New Memory-Efficient Hardware Architecture of 2-D Dual-mode Lifting-Based Discrete Wavelet Transform for JPEG2000

Chih-Hsien Hsia
Dept. of Electrical Engineering
Tamkang University
Taiwan, ROC
E-mail: chhsia@ee.tku.edu.tw

Jen-Shiun Chiang, *Member*, *IEEE*
Dept. of Electrical Engineering
Tamkang University
Taiwan, ROC
E-mail: chiang@ee.tku.edu.tw

Abstract— **This work presents new algorithms and hardware architectures to improve the critical issues of the 2-D dual-mode (supporting 5/3 lossless and 9/7 lossy coding) lifting-based discrete wavelet transform (LDWT). The proposed 2-D dual-mode LDWT architecture has the advantages of low-transpose memory, low latency, and regular signal flow, which is suitable for VLSI implementation. The transpose memory requirement of the *N*×*N* 2-D 5/3 mode LDWT is 2*N*, and that of 2-D 9/7 mode LDWT is 4*N*. According to the comparison results, the proposed hardware architecture surpasses previous architectures in the aspects of lifting-based low-transpose memory size. It can be applied to real-time visual operations such as JPEG2000, MPEG-4 still texture object decoding, and wavelet-based scalable video coding.**

Keywords-lifting-based discrete wavelet transform (LDWT), interlaced read scan algorithm (IRSA), low-transpose memory, 2-D 5/3 mode LDWT, 2-D 9/7 mode LDWT.

I. INTRODUCTION

Among the variety of DWT algorithms, the LDWT provides a new approach for constructing biorthogonal wavelet transforms, and it also provides a more efficient scheme for calculating classical wavelet transforms [1]-[15]. By factoring the classical wavelet filter into lifting steps, the computational complexity of the corresponding DWT can be reduced by up to 50%. The lifting steps can be easily implemented, which is different from the direct finite impulse response (FIR) implementations of Mallat's algorithm. Diou *et al*. [1] presented an architecture that performs the LDWT with the 5/3 filter based on the interleaving technique. Andra *et al*. [2] proposed a block-based simple four-processor architecture that compute several stages of the DWT at a time. Chen *et al*. [3] proposed a folded and pipelined architecture for the 2-D LDWT implementation, and this LDWT needs memory size of 2.5*N* for an *N*×*N* 2-D DWT. This lifting architecture for vertical filtering is divided into two parts, and each part consists of one adder and one multiplier. Because both parts are activated in different cycles, they can share the same adder and multiplier to increase the hardware utilization and reduce the latency. However, according to the characteristics of the signal flow it increases the complexity as well. Chen *et al*. [4] proposed a flexible and folded architecture for 3-level 1-D LDWT to achieve higher hardware utilization. Chiang *et al*. [5] proposed a 2-D DWT folded architecture to improve the hardware utilization. Jung *et al*. [6] presents an efficient VLSI architecture of dual-mode LDWT that is used by lossy or lossless compression of JPEG2000. Chen *et al*. [7] used a 1-D folded architecture to improve the hardware utilization for 5/3 and 9/7 filters. The recursive architecture is a general scheme to implement any wavelet filter that is decomposed into lifting steps in smaller hardware complexity. The architecture in [11] implements 2-D DWT only with transpose memory by using recursive pyramid algorithm (PRA). In [10] it has the average of N^2 computing time for all DWT levels. However, it uses many multipliers and adders. Huang *et al*. [13] proposed generic RAM-based architecture of high efficiency and feasibility for 2-D DWT. In [14], the pipelined signal path is regular and practicable. This work presents a high-performance and low-memory architecture to implement the 2-D dual-mode LDWT. Wu *et al*. [15] proposed an efficient VLSI architecture for the direct 2-D LDWT, in which the poly-phase decomposition method and the coefficient folding method are employed to increase the hardware utilization. Despite these efficient improvements of the existed architecture, further improvements in the algorithm and architecture are still needed. Some VLSI architectures of 2-D LDWT reduce the transpose memory requirements and communication between the processors, such as the architectures presented in [1]-[15]. However, these hardware architectures still need large transpose memory.

Low transpose memory requirement is the major concern in space-frequency domain implementation. For an *N*×*N* 2-D LDWT, people like to use raster scan signal flow operation. Under this approach the memory requirement ranges from 2*N* to N^2 [1]-[15] (in 2-D 5/3 and 9/7 modes LDWT). In order to solve the problem of transpose memory access, a high-performance and low-memory architecture for multi-level 2-D DWT is proposed in this paper. This work presents a new architecture to improve the 2-D dual-mode LDWT. In this paper, we revise the signal flow from row-wise to mixed row and column-wise, and further propose a new approach, interlaced read scan algorithm (IRSA), to reduce the transpose memory requirement for a 2-D dual-mode LDWT. In the IRSA approach, for an N×N DWT, a transpose memory size of 2*N* or 4*N* (5/3 or 9/7 mode) are required. Our 2-D LDWT is based on parallel and pipelined schemes to reduce the transpose memory and increase the operating speed. For hardware implementation, we use shifters and adders to replace multipliers in the computation to accomplish high hardware utilization. This 2-D LDWT includes the characteristics of higher hardware utilization, less memory requirement, and regular signal flow. A 256×256 2-D dual-

mode LDWT was designed and simulated by VerilogHDL, and further synthesized by the Synopsys design compiler with TSMC 0.18μm 1P6M CMOS process technology to verify the performance of the proposed hardware architecture.

