Generic RAM-Based Architectures for Two-Dimensional Discrete Wavelet Transform With Line-Based Method

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

Abstract—**In this paper, three generic RAM-based architectures are proposed to efficiently construct the corresponding two-dimensional architectures by use of the line-based method for any given hardware architecture of one-dimensional (1-D) wavelet filters, including conventional convolution-based and lifting-based architectures. An exhaustive analysis of two-dimensional architectures for discrete wavelet transform in the system view is also given. The first proposed architecture is for 1-level decomposition, which is presented by introducing the categories of internal line buffers, the strategy of optimizing the line buffer size, and the method of integrating any 1-D wavelet filter. The other two proposed architectures are for multi-level decomposition. One applies the recursive pyramid algorithm directly to the proposed 1-level architecture, and the other one combines the two previously proposed architectures to increase the hardware utilization. According to the comparison results, the proposed architecture outperforms previous architectures in the aspects of line buffer size, hardware cost, hardware utilization, and flexibility.**

Index Terms—**Discrete wavelet transform (DWT), lifting scheme, line-based method, recursive pyramid algorithm, VLSI architecture.**

I. Introduction

Discrete wavelet transform (DWT) has been developed as an effective tool of multi-resolution analysis since it was presented by Mallat [1]. Due to its good time-frequency characteristics, DWT has been widely used for signal analysis and compression. For example, the well-known image coding standards, MPEG-4 still texture coding and JPEG 2000 still image coding, have adopted DWT as their transform coder. However, DWT usually requires heavy arithmetic computation because it is essentially a two-channel filter bank. The lifting scheme [2] can be used to reduce the arithmetic complexity and provide in-place implementation. A tutorial lifting factorization is given in [3].

For two-dimensional (2-D) DWT, there have been many VLSI architectures proposed [4]–[8]. Nevertheless, only RAM-based architectures are the most practical for real-life designs because of their greater regularity and density of storage, compared to systolic or semisystolic routing [9]. Furthermore, the memory issues dominate the hardware cost and complexity of the architectures for 2-D DWT, instead of multipliers that decide the performance of one-dimensional (1-D) DWT architectures.

The simplest method of implementing separable 2-D DWT is to directly perform 1-D DWT in the row direction and then in the column direction. However, this direct method needs a frame buffer which is usually off-chip to store the intermediate data. According to the evaluation in [9], external memory access consumes the most power in the 2-D DWT hardware implementation. The line-based method [10] may be preferred because of the smaller external memory access. But the line-based method requires some internal line buffer which increases the die area and control complexity, and the buffer size is proportional to the image width. As a result, how to minimize the line buffer becomes a critical problem.

In order to solve the problem of internal memory access, a generic RAM-based architecture for 1-level 2-D DWT is proposed. The basic idea is to categorize the internal line buffers into data buffer and temporal buffer. A data flow for the former is proposed to give the minimal data buffer size, and a generic architecture of the latter is proposed to flexibly adopt any kind of 1-D DWT modules. Thus, the lifting-based 1-D DWT modules can be easily integrated into the 2-D architectures so as to decrease the temporal buffer size and the hardware complexity. Furthermore, two multi-level architectures based on the 1-level architecture are proposed by allocating decomposition tasks to two and three 1-D DWT modules, respectively, to meet different requirements of computing time, throughput, and hardware utilization.

This paper is organized as follows. An extensive analysis of RAM-based architectures for 2-D DWT is given in Section II. Then three generic line-based architectures for 2-D DWT are proposed in Section III. In Section IV, comparisons of the proposed architectures with the previous studies are shown to demonstrate the efficiency. Finally, conclusions of this paper are given in Section V.

II. Analysis for RAM-Based 2-D DWT Architectures

For the implementation of 2-D DWT architectures, the memory issues, including internal memory size and external frame memory access, are the most critical problems. The internal memory generally dominates the hardware cost, whereas the external frame memory access consumes the most power. In this section, we will present several RAM-based architectures

Manuscript received September 23, 2002; revised July 18, 2003. This paper was recommended by Associate Editor C.-W. Chen. This work was supported in part by the Ministry of Education (MOE) Program for Promoting Academic Excellence of Universities under Grant 89E-FA06-2-4-8, in part by National Science Council, R.O.C., under Grant 91-2215-E-002-035, and in part by the MediaTek Fellowship.

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

Digital Object Identifier 10.1109/TCSVT.2005.848307

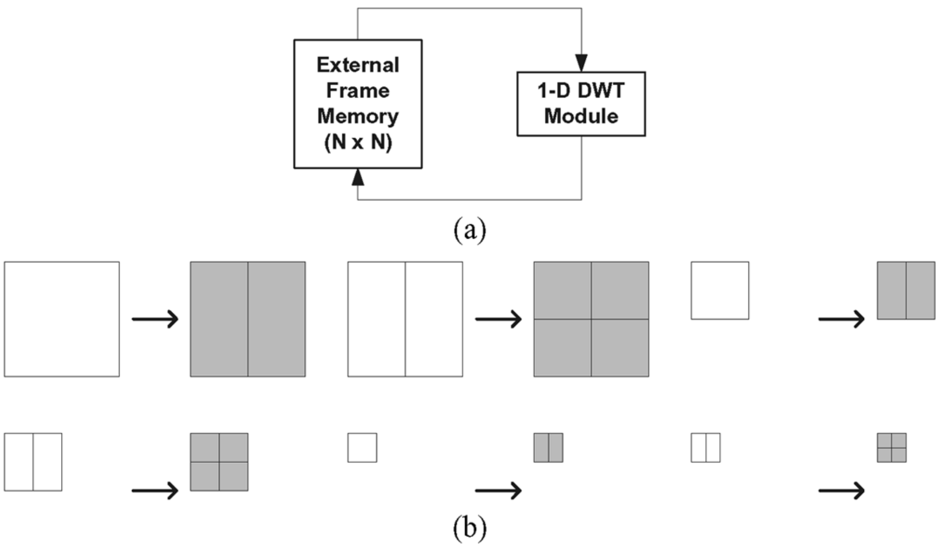


Fig. 1. Direct 2-D implementation. (a) System architecture (the number in brackets represents the memory size in terms of words). (b) Data flow of external memory access ($J = 3$; white and grey parts represent external frame memory reads and writes, respectively).

for 2-D DWT, which are categorized in the aspects of the mentioned memory issues. More specifically, these architectures are designed for separable 2-D DWT of the dyadic decomposition type, whose mathematical formulas can be expressed recursively as follows:

$$\begin{aligned} x_{LL}^{J}(n_1,n_2) &= \sum_{i_1,i_2} h(i_1)h(i_2)x_{LL}^{J-1}(2n_1-i_1,2n_2-i_2)\\ x_{LH}^{J}(n_1,n_2) &= \sum_{i_1,i_2} h(i_1)g(i_2)x_{LL}^{J-1}(2n_1-i_1,2n_2-i_2)\\ x_{HL}^{J}(n_1,n_2) &= \sum_{i_1,i_2} g(i_1)h(i_2)x_{LL}^{J-1}(2n_1-i_1,2n_2-i_2)\\ x_{HH}^{J}(n_1,n_2) &= \sum_{i_1,i_2} g(i_1)g(i_2)x_{LL}^{J-1}(2n_1-i_1,2n_2-i_2) \end{aligned} \tag{1}$$

where $h(i)$ and $g(i)$ represent the low-pass and high-pass filters, respectively, J is the DWT decomposition level, and x_{LL}^{0} is the input image. In the following, four categories of 2-D DWT architectures, including direct, row-column-column-row (RCCR), 1-level line-based, and multi-level line-based, are presented and discussed about the memory issues. A frame memory is assumed to be used for storing the original image and the DWT decomposition result.

