filter because the lifting steps cannot be all linear. Thus, the lifting-based architecture is not included in this comparison.

For this cell library, we choose the pipelining points for the proposed architectures, as shown in Fig. 9, to minimize the critical path. According to Table I, the proposed architectures could require fewer gate counts under the same timing constraints. Although the proposed architectures need more adders, most of the adders are not on the critical path such that they can be implemented with smaller area and lower speed.

E. Performance Analysis

In the following, we analyze the performance of the three categories of architectures for DWT. The result of IDWT is similar to the DWT. Furthermore, the DWT filters are assumed to be Daubechies wavelets that are optimal in the sense that they have a minimum size support with a given number of vanishing moments [21]. In other words, all DWT filter lengths are constrained with the given vanishing moments γ_H and γ_G, as follows:

$$F_H + F_G \geq 2(\gamma_H + \gamma_G) \tag{8}$$

where F_H and F_G are the filter lengths of the lowpass and highpass DWT filters, respectively. Daubechies wavelets satisfy the equal sign. In practical applications, Daubechies wavelets are usually adopted because there is no reason to use a longer filter

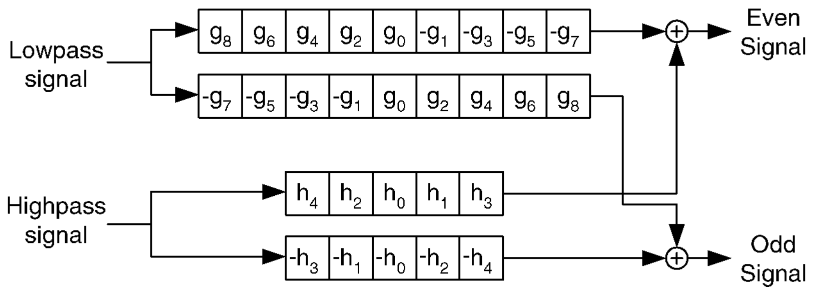


Fig. 7. Convolution-based architecture for IDWT with the (10,18) filter.

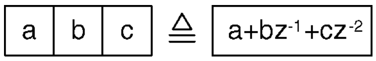


Fig. 8. Notation for filters.

of the same vanishing moment. Besides the general case, the linear filters are discussed as well. From the length condition in [22], we can derive the following:

$$F_H + F_G = 4k \tag{9}$$

where k is a positive integer. Thus, F_H and F_G can only be both odd or even. The odd-tap linear lowpass and highpass filters are both symmetric. The even-tap linear lowpass filters are symmetric, but the even-tap highpass filters are antisymmetric.

1) Multipliers and Adders: In the general case, the convolution-based architecture requires $(F_H + F_G)$ multipliers and $(F_H + F_G - 2)$ adders. For the odd-tap linear filter, the four filters in Fig. 3(a) are all symmetric such that in total, $(((F_H + F_G)/2) + 1)$ multipliers and $(F_H + F_G - 2)$ adders are required. For the even-tap filter, $H_e(z)$ and $H_o(z)$ are mirror images of each other, and so are $G_e(z)$ and $G_o(z)$. Thus, only $(F_H + F_G)/2$ multipliers and $(F_H + F_G - 2)$ adders are needed.

For comparison, the following analysis of lifting scheme will contain the normalization step. According to [7], the complexity of lifting scheme can be approximated with the assumption that all lifting steps are of degree one, except for the final stage. Then the normalization step requires two multipliers, and so does each of the $\lceil (F_G + 1)/2 \rceil$ lifting steps. The final lifting step needs $(\lceil (F_H + 1)/2 \rceil - \lceil (F_G + 1)/2 \rceil + 1 = ((F_H - F_G)/2) + 1)$ multipliers. In total, lifting scheme requires $(\lceil (F_H + 1)/2 \rceil + \lceil (F_G + 1)/2 \rceil + 3)$ multipliers in the general case. For odd-tap linear filters, each lifting step can be also linear because $H_e(z)$ and $H_o(z)$ are both symmetric. Thus, each of the $(F_G + 1)/2$ lifting steps requires only one multiplier, and the final lifting step needs $((F_H - F_G + 2)/4)$ multipliers. Therefore, in total, $(((F_H + F_G)/4) + 1)$ multipliers are required. For even-tap linear filters, the lifting step cannot be factorized into linear phase because the $H_e(z)$ is only the mirror image of $H_o(z)$. The number of required multipliers is the same as that of the general case. As for adders, each of the $\lceil (F_G + 1)/2 \rceil$ lifting steps requires two adders, and the final one requires $(\lceil (F_H + 1)/2 \rceil - \lceil (F_G + 1)/2 \rceil + 1)$ adders. For all cases, the number of adders is $(\lceil (F_H + 1)/2 \rceil + \lceil (F_G + 1)/2 \rceil + 1)$. The normalization part is also not extracted for the B-spline-based architectures here. Based on the definition of Daubechies wavelets, the following equation can be derived:

