CHAPTER 3
PARALLEL ARCHITECTURE FOR MULTI-LEVEL LIFTING 2-D DWT WITHOUT USING DATA-SELECTOR

3.1 INTRODUCTION

Different data-access methods and DWT computation schemes (convolution, lifting and flipping) are considered to optimize the on-chip memory complexity of the parallel architecture. The parallel architecture of [Mohanty and Meher (2011)] uses line-based data-access method and lifting scheme which requires nearly $10N$ on-chip memory words, where N is the image width. An overlapped stripe-based parallel architecture is proposed in [Mohanty and Meher (2013)] using convolution scheme which requires $4.75N$ less on-chip memory than the parallel architecture of [Mohanty and Meher (2011)] for decomposition level J=2. However, the parallel architecture of [Mohanty and Meher (2011)] requires more arithmetic resources than those required by [Mohanty and Meher (2011)] and involves extra frame-memory in the input buffer. Recently, [Hu and Jong (2013)] have proposed a parallel architecture based on the flipping scheme and overlapped stripe based data access scheme similar to the structure of [Mohanty and Meher (2013)]. The structure of [Hu and Jong (2013)] requires less arithmetic resources (multiplier and adder) and less on-chip memory words than the convolution based structure of [Mohanty and Meher (2013)] because of the flipping scheme. Compared with the lifting-based architecture of [Mohanty and Meher (2011)], the flipping-based architecture of [Hu and Jong (2013)] offers a saving of $7N$ on-chip memory words due to overlapped stripe-based data-access but at the cost of an overhead which include extra arithmetic resources, data-selectors and $7N$ extra frame-buffer words. Since the frame-buffer is external to the chip, the saving in $7N$ on-chip memory words offers a significant saving of core area. Therefore, the structure of [Hu and Jong (2013)] is considered the best amongst the existing parallel architectures. However, there are few issues associated with architecture of [Hu and Jong (2013)] when the design is scaled for higher DWT level and/or high-throughput realization.

The parallel architecture of [Hu and Jong (2013)] is based on overlapped stripe-based data access method involves several multiplexer and data registers for reordering the intermediate data-blocks. Multiplexer contributes to the critical-path apart from the multiplier and adder. Flipping scheme is considered over the lifting scheme to compensate the multiplexer delay and pipeline latches are inserted aggressively to achieve smaller critical path delay in the design [Hu and Jong (2013)] Multiplexers and pipeline latches contributes significant to the chip area when the parallel architecture of [Hu and Jong (2013)] is implemented for higher block-sizes. Besides, the overlapped data access scheme introduces an extra 6 multipliers and 12 extra adders and 7*N* extra frame-buffer memory words to the design. In general, overlapped stripe-based data-access scheme offers a saving in 4*N* on-chip memory over the stripe-based data-access but introduces a significant amount of overhead complexity (in terms of multipliers, adders, multiplexers, data registers and frame-buffer words) and makes the design more complex. A simple and efficient architecture can be obtained without using overlapped stripe-based data-access method. In recent study, it is observed that scaling constants of lifting scheme are perfect inverse to each other where the flipping scheme does not have this feature [Mohanty *et al.* (2015)]. Therefore, the block-based lifting 2-D DWT structure involves (*S*/2) less multipliers than those of flipping-based structure for block-size *S*. A different type of data access scheme needs to be formulated to avoid multiplexing operation in data reordering of multi-level 2-D DWT. A design method needs to be considered to develop a parallel architecture with higher degree of regularity and modularity and free from overhead complexity unlike the existing architectures

The Chapter is organized as: The proposed data access scheme is presented in Section 3.2 Block-formulation of 2-D DWT is presented in Section 3.3 The proposed architecture is presented in Section 3.4 Hardware and time complexities are discussed in Section 3.5 Conclusions are presented in Section 3.6

3.2. DATA-ACCESS SCHEME

The input image of size ($N \times N$) is assumed to be stored in an input-buffer. A block samples are read from the input-buffer in each clock cycle and processed by the parallel architecture. Line-based, stripe-based, and overlapped stripe-based data-access methods are used to read data-

blocks from the input buffer. In line-based data-access method, successive input-blocks are read from a particular row while in stripe-based data-access method, successive input-blocks are read from successive rows. Both line-based and stripe-based data-access method involve identical type of computation except the design of transposition-memory. Using stripe-based method the transposition-memory of 2-D DWT structure is implemented using registers where line-buffers are required in case of line-based data-access method. Overlapped stripe-based method is considered in parallel designs of [Hu and Jong (2013)]. The stripe-based and overlapped stripe-based data-access methods are identical except that in overlapped stripe-based data-access method, adjacent data-blocks corresponding to a particular row are overlapped by few samples. The overlapped data-block helps to avoid temporal memory required for lifting computation. But, overlapped stripe-based data access introduces few redundant computation, redundant memory access and overhead memory to the input-buffer [Hu and Jong (2013)]. However, a common feature is observed between parallel architectures based on line-based [Mohanty and Meher (2011)], stripe-based and overlapped stripe-based method [Hu and Jong (2013)] that these designs involve multiplexers for intra-level and inter-level lifting 2-D DWT. Multiplexers are required by the design for time-multiplexing and/or folding the intermediate data-blocks. A different data-access scheme needs to be formulated to avoid the data multiplexing completely which is common in the existing parallel architectures of multi-level 2-D DWT.

3.2.1 STUDY ON DATA-FLOW OF MULTI-LEVEL LIFTING 2-D DWT

A study on the data-flow of lifting 2-D DWT structure is required to analyze the data arrangement in the reordering stage before formulating a new data-access scheme. A stripe-based data-access method is assumed to schedule input data-block of size 4 for computation of 2-D DWT. However, the same type of behavior also can be observed in line-based and overlapped stripe-based data-access method as well. As shown in Figure 3.1(a)-(b), in row-wise DWT four data-blocks are processed in four clock cycles such that during every clock cycle one input data-block of size 4 is processed and one block of low-pass and one block of high-pass intermediate are obtained of size 2 each. A pair of samples from a particular column is required to start the lifting computation along column-wise. Therefore, data-blocks of row-wise DWT cannot be processed column-wise directly and these blocks are reordered as shown Figure 3.1(c). The data

reordering operation of Figure 3.1(c) creates an idle clock cycle after every alternate clock cycle which is shown by symbol '×'. Also, four separate lifting computation need to be performed in parallel to process each data-blocks of Figure 3.1(c). The idle clock cycles of Figure 3.1(c) can be removed by scheduling a lifting computation of column-wise DWT. The data-blocks of low-pass [$\mathbf{u}_l$] and high-pass [$\mathbf{u}_h$] of Figure 3.1(c) are time-multiplexed as shown in Figure 3.2 (a) to utilize the idle clock cycle. Interestingly, the multiplexed data-blocks of Figure 3.2(a) requires 2 concurrent lifting computations per clock cycle instead of 4 concurrent lifting computation of Figure 3.1(c). The time-multiplexed data-blocks of Figure 3.2 (a) are processed column-wise and produces data-blocks of four sub-band matrices [**A**, **B**, **C**, **D**] which are shown in Figure 3.2(b). As shown in Figure 3.2(c), low-low sub-band data-blocks of first-level [**a**] are obtained after every alternate clock cycles. These data-blocks are folded (shown in Figure 3.2(d)) such that lifting computation of 2^{nd} level DWT is scheduled during every clock cycles. One can find from Figure 3.1 (b)-(c) and Figure 3.2 (a) that in the data reordering stage samples of two different data-blocks corresponding to different clock cycles are grouped which involve multiplexing operation. Also, it is observed from Figure 3.2 (b)-(d) that the data-blocks [**a** and **c**] of low-low and high-low sub-bands are generated in time-multiplexed manner (after the column-wise processing) which requires folding operation. Therefore, the multiplexing operation of data reordering stage of row-column computation (referred as intra-level DWT) introduces an extra folding operation between the successive DWT levels (referred as inter-level DWT).

The multiplexing operation of data reordering stage of intra-level DWT can be avoided if a pair of samples of a particular column of both low-pass and high-pass intermediate matrices are produced through each output data-block of row-wise DWT. The input data-blocks need to be formulated differently to meet this requirement. A pair of samples of a particular row/column is processed in every lifting computation and produces one low-pass component and high-pass intermediate component of particular row/column. Therefore, one pairs of samples from minimum two consecutive rows need to be grouped to form an input data-block of size (2×2) to avoid the multiplexing operation in the data ordering stage of one decomposition level of 2-D DWT.

