4

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

Chih-Hsien Hsia and Jen-Shiun Chiang
Department of Electrical Engineering, Tamkang University
Taipei, Taiwan

1. Introduction

Discrete wavelet transform (DWT) has been adopted in a wide range of applications, including speech analysis, numerical analysis, signal analysis, image coding, pattern recognition, computer vision, and biometrics (Mallat, 1989). It can be considered as a multi-resolution decomposition of a signal into several components with different frequency bands. Moreover, DWT is a powerful tool for signal processing applications, such as JPEG2000 still image compression, denoising, region of interest, and watermarking. For real-time processing it needs small memory access and low computational complexity. Implementations of two-dimensional (2-D) DWT can be classified as convolution-based operation (Mallat, 1989) (Marino, 2000) (Vishwanath et al., 1995) (Wu. & Chen, 2001) and lifting-based operation (Sweldens, 1996). Since the convolution-based implementations of DWT have high computational complexity and large memory requirements, lifting-based DWT has been presented to overcome these drawbacks (Sweldens, 1996) (Daubechies & Sweldens, 1998). The lifting-based scheme can provide low-complexity solutions for image/video compression applications, such as JPEG2000 (Lian et al., 2001), Motion-JPEG2000 (Seo & Kim, 2007), MPEG-4 still image coding, and MC-EZBC (Ohm, 2005) (Chen & Woods, 2004). However, the real-time 2-D DWT for multimedia application is still difficult to achieve. Hereafter, efficient transformation schemes for real-time application are highly demanded. Performing 2-D (or multi-dimensional) DWT requires many computations and a large block of transpose memory for storing intermediate signals with long latency time. This work presents new algorithms and hardware architectures to improve the critical issues in 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 merits of low transpose memory, low latency, and regular signal flow, making it suitable for VLSI implementation. The transpose memory requirement of the $N \times N$ 2-D 5/3 mode LDWT is $2N$ and that of 2-D 9/7 mode LDWT is $4N$.

Low transpose memory requirement is of a priority concern in spatial-frequency domain implementation. Generally, raster scan signal flow operations are popular in $N \times N$ 2-D DWT, and under this approach the memory requirement ranges from $2N$ to N^2 (Diou et al., 2001)

(Andra et al., 2002) (Chen & Wu, 2002) (Chen, 2002) (Chiang & Hsia, 2005) (Jung & Park, 2005) (Vishwanath et al., 1995) (Huang et al., 2005) (Mei et al., 2006) (Huang et al, 2005) (Wu & Lin, 2005) (Lan ., 2005) (Wu. & Chen, 2001) in 2-D 5/3 and 9/7 modes LDWT. In order to reduce the amount of the transpose memory, the memory access must be redirected. In our approach, the signal flow is revised from row-wise only to mixed row- and column-wise, and a new approach, called interlaced read scan algorithm (IRSA), is used to reduce the amount of the transpose memory. By the IRSA approach, a transpose memory size is of 2*N* or 4*N* (5/3 or 9/7 mode) for an *N*×*N* DWT. The proposed 2-D LDWT architecture is based on parallel and pipelined schemes to increase the operation speed. For hardware implementation we replace multipliers with shifters and adders to accomplish high hardware utilization. This 2-D LDWT has the characteristics of high hardware utilization, low 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.

