An Efficient Architecture for Two-Dimensional Discrete Wavelet Transform

Po-Cheng Wu and Liang-Gee Chen, *Fellow, IEEE*

***Abstract*—This paper proposes an efficient architecture for the two-dimensional discrete wavelet transform (2-D DWT). The proposed architecture includes a transform module, a RAM module, and a multiplexer. In the transform module, we employ the polyphase decomposition technique and the coefficient folding technique to the decimation filters of stages 1 and 2, respectively. In comparison with other 2-D DWT architectures, the advantages of the proposed architecture are 100% hardware utilization, fast computing time (0.5–0.67 times of the parallel filters'), regular data flow, and low control complexity, making this architecture suitable for next generation image compression systems, e.g., JPEG-2000.**

***Index Terms*—Decimation filters, discrete wavelet transform, image compression, JPEG-2000, multirate digital signal processing.**

I. Introduction

With the rapid progress of VLSI design technologies, many processors based on audio and image signal processing have been developed recently. The two-dimensional discrete wavelet transform (2-D DWT) plays a major role in the JPEG-2000 image compression standard [1]. Presently, research on the DWT is attracting a great deal of attention [2]–[6]. In addition to audio and image compression [7]–[10], the DWT has important applications in many areas, such as computer graphics, numerical analysis, radar target distinguishing and so forth. The architecture of the 2-D DWT is mainly composed of the multirate filters. Because extensive computation is involved in the practical applications, e.g., digital cameras, high-efficiency and low-cost hardware is indispensable.

At present, many VLSI architectures for the 2-D DWT have been proposed to meet the requirements of real-time processing. However, because the filtering operations are required in both the horizontal and vertical directions, designing a highly efficient architecture at a low cost is difficult. Lewis and Knowles [11] used the four-tap Daubechies filter to design a 2-D DWT architecture. Parhi and Nishitani [12] proposed two architectures that combine the word-parallel and digital-serial methodologies. Chakrabarti and Vishwanath [13] presented the nonseparable architecture and the SIMD array architecture. Vishwanath *et al.* [14] employed two systolic array filters and two parallel filters to implement the 2-D DWT. The modified version uses four parallel filters as reported in [15] and [16]. Chuang and Chen [17] proposed a parallel pipelined VLSI array architecture for the 2-D DWT. Chen and Bayoumi [18] presented a scalable systolic array architecture. Other 2-D DWT architectures have been reported in [19]–[23].

Among the various architectures, the best-known design for the 2-D DWT is the parallel filter architecture [15], [16]. The design of the parallel filter architecture is based on the modified recursive pyramid algorithm (MRPA) [13], which intersperses the computation of the second and following levels among the computation of the first level. The MRPA is feasible for the 1-D DWT architecture, but is not suitable for the 2-D DWT, because the hardware utilization is inefficient and a complicated control circuit results from the interleaving data flow. Therefore, in this paper, we propose a new VLSI architecture for the separable 2-D DWT. The advantages of the proposed architecture are the 100% hardware utilization, fast computing time, regular data flow, and low control complexity. Additionally, because of the regular structure, the proposed architecture can easily be scaled with the filter length and the 2-D DWT level.

This paper is organized as follows. Section II introduces the 2-D DWT algorithm. Section III discusses the previous design techniques. In Section IV, an efficient architecture for the 2-D DWT is proposed. Section V compares the performance of various 2-D DWT architectures. Finally, we state our conclusions in Section VI.

II. 2-D DWT Algorithm

The proposed architecture deals with the separable 2-D DWT, whose mathematical formulas are defined as follows:

$$x^J_{\mathrm{LL}}(n_1,n_2) = \sum_{i_1=0}^{K-1}\sum_{i_2=0}^{K-1} g(i_1)\cdot g(i_2)\cdot x^{J-1}_{\mathrm{LL}}(2n_1-i_1)(2n_2-i_2) \quad (1)$$

$$x^J_{\mathrm{LH}}(n_1,n_2) = \sum_{i_1=0}^{K-1}\sum_{i_2=0}^{K-1} g(i_1)\cdot h(i_2)\cdot x^{J-1}_{\mathrm{LL}}(2n_1-i_1)(2n_2-i_2) \quad (2)$$

$$x^J_{\mathrm{HL}}(n_1,n_2) = \sum_{i_1=0}^{K-1}\sum_{i_2=0}^{K-1} h(i_1)\cdot g(i_2)\cdot x^{J-1}_{\mathrm{LL}}(2n_1-i_1)(2n_2-i_2) \quad (3)$$

$$x^J_{\mathrm{HH}}(n_1,n_2) = \sum_{i_1=0}^{K-1}\sum_{i_2=0}^{K-1} h(i_1)\cdot h(i_2)\cdot x^{J-1}_{\mathrm{LL}}(2n_1-i_1)(2n_2-i_2) \quad (4)$$

Manuscript received June 3, 1999; revised August 11, 2000. This paper was recommended by Associate Editor J.-N. Hwang.

P.-C. Wu is with the Information Technology Division, Institute for Information Industry, Taipei, Taiwan, R.O.C. (e-mail: pcwu@netrd.iii.org.tw).

L.-G. Chen is with the Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan, R.O.C. (e-mail: lgchen@cc.ee.ntu.edu.tw).

Publisher Item Identifier S 1051-8215(01)03013-0.

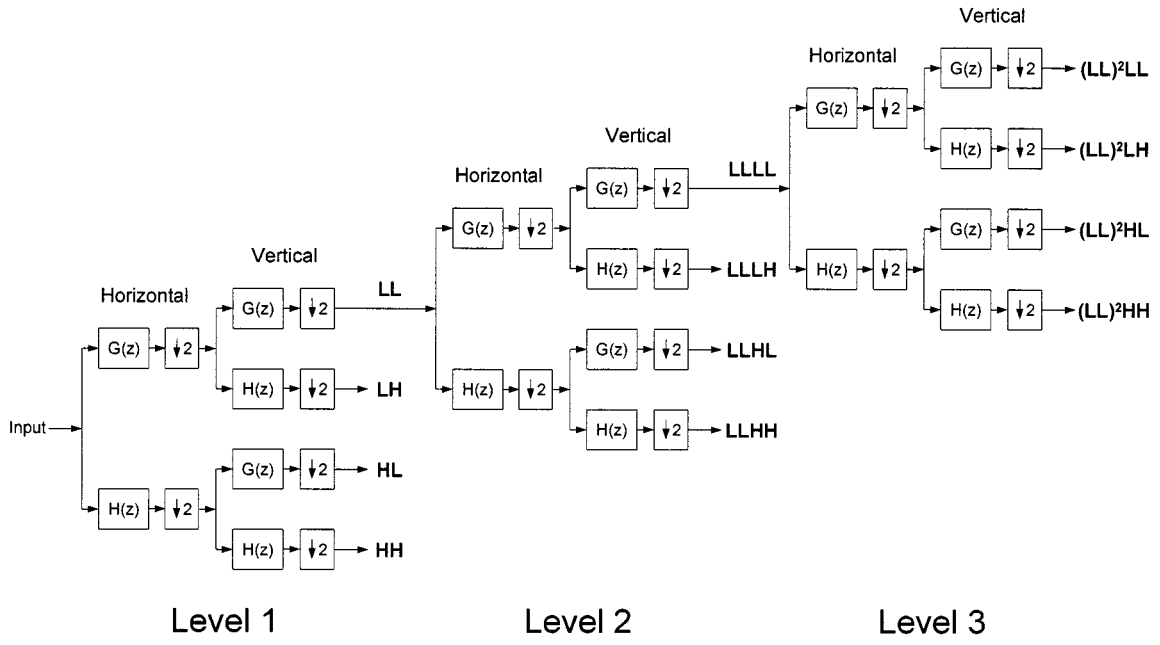

Fig. 1. Three-level 2-D DWT.

where

J	2-D DWT level;
K	filter length;
$g(n)$	impulse responses of the low-pass filter $G(z)$;
$h(n)$	impulse responses of the high-pass filter $H(z)$;
$x^0_{\mathrm{LL}}(n_1, n_2)$	input image.