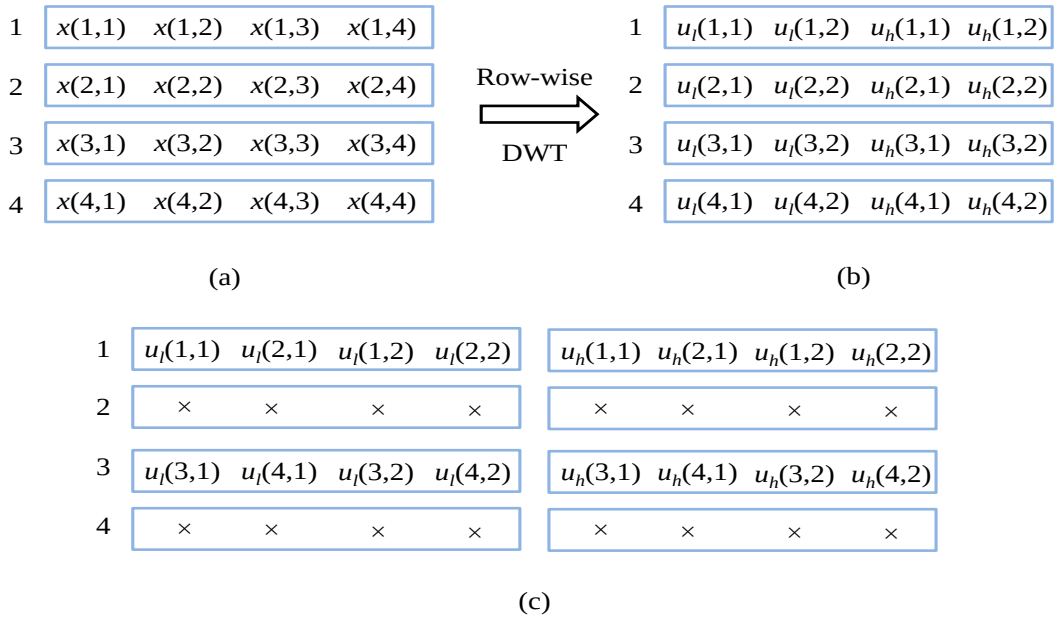

Figure 3.1: (a) Stripe-based (1×4) sized input data-blocks. (b) Intermediate low-pass and high-pass data-blocks of row-wise lifting DWT. (c) Reordering of intermediate data for column-wise lifting DWT.

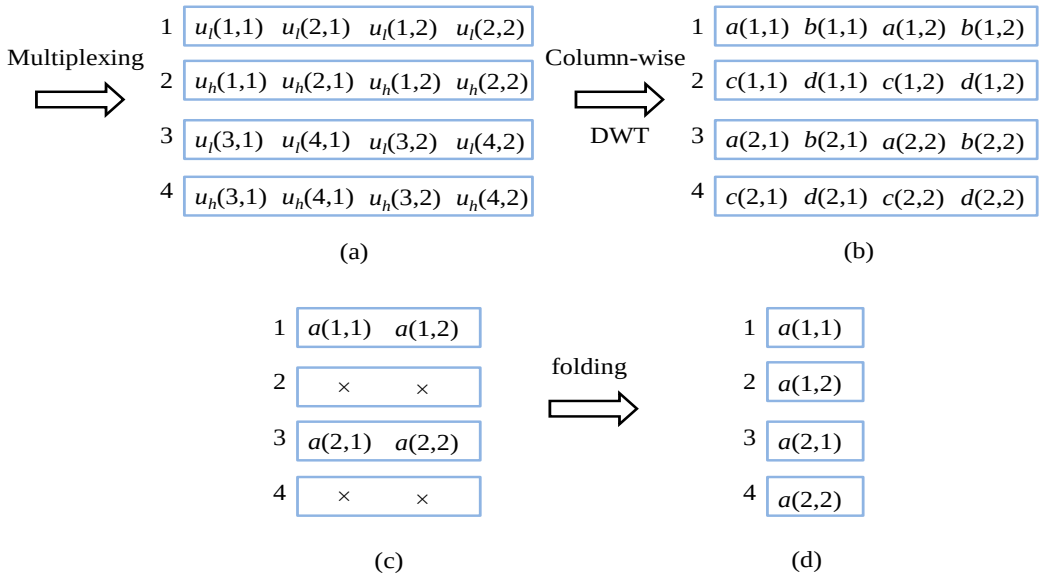

Figure 3.2: (a) Time-multiplexed low-pass and high-pass intermediate data-blocks. (b) Output data-blocks of column-wise lifting DWT. (c) Low-low sub-band data-blocks.
(d) Folded data-blocks.

Figure 3.3 depict the flow of input-output data-blocks of 1-level 2-D DWT for 4 clock cycles using (2×2) data-block. As shown in Figure 3.3 (b)-(c), samples are reordered within a data-block for the column-wise processing which does not involve multiplexing operation. Interestingly, each reordered data-block of Figure 3.3(c) contains one pair of components of a particular column of each low-pass and high-pass intermediate matrix. Therefore, the reordered data-blocks when processed column-wise then each output data-block contain one component of four sub-bands [**A**, **B**, **C**, **D**] each as shown in Figure 3.3 (d). The low-low sub-band components are extracted from each output data-block of Figure 3.3 (d) and shown separately in Figure 3.3(e). Data-blocks of Figure 3.3 (e) are identical to those of Figure 3.2 (d). Note that data-blocks of Figure 3.3 (e) are obtained directly from the output data-blocks of column-wise DWT without any folding. Therefore, using (2×2) sized input data-blocks the intra-level multiplexing operation is avoided in 1-level lifting 2-D DWT.

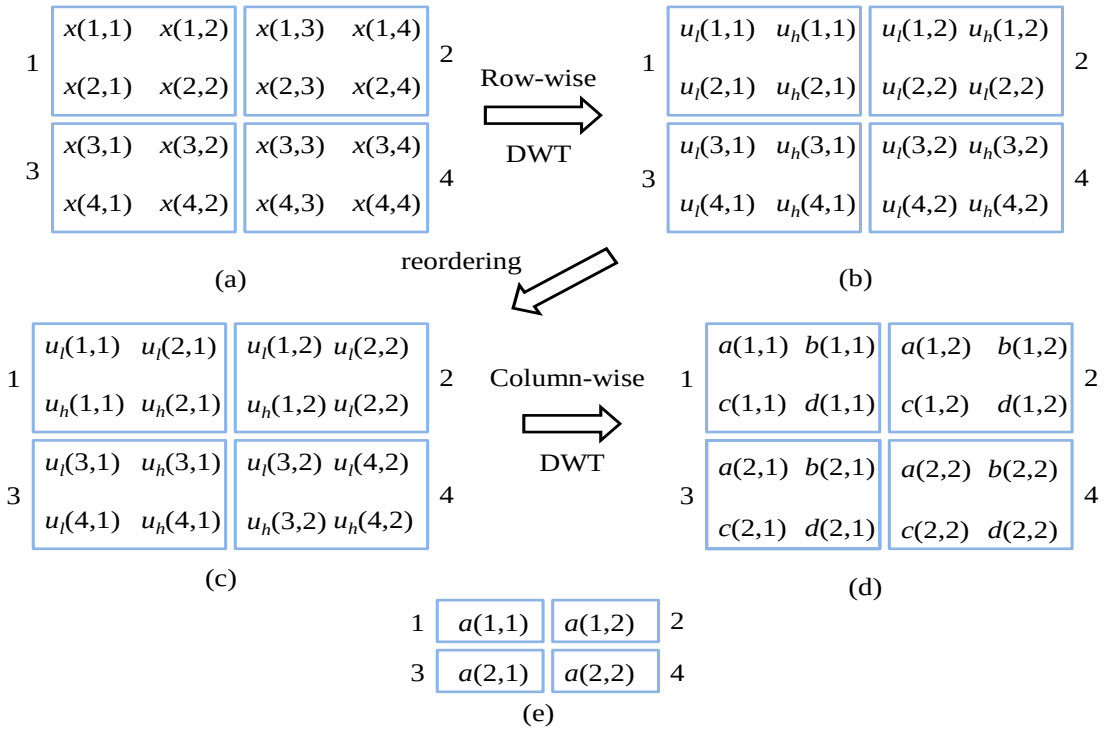

Figure 3.3: (a) Input (2×2) sized data-blocks. (b) Intermediate low-pass and high-pass data-blocks of row-wise lifting DWT. (c) Reordering of data for column-wise lifting DWT. (d) Output data-blocks of column-wise lifting DWT. (e) Low-low sub-band data-blocks.

In general, when (P×P) sized blocks are processed in every clock cycle then after 1-level lifting 2-D DWT one block each of four sub-bands [$\mathbf{A}^1$, $\mathbf{B}^1$, $\mathbf{C}^1$, $\mathbf{D}^1$] of size (P/2×P/2) are obtained, where P is assumed to be power-of-2 integer. The data-blocks of [$\mathbf{A}^1$] are processed further to compute the sub-band components of 2nd level decomposition. As discussed earlier, the minimum block size (2×2) is required to avoid the multiplexing operation in 1-level 2-D DWT and the minimum block size (4×4) is required to avoid multiplexing operation completely in 2-level parallel computation of 2-D DWT. The input-block size of 3-level 2-D DWT is shown in Figure 3.4. In general, the minimum input-block size (2^J×2^J) is required to avoid intra-level multiplexing operation and inter-level block folding completely in J-level parallel computation of 2-D DWT, where J is the DWT levels.

