Open access • Journal Article • DOI:10.1109/TSP.2004.823509

Flipping structure: an efficient VLSI architecture for lifting-based discrete wavelet transform — Source link

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

Institutions: National Taiwan University

Published on: 01 Apr 2004 - IEEE Transactions on Signal Processing (IEEE)

Topics: Lifting scheme, Second-generation wavelet transform, Discrete wavelet transform, Wavelet packet decomposition and Stationary wavelet transform

Flipping Structure: An Efficient VLSI Architecture for Lifting-Based Discrete Wavelet Transform

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

***Abstract*—In this paper, an efficient very large scale intergration (VLSI) architecture, called flipping structure, is proposed for the lifting-based discrete wavelet transform. It can provide a variety of hardware implementations to improve and possibly minimize the critical path as well as the memory requirement of the lifting-based discrete wavelet transform by flipping conventional lifting structures. The precision issues are also analyzed. By case studies of the JPEG2000 default lossy (9,7) filter, an integer (9,7) filter, and the (6,10) filter, the efficiency of the proposed flipping structure is demonstrated.**

***Index Terms*—Discrete wavelet transform, flipping structure, JPEG2000, lifting-based, VLSI architecture.**

I. Introduction

SINCE the discrete wavelet transform (DWT) was deduced by Mallat [1], many researches on wavelet-based signal analysis and compression have derived fruitful results due to the well time-frequency decomposition. The DWT has been adopted as the transform coder in emerging image coding standards, such as JPEG2000 still image coding and MPEG-4 still texture coding. However, more arithmetic operations may be required for the DWT than the discrete cosine transform (DCT) because of the filter computation. Contrary to the block-based DCT, the DWT is basically frame-based. The huge amount of the memory size and access bandwidth becomes a bottleneck of the implementation for two-dimensional (2-D) DWT [2].

For one-dimensional (1-D) DWT, several convolution-based architectures have been proposed [3]–[6] because the DWT computation is intrinsically the filter convolution. After the appearance of the lifting scheme [7] and a factorization method of lifting steps [8], the lifting scheme has been widely used to reduce the computation of DWT and the control complexity of boundary extension. Some lifting-based architectures have been proposed [9]–[13], which are all based on the direct implementation of the factorized lifting scheme. On the other hand, several architectures have been proposed to handle 2-D DWT, including the direct method, semisystolic routing, systolic routing, systolic-parallel, parallel-parallel, single input multiple data (SIMD), and one-level 2-D architectures [4]–[6], [14], [15]. However, according to the evaluation in [2], RAM-based 2-D DWT architectures are preferred due to their feasibility and regularity. Moreover, the line-based method has been proposed to shrink the internal memory requirement from a frame size to a few line buffers with proper memory management [16]. Nonetheless, the tradeoff between the critical path and the size of line buffers has not been discussed clearly in the literature. Using the lifting scheme to construct VLSI architectures for DWT could outperform using convolution in many aspects, but the critical path of lifting-based architectures is potentially longer than that of convolution-based ones. Although pipelining can reduce the critical path, it will prolong the latency and require more registers for the 1-D architecture such that larger memory size is required for the 2-D line-based architecture.

This paper is organized as follows. In Section II, the conventional architectures and problems for the convolution-based, lifting-based, and 2-D DWT are introduced. In Section III, the proposed flipping structure is presented, and the precision issues are analyzed. Three case studies, including the Daubechies (9,7) filter [17], an integer (9,7) filter [18], [19], and the (6,10) filter [13], are given to prove the efficiency in Section IV. Finally, Section V gives a summary to conclude this paper.

II. DWT Architecture

There have been many VLSI architectures proposed for hardware implementation of DWT. For 1-D DWT, the architectures are mainly convolution-based and lifting-based. On the other hand, the direct and line-based architectures are the most feasible implementations for 2-D DWT. In this section, we will introduce these architectures and discuss their advantages and shortcomings. For clarity in comparison, we only consider that the throughput is set as two inputs and two outputs per clock cycle.

A. Convolution-Based

The arithmetic computation of DWT can be expressed as basic filter convolution and downsampling as follows:

$$x_L(n) = \sum_{i=0}^{P-1} h(i) \cdot x(2n-i)$$
$$x_H(n) = \sum_{i=0}^{P-1} g(i) \cdot x(2n-i) \quad (1)$$

where x_L and x_H are the lowpass and highpass signals, respectively, and h and g are the lowpass and highpass filters.

There have been several efficient architectures proposed for convolution-based DWT, such as [3]–[5]. However, only the simplest parallel architecture, as shown in Fig. 1, can be used if the throughput is set as two-input/two-output per clock cycle,

Manuscript received September 23, 2002; revised May 22, 2003. 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 the National Science Council, Republic of China, under Grant 91-2215-E-002-035, and in part by MediaTek Inc. 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.

Digital Object Identifier 10.1109/TSP.2004.823509

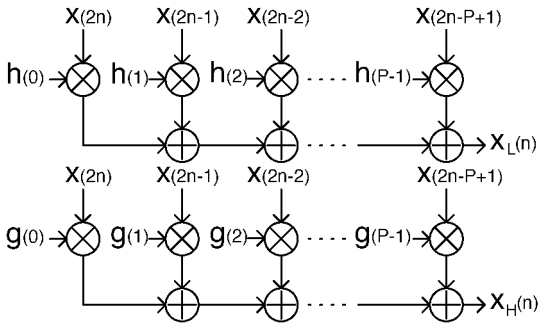

Fig. 1. Simplest convolution-based architecture.

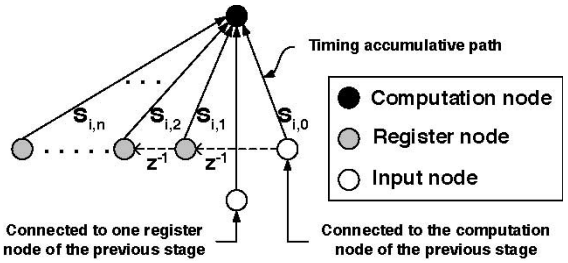

Fig. 2. Computing unit of upper triangular matrices.

as well as the minimal latency, and the fewest registers are required. In Fig. 1, the critical path is $T_m + (P-1)\cdot T_a$, where T_m is the time taken for a multiplication operation, T_a is the time needed for an addition operation, and P is the filter length. The hardware cost is $2P\cdot C_m + 2(P-1)\cdot C_a$, where C_m and C_a are the hardware cost of a multiplier and an adder, respectively. Furthermore, using adder tree to implement these additions can improve the critical path to $T_m + \lceil \log_2 P \rceil \cdot T_a$. As for the number of registers, this architecture requires $P-2$ registers without considering the input nodes $x(2n)$ and $x(2n-1)$. If the adopted DWT filter is linear, we can reduce the required number of multipliers by one half by use of the symmetric or anti-symmetric structures of linear filters. Moreover, this modification will not change the critical path and the number of registers.

