Memory Efficient Modular VLSI Architecture for High-Throughput and Low-Latency Implementation of Multilevel Lifting 2-D DWT

Basant K. Mohanty, Member, IEEE and Pramod K. Meher, Senior Member, IEEE,

Abstract

In this paper, we present a modular and pipeline architecture for lifting-based multilevel 2-D DWT, without using line-buffer and frame-buffer. Overall area-delay product is reduced in the proposed design by appropriate partitioning and scheduling of the computation of individual decomposition-levels. The processing for different levels is performed by a cascaded pipeline structure to maximize the hardware utilization efficiency (HUE). Moreover, the proposed structure is scalable for high-throughput and area-constrained implementation. We have removed all the redundancies resulting from decimated wavelet filtering to maximize the HUE. The proposed design involves $L$ pyramid algorithm (PA) units and one recursive pyramid algorithm (RPA) unit, where $R = N/P$, $L = \lceil \log_4 P \rceil$, and $P$ is the input block size, $M$ and $N$, respectively, being the height and width of the image. The entire multilevel DWT is computed by the proposed structure in $MR$ cycles. The proposed structure has $O(8R \times 2^L)$ cycles of output latency, which is very small compared to the latency of the existing structures. Interestingly, the proposed structure does not require any line-buffer or frame-buffer, unlike the existing folded structures which otherwise require a line-buffer of size $O(N)$ and frame-buffer of size $O(M/2 \times N/2)$ for multilevel 2-D computation. Instead of those buffers, the proposed structure involves only local registers and RAM of size $O(N)$. The saving of line-buffer and frame-buffer achieved by the proposed design is an important advantage, since the image size could very often be as large as $512 \times 512$. From the simulation results we find that, the proposed scalable structure offers better slice-delay-product (SDP) for higher throughput of implementation since the on-chip memory of this structure remains almost unchanged with input block size. It has 17% less SDP than the best of the corresponding existing structures on average, for different input-block sizes and image sizes. It involves 1.92 times more transistors, but offers 12.2 times higher throughput and consumes 52% less power per output (PPO) compared to the other, on average for different input sizes.

Index Terms

Systolic array, VLSI, lifting, discrete wavelet transform (DWT), 2-dimensional (2-D) DWT.

I. INTRODUCTION

The discrete wavelet transform (DWT) is well-known for its applications in image and video compression, fractal analysis, texture discrimination, computer graphic, and bioinformatics etc [1]–[5]. Due to its remarkable advantage over the discrete cosine transform (DCT) in image compression, 2-D DWT has been accepted for the JPEG-2000 compression standard [7]. The implementation of 2-D DWT, however, is highly computation-intensive and many of its applications demand real-time processing. High-speed implementation of this transform to meet the temporal requirement of real-time applications is, therefore, considered as a challenging task. Keeping these in view, several architectures have been suggested in the last decade for its efficient implementation in dedicated VLSI systems.

Many of the existing DWT architectures are based on linear convolution property of the wavelet filters [8]–[11], [29]. The lifting scheme of computation of DWT [12] has, however, become more popular over the convolution-based scheme for its...
lower computational complexity [13], [14]. Moreover, the lifting-based wavelet decomposition has many useful properties like symmetric forward and inverse transform, in-place computation and integer-to-integer wavelet transform. Most importantly, lifting-based computation involves much less number of multipliers, adders and storage elements compared to the convolution-based algorithm. The above attributes make the lifting scheme more suitable for low-complexity hardware implementation of DWT. The overall hardware complexity of the 2-D DWT structures is broadly divided into two components, arithmetic component and memory component. The arithmetic component comprises multipliers, adders and the memory component comprises transposition-buffer, temporal-buffer and the frame-buffer. Transposition and temporal buffer are usually on-chip, while frame-buffer is located off-chip because it is large. The size of the on-chip and off-chip memory greatly affects the speed and power performance of the 2-D DWT structure.

Several lifting-based architectures have been suggested for efficient implementation of 2-D DWT [15] - [41]. In the following we briefly discuss a few of them. Andra et al [15] have proposed a four-processor block-based architecture, which requires large internal buffer and involves significantly high latency. Wu et al [8] have proposed a line-based scanning scheme and a folded architecture for the computation of multilevel 2-D DWT level-by-level. By line-based scanning, a few rows of intermediate output matrix are buffered to perform the column-wise processing. Consequently, the transposition buffer size is reduced from $O(M \times N)$ to $O(N)$, where $M$ and $N$ are, respectively, the height and width of the input image. The folded structure of [8], therefore, requires a small internal buffer, involves simple control circuitry and performs multilevel DWT computation using frame-buffer of size $(M/2 \times N/2)$ with 100% HUE. An external frame-buffer is used in this structure to store the low-low subband components of the current decomposition level for the calculation of subband coefficients of the next higher levels. Several other folded structures also have been proposed [17], [19]–[37] for efficient implementation of lifting 2-D DWT. All these structures differ in terms of size of arithmetic-unit, on-chip memory, cycle period and average computation time (ACT). The dual scan 2-D architecture proposed by Liao et al [17] requires two streams of input samples and computes the folded multilevel DWT computation with 100% HUE. The folded design proposed by Barua et al [19] uses an embedded symmetric data extension scheme for 9/7 DWT computations. The structures of [22]–[24], [26], [33] have used the embedded decimation
technique for low-complexity implementation of 2-D DWT using 9/7 filters. Amongst these, the structure of [24], requires smaller line-buffer than others due to its efficient row-column processing. The structures of [23], [25], [33] on the other hand have a smaller cycle period than others. Cheng et al [20], [35] have proposed the multiple lifting scheme and $M$-scan method to reduce the memory (line-buffer) access for reducing the overall power consumption at the cost of a few local registers. Lai et al [21] have used parallel data access scheme of [20] to avoid line-buffers to implement the transposition in the folded design. Xiong et al [28], [30]–[32] have proposed several architectures for efficient implementation of lifting 2-D DWT. The folded architectures of [28] and [30] are the most efficient among them. They have used both parallel and pipeline processing techniques in [28] and [30] for achieving higher throughput rates.

Some recursive architectures based on recursive pyramid algorithm (RPA) have also been proposed [16]–[18], [28] for concurrent implementation of multilevel lifting 2-D DWT. The recursive architecture proposed by Xiong et al [28] is the most efficient and fastest one. Recently, Cheng et al [29] have proposed a convolution-based recursive architecture for 2-D DWT using 9/7 filters, where throughput rate is increased in a controlled manner. However, the structure of [29] involves large on-chip storage ($30N$) and long cycle period than the lifting-based structures. The existing folded designs and recursive designs have some inherent difficulty for efficient realization of 2-D DWT. The folded designs require small internal buffer and simple control circuitry but involve large frame-buffer, where the frame-buffer not only contributes to the power budget, but also introduces significant delay in multilevel DWT computation. A significant amount of memory bandwidth is also wasted by accessing the external buffer which ultimately limits the throughput rate. The recursive structures eliminate the requirement of external buffer but involve complex control circuits and more internal memory than the folded structures. The HUE of the recursive designs is also not favorable for efficient realization of the 2-D DWT core. The line-buffer and frame-buffer are found to contribute almost 90% of the chip area and power dissipation in the existing structures.

Various data access schemes and architectural designs are proposed in the literature to reduce size of the line-buffer and frame-buffer for efficient implementation of 2-D DWT in systolic VLSI. The size of transposition buffer used for separable 2-D DWT entirely depends on when and how the intermediate coefficients are generated and consumed. The frame-buffer can be
eliminated by pipeline implementation of multilevel algorithm. Unlike the recursive structures, the pipeline structure involves a simple control circuit, but the direct mapping of multilevel DWT into a pipeline structure results poor HUE. Keeping this in view, we attempt to develop an alternative computing scheme with an objective to derive a regular structure for multilevel 2-D DWT which would involve simple control circuitry and can perform multilevel computation with 100% HUE, without using line-buffer and frame-buffer. The key ideas used in our proposed approach is,

- to increase the level of parallel processing in order to generate sufficient number of intermediate coefficients along the row direction in each cycle, such that the processing of intermediate results along column direction can be initiated as early as possible to reduce the size of the transposition buffer.

- to process each level of 2-D DWT computation in separate computing blocks in cascaded pipeline structure, for concurrent computation of multilevel DWT without buffering the subband components.

- in order to achieve 100% HUE in each computing block, the input rows for higher DWT levels are appropriately folded to utilize the down-sampling of DWT coefficients effectively.

Using the above approach, we have avoided both the line-buffer and the frame-buffer in the proposed structure. It involves small on-chip storage and simple control circuitry for computing the multilevel 2-D DWT with 100% HUE. The proposed design not only provides high throughput rate but also helps to reduce the overall latency of computation.