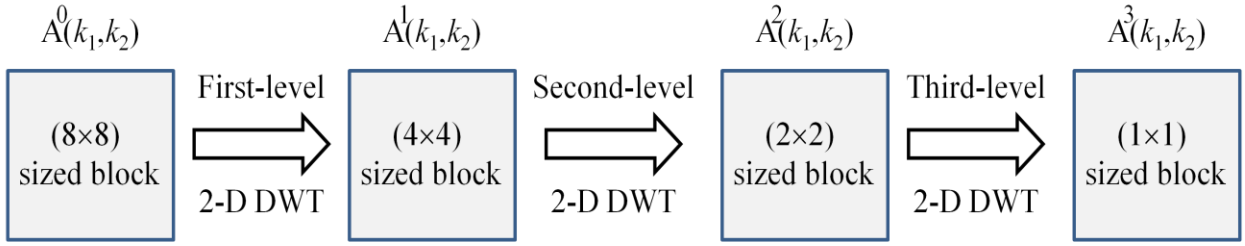

Figure 3.4: Size of input data-blocks of 3-level parallel lifting 2-D DWT

3.2.2 PROPOSED DATA-ACCESS SCHEME

As shown in Figure 3.5, data-blocks of size (P×P) are collected from the input matrix [$\mathbf{X}$] of size ($N \times N$), where P=2^J for J-level 2-D DWT. The first set of (N/P) data-blocks are obtained from the first P rows of input matrix [$\mathbf{X}$] and the second set of (N/P) data-blocks are obtained from the second set of P rows of [$\mathbf{X}$]. Similarly, the (N/P)-th set of (N/P) data-blocks are obtained from the (N/P)-th set of rows of [$\mathbf{X}$]. Therefore, (N/P)^2 data-blocks of size (P×P) are obtained from [$\mathbf{X}$]. These (N/P)^2 data-blocks are arranged in a 2-D array such that each set of (N/P) data-blocks forms a row. Input data-blocks are represented as $\mathbf{A}^0(k_1, k_2)$, where k_1 and k_2, are respectively, the row and column indices, for $0 \leq k_1 \leq \left(\frac{N}{P}\right) - 1$, and $0 \leq k_2 \leq (\frac{N}{P})$-1. The input data-blocks are processed one by one in row serial order as shown in the Figure 3.5. Therefore, the input

data-block $\mathbf{A}^0(k_1, k_2)$ is processed during [(k_1N+k_2)+1]-th clock cycle of a set of $(N/P)^2$ clock cycles.

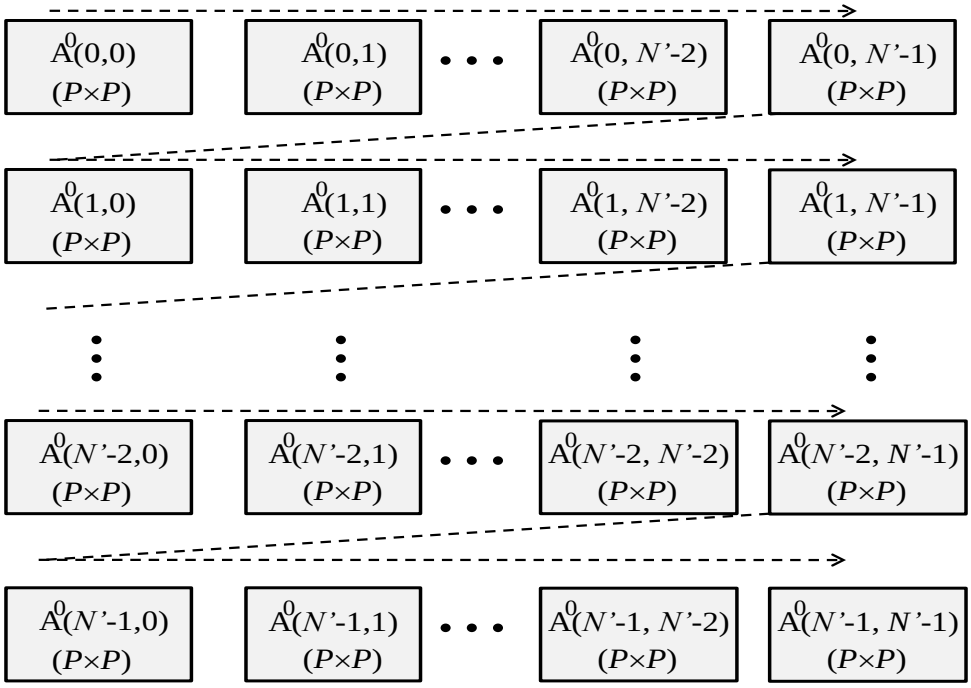

Figure 3.5: Proposed data-access scheme, $N'=(N/P)$, $P=2^J$, J is the DWT levels, N is the input image dimension.

In Multi-level 2-D DWT, ($P_j \times P_j$) size data-blocks $\{\mathbf{A}^{j-1}(k_1, k_2)\}$ of low-low sub-band matrix $[\mathbf{A}^{j-1}]$ of (j-1)-th level are processed serially block-by-block and computes four data-blocks $\{\mathbf{A}^j(k_1, k_2)$, $\mathbf{B}^j(k_1, k_2)$,$\mathbf{C}^j(k_1, k_2)$ and $\mathbf{D}^j(k_1, k_2)\}$ of size $[(P_j/2)\times(P_j/2)]$ each of j-th level low-low, low-high, high-low and high-high sub-band matrices [$\mathbf{A}^j$, $\mathbf{B}^j$, $\mathbf{C}^j$ and $\mathbf{D}^j$] respectively, where $\mathbf{A}^0$= input-image [**X**] and $1 \le j \le J$. The j-th level lifting 2-D DWT computation of input data-block $\mathbf{A}^{j-1}(k_1, k_2)$ is performed in four different stages as:

- **Stage-1:** Lifting DWT computation is performed on ($P_j \times P_j$) size input data-block $\{\mathbf{A}^{j-1}(k_1, k_2)\}$ row-wise to produce a pair of data-blocks $\mathbf{U}^j(k_1, k_2)$ and $\mathbf{V}^j(k_1, k_2)$ of size ($P_j \times P_{j-1}$) corresponding to a pair of intermediate matrices [$\mathbf{U}^j$] and [$\mathbf{V}^j$], respectively.

- **Stage-2:** Intermediate data-blocks $\{\mathbf{U}^j(k_1,k_2)$ and $\mathbf{V}^j(k_1,k_2)\}$ are transposed for reordering the samples of data-block to facilitate the column wise processing. The transposed data-blocks are represented by $\mathbf{W}^j(k_2,k_1)$ and $\mathbf{Z}^j(k_2,k_1)$ of size ($P_{j-1}\times P_j$).
- **Stage-3:** Lifting DWT computation is performed on row-wise of $\mathbf{W}^j(k_2,k_1)$ and $\mathbf{Z}^j(k_2,k_1)$ to produce four sub-band matrices $\{\mathbf{A}^j(k_{2,}k_1),\mathbf{B}^j(k_{2,}k_1),\mathbf{C}^j(k_{2,}k_1)$ and $\mathbf{D}^j(k_{2,}k_1)\}$.
- **Stage-4** Finally, the data-blocks $\{\mathbf{A}^j(k_{2,}k_1),\mathbf{B}^j(k_{2,}k_1),\mathbf{C}^j(k_{2,}k_1)$ and $\mathbf{D}^j(k_{2,}k_1)\}$ are transposed one more time to obtain the data-blocks $\mathbf{A}^j(k_{1,}k_2),\mathbf{B}^j(k_1,k_2),\mathbf{C}^j(k_{1,}k_2)$ and $\mathbf{D}^j(k_{1,}k_2)$. The data-block $\mathbf{A}^j(k_{1,}k_2)$ is further processed to calculate the sub-band components of (j+1)-th level DWT.

3.3 BLOCK FORMULATION OF LIFTING 2-D DWT

The lifting formulation of [Sweldens (1996)] is expressed in a modified form to compute a block of ($P_j\times P_j$) samples. Suppose, during j-th level lifting 2-D DWT, the low-low sub-band matrix [$\mathbf{A}^{j-1}$] of (j-1)-th level are processed in blocks $\{\mathbf{A}^{j-1}(k_1,k_2)\}$ of size ($P_j\times P_j$). From each data-block $\{\mathbf{A}^{j-1}(k_1,k_2)\}$, P_j data-vectors are derived from P_j rows of $\mathbf{A}^{j-1}(k_1,k_2)$ as:

$$\mathbf{A}^{j-1}(k_1,k_2)=[\mathbf{a}^j_{2m_1}(k_2),\mathbf{a}^j_{2m_1+1}(k_2),\ \ldots,\mathbf{a}^j_{2m_1+P_j-1}(k_2)]^{\mathrm{T}} \qquad 3.1(a)$$

$\mathbf{a}^{j-1}_{2m_1+i}(k_2)$, the i-th row of $\mathbf{A}^{j-1}(k_1,k_2)$ is defined as

$$\mathbf{a}^{j-1}_{2m_1+i}(k_2)=[a^{j-1}(2m_1+i,2n_1),a^{j-1}(2m_1+i,2n_1+1),\ldots a^{j-1}(2m_1+i,2n_1+P_j-1)] \qquad 3.1(b)$$

where, $m_{1=}k_1\left(\frac{P_j}{2}\right), n_{1=}k_2\left(\frac{P_j}{2}\right), P_j=2^{J-j}$ 3.1(c)

For $0\le i\le P_j-1$, $0\le k_1\le\left(\frac{N}{P}\right)-1$, and $0\le k_2\le(\frac{N}{P})$-1