2. Survey of 2-D LDWT Architecture

Among the variety of DWT algorithms, LDWT provides a new approach for constructing biorthogonal wavelet transforms and also provides an efficient scheme for calculating classical wavelet transforms (Sweldens, 1996) (Chen & Wu, 2002) (Andra et al., 2000) (Andra et al., 2002) (Diou et al., 2001) (Chen, 2002) (Chiang & Hsia, 2005) (Tan & Arslan, 2001) (Huang et al., 2005) (Huang et al., 2002) (Mei et al., 2006) (Weeks & Bayoumi, 2002) (Varshney et al., 2007) (Huang et al., 2004) (Tan & Arslan, 2003) (Jiang & Ortega, 2001) (Jung & Park, 2005) (Chen, 2004) (Lian et al, 2001) (Seo & Kim, 2007) (Huang et al., 2005) (Wu & Lin, 2005) (Lan et al., 2005) (Wu. & Chen, 2001). Factoring the classical wavelet filter into lifting steps can reduce the computational complexity of the corresponding DWT by up to 50% (Daubechies & Sweldens, 1998). The lifting steps can be implemented easily, which is different from the direct finite impulse response (FIR) implementations of Mallat's algorithm (Daubechies & Sweldens, 1998). Andra *et al.* (Andra et al., 2000) (Andra et al., 2002) proposed a block-based simple four-processor architecture that computes several stages of the DWT at a time. Diou *et al.* (Diou et al., 2001) presented an architecture that performs LDWT with a 5/3 filter by interleaving technique. Chen *et al.* (Chen & Wu, 2002) proposed a folded and pipelined architecture for a 2-D LDWT implementation, with memory size of 2.5*N* for an *N*×*N* 2-D DWT. This lifting architecture for vertical filtering is divided into two parts, each consisting of one adder and one multiplier. Since 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, this architecture also has high complexity due to the characteristics of the signal flow. Chen *et al.* (Chen, 2002) proposed a flexible folded architecture for 3-level 1-D LDWT to increase the hardware utilization. Chiang *et al.* (Chiang & Hsia, 2005) proposed a 2-D DWT folded architecture to improve the hardware utilization. Jiang *et al.* (Jiang & Ortega, 2001) presented a parallel processing architecture that models the DWT computation as a finite state machine and efficiently computes the wavelet coefficients near the boundary of each segment of the input signal. Lian *et al.* (Lian et al, 2001) and Chen *et al.* (Chen, 2004) used a 1-D folded architecture to improve the hardware utilization of 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. Jung *et al.* (Jung & Park, 2005) presented an efficient VLSI architecture of dual-mode LDWT that is used by lossy or lossless compression of JPEG2000. Marino (Marino, 2000) proposed a high-speed/low-power pipelined architecture for the direct 2-D DWT by four-subband transforms performed in parallel. The architecture of (Huang et al., 2002) implements 2-D DWT with only transpose memory by using recursive pyramid algorithm (PRA). In (Vishwanath et al., 1995) it has the average of N^2 computing time for all DWT levels. However, they use many multipliers and adders. Varshney *et al.* (Varshney et al., 2007) presented energy efficient single-processor and fully pipelined architectures for 2-D 5/3 lifting-based JPEG2000. The single processor performs both row-wise and column-wise processing simultaneously to achieve the 2-D transform with 100% hardware utilization. Tan *et al.* (Tan & Arslan, 2003) presented a shift-accumulator arithmetic logic unit architecture for 2-D lifting-based JPEG2000 5/3 DWT. This architecture has an efficient memory organization, which uses a small amount of embedded memory for processing and buffering. Those architectures achieve multi-level decomposition using an interleaving scheme that reduces the size of memory and the number of memory accesses, but have slow throughput rates and inefficient hardware utilization. Seo *et al.* (Seo & Kim, 2007) proposed a processor that can handle any tile size, and supports both 5/3 and 9/7 filters for Motion-JPEG2000. Huang *et al.* (Huang et al, 2005) proposed a generic RAM-based architecture with high efficiency and feasibility for 2-D DWT. Wu *et al.* (Wu & Lin, 2005) presented a high-performance and low-memory architecture to implement a 2-D dual-mode LDWT. The pipelined signal path of their architecture is regular and practical. Lan *et al.* (Lan et al., 2005) proposed a scheme that can process two lines simultaneously by processing two pixels in a clock period. Wu *et al.* (Wu. & Chen, 2001) proposed an efficient VLSI architecture for direct 2-D LDWT, in which the poly-phase decomposition and coefficient folding are adopted to increase the hardware utilization. Despite these efficient improvements to existed architectures, further improvements in the algorithm and architecture are still needed. Some VLSI architectures of 2-D LDWT try to reduce the transpose memory requirements and communication between the processors (Chen & Wu, 2002) (Andra et al., 2000) (Andra et al., 2002) (Diou et al., 2001) (Chen, 2002) (Chiang & Hsia, 2005) (Tan & Aslan, 2002) (Jiang & Ortega, 2001) (Lian et al., 2001) (Jung & Park, 2005) (Chen, 2004) (Huang et al., 2005) (Daubechies & Sweldens, 1998) (Marino, 2000) (Vishwanath et al., 1995) (Taubman & Marcellin, 2001) (Marcellin et al., 2000) (Mei et al., 2006) (Varshney et al., 2007) (Huang et al., 2004) (Tan & Arslan, 2003) (Seo & Kim, 2007) (Huang et al., 2005) (Wu & Lin, 2005) (Lan et al., 2005) (Wu. & Chen, 2001), however these hardware architectures still need large transpose memory.

3. Discrete wavelet transform and lifting-based method

This section briefly reviews the use of DWT in the coding engine of JPEG2000 (Taubman & Marcellin, 2001). The classical DWT employs filtering and convolution to achieve signal decomposition (Mallat, 1989) (Marino, 2000) (Vishwanath et al., 1995) (Wu. & Chen, 2001). Meyer and Mallat found that the orthonormal wavelet decomposition and reconstruction can be implemented in the multi-resolution signal analysis framework (Mallat, 1989). The multi-resolution analysis is now a standard method for constructing the orthonormal wavelet-bases. JPEG2000 adopts this characteristic to transform an image in the spatial domain into the frequency domain.

3.1 Classical DWT

DWT performs multi-resolution decomposition of the input signals (Mallat, 1989). The original signals are first decomposed into two subspaces, called the low- (low-pass) and high-frequency (high-pass) subbands. The classical DWT implements the decomposition (analysis) of a signal by a low-pass digital filter H and a high-pass digital filter G. Both digital filters are derived using the scaling function and the corresponding wavelets. The system downsamples the signal to decimate half of the filtered results in decomposition processing. The Z-transfer functions of $H(z)$ and $G(z)$ based on four-tap and non-recursive FIR filters with length L are represented as follows:

$$H(z)= h_0 + h_1z^{-1} + h_2z^{-2}+ h_3z^{-3}, \tag{1}$$

$$G(z)= g_0 + g_1z^{-1} + g_2z^{-2}+ g_3z^{-3}. \tag{2}$$

The reconstruction (synthesis) process is implemented using an up-sampling process. Mallat's tree algorithm or pyramid algorithm (Mallat, 1989) can be used to find the multi-resolution decomposition DWT. The decomposition DWT coefficients at each resolution level can be calculated as follows:

for (j=1 to J)
for (i=0 to $N/2^j$-1)
{

$$X_H^j(n)=\sum_{i=0}^{k-1} G(z)X^{J-1}H(2n-i) \tag{3}$$

$$X_L^j(n)=\sum_{i=0}^{k-1} H(z)X^{J-1}G(2n-i) \tag{4}$$

}

where j denotes the current resolution level, k the number of the filter tap, $X_H^j(n)$ the nth high-pass DWT coefficient at the jth level, $X_L^j(n)$ the nth low pass DWT coefficient at the jth level, and N the length of the original input sequence. Fig. 1 shows a 3-level 1-D DWT decomposition using Mallat's algorithm.

