EURASIP Journal on Image and Video Processing
a SpringerOpen Journal

RESEARCH Open Access

High-performance hardware architectures for multi-level lifting-based discrete wavelet transform

Anand D Darji[1*], Shailendra Singh Kushwah[2], Shabbir N Merchant[1] and Arun N Chandorkar[1]

Abstract

In this paper, three hardware efficient architectures to perform multi-level 2-D discrete wavelet transform (DWT) using lifting (5, 3) and (9, 7) filters are presented. They are classified as folded multi-level architecture (FMA), pipelined multi-level architecture (PMA), and recursive multi-level architecture (RMA). Efficient FMA is proposed using dual-input Z-scan block (*B1*) with 100% hardware utilization efficiency (HUE). Modular PMA is proposed with the help of block (*B1*) and dual-input raster scan block (*B2*) with 60% to 75% HUE. Block *B1* and *B2* are micro-pipelined to achieve critical path as single adder and single multiplier for lifting (5, 3) and (9, 7) filters, respectively. The clock gating technique is used in PMA to save power and area. Hardware-efficient RMA is proposed with the help of block (*B1*) and single-input recursive block (*B3*). Block (*B3*) uses only single processing element to compute both predict and update; thus, 50% multipliers and adders are saved. Dual-input per clock cycle minimizes total frame computing cycles, latency, and on-chip line buffers. PMA for five-level 2-D wavelet decomposition is synthesized using Xilinx ISE 10.1 for Virtex-5 XC5VLX110T field-programmable gate array (FPGA) target device (Xilinx, Inc., San Jose, CA, USA). The proposed PMA is very much efficient in terms of operating frequency due to pipelining. Moreover, this approach reduces and totals computing cycles significantly as compared to the existing multi-level architectures. RMA for three-level 2-D wavelet decomposition is synthesized using Xilinx ISE 10.1 for Virtex-4 VFX100 FPGA target device.

Keywords: Clock gating; DWT; Dual-scan architecture; Folding; FPGA; Lifting

1 Introduction

In recent years, multi-level two-dimensional discrete wavelet transform (2-D DWT) is used in many applications, such as image and video compression (JPEG 2000 and MPEG-4), implantable neuroprosthetics, biometrics, image processing, and signal analysis. due to good energy compaction in higher-level DWT coefficients. To meet application constraint such as speed, power, and area, there is a huge demand of hardware-efficient VLSI architectures in recent years. DWT provides high compression ratio without any blocking artifact that deprives reconstructed image of desired smoothness and continuity. However, implementation of convolution-based DWT has many practical obstacles, such as higher computational complexity and more memory requirement. Therefore, Swelden et al. [1] proposed lifting wavelet, which is also known as second-generation wavelet. DWT can be implemented using convolution scheme as well as lifting scheme. The computational complexity and memory requirement of lifting scheme is very less as compared to convolution. Several architectures have been proposed to perform lifting-based DWT, which differ in terms of numbers of multipliers, adders, register, line buffers requirement, and scanning scheme adopted. 2-D DWT can be computed by applying 1-D DWT row-wise, which produces low-frequency (L) and high-frequency (H) sub-bands and then process these sub-bands column-wise to compute one approximate (LL) and three detail (LH, HL, HH) coefficients.

Jou et al. have proposed architecture with straightforward implementation of the lifting steps and therefore this

*Correspondence: anand@ee.iitb.ac.in
[1]Department of Electrical Engineering, Indian Institute of Technology Bombay, Powai, Mumbai 400076, India
Full list of author information is available at the end of the article

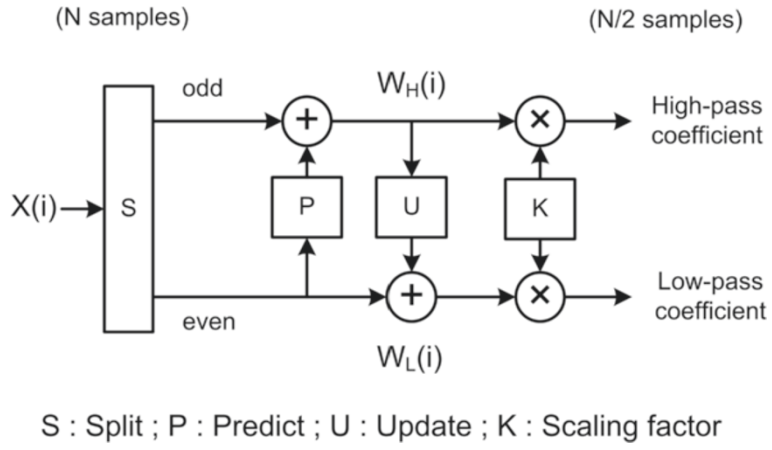

Figure 1 **Lifting scheme [11].**

architecture has long critical path [2]. An efficient pipeline architecture having critical path of only one multiplier has been proposed by merging predict and update stages by Wu and Lin [3]. Lai et al. have implemented dual-scan 2-D DWT design based on the algorithm proposed by Wu et al. with critical path delay of one multiplier and throughput of 2-input/2-output at the cost of more pipeline registers [4]. Dual-scan architecture with one multiplier as a critical path has also been proposed by Zhang et al. at the cost of complex control path [5]. Hsia et al. [6] have proposed a memory-efficient dual-scan 2-D lifting DWT architecture with temporal buffer $4N$ and critical path of two multipliers and four adders. Recently, a dual-scan parallel flipping architecture is introduced with the critical path of one multiplier, less pipeline registers, and simple control path [7].

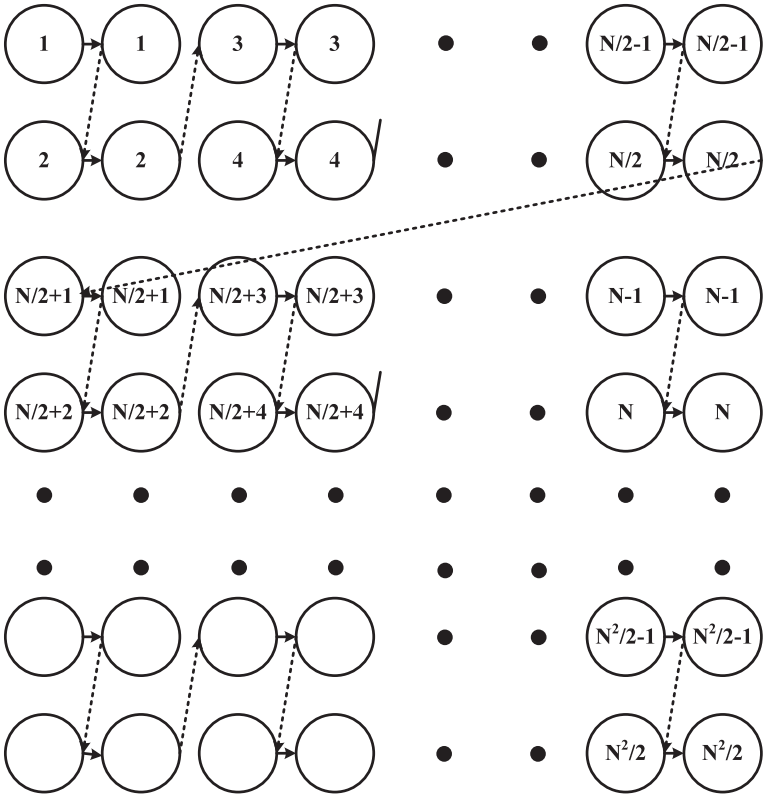

Figure 2 **Dual-input Z-scanning technique.**

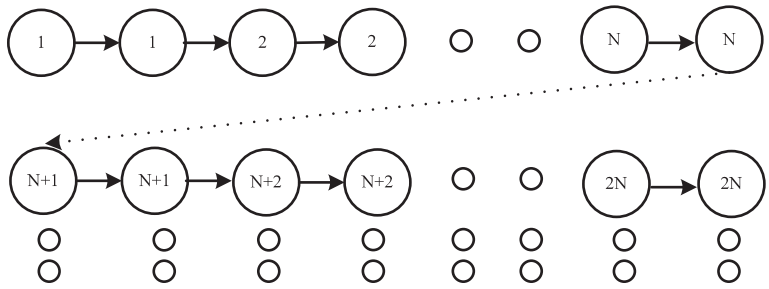

Figure 3 Dual-input raster scanning technique.