$$\begin{aligned} & F_H + F_G = (\gamma_H + F_Q) + (\gamma_G + F_R) = 2(\gamma_H + \gamma_G) \\ \Longrightarrow \quad & F_Q + F_R = \gamma_H + \gamma_G = \frac{F_H + F_G}{2} = 2k \end{aligned} \tag{10}$$

where F_Q and F_R are the lengths of $Q(z)$ and $R(z)$, respectively. For the general case, the B-spline-based architecture requires $(F_Q + F_R = (F_H + F_G)/2)$ multipliers. For the linear filters, $Q(z)$ and $R(z)$ are also linear because of the linearity of the B-spline part. Thus, they can be implemented with the symmetric property, and the number of required multipliers is $(\lceil (F_Q+1)/2 \rceil + \lceil (F_R+1)/2 \rceil)$ that can be simplified to $(((F_H + F_G)/4) + 1)$ or $(((F_H + F_G)/4) + 2)$ according to (10). Assume that the direct implementation is used for the B-spline part. Then, the number of required adders is $(F_Q - 1 + F_R - 1 + 2(\gamma_H + \gamma_G) = 3/2(F_H + F_G) - 2)$.

The above-deduced results are summarized in Table II. Relative to the convolution-based architectures, the B-spline-based ones can reduce the multipliers into by half in all cases. The lifting-based ones can reduce by half the multipliers for the general case and the odd-tap linear filters. For the even-tap linear filters, the lifting-based architectures fail to reduce multipliers and even require more multipliers than the convolution-based ones. However, the B-spline-based ones need about 1.5 times of adders than the convolution-based ones, whereas the lifting-based ones need only about one half the adders of the latter. However, the complexity of adders is always much less than that of multipliers. In summary, the B-spline-based architectures require the fewest multipliers but the most adders. The lifting-based ones fail to reduce the multipliers for the even-tap linear filters but always require the fewest adders.

2) Critical Path and Registers: The above analysis can be used to evaluate the performance of both hardware and software implementations. For hardware implementation, the critical path and the register number are very important. However, they are hard to estimate exactly because pipelining can be used to reduce the critical path with additional registers. Here, we approximate them without pipelining and ignore the difference of $O(1)$, based on the assumption that $F = \max(F_H, F_G)$ is very large.

If the convolution-based architectures are implemented with parallel filters, there will be about F registers, and the critical path is about $T_m + \log_2 F \cdot T_a$ by the use of the adder tree. The critical path can be reduced to $T_m + 2T_a$ if serial filters are

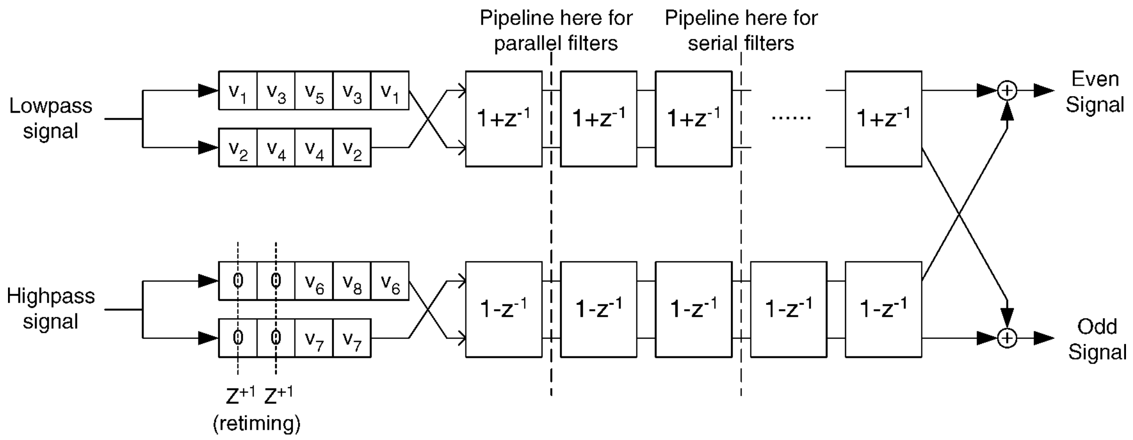


Fig. 9. Proposed B-spline-based architecture for IDWT with the (10,18) filter.

TABLE I
COMPARISONS FOR IDWT ARCHITECTURES OF THE (10,18) FILTER

Architecture	Multiplier	Adder	Critical Path	Register	Timing (ns)	Comb. Gate Count	Non-comb Gate Count
Convolution Arc. I	26	26	Tm+4Ta	14	12.5	50986.1	2182.0
Convolution Arc. II	14	26	Tm+4Ta	24	12.2	32776.6	2892.0
Bspline-serial + retiming	8	40	Tm+11Ta	24	19.5	19489.1	2630.7
Bspline-serial + retiming + pipe.	8	40	~(Tm+11Ta)/2	28	12.3	21961.0	3112.0
Bspline-parallel + retiming	8	40	Tm+13Ta	20	22.5	21121.6	2409.0
Bspline-parallel + retiming + pipe.	8	40	~(Tm+13Ta)/2	24	13.35	21952.7	2851.7