II. DISCRETE WAVELET TRANSFORM AND LIFTING-BASED METHOD

The lifting-based scheme proposed by Daubechies and Sweldens requires fewer computations than the traditional convolution-based approach. The lifting-based scheme is an efficient implementation for DWT. The lifting-based scheme can easily use integer operations and avoids the problems caused by the finite precision or rounding. A lifting-based scheme has the following four stages:

1) Split phase: The original signal is divided into two disjoint subsets. Significantly, the variable *Xe* denotes the set of even samples and *Xo* denotes the set of odd samples. This phase is called lazy wavelet transform because it does not decorrelate the data, but only subsamples the signal into even and odd samples.

2) Predict phase: The predicting operator P is applied to the subset *Xo* to obtain the wavelet coefficients *d*[n] as in (1).

$$d[n]=Xo[n]+P\times(Xe[n]) \quad (1)$$

3) Update phase: The *Xe*[n] and *d*[n] can be combined to obtain the scaling coefficients *s*[n] after an update operator U as in (2).

$$s[n]=Xe[n]+U\times(d[n]) \quad (2)$$

4) Scaling: In the final step, the normalization factor is applied on *s*[n] and *d*[n] to obtain the wavelet coefficients. For example, Equations (3) and (4) describe the implementation of the 5/3 integer lifting analysis DWT and are used to calculate the odd coefficients (high-pass coefficients) and even coefficients (low-pass coefficients), respectively.

$$d^*[n] = X(2n+1) - \lfloor X(2n) + X(2n+2)/2 \rfloor \quad (3)$$

$$s^*[n] = X(2n) + \lfloor d(2n-1) + d(2n+1) + 2/4 \rfloor \quad (4)$$

Although the lifting-based scheme involves low complexity, the long and irregular signal paths are the major limitations for efficient hardware implementation. Additionally, the increasing number of pipelined registers increases the internal memory size of the 2-D DWT architecture [9]. The 2-D LDWT uses a vertical 1-D LDWT subband decomposition and a horizontal 1-D LDWT subband decomposition to obtain the 2-D LDWT coefficients. Therefore, the memory requirement dominates the hardware cost and complexity of the architectures for 2-D LDWT. The default wavelet filters used in JPEG2000 are the dual-mode LDWT [9]. The lifting-based steps associated with dual-mode wavelets are shown in Figs. 1 and 2, respectively. Assuming that the original signals are infinite in length, we first apply the first stage lifting to perform the DWT.

Figure 1 shows the lifting-based step associated with the wavelet. The original signals including *s*0, *d*0, *s*1, *d*1, *s*2, *d*2,…, are the original input pixel sequences. If the original signals are infinite in length, the first stage lifting is first applied to update the odd index data *s*0, *s*1,…. In (5), the parameters -1/2 and H*i* denote the first stage lifting parameter and outcome, respectively. Equation (5) shows the operation of the 5/3 integer LDWT.

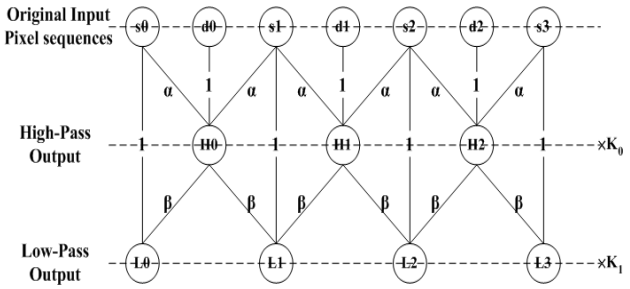


Fig. 1: 5/3 LDWT algorithm.

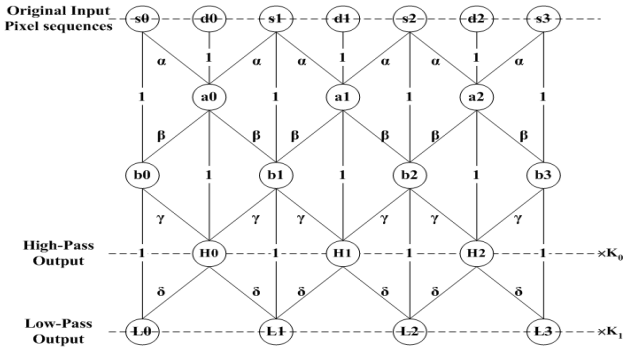


Fig. 2: 9/7 LDWT algorithm.

$$H_i=[(S_i+S_{i+1})\times\alpha+d_i]\times K_0$$
$$L_i=[(H_i+H_{i-1})\times\beta+S_i]\times K_1 \quad (5)$$

where α= -1/2, β= 1/4, and K_0= K_1= 1.

