Abstract—This paper presents a systematic high-speed VLSI implementation of the discrete wavelet transform (DWT) based on hardware-efficient parallel FIR filter structures. High-speed 2-D DWT with computation time as low as \(N^2/12\) can be easily achieved for an \(N \times N\) image with controlled increase of hardware cost. Compared with recently published 2-D DWT architectures with computation time of \(N^2/3\) and \(2N^2/3\), the proposed designs can also save a large amount of multipliers and/or storage elements. It can also be used to implement those 2-D DWT traditionally suitable for lifting or flipping-based designs, such as (9,7) and (6,10) DWT. The throughput rate can be improved by a factor of 4 by the proposed approach, but the hardware cost increases by a factor of around 3. Furthermore, the proposed designs have very simple control signals, regular structures and 100% hardware utilization for continuous images.

Index Terms—Cyclic convolution, discrete wavelet transforms (DWTs), linear convolution, very-large-scale integration (VLSI).

I. INTRODUCTION

THE discrete wavelet transform (DWT) has been widely used in audio and image processing, digital communications and other application fields. This computation transform has been widely implemented in very-large-scale integration (VLSI) [1]–[14] because of the real-time requirement. DWT has traditionally been implemented by convolution or FIR filter bank-based structures [1]–[9]. For a \(N \times N\) image, direct implementation of its 2-D DWT requires a computing time of \(4N^2\) [5]. In the past, many hardware efficient fast 2-D VLSI DWT architectures have been presented [1]–[4], [6]–[9]. Lifting- and flipping-based DWT implementations [10], [11], [13], [14] have many advantages and have recently been proposed for the JPEG2000 standard for image compression. For example, they often require fewer computations, compared to the convolution or FIR filter bank-based DWT. More detailed discussion on recent lifting-based implementations can be found in [12].

All these reported DWT architectures are usually designed for a certain processing speed and cannot be easily extended to achieve a higher processing speed. For example, as far as the authors know, the current high-speed convolution-based and lifting-based 2-D DWT VLSI architectures can achieve the lowest computing time of \(N^2/3\) [1] and \(N^2 + N^2\) [11], respectively.

When the processing speed is improved, we must also control the increase of hardware cost. A new 2-D DWT VLSI implementation scheme is proposed in this paper. Fast algorithm-based parallel FIR filter structures are designed to improve the processing speed and control the increase of the hardware cost at the same time.

The proposed design can reduce the computation time of the reported fastest 2-D DWT architectures [1] with filter length 4 from \(N^2/3\) to \(N^2/12\), but the number of required multipliers is only 3 times that of [1]. Higher processing speed can be achieved when parallel FIR filters with higher parallelism levels [15] are used. Furthermore, since the filtering structures are regular, the control signals are very simple. A generalized high-speed 2-D DWT structure is presented in this paper and can be applied to high-speed implementation of those 2-D DWT traditionally suitable for lifting and flipping-based schemes.

This paper is organized as follows. In Section II, we present the proposed fast 1-D DWT structures based on parallel FIR filter structures. The row dimension filtering and downsampling by two operations are transformed into two FIR filters each with a filter length of half of the original filters. In Section III, the proposed 2-D nonseparable DWT structures are obtained by first transforming the column dimension filtering and downsampling by two operations into two FIR filters each with a filtering length of half of the original FIR filters. Two-dimensional DWT structures with different processing speeds are generalized and illustrated by an example of \(8 \times 8\) image and filter length 4 in Section III. The proposed 2-D DWT structures can be easily extended to 2-D DWT with image size of \(N \times N\) and filter lengths of any numbers. Comparison and analysis are presented in Section IV. A comparison with lifting and flipping-based DWT implementation is also presented in Section IV.

II. PROPOSED FAST 1-D DWT STRUCTURES BASED ON PARALLEL FIR FILTER

The 2-D DWT consists of computing the 1-D DWT of each of the \(N\) rows of the original \(N \times N\) image and then computing the 1-D DWT of each of the resulting \(N\) columns [5]. Thus, we start with 1-D DWT. To illustrate our proposed DWT structure, we assume that the filter length be 4 and the original image size \(N \times N = 8 \times 8\).

Low-pass filter: \(H = \{a, b, c, d\}\); high-pass filter: \(G = \{e, f, g, h\}\); original image

\[
X = \begin{bmatrix}
x_{00} & x_{01} & x_{02} & x_{03} & x_{04} & x_{05} & x_{06} & x_{07} \\
x_{10} & x_{11} & x_{12} & x_{13} & x_{14} & x_{15} & x_{16} & x_{17} \\
x_{20} & x_{21} & x_{22} & x_{23} & x_{24} & x_{25} & x_{26} & x_{27} \\
x_{30} & x_{31} & x_{32} & x_{33} & x_{34} & x_{35} & x_{36} & x_{37} \\
x_{40} & x_{41} & x_{42} & x_{43} & x_{44} & x_{45} & x_{46} & x_{47} \\
x_{50} & x_{51} & x_{52} & x_{53} & x_{54} & x_{55} & x_{56} & x_{57} \\
x_{60} & x_{61} & x_{62} & x_{63} & x_{64} & x_{65} & x_{66} & x_{67} \\
x_{70} & x_{71} & x_{72} & x_{73} & x_{74} & x_{75} & x_{76} & x_{77} \\
\end{bmatrix},
\]
We first apply low-pass 1-D DWT to (1) in the row dimension. After low-pass filtering and downsampling by 2, we can get (2) from (1) [see (2), shown at the bottom of the page].