Fig. 1 illustrates a three-level 2-D DWT. Each decomposition level comprises two stages: stage 1 performs horizontal filtering, and stage 2 performs vertical filtering. In the first-level decomposition, the size of the input image is $N \times N$, and the outputs are the three subbands *LH*, *HL*, and *HH*, of size $N/2 \times N/2$. In the second-level decomposition, the input is the *LL* band and the outputs are the three subbands *LLLH*, *LLHL*, and *LLHH*, of size $N/4 \times N/4$. In the third-level decomposition, the input is the *LLLL* band and the outputs are the four subbands $(LL)^2LL, (LL)^2LH, (LL)^2HL$, and $(LL)^2HH$, of size $N/8 \times N/8$. The multi-level 2-D DWT can be extended in an analogous manner. Fig. 2 shows the result of the "Lena" image after a three-level 2-D DWT.

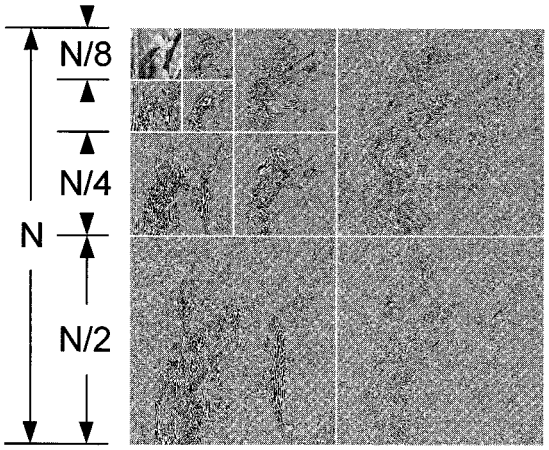

Fig. 2. Result of the "Lena" image after a three-level 2-D DWT.

III. Previous Techniques

At present, the best-known architecture for the 2-D DWT is the parallel filter architecture [15], [16]. The design of the parallel filter architecture is based on the MRPA [13]. The MRPA is initially proposed for the 1-D DWT architectures. As illustrated in Fig. 3, the MRPA intersperses the computation of the second and following levels among the computation of the first level. Because of the decimation operation, the quantity of processing data in each level is half of that in the previous level. The total quantity of processing data can be counted as follows:

$$\sum_{L=1}^{J} \frac{N}{2^{L-1}} = N + \frac{N}{2} + \frac{N}{2^2} + \frac{N}{2^3} + \ldots + \frac{N}{2^{J-1}} = 2(1 - 2^{-J})N \tag{5}$$

where

J	1-D DWT level;
N	quantity of processing data in the first level;
$N/2$	that of the second level;
$\vdots$	
$N/2^{J-1}$	that of the Jth level.

Hence, if the 1-D DWT level J is large enough, (5) will become

$$2(1 - 2^{-J})N \approx 2N = N + N. \tag{6}$$

Because the quantity of processing data in the second and following levels (i.e., N) is the same as that in the first level (i.e., N), the computing time of the first level can be filled as shown in Fig. 3. The hardware utilization is efficient. Hence, the MRPA is feasible for the 1-D DWT architectures.

However, as illustrated in Fig. 4, we find that the MRPA is not suitable for the 2-D DWT architectures. Since the quantity of processing data in each level is a quarter of that in the previous

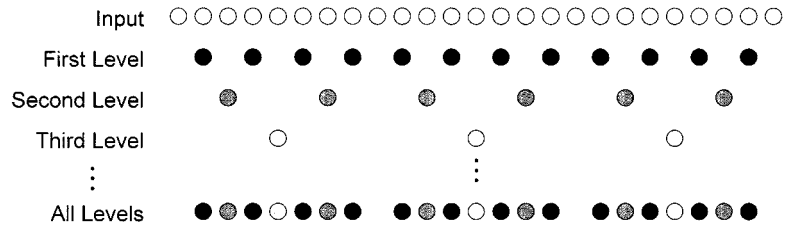

Fig. 3. MRPA for the 1-D DWT.

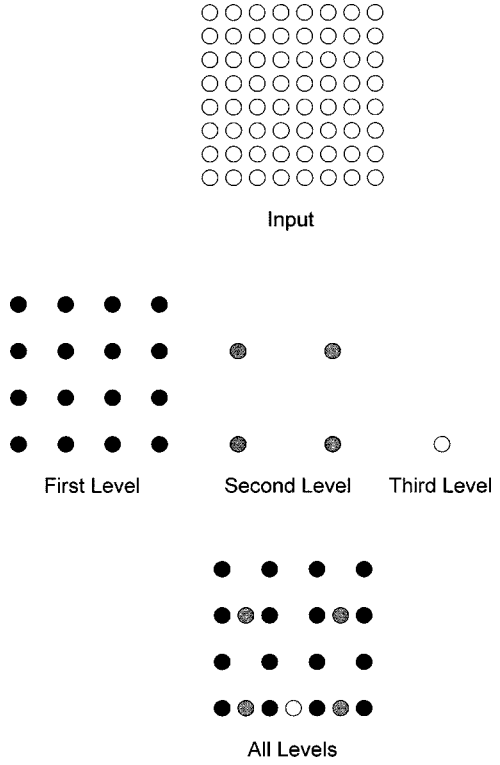

Fig. 4. MRPA for the 2-D DWT.

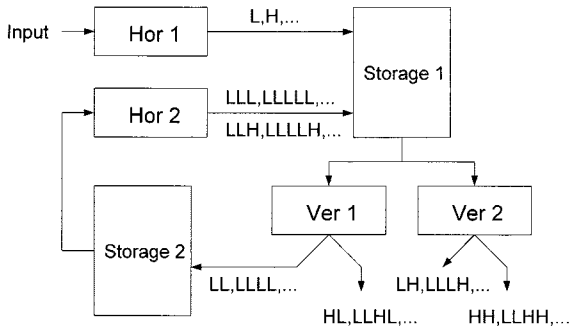

Fig. 5. Parallel filter architecture.

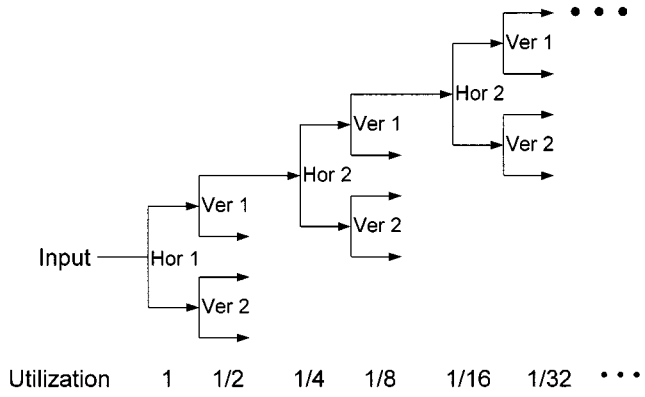

Fig. 6. Job allocation of the parallel filter architecture.

level, the total quantity of processing data is counted as follows:

$$\sum_{L=1}^{J}\frac{N^2}{4^{L-1}} = N^2+\frac{N^2}{4}+\frac{N^2}{4^2}+\frac{N^2}{4^3}+\ldots+\frac{N^2}{4^{J-1}} = \frac{4}{3}(1-4^{-J})N^2 \tag{7}$$

where

J 2-D DWT level;
N^2 quantity of processing data in the first level;
$N^2/4$ that of the second level;
$\vdots$
$N^2/4^{J-1}$ that of the Jth level.