Several lifting-based 2-D DWT multi-level architectures have been suggested for efficient VLSI implementation [3,8-15] that take into consideration various aspects like memory, power, and speed. These architectures can be classified as folded architectures [8], parallel architectures [14], and recursive architectures [16]. Andra et al. [8] proposed simple folded architecture to perform several stages of 2-D DWT with the help of memory. Systolic architecture is proposed by Huang et al. [9] for a DWT filter with finite length. Multi-level 2-D DWT decomposition is implemented by recursive architecture (RA) [10,16], but these approaches demand large amount of frame buffers to store the intermediate LL output and also require complex control path. Wu and Lin [3] proposed folded scheme, where multi-level DWT computation is performed level by level, with memory and single processing element. Unlike RA, folded architecture uses simple control circuitry, and it has 100% hardware utilization efficiency (HUE). Folded architectures consist of memory and a pair of 1-D DWT modules, a row processing unit (RPU) and a column processing unit (CPU). Mohanty and Meher [12] proposed a multi-level architecture for high throughput with more number of adders and multipliers. Xiong et al. [13] proposed two line-based high-speed architectures by employing parallel and pipelining techniques, which can perform j-level decomposition for $N \times N$ image in approximate $2(1-4^{-j})N^2/3$ and $(1-4^{-j})N^2/3$ clock cycles. Hsia et al. [17] have proposed a symmetric mask-based scheme to compute 2-D integer lifting DWT, where the separate mask is used for each sub-band. Mask-based algorithms do not require temporal buffers, but they are not suitable for area efficient implementation due to a large number of adder and multiplier requirement. A memory efficient architecture is proposed by Lai et al. [4] with low latency. Al-Sulaifanie et al. [18] designed an architecture, which is independent of input image size with moderate speed and HUE. Recently, Aziz and Pham [15] have proposed parallel architecture for lifting (5, 3) multi-level 2-D DWT with a single processing unit to calculate both predict and update values. But this architecture requires $4N$ line buffers for single-level decomposition and has less HUE due to the wastage of the alternate clock cycle. Memory management is key to design multi-level 2-D DWT architectures. The lifting-based multi-level 2-D DWT architecture is suggested using overlapped stripe-based scanning method [19].

Three types of memory buffer are generally used in any 2-D DWT architecture, i.e., frame, transposition, and temporal memory. Frame memory is required to

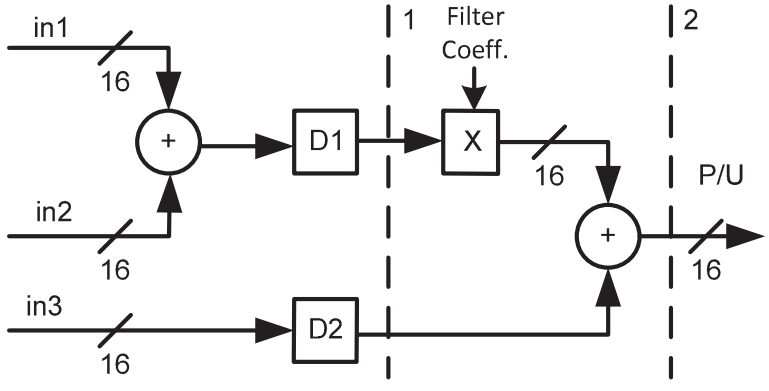

Figure 4 Generic architecture for predict/update module.

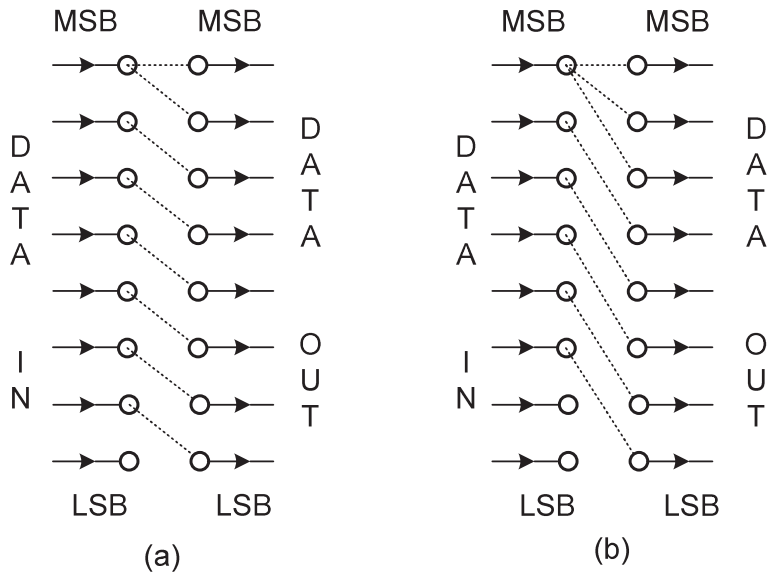

Figure 5 Hardwired scaling unit (HSU). (a) $\alpha = 1/2$. **(b)** $\alpha = 1/4$.

store intermediate LL coefficients, transposition memory is mainly required for storing row processor output, and temporal memory is required to store partial results during column processing. Temporal and transposition memories are on-chip, while the frame memory is on or off-chip, depending upon the architecture. Size of transposition memory is limited by the method adopted to scan external memory. Different scanning techniques have been proposed, such as line-based, block-based, and stripe-based [14,19-21].

This paper is organized as follows. Section 2 provides a brief overview of lifting scheme. In Section 3, design of three efficient multi-level architectures and their modules are discussed. Performance comparison, field-programmable gate array (FPGA) implementation and timing analysis are described in Section 4. Conclusions are presented in Section 5.

2 Lifting scheme

The lifting scheme is a hardware-efficient technique to perform DWT. Lifting scheme entirely relies on spatial domain and it has many advantages as compared to the convolution method of calculating DWT such as in-place computation, symmetric forward and inverse transforms, and perfect reconstruction. The basic principle of lifting scheme is to factorize the polyphase matrix of a wavelet filter into a sequence of alternating upper and lower triangular matrices and a diagonal matrix and convert the filter implementation into banded matrices multiplications [1] as shown by (1) and (2). Where, $h_e(z)$ and $g_e(z)$ ($h_o(z)$ and $g_o(z)$) represent the even parts (odd part) of the low-pass and high-pass filters, respectively. $s_i(z)$ and $t_i(z)$ are denoted as predict lifting and update lifting polynomials. K and $1/K$ are scale normalization factors. Factorization of classical wavelet filter into lifting steps reduces the computational complexity up to 50%.

$$P(z) = \begin{bmatrix} h_e(z)h_o(z) \\ g_e(z)g_o(z) \end{bmatrix} \tag{1}$$

$$P(z) = \prod_{i=1}^{m} \begin{bmatrix} 1 & s_i(z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ t_i(z) & 1 \end{bmatrix} \begin{bmatrix} K & 0 \\ 0 & 1/K \end{bmatrix} \tag{2}$$

Every reconstructible filter bank can be expressed in terms of lifting steps in general. Figure 1 shows a block diagram of a forward lifting scheme to compute 1-D DWT. It is composed of three stages: split, predict, and update. In split stage, input sequence is divided into two subsets, an even indexed sequence and an odd indexed sequence. During the second stage, the even indexed sequence is used to predict odd sequence. Two consecutive even and one odd indexed input sequences $X(i)$ are used to calculate high-pass coefficient $W_{\mathrm{H}}(i)$ (detail coefficients), as given by (3). Two consecutive high-pass coefficients (present and previous) and one even indexed

Table 1 Data flow of predict/update module

Clock	Input	D1	D2	P/U
1	$X_{1,1} : X_{1,3} : X_{1,2}$			
2	$X_{1,3} : X_{1,5} : X_{1,4}$	$X_{1,1} + X_{1,3}$	$X_{1,2}$	
3	$X_{1,5} : X_{1,7} : X_{1,6}$	$X_{1,3} + X_{1,5}$	$X_{1,4}$	$P/U_{1,1}$
4	$X_{1,7} : X_{1,9} : X_{1,8}$	$X_{1,5} + X_{1,7}$	$X_{1,6}$	$P/U_{1,2}$

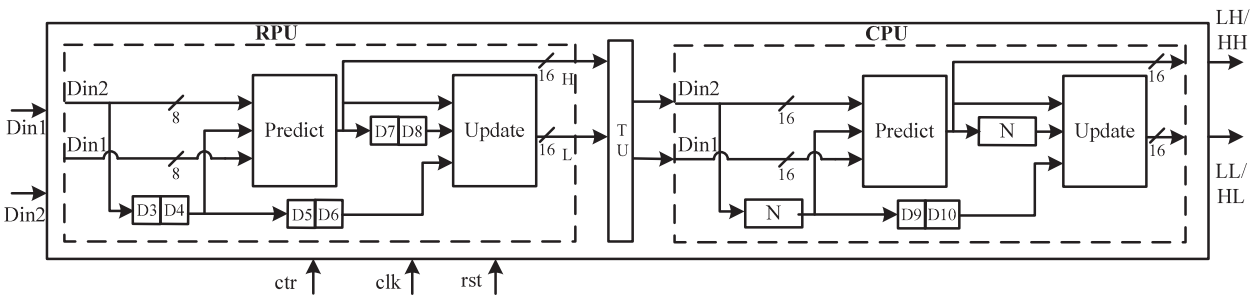

Figure 6 Dual-input Z-scan block (*B1*) for lifting (5, 3) filter.

input sequence are used to calculate low-pass coefficients $W_{\mathrm{L}}(i)$ (approximate coefficients), as given by (4).

$$W_{\mathrm{H}}(i) = X(2i-1) + \alpha[X(2i) + X(2i-2)] \tag{3}$$

$$W_{\mathrm{L}}(i) = X(2i-2) + \beta\,[W_{\mathrm{H}}(i) + W_{\mathrm{H}}(i-1)] \tag{4}$$

3 Proposed architectures

In this section, the proposed dual-input Z-scan architecture (*B1*), dual-input raster scan module (*B2*), and single-input block *B3* are discussed. All these blocks are designed with consideration of lifting wavelet to perform single-level 2-D DWT. Blocks *B1* and *B2* are designed to process two inputs and generate two outputs at every clock cycle. The total clock cycles required to perform one-level decomposition of a $N \times N$ image is $N^2/2$ without considering latency. Then, three novel architectures for multi-level 2-D DWT are presented, i.e., folded multi-level architecture (FMA), pipelined multi-level architecture (PMA), and recursive multi-level architecture (RMA). The FMA is composed of a block (*B1*) and $N^2/4$ off-chip memory, where as the PMA is composed of a block (*B1*) to perform first-level decomposition and block (*B2*) for higher levels of decompositions. The RMA is composed of block (*B1*) and block (*B3*) to compute first-level and higher-level decomposition.