In recent years, there has been a phenomenal increase in the use of portable and wireless hand-held devices for multimedia applications. Since, the portable and wireless hand-held devices are constrained by hardware resources, in this paper we propose a flexible design for multilevel lifting 2-D DWT which could easily be scalable for hardware as well as the throughput. The scalable design could be configured for maximum throughput to optimize the usage of on-chip storage and throughput could otherwise be traded for area or power. Keeping these in view, we have designed scalable array architectures for 2-D DWT specifically for 9/7 filters, which is used in JPEG-2000 as default filter for lossy compression. The proposed structure, however, can be easily modified to implement the 2-D DWT for any other wavelet filters, which could be realized through finite lifting steps.
The paper is organized as follows: Necessary mathematical formulation is presented in Section II. The proposed structures for multilevel lifting 2-D DWT are presented in Section III. Hardware and time complexities of the proposed structures are discussed in Section IV. Conclusion is presented in Section V.

II. MATHEMATICAL FORMULATION

The lifting computation of 9/7 filter can be performed using the following relations:

\[ s_1(i) = x(2i - 1) + \alpha (x(2i) + x(2i - 2)) \]  
\[ s_2(i) = x(2i - 2) + \beta (s_1(i) + s_1(i - 1)) \]  
\[ v_1(i) = s_1(i - 1) + \gamma (s_2(i) + s_2(i - 1)) \]  
\[ v_2(i) = s_2(i - 1) + \delta (v_1(i) + v_1(i - 1)) \]  
\[ v_h(i) = K^{-1} v_1(i) \]  
\[ v_l(i) = K v_2(i) \]

for \( 0 \leq i \leq (N/2) - 1 \).

where \( \alpha, \beta, \gamma \) and \( \delta \) are, lifting constants and \( K \) is the scale normalization factor.

Using separable approach, row-wise and column-wise 1-D lifting DWT computation can be performed on the input data matrix to obtain the 2-D DWT coefficients. For the \( j \)-th level decomposition, the low-low subband of \( (j - 1) \)-th level \( (A^{j-1}) \) is decomposed into four subbands namely low-low \( (A^j) \), low-high \( (B^j) \), high-low \( (C^j) \) and high-high \( (D^j) \) subbands. The input data matrix \( (X) \) represents the low-low subband of the zeroth level. The computation of 2-D lifting DWT, therefore, can be decomposed into two distinct stages as follows:

- In stage-1, 1-D lifting DWT is performed on the rows of the input matrix \( [A^{j-1}] \) of size \( [M/2^{j-1} \times N/2^{j-1}] \) to obtain the low-pass intermediate matrix \( [U^{j-1}] \), and the high-pass intermediate matrix \( [U^{j-1}_h] \), each of size \( [M/2^{j-1} \times N/2] \).
- In stage-2, 1-D lifting DWT is again performed on the columns of two intermediate output matrices \( [U^{j-1}] \) and \( [U^{j-1}_h] \) to obtain the desired 2-D DWT coefficients \( \{A^j(m, n)\}, \{B^j(m, n)\}, \{C^j(m, n)\}, \text{ and } \{D^j(m, n)\} \) of size \( [M/2^j \times N/2^j] \).

The following set of recursive relations (similar to (1)) are derived, in general form, for the computation of stage-1 and stage-2 of the \( j \)-th level 2-D lifting DWT.

\[ s_{11}(m, n) = x(m, 2n - 1) + \alpha ((x(m, 2n) + x(m, 2n - 2)) \]  
\[ s_{12}(m, n) = x(m, 2n - 2) + \beta ((s_{11}(m, n) + s_{11}(m, n - 1)) \]  
\[ u_h(m, n) = s_{11}(m, n - 1) + \gamma ((s_{12}(m, n) + s_{12}(m, n - 1)) \]  
\[ u_l(m, n) = s_{12}(m, n - 1) + \delta ((u_h(m, n) + u_h(m, n - 1)) \]

(3)
Since the computation of the low-pass and high-pass components of intermediate outputs \((u_l(m,n))\) and \((u_h(m,n))\), respectively, are of similar form, the computations of those components for stage-2 are expressed in a general form as:

\[
\begin{align*}
    s_{21}(m,n) &= u(2m-1,n) + \alpha((u(2m,n) + u(2m-2,n)) \\
    s_{22}(m,n) &= u(2m-2,n) + \beta((s_{21}(m,n) + s_{21}(m-1,n)) \\
    v_h(m,n) &= s_{21}(m-1,n) + \gamma((s_{22}(m,n) + s_{22}(m-1,n)) \\
    v_l(m,n) &= s_{22}(m-1,n) + \delta((v_h(m,n) + v_h(m-1,n))
\end{align*}
\]

(4)

where \(x(m,n)\) represents the low-low sub-band components of the \((j-1)\)-th level. \(u(m,n)\) represents the intermediate output corresponding to the input \(x(m,n)\), which could be low-pass and high-pass component \(u_l(m,n)\) and \(u_h(m,n)\), respectively.

Similarly, \(v_h(m,n)\) is the high-pass output corresponding to the intermediate output \(u_l(m,n)\) and \(u_h(m,n)\), which, respectively, represent the pair of subband outputs \(B(m,n)\) and \(\bar{D}(m,n)\). \(v_l(m,n)\) is the low-pass outputs corresponding to the intermediate output \(u_l(m,n)\) and \(u_h(m,n)\) which, respectively, represent the other two subband outputs \(\bar{A}(m,n)\) and \(C(m,n)\).

The scale normalization of stage-1 and stage-2 of lifting 2-D DWT can be integrated and performed in one step at the end of the stage-2 using the following equation:

\[
\begin{align*}
    A(m,n) &= K^2.\bar{A}(m,n) \\
    D(m,n) &= K^{-2}.\bar{D}(m,n)
\end{align*}
\]

(5)

III. THE PROPOSED ARCHITECTURE

Let us assume that a block of \(P\) samples of a row is fed in parallel to the structure such that one row of \(N\) samples is fed in \(R\) cycles, where \(N = RP\). The complete input matrix of size \((M \times N)\) is fed to the structure in row-by-row in bottom to top order in \(MR\) cycles. The data-flow graph (DFG) of the computation of the stage-1 (referred to as DFG-1) of the lifting 2-D DWT pertaining to an input block of \(P\) samples and \(P/2\) outputs corresponding to the \(r\)-th cycle is shown in Fig.1. It consists of \(P/2\) sections where each section represents lifting computation of a 9/7 filter in four separate lifting steps according to (3).

The inputs and outputs of \((i+1)\)-th section of DFG-1 pertaining to all the four lifting steps are shown in Table I.

As shown in Fig.1(a), the partial outputs \([s_{11}(m,P(r+1)/2-1), s_{12}(m,P(r+1)/2-1), u_h(m,P(r+1)/2-1)]\) of \(P/2\)-th section of the \(r\)-th cycle are used by the first section during \((r+1)\)-th cycle. \((P/2)\) samples of a particular row of both low-pass and high-pass intermediate matrix \([U_l]\) and \([U_h]\) of size \((M \times N/2)\) are computed in every cycle, and one complete row of \(N/2\) samples is processed in \(R\) cycles. Note that, all the \(R\) output samples of each section of DFG-1 correspond to the same input row. Similar to DFG-1, the DFG for the computation of stage-2 (referred to as DFG-2), also consists of \((P/2)\)
Fig. 1. Data flow graph (DFG) of the proposed computing schedule of lifting 2-D DWT using 9/7 filter. (a) DFG of stage-1 (DFG-1) for the computation of identical sections. The necessary delay to the 2-D input data stream for the processing of stage-2 along column direction.

(b) From Input-Interface Unit

TABLE I

<table>
<thead>
<tr>
<th>Lifting-step</th>
<th>Input-1</th>
<th>Input-2</th>
<th>Input-3</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(x(m, Pr + 2i))</td>
<td>(x(m, Pr + 2i + 1))</td>
<td>(x(m, Pr + 2i + 2))</td>
<td>(s_{11}(m, Pr/2 + i))</td>
</tr>
<tr>
<td>2</td>
<td>(x(m, Pr + 2i))</td>
<td>(s_{11}(m, Pr/2 + i))</td>
<td>(s_{11}(m, Pr/2 + i - 1))</td>
<td>(s_{12}(m, Pr/2 + i))</td>
</tr>
<tr>
<td>3</td>
<td>(s_{11}(m, Pr/2 + i - 1))</td>
<td>(s_{12}(m, Pr/2 + i))</td>
<td>(s_{12}(m, Pr/2 + i - 1))</td>
<td>(u_{1}(m, Pr/2 + i))</td>
</tr>
<tr>
<td>4</td>
<td>(s_{12}(m, Pr/2 + i - 1))</td>
<td>(u_{1}(m, Pr/2 + i))</td>
<td>(u_{1}(m, Pr/2 + i))</td>
<td>(v(m, Pr/2 + i))</td>
</tr>
</tbody>
</table>

\(s_{11}(m, Pr/2 + i - 1)\), \(s_{12}(m, Pr/2 + i - 1)\) and \(u_{1}(m, Pr/2 + i - 1)\) are, respectively, the partial and intermediate outputs corresponding to \(i\)-th section.

identical sections. The \((i + 1)\)-th section of DFG-2 is shown in Fig.1(b). It includes \(R\) delay elements (RD) to provide the necessary delay to the 2-D input data stream for the processing of stage-2 along column direction.
A. Scalable Parallel Architecture for 1-level Lifting 2-D DWT

To derive this structure, we take the DFG-1 of \((P/2)\) sections for parallel computation of \((P/2)\) pairs of intermediate outputs corresponding to a block of \(P\) samples of an input row in every cycle. The computation of each section is mapped to a subcell-1. As shown in Fig.2 (a), the computation of \((i + 1)\)-th section is mapped to \((P/2 − i)\)-th subcell-1. The scalable parallel structure thus consists of \((P/2)\) subcell-1 to process one complete row of \(N\) input values in \(R\) cycles. To deal with the boundary data loss due to filtering, the original image is symmetrically extended by one column before being fed to the computing structure. The \((N + 1)\)-th column is, therefore, replicated by the sample values of the \((N)\)-th column.