If the 2-D DWT level J is large enough, (7) will become

$$\frac{4}{3}(1-4^{-J})N^2 \approx \frac{4}{3}N^2 = N^2+\frac{1}{3}N^2. \tag{8}$$

Because the quantity of processing data in the second and following levels (i.e., $N^2/3$) is only one third of that in the first level (i.e., N^2), the computing time of the first level cannot be filled, as shown in Fig. 4. Hence, the hardware utilization is inefficient, and a complicated control circuit results from the interleaving data flow.

Fig. 5 illustrates the parallel filter architecture, which is composed of four filters (Hor 1, Hor 2, Ver 1, and Ver 2) and two transpose memories (Storage 1 and Storage 2) for row–column transposition. Hor 1 performs the horizontal filtering of the first level. Hor 2 performs the horizontal filtering of the second and following levels. Ver 1 and Ver 2 perform the vertical filtering of all levels.

Fig. 6 shows the job allocation of the parallel filter architecture. According to Fig. 6, we can compute the individual and average hardware utilization of these four filters for different 2-D DWT levels as follows:

$$\text{Hor } 1: 1 \tag{9}$$

$$\text{Ver } 1: \sum_{L=1}^{J}\frac{1}{2\cdot 4^{L-1}} = \frac{1}{2}+\frac{1}{8}+\frac{1}{32}+\ldots+\frac{1}{2\cdot 4^{J-1}} = \frac{2}{3}(1-4^{-J}) \tag{10}$$

TABLE I
HARDWARE UTILIZATION OF THE PARALLEL FILTER ARCHITECTURE FOR DIFFERENT 2-D DWT LEVELS

2-D DWT Levels	Hardware Utilization				
	Hor 1	Ver 1	Ver 2	Hor 2	Average
1	1=**100%**	1/2= **50%**	1/2= **50%**	**0**	**50%**
2	1=**100%**	5/8= **62.5%**	5/8= **62.5%**	1/4= **25%**	**62.5%**
3	1=**100%**	21/32= **65.63%**	21/32= **65.63%**	5/16= **31.25%**	**65.63%**
4	1=**100%**	85/128= **66.41%**	85/128= **66.41%**	21/64= **32.81%**	**66.41%**
6	1=**100%**	1365/2048= **66.65%**	1365/2048= **66.65%**	341/1024= **33.30%**	**66.65%**
8	1=**100%**	21845/32768= **66.67%**	21845/32768= **66.67%**	5461/16384= **33.33%**	**66.67%**

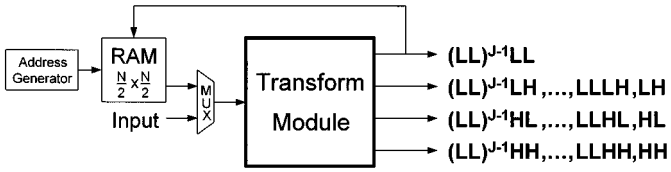

Fig. 7. Proposed 2-D DWT architecture.

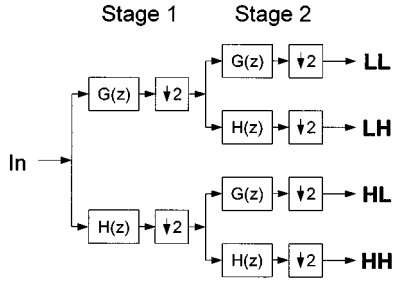

Fig. 8. Tree-structured transform module.

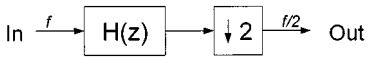

Fig. 9. Two-fold decimation filter.

$$\text{Ver } 2: \sum_{L=1}^{J} \frac{1}{2\cdot 4^{L-1}} = \frac{1}{2}+\frac{1}{8}+\frac{1}{32}+\ldots+\frac{1}{2\cdot 4^{J-1}} = \frac{2}{3}(1-4^{-J}) \tag{11}$$

$$\text{Hor } 2: \sum_{L=2}^{J} \frac{1}{4^{L-1}} = 0+\frac{1}{4}+\frac{1}{16}+\frac{1}{64}+\ldots+\frac{1}{4^{J-1}} = \frac{1}{3}\left(1-4^{-(J-1)}\right) \tag{12}$$

$$\text{Average}: \frac{1}{4}(\text{Hor } 1+\text{Ver } 1+\text{Ver } 2+\text{Hor } 2) = \frac{2}{3}(1-4^{-J}) \tag{13}$$

where J is the 2-D DWT level.

Table I lists the hardware utilization of the parallel filter architecture for different 2-D DWT levels. In one-level 2-D DWT, the hardware utilization of the parallel filter architecture is only 50%. As the 2-D DWT level increases, the utilization converges to 66.67%. Because of inefficient hardware utilization, the parallel filter architecture requires a longer computing time, the main problem of the parallel filter architecture as well as the present 2-D DWT architecture design.

IV. PROPOSED 2-D DWT ARCHITECTURE

The block diagram of the proposed 2-D DWT architecture is shown in Fig. 7, which includes a transform module, a RAM module, and a multiplexer. The size of the RAM module is $N^2/4$. The decomposition scheme is level by level and described as follows. In the first-level decomposition, the multiplexer selects data from the input image. The transform module decomposes the input image to the four subbands *LL*, *LH*, *HL*, and *HH*, and saves the *LL* band to the RAM module. After finishing the first-level decomposition, the multiplexer selects data from the RAM module. The *LL* band is then sent into the transform module to perform the second-level decomposition. The transform module decomposes the *LL* band to the four subbands *LLLL*, *LLLH*, *LLHL*, and *LLHH*, and saves the *LLLL* band to the RAM module. After finishing the second-level decomposition, the multiplexer selects data from the RAM module. The *LLLL* band is then sent into the transform module to perform the third-level decomposition. The transform module decomposes the *LLLL* band to the four subbands $(LL)^2LL, (LL)^2LH, (LL)^2HL$, and $(LL)^2HH$, and saves the $(LL)^2LL$ band to the RAM module. This procedure repeats until the desired level J (i.e., the last level) is finished.

The advantage of such a scheme is that the data flow is very regular. We can concentrate our effort to efficiently design the transform module. As shown in Fig. 8, the transform module is tree-structured and comprises two stages. Stage 1 performs horizontal filtering, and stage 2 performs vertical filtering. To design the transform module efficiently, we assume "$\mathbf{a}$" to be the area cost and "$\mathbf{t}$" to be the time cost required in stage 1. According to the original design as shown in Fig. 8, the number of filters required in stage 2 is double that of stage 1. That is, $\mathbf{2a}$ is the area cost required in stage 2. On the other hand, because of the decimation operation in stage 1, the quantity of data for filtering in each filter of stage 2 is half that of stage 1. Hence, the computing time required in stage 2 is half of $\mathbf{t}$, i.e., $\mathbf{t/2}$. Since stage 2 is cascaded after stage 1, stage 2 can not work until stage 1 finishes its job. Therefore, we find that there will be $\mathbf{2a}\times(\mathbf{t}-\mathbf{t/2}) = \mathbf{at}$ hardware idle in stage 2. In other words, the hardware utilization in the original design is inefficient.