A. Direct Architecture

The most straightforward implementation of (1) is to perform 1-D DWT in one direction and store the intermediate coefficients in the frame memory, and then to perform 1-D DWT with these intermediate coefficients in the other direction to complete 1-level 2-D DWT. For the other decomposition levels, the low-pass-lowpass (LL) subband of the current level is treated as the input signals of the next level, and the above steps are then performed recursively. As illustrated in Fig. 1(a), this direct architecture requires the least hardware cost and no internal memory due to its simplicity, but it requires much external memory access. For example, if the decomposition level is three, the data flow of the external memory access will be as shown in Fig. 1(b). Thus, the bandwidth of external memory access, including both reads and writes, can be expressed as

$$4\times\left(1+\frac{1}{4}+\frac{1}{16}+\cdots+\left(\frac{1}{4}\right)^{J-1}\right)N^2 \quad \text{(words/image)} \tag{2}$$

where N is the width and height of the image.

B. RCCR Architecture

According to (1), the priorities of row and column directions are identical. Therefore, it is unnecessary to process the row coefficients first for every level decomposition all the time, such as the direct architecture. Instead, the priority can be assumed as row-column for the odd-level decompositions and column-row for the even-level decompositions [11]. Then, the successive two row-wise or column-wise 1-D DWT decompositions can be performed simultaneously. The DWT module of this RCCR architecture can be implemented by two approaches. One is to fold the two successive decompositions into one 1-D DWT module by recursive pyramid algorithm (RPA) scheduling [12] without any line buffer. The other one is to perform the former level decomposition and store the coefficients in a line buffer of size $N/2$, and then to perform the latter level decomposition with the stored coefficients. The RCCR architecture is shown in Fig. 2(a), and the data flow of external memory access is shown in Fig. 2(b). The merging of two successive decompositions in the same direction can decrease the external memory access bandwidth by one half for every level, except the first level decomposition. The external memory access bandwidth can be formulated by

$$\left(2+2\times\left(1+\frac{1}{4}+\frac{1}{16}+\cdots+\left(\frac{1}{4}\right)^{J-1}\right)\right)N^2\text{(words/image).} \tag{3}$$

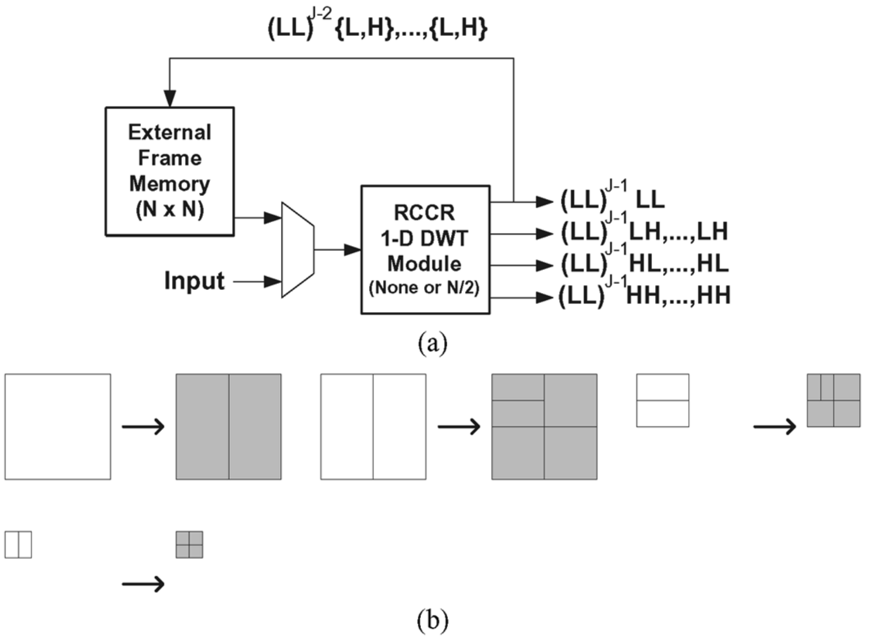


Fig. 2. RCCR 2-D implementation (a) System architecture. (b) Data flow of external memory access ($J = 3$).

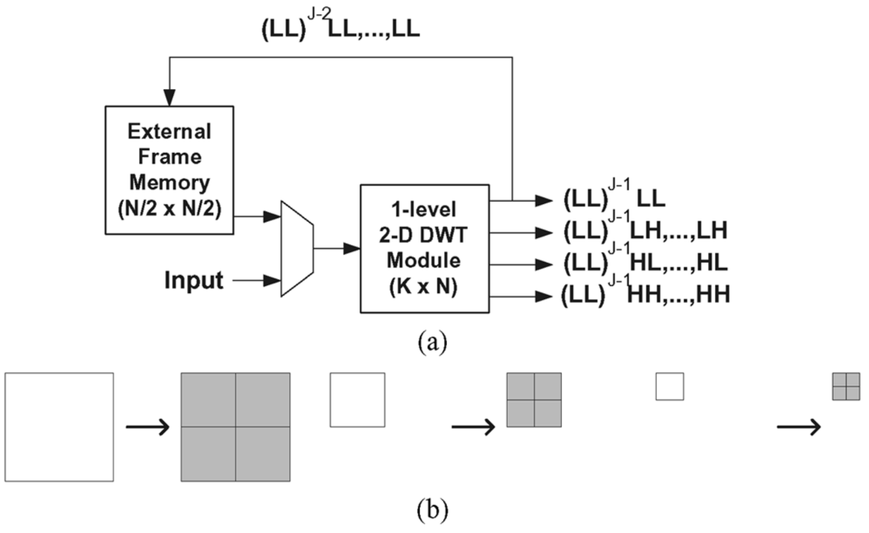


Fig. 3. 1-level line-based implementation. (a) System architecture. (b) Data flow of external memory access ($J = 3$).

C. 1-Level Line-Based Architecture

Unlike the direction-by-direction approach of direct and RCCR architectures, each level of the DWT decomposition can be performed at a time, and the multi-level decompositions can be achieved by using the level-by-level approach. However, this approach may require some internal memory, whose size is proportional to the image width, to store the intermediate DWT coefficients of one direction and to supply the input signals for the DWT decomposition in the other direction [10]. The proposed architectures of [7] and [13] are based on this approach as shown in Fig. 3(a). The size of the required internal memory, called line buffer is KN, where K represents how many line buffers are used and depends on both the adopted 1-D DWT architecture and the implementation method of the line-based architecture. Fig. 3(b) shows the data flow of external memory access, and the external memory bandwidth can be expressed by

$$2 \times \left(1 + \frac{1}{4} + \frac{1}{16} + \cdots + \left(\frac{1}{4}\right)^{J-1}\right) N^2 \quad \text{(words/image)}. \tag{4}$$

The external memory bandwidth of this 1-level line-based architecture is exactly one half of that of the direct architecture. This is due to the utilization of internal line buffers which store exactly one half of the intermediate DWT coefficients. Furthermore, unlike the direct architecture that uses the whole frame buffer of size N^2 as the intermediate coefficient buffer, the 1-level line-based architecture only uses one-quarter of the frame buffer.