Together with the high-frequency lifting parameter, α, and the input signal we can find the first stage high-frequency wavelet coefficients, H_i. After H_i is found, H_i together with the low-frequency parameter, β, and the input signals of the second stage low-frequency wavelet coefficients, L_i, can be found. The third and fourth stages lifting can be found in a similar manner.

Similar to the 1-level 1-D 5/3 mode LDWT, the calculation of a 1-level 1-D 9/7 mode LDWT is shown in (6).

$$a_i=[(S_i+S_{i+1})\times\alpha+d_i]$$
$$b_i=[(a_i+a_{i-1})\times\beta+S_i]$$
$$H_i=[(b_i+b_{i+1})\times\gamma+a_i]\times K_0$$
$$L_i=[(H_i+H_{i-1})\times\delta+b_i]\times K_1 \quad (6)$$

where α= −1.586134142, β= −0.052980118, γ= +0.882911075, δ= +0.443506852, K_0=1 and K_1= 1.230174104.

The calculation includes four lifting steps and two scaling steps.

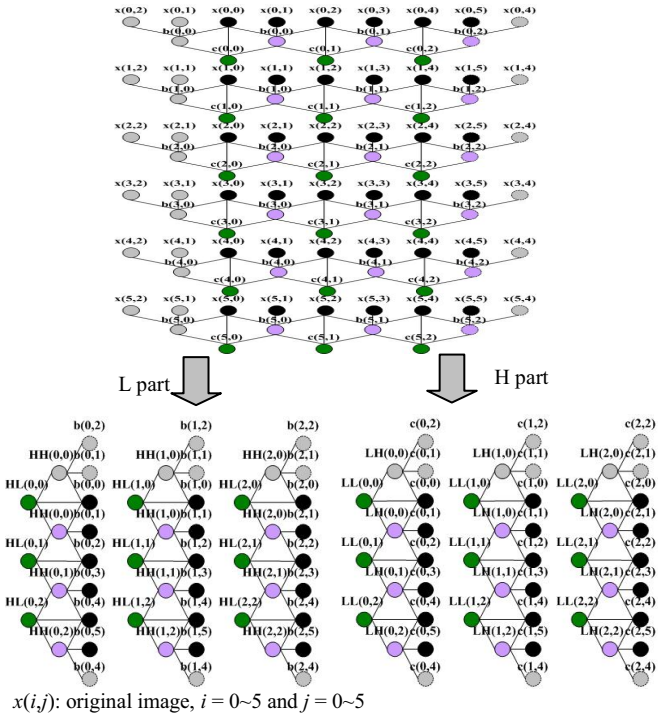
III. INTERLACED READ SCAN ALGORITHM (IRSA)

In recent years, many 2-D LDWT architectures have been proposed to meet the requirements of among on-chip memory for real-time processing. However, the hardware utilization of these architectures needs to be further improved. In DWT implementation, a 1-D DWT needs very massive computation and therefore the computation unit takes most of the hardware cost [1]-[5][7][10][13]-[15]. A 2-D DWT is composed of two 1-D DWTs and a block of transpose memory. In the conventional approach, the size of the transpose memory is equal to the size of the processed image signal. The proposed architecture is described in detail in the following subsections, and we focuses on the 2-D dual-mode LDWT.

Compared to the computation unit, the transpose memory becomes the main overhead in the 2-D DWT. Without loss of generality, the 2-D 5/3 mode LDWT is considered for the description of the 2-D LDWT. If the image dimension is $N \times N$, during the transformation we need a large block of transpose memory (order of N^2) to store the DWT coefficients after the computation of the first stage 1-D DWT decomposition. The second stage 1-D DWT then uses the stored signal to compute the 2-D DWT coefficients of the four subbands [1][7]. The computation and the access of the memory may take time and therefore the latency is long. Since memory size of N^2 is a large quantity, this work tries to use a new approach, interlaced read scan algorithm (IRSA), to reduce the required transpose memory to an order of $2N$ or $4N$ (5/3 or 9/7 mode).

Without loss of generality, let us take a 6×6-pixel image to describe the 2-D 5/3 mode LDWT operation and the proposed IRSA. Figure 3 shows the schematic diagram of the 2-D 5/3 mode LDWT operations of a 6×6 image. In Fig. 3, $x(i,j)$, $i = 0$ to 5 and $j = 0$ to 5, represents the original image signal. The left most two columns are the left boundary extension columns, and the right most column is the right boundary extension column. The details of the boundary extension are described in the previous section. The upper half of Fig. 3 shows the first stage 1-D DWT operations. The lower half of Fig. 3 shows the second stage 1-D DWT operations for finding the four subband coefficients, HH, HL, LH, and LL. In the first stage 1-D DWT, three pixels are used to find a 1-D high-frequency coefficient. For example, $x(0,0)$, $x(0,1)$, and $x(0,2)$ are used to find the high-frequency coefficient $b(0,0)$, and $b(0,0) = -[x(0,0) + x(0,2)]/2 + x(0,1)$. To calculate the next high-frequency coefficient $b(0,1)$, we need pixels signal $x(0,2)$, $x(0,3)$, and $x(0,4)$. Here $x(0,2)$ is used to calculate both of $b(0,0)$ and $b(0,1)$ and is called the overlapped pixel. The low-frequency coefficient is calculated using two consecutive high-frequency coefficients and the overlapped pixel. For example, $b(0,0)$ and $b(0,1)$ cope with $x(0,2)$ to find the low-frequency coefficient $c(0,1)$, $c(0,1) = [b(0,0) + b(0,1)]/4 + x(0,2)$. The calculated high-frequency coefficients, $b(i,j)$, and low-frequency coefficients, $c(i,j)$, are then used in the second stage 1-D DWT to calculate the four subband coefficients, HH, HL, LH, and LL.