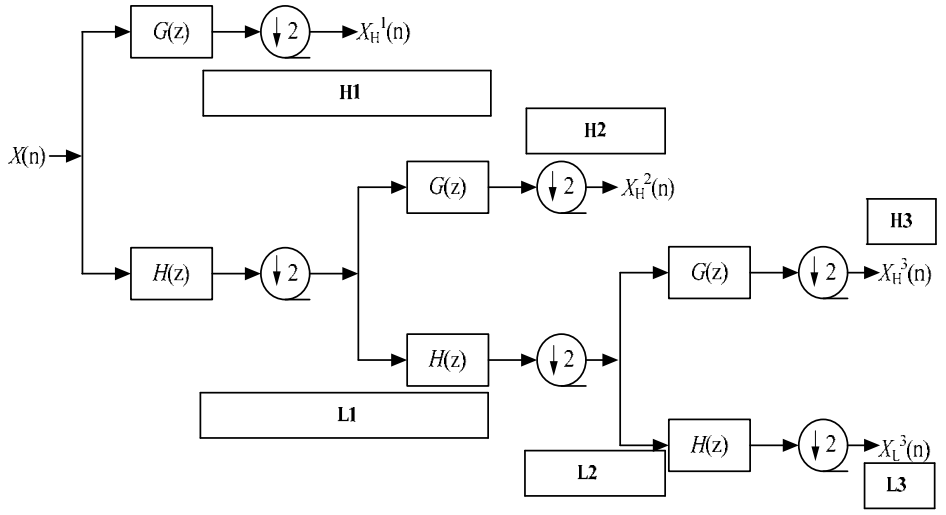


Fig. 1. 3-level 1-D DWT decomposition using Mallat's algorithm.

The downsampling operation is then applied to the filtered results. A pair of filters are applied to the signal to decompose the image into the low-low (LL), low-high (LH), high-low (HL), and high-high (HH) wavelet frequency bands. Fig. 2 illustrates the basic 2-D DWT operation and the transformed result which is composed of two cascading 1-D DWTs. The image is first analyzed horizontally to generate two subimages. The information is then sent into the second 1-D DWT to perform the vertical analysis to generate four subbands, and each with a quarter of the size of the original image. Considering an image of size $N \times N$, each band is subsampled by a factor of two, so that each wavelet frequency band contains $N/2 \times N/2$ samples. The four subbands can be integrated to generate an output image with the same number of samples as the original one.

Most image compression applications can reapply the above 2-D wavelet decomposition repeatedly to the LL subimage, each time forming four new subband images, to minimize the energy in the lower frequency bands.

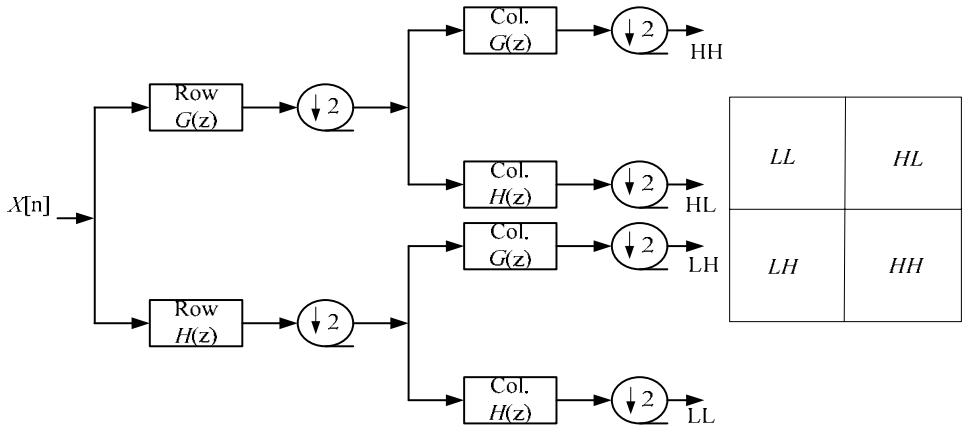


Fig. 2. The 2-D analysis DWT image decomposition process.

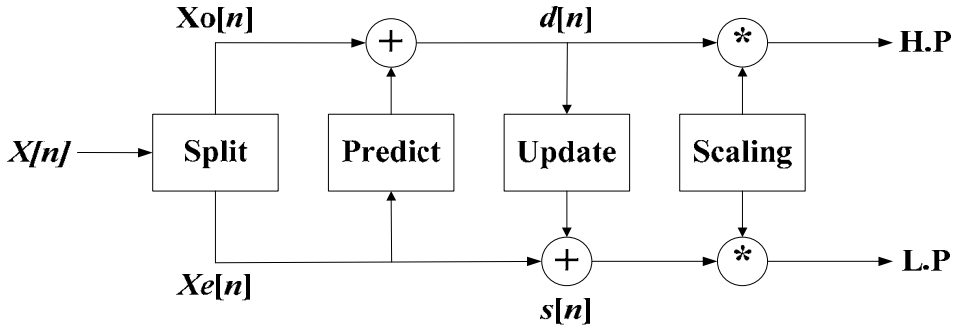


Fig. 3. Block diagram of the LDWT.