B. Lifting-Based

The lifting scheme is a method for constructing wavelets by spatial approach [7]. The lifting scheme provides many advantages, such as fewer arithmetic operations, in-place implementation, and easy management of boundary extension. According to [8], 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:

$$\begin{aligned} h(z) &= h_e(z^2) + z^{-1}h_o(z^2) \\ g(z) &= g_e(z^2) + z^{-1}g_o(z^2) \end{aligned} \tag{2}$$

$$\begin{aligned} P(z) &= \begin{bmatrix} h_e(z) & g_e(z) \\ h_o(z) & g_o(z) \end{bmatrix} \\ &= \prod_{i=1}^{m} \begin{bmatrix} 1 & s_i(z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ t_i(z) & 1 \end{bmatrix} \begin{bmatrix} K & 0 \\ 0 & \frac{1}{K} \end{bmatrix} \end{aligned} \tag{3}$$

where $h(z)$ and $g(z)$ are the lowpass and highpass analysis filters, respectively, the (2) is the polyphase decomposition, and $P(z)$ is the polyphase matrix.

Because perfect reconstruction is assumed, there are some constraints on the lowpass and highpass filters. The above lifting factorization can further reduce the arithmetic operations over convolution-based architectures by exploring the redundancy between the lowpass and highpass filters. Although the arithmetic gain of the lifting scheme over convolution-based structures may possibly reach a factor of four [20], the nonuniqueness of lifting factorizations diversifies the design space of hardware implementation for lifting-based DWT. In [11], a systematic design method of efficient VLSI architectures for lifting-based DWT is proposed, which includes specific lifting factorization, dependence graph formation, and systolic array mapping. The efficiency of lifting scheme with respect to hardware cost and the complexity of control circuits has been proven. However, the potentially long critical path has not been discussed in literature.

In order to point out this crisis, (3) should be examined first. The lifting factorization is essentially composed of a series of computing stages that correspond to the upper and lower triangular matrices. The computing unit of upper triangular matrices is shown in Fig. 2, and the case of lower triangular matrices is similar. In Fig. 2, the computation node performs the summation of all input signals, the register node stores the data in the previous clock cycle, and the input node receives the coming data in the current clock cycle. Thus, a lifting-based architecture is a serial combination of such computing units, and the computation node of the previous computing stage is connected to the right input node of the next stage. The critical path of a lifting-based architecture would be the sum of the timing delay in each computing unit without pipelining. This accumulative timing effect is due to the multipliers $s_{i,0}$ and $t_{i,0}$ being on the connection paths between computing units. Although pipelining can be used to reduce the critical path, the number of additional registers will increase more rapidly as the required critical path becomes shorter. The penalty of pipelining lifting-based architectures will become very critical for the implementation of line-based 2-D DWT because the number of registers in 1-D DWT can dominate the hardware design of line-based 2-D DWT, which will be described in the next subsection.

In order to give a clearer explanation of this crisis, the popular (9,7) filter is taken as an example without loss of generality. The (9,7) filter can be decomposed into four lifting stages as follows:

$$\begin{aligned} P(z) = &\begin{bmatrix} 1 & a(1+z^{-1}) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ b(1+z) & 1 \end{bmatrix} \begin{bmatrix} 1 & c(1+z^{-1}) \\ 0 & 1 \end{bmatrix} \\ &\times \begin{bmatrix} 1 & 0 \\ d(1+z) & 1 \end{bmatrix} \begin{bmatrix} K & 0 \\ 0 & \frac{1}{K} \end{bmatrix} \end{aligned} \tag{4}$$

where the coefficients are given as $a = -1.586\,134\,342$, $b = -0.052\,980\,118$, $c = 0.882\,911\,076$, $d = 0.443\,506\,852$, and $K = 1.149\,604\,398$. The corresponding signal flow graph can be derived as Fig. 3. To reduce the number of multipliers, each computing unit should be modified as Fig. 4. The normalization step K and $1/K$ can be implemented either independently or together with the quantization if data compression is performed. Thus, we focus on the implementation issue of the lifting stages

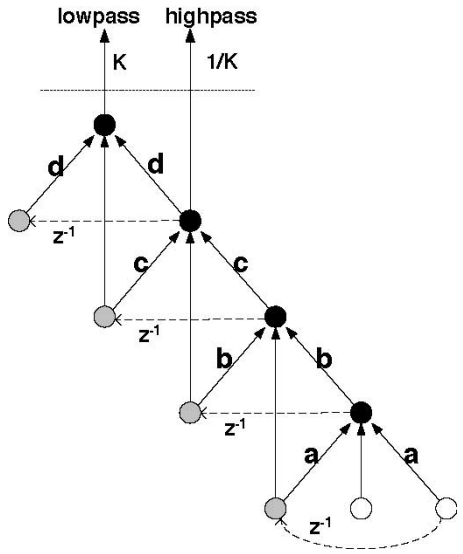

Fig. 3. SFG for lifting-based architecture of (9,7) filter.

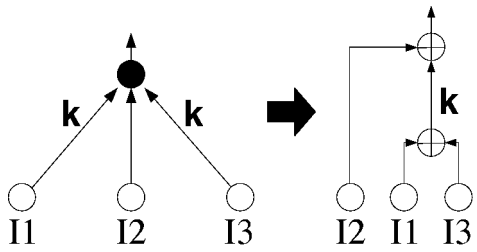

Fig. 4. Modified computing unit for Fig. 3.

in this paper. In comparison with the convolution-based architecture of the (9,7) filter that requires nine multipliers, 14 adders, and seven registers if adder tree and symmetric property are adopted, the lifting-based architecture needs only four multipliers, eight adders, and four registers (K and $1/K$ are not involved). Nonetheless, the critical path of the lifting-based architecture is $4T_m + 8T_a$, whereas the convolution-based one needs only $T_m + 4T_a$. Although pipelining the lifting-based architecture can improve the serious timing problem, this will raise the number of registers rapidly. For instance, the lifting-based architecture for the (9,7) filter can be pipelined into four stages to obtain a critical path $T_m + 2T_a$ with six additional registers, as shown in Fig. 5.

C. Two–Dimensional DWT

In contrast to systolic or semisystolic routing architectures [6], RAM-based 2-D DWT architectures have many advantages, including real-life feasibility, high regularity/density of storage, and simple control circuits. On the other hand, according to the evaluation in [2], the memory issue is the most critical part for 2-D DWT implementation, instead of the number of multipliers that dominates the 1-D DWT architectures.