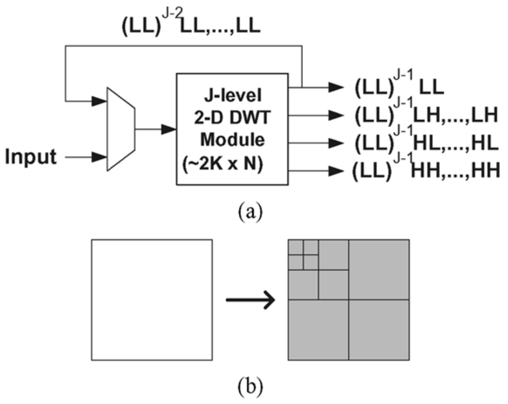


Fig. 4. Multi-level line-based implementation. (a) System architecture. (b) Data flow of external memory access ($J = 3$).

TABLE I
SUMMARY OF RAM-BASED 2-D ARCHITECTURES ($J \to \infty$)

Architecture	External Memory Access (words/image)	Line Buffer (words)	Intermediate Frame Buffer (words)	Control Complexity	System Integration
Direct	$5.33N^2$	-	N^2	Simple	Difficult
RCCR (RPA)	$4.67N^2$	-	N^2	Medium	Difficult
RCCR (N/2)	$4.67N^2$	0.5N	N^2	Simple	Difficult
1-level	$2.67N^2$	KN	$N^2/4$	Medium	Medium
Multi-level	$2N^2$	2KN	-	Complex	Simple

D. Multi-Level Line-Based Architecture

Beyond the level-by-level approach, we can perform all of the decomposition levels simultaneously as the multi-level 2-D architecture in Fig. 4(a). However, using cascaded J 1-level line-based architectures to implement directly will result in very low hardware utilization. Generally, the tasks of all decomposition levels will be allocated to a few 1-D DWT modules [5], [4] and the size of required internal line buffers is as follows:

$$\left(1+\frac{1}{2}+\frac{1}{4}+\cdots+\left(\frac{1}{2}\right)^{J-1}\right)KN \quad \text{(words)} \qquad (5)$$

where K is the number of line buffers if this kind of architecture is degenerated into the 1-level line-based architecture.

As described above, the data flow of external memory access is simple and regular as shown in Fig. 4(b). Although this multi-level 2-D architecture requires more internal buffer and suitable task assignments for 1-D DWT modules, it can reduce the external memory access bandwidth to the minimum $2N^2$.

E. Summary

The above presented architectures of 2-D DWT are summarized in Table I. The issue of system integration is mainly influenced by the bandwidth requirement of external memory, whereas the control complexity indicates how difficult it is for the control circuits with proper data flow to be implemented. Based on Table I, the multi-level line-based architecture requires the most hardware cost, including the internal line buffer, multiple 1-D DWT modules, and complex control circuits. However, it requires the least external memory bandwidth without using the external frame buffer to store intermediate data. Thus, it will be the preferred candidate if the external memory bandwidth is very limited or if no external frame buffer can be used. Furthermore, it should be noted that the access of external memory consumes the most power in the hardware implementation of 2-D DWT [9].

Table I shows the tradeoff of these 2-D DWT architectures exactly. The simplest direct architecture has the least hardware cost but requires the most external memory bandwidth. The RCCR architecture can decrease the external memory bandwidth of the direct architecture by using one small line buffer or adopting RPA scheduling. The 1-level line-based architecture uses a few line buffers to reduce a great deal of the external memory bandwidth. The minimum of external memory bandwidth can be achieved by using multi-level line-based architecture, which requires more line buffers and very complex control circuits.

The described comparisons among these architectures are independent of the adopted 1-D DWT modules and the implementation details of 2-D architectures. Those important characteristics, such as throughput, hardware utilization, and the exact size of line buffers, are related to the implementation details and will be discussed later.

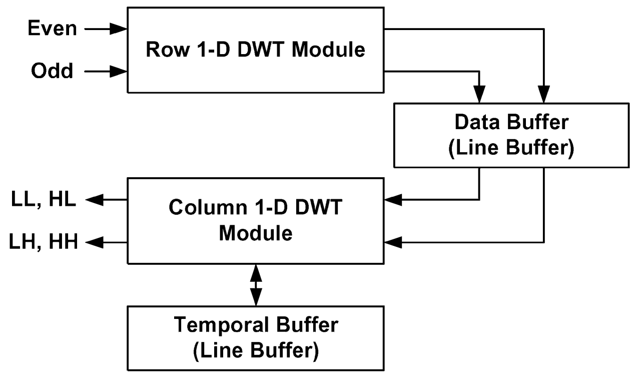


Fig. 5. Proposed generic architecture for 1-level line-based method.

III. PROPOSED LINE-BASED ARCHITECTURES

In the previously mentioned 2-D DWT architectures, direct and RCCR architectures can be implemented very trivially. In this section, generic line-based architectures for 2-D DWT will be proposed for both 1-level and multi-level methods. In addition to convolution-based 1-D DWT architectures, lifting-based ones can be easily integrated into the proposed 2-D DWT architectures to construct an optimal hardware implementation. For supporting the lifting-based DWT architectures, the throughput of the adopted 1-D DWT module is assumed to be two-input/two-output per clock cycle in the following. It is also assumed that the image pixels are inputted in a raster scan order.

A. Proposed 1-Level Line-Based Architecture

As shown in Fig. 5, the proposed 1-level line-based architecture for 2-D DWT is basically composed of some line buffers and two 1-D DWT modules, one for row-wise decomposition and the other one for column-wise decomposition. These line buffers can be classified into two categories: data buffer and temporal buffer.

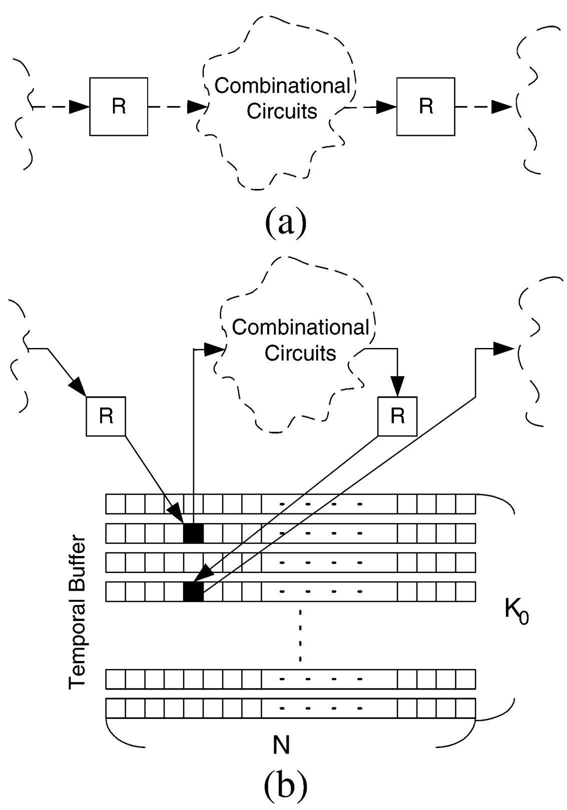


Fig. 6. (a) Original circuits. (b) Modified line-based circuits (R: register).