TABLE II
PERFORMANCE COMPARISONS OF 1-D DWT ARCHITECTURES FOR DAUBECHIES WAVELETS IN THREE CASES: GENERAL CASE, ODD LINEAR FILTERS, AND EVEN LINEAR FILTERS

Category	Convolution-based		Lifting-based		B-spline-based	
Function	Mul.	Adder	Mul.	Adder	Mul.	Adder
General Case	$F_H + F_G$	$F_H + F_G - 2$	$\left\lceil\frac{F_H+1}{2}\right\rceil + \left\lceil\frac{F_G+1}{2}\right\rceil + 3$	$\left\lceil\frac{F_H+1}{2}\right\rceil + \left\lceil\frac{F_G+1}{2}\right\rceil + 1$	$\frac{F_H + F_G}{2}$	$\frac{3}{2}(F_H + F_G) - 2$
Odd Linear	$\frac{F_H + F_G}{2} + 1$		$\frac{F_H + F_G}{4} + 1$		$\frac{F_H + F_G}{4} + 1$ or $\frac{F_H + F_G}{4} + 2$	
Even Linear	$\frac{F_H + F_G}{2}$		$\left\lceil\frac{F_H+1}{2}\right\rceil + \left\lceil\frac{F_G+1}{2}\right\rceil + 3$			

implemented instead, but the number of registers will become $(F_H + F_G)$.

The conventional lifting-based architectures implement the lifting factorization directly and would introduce a long critical path $F/2(T_m + 2T_a)$ with $F/2$ registers because there are about $F/2$ lifting stages. The flipping structure is proposed to reduce the critical path to $T_m + (F/2)T_a$ without any additional register [4].

The B-spline-based architectures require $(F_H + F_G)$ registers of which $(F_Q + F_R)$ of them are in the distributed part and $(\gamma_H + \gamma_G)$ of them are in the B-spline part. If the distributed part is implemented with parallel filters, the critical path will be $T_m + ((F/2) + \log_2 F)T_a$ of which $(F/2)T_a$ comes from the B-spline part, and the other term comes from the distributed part. The critical path can be reduced to $T_m + (F/2)T_a$ if serial filters are implemented instead. The approximation is listed in Table III. The flipping structure and the convolution-based architectures with serial filters are the most efficient in terms of the register number and the critical path, respectively.

In summary, the B-spline-based architecture can provide the smallest area, and the lifting-based one can use the fewest registers with a proper critical path if the flipping structure is used. The above analysis shows the tradeoff among hardware complexity, critical path, and the number of registers. The importance of the register number will be discussed in the following section.

III. TWO-DIMENSIONAL DWT ARCHITECTURES

When the 1-D DWT and IDWT are both implemented into two-input two-output systems, the methods extending them into 2-D DWT and IDWT are the same. The following analysis of 2-D DWT architectures is also applicable to 2-D IDWT. A 2-D DWT architecture mainly consists of two parts: One is the frame memory scan method, and the other is the implementation of the internal buffer. The former would consume the most power [8] and occupy a lot of system memory bandwidth. The latter would enlarge the chip size and also consume a lot of power.

TABLE III
APPROXIMATION OF THE REGISTER NUMBER AND THE CRITICAL PATH FOR 1-D DWT ARCHITECTURES WITHOUT PIPELINING

Category	Convolution-based		Lifting-based		B-spline-based	
Implementation method	Parallel Filter	Serial Filter	Conventional	Flipping	Parallel Filter	Serial Filter
Register Number	F	F_H+F_G	F/2	F/2	F_H+F_G	F_H+F_G
Critical Path	$T_m+(\log_2 F)T_a$	T_m+2T_a	$F/2(T_m+2T_a)$	$T_m+(F/2)T_a$	$T_m+(F/2+\log_2 F)T_a$	$T_m+(F/2)T_a$

When extending the 1-D DWT modules to the 2-D line-based architectures, the registers will become the internal line buffer, which is called the temporal buffer in [10]. Besides, if the image pixels are input in the raster scan, one additional internal buffer, called the data buffer, will be required as the transpose memory between row-wise and column-wise DWT modules because the 1-D modules are usually two-input two-output per cycle for 100% hardware utilization. The implementation of the temporal buffer is dependent on the adopted 1-D modules, which will be discussed in Section III-C. The data buffer only corresponds to the raster scan and can be minimized to only one line [10]. Furthermore, if the image pixels are input in Z-scan fashion [13], the data buffer can be eliminated. Thus, the data buffer will be excluded in the following discussion.

In [13], an optimal Z-scan is proposed for the JPEG 2000 system to minimize the total size of the internal buffer, including the temporal buffer of the DWT and the word-to-bitplane buffer of the Embedded Block Coding (EBC). However, if EBC is performed in the parallel bitplane context mode, the word-to-bitplane buffer can be discarded [23]. As a result, the temporal buffer of the DWT is a very important factor for a JPEG 2000 system. In the following, the frame memory scan methods and the implementation methods of the internal buffer are discussed.