In general, RAM-based architectures can be classified into three categories: direct, line-based, and block-based methods [2]. The direct one is the most straightforward implementation method. It performs 1-D DWT in one direction and stores the intermediate coefficients in a frame memory first. Then, it performs 1-D DWT in the other direction with these intermediate

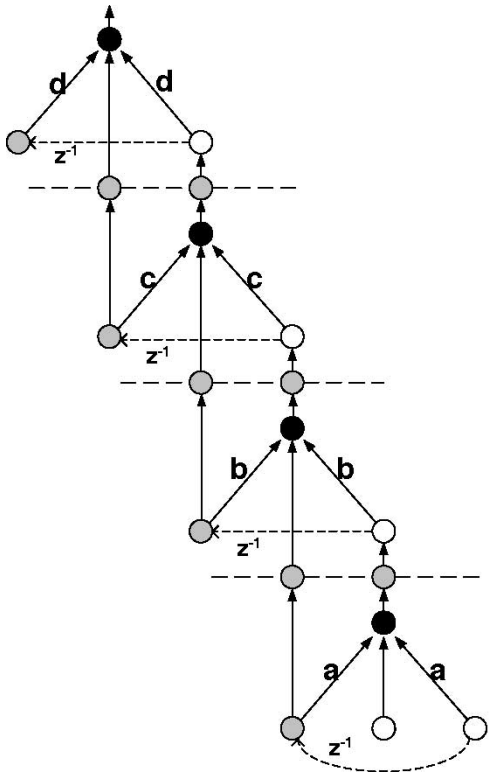

Fig. 5. Pipelining four stages for Fig. 3.

coefficients to complete one-level 2-D DWT. Because the size of this frame memory is $O(N^2)$, it is usually assumed to be off chip. However, the line-based method performs 1-D DWT in both directions simultaneously. Thus, the line-based method does not require a frame memory to store the intermediate data. Instead, some internal line buffers are used to store the intermediate data, and the required size is proportional to the image width. Contrary to the assumption that the input signals are in raster scan order for the above two methods, the block-based method performs 2-D DWT in a block-by-block way. This method can use some internal line buffers to store the boundary data among neighbor blocks such as to keep the required external frame memory bandwidth as low as the line-based method. Which method should be adopted depends on what kinds of hardware constraints are given. However, the external memory access would consume the most power [2] and become very sensitive in the case of system on chip. In addition, the required external memory bandwidth of the direct method is more than the double of the line-based and block-based methods [2].

Although line-based architectures suffer from the requirement of internal line buffers, they can effectively reduce the external memory access and shorten the latency. There have been several line-based architectures proposed for the convolution-based hardware implementation of 2-D DWT, such as systolic-parallel [5], parallel-parallel [6], and one-level 2-D [14] architectures. One configurable architecture for lifting-based 2-D DWT has also been proposed [13], in which the intermediate frame memory and the internal line buffer are included. This architecture can be configured to perform 2-D DWT for several DWT filterbanks. Furthermore, it uses the direct method for two

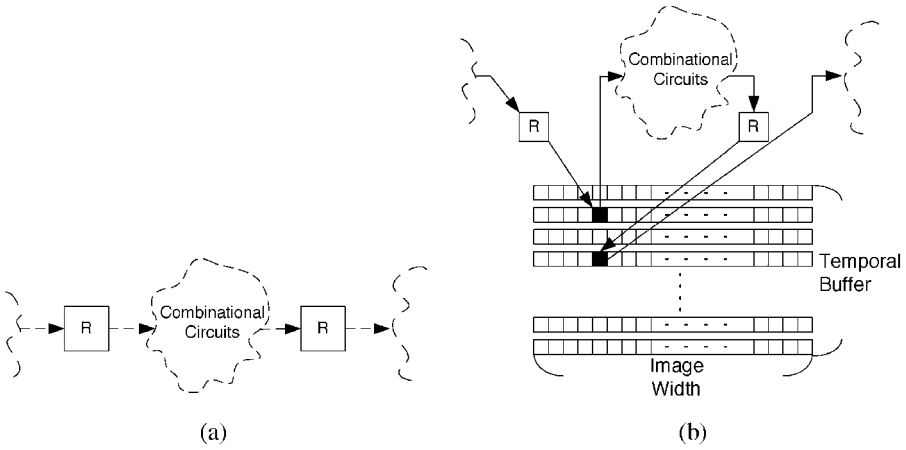

Fig. 6. (a) Original 1-D DWT architecture. (b) Corresponding temporal buffer mapping.

lifting-stage filters and the line-based method for four lifting-stage filters. However, this needs more general processing and memory units for configurable functions such that more registers and internal memory are required.

Basically, the internal memory of the line-based architectures consist of a data buffer and a temporal buffer, as described in [15]. The data buffer is related to the input nodes of the signal flow graph for 1-D DWT. Thus, its size is fixed for any kind of adopted DWT filters with the same throughput. The temporal buffer is used to store the temporal data in the column direction and can be mapped from the adopted 1-D DWT architecture, as shown in Fig. 6, where every line of the temporal buffer in Fig. 6(b) corresponds to one register in Fig. 6(a). Thus, the size of the temporal buffer is proportional to the number of registers in the adopted 1-D architecture. Since the temporal buffer is design-dependent, how to minimize the temporal buffer will be the first consideration for hardware implementation of line-based architectures under the given timing criteria.

As for the block-based architectures, the internal buffer size is also proportional to the number of registers in the adopted 1-D architecture [12]. Thus, minimizing the number of registers for the 1-D architecture is the most important issue both for the line- and block-based architectures after the timing criteria is achieved.

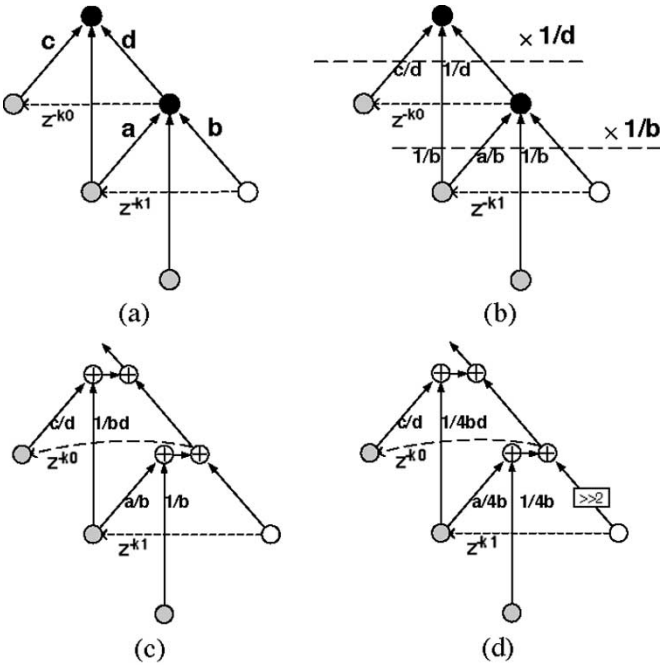

Fig. 7. (a) Two connected computing units. (b) Flipping computing units. (c) After splitting computation nodes and merging the multipliers. (d) Flipping with the inverse multiplying by 4.