Table 2 Data flow of block (*B1*): image size 256 × 256

Clk	Input	1-D DWT output	2-D DWT output
1	$X_{1,1} : X_{1,2}$		
2	$X_{2,1} : X_{2,2}$		
3	$X_{1,3} : X_{1,4}$		
4	$X_{2,3} : X_{2,4}$		
5	$X_{1,5} : X_{1,6}$		
6	$X_{2,3} : X_{2,4}$	$L_{1,1} : H_{1,2}$	
7	$X_{1,5} : X_{1,6}$	$L_{2,1} : H_{2,2}$	
8	$X_{2,5} : X_{2,6}$	$L_{1,2} : H_{1,3}$	
9	$X_{1,7} : X_{1,8}$	$L_{2,2} : H_{2,3}$	
10	$X_{2,7} : X_{2,8}$	$L_{1,3} : H_{1,4}$	
...	...	...	
257	$X_{3,1} : X_{3,2}$	$L_{1,254} : H_{1,255}$	
258	$X_{4,1} : X_{4,2}$	$L_{2,254} : H_{2,255}$	
...	...	...	
268	$X_{4,11} : X_{4,12}$	$L_{3,3} : H_{3,4}$	$LL_{1,1} : LH_{1,2}$
269	$X_{3,13} : X_{3,14}$	$L_{4,3} : H_{4,4}$	$HL_{2,1} : HH_{2,2}$
269	$X_{4,13} : X_{4,14}$	$L_{3,4} : H_{3,5}$	$LL_{1,2} : LH_{1,3}$

Three different architectures are suggested based on different VLSI optimization criteria such as area, power, speed, throughput, and memory. FMA is a straight forward design which requires simple control but has lower throughput and it demands $N^2/4$ memory. PMA is designed to satisfy the need of high throughput at the cost of area, whereas RMA gives moderate throughput and utilizes moderate area. RMA gives throughput higher than FMA but lower than PMA. Therefore, for the application which demands high throughput, PMA can be deployed. In applications with no memory constraint but need simple control, we can go for FMA design. RMA can be deployed where we have constraints of area and power but need to achieve high throughput.

3.1 Scanning schemes

Pixels are accessed by architecture *B1* in dual-input Z-scan manner as shown in Figure 2. In this method, data scanning is optimized for simultaneous operation on two rows which produces coefficients required for vertical filtering such that the latency involved in calculating 2-D DWT coefficients with boundary treatment is decreased and become independent of image size. Two pixel values are read from the first row in a single clock and processed by 1-D DWT architecture in the next clock. During the same clock, two values are read from the next row. Architecture *B2* accesses pixels in dual-input raster scan manner as shown in Figure 3.

3.2 Predict/update module

The main processing element in the blocks *B1*, *B2*, and *B3* is predict/update. Both RPU and CPU consist of pipelined

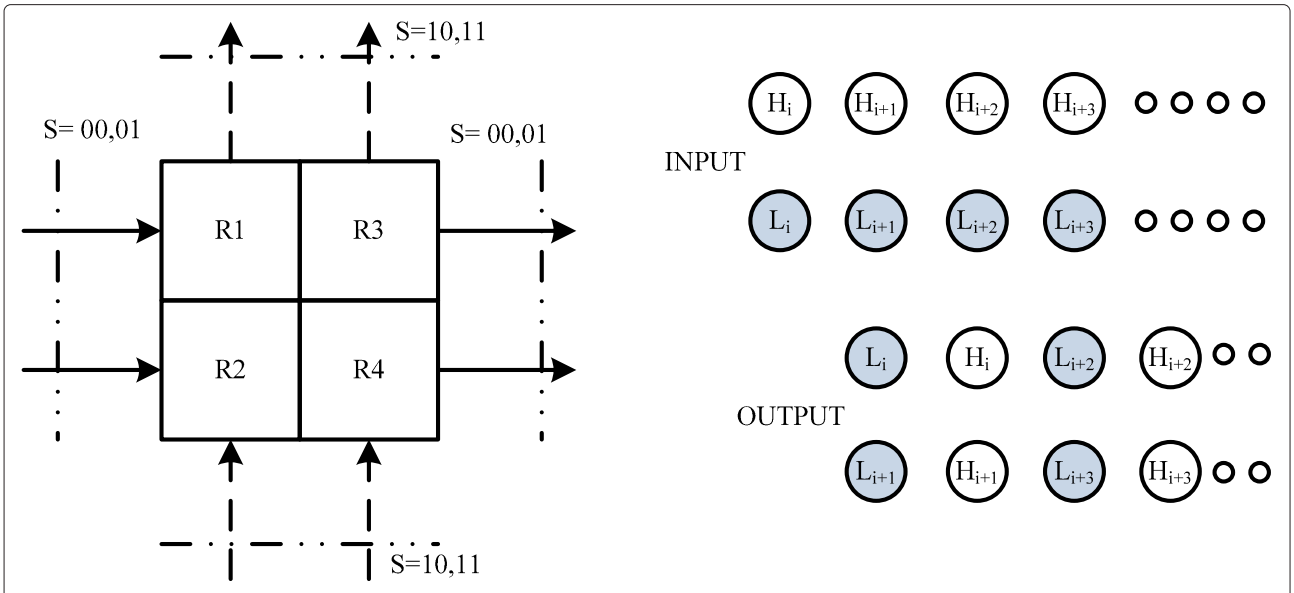

Figure 7 Transpose unit [4].

predict and update blocks to reduce the critical path. Both the predict and update have the same data path, so it can be designed as one generic processing element, as shown in Figure 4. 1-D DWT using lifting (5, 3) filter requires multiplication of two filter coefficients, α and β, whose values are $-1/2$ and $1/4$, respectively. Coefficient multiplication is designed using logical left shift operation by one and two bits, respectively. We have used simple and power efficient hardwired scaling unit (HSU), in which shifters are replaced by hardwired connections. This operation optimizes the speed without compromising power and area. The HSU to perform divide by two and divide by four operations on signed input is shown in Figure 5. In case of (9, 7) lifting, multipliers are used to reduce quantization noise generated due to fractional value of filter coefficients.

Both predict and update consist of two adders, two delay registers and one multiplier or HSU. Predict/update module takes three inputs and produces two outputs per clock cycle. Entire predict/update operation is divided into two stages. In the first pipeline stage, *in*1 and *in*2 are added and stored in *D*1 register and at the same time *in*3 is stored in *D*2. In the second stage, data of *D*2 and shifted or multiplied value of *D*1 are added to compute predict value as shown in Table 1. *D*1 and *D*2 registers are used to pipeline the predict/update module. Thus, predict/update

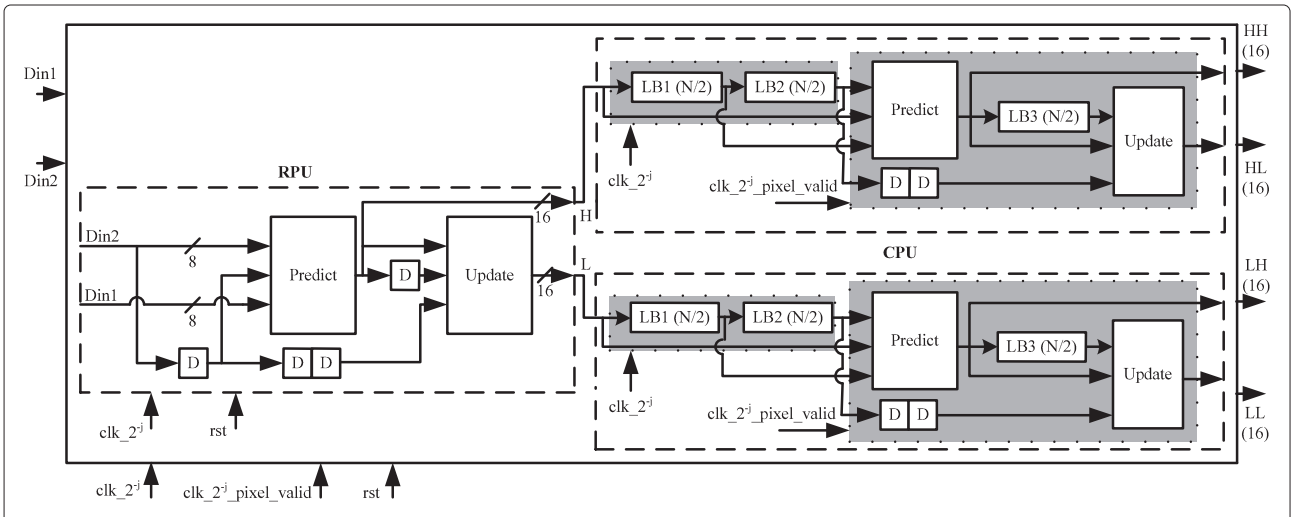

Figure 8 Dual-input raster scan block (*B2*).