An efficient 1-D DWT decimation filter has been proposed in [2] and [3]. Although this decimation filter can save the number of both multipliers by a half, it has two drawbacks: 1) the input sampling frequency must be two times as fast as the output frequency, in order to get an output at every clock cycle; 2) this decimation filter cannot be easily operated at a higher processing speed.

Equation (2) can be simplified as (2a), shown at the bottom of the page, where $i = 0, 1, \ldots, 7$. Equation (2a) can be transformed into matrix form and represented as (2b).

\[
\begin{bmatrix}
  m_{i0} \\
  m_{i1} \\
  m_{i2} \\
  m_{i3}
\end{bmatrix} = \begin{bmatrix}
  0 & 0 & x_0 & x_1 \\
  x_2 & x_3 & x_4 & x_5 \\
  x_6 & x_7 & x_8 & x_9 \\
  x_{10} & x_{11} & x_{12} & x_{13}
\end{bmatrix}
\begin{bmatrix}
  d \\
  c \\
  b \\
  a
\end{bmatrix}
\]

From (2b), we can see that 1-D DWT (2a) has been transformed into two FIR filters each with a filter length of half of the original filter.

If we apply 2-parallel FIR filter structure to (2b), the computation of $[m_{i0} \ m_{i1} \ m_{i2} \ m_{i3}]$ requires just two clock cycles, with $[m_{i0} \ m_{i1}]$ coming out first and then $[m_{i2} \ m_{i3}]$ coming out after two clock cycles; the row filtering of (1) can be completed by the architecture shown in Fig. 1.

\[
\begin{bmatrix}
  bx_0 + ax_1 \\
  bx_1 + ax_{11} \\
  bx_2 + ax_{21} \\
  bx_{20} + ax_{21} \\
  bx_{30} + ax_{31} \\
  bx_{30} + ax_{31} \\
  bx_{40} + ax_{41} \\
  bx_{40} + ax_{41} \\
  bx_{50} + ax_{51} \\
  bx_{50} + ax_{51} \\
  bx_{60} + ax_{61} \\
  bx_{60} + ax_{61} \\
  bx_{70} + ax_{71}
\end{bmatrix}

\[
= \begin{bmatrix}
  m_{i0} & m_{i1} & m_{i2} & m_{i3} \\
  m_{i0} & m_{i1} & m_{i2} & m_{i3} \\
  m_{i0} & m_{i1} & m_{i2} & m_{i3} \\
  m_{i0} & m_{i1} & m_{i2} & m_{i3} \\
  m_{i0} & m_{i1} & m_{i2} & m_{i3} \\
  m_{i0} & m_{i1} & m_{i2} & m_{i3} \\
  m_{i0} & m_{i1} & m_{i2} & m_{i3}
\end{bmatrix}
\]

\[
(2a)
\]

\[
2\text{-parallel 2-tap FIR filter with coefficients } \{b, d\} \text{ can be represented as}
\]

\[
Y_2 = P_2^T \cdot H_2 \cdot Q_2^T \cdot X_2
\]

where outputs $Y_2 = [y(2k+1) \ y(2k)]^T$
inputs $X_2 = [x(2k+1) \ x(2k) \ x(2k-1)]^T$

\[
P_2 = \begin{bmatrix}
  1 & 0 \\
  1 & 1 \\
  0 & 1
\end{bmatrix},
Q_2 = \begin{bmatrix}
  1 & 0 & 0 \\
  0 & 1 & -1 \\
  0 & 0 & 1
\end{bmatrix}.
\]
The required number of multipliers, adders and delay elements for a 2-parallel 2-tap FIR filter are 3, 4, and 1, respectively.

Note that “4D” in Fig. 1(a) is used to replace every “D” in the original parallel FIR filter shown in Fig. 1(b) because the parallel filter is shared by four rows of input data, which are processed as shown in Fig. 2. After we finish filtering the first through fourth columns of the first through fourth rows of (1), we filter the fifth through eighth columns of the first through fourth rows (1). After we finish filtering the first through fourth rows, we filter the first through fourth rows starting from the first through fourth columns. In Fig. 2, those \( x_{ij} \) included in the same dashed line are input at the same clock cycle.

The corresponding output data flow of \( m_{ij} \) is shown in Fig. 3. In Fig. 3, those \( m_{ij} \) included in the same dashed line are output at the same clock cycle.

If we apply 4-parallel FIR filter structure to (2b), the computation of requires just one clock cycle and the row filtering of (1) can be performed by the architecture shown in Fig. 4.

4-parallel 2-tap FIR filter with coefficients \( \{b, d\} \) can be represented as follows:

\[
Y_4 = (P_{21}^T \otimes P_2^T) \cdot H_4 \cdot (Q_1^T \otimes Q_2^T) \cdot A_{4,221}^T \cdot X_4
\]

(4)

where \( \otimes \) is the tensor product (see the equation at the bottom of the page). The required number of multipliers, adders and delay elements for a 4-parallel 2-tap FIR filter is 6, 6 + 7 = 13 and 1, respectively.

Note that “8D” in Fig. 4(a) is used to replace every “D” in the original parallel FIR filter shown in Fig. 4(b) because the
the computation after we transform the 1-D DWT into two FIR filters. Since the parallel FIR filters used in [10] are based on fast convolution algorithms, the increase of hardware cost can be efficiently controlled.