The input rows are folded by a factor \(R\), \((R\) is assumed to be a power of 2) and fed block-by-block serially in \(R\) successive cycles to the structure, such that the complete image of size \((M \times N)\) is fed in \(MR\) cycles. Each input data-block \((X_{m,r})\) is comprised of \((P + 1)\) consecutive samples of a given row and the successive data-blocks, for \(0 \leq r \leq R − 1\) are overlapped by one sample, where \(0 \leq m \leq M − 1\). In each cycle, \((P + 1)\) parallel samples fed to the structure such that \((P/2 − i)\)-th subcell-1 receives the samples \(x(m, Pr + 2i + 2), x(m, Pr + 2i + 1), x(m, Pr + 2i)\) for \((0 \leq i \leq (P/2) − 1)\).

Subcell-1 consists of four functional units (FU) connected in a systolic pipeline, where each FU performs one multiplication and two addition pertaining to the computation of one lifting step in every cycle using one multiplier and two adders. Internal structure of FU is shown in Fig.2 (b). The \((P/2 − i)\)-th subcell-1 produces a pair of high-pass and low-pass outputs in each
cycle after a latency of three cycles, where the duration of clock period \( T = T_M + 2T_A \). Note that successive \( P/2 \) outputs of each subcell-1 belong to the same row of the intermediate matrix \([U_i/U_h]\). These intermediate outputs are appropriately buffered for the computation of stage-2 along the column direction.

The computation of stage-2 involves separate wavelet filtering of the columns of intermediate output of stage-1 in order to calculate four subband components \([A, B, C, D]\). DFG-2 [Fig.1(b)] accepts one intermediate output in each cycle and produces a pair of subband outputs in time-multiplexed form. Two such DFGs corresponding to low-pass and high-pass subbands are mapped into a subcell-2 (Fig.3(a)), such that \([U_h]\) and \([U_l]\) are processed in time-multiplexed form. The input delay line of the DFG-2 is implemented by a serial-in parallel-out shift-register (SIPOS). Subcell-2 uses two such SIPOS for the elements of \([U_h]\) and \([U_l]\).

For scalable parallel computation of stage-2, the proposed structure requires \((P/2)\) number of subcell-2. The \((P/2 - i)\)-th subcell-2 receives pairs of intermediate outputs \(\{u_h(m, Pr/2+i), u_l(m, Pr/2+i-1)\}\) and \(\{u_h(m, Pr/2+1+i), u_l(m, Pr/2+i)\}\) from \((P/2 - i)\)-th subcell-1 in \((mr + 4)\)-th and \((mr + 5)\)-th cycle respectively. During even-numbered cycles it performs filtering of high-pass component and in the odd-numbered cycles, it performs filtering of low-pass component, such that each
subcell-2 performs the down-sampled wavelet filtering pertaining to even-indexed start values of the \((Pr/2 + i)\)-th column of the intermediate matrices \([U_l]\) and \([U_h]\) in alternate cycles.

Each subcell-2 (see Fig.3(a)) consists of four FUs, one input delay unit (ID), three middle delay units (MD) and one scale normalization unit (SU). The input delay-line of high-pass is comprised of 2 shift-registers (SR), while the input delay-line of low-pass component is comprised of 3 SRs. Each MD is comprised of 2 SRs. Except SR1, all other SRs of ID are of size \(R\) words each, while SR1 is of size \((R − 1)\) words because additional one-sample delay is imposed by subcell-1 while generating the low-pass intermediate outputs. Both the input delay-lines can be implemented using two SIPOSRs. The ID requires three 2-to-1 MUXes (MUX-1) to select the SIPOSR contents in alternate set of \(R\) cycles such that during the even numbered set of \(R\) cycles, MUX-1 selects the high-pass components and select the low-pass components of \([U_l]\) during odd-numbered set of \(R\) cycles. The MD provides one row delay to the partial result (as required by (4)) computed by each of the FUs, except FU-4 of the subcell-2. Since the columns of \([U_h]\) and \([U_l]\) are processed in successive sets of \(R\) cycles, each MD requires two SRs (of \(R\) words) to hold one row of partial results of \([U_h]\) and \([U_l]\). It is implemented by a pair of serial-in serial-out (SISO) shift-register. The \((P/2 − i)\)-th subcell-2 produces a pair of DWT coefficients of subband matrices \([C, \tilde{D}]\) or \([B, \tilde{A}]\) in every cycle.

The outputs of 9/7 filter obtained by the lifting-scheme are required to be normalized by a scaling factor \((K)\) according to (2). In the proposed structure, the scaling operation of stage-1 and stage-2 are integrated and performed at the end of stage-2 to save some multiplications, where the components of the matrices \([\tilde{D}]\), and \([\tilde{A}]\) are normalized by multiplying with the constants \(K^{-2}\) and \(K^2\) according to (5). The scaling operation is performed by a separate functional unit (SU) as shown in Fig.3(b).
is comprised of one multiplier and three 2-to-1 MUXes (MUX-2, MUX-3, MUX-4). As given by (5), the components of the subbands \([B]\) and \([C]\) do not require scaling. MUX-2 selects these components to by-pass those scaling operations. MUX-3 selects the components of \([\hat{D}], [\hat{A}]\), and MUX-4 selects the corresponding multiplying constants for the scaling. All the three MUXes of the SU are controlled by a common select signal. The FUs and SU work in separate pipeline stages. The latency of subcell-2 is 5 cycles.

A subcell-1 and a subcell-2 can be integrated to form a processing element (PE). Structure of a PE is shown in Fig.4, where subcell-1 and subcell-2 work in two separate pipeline stages. The \(((P/2) - i)\)-th PE performs the computation of stage-1 and stage-2 pertaining to \(i\)-th section of DFG-1 and DFG-2. The proposed scalable parallel structure is comprised of \((P/2)\) identical PEs, and processes a block of \(P\) input samples in one cycle, and one complete row in \(R\) cycles. All \((P/2)\) PEs are integrated to form a processing unit (PU). The PU for the computation of the first-level DWT is termed as PU-1. The structure of PU-1 is shown in Fig.5. It uses a data buffer of size 3 words (not shown in Fig.5) to store the intermediate outputs of first subcell-1. During every period of \(R\) cycles, the data buffer of PU-1 stores partial results of \((P/2)\)-th section of DFG-1 (see Fig.1(a)), such that the partial results of \((r + 1)\)-th cycle are used by \((P/2)\)-th subcell-1 during the \((r + 2)\)-th cycle, (for \(r = 0, 1, 2, ..., R - 1\)). At the end of \(R\)-th cycle, the data buffer is cleared for the next input row. During each cycle, PU-1 receives a block of \(P\) samples of an input row, and produces \(P/2\) components of a pair of subband coefficients \([C^1, D^1]\) or
Fig. 6. Proposed scalable structure for \( J \)-level lifting 2-D DWT, where \( J = \min(\lceil \log_2 M \rceil, \lceil \log_2 N \rceil) \), \( X_{m,r} = \{x(m, Rr + P), x(m, Rr + P - 1), \ldots, x(m, Rr + 1), x(m, Rr)\} \), for \( 0 \leq m \leq M - 1, 0 \leq p \leq P - 1, 0 \leq r \leq R - 1, N = PR \)

\([B^1, A^1]\), such that two complete rows of a pair of subband are computed in \( R \) cycles. In the next \( R \) cycles, PU-1 produces two complete rows of other two subbands. During each even-numbered set of \( R \) cycles, PU-1 produces one row of subbands \([C^1]\) and \([D^1]\), and during each odd-numbered set of \( R \) cycles, it produces one row of \([B^1]\) and one row of \([A^1]\). One complete row of each of the four subband matrices is obtained in \( 2R \) cycles. PU-1, therefore, completes the first level DWT of input of size \((M \times N)\) in \( MR \) cycles. Processing units similar to PU-1 also can be designed to compute the 2-D DWT of higher decomposition levels as well. All these PUs can be integrated into a pipeline structure for high-throughput implementation of multilevel lifting 2-D DWT.