In order to solve this problem, we consider a single decimation filter as shown in Fig. 9. The frequency labels "f" and

TABLE II
DATA FLOW OF THE DECIMATION FILTER EMPLOYING THE POLYPHASE DECOMPOSITION TECHNIQUE. (SW: DIRECTIONS OF THE SWITCH)

Clk	SW	In	Odd Part	Even Part	Out
0	0	$x(0)$	$a_1x(0)$		
1	1	$x(1)$		$a_0x(1)$	$a_0x(1)+a_1x(0)$
2	0	$x(2)$	$a_1x(2)+a_3x(0)$		
3	1	$x(3)$		$a_0x(3)+a_2x(1)$	$a_0x(3)+a_1x(2)+a_2x(1)+a_3x(0)$
4	0	$x(4)$	$a_1x(4)+a_3x(2)$		
5	1	$x(5)$		$a_0x(5)+a_2x(3)$	$a_0x(5)+a_1x(4)+a_2x(3)+a_3x(2)$
6	0	$x(6)$	$a_1x(6)+a_3x(4)$		
7	1	$x(7)$		$a_0x(7)+a_2x(5)$	$a_0x(7)+a_1x(6)+a_2x(5)+a_3x(4)$
8	0	$x(8)$	$a_1x(8)+a_3x(6)$		
9	1	$x(9)$		$a_0x(9)+a_2x(7)$	$a_0x(9)+a_1x(8)+a_2x(7)+a_3x(6)$

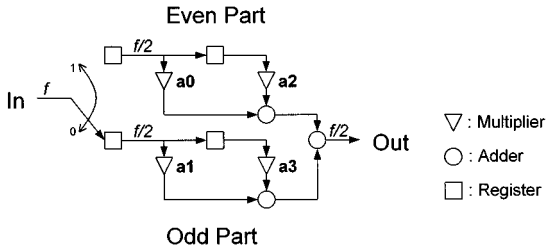

Fig. 10. Decimation filter employing the polyphase decomposition technique.

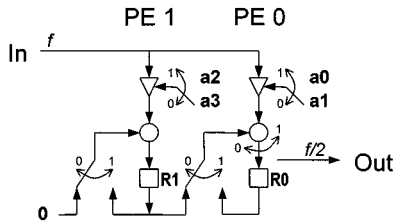

Fig. 11. The decimation filter employing the coefficient folding technique.

"$f/2$" imply that the output frequency is half the input frequency. The decimation filter can be implemented directly by a filter followed by a two-folded decimator. However, the decimator discards one sample out of every two samples at the filter output, causing poor hardware utilization. Hence, we employ two different design techniques to enhance its performance. The first technique is the polyphase decomposition technique as illustrated in Fig. 10, which decomposes the filter coefficients into even-ordered and odd-ordered parts. In the even clock cycles, the input data are fed to the odd part and multiplied with the odd-ordered coefficients. In the odd clock cycles, the input data are fed to the even part and multiplied with the even-ordered coefficients. The output data are the sum of the odd and even parts. The internal clock rate is half the input clock rate after employing the polyphase decomposition technique. Therefore, we can double the input clock rate to increase the throughput. When the quantity of processing data is the same, the computing time will be reduced to half. Thus, this technique can reduce the time cost to a half. We use the symbol "$\mathbf{T/2}$" to represent the polyphase decomposition technique. Table II shows the data flow of the decimation filter employing the polyphase decomposition technique.

The second technique is the coefficient folding technique. As illustrated in Fig. 11, every two coefficients share one set of a multiplier, adder, and register. The switches control the data path. The operation of Fig. 11 is described as follows. Viewing the PE0 first, in clock-cycle 0, the input data $x(0)$ is multiplied with the coefficient a_1 and added with the content of R1 (initially zero). The result $a_1x(0)$ is then stored to R0. In clock-cycle 1, the input data $x(1)$ is multiplied with the coefficient a_0 and added with the content of R0, i.e., $a_1x(0)$. The result $a_0x(1) + a_1x(0)$ is then output. In clock-cycle 2, the input data $x(2)$ is multiplied with the coefficient a_1 and added with the content of R1, i.e., $a_2x(1) + a_3x(0)$. The result $a_1x(2) + a_2x(1) + a_3x(0)$ is then stored to R0. In clock-cycle 3, the input data $x(3)$ is multiplied with the coefficient a_0 and added with the content of R0, i.e., $a_1x(2) + a_2x(1) + a_3x(0)$. The result $a_0x(3)+a_1x(2)+a_2x(1)+a_3x(0)$ is then output. The following clock cycles are arranged in an analogous manner. The operation of the PE1 is similar to the PE0. Because every two coefficients share one set of a multiplier, adder, and register, this technique can approximately reduce the area cost to a half. We use the symbol "$\mathbf{A/2}$" to represent the coefficient folding technique. Table III shows the data flow of the decimation filter employing the coefficient folding technique.

Now, we employ these two design techniques to the decimation filters of stages 1 and 2, respectively. Hence, four different design methods are derived for the transform module. The design strategy (including the original design) is listed in Table IV. From Table IV, we find that if we employ the polyphase decomposition technique to stage 1 and the coefficient folding technique to stage 2, the area and time cost will both be the same $\mathbf{a}$ and $\mathbf{t/2}$ in stages 1 and 2. Thus, the total area cost is $\mathbf{2a}$ and the total time cost is $\mathbf{t/2}$. The AT product is reduced from $\mathbf{3at}$ to $\mathbf{at}$, and no hardware is idle in stage 2. Therefore, the performance of the new design method is three times more efficient than the original design. In contrast, the other design methods, as listed in Table IV, cause the hardware to be idle in stage 2. Hence, they are not efficient design schemes.

In stage 2 of the transform module, because the image data are fed by a raster-scan mode, each coefficient requires a line delay to store the row data for vertical filtering. Therefore, the registers in Fig. 11 need to be replaced with the line delays for ver-

TABLE III
DATA FLOW OF THE DECIMATION FILTER EMPLOYING THE COEFFICIENT FOLDING TECHNIQUE. (SW: DIRECTIONS OF THE SWITCHES)

Clk	SW	In	PE 1	PE 0	Out
0	0	$x(0)$	$a_3x(0)$	$a_1x(0)$	
1	1	$x(1)$	$a_2x(1)+a_3x(0)$	$a_0x(1)+a_1x(0)$	$a_0x(1)+a_1x(0)$
2	0	$x(2)$	$a_3x(2)$	$a_1x(2)+a_2x(1)+a_3x(0)$	
3	1	$x(3)$	$a_2x(3)+a_3x(2)$	$a_0x(3)+a_1x(2)+a_2x(1)+a_3x(0)$	$a_0x(3)+a_1x(2)+a_2x(1)+a_3x(0)$
4	0	$x(4)$	$a_3x(4)$	$a_1x(4)+a_2x(3)+a_3x(2)$	
5	1	$x(5)$	$a_2x(5)+a_3x(4)$	$a_0x(5)+a_1x(4)+a_2x(3)+a_3x(2)$	$a_0x(5)+a_1x(4)+a_2x(3)+a_3x(2)$
6	0	$x(6)$	$a_3x(6)$	$a_1x(6)+a_2x(5)+a_3x(4)$	
7	1	$x(7)$	$a_2x(7)+a_3x(6)$	$a_0x(7)+a_1x(6)+a_2x(5)+a_3x(4)$	$a_0x(7)+a_1x(6)+a_2x(5)+a_3x(4)$
8	0	$x(8)$	$a_3x(8)$	$a_1x(8)+a_2x(7)+a_3x(6)$	
9	1	$x(9)$	$a_2x(9)+a_3x(8)$	$a_0x(9)+a_1x(8)+a_2x(7)+a_3x(6)$	$a_0x(9)+a_1x(8)+a_2x(7)+a_3x(6)$