III. PROPOSED 2-D NONSEPARABLE DWT STRUCTURE

A. Two-Dimensional Nonseparable DWT

We then apply high-pass 1-D DWT to (2) in column dimension. After high-pass filtering and downsampling by 2, we can get (5) from (2) [see (5), shown at the bottom of the page]. If we apply low-pass 1-D DWT to (2) in column dimension, after low-pass filtering and downsampling by 2, we can get (6) from (2) [see (6), shown at the bottom of the page]. It is obvious that the computations of the first row of (5) and of (6) require that of the first and second rows of (2); the second rows of (5) and (6) require that of the third and fourth rows and previously computed first and second rows of (2); the computation of the $i$th row of (5) and (6) requires that of $(2i - 1)$th and $(2i)$th and previously computed $(2i - 3)$rd and $(2i - 2)$nd rows of (2). Equation (5) can be simplified as

$$HG_{ij} = \begin{bmatrix} \frac{f_m_{0j} + e_m_{1j}}{h_{m_{0j}} + g_{m_{1j}} + f_{m_{2j}} + e_{m_{3j}}} \\ \frac{h_{m_{2j}} + g_{m_{3j}} + f_{m_{4j}} + e_{m_{5j}}}{h_{m_{ij}} + g_{m_{ij}} + f_{m_{ij}} + e_{m_{ij}}} \\ \frac{m_{0j} + e_{m_{0j}}}{m_{2j}} \\ \frac{m_{4j} + e_{m_{4j}}}{m_{6j}} \end{bmatrix} \begin{bmatrix} f \\ h \\ e \\ g \end{bmatrix}$$

(7)

where $j = 0, 1, \ldots, 3$.

$$HG = \begin{bmatrix} \frac{f_{m_{00}} + e_{m_{10}}}{h_{m_{00}} + g_{m_{10}} + f_{m_{20}} + e_{m_{30}}} & \frac{f_{m_{01}} + e_{m_{11}}}{h_{m_{01}} + g_{m_{11}} + f_{m_{21}} + e_{m_{31}}} & \frac{f_{m_{02}} + e_{m_{12}}}{h_{m_{02}} + g_{m_{12}} + f_{m_{22}} + e_{m_{32}}} & \frac{f_{m_{03}} + e_{m_{13}}}{h_{m_{03}} + g_{m_{13}} + f_{m_{23}} + e_{m_{33}}} \\ \frac{h_{m_{40}} + g_{m_{40}} + f_{m_{40}} + e_{m_{40}}}{h_{m_{40}} + g_{m_{40}} + f_{m_{40}} + e_{m_{40}}} & \frac{h_{m_{41}} + g_{m_{41}} + f_{m_{41}} + e_{m_{41}}}{h_{m_{41}} + g_{m_{41}} + f_{m_{41}} + e_{m_{41}}} & \frac{h_{m_{42}} + g_{m_{42}} + f_{m_{42}} + e_{m_{42}}}{h_{m_{42}} + g_{m_{42}} + f_{m_{42}} + e_{m_{42}}} & \frac{h_{m_{43}} + g_{m_{43}} + f_{m_{43}} + e_{m_{43}}}{h_{m_{43}} + g_{m_{43}} + f_{m_{43}} + e_{m_{43}}} \end{bmatrix}$$

(5)

$$HH = \begin{bmatrix} \frac{b_{m_{00}} + e_{m_{10}}}{d_{m_{00}} + c_{m_{10}} + b_{m_{20}} + a_{m_{30}}} & \frac{b_{m_{01}} + e_{m_{11}}}{d_{m_{01}} + c_{m_{11}} + b_{m_{21}} + a_{m_{31}}} & \frac{b_{m_{02}} + e_{m_{12}}}{d_{m_{02}} + c_{m_{12}} + b_{m_{22}} + a_{m_{32}}} & \frac{b_{m_{03}} + e_{m_{13}}}{d_{m_{03}} + c_{m_{13}} + b_{m_{23}} + a_{m_{33}}} \\ \frac{d_{m_{40}} + c_{m_{40}} + b_{m_{40}} + a_{m_{40}}}{d_{m_{40}} + c_{m_{40}} + b_{m_{40}} + a_{m_{40}}} & \frac{d_{m_{41}} + c_{m_{41}} + b_{m_{41}} + a_{m_{41}}}{d_{m_{41}} + c_{m_{41}} + b_{m_{41}} + a_{m_{41}}} & \frac{d_{m_{42}} + c_{m_{42}} + b_{m_{42}} + a_{m_{42}}}{d_{m_{42}} + c_{m_{42}} + b_{m_{42}} + a_{m_{42}}} & \frac{d_{m_{43}} + c_{m_{43}} + b_{m_{43}} + a_{m_{43}}}{d_{m_{43}} + c_{m_{43}} + b_{m_{43}} + a_{m_{43}}} \end{bmatrix}$$

(6)
Equation (6) can be simplified as (8)

\[
\begin{bmatrix}
HH_{0j} \\
HH_{1j} \\
HH_{2j} \\
HH_{3j}
\end{bmatrix} = \begin{bmatrix}
bm_{0j} + am_{1j} \\
dm_{0j} + cm_{1j} + bm_{2j} + am_{3j} \\
dm_{1j} + cm_{2j} + bm_{4j} + am_{5j} \\
dm_{3j} + cm_{5j} + bm_{6j} + am_{7j}
\end{bmatrix}
\begin{bmatrix}
b \\
d
\end{bmatrix}
\begin{bmatrix}
m_{0j} \\
m_{2j} \\
m_{4j} + m_{6j}
\end{bmatrix}
+ \begin{bmatrix}
m_{1j} \\
m_{3j} + m_{5j} \\
m_{5j} + m_{7j}
\end{bmatrix}
\begin{bmatrix}
a \\
c
\end{bmatrix}.
\]

(8)

From (7) and (8), we can see that the computations of (5) and (6) have each been transformed into two FIR filters each with a filter length of half of the original filter.