III. Proposed Flipping Structure

As the mentioned above, conventional lifting-based architectures could require fewer arithmetic operations but have the longer critical paths than convolution-based ones. In this section, an efficient VLSI architecture, called flipping structure, is proposed to solve the serious timing accumulation problem. The basic idea is presented first, and then, the precision issues of multiplication coefficients are discussed.

A. Flipping Structure

Since the timing problem is due to the accumulation of timing delays from the input node to the computation node in each computing unit, we suggest releasing the accumulation by eliminating the multipliers on the path from the input node to the computation node. This can be achieved by flipping each computing unit with the inverse of the multiplier coefficient. For example, two computing units are possibly connected as Fig. 7(a), in which $s(z)$ and $t(z)$ are assumed to be two taps for simplicity. This architecture can be flipped by the inverse coefficients of b and d, as shown in Fig. 7(b). Flipping is to multiply the inverse coefficient for every edge on the feed-forward cutset, which is through the selected multiplier. Then, every computation node can be split into two adders, of which one can process in parallel with other computing units, and the other one is on the accumulative path, as illustrated in Fig. 7(c). In addition, the multiplications on the same path can be merged together to reduce the number of multipliers. In this simple example, the critical path is reduced from $2T_m + 3T_a$ to $T_m + 3T_a$, and the reduction rate will increase as the number of serially connected computing units becomes larger.

If $s(z)$ and $t(z)$ are more than two taps, the selected flipping coefficients are still from the multipliers between the input and

computation nodes. Moreover, the computation nodes can be split into two parts: One is the summation of the multiplication results from register nodes and the other one is the adder on the accumulative path. The timing accumulation can be greatly reduced by flipping the original lifting-based architectures. Another advantage of flipping structures is that no additional multipliers will be required if the computing units are all flipped. Furthermore, the flipping coefficients should be put into the normalization step to assure that correct lowpass and highpass coefficients will be obtained.

There are also many alternatives for flipping structures. How many computing units should be flipped is case-by-case and dependent on hardware constraints. This flipping method can also be applied for lifting-based inverse DWT because the basic computing unit is exactly the same as that of lifting-based forward DWT.

B. Precision Issue

While flipping the lifting-based architectures, the precision issue should be handled carefully for the inverses of lifting coefficients. Both the scaling effect and roundoff noise will be changed entirely. Although a systematic method using state variable description to solve these problems is presented in [21], only the calculation of roundoff noise can be adopted here because the method of scaling is only suitable for single input systems and possible to change the critical path.

In fact, the flipping coefficients are not necessarily to be the inverses of the lifting coefficients exactly for releasing the timing accumulation. If the arithmetic operations all belong to two's complement computation, flipping with the inverse coefficient multiplying by 2^k, where k is an integer, can also reduce the critical path, as shown in Fig. 7(d), where the flipping coefficients are $4b$ and d. This is because shifters can handle 2^k well without multipliers. Thus, the overflow problem can be solved with the following description. Because the DWT filters under consideration are all FIR, the overflow problem of multipliers can be prevented by setting all the multiplier coefficients to be smaller than one. However, if the coefficient is too small, the precision of internal signals will be seriously reduced. Thus, we suggest keeping all the multiplication coefficients as large as possible but smaller than one. The method of adjusting the coefficients is to change the flipping coefficients with multiplying by some 2^k. Instead of exhaustive adjustment, the flipping stage that is closest to input signals can be adjusted first. Then, the second closest stage can be adjusted exactly, and so on. This stage-by-stage adjustment can assure that the multiplier coefficients all large enough and smaller than one. For example, Fig. 8(a) is the proposed flipping structure for the (9,7) filter, in which all lifting stages are flipped, and the overflow problem of the multipliers is prevented with the above adjustment.

On the other hand, the data wordlength required to prevent the adders from arising the overflow problem is dependent on the wordlength of the input signals and how many decomposition levels should be supported. There are some tradeoffs between the internal wordlength and the required precision. The internal wordlength can be easily reduced by scaling down with

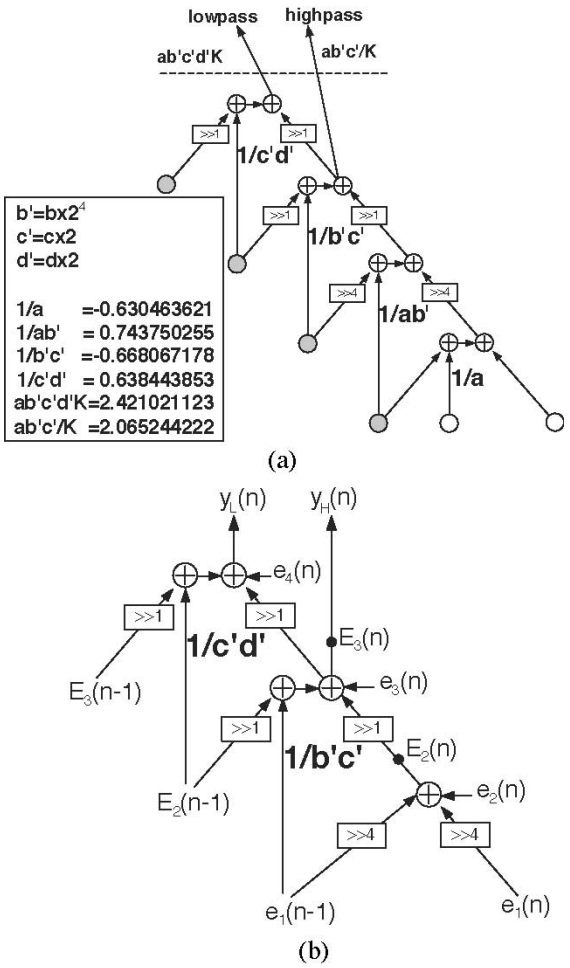

Fig. 8. (a) Flipping structure for (9,7) filter. (b) Corresponding roundoff noise model.