TABLE IV
DESIGN STRATEGY OF THE TRANSFORM MODULE. ($\mathbf{T/2}$: POLYPHASE DECOMPOSITION TECHNIQUE; $\mathbf{A/2}$: COEFFICIENT FOLDING TECHNIQUE)

Methods		Stage 1		Stage 2		Total Area	Total Time	AT Prod.	Stage 2 Idle
Stage 1	Stage 2	Area	Time	Area	Time				
Original Design		a	t	$2a$	$t/2$	$3a$	t	$3at$	at
T/2	**T/2**	a	**t/2**	$2a$	$t/4$	$3a$	**t/2**	$3at/2$	$at/2$
A/2	**A/2**	$a/2$	t	a	$t/2$	$3a/2$	t	$3at/2$	$at/2$
T/2	**A/2**	a	**t/2**	a	**t/2**	$2a$	**t/2**	at	**0**
A/2	**T/2**	$a/2$	t	$2a$	$t/4$	$5a/2$	t	$5at/2$	$3at/2$

TABLE V
DATA FLOW OF THE MODIFIED DECIMATION FILTER EMPLOYING THE COEFFICIENT FOLDING TECHNIQUE FOR VERTICAL FILTERING. (SW: DIRECTIONS OF THE SWITCHES; N: LENGTH OF THE ROW; $x^*(n)$: nTH ROW DATA)

Clk	SW	In	PE 1	PE 0	Out
0	0	$x^*(0)$	$a_3x^*(0)$	$a_1x^*(0)$	
N	1	$x^*(1)$	$a_2x^*(1)+a_3x^*(0)$	$a_0x^*(1)+a_1x^*(0)$	$a_0x^*(1)+a_1x^*(0)$
$2N$	0	$x^*(2)$	$a_3x^*(2)$	$a_1x^*(2)+a_2x^*(1)+a_3x^*(0)$	
$3N$	1	$x^*(3)$	$a_2x^*(3)+a_3x^*(2)$	$a_0x^*(3)+a_1x^*(2)+a_2x^*(1)+a_3x^*(0)$	$a_0x^*(3)+a_1x^*(2)+a_2x^*(1)+a_3x^*(0)$
$4N$	0	$x^*(4)$	$a_3x^*(4)$	$a_1x^*(4)+a_2x^*(3)+a_3x^*(2)$	
$5N$	1	$x^*(5)$	$a_2x^*(5)+a_3x^*(4)$	$a_0x^*(5)+a_1x^*(4)+a_2x^*(3)+a_3x^*(2)$	$a_0x^*(5)+a_1x^*(4)+a_2x^*(3)+a_3x^*(2)$
$6N$	0	$x^*(6)$	$a_3x^*(6)$	$a_1x^*(6)+a_2x^*(5)+a_3x^*(4)$	
$7N$	1	$x^*(7)$	$a_2x^*(7)+a_3x^*(6)$	$a_0x^*(7)+a_1x^*(6)+a_2x^*(5)+a_3x^*(4)$	$a_0x^*(7)+a_1x^*(6)+a_2x^*(5)+a_3x^*(4)$
$8N$	0	$x^*(8)$	$a_3x^*(8)$	$a_1x^*(8)+a_2x^*(7)+a_3x^*(6)$	
$9N$	1	$x^*(9)$	$a_2x^*(9)+a_3x^*(8)$	$a_0x^*(9)+a_1x^*(8)+a_2x^*(7)+a_3x^*(6)$	$a_0x^*(9)+a_1x^*(8)+a_2x^*(7)+a_3x^*(6)$

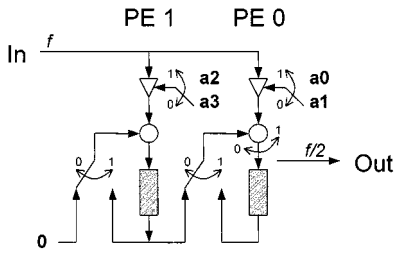

Fig. 12. The modified decimation filter employing the coefficient folding technique for vertical filtering.

tical filtering. Fig. 12 shows the modified result. The data flow is shown in Table V where $x^*(n)$ represents the nth row data. Every two input rows generate one output row. Fig. 13 shows the structure of the line delay, which is composed of J select signals, $N/2^J, N/2^{J-1}, N/2^{J-2}, \ldots, N/4, N/2$, and J storage blocks of size $N/2^J, N/2^J, N/2^{J-1}, N/2^{J-2}, \ldots, N/8, N/4$. The size of the line delay in the different decomposition levels is described below.

In the first-level decomposition, the select signal $N/2$ is enabled, and the others are disabled. The size of the line delay is the sum of all storage blocks as follows:

$$\frac{N}{2^J}+\frac{N}{2^J}+\frac{N}{2^{J-1}}+\frac{N}{2^{J-2}}+\cdots+\frac{N}{16}+\frac{N}{8}+\frac{N}{4}=\frac{N}{2}. \tag{14}$$

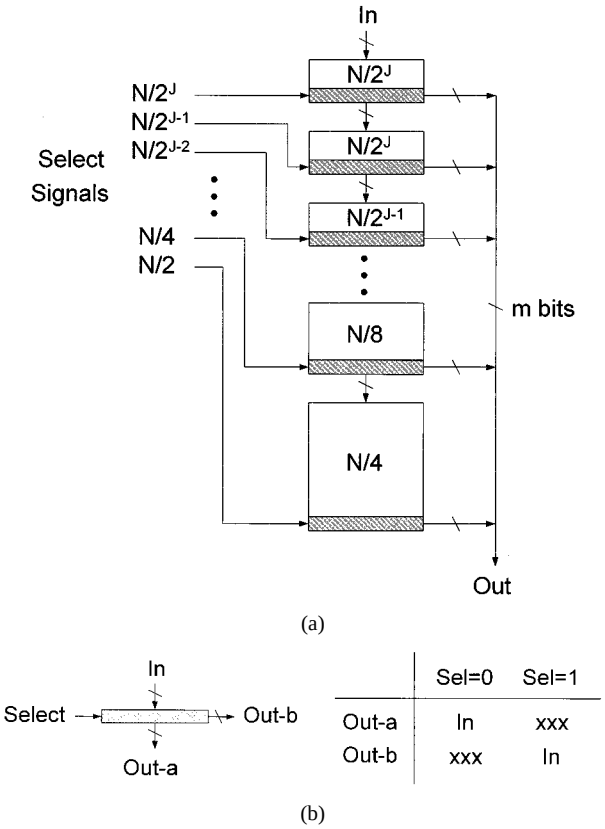

Fig. 13. (a) Line delay with select signals to change its size to $N/2, N/4, N/8, \ldots, N/2^J$ in the different decomposition levels. (b) The 1 to 2 demultiplexer used in the line delay.