A. Previous Frame Memory Scan Methods

In this section, we will focus on the one-level 2-D architectures that perform one-level 2-D DWT [10]. There are two ways to extend them to multilevel decompositions. The first one is recursively performing one-level DWT on the LL subband, which increases the frame memory access by the factor $1+(1/4)+\ldots+(1/4)^J = 4/3(1-(1/4)^J)$ [24] if J-level decompositions are required. However, the internal buffer size is unchanged. The other one is to extend the one-level architecture to the multilevel one that performs all levels of DWT decomposition at a time, which will be mentioned for each scan method except the first one.

1) Direct Scan: The direct scan is the straightforward implementation of 2-D DWT and uses the frame memory to store the intermediate DWT coefficients. The row-wise DWT is performed first, and then, the column-wise DWT is performed. Thus, the frame memory reads and writes are both $2N^2$ words for the one-level DWT, where N is the image width and height.

2) Line-Based Scan: The line-based scan method uses some internal line buffer to store the intermediate DWT coefficients [9], [10]. The scan order is the raster scan. The size of the temporal line buffer is LN, where L is the number of registers in the adopted 1-D DWT module. For example, L is four and

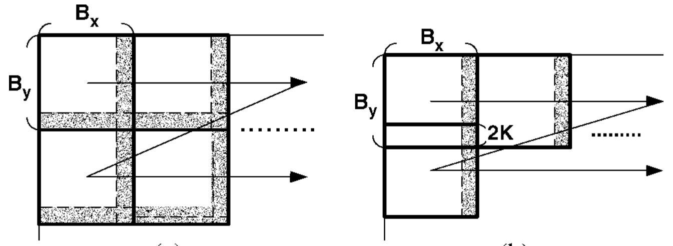


Fig. 10. Block-based scan methods. (a) Nonoverlapped block-based scan. (b) Overlapped block-based scan.

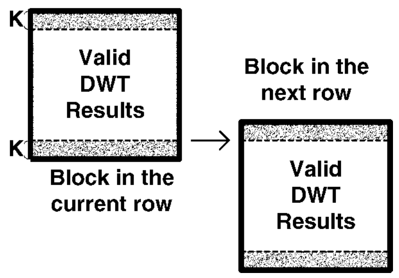


Fig. 11. Details of the overlapped blocks.

seven for the lifting-based and convolution-based (9,7) filters without pipelining, respectively [4]. The frame memory reads and writes are both N^2 words for one-level DWT. For the multilevel architecture, the size of line buffer is increased by the factor $\alpha = 1+(1/2)+\ldots+(1/2)^J = 2(1-(1/2)^J)$.

3) Nonoverlapped and Overlapped Block-Based Scan: Block-based methods scan the frame memory block-by-block, and the DWT coefficients are also computed block-by-block. If the blocks are not overlapped with each other, the frame memory reads and writes are both N^2 words for one-level DWT [11]. For each block, some intermediate data need to be stored for two neighboring blocks, as shown in Fig. 10(a), where the grey area represents the intermediate data. The blocks are assumed to be scanned in the row direction first. Then, the size of the internal buffer is $LN+LB_y$. For the multilevel architecture, the size of line buffer is also increased by the factor α.

On the other hand, the buffer LN can be eliminated if the column-wise intermediate data are not stored. Instead, we can retransmit the required data at the block boundary from the frame memory. The block-based scheme in [12] is generalized to the overlapped block-based scan method, as shown in Fig. 10(b), where $K = \lceil (F-1)/2 \rceil$. That is, the blocks are overlapped $2K$ pixels in the column direction. The overlapped area is described in Fig. 11 in more detail. The DWT coefficients of the first K and last K columns are not valid. Thus, the

TABLE IV
COMPARISONS OF SCAN METHODS FOR ONE-LEVEL 2-D DWT

	Direct	Line-based	Non-overlapped Block-based	Overlapped Block-based	Non-overlapped Stripe-based	Proposed Overlapped Stripe-based
Frame Memory Read (words/image)	$2N^2$	N^2	N^2	$N^2B_y/(B_y-2K)$	N^2	$N^2S/(S-2K)$
Frame Memory Write (words/image)	$2N^2$	N^2	N^2	N^2	N^2	N^2
Internal Buffer Size (words)	0	LN	$L(N+B_y)$	LB_y	L(N+S)	LS
Control Complexity	Low	Medium	High	High	Medium	Medium

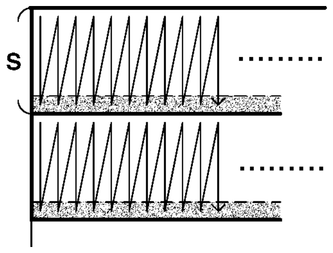


Fig. 12. Nonoverlapped stripe-based scan method.