**B. Computing the First Resolution Level of an \( N \times N \) Image in \( N^2/2 \) Clock Cycles (\( N^2/2 \) DWT Structure)**

If we use two sequential FIR filters, we can compute one column element in each row of (2) at each clock cycle. The input data flow is shown in Fig. 7. After we finish the computation of the first column of the first and second rows in (2), we compute the second column of the first and second rows in (2).

If we use two sequential FIR filters with \( m_{i,j} \) as input, we can compute one row element in each column of (7) or (8) in one clock cycle. The only problem is that the computation of (2) with two sequential FIR filters outputs one data \( m_{1j} \) in one clock cycle, which has been shown in Fig. 8, but that of (7) or (8) with two 2-parallel FIR filters requires two data \( \begin{bmatrix} m_{0j} & m_{1j} \\ m_{2j} & m_{3j} \end{bmatrix} \), \( \begin{bmatrix} m_{4j} & m_{5j} \\ m_{6j} & m_{7j} \end{bmatrix} \), and \( \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \) as input in each clock cycle, which should have a data flow pattern as shown in Fig. 9. A \( 2 \times 2 \) delay element matrix (DEM) similar to Fig. 10(b) is used to solve this problem.

The proposed 2-D nonseparable DWT structure for the image size of \( 8 \times 8 \), which computes the first resolution level of an \( N \times N \) image in \( N^2/2 \) clock cycles, is shown in Fig. 10.

The data flow of the delay element matrix in Fig. 10(b) is “horizontal in, vertical out” or “vertical in, horizontal out” and is controlled by \( C_0, C_1, \) and \( C_2 \) signals. \( C_0 \) signal controls whether the data are “horizontal in” or “vertical in.” \( C_1 \) signal controls whether the data are “horizontal out” or “vertical out.” \( C_2 \) signal controls whether the data flow horizontally or vertically in the delay element matrix.

In Fig. 10, \( m_{i,j} \) is the first level filtering output of high-pass filter \( \{ e, f, g, h \} \). The output data flow of the \( 2 \times 2 \) DEM is...
shown is Fig. 11. Although both $m_{ij}$ and $m'_{ij}$ are generated by one at each clock cycle, after interleaving them as the input to $2 \times 2$ DEM, we can get two of them in parallel from the $2 \times 2$ DEM in each cycle.

From Fig. 10, we can see that the proposed 2-D nonseparable DWT can finish computing the first-resolution-level $HG$, $HH$, $GH$ and $GG$ of an $8 \times 8$ image in 32 clock cycles.

Note that the two input data to the sequential filters $\{a, c\}$ are the same as those to $\{e, g\}$. Thus, these two sequential filters can share the delay elements if the direction implementation of FIR filters is used. This sharing can also be applied to the sequential filters $\{b, d\}$ and $\{f, h\}$. This sharing can lead to the saving of $2 \times 2 = 4$ delay elements of the first level computation and $8 \times 2 = 16$ delay elements of the second level computation.

The total hardware cost of this design is 16 multiplications, 12 additions, and 24 delay elements. In general, the computation time for the first-resolution-level 2-D DWT of an $N \times N$ image is $N^2/2$ clock cycles.

C. Computing the First Level of an $N \times N$ Image in $N^2/4$ Clock Cycles ($N^2/4$ DWT Structure)

If we use two 2-parallel FIR filter as shown in Fig. 1, we can compute two column elements in each row of (2) at each clock cycle; and then if we use two 2-parallel FIR filters with $m_{ij}$ as input, we can compute four row elements in each column of (7) or (8) in one clock cycle. The only problem is that the computation of (2) with two 2-parallel FIR filters outputs two data $[m_{00} m_{01}]$ or $[m_{12} m_{13}]$ in one clock cycle, which has been shown in Fig. 3, but that of (7) or (8) with two 2-parallel FIR filters requires four data $[m_{0j} m_{1j} m_{2j} m_{3j}]$ as input in each clock cycle. A $4 \times 4$ DEM similar to Fig. 8(b) is used to solve this problem.

The proposed 2-D nonseparable DWT structure for image size of $8 \times 8$, which computes the first resolution level of an $N \times N$ image in $N^2/4$ clock cycles, is shown in Fig. 12. The $4 \times 4$ DEM in Fig. 12(b) works in the same way as the $2 \times 2$ DEM in Fig. 10(b). The delay elements in the $4 \times 4$ DEM have the same functionality as shown in Fig. 10(c).

From Fig. 10, we can see that the proposed 2-D nonseparable DWT can finish computing the first-resolution-level $HG$, $HH$, $GH$ and $GG$ of an $8 \times 8$ image in 16 clock cycles.

Output data flow of $4 \times 4$ delay element matrix is shown in Fig. 13.