3.2 LDWT algorithm

The lifting-based scheme proposed by Daubechies and Sweldens requires fewer computations than the traditional convolution-based approach (Sweldens, 1996) (Daubechies & Sweldens, 1998). The lifting-based scheme is an efficient implementation for DWT; it can easily use integer operations and avoid the problems caused by the finite precision or rounding. The Euclidean algorithm can be used to factorize the poly-phase matrix of a DWT filter into a sequence of alternating upper and lower triangular matrices and a diagonal matrix. The variables $h(z)$ and $g(z)$ in (5) respectively denote the low- and high-pass analysis filters, which can be divided into even and odd parts to generate a poly-phase matrix $P(z)$ as in (6).

$$g(z)=g_e(z^2)+ z^{-1}g_o(z^2),$$

$$h(z)=h_e(z^2)+z^{-1}h_o(z^2). \quad (5)$$

$$P(z) = \begin{bmatrix} h_e(z) & g_e(z) \\ h_o(z) & g_o(z) \end{bmatrix}. \tag{6}$$

The Euclidean algorithm recursively finds the greatest common divisors of the even and odd parts of the original filters. Since h(z) and g(z) form a complementary filter pair, P(z) can be factorized into (7):

$$P(z) = \prod_{i=1}^{m} \begin{pmatrix} 1 & s_i(z) \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ t_i(z) & 1 \end{pmatrix} \begin{pmatrix} k & 0 \\ 0 & 1/k \end{pmatrix}. \tag{7}$$

where s_i(z) and t_i(z) are Laurent polynomials corresponding to the prediction and update steps, respectively, and k is a nonzero constant. Therefore, the filter bank can be factorized into three lifting steps.

As illustrated in Fig. 3, 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 also 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 (8).

$$d[n]=Xo[n]+P\times(Xe[n]). \tag{8}$$

3) Update phase: *Xe*[n] and d[n] are combined to obtain the scaling coefficients s[n] after an update operator U as in (9).

$$s[n]=Xe[n]+U\times(d[n]). \tag{9}$$

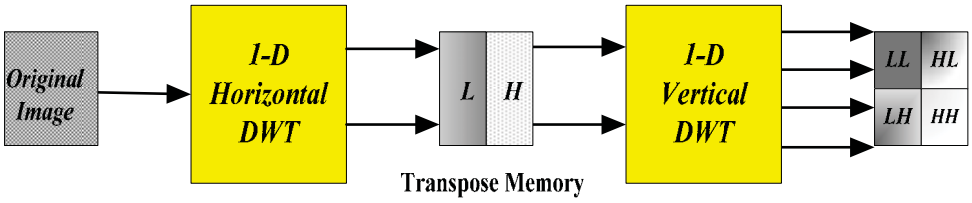
4) Scaling: In the final step, the normalization factor is applied on s[n] and d[n] to obtain the wavelet coefficients. For example, (10) and (11) describe the implementation of the 5/3 integer lifting analysis DWT and are used to calculate the odd (high-pass) and even coefficients (low-pass), respectively.

$$d^*[n] = X(2n+1) - \lfloor X(2n) + X(2n+2)/2 \rfloor. \tag{10}$$

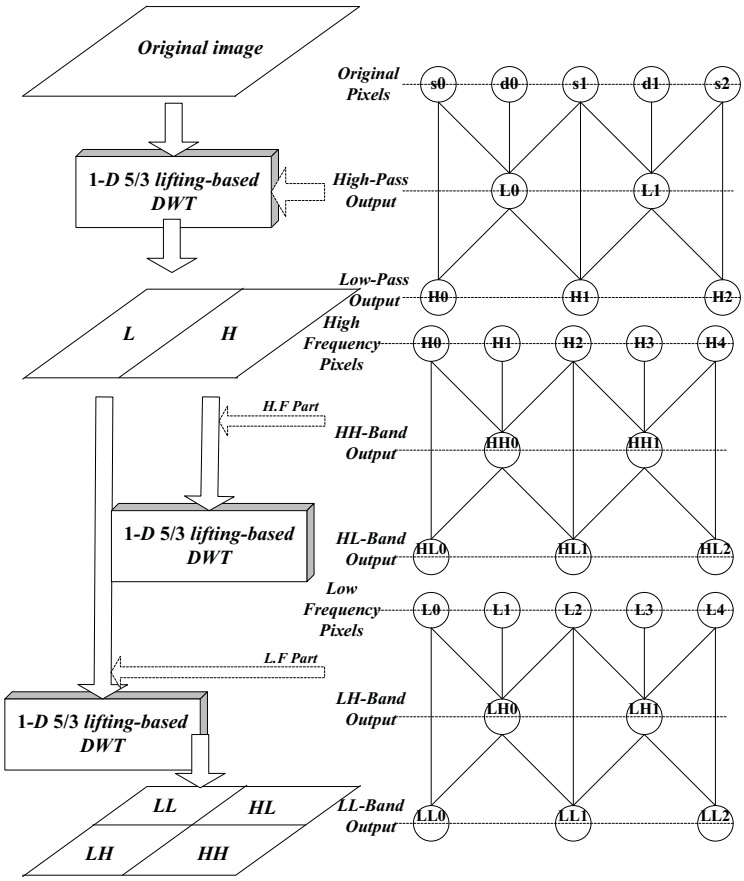
$$s^*[n] = X(2n) - \lfloor d(2n-1) + d(2n+1) + 2/4 \rfloor. \tag{11}$$

Although the lifting-based scheme has low complexity, its long and irregular signal paths cause the major limitation for efficient hardware implementations. Additionally, the increasing number of pipelined registers increases the internal memory size of the 2-D DWT architecture. The 2-D LDWT uses a vertical 1-D LDWT subband decomposition and a horizontal 1-D LDWT subband decomposition to find the 2-D LDWT coefficients. Therefore, the memory requirement dominates the hardware cost and architectural complexity of 2-D LDWT. Fig. 4 shows the 5/3 mode 2-D LDWT operation. The default wavelet filters in JPEG2000 are dual-mode (5/3 and 9/7 modes) LDWT (Taubman & Marcellin, 2001). The lifting-based steps associated with the dual-mode wavelets are shown in Figs. 5 and 6,

respectively. Assuming that the original signals are infinite in length, the first lifting stage is first applied to perform the DWT.



(a)



(b)

Fig. 4. 5/3 mode 2-D LDWT operation. (a) The block diagram flow of a traditional 2-D DWT. (b) Detailed processing flow.