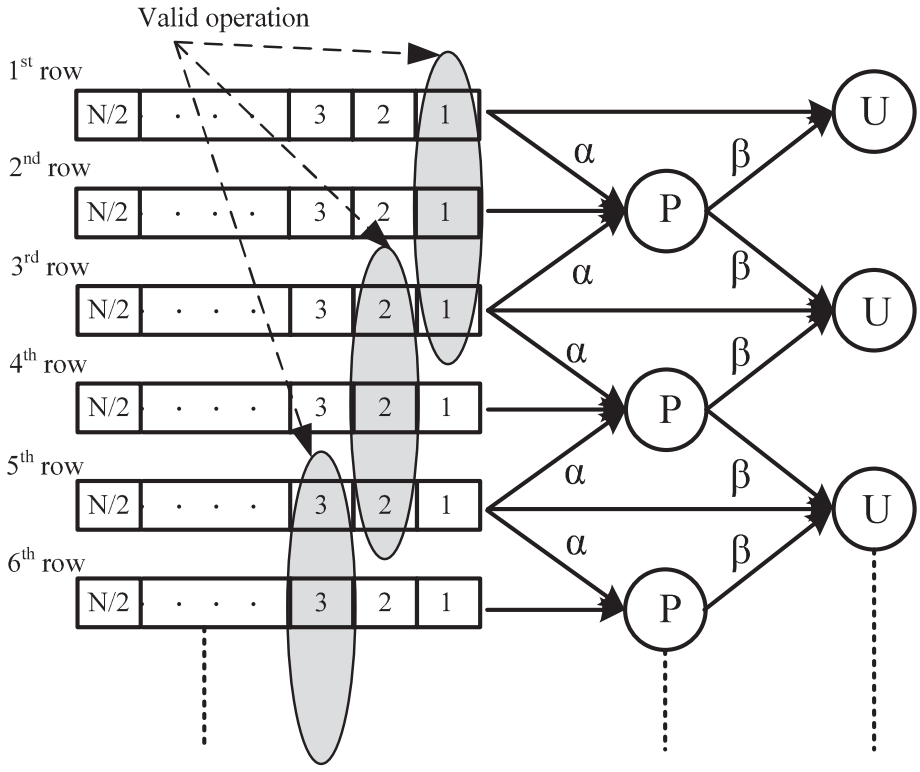

Figure 9 Data flow graph of CPU of block (*B2*) for lifting (5, 3) filter.

module takes two clock cycles to compute predict/update coefficients.

3.3 Dual-input Z-scan block (*B1*)

The proposed Z-scan-based block (*B1*) for lifting (5, 3) is composed of RPU, transpose unit (TU), and CPU as shown in Figure 6. Each RPU and CPU uses two processor elements for predict and update operations. This block uses $2N$ line buffers and 8 registers for temporary storage. Structure of CPU is identical to RPU, except temporal buffers. Two buffers of length N are used in the CPU. These buffers are initialized to zero for zero boundary extension technique for boundary treatment. Pixels are accessed in Z-scan manner as shown in Figure 2 and processed by RPU to produce high-pass (H) and low-pass (L) 1-D coefficients in a single clock. The 1-D coefficients are then given to TU for transposition, so that the CPU can process them in order to produce 2-D DWT coefficient. Block (*B1*) takes two inputs and produces two 2-D DWT coefficients (LL,LH) and (HL,HH) at alternate clock cycle. Predict module of RPU uses three input *Din*1, *Din*2 and a delayed version of *Din*2. Delay registers *D*5 and *D*6 are

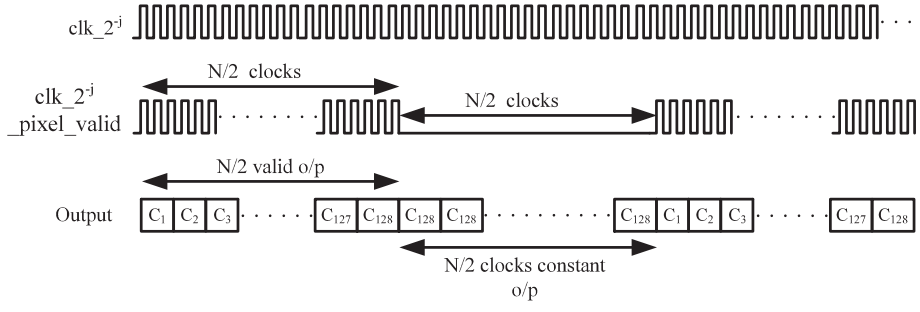

Figure 10 Block (*B2*) output control (image size 256 × 256).

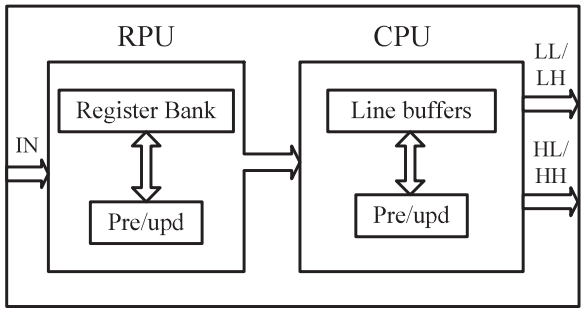

Figure 11 Block diagram of recursive block *B3*.

used to synchronize the predict output and *Din*2 (input signal of update module) because predict operation consumes two clocks as shown in Table 1. So, the RPU and CPU takes 6 and $N + 4$ clock cycle, respectively, and TU consumes two clocks; hence, latency of block (*B1*) to compute 2-D DWT is $N + 12$ clock cycle as shown in Table 2. The latency is reduced to only 12 clock cycles if the output is considered with boundary treatment. The advantage of Z-scan is 100% HUE and the simple control path as compared to [10] and [13] to achieve 100% HUE. The simple control path of architecture further reduces the power and area requirement.

CPU accepts column-wise data, which is managed through efficient TU architecture described in [4]. TU exploits the decimation characteristic of lifting scheme. TU is composed of four registers and two 4×2 multiplexers. Input/output sequences and block diagram of TU are shown in Figure 7, where S is select input of multiplexer. Block (*B1*) for lifting (9, 7) filter can be extended by using two instances of the predict and update modules with $4N$ memory buffers.

3.4 Dual-input raster scan block (*B2*)

A novel architecture block (*B2*) with dual-input raster scan is proposed for lifting (5, 3) DWT operation as shown in Figure 8. Block (*B2*) is composed of one RPU and two CPUs. Block (*B2*) takes two inputs at the rising edge of clk_2^{-j} in raster scan manner and produces four outputs at every clock cycle. TU is not required because of the two dedicated 1-D CPUs. RPU of block (*B2*) works in the same manner as of block (*B1*) and produces two 1-D DWT coefficient (L and H). The 1-D coefficients are then given to two independent CPU blocks to produce 2-D DWT coefficients as shown in Figure 8. CPU constitutes of three line buffers *LB*1, *LB*2, and *LB*3 of length $N/2$ to process $N \times N$ image. Predict operation of the CPU is managed by two previous rows of 1-D coefficient stored in *LB*1 and *LB*2 along with one current on-line coefficient as third-row input from RPU. Past predict values are stored in *LB*3 to compute update operation. Therefore, line buffers used for one CPU is $1.5 \times N$, and since block (*B2*) uses two instances of CPU, a total of $3 \times N$ line buffers are required. Block (*B2*) for *j*th-level calculation uses two clock signals, clk_2^{-j} and clk_2^{-j}_pixel_valid. RPU of block (*B2*) uses only clk_2^{-j}, while CPU uses both the clock signals. RPU remains busy in each clock to process two incoming pixels/clock; this justifies 100% HUE for RPU. The CPU uses three rows of 1-D coefficient to produce one row of valid 2-D coefficients as shown in Figure 9. But the RPU of block (*B2*) produces two rows of 1-D high and low coefficients, respectively, so the CPU has to wait for $N/2$ clock to save required 1-D coefficient in line buffers

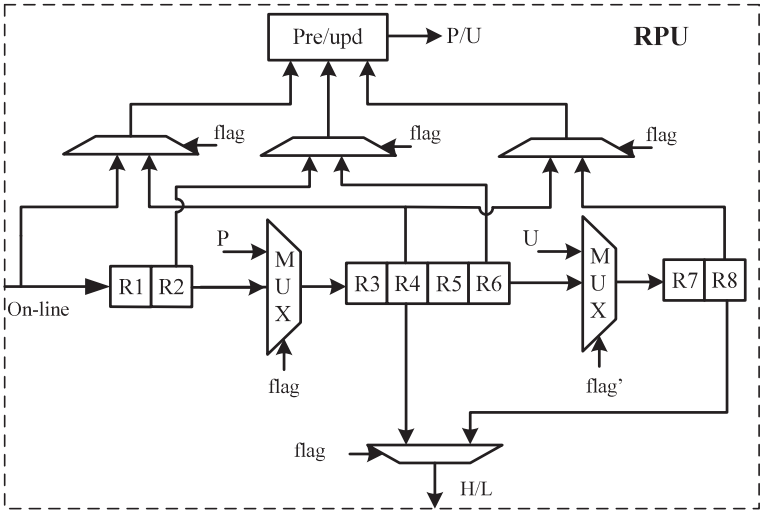

Figure 12 RPU of recursive block *B3* for lifting (5, 3) filter.