Note that the two input data to the 2-parallel filter $\{a, c\}$ are the same as those to $\{e, g\}$; thus, these two 2-parallel filters can share the same $Q_2$ block and delay element of the input side as shown in Fig. 1(b) and four addition operations can be saved. Another four adders can be saved from sharing the same input data to the two 2-parallel filters $\{b, d\}$ and $\{f, h\}$. This sharing can also lead to the saving of $4 \times 2 = 8$ delay elements of the first level and $8 \times 2 = 16$ delay elements of the second level.

The total hardware cost of this architecture is 24 multiplications, 32 additions, and 40 delay elements. In general, the computation time for the first-resolution-level 2-D DWT of an $N \times N$ image is $N^2/4$ clock cycles.

D. Computing the First Level 2-D DWT of an $N \times N$ Image By Applying L- Parallel Fir Filtering

From the discussion shown above, we can generalize our proposed algorithm for computing the first level 2-D DWT of an $N \times N$ image by applying L-parallel FIR filtering.

For a 2-D DWT with low-pass filter $H$ and high-pass filter $G$, we first decimate $H$ and $G$ by factor 2 into $\{H_0, H_1\}$ and $\{G_0, G_1\}$. $H_0$, $H_1$, $G_0$ and $G_1$ are all subfilters. The proposed 2-D nonseparable DWT structure, which computes the first-resolution-level of an $N \times N$ image in $N^2/(2L)$ clock cycles, is shown in Fig. 14.

From Fig. 14, we can see that subfilters $H_0$ and $G_0$ share the same input data for both the first and the second level of computation. Subfilters $H_1$ and $G_1$ also share the same data for both levels of computation. This property can save large number of delay elements especially for the second level of computation,
because the saving of each delay element in the second level filter structure will lead to the saving of \( N \) delay elements of the final 2-D DWT structure. When parallelism level \( L \) is greater than 1, adders can also be saved.

E. Proposed 2-D Nonseparable DWT Structure

Using the proposed first-level 2-D nonseparable DWT structures in subsections B-D, we can get our proposed 2-D nonseparable DWT structure with higher resolution levels.

For example, let the resolution level be \( J \) and we compute the DWT of an \( N \times N \) image with the \( N^2/4 \) DWT structure discussed in subsection C. The first resolution level of \( N^2 \) input data can be computed in \( 1/4 \times (N/2)^2 \) clock cycles. The second resolution level of \( N^2/4 \) data can be computed in \( 1/4 \times (N/2)^2 \) clock cycles, and so on. Thus, the total computational time is given by

\[
\frac{N^2}{4} + \frac{1}{4} \times \left( \frac{N}{2} \right)^2 + \frac{1}{4} \times \left( \frac{N}{2} \right)^2 + \cdots = \frac{N^2}{4} \times \frac{4}{3} = \frac{N^2}{3}.
\]

This example is discussed in detail in the following example.

For an \( 8 \times 8 \) image in (1), the hardware implementation of its 2-D DWT with \( J \)-level resolution and a computation time of \( N^2/3 \) is shown in Fig. 15. We now explain how the proposed 2-D DWT structure in Fig. 15 works.

The outputs \( HH_{1}^{i} \) of the first resolution level \( N^2/4 \) DWT structure are also the inputs of the second resolution level \( N^2/4 \) DWT structure. As shown in Fig. 12, \( HH_{1}^{i} \) from a \( N^2/4 \) DWT structure are generated in the sequence shown in Fig. 16.

Comparing the data flow in Fig. 16 with the input data flow of a \( N^2/4 \) DWT structure as shown in Fig. 2, we can see that we need \( N+8 \) storage elements for the output of the first resolution level \( HH \) before we can start the computation of the second resolution level 2-D DWT, and we need \( N/2+8 \) storage elements for the output of the second level \( HH \) before we can start the computation of the third level 2-D DWT, and so on. Thus, the total storage elements for \( HH^{i}, i < j \) is given by

\[
(N+8)+\left(\frac{N}{2}+8\right)+\left(\frac{N}{4}+8\right)+\cdots = 2N+8\times(J-1),
\]

Since we share the computation of different resolution levels of 2-D DWT, the computation of \( HH^{i} \) will be interrupted whenever higher level \( HH^{i+1} \) is ready to be fetched from the storage and computed. In order to resume the computation of lower level \( HH^{i} \), we need to save the intermediate computation results for \( HH^{i} \) and interleaving filtering is used. To interleave the filtering of the first-resolution-level \( N^2/4 \) DWT structure shown in Fig. 12, we can easily replace “4D” and “8D” in Fig. 12 with Fig. 17(a) and (b), respectively [16].

In Fig. 17, every pair of DEMUX and MUX switch to next channel \( i \) when the \( i \)th level of 2-D DWT is being computed.

The data flow of the proposed hardware implementation of the 2-D DWT of an \( 8 \times 8 \) image with two-level resolution and a computation time of \( N^2/3 \) is shown in Table I.

In Table I, \( x_{ij} \) and \( y_{ij} \) are from two different \( 8 \times 8 \) images. We assume these two images are processed in a row. \( HH_{10}^{i} \) and \( HH_{20}^{i} \) are the outputs of the first and second resolution levels of the image represented as \( x_{ij} \).