The data buffer is used to store the intermediate decomposition coefficients after 1-D DWT in the row direction and to provide the input signals of the column 1-D DWT module. Thus, the data buffer is independent of 1-D DWT architectures and only related to the throughput. On the other hand, the temporal buffer is highly dependent on the adopted 1-D DWT architecture. If the original circuits in the adopted 1-D DWT architecture are as constructed as Fig. 6(a), the column 1-D DWT module and the temporal buffer need to be modified as Fig. 6(b). That is, use the temporal buffer to store the data, which should originally be stored in the registers, for the column 1-D DWT module. Because the memory access of temporal buffer is very regular and simple, K_0 copies of two-port RAMs of size N can be used to implement it, where K_0 is the number of registers in the adopted 1-D DWT module. Moreover, these K_0 RAMs can be merged into a single RAM of size K_0N for higher density. The memory address of these temporal buffers can be generated easily by a rotation-like address calculator. As for the data buffer, the lower bound of its size is $1N$ because the data flow always keeps two-input/two-output per clock cycle after the 1-D DWT coefficients of the first row are obtained. However, extremely high complexity will be required with this minimal data buffer size. In the following, we will present the feasible data flows for data buffer of sizes $1.5N$ and $(1+(1/2)^k)N$ by using two-port RAM, followed by a numerical analysis of the realistic data buffer size in the aspect of implementation. It is possible to eliminate two multipliers if the adopted 1-D DWT module is lifting-based, as discussed at the end of this section.

1) Data Flow for Data Buffer of Size $1.5N$*:* Since the adopted 1-D DWT module is defined as two-input/two-output per clock cycle, the column DWT module requires that two lines of signals are inputted simultaneously. However, the row DWT module gives the output signals in a line-by-line way. The first line of these two lines of signals which are immediately required by the column DWT module is defined as LINE_1, and the second line is defined as LINE_2. The data flow of these signals should be designed carefully by using the data buffer and its address generator. Contrary to [14] where first-in first-out (FIFO) is used, the proposed data flow uses feasible two-port RAM, which can provide high regularity and density.

The proposed data flow is described as follows. As shown in Fig. 7, one RAM, called RAM_A, of size N is used for storing the coefficients of LINE_1, and the other one, called RAM_B, of size $0.5N$ is for LINE_2. After the LINE_1 coefficients fill up RAM_A, as Fig. 7(a), the data flow can be performed as the following steps. From Fig. 7(a)–(d), the LINE_2 coefficients enter RAM_B while the column 1-D DWT module consumes data from both RAM_A and RAM_B. Fig. 7(b)–(d) represent that $1/2, 3/4$, and all of the LINE_2 coefficients are filled into RAM_B, respectively. Then the next LINE_1 coefficients feed RAM_A from Fig. 7(d)–(e), and the data flow continues from Fig. 7(b), recursively. In summary, the data flow is illustrated by Fig. 7 in the order of (a) $\rightarrow$ (b) $\rightarrow$ (c) $\rightarrow$ (d) $\rightarrow$ (e) $\rightarrow$ (b) $\rightarrow$ (c)...

This data flow is without conflict because the speed of writing data is double that of reading data for each line buffer. The above steps illustrate only the main idea of the data flow. In fact, RAM_A and RAM_B should both be composed of two two-port RAMs, one for low-pass signals and the other for high-pass signals. Nonetheless, the data flow can work well after this modification.

2) Complete Data Flow for Data Buffer of Size $(1+(1/2)^k)N$*:* In this subsection, we present how to be close to the lower bound of the data buffer. For this data flow, 2^k+1 split RAMs of size $N/2^k$ are required, where k is an arbitrary integer. At first, the LINE_1 coefficients are filled into 2^k split RAMs. Then the data flow can be recursively performed for every two new lines of coefficients as follows. The first $1/2^{k-1}$ of the entering LINE_2 coefficients are filled into the left split RAM as the RAM_B from Fig. 7(b)–(d). Then the following steps are performed recursively until the next LINE_1 coefficients are filled into 2^k split RAMs. In the beginning, take the two split RAMs, in which the stored data are required by the column 1-D DWT module immediately, as RAM_C and RAM_D while the empty split RAM is called RAM_E, as shown in Fig. 8. Then the data can be transferred from Fig. 8(a)–(d). Thus, another empty split RAM is obtained as RAM_D, and the entering coefficients are stored in RAM_C and RAM_E. The data flow is summarized in Fig. 9.

Although the lower bound can be approached very closely, the additional control circuits and complex address generators may become impractical if k is too large. Moreover, some additional storage devices for controlling data flow would be inevitable, and the size of these storage devices will become larger as k increases. In addition to the data flows mentioned above, no control circuits will be required if a data buffer of size $3N$ is allowable. In [11], six two-port RAMs, three for low-pass signals and three for high-pass signals, and a rotation-like address generator are used to perform the data transfer between the row and column 1-D DWT modules.

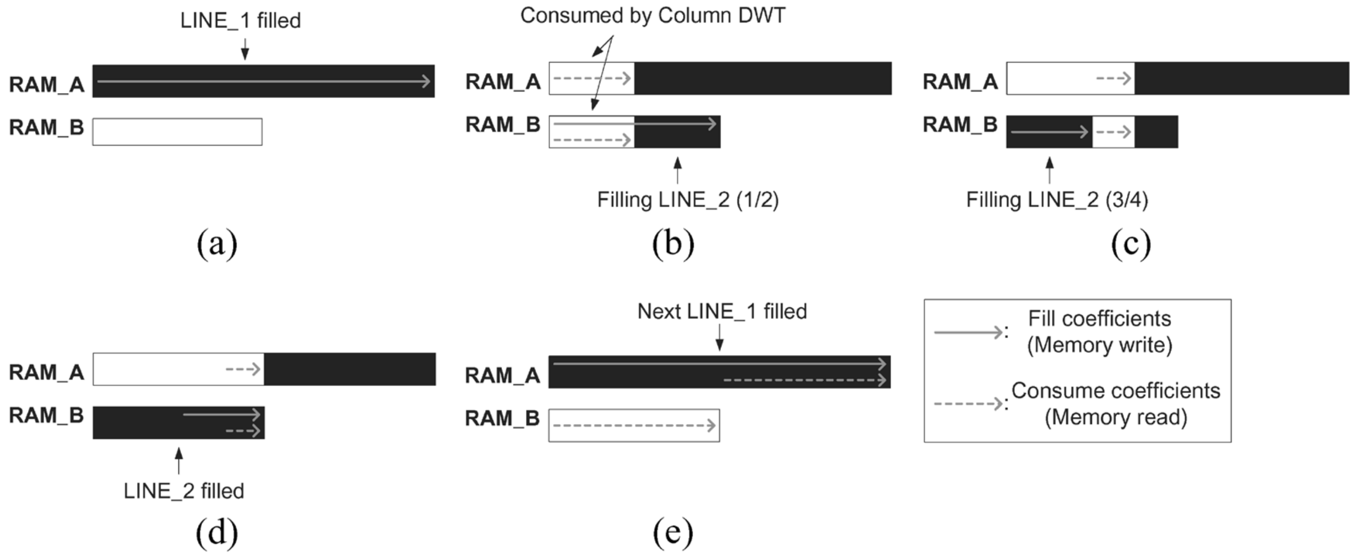


Fig. 7. Data flow for data buffer of size $1.5N$ (black: data filled; white: no data).