In the second stage 1-D DWT in Fig. 3, the first HH coefficient, HH(0,0), is calculated by using $b(0,2)$, $b(0,1)$, and $b(0,0)$, $HH(0,0) = -[b(0,0) + b(0,2)]/2 + b(0,1)$. The other HH coefficients can be computed in the same manner using 3 column consecutive $b(i,j)$ signals. For two column consecutive HH coefficients we have an overlapped $b(i,j)$ signal. For example $b(0,3)$ is the overlapped signal for computing HH(0,0) and HH(0,1). To compute HL coefficients, we need two column consecutive HH coefficients and an overlapped $b(i,j)$ signal. For example, HL(0,1) is computed from HH(0,0), HH(0,1), and $b(0,3)$, $HL(0,1) = [HH(0,0) + HH(0,1)]/4 + b(0,3)$. The LH coefficients are computed from the $c(i,j)$ signal, and each LH coefficient needs the calculation of three $c(i,j)$ signals. For example, LH(0,1) is computed from $c(0,2)$, $c(0,3)$, and $c(0,4)$, $LH(0,1) = -[c(0,2) + c(0,4)]/2 + c(0,3)$. For two column consecutive LH coefficients we have an overlapped $c(i,j)$ signal. For example, $c(0,3)$ is the overlapped signal of computing LH(0,0) and LH(0,1). To compute LL coefficients, we need two column consecutive LH coefficients and an overlapped $c(i,j)$ signal. For example, LL(0,1) is computed from LH(0,0), LH(0,1), and $c(0,2)$, $LL(0,1) = [LH(0,0) + LH(0,1)]/4 + c(0,2)$.



$x(i,j)$: original image, i = 0~5 and j = 0~5
$b(i,j)$: high frequency wavelet coefficient of 1-D LDWT
$c(i,j)$: low frequency wavelet coefficient of 1-D LDWT
HH: high-high frequency wavelet coefficient of 2-D LDWT
HL: high-low frequency wavelet coefficient of 2-D LDWT
LH: low-high frequency wavelet coefficient of 2-D LDWT
LL: low-low frequency wavelet coefficient of 2-D LDWT

Fig. 3: Example of 2-D 5/3 mode LDWT operations.

From the description of the operations of the 2-D 5/3 mode LDWT we find that each 1-D high-frequency coefficient, $b(i,j)$, is calculated from three image signals, and one of the image signal is overlapped with the previous $b(i,j)$. The 1-D low-frequency coefficient, $c(i,j)$, is calculated from two row consecutive $b(i,j)$'s and an overlapped pixel. The HH, HL, LH, and LL coefficients are computed from $b(i,j)$'s and $c(i,j)$'s. If we can change the scanning order of the first stage 1-D LDWT and the output order of the second stage 1-D LDWT, during the 2-D LDWT operation we need only to store the $b(i,j)$'s to the transpose memory (FIFO size of N) and the overlapped pixels to the internal memory (R4+R9 size of N). For an $N\times N$ image, the transpose memory block can be reduced to only size of $2N$ as shown in Fig. 4. Based on this idea, the IRSA is proposed. The IRSA can reduce the requirement of the transpose memory significantly. The block diagram of the IRSA with several pixels of an image is shown in Fig. 4. In Fig. 4, the numbers on top and left represent the coordinate indexes of a 2-D image. In order to increase the operation speed, the IRSA scans two pixels in the consecutive rows a time. IN1 and IN2 are the scanning inputs at the beginning. At the first clock, the system scans two pixels, $x(0,0)$ and $x(1,0)$, from IN1 and IN2, respectively. At the second clock, IN1 and IN2 read pixels $x(0,1)$ and $x(1,1)$, respectively. At clock 3, IN1 and IN2 read pixels $x(0,2)$ and $x(1,2)$, respectively. After IN1 and IN2 have read three pixels, the DWT tries to compute two 1-D high-frequency coefficients, $b(0,0)$ and $b(0,1)$, and these two high-frequency coefficients are stored in the transpose memory for the subsequent computation of the low-frequency coefficients. Pixels $x(0,2)$ and $x(1,2)$ are stored in the internal memory for the subsequent computation of the 1-D high-frequency coefficients.