From Table I, we can see that there is a latency of 4 clock cycles between the input of \( HH_{10}^{i} \) and the output of \( HH_{20}^{i} \) because of the \( 4 \times 4 \) delay element matrix in the first level \( N^2/4 \) DWT structure as shown in Fig. 12.
TABLE I
DATA FLOW OF PROPOSED 2-D DWT OF A 8 × 8 IMAGE WITH 2-LEVEL RESOLUTION AND A COMPUTATION TIME OF N²/3

<table>
<thead>
<tr>
<th>t</th>
<th>HH</th>
<th>HG</th>
<th>GH</th>
<th>GG</th>
<th>Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>x₀₀ , x₀₁ , x₀₂ , x₀₃</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>x₁₀ , x₁₁ , x₁₂ , x₁₃</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>x₂₀ , x₂₁ , x₂₂ , x₂₃</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>x₃₀ , x₃₁ , x₃₂ , x₃₃</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>-</td>
<td>GH₁₀ , GH₁₁</td>
<td>GG₁₀ , GG₁₁</td>
<td>x₄₀ , x₄₁ , x₄₂ , x₄₃</td>
</tr>
<tr>
<td>5</td>
<td>HH₀₀ , HH₀₁</td>
<td>HG₀₀ , HG₀₁</td>
<td>GG₀₀ , GG₀₁</td>
<td>-</td>
<td>x₅₀ , x₅₁ , x₅₂ , x₅₃</td>
</tr>
<tr>
<td>6</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>x₆₀ , x₆₁ , x₆₂ , x₆₃</td>
</tr>
<tr>
<td>7</td>
<td>HH₀₁</td>
<td>HH₀₂</td>
<td>HG₀₁ , HG₀₂</td>
<td>GG₀₁ , GG₀₂</td>
<td>x₇₀ , x₇₁ , x₇₂ , x₇₃</td>
</tr>
<tr>
<td>8</td>
<td>-</td>
<td>-</td>
<td>GH₂₀ , GH₂₁</td>
<td>GG₂₀ , GG₂₁</td>
<td>x₈₀ , x₈₁ , x₈₂ , x₈₃</td>
</tr>
<tr>
<td>9</td>
<td>HH₂₀</td>
<td>HH₂₁</td>
<td>HG₂₀ , HG₂₁</td>
<td>-</td>
<td>x₉₀ , x₉₁ , x₉₂ , x₉₃</td>
</tr>
<tr>
<td>10</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>x₁₀₀ , x₁₀₁ , x₁₀₂ , x₁₀₃</td>
</tr>
<tr>
<td>11</td>
<td>HH₁₀</td>
<td>HH₁₁</td>
<td>HG₁₀ , HG₁₁</td>
<td>GG₁₀ , GG₁₁</td>
<td>x₁₁₀ , x₁₁₁ , x₁₁₂ , x₁₁₃</td>
</tr>
<tr>
<td>12</td>
<td>-</td>
<td>-</td>
<td>GH₃₀ , GH₃₁</td>
<td>GG₃₀ , GG₃₁</td>
<td>x₁₂₀ , x₁₂₁ , x₁₂₂ , x₁₂₃</td>
</tr>
<tr>
<td>13</td>
<td>HH₃₀</td>
<td>HH₃₁</td>
<td>HG₃₀ , HG₃₁</td>
<td>-</td>
<td>x₁₃₀ , x₁₃₁ , x₁₃₂ , x₁₃₃</td>
</tr>
<tr>
<td>14</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>x₁₄₀ , x₁₄₁ , x₁₄₂ , x₁₄₃</td>
</tr>
<tr>
<td>15</td>
<td>HH₁₂</td>
<td>HH₁₃</td>
<td>HG₁₂ , HG₁₃</td>
<td>-</td>
<td>x₁₅₀ , x₁₅₁ , x₁₅₂ , x₁₅₃</td>
</tr>
<tr>
<td>16</td>
<td>-</td>
<td>-</td>
<td>GH₄₀ , GH₄₁</td>
<td>GG₄₀ , GG₄₁</td>
<td>x₁₆₀ , x₁₆₁ , x₁₆₂ , x₁₆₃</td>
</tr>
<tr>
<td>17</td>
<td>HH₄₀</td>
<td>HH₄₁</td>
<td>HG₄₀ , HG₄₁</td>
<td>-</td>
<td>x₁₇₀ , x₁₇₁ , x₁₇₂ , x₁₇₃</td>
</tr>
<tr>
<td>18</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>x₁₈₀ , x₁₈₁ , x₁₈₂ , x₁₈₃</td>
</tr>
<tr>
<td>19</td>
<td>HH₂₃</td>
<td>HH₃₄</td>
<td>HG₂₃ , HG₃₄</td>
<td>-</td>
<td>x₁₉₀ , x₁₉₁ , x₁₉₂ , x₁₉₃</td>
</tr>
<tr>
<td>20</td>
<td>-</td>
<td>-</td>
<td>GH₅₀ , GH₅₁</td>
<td>GG₅₀ , GG₅₁</td>
<td>x₂₀₀ , x₂₀₁ , x₂₀₂ , x₂₀₃</td>
</tr>
<tr>
<td>21</td>
<td>HH₅₀</td>
<td>HH₅₁</td>
<td>HG₅₀ , HG₅₁</td>
<td>-</td>
<td>x₂₁₀ , x₂₁₁ , x₂₁₂ , x₂₁₃</td>
</tr>
<tr>
<td>22</td>
<td>-</td>
<td>-</td>
<td>GH₆₀ , GH₆₁</td>
<td>GG₆₀ , GG₆₁</td>
<td>x₂₂₀ , x₂₂₁ , x₂₂₂ , x₂₂₃</td>
</tr>
<tr>
<td>23</td>
<td>HH₆₀</td>
<td>HH₆₁</td>
<td>HG₆₀ , HG₆₁</td>
<td>-</td>
<td>x₂₃₀ , x₂₃₁ , x₂₃₂ , x₂₃₃</td>
</tr>
<tr>
<td>24</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>x₂₄₀ , x₂₄₁ , x₂₄₂ , x₂₄₃</td>
</tr>
<tr>
<td>25</td>
<td>HH₀₀</td>
<td>HH₁₀</td>
<td>HG₀₀ , HG₁₀</td>
<td>-</td>
<td>x₂₅₀ , x₂₅₁ , x₂₅₂ , x₂₅₃</td>
</tr>
<tr>
<td>26</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>x₂₆₀ , x₂₆₁ , x₂₆₂ , x₂₆₃</td>
</tr>
<tr>
<td>27</td>
<td>HH₀₁</td>
<td>HH₁₁</td>
<td>HG₀₁ , HG₁₁</td>
<td>-</td>
<td>x₂₇₀ , x₂₇₁ , x₂₇₂ , x₂₇₃</td>
</tr>
</tbody>
</table>