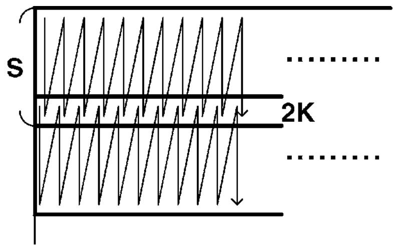


Fig. 13. Proposed overlapped stripe-based scan method.

overlapped pixels are $2K$ for deriving all DWT coefficients. The retransmission scheme increases the frame memory reads to $N^2(B_y/(B_y - 2K))$ words, whereas the frame memory writes are still N^2 words. The internal buffer size LB_y can be reduced by shrinking the block size, but it would also increase the frame memory read bandwidth. As for the multilevel architecture, the overlapped area will become 2^JK, which increases exponentially as J, and the frame memory reads become $N^2(B_y/(B_y - 2^JK))$ words. Thus, the overlapped block-based scan is not feasible for multilevel architectures.

4) Nonoverlapped Stripe-Based Scan: The optimal Z-scan method is proposed in [13], which is equivalent to performing the line-based scan in the wide block $(B_x = N)$. As a concept, the wide blocks can be viewed as stripes. Therefore, this kind of method is categorized as the nonoverlapped stripe-based scan, as shown in Fig. 12. The internal buffer size is $LN + LS$, where S is the width of the stripe. The first term is for the intermediate buffer between stripes, and the second term is for the line buffer inside stripes. For the multilevel architecture, the size of line buffer is also increased by the factor α.

B. Proposed Overlapped Stripe-Based Scan Method

The overlapped stripe-based scan method is proposed as shown in Fig. 13. All parameters are the same as that of the overlapped block-based scan method, except the stripe width S is used instead of B_y. This scan method can avoid the complex control circuits for block-based DWT architectures. It can be implemented by use of a line-based 2-D DWT architecture with the width S and an external memory address generator that provides the address of the scan patterns. One simple control circuit is also required to inform the line-based architecture about the first and last lines of the stripe for boundary extension.

1) Comparison: The comparisons of the six scan methods are listed in Table IV. The tradeoff between the external memory access and the internal buffer size is presented. The direct scan requires no internal buffer but suffers nearly double external memory bandwidth than other scan methods. The line-based scan method minimizes the external memory access but requires larger internal buffer size. According to this table, the two nonoverlapped scan methods are worse than the line-based scan method due to the larger internal buffer and the higher control complexity. Between the two overlapped scan methods, the proposed stripe-based one may be preferred for its simplicity. Especially, the proposed scan method is degenerated to the line-based scan when $S = N$. As a result, the proposed overlapped stripe-based scan method can provide the best tradeoff among external memory access, internal buffer size, and the control complexity.

2) Case Study: In the following, the HDTV image quality (1080 p 24 frames/s YUV 4:2:0) is considered as the implementation specification for a five-level 2-D DWT with the (9,7) filter. The column DWT module is assumed to be flipping-based, and $L = 4$ for the internal buffer. The image pixel rate can be calculated as follows:

$$1920 \times 1080 \times 24 \times 1.5 \simeq 74.65 \times 10^6 \quad \text{(words/s)}.$$

If the one-level architecture is used to recursively perform five-level decomposition, the memory reads and writes need be timed by $1 + (1/4) + (1/16) + (1/64) + (1/256) = 1.332$. On the other hand, the size of the internal buffer would be increased by $\alpha = 1.9375$ if the multilevel architecture is used.

The comparisons of different scan methods are shown in Table V. Since the two nonoverlapped scan methods are worse than the line-based one, we only show the line-based one in this table. The direct scan and two overlapped scan methods are considered to perform one-level DWT recursively because the extension to multilevel architectures is infeasible. Besides, the DWT can be performed on the whole image or only on smaller tiles separately in JPEG 2000 systems. Two cases are considered here, in which no tiles and the tile size 256 are

TABLE V
COMPARISONS OF 2-D DWT ARCHITECTURES WITH FIVE-LEVEL DECOMPOSITION FOR HDTV IMAGES ($K = 4$ FOR THE (9,7) FILTER)

		Direct	Line-based		Overlapped Block/Stripe-based		
			Recursive 1-level	Multi-level	$B_V=S=16$	$B_V=S=64$	$B_V=S=256$
Frame Memory Read (Mwords/sec)		198.9	99.4	74.6	198.9	113.7	102.6
Frame Memory Write (Mwords/sec)		198.9	99.4	74.6	99.4	99.4	99.4
Internal Buffer Size (words)	No Tile	0	4320	8370	64	256	1024
	Tile Size=256		1024	1984			-

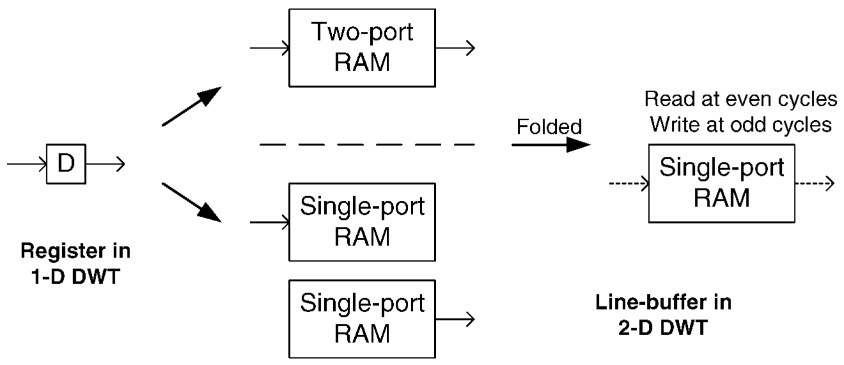


Fig. 14. Three methods to extend the registers of 1-D DWT modules into the internal buffer of 2-D DWT, which use one two-port RAM, two single-port RAM, and one single-port RAM, respectively.