Fig. 5 shows the lifting-based step associated with the wavelet algorithm. The original signals including $s0$, $d0$, $s1$, $d1$, $s2$, $d2$, … are the input pixel sequences. If the original signals are infinite in length, then the first-stage lifting is applied to update the odd index data $s0$, $s1$, …. In (12), the parameters $-1/2$ and H_i denote the first stage lifting parameter and outcome, respectively. Equation (12) shows the operation of the 5/3 integer LDWT (Wu & Lin, 2005) (Martina & Masera, 2007) (Hsia & Chiang, 2008).

$$H_i = [(S_i+S_{i+1})\times\alpha+d_i]\times K_0,$$

$$L_i = [(H_i+H_{i-1})\times\beta+S_i]\times K_1, \tag{12}$$

where $\alpha= -1/2$, $\beta=1/4$, and $K_0= K_1= 1$.

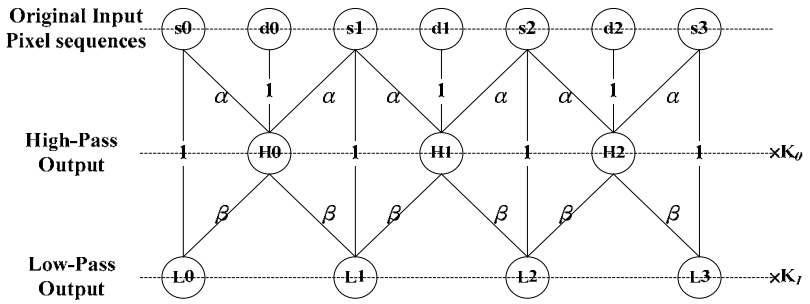


Fig. 5. 5/3 LDWT algorithm.

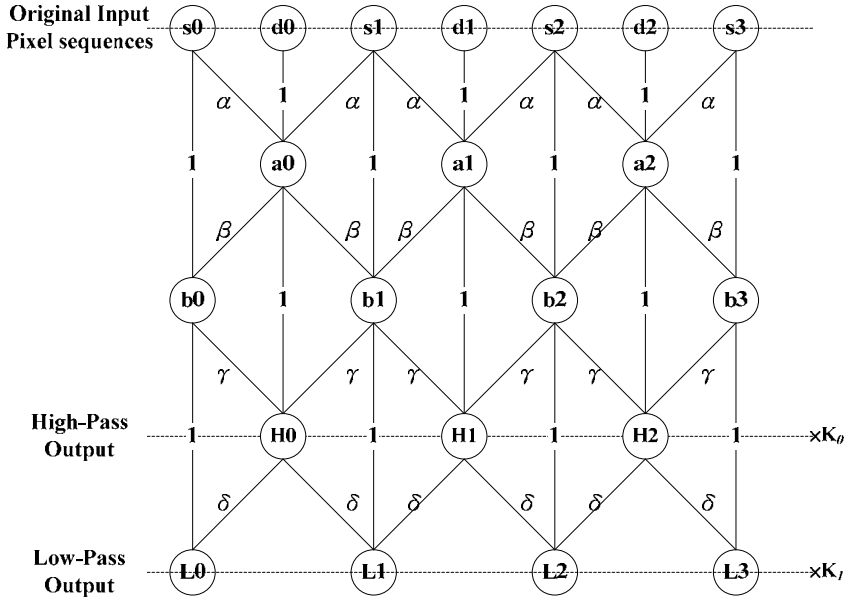


Fig. 6. 9/7 LDWT algorithm.

Together with the high-frequency lifting parameter, α, and the input signal we can find the first stage high-frequency wavelet coefficient, 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 (13).

$$\begin{aligned} a_i &= [(S_i+S_{i+1})\times\alpha+d_i], \\ b_i &= [(a_i+a_{i-1})\times\beta+S_i], \\ H_i &= [(b_i+b_{i+1})\times\gamma+a_i]\times K_0, \\ L_i &= [(H_i+H_{i-1})\times\delta+b_i]\times K_1. \end{aligned} \tag{13}$$

where α= −1.586134142, β= −0.052980118, γ= +0.882911075, δ= +0.443506852, K_0= 1.230174104, and K_1= 1/K_0.
The calculation comprises four lifting steps and two scaling steps.

3.3 Boundary extension treatment for LDWT

The finite-length signal processed by DWT leads to the boundary effect. The JPEG2000 standard enhances the symmetric extension pixel at the edge, as shown in Table 1. An appropriate signal extension is required to maintain the same number of the wavelet coefficients as in the original signal. The embedded signal extension algorithm (Tan & Arslan, 2001) can be used to compute the boundary of the image.
Since both the extended signal and the lifting structure are symmetrical, all the intermediate and final results of the lifting-based DWT are also symmetrical with regard to the boundary points, and the boundary extension can be performed without additional computational complexity. The inverse lifting structure is easily derived from Table 1 (Tan & Arslan, 2001). The boundary extension signal reflects the signal to i_{left} samples on the left and i_{right} samples on right. Table 1 shows the extension parameters i_{left} and i_{right} for the reversible transform 5/3 mode and the irreversible transform 9/7 mode.

i_0	i_{left}(5/3)	i_1	i_{right}(5/3)	i_0	i_{left}(9/7)	i_1	i_{right}(9/7)
even	2	odd	1	even	4	odd	3

* i_0: first sample index; i_1: last sample index.
Table 1. Boundary extension to the left and to the right for JPEG2000.