From Table I, we can also see that input data of the i-th resolution level will be interrupted by those of the (i+1)th resolution level, N/2⁻¹ = (2L)² clock cycles after the first input data of the i-th resolution level have been given. Only after the available data of resolution level higher than i are processed, the available data of resolution level i can be processed.

IV. ANALYSIS AND COMPARISON

From the above example, we can see that the first-resolution-level 2-D DWT structures with different processing speeds in Section III are used to design the proposed 2-D DWT with higher resolution levels. We summarize the hardware cost of these structures for the 2-D DWT of an N × N image in terms of the number of required multipliers (R.M.), adders (R.A.), and delay elements (R.D.) as

\[ R.M. = 8 \cdot M \left( \frac{K}{2}, L \right) \]
\[ R.A. = 8 \cdot A \left( \frac{K}{2}, L \right) + 4 \cdot L \]
\[ - \cdot A \left( Q_L^F, L \right) \]
\[ R.D. = 2 \cdot D \left( \frac{K}{2}, L \right) + (2L) + 4N \cdot D \left( \frac{K}{2}, L \right) + 4 \cdot L^2 \]

where \( M(K/2, L) \), \( A(K/2, L) \) and \( D(K/2, L) \) are the number of required multipliers, adders and delay elements for implementation of a L-parallel K/2-length FIR filter, respectively. Here, we have \( K = 4 \). \( A(Q_L^F, L) \) is the number of required adders for the \( Q_L^F \) blocks of an L-parallel K/2-length FIR filter.
The first-resolution-level DWT with higher processing speed can also be derived similarly based on \( L \)-parallel 2-tap FIR filter with higher parallelism level. For example, the \( N^2/8 \) and \( N^2/16 \) DWT structures can also be derived based on 4-parallel 2-tap FIR filter given in Section II and 8-parallel 2-tap FIR filter.

In some cases, hardware cost is an important concern. We can trade processing speed for hardware savings by using first-resolution-level DWT at lower processing speed than the proposed first-level DWT structure.

When these first-resolution-level 2-D DWT structures are used to implement DWT of higher resolution levels, the number of required multiplications and additions remain the same, but the number of delay elements and that of storage elements increase.

Comparison of the proposed 2-D DWT structures with previous structures is presented in Table II. From Table II, we can see that the proposed designs provide systematic high-speed 2-D DWT solutions and can be easily extended to achieve high-speed hardware implementations of 2-D DWT with controlled increase of hardware cost.

We also see that the proposed designs are hardware efficient. For example, among the designs with computing time \( N^2/3 \), the proposed design 3 can save eight multipliers at the cost of eight adders, and save more storage elements when the image size \( N \times N \) is large. It is obvious that our proposed design 4 can also save large amount of multiplications and storage elements when compared with other previous designs with computing time \( 2N^2/3 \). Furthermore, the hardware cost of our proposed design with computing time \( 2N^2/3 \) is even comparable with or is less than those previous designs with computing time \( N^2 \).

When the filter length is larger than 4, the parallel FIR filter structures [15] can be used to develop hardware efficient parallel FIR filter structures for the proposed 2-D DWT schemes. For example, when filter length is 8, an 8-parallel 4-tap FIR filter structure.

From Fig. 14, we can see that, the proposed high-speed 2-D DWT architectures have regular structures. The first level and second level have the same filter structure. Although the architecture is derived based on 2-D DWT with high-pass and low-pass filter of the same lengths, it can be applied to 2-D DWT of any filter lengths. The computation efficiency will be determined by that of the filtering structure used for both the first- and second-level DWT computation.

We summarize the hardware cost of the proposed structures for any 2-D DWT of an \( N \times N \) image in terms of the number of required multipliers (R.M.), adders (R.A.), and delay elements (R.D.) as

\[
R.M. = 2 \sum_{i=0}^{1} \left\{ M(H_i, L) + M(G_i, L) \right\}
\]

\[
R.A. = 2 \sum_{i=0}^{1} \left\{ A(H_i, L) + A(G_i, L) - A\left(Q_{H_i}^{T}, L\right) - A\left(Q_{G_i}^{T}, L\right) \right\} + 4 \cdot L
\]

\[
R.D. = [(2L) \cdot J + 2N] \sum_{i=0}^{1} \{ D(H_i, L) + D(G_i, L) \} + 4 \cdot L^2
\]

where \( M(H_i, L), A(H_i, L), \) and \( D(H_i, L) \) are the number of required multipliers, adders and delay elements for implementation of a \( L \)-parallel subfilter \( H_i \), respectively. \( A(Q_{H_i}^{T}, L) \) is the number of required adders for the \( Q_{H_i}^{T} \) block of a \( L \)-parallel subfilter \( H_i \) and \( J \) is the resolution level.