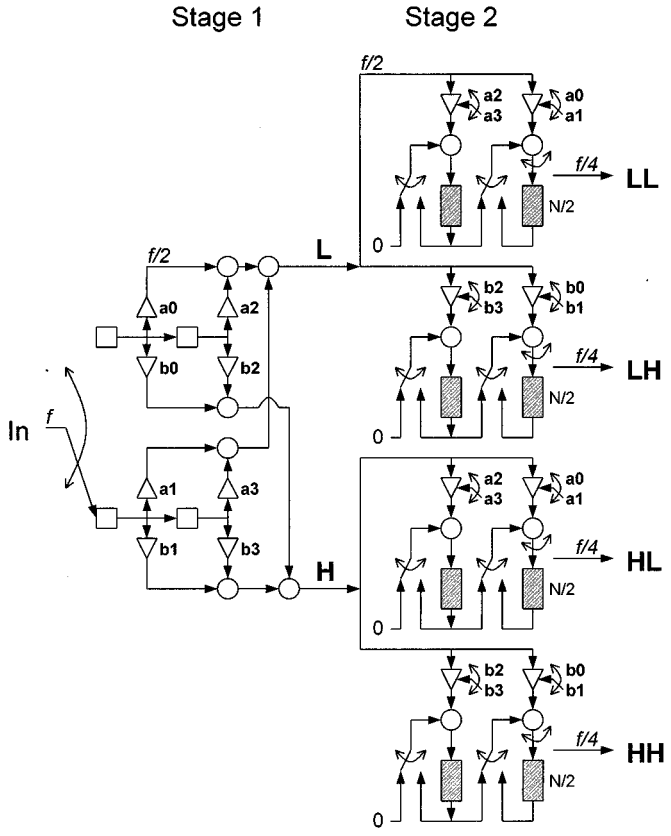

Fig. 14. Transform module employing the polyphase decomposition technique to stage 1 and the coefficient folding technique to stage 2.

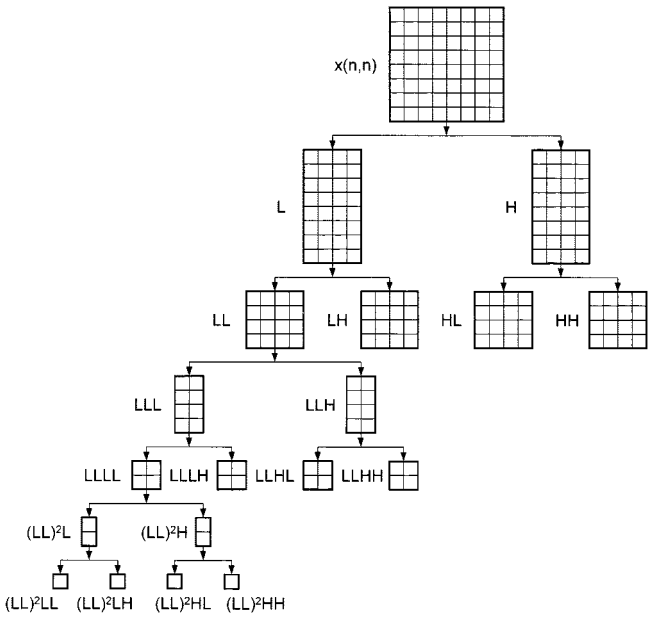

Fig. 15. Three-level 2-D DWT in the proposed architecture. The decomposition begins with an 8×8 block in the first level, and ends with four 1×1 pixels in the third level.

Thus, it can store the row data output from the decimation filter of stage 1. In the second-level decomposition, the select signal $N/4$ is enabled, and the others are disabled. The size of the line delay is the sum of the previous $J-1$ storage blocks as follows:

$$\frac{N}{2^J}+\frac{N}{2^J}+\frac{N}{2^{J-1}}+\frac{N}{2^{J-2}}+\cdots+\frac{N}{16}+\frac{N}{8}=\frac{N}{4}. \quad (15)$$

In the following decomposition levels, the select signals change the size of the line delay to $N/8$ in the third level, $N/16$ in the fourth level, $\ldots, N/2^{J-1}$ in the $(J-1)$th level, $N/2^J$ in the Jth level.

Assume that the low-pass filter has four taps: a_0, a_1, a_2, and a_3, and the high-pass filter has four taps: b_0, b_1, b_2, and b_3. Fig. 14 illustrates the transform module employing both the polyphase decomposition and the coefficient folding techniques. The frequency labels "f," "$f/2$," and "$f/4$" imply that the output frequency is a quarter of the input frequency. In stage 1, because we use the FIR direct-form to implement the polyphase decomposition technique, the low- and high-pass decimation filters can share the same registers. Here, we have assumed that the filters in stages 1 and 2 have the same length, but in practice, this condition is not necessary for the correct operation. In addition, because we employ the polyphase decomposition technique in the decimation filters of stage 1, the internal clock rate of the transform module is half the input clock rate. Fig. 15 illustrates the three-level 2-D DWT in the proposed architecture. The decomposition begins with an 8×8 block in the first level, and ends with four 1×1 pixels in the third level. Table VI shows the data flow according to the ports of the transform module. The clock cycles 0–63 perform the first-level decomposition, the clock cycles 64–79 perform the second-level decomposition, and the clock cycles 80–83 perform the third-level decomposition. Because of the regular structure, the proposed architecture can be easily scaled with the filter length and the 2-D DWT level.