2^k, where k is a negative integer, for any cutset, but this will reduce the precision of the internal data. Thus, we suggest a more straightforward method to solve the overflow problem. After determining the multiplier coefficients with the above-mentioned adjustment, the possible maximum of the internal signals can be calculated easily with the given maximum of input signals. For example, if the maximum of input signals is given as S, the candidates for the maximum of internal signals are located in the outputs of the four adders on the accumulative path in Fig. 8(a). The four possible values can be calculated as $2.630\,46S$, $1.072\,56S$, $1.740\,62S$, and $2.379\,06S$ while performing the summation of the maximal inputs. Thus, the maximum of internal signals is $2.630\,46S$, and it is sufficient for the internal wordlength to be 2 bits more than the input signals.

As for the roundoff noise, the noise model of Fig. 8(a) can be established as Fig. 8(b). The quantization noise sources are assumed to be located after the four adders on the accumulative path. That is, the quantization of internal signals is only performed after these adders and produces the noise signals $e_k(n)$. Furthermore, the stored data of registers are also contaminated by the noise signals and represented by $E_k(n-1)$ or $e_1(n-1)$. This kind of roundoff noise model can be used to analyze the noise effects of each noise source and compute the variances of roundoff noises for the output signals $y_L(n)$

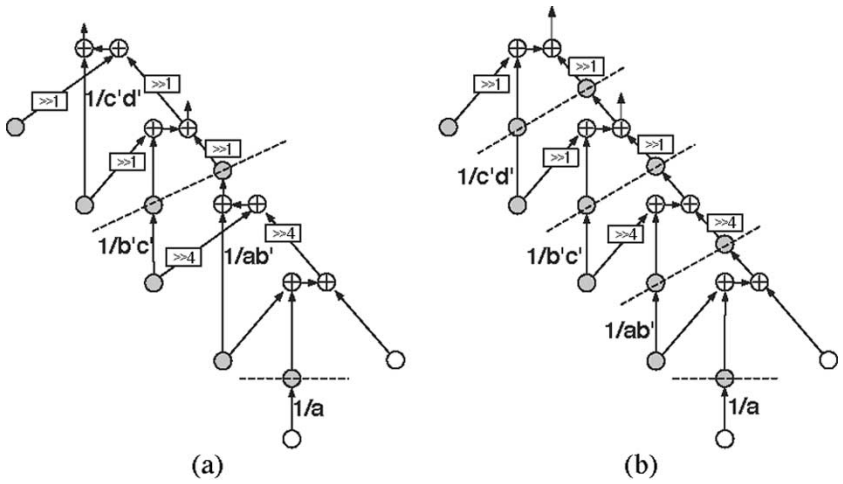

Fig. 9. (a) Cut 3 pipelining stages for Fig. 8(a). (b) Cut 5 pipelining stages.

TABLE I
COMPARISON OF SEVERAL ARCHITECTURES FOR (9,7) FILTER

Architecture	Multiplier	Adder	Critical path	Temp. buffer	Timing(ns)	Gate count
Lifting + no pipe.	4	8	4Tm+8Ta	4L	55	12448
Lifting + 4 stages	4	8	Tm+2Ta	10L	16	12886
Lifting + fully pipe.	4	8	Tm	32L	10	12116
Flipping + no pipe.	4	8	Tm+5Ta	4L	21	10102
Flipping + 3 stages	4	8	Tm+1Ta	7L	12.3	9703
Flipping + 5 stages	4	8	Tm	11L	10.1	10752
Convolution + no pipe.	9	14	Tm+4Ta	7L	19	14648
Convolution + 3 stages	9	14	Tm	23L	10	13639

L: Image width

and $y_H(n)$, which correspond to the lowpass and highpass signals, respectively. All noise signals $e_k(n)$ can be assumed zero mean, uniform distributed, and independent. The variance is dependent on how many bits are quantized [21]. Nonetheless, it is not certain whether the original lifting-based architecture or the corresponding flipping structure has less roundoff noise. This should be determined case-by-case.

IV. CASE STUDY

To show the feasibility and efficiency of the proposed flipping structure, three case studies are given in this section, including the JPEG2000 default lossy DWT (9,7) filter, an integer (9,7) filter, and the (6,10) filter.

A. JPEG2000 Default Lossy (9,7) Filter

Since the lifting structure of (9,7) filter is very regular and typical, it is meaningful to compare its conventional lifting-based architecture with the flipping structure. The lifting-based architecture, as shown in Fig. 3, needs only four multipliers, eight adders, and four registers with a critical path of $4T_m + 8T_a$. This critical path can be reduced to $T_m + 2T_a$ by cutting four pipelining stages with six additional registers, as shown in Fig. 5. Moreover, 32 registers can be used for fully pipelining to minimize the critical path to a multiplier delay.

TABLE II
COMPARISON OF CONVENTIONAL LIFTING-BASED ARCHITECTURES AND FLIPPING STRUCTURES

Architecture	Temp. buffer	Timing(ns)	Gate count
Lifting + no pipe.	4L	55	12448
Flipping + no pipe.	4L	30.5	6591
Lifting + 4 stages	10L	16	12886
Flipping + 5 stages	11L	13.9	7042

An efficient flipping structure is designed for the (9,7) filter, as shown in Fig. 8(a), by flipping all computing units. The critical path of this flipping structure is only $T_m + 5T_a$ without any additional hardware cost over Fig. 3. In fact, this flipping structure can be represented by the equation below, with respect to (4) of the lifting-based architecture.

$$P(z) = \begin{bmatrix} \frac{1}{a} & 1+z^{-1} \\ 0 & \frac{1}{a} \end{bmatrix} \begin{bmatrix} \frac{1}{16b} & 0 \\ \frac{1+z}{16} & \frac{1}{16b} \end{bmatrix} \begin{bmatrix} \frac{1}{2c} & \frac{1+z^{-1}}{2} \\ 0 & \frac{1}{2c} \end{bmatrix} \times \begin{bmatrix} \frac{1}{2d} & 0 \\ \frac{1+z}{2} & \frac{1}{2d} \end{bmatrix} \begin{bmatrix} 64abcdK & 0 \\ 0 & \frac{64abcd}{K} \end{bmatrix}. \quad (5)$$

Furthermore, if three pipelining stages are cut and some arithmetic operators are rearranged as in Fig. 9(a), the critical path will be reduced to $T_m + T_a$ with three additional registers. The critical path can also be minimized to a multiplier delay by using five pipelining stages, as shown in Fig. 9(b), which needs only 11 registers.