Table 3 Mapping of register bank and predict/update module

Flag	Operation	Predict/update module		
		in1	in2	in3
0	Predict	On-line	R4	R2
1	Update	R4	R8	R6

*LB*1 and *LB*2 to start predict operation. Due to this fact, output from the CPU of block (*B2*) is valid for $N/2$ clock cycles and remains constant for next $N/2$ clock cycles as shown in Figure 10. Gated clock clk_2^{-j}_pixel_valid is used to save an extra $N/2$ buffer, otherwise is required as *LB*3, when the CPU does not produce valid coefficients.

3.5 Single-input recursive block (*B3*)

Single-input recursive block (*B3*) consists of RPU and CPU is shown in Figure 11. RPU and CPU use only single processing unit (pre/upd) to compute predict and update alternately. RPU is shown in Figure 12; it consists of a register bank with eight registers and a predict/update module. Data of register bank are right shifted at every clock cycle. The predict/update receives inputs from the register bank and performs predict and update operations alternately, i.e., for two clock cycles predict and for the next two clock cycles update. Mapping of the register bank with inputs of predict/update module is shown in Table 3. Flag input is used to determine predict or update operation. The flag remains '0' for two clock cycles and '1' for next two clock cycles. Input pixel (on-line) is given to register *R*1, and output is taken from register *R*4 and *R*8. The output of predict/update module is written in register *R*3

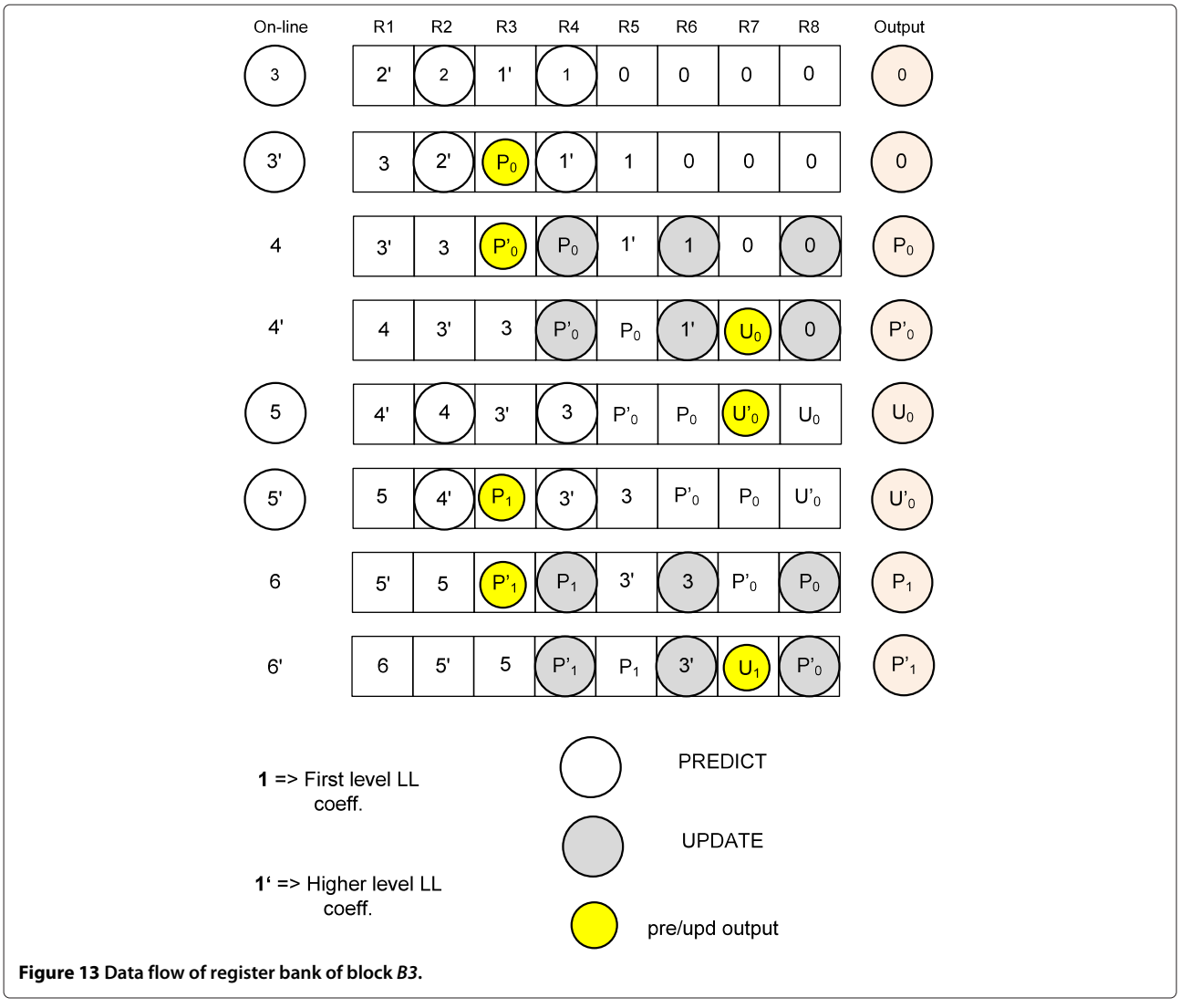

Figure 13 Data flow of register bank of block *B3*.

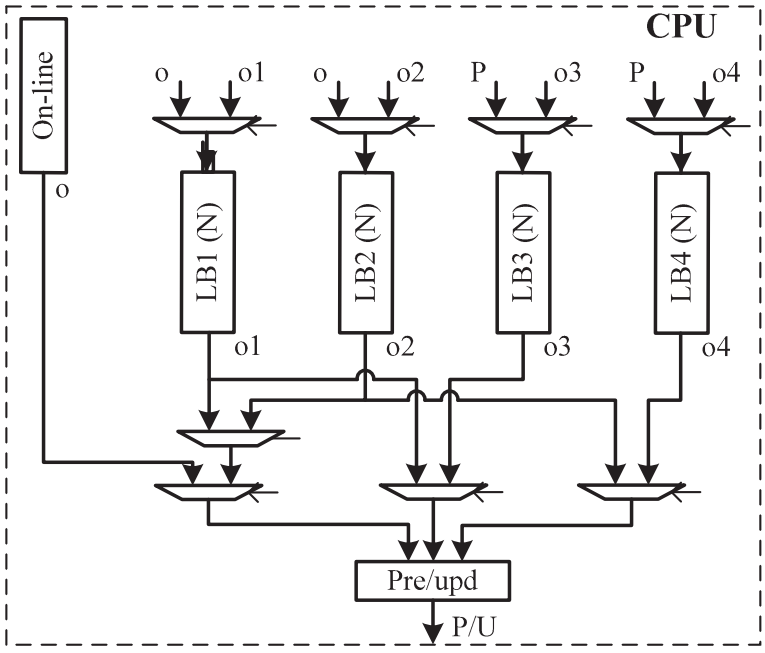

Figure 14 CPU of recursive block *B3* for lifting (5, 3) filter.

and *R*7. Complete data flow of register bank of RPU is shown Figure 13.