assumed, respectively. In this table, the tradeoff between the frame memory access and the internal buffer size is shown clearly. The overlapped scan methods would be preferred in the case of no tiles because the frame memory access can be decreased very much with few overheads of the internal buffer. However, the line-based scan would be better in the case of the tile size 256 because the overheads of the overlapped scan methods are quite large.

C. Internal Buffer Implementation Methods

In the following, we discuss the implementation methods for the internal buffer. The resulting performance is related to not only the required memory bits but to the memory structures as well, such as two-port or single-port memories. However, only the required memory size is discussed in the literature. Based on Table III, the B-spline-based DWT module is not suitable to be extended to line-based 2-D DWT because of the large number of registers, so it will be excluded in the discussion.

1) Lifting-Based DWT Module: The registers in the lifting-based 1-D DWT module can be constructed to the internal buffer, as shown in Fig. 14. One method is to use a two-port memory to replace the registers because one-read one-write per cycle is required. The other method is to use two single-port memories and exchange the roles of reads and writes for each different line in a ping-pong fashion. The required memory bits for the second method are double that of the first one. However, two-port memories are always larger and more power-consuming than single-port memories due to the memory cell design technologies.

Besides the above methods, we can slow down the DWT module by two to change the memory access to only one-read or one-write in one cycle. Thus, one single-port memory is sufficient, but the throughput of this method is halved. All registers can be merged into the same corresponding address for higher density. Thus, only one two-port memory, two single-port memories, and one single-port memory are required for the first, second, and folded methods, respectively.

2) Convolution-Based DWT Module: The registers in the convolution-based DWT module can be constructed as Fig. 14 as well. The number of registers is usually larger than the lifting-based architectures, so the required memory size is also larger. However, the first-in first-out (FIFO) register chain of the parallel filters can be implemented in a more efficient way. The data in the memories are not necessarily changed in every cycle. Instead, only two data need to be updated in every cycle. Thus, we can use F separate single-port memories for this rotation-like read-and-write method. In every cycle, two of them are written, and others are read.

As for the folded architecture, it can be implemented as the lifting-based module to a unified single-port memory. It can also be constructed by (F-2) separate single-port memories. Two of them are written in every other cycle, and all of them are read in every other cycle.

3) Experiments: To examine the real-life implementation, we use the Artisan TSMC 0.25-μm Process High-Density Single-Port SRAM Generator and the High-Speed Two-Port Register File Generator to generate the required single-port and two-port memories. In these experiments, the (9,7) DWT filter is adopted, and the width of line buffers is set as 64 or 128. The

TABLE VI
COMPARISONS OF REAL-LIFE IMPLEMENTATION FOR THE INTERNAL LINE BUFFER AT 50 MHz

	N=64			N=128		
DWT module	Flipping	Flipping	Conv.	Flipping	Flipping	Conv.
Memory Type	Two-port	Single-port	Single-port	Two-port	Single-port	Single-port
Memory Configuration (bit)	64x64	2 64x64	9 64x16	128x64	2 128x64	9 128x16
Total Area (mm^2)	**0.2500**	0.2515	0.3457	0.3988	**0.3535**	0.495
Power (mW)	**39.71**	45.02	65.25	52.89	**45.89**	67.23

TABLE VII
COMPARISONS OF REAL-LIFE IMPLEMENTATION FOR THE INTERNAL LINE BUFFER AT 100 MHz

	N=64			N=128		
DWT module	Flipping	Flipping	Conv.	Flipping	Flipping	Conv.
Memory Type	Two-port	Single-port	Single-port	Two-port	Single-port	Single-port
Memory Configuration (bit)	64x112	2 64x112	9 64x16	128x112	2 128x112	9 128x16
Total Area (mm^2)	0.3786	0.4261	**0.3457**	0.6038	0.5969	**0.495**
Power (mW)	**62.73**	162.0	130.5	**82.90**	165.18	134.46

TABLE VIII
FOLDED METHODS OF THE INTERNAL BUFFER IMPLEMENTATION AT 100 MHz FOR EQUIVALENT THROUGHPUT AT 50 MHz

	N=64		N=128	
DWT module	Flipping/Conv.	Conv.	Flipping/Conv.	Conv.
Memory Type	Single-port	Single-port	Single-port	Single-port
Memory Configuration (bit)	64x112	7 64x16	128x112	7 128x16
Total Area (mm^2)	**0.2130**	0.2689	**0.2985**	0.3850
Power (mW)	81.16	65.25	82.59	67.19

internal wordlength is set as 16-bit. We use the flipping structure to implement lifting-based architectures to save the internal buffer [4]. The flipping structures can be synthesized by the Synopsys Design Compiler to about 70 and 130 MHz by using four and seven registers, respectively, with the Artisan 0.25-μm cell library, while the conventional lifting-based architectures need four and ten registers for 30 and 100 MHz, respectively. The convolution-based architectures are implemented by use of parallel filters, and 93 MHz can be achieved with seven registers.