TABLE III
COMPARISON OF ROUNDOFF NOISE

Architecture	Signal	Lena	Baboon	Scene	Couple	Jet	Map	Pepper	Average	Theoretical Value
Conventional Lifting	Lowpass	0.0016478	0.0016527	0.0016535	0.0016678	0.0016668	0.0016480	0.0016515	0.0016554	0.0016589
	Highpass	0.0010960	0.0010992	0.0010997	0.0011234	0.0011069	0.0010950	0.0011024	0.0011032	0.0011048
Flipping Structure	Lowpass	0.0009940	0.0009920	0.0009941	0.0009883	0.0009964	0.0009860	0.0009880	0.0009912	0.0009908
	Highpass	0.0006105	0.0006066	0.0006073	0.0006100	0.0006013	0.0005978	0.0006056	0.0006056	0.0006083

To obtain more realistic results, we have verified the above architectures, including conventional lifting-based architectures, flipping structures, and convolution-based architectures, with 12-bit precision for multiplier coefficients and 16-bit wordlength for the internal data. The least significant four bits of the internal data are used to preserve the decimal precision. The method of verification is to write Verilog HDL codes first and examine the simulation results. Then, the Synopsys Design Compiler is used to synthesize the circuits with standard cells from the Avant! 0.35-μm cell library. For comparison, the timing constraints for circuit synthesis are set as tight as possible. As a result, the synthesis results for critical paths are very close to our expectation. The experimental results are summarized in Table I, where the case of the pipelined convolution-based architecture is also included.

In Table I, the required size of the temporal buffer is used, instead of the number of registers, to emphasize the critical problem in the line-based 2-D DWT architectures. The flipping structures outperform the conventional lifting-based and the convolution-based architectures in every aspect, such as timing, logic gate count, and the required size of the temporal buffer. Furthermore, in order to show the performance of flipping structures, we synthesize the flipping structures under the timing constraints that are the limits of the lifting-based architectures with nearly the same temporal buffer size. The comparison results are given in Table II, which shows that flipping structures can achieve the timing limits of lifting-based architectures with only about one half of the hardware cost. In the case of no pipelining stages, the maximum of the critical path of the flipping structure (30.5 ns) is much smaller than the minimum of the critical path of the conventional lifting-based architecture (55 ns). Furthermore, the gate counts of flipping structures in Table II are much less than those in Table I because different adder architectures can be used under different timing constraints. For example, the simplest ripple-carry adders are used under the loose timing constraint in Table II. On the contrary, carry-look-ahead adders are used to meet the tight timing constraint in Table I, which require much more logic gates.

In the following, the precision issues of Figs. 3 and 8 are discussed. The following discussion is for comparing the relative performance of roundoff noises between these two architectures under the same assumption of quantization schemes. First, these two architectures are both assumed to perform quantization only four times. For Fig. 3, quantization is performed at the outputs of the computation nodes. As illustrated in Section III-B, Fig. 8(b) is the noise model for Fig. 8(a), and it is equivalent to performing quantization only at the outputs of the adders on the accumulative path. Thus, the roundoff noises of two output signals for each architecture can be calculated by the method with state variable description and matrix computation [21].

TABLE IV
COMPARISON OF FINITE MULTIPLIER COEFFICIENT PRECISION (12-BIT)

Decomposition Level	Lifting		Flipping	
	Ave. MSE	PSNR	Ave. MSE	PSNR
1	0.59363	50.40	0.000037	92.44
2	1.87685	45.40	0.007347	69.47
3	4.11688	41.99	0.058848	60.43
4	7.07015	39.64	0.159092	56.11
5	10.8243	37.79	0.271156	53.80

For verification, the decimal precision is set to be 4-bit; therefore, the variance of all error signals σ_e^2 is $(2^{-10}/3)$. Then, the theoretical values of these two roundoff noises can be calculated with this variance. Moreover, seven images, comprising Lena, Baboon, Scene, Couple, Jet, Map, and Pepper, are used as the input signals of these two architectures to evaluate the real roundoff noises. The experimental results are listed in Table III. According to this table, the roundoff noises of the flipping structure are smaller than those of the lifting-based architecture, and the experimental values are nearly the same as the theoretical values.

For comparison in detail, the noises of the highpass signals in the two architectures $y_{H,\text{lifting}}(n)$ and $y_{H,\text{flipping}}(n)$ are expressed as the linear combination of the quantization noise signals:

$$\begin{aligned} y_{H,\text{lifting}}(n) &= e_3(n) + 0.882\,91\,(e_2(n) + e_2(n-1)) \\ &\quad - 0.046\,78\,(e_1(n) + e_1(n-2)) \\ &\quad + 0.906\,44 e_1(n-1) \\ y_{H,\text{flipping}}(n) &= e_3(n) + 0.5(e_2(n) + e_2(n-1)) \\ &\quad + 0.031\,25\,(e_1(n) + e_1(n-2)) \\ &\quad - 0.605\,57 e_1(n-1). \end{aligned} \tag{6}$$

Thus, variances of these two noises are

$$\begin{aligned} E\left[y_{H,\text{lifting}}^2(n)\right] &= 3.385\,07\sigma_e^2 \\ E\left[y_{H,\text{flipping}}^2(n)\right] &= 1.868\,67\sigma_e^2 \end{aligned} \tag{7}$$

The reason why the roundoff noise of the flipping structure is smaller is that the propagations of noise signals $e_k(n)$ are mitigated by use of the right shifters and the multipliers of coefficients smaller than one. These two kinds of operations can decrease the weights of noise signals from previous stages. Furthermore, the required wordlength of the flipping structure is also shorter than that of the lifting-based architecture. This

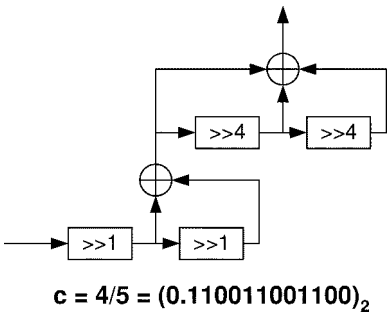

Fig. 10. Twelve-bit precision multiplication of 4/5.

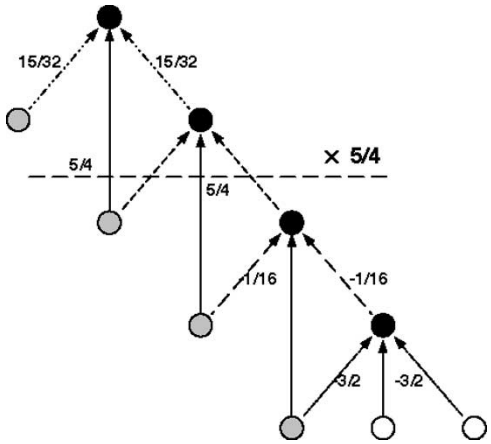

Fig. 11. Flipping structure of integer (9,7) filter.

can be evaluated by examining the maximum of possible internal signals in these two architectures. As described in Section III-B, the maximum of internal signals in the flipping structure is $2.630\,46S$, whereas that in the lifting-based architecture can be calculated as $7.401\,72S$. Thus, the internal wordlength of the lifting-based architecture is required to be one-bit longer than that of the flipping structure.