**B. Scalable Parallel Architecture for Multilevel \((J > 1)\) lifting 2-D DWT**

According to the pyramid-algorithm (PA), the low-low subband of a given decomposition level is processed further to generate DWT coefficients of the next higher level, and due to down-sampling, after each level of decomposition, the computational complexity steadily decreases by a factor of four. The amount of hardware resources required to calculate the DWT coefficients of every higher-level of decomposition should, therefore, be reduced by a factor of 4, in order to achieve 100% HUE. The proposed scalable parallel structure for multilevel DWT, based on the above view point, is shown in Fig.6. It is comprised of \((L + 1)\) PUs, where \( L = \lceil \log_4 P \rceil \). The PU-\((L + 1)\) is a RPA based structure [17], while all other PUs are simple PA units. In each cycle PU-1 receives a block of \( P \) samples of an input row and produces a block of components of a particular row of subband matrices \([C^1, D^1]\) or \([B^1, A^1]\). One row of subband \((A^1)\) is obtained from PU-1 after every \( 2R \) cycles.

Each block of subband output \((A^1)\) of PU-1 is split into two blocks of \((P/4)\) samples each, and fed to PU-2 in each cycle,
so that one complete row of \((A^1)\) is fed in \(2R\) cycles. The computation of PU-\(j\), (for \(j > 1\)) is similar to that of PU-1. Similar to PU-1 (see Fig.5), PU-\(j\) can be designed for the computation of \(j\)-th level DWT. It consists of \((P/2^{2j-1})\) number of PEs and calculates the \(j\)-th level DWT of an input block of \((P/2^{2j-2})\) samples in every cycle, where \(1 \leq j \leq L\) and \(L = \lceil \log_4 N \rceil\). Each row of matrix \(A^{j-1}\) is folded and fed to PU-\(j\) in \(2^{j-1} \times R\) cycles. Subcell-1 and subcell-2 of each PE of PU-\(j\) perform the computations involved in DFG-1 and DFG-2 of Fig.1(a) and (b), respectively, where in this case \(R\) is replaced by \(2^{j-1} \times R\) and \(P/2\) is replaced by \((N/2^{2j-1})\). Structures of PU-\(j\) is similar to that of PU-1, except that the size of each shift register of ID and MD of subcell-2 is \(2^{j-1}\) words. Each of the \((PU-\text{j})s\) (for \(2 \leq j \leq L\)) uses a separate input buffer \((\text{IB}_j)\). The structure of \(\text{IB}_j\) is shown in Fig.7. It is comprised two RAM units (RAM-1, RAM-2) and two multiplexers (MUX-1, MUX-2). \(\text{IB}_j\) receives a block of \(N'\) components in each cycle and a complete row of \([A^{j-1}]\) from PU-\((j-1)\) in \(R'\) cycles, after an interval of \(R'\) cycles. It yields a block of \(N'/2\) components of \([A^{j-1}]\) in one cycle, and one complete row in \(2R'\) cycles, where \(R' = 2^{j-2} R, N' = P/2^{2j-3}\) and \((2 \leq j \leq L)\).

Each RAM unit is comprised \(N'\) single-port RAM modules of \(R'\) words, where \(N'R' = N/2^{j-1}\). A block of \(N'\) components is buffered in a RAM unit in one cycle. In a period of \(4R'\) cycles, two consecutive rows of \([A^{j-1}]\) are buffered in \(\text{IB}_j\) since a row of \([A^{j-1}]\) arrives at \(\text{IB}_j\) after an interval of \(R'\) cycles. During the first quarter of every sets of \(4R'\) cycles, an even numbered row of \([A^{j-1}]\) is buffered in RAM-1 and an odd-numbered rows is buffered in RAM-2 during the third quarter. Each RAM unit uses \(R'\) memory-write cycles to buffer \(R'\) input blocks. Memory-write cycles of both the RAM units start
just after the memory-read cycles without any delay. Data blocks are accessed from the RAM unit in the same sequence as they were stored (first-in first-out order). Each data block is read in two consecutive memory-read cycles. Consequently, the entire contents of the RAM unit is read in $2R'$ memory-read cycles. Both RAM-1 and RAM-2 have the same number of non-overlapping read-write cycles such that, the input blocks are read from RAM-1 or RAM-2 without interruptions.

RAM-1 uses $Addr_1$ and $RW_1$ as address and read-write control signal, respectively, while RAM-2 uses $Addr_2$ and $RW_2$ as its address and read-write control signal, respectively. $Addr_2$ and $RW_2$ are the delayed versions of $Addr_1$ and $RW_1$, respectively. In each set of $4R'$ cycles, both the RAM units perform read-write operations during $3R'$ cycles. Select signals $cs_1$ and $cs_2$ are used to select RAM-1 and RAM-2, respectively. Input rows are buffered alternately in RAM-1 and RAM-2, as they arrive at $IB_j$ and retrieved in alternate sets of $2R'$ cycles. MUX-1 of $IB_j$ selects the values from the RAM units in alternate sets of $2R'$ cycles. In each cycle, MUX-1 therefore, selects a block of $N'$ values of RAM-1 or RAM-2. Each output block of MUX-1 is split into two equal parts $Y_1$ and $Y_2$. MUX-2 selects (using $sel_2$) input blocks $Y_1$ and $Y_2$ in alternate cycles. $IB_j$ uses one register (REG) to implement the necessary data extension and overlap of successive blocks ($Y_1$, $Y_2$) by one sample to meet the requirement of lifting computation. MUX-1 and MUX-2 are comprised of $N'$ and $(N'/2)$ 2-to-1 line MUXes, respectively. Since the input block size of $IB_j$ decreases steadily by factor of four after every higher values of $j$, minimum input block size $N'$ of $IB_j$ is taken to be 4.

PU-$j$ computes the four subbands matrices $(A^j, B^j, C^j, D^j)$ of the $j$-th decomposition level in $MR$ cycles, where $(1 \leq j \leq L)$. If $P$ is a power of 4, then PU-$L$ contains only two PEs, and produces a pair of components of $(A^L)$, in every couple of cycles, which belong to the adjacent even and odd columns of a particular row of $(A^L)$. A pair of components of $(A^L)$ are fed to the $(L+1)$-th PU in every alternate cycle to calculate DWT coefficients of higher levels ($J > L$) for $J = \min(\lfloor \log_2 M \rfloor, \lfloor \log_2 N \rfloor)$. For input block size $(N' = 2)$, $IB_j$ is not required by PU-$j$. If $P$ is power of 2 and not a power of 4, then PU-$L$ contains only one PE, and calculates one component of $(A^L)$ in alternate cycles. PU-$(L+1)$, therefore, receives a pair of DWT components from PU-$L$ in four cycles and performs the RPA computation. $IB_{L+1}$ in this case is comprised of three registers only.
In the recursive structure of [17], low-pass and high-pass DWT coefficients of lifting 2-D DWT are scaled separately after row and column transformation. But we have integrated these scaling operations into one step which is performed after the final column transformation according to (5). Consequently, we save three multipliers over the existing RPA structure. We perform the scaling operation of the RPA unit by using an SU (shown in Fig.3b) which involves only one multiplier. PU-$(L + 1)$, therefore, computes the DWT coefficients of all the higher subbands $(J > L)$ of the subband matrix $(A^L)$ of size $(M/2^L \times 2^L R)$ in $MR$ cycles, where $J = \min(\lfloor \log_2 M \rfloor, \lfloor \log_2 N \rfloor)$. Each PU of the proposed scalable structure operates in separate pipeline stages and, therefore, requires $MR$ cycles to complete the DWT computations of an input matrix of size $(M \times N)$.

IV. HARDWARE COMPLEXITY AND PERFORMANCE CONSIDERATION

A. Hardware- and Time-Complexity

The proposed scalable structure is comprised of $(L + 1)$ PUs. The PU-$j$ (for $1 \leq j \leq \lceil \log_4 P \rceil$) involves $(P/2^{2j-1})$ PEs. Subcell-1 of each PE consists of 4 multipliers, 8 adders and 6 pipeline delay registers; while subcell-2 consists of 5 multipliers, 8 adders, 6 MUXex and $(11 \times 2^{j-1} + 9)$ data/pipeline registers. Each PE, therefore, involves 9 multipliers, 16 adders, $(11 \times 2^{j-1}R + 15)$ registers and 6 MUXes, where $P = N/R$. PU-$j$ consists of $(9P/2^{2j-1})$ multipliers and $(P/2^{2j-5})$ adders. Apart from that, PU-$j$ except the PU-1, involves an intermediate-buffer $IB_j$ (for $2 \leq j \leq L + 1$), which comprises $(4N/2^j)$ RAM words and $(3P/2^{2j-2})$ MUXes. The proposed scalable structure for $J$ level DWT involves $(6P(1 - 2^{-2L}) + 9)$ multipliers, $\frac{32}{3} P(1 - 2^{-2L}) + 16$ adders, $[(11N(1 - 2^{-L}) + 10P(1 - 2^{-2L}) + (10N \times 2^{-L}(1 - 2^{-J+L})] + 5J - 4L)$ registers, $2N(1 - 2^{-L+1})$ RAM words, $P[4(1 - 2^{-2L}) + (1 - 2^{-2L+2})] + 33$ MUXes/DMUXes. For $L$ level DWT, if $P$ is power of 4, the proposed scalable structure involves $(6P(1 - 2^{-2L}))$ multipliers, $\frac{32}{3} P(1 - 2^{-2L})$ adders, $[(11N(1 - 2^{-L}) + 10P(1 - 2^{-2L}) + L)$ registers, $2N(1 - 2^{-L+1})$ RAM words, $P[4(1 - 2^{-2L}) + (1 - 2^{-2L+2})]$ MUXes/DMUXes. If $P$ is not a power of 4, (except the PU-$L$) all other PUs of the scalable structure has 100% HUE, while PU-$L$ has less than 100% HUE, and involves only one PE. The complexity of the scalable structure for $(L - 1)$ levels are estimated using the formula given above since these PUs have 100% HUE and complexity of the PU-$L$ is estimated separately.