In stage-1, the lifting computation is performed on each row of $\mathbf{A}^{j-1}(k_1,k_2)$ and a pair of low pass and high pass intermediate data-vectors $\{\mathbf{u}^j_{2m_1+i}(k_2)$ and $\mathbf{v}^j_{2m_1+i}(k_2)\}$ are obtained. The lifting computation of $\mathbf{a}^{j-1}_{2m_1+i}(k_2)$ is expressed as:

$$[\mathbf{u}^{j}_{2m_1+i}(k_2), \mathbf{v}^{j}_{2m_1+i}(k_2) = \mathbf{a}^{j-1}_{2m_1+i}(k_2) \otimes \Omega \quad 3.2$$

where, $\otimes$ denotes the lifting operation between the coefficient matrix [Ω] and the data-vector $\mathbf{a}^{j-1}_{2m_1+i}(k_2)$, such that lifting computation is performed between a pair of components of $\mathbf{a}^{j-1}_{2m_1+i}(k_2)$ with one row of coefficients of [Ω]. The low-pass and high-pass intermediate data vectors $\mathbf{u}^{j}_{2m_1+i}(k_2)$ and $\mathbf{v}^{j}_{2m_1+i}(k_2)$ are defined as:

$$\mathbf{u}^{j}_{2m_1+i}(k_2) = \left[u^j(2m_1+i, n_1), u^j(2m_1+i, n_1+1), \ldots, u^j\left(2m_1+i, n_1+\tfrac{P_j}{2}-1\right)\right] \quad 3.3(a)$$

$$\mathbf{v}^{j}_{2m_1+i}(k_2) = [v^j(2m_1+i, n_1), v^j(2m_1+i, n_1+1), \ldots, v^j\left(2m_1+i, n_1+P_j/2-1\right)] \quad 3.3(b)$$

and, the lifting coefficient matrix [**Ω**] of size [(P_j/2)×4] is defined as:

$$[\mathbf{\Omega}] = \begin{bmatrix} \alpha & \beta & \gamma & \delta \\ \alpha & \beta & \gamma & \delta \\ & \vdots & & \\ \alpha & \beta & \gamma & \delta \\ \alpha & \beta & \gamma & \delta \end{bmatrix} \quad 3.4$$

The computation of (3.2) is expressed in split form as:

$$\begin{bmatrix} u^j(2m_1+i, n_1) \\ v^j(2m_1+i, n_1) \end{bmatrix} = [\alpha \quad \beta \quad \gamma \quad \delta] \otimes \begin{bmatrix} a^{j-1}(2m_1+i, 2n_1) \\ a^{j-1}(2m_1+i, 2n_1+1) \end{bmatrix} \quad 3.5$$

$$\begin{bmatrix} u^j(2m_1+i, n_1+1) \\ v^j(2m_1+i, n_1+1) \end{bmatrix} = [\alpha \quad \beta \quad \gamma \quad \delta] \otimes \begin{bmatrix} a^{j-1}(2m_1+i, 2n_1+2) \\ a^{j-1}(2m_1+i, 2n_1+3) \end{bmatrix} \quad 3.6$$

$$\vdots \qquad\qquad \vdots$$

$$\begin{bmatrix} u^j(2m_1+i, n_1+\frac{P_j}{2}-1) \\ v^j(2m_1+i, n_1+\frac{P_j}{2}-1) \end{bmatrix} = [\alpha \quad \beta \quad \gamma \quad \delta] \otimes \begin{bmatrix} a^{j-1}(2m_1+i, 2n_1+P_j-2) \\ a^{j-1}(2m_1+i, 2n_1+P_j-1) \end{bmatrix} \quad 3.7$$

The lifitng computation of (3.5) is expressed in expanded form as

$$r_1^j(2m_1+i,n_1)=a^{j-1}(2m_1+i,2n_1+1)+\alpha(a^{j-1}(2m_1+i,2n_1)+a^{j-1}(2m_1+i,2n_1-2))$$

$$r_2^j(2m_1+i,n_1)=a^{j-1}(2m_1+i,2n_1-2)+\beta(r_1^j(2m_1+i,n_1)+r_1^j(2m_1+i,n_1-1))$$

$$v^j(2m_1+i,n_1)=r_1^j(2m_1+i,n_1-1)+\gamma(r_2^j(2m_1+i,n_1)+r_2^j(2m_1+i,n_1-1))$$

$$u^j(2m_1+i,n_1)=r_2^j(2m_1+i,n_1-1)+\delta(v^j(2m_1+i,n_1)+v^j(2m_1+i,n_1-1)) \quad 3.8$$

The row-wise lifting computation of $\mathbf{A}^{j-1}(k_1,k_2)$ results a pair of intermediate matrices $\mathbf{U}^j(k_1,k_2)$ and $\mathbf{V}^j(k_1,k_2)$ of size $\left(P_j\times(\frac{P_j}{2})\right)$ each. The $\mathbf{U}^j(k_1,k_2)$ and $\mathbf{V}^j(k_1,k_2)$ are defined as:

$$\mathbf{U}^j(k_1,k_2)=[\mathbf{u}^j_{2m_1}(n_1),\mathbf{u}^j_{2m_1+1}(n_1+1),\ \dots,\mathbf{u}^j_{2m_1+P_j-1}(n_1+P_j/2-1)]^{\mathrm{T}} \quad 3.9$$

$$\mathbf{V}^j(k_1,k_2)=[\mathbf{v}^j_{2m_1}(n_1),\mathbf{v}^j_{2m_1+1}(n_1+1),\ \dots,\mathbf{v}^j_{2m_1+P_j-1}(n_1+P_j/2-1)]^{\mathrm{T}} \quad 3.10$$

In stage-2, $\mathbf{U}^j(k_1,k_2)$ and $\mathbf{V}^j(k_1,k_2)$ are transposed:

$$\mathbf{W}^j(k_2,k_1)=[\mathbf{U}^j(k_1,k_2)]^{\mathbf{T}} \quad 3.11$$

$$\mathbf{Z}^j(k_2,k_1)=[\mathbf{V}^j(k_1,k_2)]^{\mathbf{T}} \quad 3.12$$

Where

$$\mathbf{W}^j(k_2,k_1)=[\mathbf{w}^j_{n_1}(k_1),\mathbf{w}^j_{n_1+1}(k_1),\ \dots,\mathbf{w}^j_{n_1+P_j/2-1}(k_1)]^{\mathrm{T}} \quad 3.13(a)$$

$$\mathbf{Z}^j(k_2,k_1)=[\mathbf{z}^j_{n_1}(k_1),\mathbf{z}^j_{n_1+1}(k_1),\ \dots,\mathbf{z}^j_{n_1+P_j/2-1}(k_1)]^{\mathrm{T}} \quad 3.13(b)$$

$$\mathbf{w}^j_{n_1+i}(k_1)=[u^j(2m_1,n_1+i),u^j(2m_1+1,n_1+i),\dots,u^j \quad 3.13(c)$$

$$\mathbf{z}^j_{n_1+i}(k_1)=\left[v^j(2m_1,n_1+i),v^j(2m_1+1,n_1+i),\dots,v^j\left(2m_1+P_j-1,n_1+i\right)\right] \quad 3.13(d)$$

In stage-3, lifting computation is performed on each row of data-blocks $\{\mathbf{W}^j(k_2,k_1)$ and $\mathbf{Z}^j(k_2,k_1)\}$. The lifting computation of *i*-th row of $\{\mathbf{W}^j(k_2,k_1)$ and $\mathbf{Z}^j(k_2,k_1)\}$ is expressed as:

$$[\mathbf{aa}^j_{n_1+i}(k_1),\mathbf{b}^j_{n_1+i}(k_1)=\mathbf{w}^j_{n_1+i}(k_1)\otimes\Omega \quad 3.14(a)$$

$$[\mathbf{c}^j_{n_1+i}(k_1),\mathbf{dd}^j_{n_1+i}(k_1)=\mathbf{z}^j_{n_1+i}(k_1)\otimes\Omega \quad 3.14(b)$$

where, $\mathbf{aa}^j_{n_1+i}(k_1)$ and $\mathbf{b}^j_{n_1+i}(k_1)$ represent the low-pass and high-pass data-vectors of intermediate data-vector $\mathbf{w}^j_{n_1+i}(k_1)$, while $\mathbf{c}^j_{n_1+i}(k_1)$ and $\mathbf{dd}^j_{n_1+i}(k_1)$ represent the low-pass and high-pass data-vectors of intermediate data-vector $\mathbf{z}^j_{n_1+i}(k_1)$.

In stage-4, the data-vectors $\mathbf{aa}^j_{n_1+i}(k_1)$ and $\mathbf{dd}^j_{n_1+i}(k_1)$ are normalized as:

$$\mathbf{a}^j_{n_1+i}(k_1) = k^2.\mathbf{aa}^j_{n_1+i}(k_1) \quad \text{3.15(a)}$$

$$\mathbf{d}^j_{n_1+i}(k_1) = (1/k^2).\mathbf{dd}^j_{n_1+i}(k_1) \quad \text{3.15(b)}$$

According to 3.16(a), (P_j/2) data vectors $\{\mathbf{a}^j_{n_1+i}(k_1)$, for $0 \le i \le (P_j/2)$-1$\}$ are concatenated to obtain the data-block $\mathbf{A}^j_1(k_1,k_2)$ of size [(P_j/2)×(P_j/2)] of low-low sub-band matrix [$\mathbf{A}^j_1$]. Similarly, (P_j/2) data vectors $\{\mathbf{b}^j_{n_1+i}(k_1)$, $\mathbf{c}^j_{n_1+i}(k_1)$ and $\mathbf{d}^j_{n_1+i}(k_1)\}$ are concatenated according to 3.16(b)-3.16(d) to obtain [(P_j/2)×(P_j/2)] sized data-blocks $\{\mathbf{B}^j_1(k_2,k_1), \mathbf{C}^j_1(k_2,k_1)$ and $\mathbf{D}^j_1(k_2,k_1)\}$ of low-high, high-low and high-high sub-band matrices $\mathbf{B}^j_1$, $\mathbf{C}^j_1$, and $\mathbf{D}^j_1$, respectively.