4. Interlaced read scan algorithm (IRSA)

In recent years, many 2-D LDWT architectures have been proposed to meet the requirements of 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 (Chen & Wu, 2002) (Andra et al., 2000) (Andra et al., 2002) (Diou et al., 2001) (Chen, 2002) (Chiang & Hsia, 2005) (Chen, 2004) (Huang et al., 2005) (Daubechies & Sweldens, 1998) (Marino, 2000) (Vishwanath et al., 1995) (Taubman & Marcellin, 2001)

(Varshney et al., 2007) (Huang et al., 2004) (Tan & Arslan, 2003) (Seo & Kim, 2007) (Huang et al., 2005) (Wu & Lin, 2005) (Lan et al., 2005) (Wu. & Chen, 2001). 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. Fig. 7(a) shows the concept of the proposed dual-mode LDWT architecture, which consists of signal arrangement unit, processing element, memory unit, and control unit, as shown in Fig. 7(b). The outputs are fed to the 2-D LDWT four-subband coefficients, HH, HL, LH, and LL. The proposed architecture is described in detail in this section, and we focus on the 2-D dual-mode LDWT.

Compared to the computation unit, the transpose memory becomes the main overhead in the 2-D DWT. The block diagram of a conventional 2-D DWT is shown in Fig. 4. 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 data to compute the 2-D DWT coefficients of the four subbands (Chen & Wu, 2002) (Andra et al., 2000) (Andra et al., 2002) (Diou et al., 2001) (Chen, 2002) (Varshney et al., 2007) (Huang et al., 2004) (Tan & Arslan, 2003) (Seo & Kim, 2007) (Huang et al., 2005) (Wu & Lin, 2005) (Lan et al., 2005) (Wu. & Chen, 2001). The computation and the access of the memory may take time and therefore the latency is long. Since the memory size of N^2 is a large quantity, here we try to use the 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).

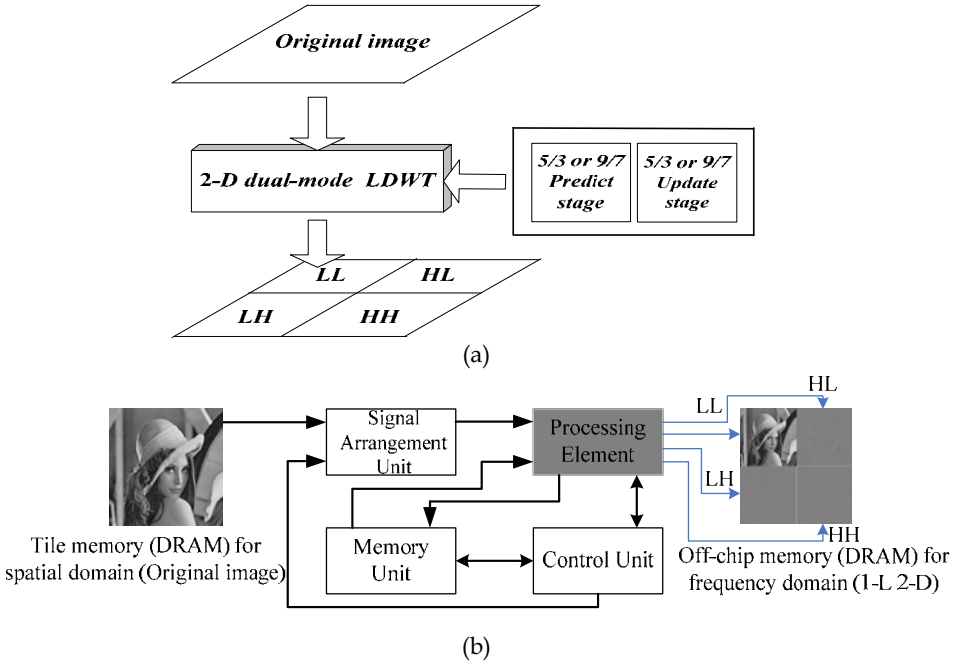
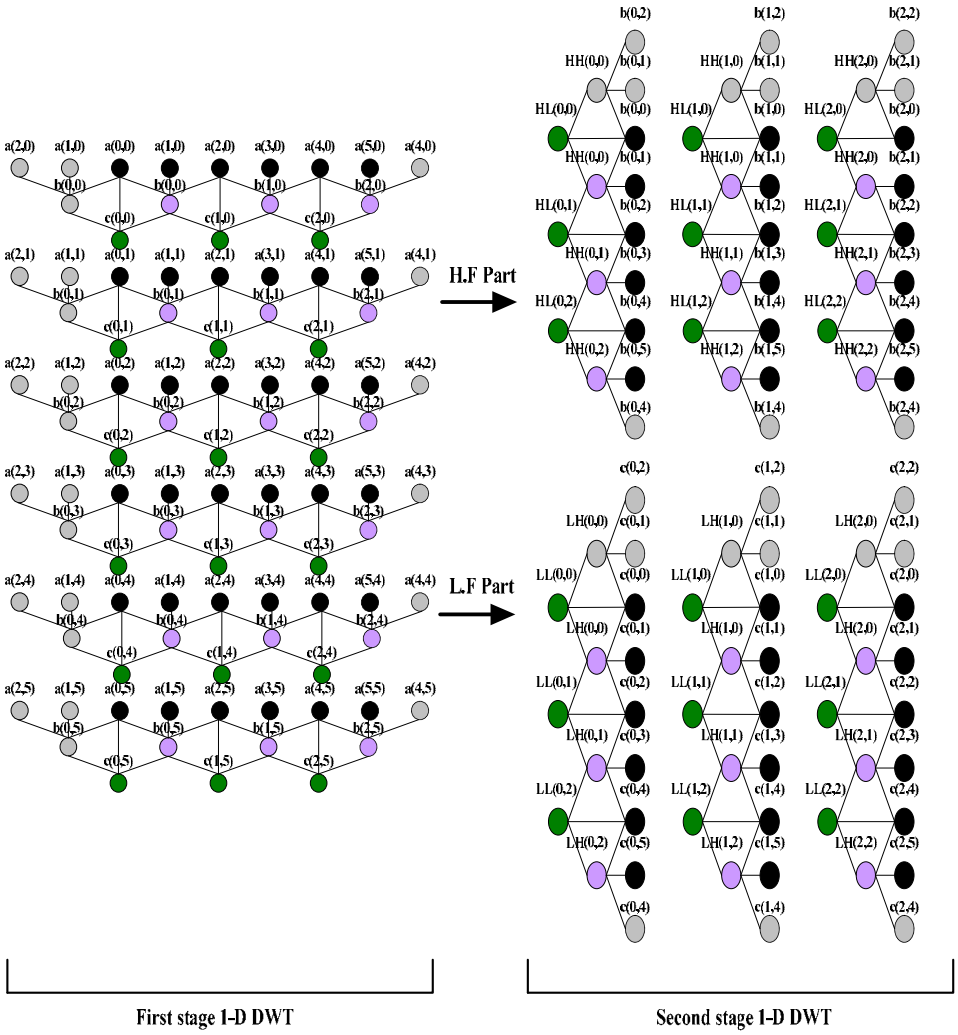