The results of all implementation methods are listed in Tables VI and VII for working frequencies 50 and 100 MHz, respectively. The best performance in terms of area and power is dependent on the image width and working frequency. This illustrates that the performance is highly related to the memory structures and design technology. The smallest number of the required memory bits cannot guarantee the best performance.

The throughput of folded methods at 100 MHz is equivalent to that of nonfolded ones at 50 MHz. The results of folded methods are shown in Table VIII. Compared with Table VI, the folded method in Section II-B can provide the smallest area in these cases.

IV. FLEXIBLE AND EFFICIENT ONE-LEVEL 2-D DWT ARCHITECTURE

In this section, we propose a flexible and efficient one-level 2-D DWT architecture, as shown in Fig. 15. The B-spline-based architecture is adopted as the row 1-D DWT module because of its smallest area. The column DWT module is proposed to be constructed by use of the flipping structure or the convolution-based architecture. Which one should be chosen will depend on the specification and the implementation results, as described in Section III-C.

The frame memory is scanned by use of the overlapped stripe-based scan method. It can provide the flexibility to perform 2-D DWT on images of any size if the address generator is designed

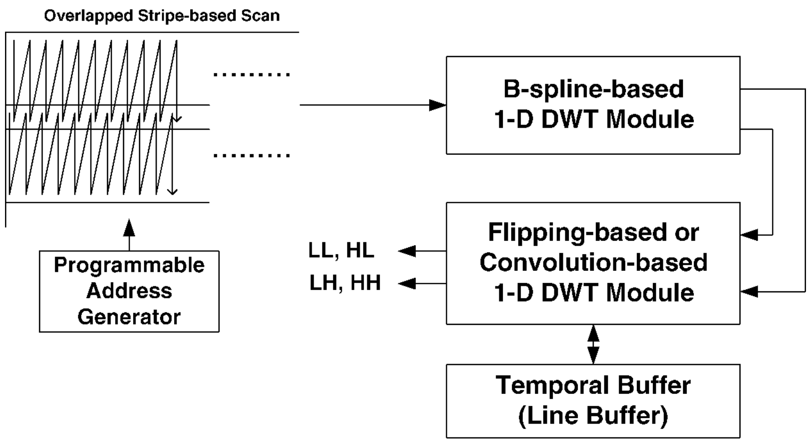

Overlapped Stripe-based Scan
B-spline-based 1-D DWT Module
Flipping-based or Convolution-based 1-D DWT Module
Programmable Address Generator
LL, HL
LH, HH
Temporal Buffer (Line Buffer)

Fig. 15. Proposed one-level 2-D DWT architecture.

as programmable or parameterizable. The overheads of the external memory bandwidth and the internal buffer size are tradeoffs but can be very small. For example, if the 2-D DWT with the (9,7) filter is to be implemented and the column DWT module adopts the flipping structure $(L = 4)$, the external memory bandwidth will be 7.14% more than the minimum, and the required memory size will be 256 words for the stripe size $S = 64$. If the stripe size is 128, the external memory bandwidth will be only 3.33% more than the minimum, and the required memory size will be 512 words.

V. Conclusion

This paper provides a detailed and practical analysis of the VLSI architectures for 1-D and 2-D DWT/IDWT. The 1-D DWT and IDWT architectures are categorized into convolution-based, lifting-based, and B-spline-based. The B-spline-based one can provide the smallest hardware cost, whereas the lifting-based one can use the fewest registers with a proper critical path if the flipping structure is used. Many frame memory scan methods for 2-D DWT are analyzed in terms of the tradeoffs between the external memory bandwidth and the internal buffer size. The implementation methods of the internal buffer are also discussed and simulated. According to experimental results, the resulting performance is related to not only the required memory size but also to the memory technology.

Besides the analysis, three architectures are also proposed. The first one is the B-spline-based architecture for IDWT. The second one is the overlapped stripe-based scan method that can provide an efficient tradeoff for the 2-D DWT. At last, we propose a flexible and efficient architecture for one-level 2-D DWT by involving many advantages of the presented analysis.

References

[1] D. Taubman, "Successive refinement of video: fundamental issues, past efforts and new directions," in *Proc. Int. Symp. Visual Commun. Image Process.*, 2003.

[2] C. Chakrabarti, M. Vishwanath, and R. M. Owens, "Architectures for wavelet transforms: a survey," *J. VLSI Signal Process.*, vol. 14, pp. 171–192, 1996.

[3] K. Andra, C. Chakrabarti, and T. Acharya, "A VLSI architecture for lifting-based forward and inverse wavelet transform," *IEEE Trans. Signal Process.*, vol. 50, no. 4, pp. 966–977, Apr. 2002.