At clock 4, the DWT scans pixels on row 2 and row 3, and IN1 and IN2 read pixels $x(2,0)$ and $x(3,0)$, respectively. At clock 5, IN1 and IN2 read pixels $x(2,1)$ and $x(3,1)$, respectively. At clock 6, IN1 and IN2 read pixels $x(2,2)$ and $x(3,2)$, respectively. At this moment the DWT tries to compute the two high-frequency coefficients, $b(2,0)$ and $b(3,0)$, upon pixels $x(2,0)$ to $x(2,2)$ and $x(3,0)$ to $x(3,2)$ respectively and these two high-frequency coefficients are stored in the transpose memory for the subsequent computation of the low-frequency coefficients. Pixels $x(2,2)$ and $x(3,2)$ are stored in the internal memory for the subsequent computation of the high-frequency coefficients. Then (at clock 7) the DWT jumps to the subsequent 2 rows to read three consecutive pixels in each row and compute the high-frequency coefficients. The coefficients are stored in the transpose memory and pixels $x(4,2)$ and $x(5,2)$ are stored in the internal memory. This procedure will continue to read three pixels and compute the high-frequency coefficients and store the coefficients to the transpose memory and store pixels $x(2,j)$ and $x(2,j+1)$ to the internal memory in each row until the last row.

Then IN1 and IN2 of the DWT will jump to row 0 and row 1 to read pixels $x(0,3)$ and $x(1,3)$, respectively. At the next clock, IN1 and IN2 read pixels $x(0,4)$ and $x(1,4)$, respectively. The DWT of IN1 then copes with pixels $x(0,3)$, $x(0,4)$, and $x(0,2)$ that were stored previously to compute the high-frequency coefficient. Simultaneously the DWT of IN2 copes with pixels $x(1,3)$, $x(1,4)$, and $x(1,2)$ that were stored previously to compute the high-frequency coefficients, $b(0,1)$ and $b(1,1)$. As soon as $b(0,1)$ and $b(1,1)$ are found, $b(0,0)$, $b(0,1)$, and $x(0,2)$ are used to generate the low-frequency coefficient $c(0,1)$, and $b(1,0)$, $b(1,1)$, and $x(1,2)$ are used to generate the low-frequency coefficient $c(1,1)$. The computed high-frequency coefficients are then stored in the transpose memory, and pixels $x(0,4)$ and $x(1,4)$ replace pixels $x(0,2)$ and $x(1,2)$ to be stored in the internal memory. IN1 and IN2 then jump to rows 2 and 3 to process the same operations until the end of the last pixel. The detail operations are shown in Fig. 5.

The second stage 1-D DWT works in the similar manner as the first stage 1-D DWT. In the HH and HL operations, when three column consecutive $b(i,j)$'s are found in the first stage 1-D DWT, an HH coefficient can be computed. As soon as two column consecutive HH coefficients are found, the two HH coefficients can cope with the overlapped $b(i,j)$'s to compute an HL coefficient. Similarly, when three column consecutive $c(i,j)$'s are found in the first stage 1-D DWT, an LH coefficient can be computed. As soon as two LH coefficients are found, the two LH coefficients can cope with the overlapped $c(i,j)$ to compute an LL coefficient. The detailed operations for the second stage 1-D DWT are shown in Fig. 6.

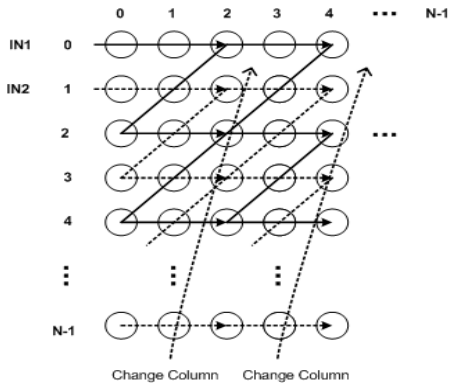


Fig. 4: IRSA of the 2-D LDWT.

IV. PROPOSED VLSI ARCHITECTURE AND IMPLEMENTATION FOR THE 2-D DUAL-MODE LDWT

We have discussed the IRSA in the previous section, and the architecture of IRSA is described in this section. We can manipulate the control unit to read off-chip memory. In IRSA, two pixels are scanned concurrently, and the system needs two processing units. For the 2-D LDWT processing, the pixels are processed by the first stage 1-D DWT first. The outputs are then fed to the second stage 1-D DWT to find the four subband coefficients, HH, HL, LH, and LL. Using our approach, the transpose memory can be reduced significantly. There are two parts in the architecture, the first stage 1-D DWT and the second stage 1-D DWT. Here we concentrate on the 2-D 5/3 mode LDWT.

A. The First Stage 1-D LDWT

The first stage 1-D LDWT architecture consists of the following units: signal arrangement, multiplication and accumulation cell (MAC), multiplexer (MUX), and first-in-first-out (FIFO) register. The block diagram is shown in Fig. 7.

The signal arrangement unit consists of three registers, R1, R2, and R3. The pixels are input to R1 first, and subsequently the content of R1 is transferred to R2 and then R3, and R1 keeps reading the following pixels. The operation is like a shift register. As soon as R1, R2, and R3 get signal data, MAC starts operating.