The proposed scalable structure computes $J$-level DWT in $MR$ cycles after an initial latency of $(8L + 8R \times 2^L + R \times$
TABLE II

COMPARISON OF HARDWARE- AND TIME-COMPLEXITIES OF THE PROPOSED AND THE EXISTING STRUCTURES FOR MULTILEVEL-LEVEL 2-D DWT USING 9/7 FILTERS. (M: IMAGE HEIGHT, N: IMAGE WIDTH)

<table>
<thead>
<tr>
<th>Structures</th>
<th>MULT</th>
<th>ADD</th>
<th>REG</th>
<th>Line buffer in words</th>
<th>frame buffer in words</th>
<th>MUX/ DMUX</th>
<th>cycle period in cycles</th>
<th>Latency in cycles</th>
<th>ACT in cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Liao et al [17]</td>
<td>12</td>
<td>16</td>
<td>5</td>
<td>5.5N</td>
<td>NM/4</td>
<td>1</td>
<td>$T_M + 2T_A$</td>
<td>$O(NM/4)$</td>
<td>$2MNx_2/3$</td>
</tr>
<tr>
<td>Barua et al [19]</td>
<td>12</td>
<td>16</td>
<td>5</td>
<td>7N</td>
<td>NM/4</td>
<td>1</td>
<td>$T_M + 2T_A$</td>
<td>$O(NM/4)$</td>
<td>$2MNx_2/3$</td>
</tr>
<tr>
<td>Cheng et al [20]</td>
<td>12</td>
<td>16</td>
<td>24</td>
<td>4N</td>
<td>NM/4</td>
<td>1</td>
<td>$T_M + 2T_A$</td>
<td>$O(NM/4)$</td>
<td>$2MNx_2/3$</td>
</tr>
<tr>
<td>Lai et al [21]</td>
<td>10</td>
<td>16</td>
<td>4</td>
<td>4N</td>
<td>NM/4</td>
<td>1</td>
<td>$T_M$</td>
<td>$O(NM/4)$</td>
<td>$2MNx_2/3$</td>
</tr>
<tr>
<td>Li et al [37]</td>
<td>12</td>
<td>16</td>
<td>13</td>
<td>4N</td>
<td>NM/4</td>
<td>17</td>
<td>$T_M + 2T_A$</td>
<td>$O(NM/4)$</td>
<td>$2MNx_2/3$</td>
</tr>
<tr>
<td>Xiong et al [30] HA</td>
<td>18</td>
<td>32</td>
<td>6</td>
<td>5.5N</td>
<td>NM/4</td>
<td>3</td>
<td>$T_M + 2T_A$</td>
<td>$O(NM/4)$</td>
<td>$MNx_2/3$</td>
</tr>
<tr>
<td>Liu et al [27]</td>
<td>12</td>
<td>16</td>
<td>12</td>
<td>4N</td>
<td>NM/4</td>
<td>1</td>
<td>$T_M$</td>
<td>$O(NM/4)$</td>
<td>$2MNx_2/3$</td>
</tr>
<tr>
<td>Gao et al [22]</td>
<td>5</td>
<td>8</td>
<td>10</td>
<td>10N</td>
<td>NM/4</td>
<td>17</td>
<td>$T_M + 2T_A$</td>
<td>$O(NM/4)$</td>
<td>$4MNx_2/3$</td>
</tr>
<tr>
<td>Jain et al [26]</td>
<td>5</td>
<td>8</td>
<td>8</td>
<td>12N</td>
<td>NM/4</td>
<td>--</td>
<td>$T_M + T_A$</td>
<td>$O(NM/4)$</td>
<td>$4MNx_2/3$</td>
</tr>
<tr>
<td>Wu et al [23]</td>
<td>6</td>
<td>8</td>
<td>5</td>
<td>5.5N</td>
<td>NM/4</td>
<td>11</td>
<td>$T_M$</td>
<td>$O(NM/4)$</td>
<td>$4MNx_2/3$</td>
</tr>
<tr>
<td>Fatemi et al [24]</td>
<td>6</td>
<td>12</td>
<td>26</td>
<td>3N</td>
<td>NM/4</td>
<td>11</td>
<td>$T_M + 2T_A$</td>
<td>$O(NM/4)$</td>
<td>$4MNx_2/3$</td>
</tr>
<tr>
<td>Cao et al [25]</td>
<td>6</td>
<td>8</td>
<td>36</td>
<td>5.5N</td>
<td>NM/4</td>
<td>26</td>
<td>$T_M$</td>
<td>$O(NM/4)$</td>
<td>$4MNx_2/3$</td>
</tr>
<tr>
<td>Liao et al [17]</td>
<td>12</td>
<td>16</td>
<td>5J</td>
<td>$10N_{x_1}$</td>
<td>0</td>
<td>29</td>
<td>$T_M + 2T_A$</td>
<td>$O(N)$</td>
<td>$MN$</td>
</tr>
<tr>
<td>Huang et al [9]</td>
<td>10</td>
<td>16</td>
<td>5J</td>
<td>$10N_{x_1}$</td>
<td>0</td>
<td>29</td>
<td>$T_M + 2T_A$</td>
<td>$O(N)$</td>
<td>$MN$</td>
</tr>
<tr>
<td>Xiong et al [28] Par</td>
<td>28</td>
<td>48</td>
<td>8</td>
<td>$+5J$</td>
<td>$(5.5+5x_1)N$</td>
<td>0</td>
<td>$T_M + 2T_A$</td>
<td>$O(N)$</td>
<td>$MN/4$</td>
</tr>
<tr>
<td>Cheng et al [29]</td>
<td>24</td>
<td>76</td>
<td>16</td>
<td>$+60J$</td>
<td>$30N$</td>
<td>1</td>
<td>$T_M + 3T_A$</td>
<td>$O(N)$</td>
<td>$MN/3$</td>
</tr>
<tr>
<td>Proposed SC (J-level DWT)</td>
<td>$6P_{x_4}$</td>
<td>$32P_{x_4}/3$</td>
<td>$+16$</td>
<td>$N(11x_3 + 10x_5)/10P_{x_4} + x_8$</td>
<td>0</td>
<td>$P(4x_4 + 6x)$</td>
<td>$+33$</td>
<td>$O(8 \times 2^L + 10N/2^L)$</td>
<td>$MN/P$</td>
</tr>
<tr>
<td>Proposed SC (L-level DWT)</td>
<td>$6P_{x_4}$</td>
<td>$32P_{x_4}/3$</td>
<td>$+16$</td>
<td>$11N_{x_3} + 10P_{x_4}$</td>
<td>0</td>
<td>$P(4x_4 + 6x)$</td>
<td>$+33$</td>
<td>$O(8 \times 2^L)$</td>
<td>$MN/P$</td>
</tr>
</tbody>
</table>


When a new PU (PA unit) is cascaded to the proposed scalable structure, then the size of the SRs (in subcell-2) of other PUs (both PA and RPA units) of the structure are reduced by half as these PUs now perform the DWT computations for the next higher decomposition levels. Consequently, the total on-chip storage of the scalable structure remains almost unchanged.

$$(2^{L-1} - 1) + 10N/2^L$$ cycles, where $$(8L + 8R \times 2^L)$$ cycles of delay is introduced by $L$ PA units, $R \times (2^{L-1} - 1)$$ cycles of delay is introduced by the intermediate-buffer and $$(10N/2^L)$$ cycles of delay is introduced by the RPA unit, for $J = \min(\lfloor \log_2 M \rfloor, \lfloor \log_2 N \rfloor)$, $P = \lfloor \log_4 P \rfloor$, and $2 \leq L \leq 7$ to calculate the value of $x_6, x_7$.