$$\mathbf{A}^j_1(k_2,k_1) = [\mathbf{a}^j_{n_1}(k_1), \mathbf{a}^j_{n_1+1}(k_1), \ \dots, \mathbf{a}^j_{n_1+(P_j/2)-1}(k_1)]^{\mathrm{T}} \quad \text{3.16(a)}$$

$$\mathbf{B}^j_1(k_2,k_1) = [\mathbf{b}^j_{n_1}(k_1), \mathbf{b}^j_{n_1+1}(k_1), \ \dots, \mathbf{b}^j_{n_1+(P_j/2)-1}(k_1)]^{\mathrm{T}} \quad \text{3.16(b)}$$

$$\mathbf{C}^j_1(k_2,k_1) = [\mathbf{c}^j_{n_1}(k_1), \mathbf{c}^j_{n_1+1}(k_1), \ \dots, \mathbf{c}^j_{n_1+(P_j/2)-1}(k_1)]^{\mathrm{T}} \quad \text{3.16(c)}$$

$$\mathbf{D}^j_1(k_2,k_1) = [\mathbf{d}^j_{n_1}(k_1), \mathbf{d}^j_{n_1+1}(k_1), \ \dots, \mathbf{d}^j_{n_1+(P_j/2)-1}(k_1)]^{\mathrm{T}} \quad \text{3.16(d)}$$

Finally in stage-5, data-blocks $\{\mathbf{A}^j(k_2,k_1), \mathbf{B}^j(k_2,k_1), \mathbf{C}^j(k_2,k_1), \mathbf{D}^j(k_2,k_1)\}$ are transposed to obtain the data-blocks $\{\mathbf{A}^j(k_2,k_1), \mathbf{B}^j(k_2,k_1), \mathbf{C}^j(k_2,k_1), \mathbf{D}^j(k_2,k_1)\}$ as:

$$\mathbf{A}^j(k_1,k_2) = [\mathbf{A}^j_1(k_2,k_1)]^{\mathrm{T}} \quad \text{3.17(a)}$$

$$\mathbf{B}^j(k_1,k_2) = [\mathbf{B}^j_1(k_2,k_1)]^{\mathrm{T}} \quad \text{3.17(b)}$$

$$\mathbf{C}^j(k_1,k_2) = [\mathbf{C}^j_1(k_2,k_1)]^{\mathrm{T}} \quad \text{3.17(c)}$$

$$\mathbf{D}^j(k_1,k_2) = [\mathbf{D}^j_1(k_2,k_1)]^{\mathrm{T}} \quad \text{3.17(d)}$$

The data-blocks $\mathbf{A}^{j}(k_1,k_2)$ of j-th level of low-low sub-band matrix $\mathbf{A}_1^{j}$ are processed similar to the data-blocks $\mathbf{A}^{j-1}(k_1,k_2)$ using (3.1)-(3.17) to obtain the sub-bands components of (j+1)-th level DWT. A parallel architecture is presented in the following section for computation of multi-level lifting 2-D DWT.

3.4. PROPOSED ARCHITECTURE

The proposed architecture for computation of J-level 2-D DWT is shown in Figure 3.6. It consists of J PUs integrated in a pipeline structure. The first PU receives the input image [$\mathbf{A}^0$=$\mathbf{X}$] of size (N×N) and perform computations of first-level lifting 2-D DWT. According to the data-accessing scheme given in Figure 3.5, the input matrix [$\mathbf{A}^0$] is fed to the first PU in the form of data-blocks $\mathbf{A}^0(k_1,k_2)$ of size (2^J×2^J), for $0 \le k_1 \le \left(\frac{N}{2^J}\right) - 1$, and $0 \le k_2 \le (\frac{N}{2^J})$-1. ($M^2$) data-blocks of are derived from input-matrix [$\mathbf{A}^0$], where M=(N/P), and P=2^J. The first PU receives one data-block $A^0(k_1, k_2)$ in every clock cycle and the entire input matrix [$\mathbf{A}^0$] in (M^2) clock cycles.

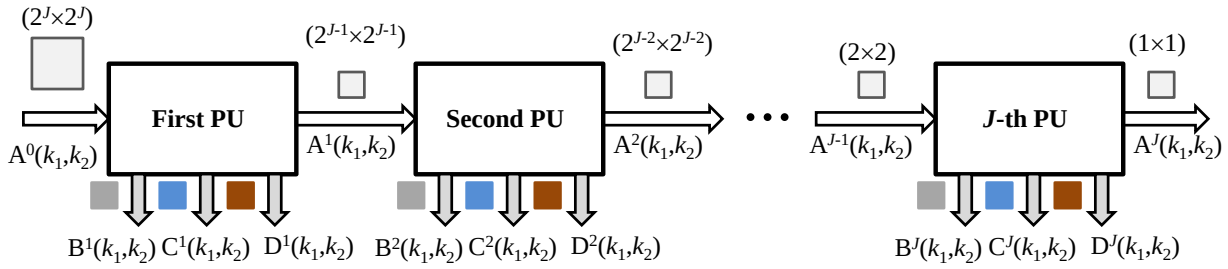

Figure 3.6: Proposed structure for J-level lifting 2-D DWT.

The first PU receives the input data-block $A^0(k_1, k_2)$ during the [(k_1M+k_2)] clock cycle and calculates (2^{J-1}×2^{J-1}) size four data-blocks $[\mathbf{A}^1(k_1,k_2),\mathbf{B}^1(k_1,k_2),\mathbf{C}^1(k_1,k_2),\mathbf{D}^1(k_1,k_2)]$ of sub-bands $[\mathbf{A}^1,\mathbf{B}^1,\mathbf{C}^1,\mathbf{D}^1]$ of first-level DWT, where M=N/P. Data-blocks $A^1(k_1, k_2)$ (of low-low sub-band of first-level) are sent to the second PU to calculate the data-blocks $[\mathbf{A}^2(k_1,k_2),\mathbf{B}^2(k_1,k_2),\mathbf{C}^2(k_1,k_2),\mathbf{D}^2(k_1,k_2)]$ of sub-bands [$\mathbf{A}^2$, $\mathbf{B}^2$, $\mathbf{C}^2$, $\mathbf{D}^2$] of second-level DWT. Like this the j-th PU receives data-blocks $\mathbf{A}^{j-1}(k_1,k_2)$ of low-low sub-band matrix [$\mathbf{A}^{j-1}$] from the

(j-1)-th PU and computes the data-blocks $\{\mathbf{A}^j(k_1,k_2), \mathbf{B}^j(k_1,k_2), \mathbf{C}^j(k_1,k_2), \mathbf{D}^j(k_1,k_2)\}$ of four sub-bands [$\mathbf{A}^j$, $\mathbf{B}^j$, $\mathbf{C}^j$, $\mathbf{D}^j$] of j-th level DWT. In general, the j-th PU receives ($P_j \times P_j$) size data-blocks $\mathbf{A}^{j-1}(k_1,k_2)$ in every clock cycle and the entire sub-band matrix [$\mathbf{A}^{j-1}$] of size [($N/2^{j-1}$)×($N/2^{j-1}$)] in (M^2) clock cycles. All the PUs of proposed architecture performs identical type computation and they have identical structures except that they process different size data-block. Interestingly, the size of output data-block of each PU is one-fourth of the size of input data-block. Therefore, a generic design for the j-th PU is presented in the following section which can be scaled according to the size of input data-block to obtain different PU structures of proposed parallel architecture.

3.4.1 A GENERIC DESIGN OF *J*-th PU

The j-th PU of the proposed architecture receives ($P_j \times P_j$) size data-blocks $\mathbf{A}^{j-1}(k_1, k_2)$ from the (j-1)-th PU and computes the data-blocks $\{\mathbf{A}^j(k_1,k_2), \mathbf{B}^j(k_1,k_2), \mathbf{C}^j(k_1,k_2), \mathbf{D}^j(k_1,k_2)\}$ of size (P_j/2)×(P_j/2) corresponding to four sub-bands [$\mathbf{A}^j$, $\mathbf{B}^j$, $\mathbf{C}^j$, $\mathbf{D}^j$] of j-th level DWT. The structure of the j-th PU is shown in Figure 3.7. It consists of a row-processor, column-processor, and one scaling unit (SU). Both row-processor and column-processor is comprised of (P_j) processing elements (PEs). The input data-block $\mathbf{A}^{j-1}(k_1, k_2)$ is fed to the row-processor in terms of (P_j) data-vectors where each row of $\mathbf{A}^{j-1}(k_1, k_2)$ represents a data-vector. The (i+1)th PE receives the (i+1)-th data-vector $\mathbf{a}^{j-1}_{2m_1+i}(k_2)$ of size (P_j) as defined in 3.1(b), and produces a pair of low-pass and high-pass data vectors $\{\mathbf{u}^j_{2m_1+i}(k_2), \mathbf{v}^j_{2m_1+i}(k_2)\}$ of size (P_j/2) each. The (i+1)-th PE performs the computation of (3.2) and its structure is derived for block-size (P_j) which is shown in Figure 3.8. It consists of ($2P_j$) lifting cells (LC) and they are arranged in a 2-D array of (P_j/2) rows and 4 columns. According to (3.5)-(3.7), each row of LC performs lifting DWT computation on a pair of input samples and produces one low-pass and one high-pass DWT component. The (n_1+1)-th row of LCs of (i+1)-th PE receives the pair of input samples $\{a^{j-1}(2m_1+i,2n_1), a^{j-1}(2m_1+i, 2n_1+1)\}$ of input data-vector $\mathbf{a}^{j-1}_{2m_1+i}(k_2)$ and produces one low-pass and one high-pass component $\{u^j(2m_1+i, n_1), v^j(2m_1+i,n_1)\}$, where $m_1=k_1(P_j/2)$, $n_1=k_2(P_j/2)$, for $0 \le i \le P_j$-1. The function of a LC is depicted in shown in Figure 3.9.