Besides the roundoff noise, the effect of finite precision multiplier coefficients is also very important for VLSI architectures. However, this effect is difficult to be analyzed independently from the input signals as the roundoff noise. Instead of precise analysis, simulation results are given to compare the finite effects between the lifting and flipping architectures. Multilevel 2-D DWT decomposition for the above-mentioned seven images is simulated on the condition that all multiplier coefficients are 12-bit precision for the two architectures. Because this finite coefficient effect is related to the input signals, the dynamic ranges of the two architectures in every decomposition level are kept the same with adjustment of the normalization coefficients that are assumed infinite precision in the simulation. For showing the influence of image quality, the multilevel DWT decomposition coefficients are reconstructed to images through the multilevel Inverse-DWT of infinite coefficient precision. The simulation results are shown in Table IV. According to Table IV, the flipping structure outperforms the conventional lifting-based architecture under the same finite precision condition.

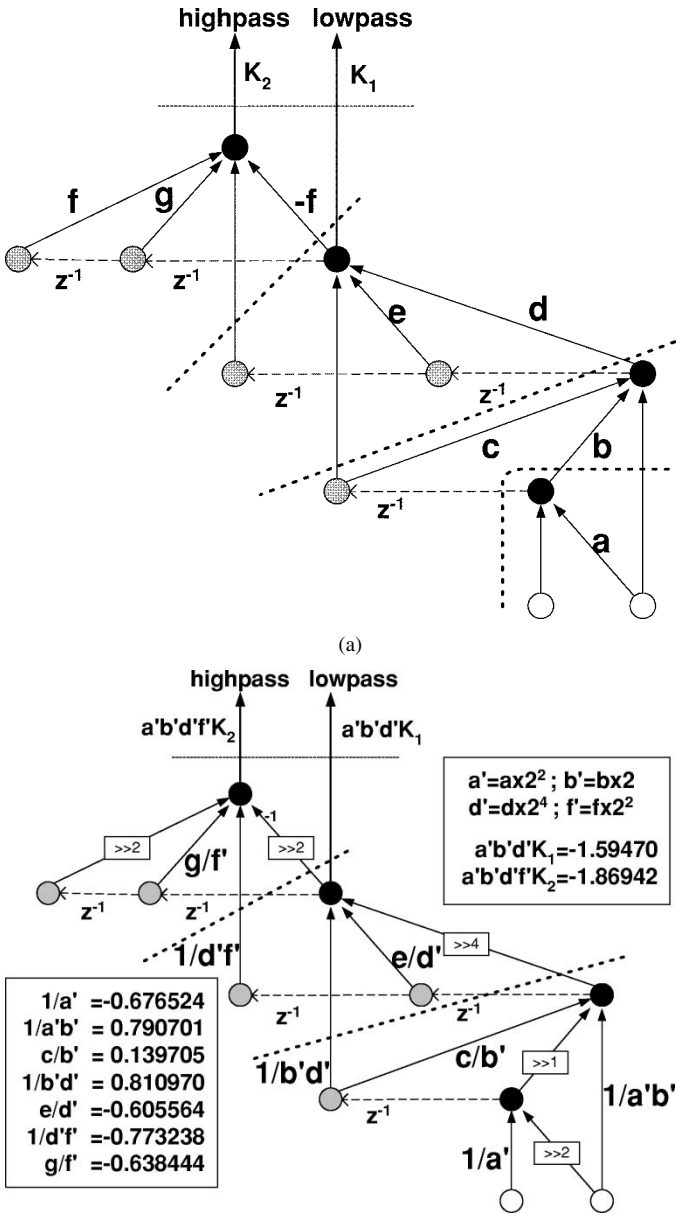

Fig. 12. (a) Lifting-based architecture for the (6,10) filter. (b) Corresponding flipping structure.

B. Integer (9,7) Filter

In addition to the advantages of implementation issues, the lifting scheme can also provide a well-constructed architecture to design wavelet-like filters. At the same time, [18] and [19] propose a class of integer DWT filters based on the lifting scheme of the (9,7) filter. Moreover, the results of these two classes both can give an excellent integer DWT filter with lifting coefficients $a = -3/2$, $b = -1/16$, $c = 4/5$, and $d = 15/32$. This integer wavelet filter can achieve nearly the same performance as the original (9,7) filter but requires much less hardware cost [19]. The multiplication of a, b, and d can be implemented with a few shifters and two adders. However, c needs a floating-point operation. If 12-bit precision of c is considered, three adders and four shifters are sufficient for the implementation of the multiplier c, as shown in Fig. 10.

TABLE V
COMPARISON OF SEVERAL ARCHITECTURES FOR (6,10) FILTER

Architecture	Multiplier	Adder	Critical path	Temp. buffer	Timing(ns)	Gate count
Lifting + no pipe.	7	8	4Tm+5Ta	5L	42	13576
Lifting + 4 stages	7	8	Tm+2Ta	11L	14.8	13741
Lifting + fully pipe.	7	8	Tm	27L	9	14466
Flipping + no pipe.	7	8	Tm+5Ta	5L	22.8	13257
Flipping + 3 stages	7	8	Tm+2Ta	9L	14.3	12966
Flipping + 5 stages	7	8	Tm	15L	9.8	11123
Convolution + no pipe.	8	14	Tm+4Ta	8L	20.5	15151
Convolution + 3 stages	8	14	Tm	24L	9.6	15411

Therefore, if the conventional lifting-based architecture is used for this integer DWT filter, 13 adders and some shifters are required, and the critical path is $13T_a$.

However, if the computing unit of c is flipped as Fig. 11, 13 adders are required for the hardware implementation with a critical path $7T_a$ after some modification of computation nodes. Furthermore, there is no floating-point operation in Fig. 11, and the precision of internal data can also be reduced. In this case study, we show that flipping structures can be used not only to reduce the critical path but also to provide other efficient design techniques for lifting-based architectures.