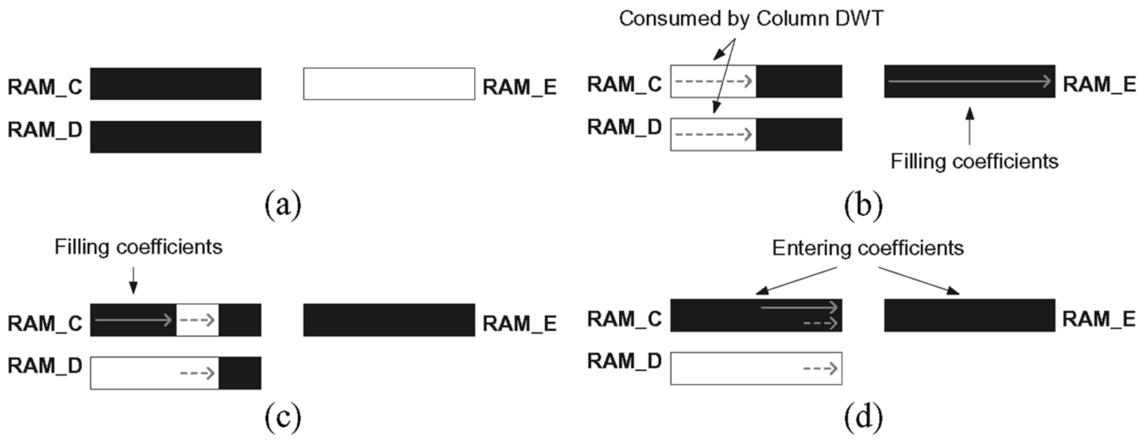


Fig. 8. Data transfer among the three working split RAMs for the data buffer of size $(1 + (1/2)^k)N$.

3) Numerical Analysis of the Data Buffer Size: In the above discussion of data flows, the memory access is assumed as requiring no delay cycles, and thus the minimum of data buffer, $1N$, can be approached. However, this ideal data flow cannot be practically implemented because the memory access always needs some delay cycles in the real-life implementation. To discover a practical size for the data buffer, the memory access delay cycle should be defined first as d. Thus, the consecutive memory accesses of the same memory address must be separated by more than $d + 1$ cycles.

The conflict of memory write and read in the same address will happen in Fig. 7(d) because the memory writes and reads of RAM_B are overlapped and the write speed is double the read speed. The realistic memory accesses of the split RAMs as RAM_B from Fig. 7(b)–(d) should be as shown in Fig. 10, where the numbers represent the operating cycle numbers and $N_k = N/2^{k+1}$. In this figure, only one half of the split RAM is shown, where $e = 0$ is for the one for low-pass signals and $e = 1$ is for high-pass signals. Thus, the memory write is performed in every cycle, and the memory reading is performed in every other cycle. Since the write speed is double the read speed, some conflict will occur in the writing cycle number, $N_k + n$. This conflict is because the distance between the writing and reading cycles on the same memory address is less than $d + 1$, and can be expressed as the following equation:

$$\begin{aligned}(N_k + n) - (d + 1 + e + 2n) &= d \\ \Rightarrow N_k - n &= 2d + e + 1\end{aligned} \tag{6}$$

and this means that every split RAM, which consists of two parts for low-pass and high-pass signals, respectively, requires additional storage of size $4d + 3(=2d + 1 + 2d + 2)$ to prevent the conflict. So, the real number of total storage size S should be

$$S = (2^k + 1)\left(\frac{N}{2^k} + 4d + 3\right), \qquad \text{when } k > 1. \tag{7}$$

Thus

$$S_{\min} = N + 4d + 3 + 2\sqrt{N(4d + 3)}$$

while

$$k = \log_2 \sqrt{\frac{N}{4d + 3}}. \tag{8}$$

According to these equations, if the image size becomes larger, k will also be required to be larger for less data buffer size.

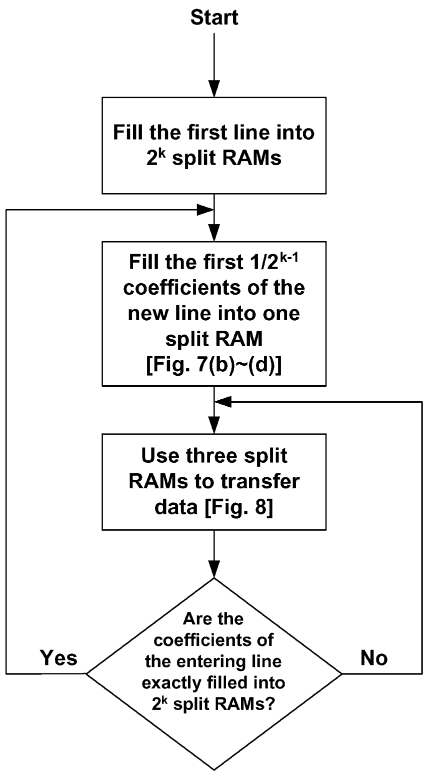

Start
Fill the first line into 2^k split RAMs
Fill the first $1/2^{k-1}$ coefficients of the new line into one split RAM [Fig. 7(b)~(d)]
Use three split RAMs to transfer data [Fig. 8]
Are the coefficients of the entering line exactly filled into 2^k split RAMs?
Yes
No

Fig. 9. Data flow for data buffer of $(1 + (1/2)^k)N$.

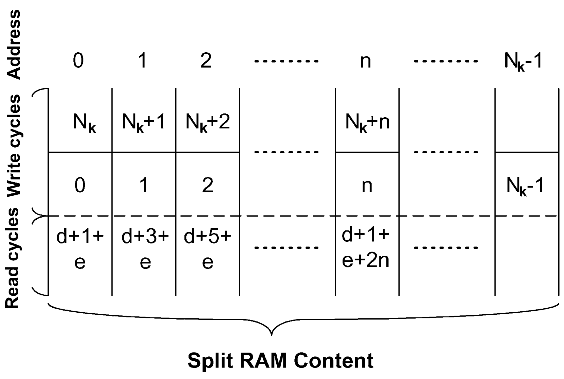


Fig. 10. Memory access of the split RAM ($e = 0$ for low-pass signals; $e = 1$ for high-pass signals).