During every clock cycle, (P_j) PEs of row-processor receives (P_j) data-vectors (rows) of $\mathbf{A}^{j-1}(k_1, k_2)$, and produces ($P_j$) data-vectors (rows) of low-pass and high-pass intermediate data-blocks $\{\mathbf{U}^j(k_1, k_2)$ and $\mathbf{V}^j(k_1, k_2)\}$. The samples of intermediate data-blocks $\{\mathbf{U}^j(k_1, k_2)$ and $\mathbf{V}^j(k_1, k_2)\}$ are reordered for column-wise processing. According to (3.11)-(3.12), the data-blocks $\{\mathbf{U}^j(k_1, k_2)$ and $\mathbf{V}^j(k_1, k_2)\}$ are transposed to obtain the reordered data-blocks $\{\mathbf{W}^j(k_2, k_1)$ and $\mathbf{Z}^j(k_2, k_1)\}$. Since all the (P_j) rows of $\{\mathbf{U}^j(k_1, k_2)$ and $\mathbf{V}^j(k_1, k_2)\}$are obtained in one clock cycle, the transpose operation of $\{\mathbf{U}^j(k_1, k_2)$ and $\mathbf{V}^j(k_1, k_2)\}$is implemented by hard-wiring.

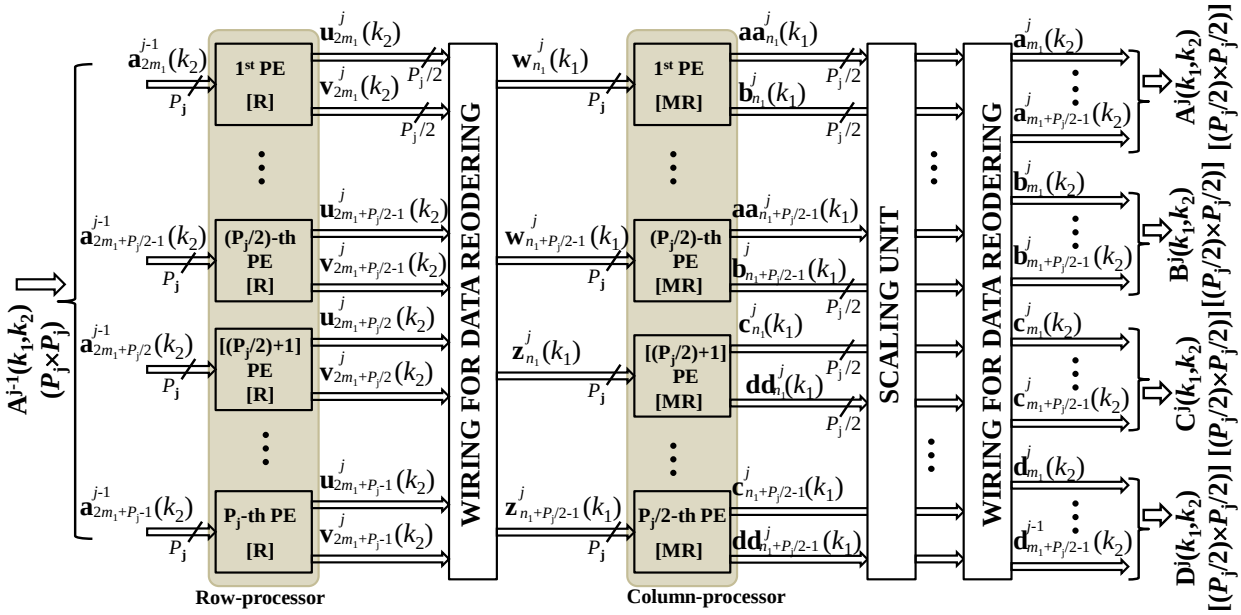

Figure 3.7: Structure of j-th processing unit, where $m_1=k_1(P_j/2)$, $n_1=k_2(P_j/2)$, $0 \leq k_1 \leq M$-1 and $0 \leq k_2 \leq M$ -1, $M=N/P$, $P=2^J$, $P_j=2^{J-j}$.

According to (3.14), a pair of reordered intermediate data-blocks $\{\mathbf{W}^j(k_2, k_1)$ and $\mathbf{Z}^j(k_2, k_1)\}$ of size $[(P_j/2)\times P_j]$ each are processed concurrently by the column processor. During every clock cycle, (P_j) PEs of column-processor receives (P_j/2) data-vectors (rows) of intermediate data-block $\mathbf{W}^j(k_2, k_1)$ and $\mathbf{Z}^j(k_2, k_1)$ in parallel such that first (P_j/2) PEs receives (P_j/2) data-vectors (rows) of data-block $\mathbf{W}^j(k_2, k_1)$ while the remaining (P_j/2) PEs of column-processor receives (P_j/2) rows of data-block $\mathbf{Z}^j(k_2, k_1)$. The structure of the PE of the column-processor is identical to the PE structure of the row-processor shown in Figure 3.8 except that each data register (R) is

replaced by a shift register of size M words since samples of successive M-th data-block of a set of (M^2) data-blocks of sub-band matrix are belong to the same column.

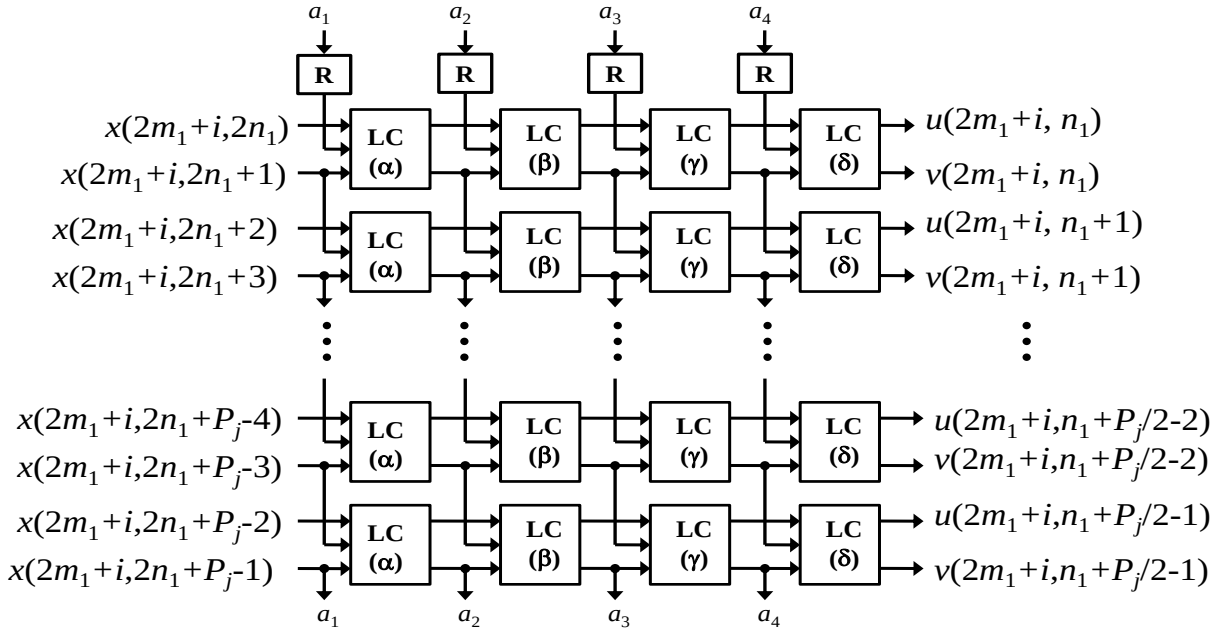

Figure 3.8: Structure of (i+1)-th processing element (PE).

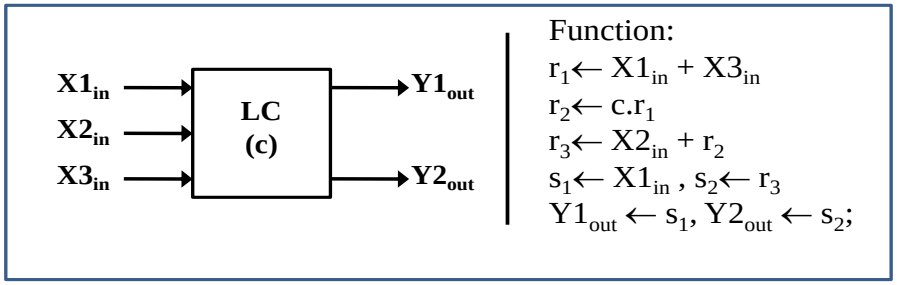

Figure 3.9: Function of the lifting cell (LC)

As shown in Figure 3.7, each PE of the first set of (P_j/2) PEs of column-processor receives one data-vector $\mathbf{w}^j_{n_1+i}(k_1)$ of intermediate data-block $\mathbf{W}^j(k_2, k_1)$ and calculate one low-low data-vector $\mathbf{aa}^j_{n_1+i}(k_1)$ and one low-high data-vector $\mathbf{b}^j_{n_1+i}(k_1)$ of size (P_j/2) each. Similarly, each PE of the second set of (P_j/2) PEs of column-processor receives one data-vector $\mathbf{z}^j_{n_1+i}(k_1)$ of

intermediate data-block $\mathbf{Z}^j(k_2, k_1)$ and calculate one high-low data-vector $\mathbf{c}^j_{n_1+i}(k_1)$ and one high-high data-vector $\mathbf{dd}^j_{n_1+i}(k_1)$ of size $(P_j/2)$ each. Therefore, the first set of $(P_j/2)$ PEs of column-processor receive $(P_j/2)$ data-vectors of $\mathbf{W}^j(k_2, k_1)$ during every clock cycle and produces one set of $(P_j/2)$ data-vectors $\mathbf{aa}^j_{n_1+i}(k_1)$ and one set of $(P_j/2)$ data-vectors $\mathbf{b}^j_{n_1+i}(k_1)$. During the same period, the second set of $(P_j/2)$ PEs of column-processor receive $(P_j/2)$ data-vectors of $\mathbf{Z}^j(k_2, k_1)$ and produces one set of $(P_j/2)$ data-vectors $\mathbf{c}^j_{n_1+i}(k_1)$ and one set of $(P_j/2)$ data-vectors $\mathbf{dd}^j_{n_1+i}(k_1)$, for $0 \le i \le (P_j/2)\text{-}1$.