Fig. 7. The system block diagram of the proposed 2-D DWT. (a) 2-D dual-mode LDWT. (b) Block diagram of the proposed system architecture.



$x(i,j)$: 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. 8. Example of 2-D 5/3 mode LDWT operations.

Without loss of generality, let us take a 6×6-pixel image to describe the 2-D 5/3 mode LDWT operation and IRSA. Fig. 8 shows the operation diagram of the 2-D 5/3 mode LDWT operations of a 6×6 image. In Fig. 8, $x(i,j)$, 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 were described in the previous section. The left half of Fig. 8 shows the first stage 1-D DWT operations. The right half of Fig. 8 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 $b(0,0)$, $b(0,0) = -[x(0,0) + x(0,2)]/2 + x(0,1)$. To calculate the next high-frequency coefficient $b(0,1)$, we need pixels $x(0,2)$, $x(0,3)$, and $x(0,4)$. Here $x(0,2)$ is used to calculate both $b(0,0)$ 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 $b(0,1)$ are together 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 of Fig. 8, 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 three column consecutive $b(i,j)$ signals. For two column consecutive HH coefficients it has 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, it needs 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 it has an overlapped $c(i,j)$ signal. For example, $c(0,3)$ is the overlapped signal for computing LH(0,0) and LH(0,1). To compute LL coefficients, it needs 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 $c(0,2)$, $LL(0,1) = [LH(0,0) + LH(0,1)]/4 + c(0,2)$. The detail calculation equations for the four subband coefficients are summarized in the following equations:

$$HH(h,v) = x(2h+1,2v+1)+(1/4)\sum_{s=0}^{1}\sum_{t=0}^{1} x(2h+2s,2v+2t)+(-1/2)\sum_{s=-1}^{2} x(2h+|s|,2v+|-1+s|). \quad (14)$$

$$\begin{aligned} HL(h,v) &= (1/4)[HH(h,v-1)+HH(h,v)]+b(h,2v) \\ &= (1/4)[HH(h,v-1)+HH(h,v)]+(-1/2)[x(2h,2v)+x(2h+2,2v)]+x(2h+1,2v). \end{aligned} \quad (15)$$

$$LH(h,v) = (1/4)[HH(h-1,v)+HH(h,v)]+(-1/2)[x(2h,2v)+x(2h,2v+2)]+x(2h,2v+1). \quad (16)$$

$$\begin{aligned} LL(h,v) &= (1/4)[LH(h,v-1)+LH(h,v)]+c(h,2v) \\ &= (1/4)[LH(h,v-1)+LH(h,v)]+(1/4)[b(h-1,2v)+b(h,2v)]+x(2h,2v) \\ &= (1/4)[LH(h,v-1)+LH(h,v)]+(1/4)[(-1/2)x(2h-2,2v)+x(2h-1,2v)+(-1)x(2h,2v)+x(2h+1,2v)+(-1/2)x(2h+2,2v)]+x(2h,2v). \end{aligned} \quad (17)$$

The parameters in the above equations are defined as follows:

h: horizontal row, *v*: vertical column, *x*: original image, b: high-frequency wavelet coefficient of 1-D DWT, c: low-frequency wavelet coefficient of 1-D DWT, –1/2: Prediction parameter, and 1/4: Updating parameter.

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 (First-In First-Out, 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 a size of $2N$ as shown in Fig. 9. IRSA is based on this idea, and it can reduce the requirement of the transpose memory significantly. The block diagram of IRSA with several pixels of an image is shown in Fig. 9. In Fig. 9, the numbers on top and left represent the coordinate indexes of a 2-D image. In order to increase the operation speed, IRSA scans two pixels in the consecutive rows a time. IN1 (Initial in $x(0,0)$) and IN2 (Initial in $x(1,0)$) are the scanning inputs at 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 processor 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 processor 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 processor 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 processor scans the subsequent two 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 the DWT processor scans row 0 and row 1 and makes N1 and N2 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 processor uses pixels $x(0,3)$, $x(0,4)$, and $x(0,2)$, which was stored previously, to compute the high-frequency coefficient $b(0,1)$. Simultaneously the DWT processor uses pixels $x(1,3)$, $x(1,4)$, and $x(1,2)$, which was stored previously, to compute the high-frequency coefficients $b(1,1)$. 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 $c(0,1)$; $b(1,0)$, $b(1,1)$, 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 read the data in rows 2 and 3 to process the same operations until the end of the last pixel. The detail operations are shown in Fig. 10.