CPU consists of line buffers and a predict/update module as shown in Figure 14 and works in a similar manner to the RPU. The CPU consists of four line buffers of size *N* to store 1-D coefficients and intermediate predict values. Data flow of memory bank is similar to register bank. 1-D coefficients for first and second rows of input (1st and 2nd) are stored in line buffer LB1 and LB2, respectively. Then, when the first 1-D coefficient for the

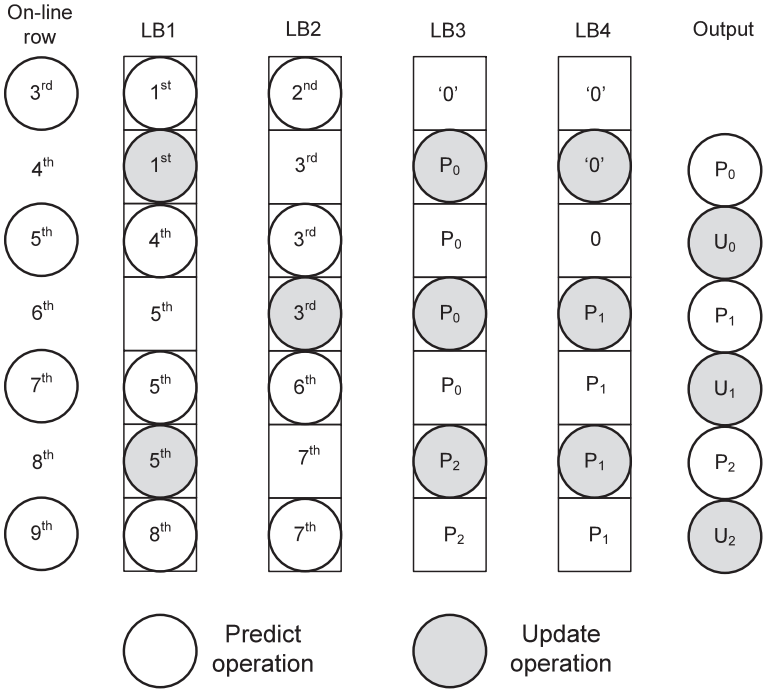

Figure 15 Data flow for line buffers LB1, LB2, LB3, and LB4 of block *B3*.

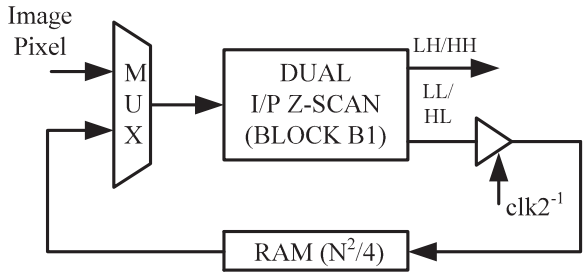

Figure 16 Folded multi-level architecture (FMA).

third row (3rd) comes, the predict/update module starts operation on third row (on-line) with data of LB1 and LB2 as shown in Figure 15. Predict/update module computes predict value and stores it in the first location of LB3, at the same time CPU stores third row of 1-D coefficient in LB2, whose current content value in 2nd row is no longer needed. Then onward, when CPU stores 1-D coefficient for third row in LB2, predict value is stored in the LB3, moving from left to right, until all columns are processed. When the 4th row is on-line, predict/update module computes update value from output of LB1, LB3, and LB4, at the same time the 4th row of 1-D coefficient is stored in LB1, whose current content value 1st row is no longer needed. The process continues until all 2-D coefficients are generated for the input image. The proposed memory organization is extremely efficient such that four line buffers are used to store all intermediate 1-D coefficients and predict values. RMA for lifting (9, 7) can be designed using two instances of RPU and CPU in block (*B3*).

Table 4 Function of control pin in PMA

Control (ctr)	Level of decomposition
000	1
001	2
010	3
011	4
100	5

3.6 Multi-level design

3.6.1 Folded multi-level architecture

Block diagram of the proposed folded multi-level architecture is shown in Figure 16. It is composed of dual-input Z-scan block (*B1*), multiplexer and memory module (RAM). The multiplexer either selects input data from the image for the first-level decomposition or from RAM for higher-level decompositions. Block (*B1*) takes two inputs and produces two outputs. Since LL coefficient is produced alternately, the output of block (*B1*) is saved in RAM at half clock rate. Complete data flow is shown in Table 2. RAM of length $N^2/4$ is used to store *j*th-level LL coefficients to obtain *j*th+1 level of decomposition. In this scheme, next level decomposition can start only after completion of previous level decomposition, known as level-by-level decomposition. This procedure is repeated until the desired level of decomposition is obtained. FMA for lifting (9, 7) filter is implemented by replicating predict and update modules in block (*B1*).

3.6.2 Pipelined multi-level architecture (PMA)

The proposed PMA to perform 5-level lifting (5,3)-based 2-D DWT is shown in Figure 17. The proposed architecture is composed of one instance of dual-input Z-scan

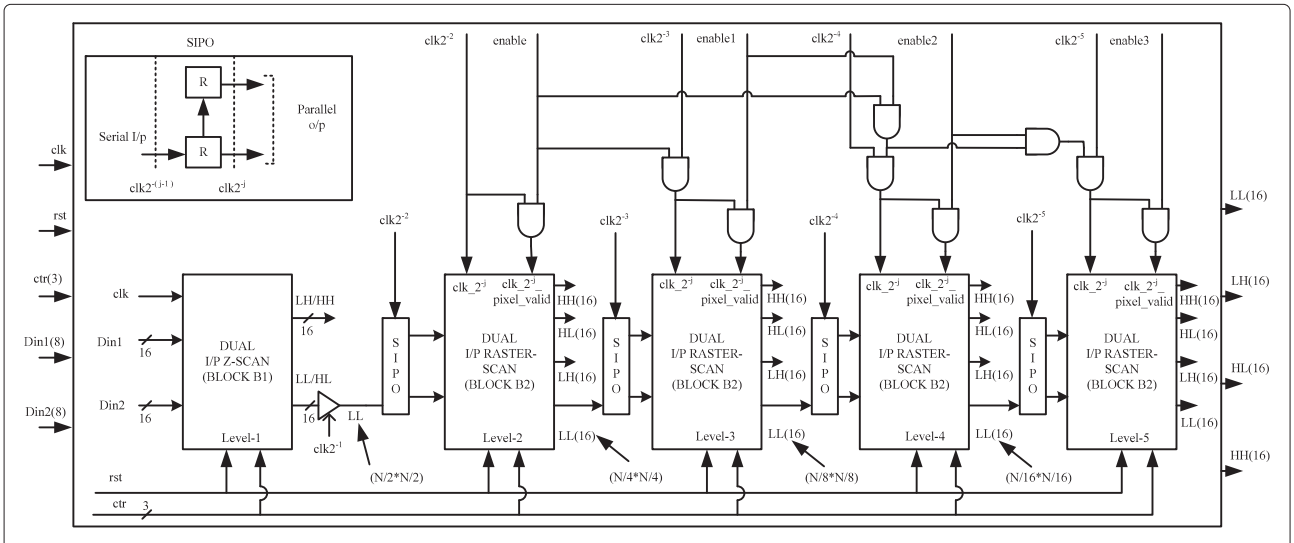

Figure 17 PMA block diagram for five-level DWT.

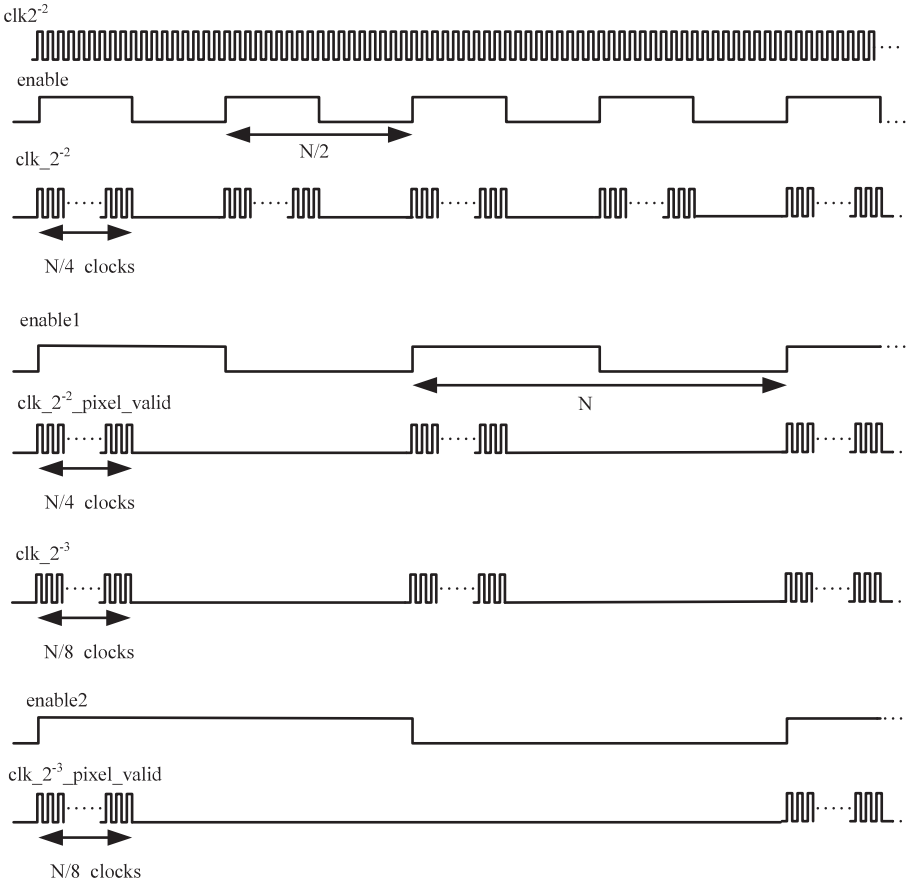

Figure 18 Timing diagram of clock, gated clocks, and enable signals for lifting (5, 3)-based PMA. *N* is the input image width/height.