$$(2^{L-1} - 1) + 10N/2^L$$ cycles, where $$(8L + 8R \times 2^L)$$ cycles of delay is introduced by $L$ PA units, $R \times (2^{L-1} - 1)$$ cycles of delay is introduced by the intermediate-buffer and $$(10N/2^L)$$ cycles of delay is introduced by the RPA unit, for $J = \min(\lfloor \log_2 M \rfloor, \lfloor \log_2 N \rfloor)$, $P = \lfloor \log_4 P \rfloor$, and $2 \leq L \leq 7$ to calculate the value of $x_6, x_7$. It is interesting to note that, depending on the desired sampling rate ($P$), the number of PA units of the proposed scalable structure can be selected from the set $\{1 \leq L \leq \lfloor \log_4 P \rfloor\}$. We can cascade an additional PA unit at the front-end of the scalable structure (provided $L < \lfloor \log_4 P \rfloor$) to scale its throughput rate by 4 times.
TABLE III

<table>
<thead>
<tr>
<th>Structures</th>
<th>Image-size</th>
<th>Multiplier</th>
<th>Adder</th>
<th>REG/LB</th>
<th>Single-port RAM</th>
<th>MUX/DMUX</th>
<th>Latency</th>
<th>ACT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(M × N)</td>
<td>in words</td>
<td>in words</td>
<td>in cycles</td>
<td>in cycle</td>
<td>words</td>
<td>in cycles</td>
<td>in cycle</td>
</tr>
<tr>
<td>Xiong et al [30]</td>
<td>(256 × 256)</td>
<td>18</td>
<td>32</td>
<td>1414</td>
<td>32768</td>
<td>3</td>
<td>22298</td>
<td>21760</td>
</tr>
<tr>
<td></td>
<td>(512 × 512)</td>
<td>18</td>
<td>32</td>
<td>2822</td>
<td>131072</td>
<td>3</td>
<td>84427</td>
<td>87040</td>
</tr>
<tr>
<td>Li et al [37]</td>
<td>(256 × 256)</td>
<td>12</td>
<td>16</td>
<td>1037</td>
<td>32768</td>
<td>17</td>
<td>22528</td>
<td>43520</td>
</tr>
<tr>
<td></td>
<td>(512 × 512)</td>
<td>12</td>
<td>16</td>
<td>2061</td>
<td>131072</td>
<td>17</td>
<td>88064</td>
<td>174080</td>
</tr>
<tr>
<td>Xiong et al [28]</td>
<td>(256 × 256)</td>
<td>28</td>
<td>48</td>
<td>2636</td>
<td>0</td>
<td>34</td>
<td>1984</td>
<td>65536</td>
</tr>
<tr>
<td>(PA+RPA)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(512 × 512)</td>
<td>28</td>
<td>48</td>
<td>5216</td>
<td>0</td>
<td>34</td>
<td>3968</td>
<td>87040</td>
</tr>
<tr>
<td>Prop. SC (P = 4)</td>
<td>(256 × 256)</td>
<td>27</td>
<td>48</td>
<td>2574</td>
<td>0</td>
<td>45</td>
<td>1800</td>
<td>16384</td>
</tr>
<tr>
<td>(PA+RPA)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(512 × 512)</td>
<td>27</td>
<td>48</td>
<td>5102</td>
<td>0</td>
<td>45</td>
<td>3592</td>
<td>65536</td>
</tr>
<tr>
<td>Proposed SC (P = 16)</td>
<td>(256 × 256)</td>
<td>99</td>
<td>166</td>
<td>2754</td>
<td>256</td>
<td>105</td>
<td>1056</td>
<td>4096</td>
</tr>
<tr>
<td>(PA+RPA)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(512 × 512)</td>
<td>99</td>
<td>166</td>
<td>5346</td>
<td>512</td>
<td>105</td>
<td>2096</td>
<td>16384</td>
</tr>
<tr>
<td>Proposed SC (P = 32)</td>
<td>(256 × 256)</td>
<td>192</td>
<td>340</td>
<td>2936</td>
<td>384</td>
<td>162</td>
<td>1016</td>
<td>2048</td>
</tr>
<tr>
<td>(PA)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(512 × 512)</td>
<td>192</td>
<td>340</td>
<td>5752</td>
<td>768</td>
<td>162</td>
<td>2000</td>
<td>8092</td>
</tr>
<tr>
<td>Proposed SC (P = 64)</td>
<td>(256 × 256)</td>
<td>387</td>
<td>688</td>
<td>3461</td>
<td>384</td>
<td>321</td>
<td>540</td>
<td>1024</td>
</tr>
<tr>
<td>(PA)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(512 × 512)</td>
<td>387</td>
<td>688</td>
<td>6277</td>
<td>768</td>
<td>321</td>
<td>1040</td>
<td>4096</td>
</tr>
<tr>
<td>Proposed SC (P = 128)</td>
<td>(256 × 256)</td>
<td>765</td>
<td>1360</td>
<td>3915</td>
<td>448</td>
<td>636</td>
<td>286</td>
<td>512</td>
</tr>
<tr>
<td>(PA)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(512 × 512)</td>
<td>765</td>
<td>1360</td>
<td>6555</td>
<td>896</td>
<td>636</td>
<td>540</td>
<td>2048</td>
</tr>
</tbody>
</table>

$$[(2.75N + NM(1 - 2^{2J+2})/3)]$$ and 
$$[(4N + NM(1 - 2^{2J+2})/3)]$$ cycles respectively. Latency of [28] is 
$$[(2.75N + 5N)]$$ cycles.

for different values of $P$. The HUE of each PE of the PA and the RPA unit is, respectively, 1 and 0.66. Here we have assumed $P$ to be power of 4. Using these values, we have estimated the HUE of the proposed scalable structure \(^1\) and found that, it has minimum HUE = 88.66 (for $L=1$). The HUE could be nearly 100% for higher values of $L$. For example, it has HUE = 99.8 for $L = 4$. Depending on the sampling rate ($P$), the scalable structure can have $L$ PA units and compute the DWT of $L$ decomposition levels (for $L = \lceil \log_4 P \rceil$). If the application requires decomposition levels higher than $L$, then the RPA unit could be cascaded to the scalable structure. For resource constrained applications, the number of PA units of the scalable structure may be decided on the available resources of the target device.

B. Performance Comparison

The hardware and time-complexity of the proposed scalable structures as well as the existing structures using 9/7 filters [9], [17], [19]–[30], [37] are listed in Table II in terms of storage space (registers, line-buffers, frame-buffers in words), arithmetic resources (multipliers and adders), cycle period, average computation time (ACT) in cycles, latency and HUE for comparison.

If we assume $(x_1, x_2, x_3, x_4)$ to be 1 for higher values of $L$ and $J$, then it can be observed from Table II that, the folded

\[^1\text{hardware utilization efficiency (HUE) is defined as HUE (\%) = \left\{\frac{[(\text{Number of PE})_{PA} \times (\text{PE}_{\text{Efficiency}}) + (\text{Number of PE})_{RPA} \times (\text{PE}_{\text{Efficiency}})]}{[(\text{Number of PE})_{PA} + (\text{Number of PE})_{RPA}]} \times 100\right\}}\]
structure of [30] and the recursive structure of [28] are the most efficient among the existing structures. Compared with [30],
the proposed design (for $L$-level DWT) has $P/3$ times less ACT and involves proportionately more arithmetic resources than
[30]. It provides $(P/4)$ times higher throughput rate and involves, respectively, $(3P/14)$ and $(2P/9)$ times more multipliers and
adders than those of [28], of $L$-level DWT. The proposed scalable structure for $J$-level DWT involves, respectively, $(6P + 9)/28$
and $((3P + 1)/3)$ times more multipliers and adders than those of [28] and provides $(P/4)$ times higher throughput.

It is important to note from Table II that, registers, line-buffers and frame-buffers contribute to the total storage space where
the size of the line-buffers is equal to the width of the image ($N$) and the size of the frame-buffer is $O(MN/4)$. Since the
images sizes are often more than $(512 \times 512)$, the storage complexity of DWT architectures is usually very high. The frame-
buffer could be implemented using single-port RAM of size $(MN/2)$ words to perform both the read and write operations
in the same cycle. Although the structure of [29] requires a small frame-buffer of size $N + 2(J - 1)$, but involves a large
line-buffers compared with the other lifting-based structures. The line-buffers required by the folded structures of [17], [19],
[20], [22]–[27], [29], [30] vary from $3N$ to $12N$. The proposed structure, however, does not involves any line-buffer and
frame-buffer. Instead of that, it involves registers and single-port RAM words of $O(N)$. If we assume $(x_3, x_4)$ to be 1 and
neglecting the term $10Nx_5$, as $x_5 = (1 - 2^{-J+L}) \times 2^{-L}$ is a small fraction for higher values of $L$, the proposed structure
involves storage space of nearly $(11N + 10P)$ and calculates $P$ DWT components in every cycle.

We have estimated the hardware and time-complexity of the proposed structure and structure of [30], [37], [28] for DWT
levels $J = 4$, input-sizes $(256 \times 256)$ and $(512 \times 512)$, and the values are listed in Table III. It can be found from Table III
that, the structure of [30], respectively, requires 1.5 times and 2 times more multipliers and adders than those of [37], and
involves 2 times less ACT than later.