This means that more partitions of split RAMs are beneficial for reducing the total memory size.

To give more practical comparisons, the method of Section III-A.1 and the realistic minimum $S_{\min}$ are tested for four different conditions. Table II shows the comparison results. The realistic data buffer size of the method in Section III-A.1 is $1.5N + 2(4d + 3)$. In this table, the improvement of the minimum buffer size is higher as the image size becomes larger. The control complexity required for the minimum buffer size is higher than that for the method of Section III-A.1. Since no additional storage devices for each split RAM will be used at the same time, two two-port RAMs, for low-pass and high-pass signals, respectively, afford to serve as all additional storage devices. Thus, the minimum buffer size can be achieved if the address controls are well managed.

TABLE II
COMPARISON OF REALISTIC DATA BUFFER SIZES FOR METHODS IN SECTION III-A.1 AND THE MINIMUM CASE

Condition	$S_{k=1}$ (words)	$S_{\min}$ (words)	k ($S_{\min}$)	$(S_{k=1}-S_{\min})/S_{k=1}$
d=1; N=256	398	351	3	11.8%
d=1; N=2048	3086	2295	4	25.6%
d=2; N=256	406	375	2	7.6%
d=2; N=2048	3094	2363	3	23.6%

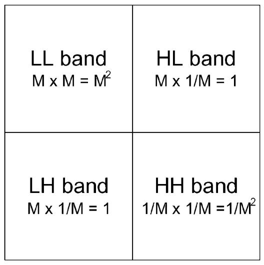


Fig. 11. Normalization step of 2-D lifting-based DWT.

4) Integration of Lifting-Based DWT Module: If lifting-based 1-D DWT module is adopted, the number of multipliers can be further reduced by two because of the scaling effect of the normalization step in Fig. 11, where M is the normalization coefficient. Originally, two multipliers are required for both the row and column 1-D DWT modules. However, the normalization multipliers can be taken out of the two 1-D DWT modules, and instead, two multipliers, M^2 and $1/M^2$, are used to implement the normalization step at the output of the column 1-D DWT module.

B. Proposed Multi-Level Line-Based Architectures

The concepts about data buffer and temporal buffer presented above can be used to construct multi-level architectures with allocating the multilevel DWT decomposition tasks to a few 1-D DWT modules. In this subsection, two multi-level line-based architectures are proposed by using two and three 1-D DWT modules, and the throughputs are one-input/one-output and two-input/two-output per clock cycle, respectively.

1) Adopting Two 1-D DWT Modules (2DWTM): The proposed 1-level architecture can be easily extended to the multi-level architecture by scheduling the multi-level DWT decomposition tasks to the two 1-D DWT modules with RPA [12], as shown in Fig. 12. The RPA schedule for three-level 2-D DWT is shown as an example in Fig. 13, where the numbers represent which level of DWT decomposition should be performed. The throughput of this multi-level architecture is one-input/one-output per clock cycle in average, which is one half that of the 1-level architecture. The number of registers in the row DWT module should be increased to J times the original number in order to store the intermediate data of J level DWT decompositions. Because the amount of data in the next level is only one quarter of the current level, the hardware utilization is only

$$\frac{1}{2} \times \left(1 + \frac{1}{4} + \frac{1}{16} + \cdots + \left(\frac{1}{4}\right)^{J-1}\right) \tag{9}$$

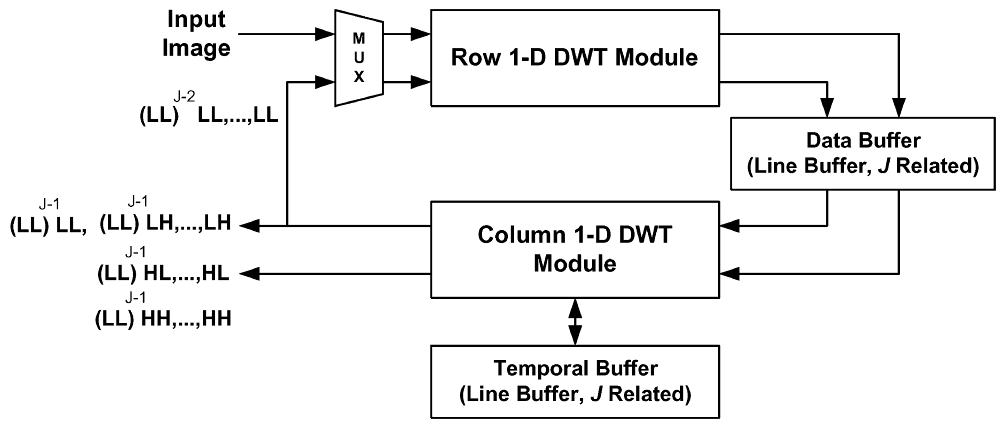


Fig. 12. Proposed generic architecture for multi-level 2-D DWT using 2DWTM.

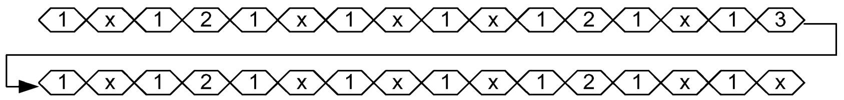

1 x 1 2 1 x 1 x 1 x 1 2 1 x 1 3
1 x 1 2 1 x 1 x 1 x 1 2 1 x 1 x

Fig. 13. RPA schedule for 3-level 2-D DWT. The numbers represent which level of decomposition is performed, and the "×" means a bubble cycle.

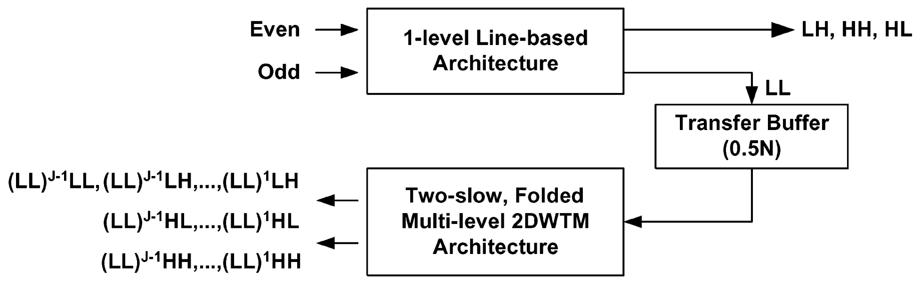

Even
Odd
1-level Line-based Architecture
LH, HH, HL
LL
Transfer Buffer (0.5N)
(LL)^{J-1}LL,(LL)^{J-1}LH,...,(LL)^1LH
(LL)^{J-1}HL,...,(LL)^1HL
(LL)^{J-1}HH,...,(LL)^1HH
Two-slow, Folded Multi-level 2DWTM Architecture

Fig. 14. Proposed generic architecture for multi-level 2-D DWT using 3DWTM.

and the limit is $2/3$ as J is indefinitely large. The total size of data buffer and temporal buffer becomes

$$\left(1+\frac{1}{2}+\frac{1}{4}+\cdots+\left(\frac{1}{2}\right)^{J-1}\right)(K_0+K_1)N \quad \text{(words)} \tag{10}$$

where K_0 and K_1 are the numbers of line buffers for the temporal buffer and data buffer, respectively, in the original 1-level line-based architecture.