block (*B1*) and four instances of dual-input raster scan block (*B2*). Here, the blocks (*B1*) and (*B2*) perform first-level and higher-level 2-D DWT decomposition, respectively. In the proposed PMA, all 2-D DWT modules operate parallel with dual input. Each single-level processor computes 2-D DWT independently and outputs the low-frequency (LL) coefficient to the next level. The proposed architecture takes two 8-bit inputs and produces four 16-bit output coefficients. Designed control path is responsible to decide the number of decomposition level as per

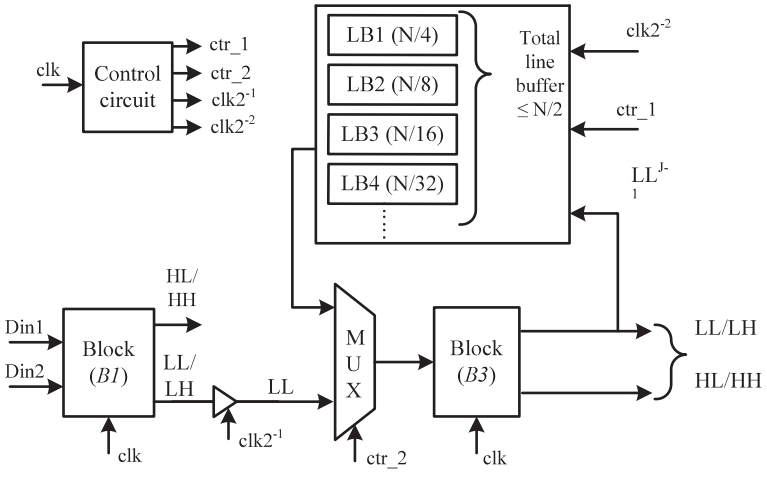

Figure 19 Block diagram of RMA.

Table 5 Performance comparison of 2-D DWT architectures

Architecture	Mul.	Add.	Buff.	C.P.	Thr.	C.C.	HUE (%)
DSA [10]	12	16	4N	$4T_m + 8T_a$	1	$N^2/2$	100
Wu [3]	6	8	4N	T_m	1	N^2	100
FA [13]	10	16	5.5N	$T_m + 2T_a$	2	$N^2/2$	100
HA [13]	18	32	5.5N	$T_m + 2T_a$	4	$N^2/4$	100
Lai [4]	10	16	4N	T_m	2	$N^2/2$	100
Zhang [5]	10	16	4N	T_m	2	$N^2/2$	–
Hsia [6]	0	16	4N	$2T_m + 4T_a$	2	$3N^2/4$	–
Darji [7]	10	16	4N	T_m	2	$N^2/2$	100

Mul., multipliers; Add., adders; Buff., buffers; C.P., critical path; Thr., throughput; C.C., computational cycle; P, parallel factor; HUE, hardware utilization efficiency.

Table 6 Hardware and time complexity comparison of the proposed FMA for lifting (5, 3) filter

Architecture	Multipliers/ shifters	Adders/ subtractors	Memory		Computing time for *j* level	Output latency	Critical path delay	HUE (%)
			On-chip	Off-chip				
Andra [8]	4	8	$N^2 + 4N$	0	$2N^2(1-4^{-j})/3$	$2N$	$2Ta + 2Ts$	100
Wu [3]	4	8	$5N$	$N^2/4$	$2N^2(1-4^{-j})/3$	$2N$	$2Ta + Tm$	100
Barua [22]	4	8	$5N$	$N^2/4$	$2N^2(1-4^{-j})/3$	$5N$	Tm	100
Xiong [13] FA	4	8	$3.5N$	$N^2/4$	$2N^2(1-4^{-j})/3$	N	$2Ta + Tm$	100
Xiong [13] HA	8	16	$3.5N$	$N^2/4$	$N^2(1-4^{-j})/3$	N	$2Ta + Tm$	100
FMA	4	8	$2N$	$N^2/4$	$2N^2(1-4^{-j})/3$	N	$Ta + Ts$	100

Input image size $N \times N$ and $j \geq 1$. Ta, adder delay; Tm, multiplier delay; Ts, shifter delay.

Table 7 Comparison of hardware and time complexity of the proposed FMA for lifting (9, 7) filter

Architecture	Multipliers/ shifters	Adders/ subtractors	Memory		Output latency	Computing time for *j* level	Critical path delay	HUE (%)
			On-chip	Off-chip				
Andra [8]	32	32	N^2	0	$N^2/2$	$4N^2(1-4^{-j})/3$	$4Ta + 2Tm$	100
Wu [3]	6	8	$5.5N$	$N^2/4$	$\sim$	$2N^2(1-4^{-j})/3$	Tm	100
Barua [22]	12	16	$7N$	$N^2/4$	$7N$	$2N^2(1-4^{-j})/3$	$2Ta + Tm$	100
Xiong [13] FA	10	16	$5.5N$	$N^2/4$	$2N$	$2N^2(1-4^{-j})/3$	$2Ta + Tm$	100
Xiong [13] HA	18	32	$5.5N$	$N^2/4$	N	$N^2(1-4^{-j})/3$	$2Ta + Tm$	100
FMA	10	16	$4N$	$N^2/4$	N	$2N^2(1-4^{-j})/3$	$Ta + Tm$	100

Input image size $N \times N$ and $j \geq 1$. Ta, adder delay; Tm, multiplier delay; Ts, shifter delay; $\sim$, not available.

Table 8 Comparison of hardware and time complexity of the proposed PMA for lifting (5, 3) filter

Architecture	Multipliers/ shifters	Adders/ subtractors	Memory $j = 1$		Computing cycle for *j* level	Latency $j = 1$	Critical path delay	HUE (%)
			On-chip	Off-chip				
Hasan [23]	$2j$	j	$3N$	0	$O(N^2)$	$3N$	$2Ta + Ts$	100
Aziz [15]	$2j$	$4j$	$4N$	0	$\sum_{m=1}^{j}\left(1 + \frac{3N}{2^{m-1}} + \frac{N^2}{2^{2(m-1)}}\right)$	$3N + 1$	$2Ta$	50 to 60
PMA	$4 + 6(j-1)$	$8 + 12(j-1)$	$2N$	0	$2 + \sum_{m=1}^{j}\left(10 + \frac{N}{2^{m-1}} + \frac{N^2}{2^{2m-1}}\right)$	$N + 12$	$Ta + Ts$	60 to 75

Input image size $N \times N$ and $j \geq 1$. Ta, adder delay; Tm, multiplier delay; Ts, shifter delay; j, level of decomposition.

Table 9 Line buffer comparison for PMA for lifting (5, 3) filter

Architecture	Memory	
	For $j = 1$	For $j = 5$
Aziz [15]	$4N$	$4N + 2N + N + \frac{N}{2} + \frac{N}{4} \approx 7.7N$
PMA	2N	$2N + \frac{3N}{2} + \frac{3N}{4} + \frac{3N}{8} + \frac{3N}{16} \approx 4.8N$

Input image size $N \times N$.

the need of application as described in Table 4. Due to decimation property of wavelet transform, at each level, total computation is reduced to 1/4 than the preceding stage. To maintain a dual-input processing at every level of decomposition, the clock rate is set as 1/2 than the preceding stage through a timing control circuit. Serial input and parallel output (SIPO) is used in between 2-D DWT modules for pipelining as shown in Figure 17. SIPO block is composed of two registers, which are filled with serial input coming from the previous pipelined stage at the rising edge of clock ($\text{clk}2^{-(j-1)}$) and output is taken at half clock rate ($\text{clk}2^{-j}$) as shown in Figure 17. PMA for lifting (9, 7) filter can be implemented by replicating predict and update modules twice in blocks (*B1*) and (*B2*).

Clock gating is a well-known method to reduce dynamic power dissipation in synchronous digital circuits by adding more logic to a circuit to prune the clock tree (clock distribution network). Pruning the clock disables portions of the circuitry and thus saves the power. Clock gating also saves significant die area, as it removes a large number of multiplexers and replaces them with clock gating logic. Mainly, five clock signals are generated $\text{clk}2^{-1}, \text{clk}2^{-2}, \text{clk}2^{-3}, \text{clk}2^{-4}, \text{clk}2^{-5}$ from *clk*. Clock $\text{clk}2^{-j}$ is $1/2^{-j}$ rate of clk, where *j* indicates decomposition level. Period of enable signal is $N/2$ and signals enable1, enable2, and enable3 are generated from it with period $N, 2N, 4N$, respectively. $\text{clk\_}2^{-2}$ is generated by logical AND operation of $\text{clk}2^{-2}$ and enable signal. $\text{clk\_}2^{-3}$ is generated by logical AND operation of $\text{clk}2^{-3}$, enable, and enable1, and in the same way, $\text{clk\_}2^{-4}$ and $\text{clk\_}2^{-5}$ are generated. $\text{clk_}2^{-2}$_pixel_valid is generated from logical AND of $\text{clk}2^{-2}$, enable, and enable1; in the same way, $\text{clk_}2^{-3}$_pixel_valid is generated from $\text{clk}2^{-3}$, enable, enable1, and enable2 as shown in Figure 18. Clocks and enable used by second-level and third-level block (*B2*) are shown in Figure 17. Every block (*B2*) operates on two gated clocks signals, $\text{clk\_}2^{j}$ and $\text{clk_}2^{j}$_pixel_valid.