TABLE VI
DATA FLOW OF FIG. 15

Clk	In	L	H	LL	LH	HL	HH
0	$x(0,0), x(0,1)$	$L(0,0)$	$H(0,0)$				
2	$x(0,2), x(0,3)$	$L(0,1)$	$H(0,1)$				
4	$x(0,4), x(0,5)$	$L(0,2)$	$H(0,2)$				
6	$x(0,6), x(0,7)$	$L(0,3)$	$H(0,3)$				
8	$x(1,0), x(1,1)$	$L(1,0)$	$H(1,0)$	$LL(0,0)$	$LH(0,0)$	$HL(0,0)$	$HH(0,0)$
10	$x(1,2), x(1,3)$	$L(1,1)$	$H(1,1)$	$LL(0,1)$	$LH(0,1)$	$HL(0,1)$	$HH(0,1)$
12	$x(1,4), x(1,5)$	$L(1,2)$	$H(1,2)$	$LL(0,2)$	$LH(0,2)$	$HL(0,2)$	$HH(0,2)$
14	$x(1,6), x(1,7)$	$L(1,3)$	$H(1,3)$	$LL(0,3)$	$LH(0,3)$	$HL(0,3)$	$HH(0,3)$
16	$x(2,0), x(2,1)$	$L(2,0)$	$H(2,0)$				
18	$x(2,2), x(2,3)$	$L(2,1)$	$H(2,1)$				
20	$x(2,4), x(2,5)$	$L(2,2)$	$H(2,2)$				
22	$x(2,6), x(2,7)$	$L(2,3)$	$H(2,3)$				
24	$x(3,0), x(3,1)$	$L(3,0)$	$H(3,0)$	$LL(1,0)$	$LH(1,0)$	$HL(1,0)$	$HH(1,0)$
26	$x(3,2), x(3,3)$	$L(3,1)$	$H(3,1)$	$LL(1,1)$	$LH(1,1)$	$HL(1,1)$	$HH(1,1)$
28	$x(3,4), x(3,5)$	$L(3,2)$	$H(3,2)$	$LL(1,2)$	$LH(1,2)$	$HL(1,2)$	$HH(1,2)$
30	$x(3,6), x(3,7)$	$L(3,3)$	$H(3,3)$	$LL(1,3)$	$LH(1,3)$	$HL(1,3)$	$HH(1,3)$
32	$x(4,0), x(4,1)$	$L(4,0)$	$H(4,0)$				
34	$x(4,2), x(4,3)$	$L(4,1)$	$H(4,1)$				
36	$x(4,4), x(4,5)$	$L(4,2)$	$H(4,2)$				
38	$x(4,6), x(4,7)$	$L(4,3)$	$H(4,3)$				
40	$x(5,0), x(5,1)$	$L(5,0)$	$H(5,0)$	$LL(2,0)$	$LH(2,0)$	$HL(2,0)$	$HH(2,0)$
42	$x(5,2), x(5,3)$	$L(5,1)$	$H(5,1)$	$LL(2,1)$	$LH(2,1)$	$HL(2,1)$	$HH(2,1)$
44	$x(5,4), x(5,5)$	$L(5,2)$	$H(5,2)$	$LL(2,2)$	$LH(2,2)$	$HL(2,2)$	$HH(2,2)$
46	$x(5,6), x(5,7)$	$L(5,3)$	$H(5,3)$	$LL(2,3)$	$LH(2,3)$	$HL(2,3)$	$HH(2,3)$
48	$x(6,0), x(6,1)$	$L(6,0)$	$H(6,0)$				
50	$x(6,2), x(6,3)$	$L(6,1)$	$H(6,1)$				
52	$x(6,4), x(6,5)$	$L(6,2)$	$H(6,2)$				
54	$x(6,6), x(6,7)$	$L(6,3)$	$H(6,3)$				
56	$x(7,0), x(7,1)$	$L(7,0)$	$H(7,0)$	$LL(3.0)$	$LH(3,0)$	$HL(3.0)$	$HH(3,0)$
58	$x(7,2), x(7,3)$	$L(7,1)$	$H(7,1)$	$LL(3,1)$	$LH(3,1)$	$HL(3,1)$	$HH(3,1)$
60	$x(7,4), x(7,5)$	$L(7,2)$	$H(7,2)$	$LL(3,2)$	$LH(3,2)$	$HL(3,2)$	$HH(3,2)$
62	$x(7,6), x(7,7)$	$L(7,3)$	$H(7,3)$	$LL(3,3)$	$LH(3,3)$	$HL(3,3)$	$HH(3,3)$
64	$LL(0,0), LL(0,1)$	$LLL(0,0)$	$LLH(0,0)$				
66	$LL(0,2), LL(0,3)$	$LLL(0,1)$	$LLH(0,1)$				
68	$LL(1,0), LL(1,1)$	$LLL(1,0)$	$LLH(1,0)$	$LLLL(0,0)$	$LLLH(0,0)$	$LLHL(0,0)$	$LLHH(0,0)$
70	$LL(1,2), LL(1,3)$	$LLL(1,1)$	$LLH(1,1)$	$LLLL(0,1)$	$LLLH(0,1)$	$LLHL(0,1)$	$LLHH(0,1)$
72	$LL(2,0), LL(2,1)$	$LLL(2,0)$	$LLH(2,0)$				
74	$LL(2,2), LL(2,3)$	$LLL(2,1)$	$LLH(2,1)$				
76	$LL(3,0), LL(3,1)$	$LLL(3.0)$	$LLH(3.0)$	$LLLL(1,0)$	$LLLH(1,0)$	$LLHL(1,0)$	$LLHH(1,0)$
78	$LL(3,2), LL(3,3)$	$LLL(3,1)$	$LLH(3,1)$	$LLLL(1,1)$	$LLLH(1,1)$	$LLHL(1,1)$	$LLHH(1,1)$
80	$LLLL(0,0), LLLL(0,1)$	$(LL)^2L(0,0)$	$(LL)^2H(0,0)$				
82	$LLLL(1,0), LLLL(1,1)$	$(LL)^2L(1,0)$	$(LL)^2H(1,0)$	$(LL)^2LL(0,0)$	$(LL)^2LH(0,0)$	$(LL)^2HL(0,0)$	$(LL)^2HH(0,0)$

V. PERFORMANCE COMPARISONS

The typical 2-D DWT architectures include the parallel filter architecture [16], direct architecture [14], nonseparable architecture [13], SIMD architecture [13], and systolic-parallel architecture [14]. In Table VII, we compare the performance of our architecture and these 2-D DWT architectures in terms of the number of multipliers, the number of adders, storage size, computing time, control complexity, and hardware utilization. The computing time has been normalized to the same internal clock rate. The parameter K is the filter length, N^2 is the image size, and J is the 2-D DWT level. The computing time of our architecture is derived as follows:

$$\begin{aligned} T &= \frac{1}{2}\sum_{L=1}^{J}\frac{N^2}{4^{L-1}} \\ &= \frac{1}{2}\left(N^2+\frac{N^2}{4}+\frac{N^2}{4^2}+\frac{N^2}{4^3}+\ldots+\frac{N^2}{4^{J-1}}\right) \\ &= \frac{2}{3}(1-4^{-J})N^2 \end{aligned} \tag{16}$$

TABLE VII
PERFORMANCE COMPARISONS OF VARIOUS 2-D DWT ARCHITECTURES. (K: FILTER LENGTH; N^2: IMAGE SIZE; AND J: 2-D DWT LEVEL)

Architectures	Multipliers	Adders	Storage Size	Computing Time	Control Complexity	Hardware Utilization
Ours	$4K$	$4K$	$N^2/4+KN+K$	$0.5N^2 \sim 0.67N^2$	Simple	100%
Parallel Filter [16]	$4K$	$4K$	$2KN+N$	N^2	Complex	Low
Direct [14]	K	K	N^2	$4N^2$	Complex	Low
Non-Separable [13]	$2K^2$	$2(K^2-1)$	$2KN$	N^2	Complex	High
SIMD [13]	$2N^2$	$2N^2$	N^2	K^2J	Complex	Low
Systolic-Parallel [14]	$4K$	$4K$	$2KN+4N$	N^2	Complex	Low

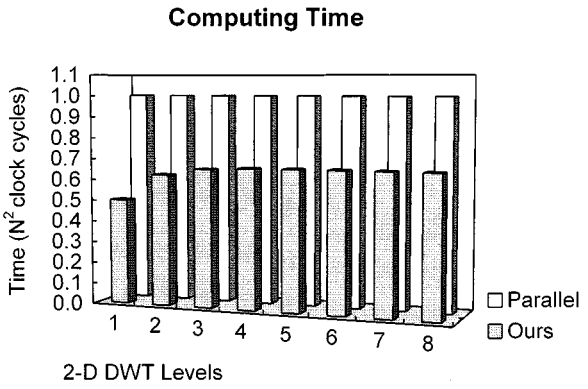

Fig. 16. Computing time of the proposed architecture and the parallel filter architecture for different 2-D DWT levels.

where the factor $1/2$ is because the internal clock rate of our architecture is half the input clock rate. Therefore, if our architecture and other architectures have the same internal clock rate, the throughput of our architecture is twice that of other architectures. To do this, doubling the input clock rate for the pixel input can be used. The outcome of the comparisons shows that our design outperforms other architectures, especially in computing time, control complexity, and hardware utilization.

We also compare the computing time and the hardware utilization between our architecture and the parallel filter architecture [16] for different 2-D DWT levels. The design of the parallel filter architecture is based on the MRPA, which intersperses the computation of the second and following levels among the computation of the first level. Fig. 16 plots the computing time for different 2-D DWT levels, showing that in one-level 2-D DWT ($J = 1$), the computing time of our architecture is $(2/3)(1 - 4^{-1})N^2 = 0.5N^2$ clock cycles. As the 2-D DWT level increases ($J > 4$), the computing time converges to $(2/3)(1 - 4^{-5})N^2 \approx 0.67N^2$ clock cycles. In contrast, the parallel filter architecture always requires the computing time of N^2 clock cycles. On the other hand, as shown in Table I, the hardware utilization of the parallel filter architecture is only 50% in one-level 2-D DWT ($J = 1$). As the 2-D DWT level increases ($J > 7$), the utilization converges to 66.67%. However, our architecture can consistently maintain 100% hardware utilization.