2) Adopting Three 1-D DWT Modules (3DWTM): The throughput of the 2DWTM architecture is one-input/one-output because of only using two 1-D DWT modules. Moreover, the hardware utilization is not high due to the imbalanced task allocation with RPA scheduling. The problem of the above RPA scheduling is allocating only one half of the hardware resources to perform the first level decomposition in which the arithmetic operations are more than the three-fourths of the total operations.

To increase the hardware utilization, we propose to allocate the DWT decomposition tasks to three 1-D DWT modules of which two are for the first level and one is for all higher levels. Fig. 14 shows the proposed multi-level line-based architecture using 3DWTM. This architecture consists of one proposed 1-level architecture, one two-slow and folded 2DWTM architecture, and a transfer buffer of size $0.5N$. The two-slow multi-level 2DWTM architecture folds the row and column DWT modules into the same module and serves for DWT decompositions of all levels, except the first level. The transfer buffer is used to store every line of signals in the LL band from the 1-level architecture and to provide the input signals to the two-slow 2DWTM architecture. The throughput can achieve two-input/two-output again, and the hardware utilization is increased to

$$\left(2+\frac{1}{2}\times\left(1+\frac{1}{4}+\frac{1}{16}+\cdots+\left(\frac{1}{4}\right)^{J-2}\right)\right)\Big/3 \tag{11}$$

and the limit is $8/9$ as J is indefinitely large. Furthermore, the total size of data buffer and temporal buffer becomes

$$\left(\left(1+\frac{1}{2}+\frac{1}{4}+\cdots+\left(\frac{1}{2}\right)^{J-1}\right)\times(K_0+K_1)+\frac{1}{2}\right)N \quad \text{(words)}. \tag{12}$$

Compared to (9) and (10), the hardware utilization is increased by about four-thirds, but the total buffer size should be increased by $0.5N$ for the transfer buffer.

TABLE III
COMPARISON OF 2-D ARCHITECTURES FOR THE GENERAL CASE. ASSUME $J \rightarrow \infty$ AND THE 1-D DWT MODULE IS CONVOLUTION-BASED. THE DATA BUFFER OF THE PROPOSED ARCHITECTURES IS ASSUMED TO BE $1.5N$ ONLY FOR COMPARISONS, AND IT CAN BE FURTHER REDUCED TO ABOUT $1N$. (THE UNIT OF LINE BUFFER SIZE IS WORD)

Architecture	Multipliers	Adders	Line Buffer Size	Computing Time	Control Complexity	Hardware Utilization	Through-put
1-level 2-D	4L	4L	LN	$0.67N^2$	Simple	100%	2
Proposed 1-level	4L	4L	(L-1/2)N	$0.67N^2$	Simple	100%	2
Systolic-Parallel	4L	4L	(2L+4)N	N^2	Complex	66.7%	1
Parallel-Parallel	4L	4L	(2L+1)N	N^2	Complex	66.7%	1
Proposed 2DWTM	4L	4L	(2L-1)N	N^2	Complex	66.7%	1
Proposed 3DWTM	6L	6L	(2L-1/2)N	$0.5N^2$	Complex	88.9%	2

IV. COMPARISON

In this section, we show the efficiency of the proposed architectures by presenting comparison results with different 2-D DWT line-based architectures. The following comparisons consist of three topics, including the general case, adopting JPEG 2000 default lossy (9, 7) filter, and adopting JPEG 2000 default lossless (5, 3) filter. The parameter K_1 of the proposed architectures is assumed to be 1.5 for comparison because the minimum data buffer size depends on the image size and the memory access delay time.

A. General Case

The comparison results of systolic-parallel [5], parallel-parallel [4], 1-level 2-D [7], and the proposed architectures are given in Table III, where L represents the filter length, and J is assumed to be infinitely large. In this table, only general convolution-based 1-D DWT modules are considered, so the required numbers of multipliers and adders of these architectures are all proportional to the number of adopted 1-D DWT modules.

Among these architectures, only the proposed 1-level line-based and the 1-level 2-D architectures are classified in Section II-C, whereas the others belong to the Section II-D. The hardware utilization of the two 1-level architectures can reach 100%. The product of hardware utilization, computing time, and the number of adopted 1-D DWT modules, should be a constant which is equal to the total decomposition operations. Thus, raising hardware utilization can lower the computing time if the numbers of 1-D DWT modules are the same, as described in [7]. In addition, the proposed 1-level line-based architecture outperforms the 1-level 2-D one in the aspect of the number of line buffers due to careful management of the data buffer.

As for the multi-level line-based architectures, the sizes of line buffers in the proposed architectures are all smaller than those of the systolic-parallel and parallel-parallel architectures. Moreover, the proposed architecture using three 1-D DWT modules can increase the hardware utilization and throughput so as to decrease the computing time to one half of other multi-level architectures.

B. JPEG 2000 Default Lossy (9, 7) Filter

The previous architectures for 2-D DWT in the literature are primarily based on the convolution-based DWT architectures. Only the simplest lifting scheme, the JPEG 2000 default lossless (5, 3) filter, is considered in [14]. However, the proposed architectures can adopt any kind of lifting-based DWT architecture to obtain the optimal hardware implementation. The popular JPEG 2000 default lossy (9, 7) filter is used as the target 1-D DWT module for comparison because of its good time-frequency decomposition. For clarity, only the multi-level line-based architectures, which adopt the same number of 1-D DWT modules, are considered. The comparison of the proposed architecture using two 1-D DWT modules, systolic-parallel architecture, and parallel-parallel architecture is given in Table IV. The lifting-based and the convolution-based DWT modules, which adopt the symmetric property of this linear filter, are from [15].

TABLE IV
COMPARISON OF MULTILEVEL 2-D ARCHITECTURES FOR (9, 7) FILTER ($J \rightarrow \infty$). THE DATA BUFFER OF THE PROPOSED ARCHITECTURES IS ASSUMED TO BE $1.5N$ FOR COMPARISONS. (THE UNIT OF LINE BUFFER SIZE IS WORD)

Architecture (2-D)	Multiplier	Adder	Line Buffer	DWT module
Systolic-Parallel	20	36	22N	Convolution
Parallel-Parallel	20	36	19N	Convolution
Proposed 2DWTM	18	28	17N	Convolution
Proposed 2DWTM	10	16	11N	Lifting Scheme

The systolic and parallel filters also can utilize the symmetric property. Although the proposed architecture adopting convolution-based module has nearly the same number of multipliers and adders as those of systolic-parallel and parallel-parallel architectures, the details of the implementation of 1-D DWT modules are extremely different. The multipliers of the proposed architectures are all fixed coefficients, whereas those of the other two all require programmable coefficients. Thus, the hardware cost of the proposed architecture is much smaller than those of the other two architectures. In addition, the proposed architecture outperforms those two architectures in terms of the number of line buffers.