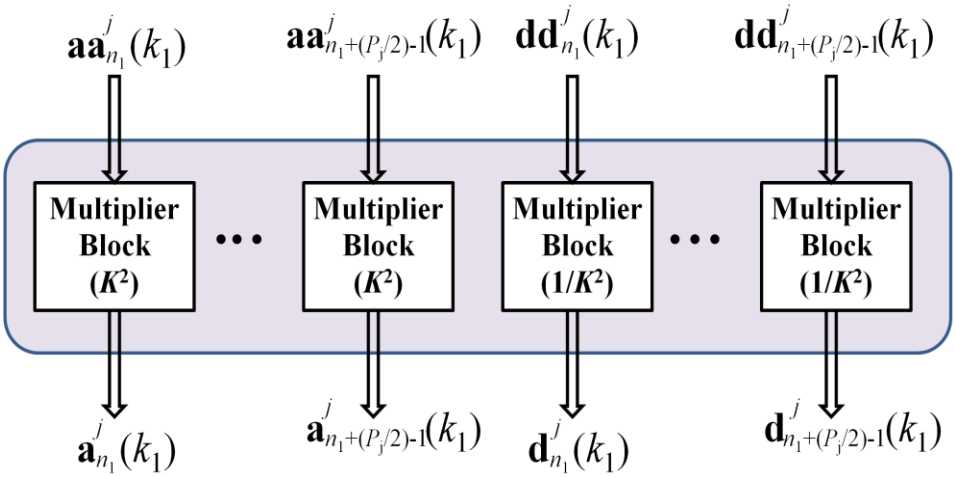

Figure 3.10: Internal structure of scaling unit

According to (3.15), the low-low data-vectors $\mathbf{aa}^j_{n_1+i}(k_1)$ and the high-high data-vectors $\mathbf{dd}^j_{n_1+i}(k_1)$ need to be scaled. The scaling operation is performed on the data-vector by a separate scaling unit (SU). The SU comprises of $(P_j^2/2)$ multipliers. The internal structure of SU is shown in Figure 3.10. The SU receives one set of $(P_j/2)$ data-vectors $\mathbf{aa}^j_{n_1+i}(k_1)$ and one set of $\mathbf{dd}^j_{n_1+i}(k_1)$ from the column-processor during every clock cycle and produces one set of $(P_j/2)$ scaled data-vectors $\mathbf{a}^j_{n_1+i}(k_1)$ and $\mathbf{d}^j_{n_1+i}(k_1)$ each. The $(P_j/2)$ scaled data-vectors $\mathbf{a}^j_{n_1+i}(k_1)$ and $\mathbf{d}^j_{n_1+i}(k_1)$ are concatenated row-wise to form a pair of data-block $\mathbf{A}_1{}^j(k_2, k_1)$ and $\mathbf{D}_1{}^j(k_2, k_1)$. Similarly, $(P_j/2)$ data-vectors $\mathbf{b}^j_{n_1+i}(k_1)$ and $\mathbf{c}^j_{n_1+i}(k_1)$ obtained from the column processor are

concatenated row-wise to form another pair data-block $\mathbf{B}_1^j(k_2, k_1)$ and $\mathbf{C}_1^j(k_2, k_1)$. Finally, data-blocks $[\mathbf{A}_1^j(k_2, k_1), \mathbf{B}_1^j(k_2, k_1), \mathbf{C}_1^j(k_2, k_1), \mathbf{D}_1^j(k_2, k_1)]$ are reordered to produce the output-blocks in the same order as the input-block $\mathbf{A}^{j-1}(k_1, k_2)$. According to (3.17), data-blocks are transposed for data-reordering. The transpose operation is implemented through hard-wiring. Therefore, the j-th PU receive one data-block $\mathbf{A}^{j-1}(k_1, k_2)$ of size $(P_j \times P_j)$ from $(j$-1)-th PU during every clock cycle and produces four data-blocks $[\mathbf{A}^j(k_1, k_2), \mathbf{B}^j(k_1, k_2), \mathbf{C}^j(k_1, k_2), \mathbf{D}^j(k_1, k_2)]$ of four sub-band matrices $[\mathbf{A}^j, \mathbf{B}^j, \mathbf{C}^j, \mathbf{D}^j]$ of j-th level DWT in parallel. The data-block $\mathbf{A}^j(k_1, k_2)$ is sent to the $(j$+1)-th PU to compute the data-blocks of $(j$+1)-th level sub-band matrices.

As shown in Figure 3.7 and Figure 3.8, both the row-processor and column-processor uses (P_j) identical PEs and each PE consists of $(P_j/2)$ rows of LCs. Note that the PE design of column-processor is obtained directly from the PE design of row-processor by simply replacing each data register (R) by shift-register (SR) of size M words. The PE design of Figure 3.8 is scaled according to the input vector size, to obtain the PE design for the row-processor or column-processor of the PU of particular DWT level. The number of rows of input data-block decides the number of PEs of row/column processor of a particular PU. For a $(P_j \times P_j)$ size data-block, the size of data-vector is equal to the number of rows. A single parameter (P_j), controls the PE design as well as the number of PEs of row-processor and column processor of the PU. Therefore, the PU design for any DWT level can be obtained directly from the generic design of Figure 3.7. For example: suppose a 4-level parallel structure is required for lifting 2-D DWT. The input-block size of first PU is (16×16). The size of input-vector and the number of rows is P_j=16. The generic PU design of Figure 3.7 and the PE design of Figure 3.8 are configured for P_j=16 to obtain the design of first PU. The input-block size of second PU, third PU and fourth PU is $[(P_j/2) \times P_j]$ = (8×8), (4×4), and (2×2). Therefore, the generic PU design of Figure 3.7 and the PE design of Figure 3.8 are configured for block size P_j=8, 4 and 2 to obtain the design of second PU, third PU and fourth PU, respectively. The proposed generic PU simplifies the PU design complexity of parallel architecture at great extent and introduces higher degree of circuit regularity and modularity which is a major concern for the existing parallel architectures of 2-D DWT.

3.5 HARDWARE AND TIME COMPLEXITIES

The proposed architecture consists of J PUs and performs J-level lifting 2-D DWT computation concurrently. Each PU consists of one row-processor, one column processor and one scaling unit. Both the row-processor and column processor of j-th PU is comprised of (P_j) identical PEs, where each PE is further comprised of ($2P_j$) LCs. In addition to this each PE of row-processor involves 4 registers and $4M$ registers in case of column-processor. Each LC is comprised of one multiplier, two adders and two pipeline latches. The scaling unit involves ($P_j^2/2$) multipliers and ($P_j^2/2$) pipeline-latches. The row-processor of j-th PU involves ($2P_j^2$) multipliers, ($4P_j^2$) adders and $4P_j$ ($1+P_j$) registers/pipeline-latches. Similarly, the column processor involves ($5P_j^2/2$) multipliers, ($4P_j^2$) adders and [$4P_j$ ($M+P_j$)] registers/pipeline-latches. Therefore, the j-th PU requires ($9P_j^2/2$) multipliers, ($8P_j^2$) adders and [$4P_j$ ($1+M+2P_j$)] registers/pipeline-latches and computes j-th level 2-D DWT of ($P_j \times P_j$) size data block in one clock cycle. Note that out of [$4P_j$ ($1+M+2P_j$)] registers/pipeline-latches, $4P_j$ ($1+M$) represent the data-registers and the remaining ($8P_j^2$) are the pipeline-latches. The complexity of each PU is estimated using the generalized formula given above. The first PU requires ($9P^2/2$) multipliers, ($8P^2$) adders and [$4P$ ($1+M+2P$)] registers/pipeline-latches, where the second PU requires ($9P^2/8$) multipliers, (P^2) adders and [$2P$ ($1+M+P$)] registers/pipeline-latches. The J-th PU requires 18 multipliers, 36 adders and [$8(M+5)$] registers/pipeline-latches. Therefore, the proposed parallel structure for J-level 2-D DWT involves: $6P^2$ ($1-2^{-2J}$) multipliers, [$32P^2$ ($1-2^{-2J}$)/3] adders, [$8(P+N)$ ($1-2^{-J}$)] data-registers and [$32P^2$ ($1-2^{-2J}$)/3] pipeline-latches. The proposed structure receives one data-block of size ($P \times P$) in every clock cycle and the entire input matrix of size ($N \times N$) in (M^2) clock cycles, where $M=N/P$, one clock cycle period is defined as $T=T_M+2T_A$, T_M and T_A are the multiplication and addition delay, respectively.