The total computation time is

\[
\frac{N^2}{2L} \times \left( \frac{1}{4} + \frac{1}{16} + \cdots \right) = \frac{2N^2}{3L}.
\]

For example, the proposed design can also be used to implement those 2-D DWT suitable for lifting- or flipping-based implementation schemes. A comparison between proposed design and lifting-based design for (9,7) and (6,10) 2-D DWT is given in Tables III and IV, respectively.

For (9,7) DWT with low-pass filter \( \{a, b, c, d, e, d, e, c, b, a\} \) and high-pass filter \( \{e, f, g, h, g, f, e, f\} \), we have the following relations: \( 2a + 2c + e = 0.5 \), \( b + d = 0.25 \), \( e_1 + g = -0.5 \) and \( 2f + h = 1 \). Multiplication by a number \( 2k \) just needs shift operation, which does not not require any hardware cost. These relations are combined with the symmetric property of the low- and high-pass filters to cut down the hardware cost.

For (6,10) DWT with low-pass filter \( \{f, g, h, h, g, f\} \) and high-pass filter \( \{a, b, c, d, e, c, d, b, c, a\} \), we have the following relations: \( g = f + 0.125 \) and \( h = 0.375 - 2f \). These relations are also combined with the symmetric property of the low- and high-pass filters to cut down the hardware cost.
From Tables III and IV, we can see that the proposed (9,7) and (6,10) 2-D DWT can reduce the traditional computation time to $4N^2/3$ to $N^2/3$ clock cycles and the hardware cost is controlled in a reasonable range. The throughput rate is improved by a factor of 4 but the hardware cost only increases by a factor of about 3.

**V. CONCLUSION**

A systematic scheme for high-speed VLSI implementation of 2-D DWT is presented in this paper. Hardware efficient parallel FIR filter structures are developed to speed up the processing speed of 2-D DWT and to control the increase of hardware cost at the same time. The proposed design can be easily extended to achieve higher processing speed than the given highest processing speed with computing time $N^2/12$ in this paper. The proposed design is suitable for high-speed VLSI implementation of 2-D DWT because of its regular structure, simple control and 100% hardware utilization for continuous images.

**REFERENCES**


APPENDIX. 8-PARALLEL 2-TAP AND 4-TAP FIR FILTERS

A. 8-parallel 2-tap FIR filter

8-parallel 2-tap FIR filter with coefficients \{b, d\} can be represented [15] as:

\[ Y_8 = (P_{21}^T \otimes (P_{21}^T \otimes P_{21}^T)) \cdot H_8 \cdot (Q_{21}^T \otimes (Q_{21}^T \otimes Q_{21}^T)) \cdot A_{8.222,1}^T \cdot X_8 \]  \hfill (A.1)

where, \( \otimes \) is the tensor product;

- outputs \( Y_8 = \left[ y(8k+7) \ y(8k+6) \ y(8k+5) \ y(8k+4) \ y(8k+3) \ y(8k+2) \ y(8k+1) \ y(8k) \right] \);  
- inputs \( X_8 = \left[ x(8k+7) \ x(8k+6) \ x(8k+5) \ x(8k+4) \ x(8k+3) \ x(8k+2) \ x(8k+1) \ x(8k) \right] \);

\( k = 0, 1, 2, \ldots \)

\( H_8 = \text{diag} \left[ b \ b \ b \ b \ d \ b \ b \ d \right] \);

\[ P_{21} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad P_2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad Q_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \quad A_{8.222,1}^T = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \]

The number of required multipliers, additions and delay elements for a 8-parallel 2-tap FIR filter is 12, 16 + 20 = 36 and 1, respectively.

B. 8-parallel 4-tap FIR filter

8-parallel 4-tap FIR filter with coefficients \{a, b, c, d\} can be represented as:

\[ Y_8 = (P_{21}^T \otimes (P_{21}^T \otimes P_{21}^T)) \cdot H_8 \cdot (Q_{21}^T \otimes (Q_{21}^T \otimes Q_{21}^T)) \cdot A_{8.222,2}^T \cdot X_8 \]  \hfill (B.1)

where, \( \otimes \) is the tensor product;

- outputs \( Y_8 = \left[ y(8k+7) \ y(8k+6) \ y(8k+5) \ y(8k+4) \ y(8k+3) \ y(8k+2) \ y(8k+1) \ y(8k) \right] \);
- inputs \( X_8 = \left[ x(8k+7) \ x(8k+6) \ x(8k+5) \ x(8k+4) \ x(8k+3) \ x(8k+2) \ x(8k+1) \ x(8k) \right] \);

\( k = 0, 1, 2, \ldots \)

\( H_8 = \text{diag} \left[ a \ a \ a \ a \ b \ b \ b \ b \ c \ c \ c \ c \ d \ d \ d \ d \ d \ d \right] \);
The number of required multipliers, additions and delay elements for a 8-parallel 2-tap FIR filter is 18, 24+33=57 and 3, respectively.