Concerning the storage size, the proposed architecture requires a RAM module of size $N^2/4$ to save the intermediate data. However, because the proposed architecture is mainly applied in the image compression systems, it can use the memory already existing in the systems to save the intermediate data. Hence, in this condition, the proposed architecture will not require the RAM module. The remain storage size is $KN + K$, where KN is the line delays required in stage 2 for vertical filtering, and K is the registers required in stage 1 for horizontal filtering.

VI. CONCLUSION

In recent years, many 2-D DWT architectures have been proposed to meet the requirements of real time processing. However, the hardware utilization of these architectures needs to be further improved. Therefore, in this paper, we have proposed an efficient architecture for the 2-D DWT. The proposed architecture has been correctly verified by the Verilog Hardware Description Language (Verilog HDL). The advantages of the proposed architecture are the 100% hardware utilization, fast computing time, regular data flow, and low control complexity, making this design suitable for next generation image compression systems, e.g., JPEG-2000.

REFERENCES

[1] *Coding of Still Pictures: JPEG 2000 Part I Final Committee Draft Version 1.0*, ISO/IEC JTC 1/SC 29/WG 1 (ITU-T SG8), Mar. 2000.
[2] 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.
[3] ——, "Multifrequency channel decompositions of image and wavelet models," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 37, pp. 2091–2110, Dec. 1989.
[4] P. P. Vaidyanathan, *Multirate Systems and Filter Banks*. Englewood Cliffs, NJ: Prentice-Hall, 1993.
[5] M. Vetterli and J. Kovacevic, *Wavelets and Subband Coding*. Englewood-Cliffs, NJ: Prentice Hall, 1995.
[6] G. Strang and T. Q. Nquyen, *Wavelets and Filter Banks*. Cambridge, MA: Wellesley-Cambridge Press, 1996.
[7] E. B. Richardson and N. S. Jayant, "Subband coding with adaptive prediction for 56 kbits/s audio," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-34, pp. 691–696, Aug. 1986.
[8] J. W. Woods and S. D. O'Neil, "Subband coding of images," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-34, pp. 1278–1288, Oct. 1986.
[9] J. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Trans. Signal Processing*, vol. 41, pp. 3445–3462, Dec. 1993.
[10] A. Said and W. A. Pearlman, "A new fast and efficient image codec based on set partitioning in hierarchical trees," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, pp. 243–250, June 1996.
[11] A. S. Lewis and G. Knowles, "VLSI architecture for 2-D Daubechics wavelet transform without multipliers," *Electron. Lett.*, vol. 27, pp. 171–173, Jan. 1991.
[12] K. K. Parhi and T. Nishitani, "VLSI architectures for discrete wavelet transforms," *IEEE Trans. VLSI Syst.*, vol. 1, pp. 191–202, June 1993.

[13] C. Chakrabarti and M. Vishwanath, "Efficient realization 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.
[14] M. Vishwanath, R. M. Owens, and M. J. Irwin, "VLSI architectures for the discrete wavelet transform," *IEEE Trans. Circuits Syst. II*, vol. 42, pp. 305–316, May 1995.
[15] C. Chakrabarti and C. Mumford, "Efficient realizations of analysis and synthesis filters based on the 2-D discrete wavelet transform," in *Proc. IEEE ICASSP*, May 1996, pp. 3256–3259.
[16] C. Chakrabarti and M. Vishwanath, "Architectures for wavelet transforms: A survey," *J. VLSI Signal Processing*, vol. 14, pp. 171–192, 1996.
[17] H. Y. H. Chuang and L. Chen, "VLSI architecture for fast 2-D discrete orthonormal wavelet transform," *J. VLSI Signal Processing*, vol. 10, pp. 225–236, 1995.
[18] J. Chen and M. A. Bayoumi, "A scalable systolic array architecture for 2-D discrete wavelet transforms," in *Proc. IEEE VLSI Signal Processing Workshop*, 1995, pp. 303–312.
[19] R. Rumian, "An architecture for real-time wavelet image decomposition," *Proc. IEEE Int. Symp. Circuits and Systems*, pp. 73–76, May 1994.
[20] T. C. Denk and K. K. Parhi, "Calculation of minimum number of registers in 2-D discrete wavelet transforms using lapped block processing," *Proc. IEEE Int. Symp. Circuits and Systems*, pp. 77–81, May 1994.
[21] J. Bae and V. K. Prasanna, "Synthesis of VLSI architecture for two dimensional discrete wavelet transforms," in *Proc. IEEE Int. Conf. Application Specific Array Processors*, July 1995, pp. 174–181.
[22] J. C. Limqueco and M. A. Bayoumi, "A VLSI architecture for separable 2-D discrete wavelet transforms," *J. VLSI Signal Processing*, vol. 18, pp. 125–140, 1998.
[23] G. Lafruit, F. Catthoor, J. P. H. Cornelis, and H. J. D. Man, "An efficient VLSI architecture for 2-D wavelet image coding with novel image scan," *IEEE Trans. VLSI Syst.*, vol. 7, pp. 56–68, Jan. 1999.

Po-Cheng Wu received the B.S. degree in computer science and information engineering from National Chiao Tung University, Hsinchu, Taiwan, in 1991, and the M.S. and Ph.D. degrees in electrical engineering from National Taiwan University, Taipei, Taiwan, in 1994 and 1999, respectively.

He is currently a Senior Engineer at the Institute for Information Industry, Taipei, Taiwan. His major research interests include DSP architecture design, video coding, and multimedia networking.

Dr. Wu received Acer's Long-Term Paper Award and Xerox's Academic Paper Award, both in 1994. In 1999, he received the Best Paper Award of the IEEE Workshop on Consumer Electronics.

Liang-Gee Chen (S'84–M'86–SM'94–F'00) received the B.S., M.S., and Ph.D. degrees in electrical engineering from National Cheng Kung University, Tainan, Taiwan, in 1979, 1981, and 1986, respectively.

He was an Associate Professor during 1986–1988 in the Department of Electrical Engineering, National Cheng Kung University. In 1988, he joined the Department of Electrical Engineering, National Taiwan University. During 1993–1994 he was a Visiting Consultant at 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 at 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 was the General Chairman of the 7th VLSI Design/CAD Symposium and is currently the General Chairman of the 1999 IEEE Workshop on Signal Processing Systems: Design and Implementation. He has served as Associate Editor of IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY since June 1996, and Associate Editor of IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION SYSTEMS since January 1999. He was the Associate Editor of the *Journal of Circuits, Systems, and Signal Processing* since 1999, and previously served as the Guest Editor of the *Journal of Signal Processing Systems*. He received the Best Paper Award from the R.O.C. Computer Society in 1990 and 1994, the Long-Term (Acer) Paper Award annually from 1991 to 1999, the Best Paper Award of the Asia-Pacific Conference on Circuits and Systems in the VLSI Design track in 1992, the Annual Paper Award of Chinese Engineer Society in 1993, and in 1996, the Outstanding Research Award from NSC and the Dragon Excellence Award for Acer. He is a member of Phi Tan Phi.