[4] C.-T. Huang, P.-C. Tseng, and L.-G. Chen, "Flipping structure: an efficient VLSI architecture for lifting-based discrete wavelet transform," *IEEE Trans. Signal Process.*, vol. 52, no. 4, pp. 1080–1089, Apr. 2004.

[5] ——, "VLSI architecture for discrete wavelet transform based on B-spline factorization," in *Proc. IEEE Workshop Signal Process. Syst.*, 2003, pp. 346–350.

[6] W. Sweldens, "The lifting scheme: a custom-design construction of biorthogonal wavelets," *Appl. Comput. Harmon. Anal.*, vol. 3, no. 15, pp. 186–200, 1996.

[7] I. Daubechies and W. Sweldens, "Factoring wavelet transforms into lifting steps," *J. Fourier Anal. Appl.*, vol. 4, pp. 247–269, 1998.

[8] 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 Syst. Video Technol.*, vol. 11, no. 12, pp. 1246–1262, Dec. 2001.

[9] C. Chrysafis and A. Ortega, "Line-based, reduced memory, wavelet image compression," *IEEE Trans. Image Process.*, vol. 9, no. 3, pp. 378–389, Mar. 2000.

[10] P.-C. Tseng, C.-T. Huang, and L.-G. Chen, "Generic RAM-based architecture for two-dimensional discrete wavelet transform with line-based method," in *Proc. Asia-Pacific Conf. Circuits Syst.*, 2002, pp. 363–366.

[11] W. Jiang and A. Ortega, "Lifting factorization-based discrete wavelet transform architecture design," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 5, pp. 651–657, May 2001.

[12] H. Yamauchi *et al.*, "Image processor capable of block-noise-free JPEG2000 compression with 30 frames/s for digital camera applications," in *Proc. IEEE Int. Solid-State Circuits Conf.*, 2003, pp. 46–47.

[13] M.-Y. Chiu, K.-B. Lee, and C.-W. Jen, "Optimal data transfer and buffering schemes for JPEG 2000 encoder," in *Proc. IEEE Workshop Signal Process. Syst.*, 2003, pp. 177–182.

[14] S. G. Mallat, "A theory for multiresolution signal decomposition: the wavelet representation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 11, no. 7, pp. 674–693, Jul. 1989.

[15] K. K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*. New York: Wiley, 1999.

[16] P. P. Vaidyanathan, *Multirate Systems and Filterbanks*. Englewood Cliffs, NJ: Prentice-Hall, 1993.

[17] M. Unser and T. Blu, "Wavelet theory demystified," *IEEE Trans. Signal Process.*, vol. 51, no. 2, pp. 470–483, Feb. 2003.

[18] C.-T. Huang, P.-C. Tseng, and L.-G. Chen, "VLSI architecture for forward discrete wavelet transform based on B-spline factorization," *J. VLSI Signal Process.*, Apr. 2005, to be published.

[19] M. J. Tsai, J. D. Villasenor, and F. Chen, "Stack-run image coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 5, pp. 519–521, Oct. 1996.

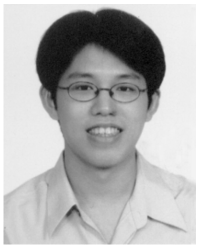
[20] N. Polyak and W. A. Pearlman, "A new flexible bi-orthogonal filter design for multiresolution filterbanks with application to image compression," *IEEE Trans. Signal Process.*, vol. 48, no. 8, pp. 2279–2288, Aug. 2000.

[21] S. Mallat, *A Wavelet Tour of Signal Processing*. New York: Academic, 1998.

[22] T. Q. Nguyen and P. P. Vaidyanathan, “Two-channel perfect-reconstruction FIR QMF structures which yield linear-phase analysis and synthesis filters,” *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 37, no. 5, pp. 676–690, May 1989.

[23] H.-C. Fang, C.-T. Huang, Y.-W. Chang, T.-C. Wang, P.-C. Tseng, C.-J. Lian, and L.-G. Chen, “81 M samples/s JPEG2000 single-chip encoder with rate-distortion optimization,” in *Proc. IEEE Int. Solid-State Circuits Conf.*, 2004, pp. 328–329.

[24] P.-C. Wu and L.-G. Chen, “An efficient architecture for two-dimensional discrete wavelet transform,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 4, pp. 536–545, Apr. 2001.



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, in 2001. He is currently 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 the one-, two-, and three-dimensional discrete wavelet transform.

Po-Chih Tseng was born in Tao-Yuan, Taiwan, R.O.C., in 1977. He received the B.S. degree in electrical and control engineering from National Chiao Tung University, Hsinchu, Taiwan, in 1999 and the M.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, 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.

Liang-Gee Chen (S’84–M’86–SM’94–F’01) 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. From 1993 to 1994, he was a Visiting Consultant with the DSP Research Department, AT&T Bell Labs, Murray Hill, NJ. In 1997, he was a Visiting Scholar with the Department of Electrical Engineering, University of Washington, Seattle. Currently, he is a Professor at National Taiwan University. 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 Seventh 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. From 2001 to 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. From 1991 to 1999, he annually 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.