For block sizes, $P = 16, 32, 64$ and $128$, the scalable structure, respectively, offers 5.3, 10.62, 21.25 and 42.5 times higher
throughput rate than [30], and involves proportionately more arithmetic resources. For input-size $(256 \times 256)$ and $(512 \times 512)$,
the structure of [30] involves, respectively, 2.2 and 2 times less registers/line-buffer words, 44.5 and 89 times more frame-buffer
words than those of the proposed structure on average for different throughput rates of implementations. For the same input
size images, the proposed structure, respectively, involves 23.6 and 45.54 times less output latency than [30] on average, for different input block-sizes. Compared with [28], the proposed structure for input block-size $P = 32, 64$ and 128, respectively, requires 6.85, 13.85 and 27.32 times more multipliers, 7.08, 14.33 and 28.33 more adders than those of [28], and involves 8, 16 and 32 times less ACT than the other. For input-size $(256 \times 256)$ and $(512 \times 512)$, the structure of [28] requires, respectively, 1.29 and 1.75 times less registers/line-buffer words and involves 4.16 and 4.36 times more output latency than the proposed structure on average for different block sizes. It is also interesting to observe from Table III that, memory space of the scalable structure do not increases proportionately with the throughput rate. It increases by 14% on average when the throughput rate is doubled.

Even for the case of $P = 4$, the proposed scalable structure is better than the structure of [28]. It involves one multiplier less than [28] and offers same throughput rate. It requires 2.3% less on-chip memory words than those of [28] for same throughput of implementations. It involves 9.3% less output latency than [28] on average, for different image sizes. Apart from these, the proposed structure has the advantage of scalability for throughput rate. The proposed structure could easily be scaled for higher throughput rate simply by adding additional PA units at the front end of this structure. Since, the on-chip memory of the proposed structure remains almost unchanged for higher values of $L$, the hardware utilization efficiency of the structure increases with $L$. For example, $L = 2$ or $P = 16$, the proposed structure offers 4 times more throughput rate than [28] at the cost of 3.53 times more multipliers, 3.45 times more adders and 2.9% (on average for different image sizes) more on-chip

### Table IV

<table>
<thead>
<tr>
<th>Designs</th>
<th>Scheme</th>
<th>Block Size $(P)$</th>
<th>Image Size $(256 \times 256)$</th>
<th>Image Size $(512 \times 512)$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Slices</td>
<td>RAM words</td>
<td>MUF (MHz)</td>
</tr>
<tr>
<td>Struct. of [28]</td>
<td>PA+RPA</td>
<td>4</td>
<td>3334</td>
<td>0</td>
</tr>
<tr>
<td>Struct. of [37]</td>
<td>Folded</td>
<td>2</td>
<td>1342</td>
<td>32768</td>
</tr>
<tr>
<td>Proposed SC</td>
<td>PA+RPA</td>
<td>4</td>
<td>3391</td>
<td>0</td>
</tr>
<tr>
<td>Proposed SC</td>
<td>PA+RPA</td>
<td>16</td>
<td>9896</td>
<td>256</td>
</tr>
<tr>
<td>Proposed SC</td>
<td>PA</td>
<td>32</td>
<td>18523</td>
<td>384</td>
</tr>
<tr>
<td>Proposed SC</td>
<td>PA</td>
<td>64</td>
<td>35943</td>
<td>384</td>
</tr>
<tr>
<td>Proposed SC</td>
<td>PA</td>
<td>128</td>
<td>75434</td>
<td>448</td>
</tr>
</tbody>
</table>

**LEGEND:** PA: Pyramid-Algorithm, RPA: Recursive Pyramid Algorithm, SC: Scalable. RAM words represent the size of single-port RAM. SDP: slice delay product. CT: Computation Time, MUF: Maximum usable frequency. SDP = (number of slices) $\times$ CT, where CT = ACT/MUF.
memory words than [28]. The HUE of the proposed structure is 94.33 which is 5.47 higher than [28]. Compared with the folded structure of [30], the proposed scalable structure with \( P = 4 \), respectively, involves 1.5 times more multipliers and adders, \( 4.5N \) more on-chip storage than those of [30], and offers 1.32 times more throughput rate of the other. However, the proposed one does not involve any frame-buffer unlike [30] which otherwise involves a frame-buffer of size \( O(MN) \). The saving in buffer space results in a substantial saving in hardware resource and output latency since the image size is very often as large as \((512 \times 512)\).

C. Simulation Result

To validate the proposed design, we have coded it in VHDL for block-size \( P = 4, 16, 32, 64 \) and 128, for \( J = 4 \) and image-size \((256 \times 256)\) and \((512 \times 512)\), and synthesized using Xilinx ISE 12.1i tools. We have also synthesized the RPA based design of [28] and folded design of [30] and [37], since these are the best among the existing designs. For block-size \((P = 4 \text{ and } 16)\), the scalable structure consists 2 PA units and one RPA unit; while it consists four PA units for all other values of \( P = 32, 64 \) and 128. We have used single-port block RAM for implementing the frame-buffer of the folded structures. All the designs are synthesized for the FPGA device 6VHX760FF1760-1, and the results obtained from the synthesis report for image-size \((256 \times 256)\) and \((512 \times 512)\) are listed in Table IV. We have estimated the parameter slice-delay-product (SDP) to measure area-time complexity of the designs in FPGA platform. SDP is defined as the product of number of slices required by each design and the computation time (CT), where \( CT = ACT / \text{(maximum usable frequency (MUF))} \).

From the synthesis report, it is found that the cycle period of the proposed designs and the structures of [28], [30] and [37] are nearly the same, as expected from the theoretical comparison (Table II). The small difference in cycle period due to MUX/DMUX components of the RPA unit and the input-buffer (IB) of the scalable structure. It is shown that, the scalable structure for block-size \( P = 4, 16, 32, 64 \) and 128, respectively, involves 1.01, 2.77, 5.10, 9.85, and 20.64 times more slices than those of [28] on average for different input image-sizes, and offers 1.08, 3.74, 7.75, 15.5 and 41.11 times more throughput rate than [28]. The structure of [30], respectively, involves 13.99 times more computation time and 13.35 times less slices than the scalable structure on average for different throughput rates and image-sizes of implementations. For image-size \((256 \times 256)\) and \((512 \times 512)\) the scalable structure, respectively, involves 22.5\% and 28.6\% less SDP than [28], 10.5\% and 23.5\% less SDP
than [37] and 30% and 15% more SDP than [30] on average for different throughput rates of implementation. It is interesting to note that, the folded structure of [28] involves 32382 and 130336 more RAM words than the scalable structure for image-size (256 x 256) and (512 x 512), respectively, on average, for different blocks sizes.

D. Comparison of Power Consumption

We have estimated the power consumption of the proposed design for input block size $P = 4, 16$ and 32, and the design of [28], [30] and [37] using Xilinx Xpower tools by implementing them in FPGA device 6VLX760FF1760-1. Xpower analyzer report for 60 MHz clock frequency is listed in Table V for comparison. From Table V we find that, the proposed structure dissipate 52% less power per output (PPO) than [28], 51.76% less PPO than [30] and 75% less PPO than [37], on average for different throughput rates and image-sizes of implementations. Since the proposed structure and the existing structures have different throughput rates, we have estimated power consumption of the proposed structure (for block size $P = 16$ and 32) and the structure of [37] at the same throughput rate as that of [28] and [30] and the values are listed in Table VI. It is shown that, at the same throughput rate, the proposed structure consumes 34 mW less power than [28], 9 mW less power than [30] and 65 mW less power than [37] on average for different input block and image sizes.

### Table V

<table>
<thead>
<tr>
<th>Designs</th>
<th>Block size $P$</th>
<th>Power (watt)</th>
<th>PPO (watt)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Input-size $(256 \times 256)$</td>
<td>Input-size $(512 \times 512)$</td>
</tr>
<tr>
<td>Structure of [28]</td>
<td>4</td>
<td>3.071</td>
<td>3.099</td>
</tr>
<tr>
<td>Structure of [30]</td>
<td>4</td>
<td>3.050</td>
<td>3.070</td>
</tr>
<tr>
<td>Structure of [37]</td>
<td>2</td>
<td>3.007</td>
<td>3.025</td>
</tr>
<tr>
<td>Proposed structure</td>
<td>4</td>
<td>3.073</td>
<td>3.079</td>
</tr>
<tr>
<td>Proposed structure</td>
<td>16</td>
<td>3.523</td>
<td>3.519</td>
</tr>
<tr>
<td>Proposed structure</td>
<td>32</td>
<td>3.874</td>
<td>3.856</td>
</tr>
</tbody>
</table>

LEGEND: PPO: Power per output.