C. (6,10) Filter

Besides the above odd symmetric wavelet filters, the case of the even linear (6,10) filter [13] has been studied as well. The polyphase matrix of the (6,10) filter can be decomposed as follows:

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

where the coefficients are given as $a = -0.369\,536$, $b = -0.427\,80$, $c = -0.119\,532$, $d = -0.090\,075$, $e = 0.872\,739$, $g = -0.572\,909$, $f = 0.224\,338$, $K_1 = 0.874\,919$, and $K_2 = 1.142\,963$ [13]. Thus, the lifting-based architecture can be shown as Fig. 12(a), where seven multipliers, eight adders, and five registers are required if K_1 and K_2 are excluded. However, the convolution-based architecture requires eight multipliers, 14 adders, and eight registers if the symmetric and antisymmetric properties are adopted. The hardware-saving ratio of multipliers between lifting-based and convolution-based architectures is not as high as the case of (9,7) filter. Moreover, the lifting-based architecture would require more multipliers than the convolution-based one if K_1 and K_2 are considered. The main advantage of lifting scheme is the number of adders and registers in this case.

The timing accumulation of Fig. 12(a) results in a long critical path $4T_m + 5T_a$. By pipelining through the dot lines in Fig. 12(a), the critical path can be reduced to $T_m + 2T_a$ with four additional registers. Furthermore, the critical path can be minimized to T_m by fully pipelining with 22 additional registers. On the other hand, the critical path of the convolution-based architecture is only $T_m + 4T_a$ and can be reduced to T_m with three pipelining stages by use of 16 additional registers.

The proposed flipping structure can reduce the critical path of Fig. 12(a) to $T_m + 5T_a$ as shown in Fig. 12(b), in which all lifting stages are flipped. The critical path of Fig. 12(b) can be further reduced to $T_m + 2T_a$ by pipelining through the dot lines. The minimum critical path T_m can be achieved with ten pipelining registers.

The above architectures are all verified and synthesized in the same way as in Section IV-A with 16-bit wordlength for internal data and 12-bit precision for multiplier coefficients. Table V gives the experimental results and shows that the flipping structures have better performance than the lifting-based and convolution-based ones in all aspects.

V. CONCLUSION

In this paper, an efficient VLSI architecture, called flipping structure, for lifting-based DWT is proposed. The problem of serious timing accumulation for the conventional lifting-based architectures is addressed by flipping some computing units with the inverses of multiplier coefficients such that the critical path can be greatly reduced. Thus, the flipping structure can minimize the size of the internal buffer for line-based DWT architectures under specified timing constraints. Moreover, the precision issues are also discussed. By case studies of the JPEG2000 default lossy (9,7) filter and the even linear (6,10) filter, the efficiency of flipping structures over conventional lifting-based architectures is shown in all aspects, including timing, hardware cost, and line buffer size. According to the precision analysis of the (9,7) filter, the flipping structure also has better performance in the issues of roundoff noise and internal wordlength. In addition to reducing the critical path, the flipping structure can also provide well-featured architectures for lifting-based DWT filters, as illustrated in the case study of an integer (9,7) filter.

REFERENCES

[1] S. G. Mallat, "A theory for multiresolution signal decomposition: the wavelet representation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 11, pp. 674–693, July 1989.

[2] 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, pp. 1246–1262, Dec. 2001.

[3] K. K. Parhi and T. Nishitani, "VLSI architectures for discrete wavelet transforms," *IEEE Trans. Very Large Scale Integration Syst.*, vol. 1, pp. 191–202, June 1993.

[4] C. Chakrabarti and M. Vishwanath, "Efficient realizations of the discrete and continuous wavelet transforms: from single chip implementations to mappings on SIMD array computers," *IEEE Trans. Signal Processing*, vol. 43, pp. 759–771, Mar. 1995.

[5] M. Vishwanath, R. M. Owens, and M. J. Irwin, "VLSI architectures for the discrete wavelet transform," *IEEE Trans. Circuis Syst. II*, vol. 42, pp. 305–316, May 1995.
[6] C. Chakrabarti, M. Vishwanath, and R. M. Owens, "Architectures for wavelet transforms: A survey," *J. VLSI Signal Process.*, vol. 14, pp. 171–192, 1996.
[7] W. Sweldens, "The lifting scheme: a custom-design construction of biorthogonal wavelets," *Applied Comput. Harmon. Anal.*, vol. 3, no. 15, pp. 186–200, 1996.
[8] I. Daubechies and W. Sweldens, "Factoring wavelet transforms into lifting steps," *Journal Fourier Anal. Applicat.*, vol. 4, pp. 247–269, 1998.
[9] J.-M. Jou, Y.-H. Shiau, and C.-C. Liu, "Efficient VLSI architectures for the biorthogonal wavelet transform by filter bank and lifting scheme," in *Proc. IEEE Int. Symp. Circuits Syst.*, vol. 2, 2001, pp. 529–532.
[10] P.-C. Tseng, C.-T. Huang, and L.-G. Chen, "VLSI implementation of shape-adaptive discrete wavelet transform," in *Proc. SPIE Int. Conf. Visual Commun. Image Process.*, 2002, pp. 655–666.
[11] C.-T. Huang, P.-C. Tseng, and L.-G. Chen, "Efficient VLSI architectures of lifting-based discrete wavelet transform by systematic design method," in *Proc. IEEE Int. Symp. Circuits Syst.*, vol. 5, 2002, pp. 565–568.
[12] W. Jiang and A. Ortega, "Lifting factorization-based discrete wavelet transform architecture design," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, pp. 651–657, May 2001.
[13] K. Andra, C. Chakrabarti, and T. Acharya, "A VLSI architecture for lifting-based forward and inverse wavelet transform," *IEEE Trans. Signal Processing*, vol. 50, pp. 966–977, Apr. 2002.
[14] P.-C. Wu and L.-G. Chen, "An efficient architecture for two-dimensional discrete wavelet transform," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, pp. 536–545, Apr. 2001.
[15] 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.
[16] C. Chrysafis and A. Ortega, "Line-based, reduced memory, wavelet image compression," *IEEE Trans. Image Processing*, vol. 9, pp. 378–389, Mar. 2000.
[17] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform," *IEEE Trans. Image Processing*, vol. 1, pp. 205–220, Apr. 1992.
[18] D. B. H. Tay, "A class of lifting based integer wavelet transform," in *Proc. Int. Conf. Image Process.*, vol. 1, 2001, pp. 602–605.
[19] Z. Guangjun, C. Lizhi, and C. Huowang, "A simple 9/7-tap wavelet filter based on lifting scheme," in *Proc. Int. Conf. Image Processing*, vol. 2, 2001, pp. 249–252.
[20] J. Reichel, "On the arithmetic and bandwidth complexity of the lifting scheme," in *Proc. Int. Conf. Image Process.*, 2001, pp. 198–201.
[21] K. K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*. New York: Wiley, 1999.

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 currently is pursuing 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.

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 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 with 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. 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 Tau Phi.