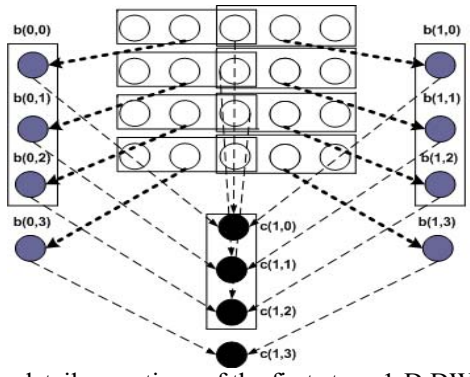
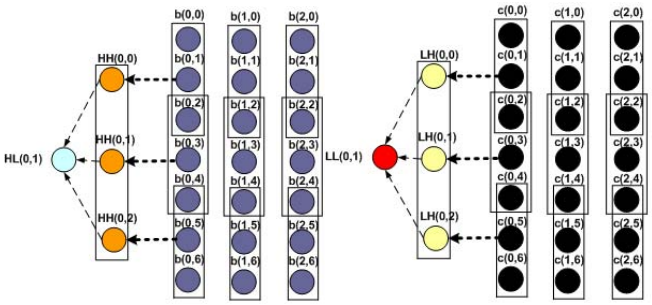


Fig. 5: The detail operations of the first stage 1-D DWT.



(a) (b)

Fig. 6: The detailed operations of the second stage 1-D DWT. (a) The HF (HH and HL) part operations. (b) The LF (LH and LL) part operations.

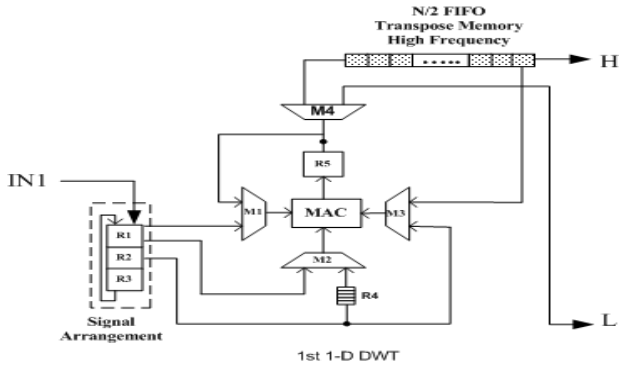


Fig. 7: The architecture of the first stage 1-D DWT.

For the low-frequency coefficients calculation we need two high-frequency coefficients and an original pixel. Internal register R4 is used to store the original even pixel (N1) and internal register R9 is used to store the original odd pixel (N2). We can simply shift the content of R3 to R4 after the MAC operation. FIFO is used to store the high-frequency coefficients to calculate the low-frequency coefficients. Register R5 has two functions: *1)* It can store the high-frequency coefficients for the low-frequency coefficient calculation. *2)* It is used to be a signal buffer for MAC. MAC needs time to compute the signal, and the output of MAC cannot directly feed the result to the output or the following operation may be incorrect due to the synchronization problems. R5 acts as an output buffer for MAC to prevent the error in the following operations. In the 5/3 integer lifting-based operations, MAC is used to find the results of the high frequency output, $-(a1+a3)/2 + a2$, and the low frequency output, $(a1+a3)/4+a2$. There are two multiplication coefficients, $-1/2$ and $1/4$. To save hardware, we can use shifters to implement the $-1/2$ and $1/4$ multiplications. Therefore the MAC needs adders, complementer, and shifters.

B. The Second Stage 1-D LDWT

Similar to the first stage 1-D DWT, the second stage 1-D DWT consists of the following units: signal arrangement, MAC, and MUX, as shown in Fig. 8. Due to the parallel architecture, two outputs are generated concurrently from the first stage 1-D DWT, and these two outputs must be merged in the second stage 1-D DWT. At the beginning, signal H0 and H1 are from IN1 and IN2 and these two signals are stored in R3 and R4 respectively. At the next clock, H0 and H1 are moved to R1 and R2 respectively, and concurrently new signals H3 and H4 from IN1 and IN2 are stored to R3 and R4 respectively. The signal arrangement unit operates repeatedly to input signal for the second stage 1-D DWT.

C. 2-D LDWT Architecture

In our IRSA operation, IN1 and IN2 read signals of even row and odd row in zig-zag orders, respectively. The block diagram of the proposed 2-D LDWT is shown in Fig. 9. It consists of two stages, the first stage 1-D DWT and the second stage 1-D DWT. This architecture needs only a small amount of the transpose memory.

The signal processing of the second stage 1-D DWT is shown in Fig. 8. The 2×4 signals in each second stage 1-D DWT are then processed, and then HH, HL, LH, and LL are generated and each has 2×2 signal data. The complete architecture of the 2-D LDWT is shown in Fig. 9. The complete 2-D LDWT consists of four parts, two sets of the first stage 1-D DWT, two sets of the second stage 1-D DWT, control unit, and MAC unit.