E. Comparison of Transistor-counts

To estimate the transistor counts, we have assumed ripple-carry adders (RCA) and RCA-based multipliers of 8-bit input width for all the structures. The on-chip storage and the frame-buffer are assumed to be of D flip-flops, and single-port SRAM, respectively. The multiplier, adder, 8-bit register, 8-bit single-port SRAM word and 8-bit MUX/DMUX are taken to be 1085, 248, 128, 48 and 48 transistors, respectively [42]. Using these values we have estimated transistor-counts of proposed
TABLE VI
Comparison of Power Consumption of the Proposed Structure and the Structure of [37] at the Same Throughput Rate of [28] and [30], i.e. 4 samples processed per cycle.

<table>
<thead>
<tr>
<th>Designs</th>
<th>Power (watt) Input-size</th>
<th>Power (watt) Input-size</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(256 × 256)</td>
<td>(512 × 512)</td>
</tr>
<tr>
<td>Structure of [28]</td>
<td>3.071</td>
<td>3.099</td>
</tr>
<tr>
<td>Structure of [30]</td>
<td>3.050</td>
<td>3.070</td>
</tr>
<tr>
<td>Structure of [37]</td>
<td>3.098</td>
<td>3.134</td>
</tr>
<tr>
<td>Proposed structure (P = 16)</td>
<td>3.068</td>
<td>3.070</td>
</tr>
<tr>
<td>Proposed structure (P = 32)</td>
<td>3.036</td>
<td>3.033</td>
</tr>
</tbody>
</table>

TABLE VII
Transistor-counts of the Proposed Structure and the Structure of [28], [30] and [37].

<table>
<thead>
<tr>
<th>Designs</th>
<th>Block size P</th>
<th>TC (×10^5) Input-size</th>
<th>TC (×10^5) Input-size</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>(256 × 256)</td>
<td>(512 × 512)</td>
</tr>
<tr>
<td>Structure of [30]</td>
<td>4</td>
<td>17.81</td>
<td>66.8</td>
</tr>
<tr>
<td>Structure of [37]</td>
<td>2</td>
<td>17.23</td>
<td>65.73</td>
</tr>
<tr>
<td>Structure of [28]</td>
<td>4</td>
<td>3.81</td>
<td>7.11</td>
</tr>
<tr>
<td>Proposed structure</td>
<td>4</td>
<td>3.72</td>
<td>6.96</td>
</tr>
<tr>
<td>Proposed structure</td>
<td>16</td>
<td>4.95</td>
<td>8.65</td>
</tr>
<tr>
<td>Proposed structure</td>
<td>32</td>
<td>7.01</td>
<td>10.73</td>
</tr>
<tr>
<td>Proposed structure</td>
<td>64</td>
<td>10.67</td>
<td>14.46</td>
</tr>
<tr>
<td>Proposed structure</td>
<td>128</td>
<td>17.23</td>
<td>20.83</td>
</tr>
</tbody>
</table>

LEGEND: TC: Transistor-count.

structure and the structure of [28], [30] and [37] and listed in Table VII. It can be observed from Table VII that the folded structures of [30] and [37] involve larger number of transistors than other designs although they involve relatively less arithmetic and on-chip memory than those of [28] and the proposed structure. This is mainly due to the complexity of the frame-buffer which is almost contributes 90% to the transistor counts of the folded structure. As shown in Table VII, the proposed structure for block size \( P = 4 \) involves 2.23% less transistors than those of [28] on average for different image-sizes and offers the same throughput rate. The scalable structure for block sizes \( P = 16, 32, 64 \) and \( 128 \) involves, respectively, 1.25, 1.66, 2.41 and 3.72 times more transistors than those of [28] on average for different images-sizes and offers 4, 8, 16 and 32 times more throughput rate than other. Besides, the proposed structure has additional advantage of throughput scalability. Since the on-chip memory size of the proposed structure is almost independent of the input block size, its transistor counts does not increase much for higher throughput of implementation.

The proposed scalable structure provides higher throughput per unit hardware cost. The scalable structure can be configured
according to available device resource and target application. One can select highest possible input-block size for the proposed
design, because its HUE increases with the block size. If high throughput rate is not desirable for the target application, then
the area and speed of the scalable design could be traded for reduction of dynamic power consumption.

V. CONCLUSION

By appropriate partitioning of computation and mapping to a scalable architecture along with appropriate scheduling of the
decomposition-levels, we have derived modular and regular pipeline computing structures for lifting-based multilevel 2-D DWT.
The proposed structure is scalable for high-throughput as well as for area-constrained implementations. We have maximized
the HUE by removing all the redundancies resulting from decimation process in the wavelet filtering. The remarkable feature
of the proposed structures is that, it does not require any line-buffer of size \(O(N)\) and frame-buffer of size \(O(M/2 \times N/2)\)
unlike the existing structures for multilevel 2-D DWT computations. Instead of that, the proposed structure involves only
local registers and RAM words of \(O(N)\). The saving of line-buffer and frame-buffer achieved by the proposed design is of
substantial advantage, since the image size is very often as large as \(512 \times 512\). The proposed structure has very small latency
compared to the latency of the existing structures.

Compared with the best of the existing recursive structure [28], the proposed structure involves proportionately less arithmetic
resources and offer higher throughput rate. For block size 4, it involves 2.24\% less register/line-buffer words than those of
[28], on average, for different image sizes, and offers the same throughput rate. For image-size \((256 \times 256)\) and \((512 \times 512)\), it
involves 22.5\% and 28.6\% less SDP than [28], on average, for different throughput rates of implementation. For the same image
size, the scalable structure involves 1.25, 1.66, 2.41 and 3.72 times more transistors than those of [28] and offers 4, 8, 16 and 32
times more throughput than the other. The best of the folded structure requires 13.35 times less slices and involves 13.99 times
more ACT compared with the proposed scalable structure on average for different throughput rates of implementation. The
proposed structure, however, involves 0.5\% of the RAM words of the folded structure, on average, for image size \((512 \times 512)\).
It involves 4 times less transistors than [30] and offers 16.42 more throughput rate than other, on average, for different input-
block and image sizes. The proposed scalable structure dissipate 52\% less PPO than [28] and 51.76\% less PPO than [30], on
average, for different input-block and image sizes. At the same throughput rate, the scalable structure consumes 34 mW less
power than [28] and 9 mW less power than [30], on average, for different input block sizes and image sizes. It is interesting to note that, the throughput rate of the proposed scalable structure can be scaled and the HUE of the structure can be maximized for higher input block size.

REFERENCES


Pramod Kumar Meher (SM’03) received the B.Sc. and M.Sc. degrees in physics and the Ph.D. in science from Sambalpur University, Sambalpur, India, in 1976, 1978, and 1996, respectively. He has a wide scientific and technical background covering physics, electronics, and computer engineering. Currently, he is a Senior Scientist with the Institute for Infocomm Research, Singapore. Prior to this assignment he was a visiting faculty with the School of Computer Engineering, Nanyang Technological University, Singapore. Previously, he was a Professor at Utkal University, Bhubaneswar, India, from 1997 to 2002, a Reader in electronics with Berhampur University, Berhampur, India, from 1993 to 1997, and a Lecturer in physics with various Government Colleges in India from 1981 to 1993. His research interest includes design of dedicated and reconfigurable architectures for computation-intensive algorithms pertaining to signal processing, image processing, communication, and intelligent computing. He has published more than 100 technical papers in various reputed journals and conference proceedings.

He is a Fellow of the Institution of Engineering and Technology (IETE), India and a Fellow of the Institution of Engineering and Technology (IET), UK. He is currently serving as Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS-II: EXPRESS BRIEFS, IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, and Journal of Circuits, Systems, and Signal Processing. He was the recipient of the Sanampa Chandrasekhar Award for excellence in research in engineering and technology for the year 1999.

Dr. Meher is a life time member of The Institution of Electronics and Telecommunication Engineering, New Delhi, India.

Basant K Mohanty (M06) received B.Sc and M.Sc degree (both with first-class honors) in Physics from Sambalpur University, Orissa, in 1987 and 1989, respectively. Received Ph.D degree in the field of VLSI for Digital Signal Processing from Berhampur University, Orissa in 2000.

In 1992 he was selected by OPSC (Orissa Public Service Commission) and joined as faculty member in the Department of Physics, SKCG College Paralakhemundi, Orissa. In 2001 he joined as Lecturer in EEE Department, BITS Pilani, Rajasthan. Then he joined as Assistant Professor in the Department of ECE, Mody Institute of Science and Technology (Deemed University), Rajasthan. In 2003 he has joined as Assistant Professor in the Department of ECE, Mody Institute of Science and Technology (Deemed University), Rajasthan. In 2005 he has joined the Jaypee University of Engineering and Technology, Guna, Madhya Pradesh as Assistant Professor. He was promoted to Associate Professor in year 2005 and Professor in year 2007. His research interest includes design and implementation of re-configurable VLSI architectures for resource-constrained digital signal processing applications. He has published nearly 25 technical papers. His name has appeared in Marquis Who's Who in the World in 1999. Currently he serves as the reviewers of IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS-II: EXPRESS BRIEFS, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, and IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS.

Dr Mohanty is a life time member of The Institution of Electronics and Telecommunication Engineering, New Delhi, India.