The hardware and time complexities of the proposed structure and the existing parallel structures of [Mohanty and Meher (2013)], Mohanty and Mahajan (2013)] and [Hu and Jong (2013)] are listed in Table 3.1 in terms of multiplier, adder, registers, and computation time for comparison. Note that existing structures processes data-blocks of size ($P^2 \times 1$) while the proposed processes data-blocks of size ($P \times P$). Compared with the structure of [Mohanty and Meher (2013)] and Mohanty and Mahajan (2013)] the structure of [Hu and Jong (2013)] requires

more adders and more multiplier and offers nearly the same throughput rate. But the structure of [Hu and Jong (2013)] involves nearly 7*N* less on-chip memory words than those required by the structure of [Mohanty and Meher (2013)] and [Mohanty and Mahajan (2013)] involves nearly 4*N* more on-chip memory words. Compare with the structure of [Hu and Jong (2013)], the proposed structure involves less multiplier, less adders, less pipeline registers, 4*N* more on-chip memory words, and offers nearly the same throughput rate. But the proposed structure involves nearly 4*N* more on-chip memory words. Besides, the proposed structure has higher degree of circuit regularity and modularity than the structure of [Hu and Jong (2013)]. Each PU of the parallel structure of [Hu and Jong (2013)] needs separate design whereas PUs of proposed structure is obtained from a generic structure. PUs of the parallel structure of [Hu and Jong (2013)] involves PEs of two different types whereas the proposed structure involves only one type of PE design. Moreover, except the first PU, all other PUs of [Hu and Jong (2013)] involves specially designed transposition unit and a splitter whereas the proposed structure uses data-reordering units which are implemented using hard-wiring.

Table 3.1: General comparison of hardware and time complexities

Designs	**Block Size**	**Multiplier**	**Adder**	**MUX /DEMUX**	**Reg.**	**Clock cycle period**	**CT**
Mohanty and Meher * [2013]	S	$6Sx_1$	$32Sx_1/3$	$S(4x_1+x_5)$	$(11Nx_2+2Nx_4)$ $32Sx1/3$	T_M+2T_A $+T_x$	N^2/S
Mohanty and Mahajan [2013]	S	$6Sx_1$	$32Sx_1/3$	$2Sx_1$	$8(S+N)x_2$ $+32Sx_1/3+2Nx_1$	T_M+2T_A	N^2/S
Hu and Jong + [2013]	S	$(40Sx_1/6)$ $+6$	$(64Sx_1/6)$ $+12$	$4Sx_1$	$S[12x_1+8x_2+x_5+4$ $(2-3\times 2^{-J}+2^{-J})/3]$ $+4Nx_1$	T_M+T_A+ $(J-1)T_X$	N^2/S
Proposed	S	$6Sx_1$	$32Sx_1/3$	0	$8(S+N)x_2$ $+32Sx_1/3$	T_M+2T_A	N^2/S

*extra fame-buffer $2Nx_4$ required for folding the data-blocks
+extra frame-buffer $7N$ for overlapping input data-blocks
LEGEND: MULT: multiplier, ADD: adder, CP: clock cycle period, CT: computation time to process on complete image of size ($N\times N$), $x_1=1-2^{-2J}$, $x_2=1-2^{-J}$, $x_3=1-2^{1-J}$, $x_4=1-2^{-J+1}$, $x_5=1-2^{-2J+2}$, T_A=adder delay, T_M=multiplier delay, T_X=2:1 multiplexer delay.

Table 3.2: Hardware and time complexities of the proposed structure and the existing structures for 9/7 wavelet filters and input image size (256 × 256)

Designs	DWT level	Block Size	MULT.	ADD	MUX	REG (data)	REG (Pipeline)	CT (in ccs)
Mohanty and Meher (2013)	2	16	90	160	72	9.25N	160	N^2/16
	3	64	378	672	312	11.5N	672	N^2/64
Mohanty and Mahajan** (2013)	2	16	90	160	30	~ 6N	196	N^2/16
Hu and Jong (2013)	2	16	106	172	60	2N	256	N^2/16
	3	64	426	684	252	3N	908	N^2/64
Proposed	2	(4×4)	90	160	0	~ 6N	160	N^2/16
	3	(8×8)	378	672	0	~ 7N	672	N^2/64

LEGEND: MULT: multiplier, ADD: adder, BM: Booth multiplier, CT: computation time, ccs: clock cycles. **Structure is only for J=2.

The hardware complexity of the proposed structure and existing structure varies widely for different block-sizes and DWT levels. Therefore, the hardware complexity of the proposed structure and structure [Hu and Jong (2013)] which is the best structure amongst the existing similar designs is estimated for block sizes 16, 64, and DWT level J=2, 3. The estimated values are listed in Table 3.2 for comparison. As shown in Table 3.2, for J=2 and block size 16, the proposed structure involves 16 less multipliers, 12 less adders, 60 less multiplexers, 96 less pipeline registers than those required by the structure of [Hu and Jong (2013)], and involves nearly the same computation time. For J=3 and block size 64, the proposed structure involves 48 less multipliers, 12 less adders, 252 less multiplexers and 236 less pipeline registers than those required by the structure of [Hu and Jong (2013)]. However, the proposed structure involves extra 4N on-chip memory words against 7N extra frame-buffer word required by the structure of [Hu and Jong (2013)] for overlapped input data blocks. The structure of [Hu and Jong (2013)] also involves extra memory bandwidth than the proposed structure due to overlapped input data-

blocks. Since the frame-buffer implemented external to the chip, the extra 4*N* on-chip memory words of the proposed structure contribute significantly to the core area in-spite of having significant saving in area from multiplier, adder and pipeline registers than the existing structure.

Table 3.3: Synthesis results of the proposed structure using radix-4 Booth multiplier and the structures of [Hu and Jong (2013)] for 9/7 wavelet filters and image size (256 × 256)

Designs	DWT level	Block Size	MCP (ns)	Combinational area(μm^2)			Sequential area (μm^2)		Total area (μm^2)	Power (mW)	ADP (μm^2 ×ms)
				MULT	ADD	MUX	REG (data)	REG (Pipeline)			
Hu and Jong (2013)	2	16	1.76	169737	25820	7128	84050	20992	307727	12.1	2.23
	3	64	1.81	453185	102682	9608	126074	74456	766000	29.9	1.42
Pro-posed	2	(4×4)	1.78	159647	24019	0	233711	13120	430497	13.8	3.13
	3	(8×8)	1.84	428689	100880	0	294174	55104	878847	31.6	1.65

LEGEND: MULT: multiplier, ADD: adder, ADP: area-delay product =Area ×MCP×CT.

3.5.1 SYNTHESIS RESULTS

We have coded the proposed structure and the structure of [Hu and Jong (2013)] for block-size 16 and 64 in VHDL without the frame-buffer to estimate area, delay and power consumption of the DWT core. We have assumed 8-bit pixel for the input data-block and 12-bit for the samples of intermediate and output data-blocks and considered radix-4 Booth multiplier for the implementation of multipliers of both the designs. Multiplier results are post-truncated to 12-bit for fixed-point implementation. As shown in Table 3.3, the proposed structure has marginally higher minimum clock period (MCP) than the structure of [Hu and Jong (2013)] for block-size 16 and J=2 as well as for block-size 64 and J=3. For J=2 and block size 16, the proposed structure involves 8% less multiplier area, 7% less adder area, 18% less pipeline register area and 64% higher on-chip memory area than those of [Hu and Jong (2013)] respectively. Similarly, for J=3 and block-size 64, the proposed structure involves 11% less multiplier area, 7% less adder area, 25% less pipeline register area and 64% higher on-chip memory area than those of [Hu and

Jong (2013)] respectively. The area penalty due to extra on-chip memory of proposed structure is higher than the total area saving from the multiplier, adder and pipeline register. Therefore, the proposed structure involve 40% and 16% more ADP and consumes 14% and 5% more power than those of the structure of [Hu and Jong (2013)] for (J=2, block size 16) and (J=3, block size 64), respectively.

3.6 CONCLUSION

In this chapter, the data-reordering in row-column computation of 2-D DWT for line-based, stripe-based and overlapped stripe-based data-access schemes are studied. Based on this study a novel data-access scheme is formulated to avoid data-multiplexing which is common in the existing parallel architectures. A block formulation is presented for vector computation of multi-level lifting 2-D DWT. Using the proposed block-formulation a generic processing unit design is presented. The proposed generic design is controlled by a single parameter *i.e.* the input-vector size. A regular and modular parallel architecture is derived using the generic processing unit design. The proposed parallel architecture is easily scalable for higher block-sizes as well as higher DWT levels without sacrificing its circuit regularity and modularity. This is an important feature of the proposed architecture. The proposed parallel architecture is, therefore, suitable for high-throughput realization of multi-level lifting 2-D DWT. Compared to [Hu and Jong [(2013)] structure, the proposed structure involves less multiplier area, less adder area, less pipeline register area. This is mainly due to the higher number data-registers required by the proposed structure than the existing structure for the on-chip memory. Note that the existing structure uses overlapped data accessing scheme to save 4N on-chip memory words at the cost of 7N extra input buffer words which is an overhead. Using overlapping data-accessing scheme the on-chip memory complexity of proposed structure could be reduced, but the use of overlapping data accessing scheme for reducing the on-chip memory is not a good design approach. The synthesis result also reveals that memory complexity no longer become a major component when the parallel design is considered for higher block sizes (block size 64 and beyond). Therefore, optimizing multiplier complexity in parallel design could be a better design approach to find a hardware efficient parallel design. This issue is addressed in **Chapter 4** and **Chapter 5**.