According to (5) and (6), the proposed IRSA architecture can be also applied to the 9/7 mode LDWT. From Figs. 5 and 6 in section IV, the original signals (denoted as black circles) for both 5/3 and 9/7 modes LDWT can be processed by the same IRSA for the first stage 1-D DWT operation. The high-frequency signals (denoted as grey circles) and the correlated

low-frequency signals together with the results of the first stage are used to compute the second stage 1-D DWT coefficients. Compared to the 9/7 mode LDWT computation, the 5/3 mode LDWT is much easier for computation, and the registers arrangement in Figs. 7 and 8 is simple. For 9/7 mode LDWT implementation with the same system architecture of 5/3 mode LDWT, we have to do the following modifications: *1)* The control signals of the MUX in Figs. 7 and 8 must be modified. We have to rearrange the registers for the MAC block to process the 9/7 parameters. *2)* The wavelet coefficients of dual-mode LDWT are different. The coefficients are α= -1/2 and β= 1/4 for 5/3 mode LDWT, but the coefficients are α= −1.586134142, β= −0.052980118, γ= +0.882911075, and δ= +0.443506852 for 9/7 mode LDWT. For calculation simplicity and good precision, we can use the integer approach proposed by Hwang *et al*. [8] for 9/7 mode LDWT calculation. Similar to the multiplication implementation by shifters and adders in the 5/3 mode LDWT, we can adopt the shifters approach proposed in [11] further to implement the 9/7 mode LDWT. *3)* According to the characteristics of the 9/7 mode LDWT, the control unit in Fig. 9 must be modified accordingly.

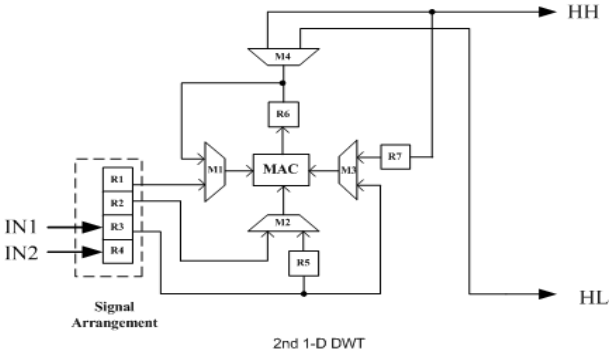


Fig. 8: The block diagram of the second stage 1-D LDWT.

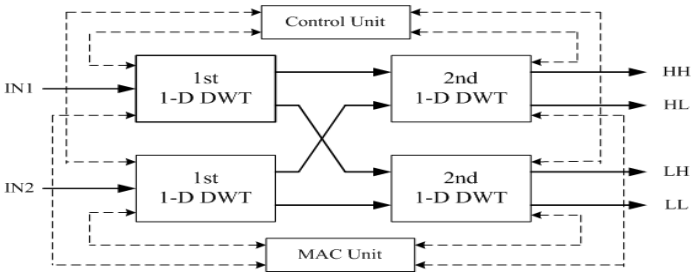


Fig. 9: The complete 2-D DWT block diagram.

The multilevel DWT computation can be implemented in a similar manner by our proposed high performance 1-level 2-D LDWT. For the multi-level computation, this architecture needs $N^2/4$ off-chip memory. The off-chip memory is used to temporarily store the LL subband coefficients for the next iteration computations. The second level computation requires $N/2$ counters and $N/2$ FIFO's for the control unit. The third level computation requires $N/4$ counters and $N/4$ FIFO's for the control unit. Generally in the *j*th level computation, we need $N/2^{j-1}$ counters and $N/2^{j-1}$ FIFO's.

V. EXPRIMENTAL RESULTS AND COMPARISONS

The 2-D dual-mode LDWT considers the trade-off between low transpose memory and low complexity in the design of VLSI architecture. The performance comparisons of our architecture and other similar architectures are listed in Tables I and II. According to the compression results, the proposed VLSI architecture outperforms previous works in the aspects of transpose memory size that we can reduce about 50% memory in comparison with JPEG2000 standard [9] architecture. Moreover, the 2-D LDWT is frame-based with the implementation bottleneck being the huge amount of the transpose memory size. The proposed IRSA approach has the advantages of memory-efficient and high-speed. The proposed 2-D dual-mode LDWT adopts parallel and pipeline schemes to reduce the transpose memory and increase the operating speed. The shifters and adders replace multipliers in the computation to reduce the hardware cost. A 256×256 2-D dual-mode LDWT is designed and simulated with VerilogHDL, and further synthesized by the Synopsys design compiler with TSMC 0.18μm 1P6M CMOS standard process technology to verify the performance of the proposed hardware architecture; the performance specifications are listed in Table III.

VI. CONCLUSIONS

In this paper, we proposed a new architecture of the 2-D LDWT for JPEG2000 and developed an efficient hardware architecture based on the proposed IRSA and a parallel scheme. Compared to previous architectures of the 2-D LDWT, the proposed 2-D architecture are efficient alternatives in tradeoff low transpose memory requirement, output latency, control complexity, and regular memory access order. Our proposed architecture reduces transpose memory significantly to a memory size of only *2N* or 4*N* (5/3 or 9/7 mode), and reduce the latency to (3/2)*N*+3. Based on the hardware architecture for the 2-D LDWT is designed by the TSMC 0.18μm 1P6M standard CMOS technology. The design is regular, simple, and well suited for VLSI implementation. Finally, the 5/3 and 9/7 filters with the different lifting steps can be realized by cascading the four (split phase, predict phase, update phase, scaling) modules.