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 and the overlapped $b(i,j)$'s can be combined 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 and the overlapped $c(i,j)$ are used to compute an LL coefficient. The detailed operations for the second stage 1-D DWT are shown in Fig. 11.

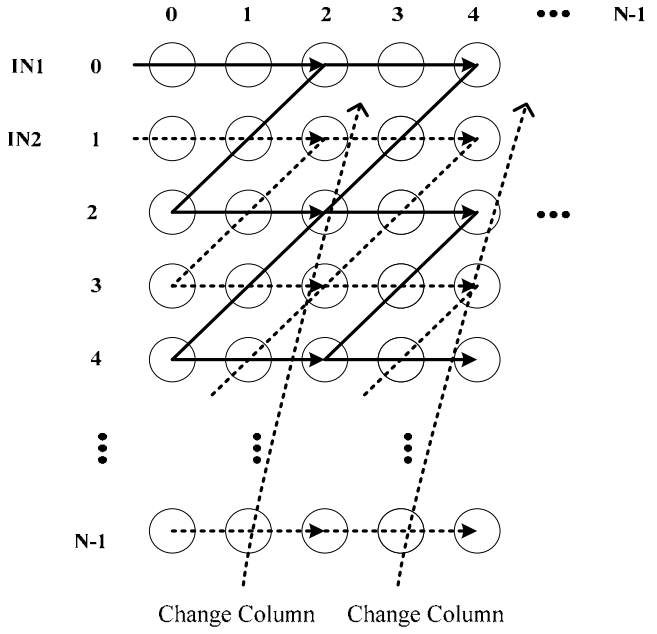


Fig. 9. IRSA of the 2-D LDWT.

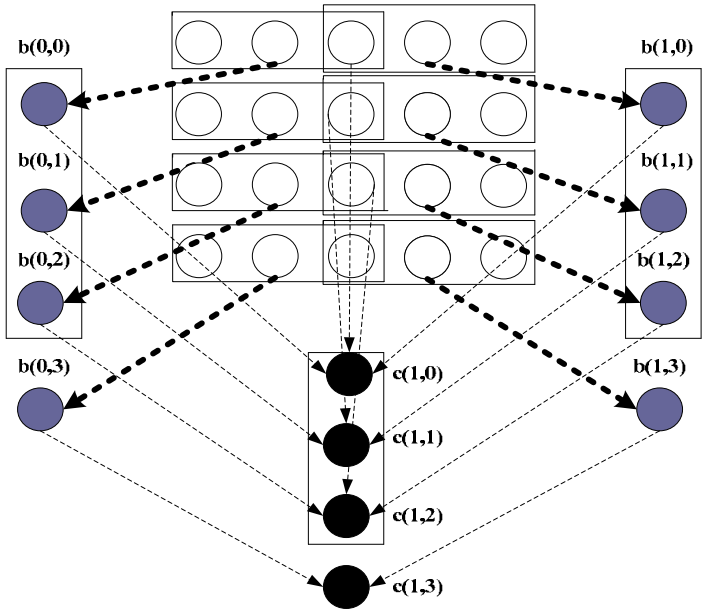
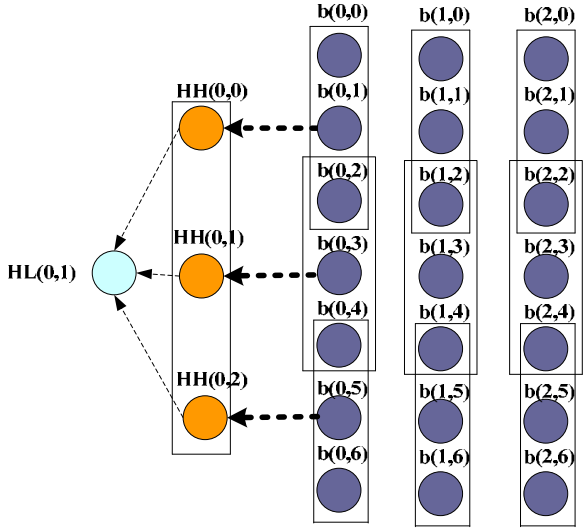
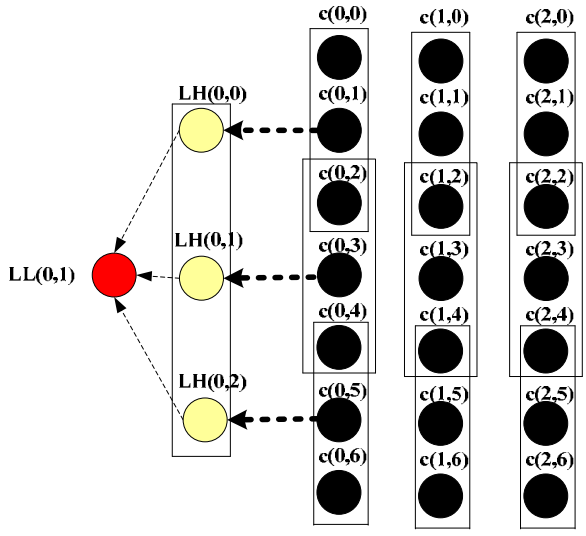


Fig. 10. The detail operations of the first stage 1-D DWT.



(a)



(b)

Fig. 11. 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.

5. VLSI architecture and implementation for the 2-D dual-mode LDWT

The IRSA approach has been discussed 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. 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.

5.1 The first stage 1-D LDWT

The first stage 1-D LDWT architecture consists of the following units: signal arrangement unit, multiplication and accumulation cell (MAC), multiplexer (MUX), and FIFO register. The block diagram is shown in Fig. 12.

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. The signal arrangement unit is shown in Fig. 13. In Fig. 13 MAC operates at the clock with gray circles.