Furthermore, the proposed architecture adopting lifting-based 1-D DWT module outperforms other architectures in all aspects. This outstanding performance comes from the adoption of lifting-based modules, which not only decreases the number of multipliers and adders but also greatly reduces the size of the temporal buffer.

C. JPEG 2000 Default Lossless (5, 3) Filter

The above comparison shows that the ability to adopt lifting-based modules can greatly reduce the size of the temporal buffer.

TABLE V
COMPARISON OF 2-D ARCHITECTURES FOR (5, 3) FILTER. (a) MULTILEVEL ARCHITECTURES ($J \to \infty$). (b) 1-LEVEL ARCHITECTURES. THE DATA BUFFER OF THE PROPOSED ARCHITECTURES IS ASSUMED TO BE $1.5N$ FOR COMPARISONS. (THE UNIT OF LINE BUFFER SIZE IS WORD)

Architecture (2-D)	Line Buffer	DWT module
Systolic-Parallel	14N	Convolution
Parallel-Parallel	11N	Convolution
Proposed 2DWTM	9N	Convolution
Proposed 2DWTM	7N	Lifting Scheme

(a)

Architecture (2-D)	Line Buffer	DWT module
1-level 2-D	5N	Convolution
Proposed 1-level	4.5N	Convolution
Proposed 1-level	3.5N	Lifting Scheme

(b)

However, we will also show the benefit of the proposed data flow of data buffers over previous designs. The (5, 3) filter, whose filter length is very short, is chosen for comparison because the ratio of data buffer size to temporal buffer size is higher than that in the case of adopting longer tap filters.

The multi-level and 1-level architectures are listed in Table V(a) and (b), respectively. Only the line buffer size are given since the objective is to compare the sizes of line buffers, and moreover, the (5, 3) filter can be implemented without multipliers. Clearly, the proposed architectures outperform the previous designs, especially when the lifting-based module is adopted.

In the above comparisons, the size of the data buffer in the proposed architectures is assumed as $1.5N$. If the minimal data buffer size is adopted, the advantage of the proposed architectures will be more significant.

V. CONCLUSION

In this paper, we have proposed three generic RAM-based architectures of high efficiency and feasibility for both 1-level and multi-level line-based 2-D DWT architectures. The carefully designed data flow of data buffer and the modified method of temporal buffer in the proposed architectures can provide a variety of advantages, including easy control circuits, real-life implementation, and minimal internal memory size. Both conventional convolution-based and advanced lifting-based 1-D DWT modules can be flexibly integrated into the proposed architectures to obtain the optimal hardware design, whereas most previous studies can only support the former. The outstanding performance of the proposed architectures is demonstrated by the comparison results of the general case as well as adopting JPEG 2000 default (9, 7) and (5, 3) filters.

REFERENCES

[1] S. G. Mallat, “A theory for multiresolution signal decomposition: The wavelet representation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 7, pp. 674–693, Jul. 1989.

[2] W. Sweldens, “The lifting scheme: A custom-design construction of biorthogonal wavelets,” *Appl. Comput. Harmonic Anal.*, vol. 3, no. 15, pp. 186–200, 1996.

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

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

[5] M. Vishwanath, R. M. Owens, and M. J. Irwin, “VLSI architectures for the discrete wavelet transform,” *IEEE Trans. Circuis Systems II, Analog Digit. Signal Process.*, vol. 42, no. 5, pp. 305–316, May 1995.

[6] 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 Process.*, vol. 43, no. 3, pp. 759–771, Mar. 1995.

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

[8] M. Weeks and M. Bayoumi, “Discrete wavelet transform: Architectures, design and performance issues,” *J. VLSI Signal Process. Syst.*, vol. 35, no. 2, pp. 155–178, 2003.

[9] N. D. Zervas, G. P. Anagnostopoulos, V. Spiliotopoulos, Y. Andreopoulos, and C. E. Goutis, “Evaluation of design alternatives for the 2-D-discrete wavelet transform,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 12, pp. 1246–1262, Dec. 2001.

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

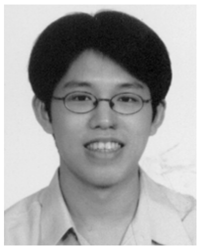
[11] P.-C. Tseng, C.-T. Huang, and L.-G. Chen, “VLSI implementation of shape-adaptive discrete wavelet transform,” in *Proc. SPIE Int. Conf. Visual Communications and Image Processing*, 2002, pp. 655–666.

[12] M. Vishwanath, “The recursive pyramid algorithm for the discrete wavelet transform,” *IEEE Trans. Signal Process.*, vol. 42, no. 3, pp. 673–677, Mar. 1994.

[13] 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 and Systems*, 2002, pp. 363–366.

[14] C. Diou, L. Torres, and M. Robert, “A wavelet core for video processing,” in *Proc. Int. Conf. Image Processing*, 2000, pp. 395–398.

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



Chao-Tsung Huang was born in Kaohsiung, Taiwan, R.O.C., in 1979. He received the B.S. degree from the Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan, R.O.C., in 2001. He is currently working toward the Ph.D. degree at the Graduate Institute of Electronics Engineering, National Taiwan University.

His major research interests include VLSI design and implementation for 1-D, 2-D, and 3-D discrete wavelet transform.

Po-Chih Tseng was born in Tao-Yuan, Taiwan, R.O.C., in 1977. He received the B.S. degree in electrical and control engineering from National Chiao Tung University, Hsinchu, Taiwan, R.O.C., in 1999 and the M.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, R.O.C., in 2001. He is currently working toward 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, R.O.C. During 1993–1994, he was a Visiting Consultant in the DSP Research Department, AT&T Bell Laboratories, Murray Hill, NJ. In 1997, he was a Visiting Scholar in the Department of Electrical Engineering, University of Washington, Seattle. Currently, he is a Professor at National Taiwan University. His current research interests are DSP architecture design, video processor design, and video coding systems.

Dr. Chen has served as an Associate Editor of IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY since 1996, as Associate Editor of the IEEE TRANSACTIONS ON VLSI SYSTEMS since 1999, and as Associate Editor of IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS PART II: EXPRESS BRIEFS since 2000. He has been the Associate Editor of the *Journal of Circuits, Systems, and Signal Processing* since 1999, and a Guest Editor for the *Journal of Video Signal Processing Systems*. He is also the Associate Editor of the PROCEEDINGS OF THE IEEE. He was the General Chairman of the 7th VLSI Design/CAD Symposium in 1995 and of the 1999 IEEE Workshop on Signal Processing Systems: Design and Implementation. He is the Past-Chair of Taipei Chapter of IEEE Circuits and Systems (CAS) Society, and is a Member of the IEEE CAS Technical Committee of VLSI Systems and Applications, the Technical Committee of Visual Signal Processing and Communications, and the IEEE Signal Processing Technical Committee of Design and Implementation of SP Systems. He is the Chair-Elect of the IEEE CAS Technical Committee on Multimedia Systems and Applications. During 2001–2002, he served as a Distinguished Lecturer of the IEEE CAS Society. He received Best Paper Awards 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 Engineers Society. In 1996 and 2000, he received the Outstanding Research Award from the National Science Council, and in 2000, the Dragon Excellence Award from Acer. He is a Member of Phi Tan Phi.