Table I. COMPARISON OF 2-D ARCHITECTURE FOR 5/3 LDWT

Proposed	Ours	[1]	[2]	[3]	[4]	[5]	[9]	[12]	[13]	[14]
T.M.(bytes)	$2N$	$3.5N$	$3.5N$	$2.5N$	$3N$	$N^2/4+5N$	N^2	$2N$	$3.5N$	$3.5N$
Latency	$(3/2)N+3$	---	$2N+5$	3	---	3	---	---	Yes	Yes
C.T.	$(3/4)N^2+(3/2)N+7$	---	$(N^2/2)+N+5$	N^2	$(N^2/2)+N+5$	N^2	---	---	---	---

[1] Transpose memory (T.M.) size is used to store frequency coefficients in the 1-L 2-D DWT.
[2] In a system, latency is often used to mean any delay or waiting time that increases real or perceived response time beyond the response time desired. For example, specific contributors to 2-D DWT latency include from original image input to first subband output in signal.
[3] In a system, computing time (C.T.) represents the time used to compute an image of size $N\times N$.
[4] Suppose image is of size $N\times N$.

Table II: COMPARISON OF 2-D ARCHITECTURE FOR 9/7 LDWT

Proposed	Ours	[2]	[6]	[7]	[10]	[11]	[13]	[14]	[15]
T.M.	$4N$	N^2	$12N$	$N^2/4+LN+L$	$22N$	$14N$	$5.5N$	$5.5N$	N^2+4N+4
Latency	$(3/2)N+3$	---	---	---	---	---	---	---	---
C.T.	$(3/4)N^2+(3/2)N+7$	$4N^2/3+2$	N^2	$N^2/2\sim(2/3)N$	N^2	---	---	---	$2N^2/3$

[5] L is filter length.

Table III: DESUGN SPECIFICATION OF THE PROPOSED 2-D DWT

CHIP SPECIFICATION	N= 256, TILE SIZE = 256×256
POWER SUPPLY	1.8V
TECHNOLOGY	TSMC 0.18μM 1P6M (CMOS)
ON-CHIP MEMORY SIZE (TRANSPOSE+ INTERNAL)	2-D 5/3 DWT: 512 BYTES 2-D 9/7 DWT: 1,024 BYTES
LATENCY	$(3/2)N+3$ = 387
COMPUTING TIME	$(3/4)N^2+(3/2)N+7$ = 49,543

REFERENCES

[1] C. Diou, L. Torres, and M. Robert, "An embeded core for the 2-D wavelet transform," *IEEE on Emerging Technologies and Factory Automation Proceedings*, vol. 2, pp. 179-186, 2001.

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

[3] S.-C. Chen and C.-C. Wu, "An architecture of 2-D 3-level lifting-based discrete wavelet transform," *Proc. of the VLSI Design/ CAD Symposium*, pp. 351-354, 2002.

[4] P.-Y. Chen, "VLSI implementation of discrete wavelet transform using the 5/3 filter," *IEICE Trans. on Information and Systems*, vol. E85-D, no.12, pp. 1893-1897, 2002.

[5] J.-S. Chiang and C.-H. Hsia, "An efficient VLSI architecture for 2-D DWT using lifting scheme" *IEEE Int. Conference on Systems and Signals*, pp. 528-531, 2005.

[6] G.-C Jung and S.-M. Park, "VLSI implement of lifting wavelet transform of JPEG2000 with efficient RPA (recursive pyramid algorithm) realization," *IEICE Trans. on Fundamentals*, vol. E88-A, no. 12, pp. 3508-3515, 2005.

[7] P.-Y. Chen, "VLSI implementation for one-dimensional multilevel lifting-based wavelet transform," *IEEE Trans. on Computer*, vol. 53, no. 4, pp. 386-398, 2004.

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

[9] *JPEG 2000 Part 1 Final Committee Draft Version 1.0, ISO/IEC 15444-1 JTC1/SC29 WG1*, Information Technology, March 2000.

[10] M. Vishwanath, R. M. Owens, and M. J. Irwin, "VLSI architecture for the discrete wavelet transform," *IEEE Trans. on Circuits and Systems II*, vol. 42, no. 5, pp. 305-316, 1995.

[11] C.-T. Huang, P.-C. Tseng, and L.-G. Chen, "Efficient VLSI architecture of lifting-based discrete wavelet transform by systematic design method," *IEEE Int. Symposium Circuits and Systems*, vol. 5, pp. 26-29, 2002.

[12] K. Mei, N. Zheng, and H. van de Wetering, "High-speed and memory-efficient VLSI design of 2-D DWT for JPEG2000 applications," *IET Electronics Letter*, vol. 42, no. 16, 2006.

[13] C.-T. Huang, P.-C. Tseng, and L.-G. Chen, "Generic RAM-based architecture for two-dimensional discrete wavelet transform with line-based method," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 15, no.7, pp. 910-919, 2005.

[14] B.-F. Wu and C.-F. Lin, "A high-performance and memory-efficient pipeline architecture for the 5/3 and 9/7 discrete wavelet transform of JPEG2000 codec," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 15, no.12, pp. 1615-1628, 2005.

[15] P.-C. Wu and L.-G. Chen, "An efficient architecture for two-dimensional discrete wavelet transform," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 11, no.4, pp. 536-545, 2001.