3.6.3 Recursive multi-level architecture

The proposed RMA is composed of blocks (*B1*) and (*B3*) as shown in Figure 19. Block (*B1*) is employed to process the input image to get first-level 2-D DWT coefficients, and block (*B3*) works recursively for computing multi-level coefficients. RMA requires line buffer for temporary storage of coefficients generated from the block (*B3*). A multiplexer is used at the input of block (*B3*) to select the input sample either from block (*B1*) or from line buffers. Careful management of the switching of this multiplexer utilizes the idle interleaved clock cycles of block (*B3*) to process higher-level coefficients such that maximum hardware utilization is achieved.

Block (*B1*) produces LL coefficient at alternate clock cycle, which is fed to block (*B3*) for multi-level processing. Block (*B3*) is designed such that it operates one sample per clock cycle. The block (*B3*) also has a feedback mechanism that brings the higher-level LL coefficients at the input to compute next higher-level coefficients. A buffer of size less than $N/2$ is used to store intermediate LL coefficients. This buffer is divided into different lengths, such as $N/4, N/8, N/16 \ldots$ to store one row of LL coefficients. These buffered LL coefficients are serially provided to block (*B3*) to get next higher-level DWT coefficients. Here, a multiplexer is used, which is operated on the control signal ctr_2 and provides first- and higher-level LL coefficients at alternate clock cycle. Thus, every clock cycle is utilized to process the first- and higher-level LL coefficients alternatively. Valid multi-level coefficients are available at every fourth clock cycle. The depth of

Table 10 Comparison of hardware and time complexity of the proposed PMA for lifting (9, 7) filter

Architecture	Mohanty [14]	Mohanty [21]	Hu [19]	Proposed PMA
Scheme	LT	CV	LT	LT
Multiplier	$6Px_3$	189	$\frac{105S}{8} + 6$	32
Adder	$32Px_3/3$	294	21S+12	64
Registers	$N(11x_2 + 10x_5)$	$\frac{21N}{4} + 443$	$3N + \frac{341S}{8}$	48
Line buffers	0	0	0	$4N + 24N$
ACT	$\frac{N^2}{P}$	$\frac{N^2}{16}$	$\frac{N^2}{25}$	$\frac{N^2}{2} + \sum \frac{N}{2^2}$
Critical path delay	$2Ta + Tm$	$\approx Tm$	$Ta + Tm$	$Ta + Tm$

Input image size $N \times N$ and $j = 3$. L.B., line buffer; P, number of samples processed per clock cycle; S, strip size; Ta, adder delay; Tm, multiplier delay; $J = \min(\log_2 M, \log_2 N)$; $x_1 = 2/3 \times (1 - 4^{-L})$; $x_2 = (1 - 2^{-L})$; $x_3 = (1 - 2^{-2L})$.

Table 11 Hardware and time complexity of the proposed RMA

Architecture	Mult./ shift.	Add./ sub.	Memory $j = 1$		Output latency	Critical path delay	HUE (%)
			On-chip	Off-chip			
RMA (5, 3)	6	12	$6.5N$	0	$2N$	$2Ta + Tm$	75
RMA (9, 7)	16	24	$12.5N$	0	$4N$	$2Ta + Tm$	75

Input image size $N \times N$ and $j \geq 1$. Ta, adder delay; Tm, multiplier delay; Ts, shifter delay.

the data is decreased fourfold at each level of operation, i.e., first-level data depth is decreased from N^2 to $N^2/4$, second level to $N^2/16$, and so on. This property of inherent compression is utilized for pushing the higher-level coefficients into buffers.

4 Performance analysis and comparison

In this section, architecture performance is evaluated based on following parameters: HUE, speed, computing time, output latency, line buffers, complexity of control circuit, number of adders and multipliers, system power consumption, configurable logic block (CLB) slices, critical path delay, memory, and maximum frequency of operation. The % HUE is defined by (5).

$$\%\text{HUE} = \frac{\text{Number of block in use}}{\text{Total number of blocks}} \times 100 \tag{5}$$

Pipelining is done between predict and update stages in FMA, PMA, and RMA to reduce the critical path delay from conventional $4Ta + 2Tm$ to only $Ta + Ts$ for lifting (5, 3) and Tm for lifting (9, 7) at the cost of latency of few clock cycles. Total $2N$ on-chip line buffers are required in the proposed FMA and PMA for lifting (5, 3) and $j = 1$ level, which is lowest among existing architectures. The latency of the proposed scheme is N cycles (without boundary treatment). The proposed RMA utilizes only one processing element to calculate both predict and update, resulting into 50% reduction in number of adders and multipliers.

4.1 Hardware complexity and timing analysis

Hardware complexity is mainly sensitive to number of adders, multipliers, on-chip buffers, and control path. Performance comparison of 2-D DWT architecture is given in Table 5. Hardware complexity of folded multi-level structure is reported in literature as [3,8,10,13,22] is compared in Table 6 and Table 7, respectively, for the j-level of decomposition. HA structure in [13] appears to be the best in terms of computing cycles, but it requires 8 multipliers and 16 adders. The proposed FMA has only half-area requirement in terms of on-chip memory buffers, adders/subtractors, and multiplier/shifter required as compared to HA. Moreover, the critical path delay of the proposed FMA for lifting (5, 3) is $Ta + Ts$, which is lowest among similar architectures. Structure in Wu and Lin [3] appears to be the best for lifting (9, 7) in terms adder, multiplier and critical path delay but requires $5.5N$ on-chip buffers for processing as compared to the proposed FMA. The critical path delay of the proposed FMA for (9, 7) is $Ta + Tm$, and it requires only $4N$ on-chip buffers. Hardware and time complexity of the proposed PMA for lifting (5, 3) filter are compared in Table 8 with the architectures described in literature with similar specifications, such as [15] and [23]. The proposed PMA has not only the lowest critical path but it also requires less computing cycles as shown in Table 8. The extra two cycles in the total computing cycle are added due to transposition in block (*B1*). It must be also observed that the architecture proposed by Hasan et al. [23] utilizes less number of multipliers and adders but requires more line buffers to compute multi-level 2-D DWT. It is evident from Table 9 that the PMA uses $4.8N$ line buffers, while the architecture given [15] required $7.7N$ line buffers for five-level DWT decomposition using lifting (5,3) filter.

$$\left(\frac{N^2}{2} + 10\right) + \left[\left(\frac{N}{2} + 8\right) + \left(\frac{N}{4} + 8\right) + \left(\frac{N}{8} + 8\right) + \left(\frac{N}{16} + 8\right)\right] = \frac{N^2}{2} + \frac{15N}{16} + 42 \tag{6}$$

Table 12 Comparison of hardware and time complexity of the proposed RMA for (9, 7) with existing architectures

Arch.	Mult.	Add.	Line buffers	Control complexity	Computing time
Liao [10]	12	16	$10N(1 - 2^{-j})$	Medium	$N^2/2$
Xiong [16]	28	48	$10N(1 - 2^{-j}) + 0.5N$	Medium	$N^2/4$
RMA	16	24	$12.5N$	Simple	$N^2/2$

Image size $N \times N$ and $j \geq 1$. j, level of decomposition.

Table 13 FPGA synthesis results for FMA: image size 256 × 256

Architecture	FMA (5, 3)	FMA (9, 7)
FPGA	Virtex-5	Virtex-5
Device	5VLX110TFF1136-3	5VLX110TFF1136-3
Slice LUTs	494 (0%)	1008 (1%)
Slice registers	633 (0%)	1091 (1%)
MUF (MHz)	537	210

MUF, maximum utilization frequency.

A total computing cycle required by PMA for five-level decomposition of an image (size $N \times N$) is shown in (6). All single-level processors (*B1* and *B2*) work in a parallel fashion in PMA. Block (*B1*) utilizes $\left(\frac{N^2}{2} + 10\right)$, and remaining clocks are consumed in consecutive four blocks of (*B2*). So, most of the processing is done within $N^2/2$ clock cycles and very small additional cycles, i.e., (N/2+8), (N/4+8), (N/8+8), and (N/16+8) are needed to compute further levels.

Architecture comparison of the proposed PMA with [14,19,21] for three-level 2-D DWT using lifting (9, 7) filter is shown in Table 10. The proposed architecture uses lowest number of multipliers and adders as compared to other architectures. Architectures [14,19,21] do not use any line buffers at the cost of more computation logic and complex control path.

The hardware complexity of the proposed RMA for lifting (5, 3) and (9, 7) is shown in Table 11. The RMA for lifting (5, 3) uses a total of six multipliers, out of which four are used in block (*B1*) and two are used in block (*B3*). In case of RMA for lifting (9, 7) filter, a total of 16 multipliers are required, out of which 10 are utilized in block (*B1*) and 6 are used in block (*B3*). The additional two multipliers are required to scale output coefficients. Block (*B1*) uses a total $2N$ and $4N$ line buffers for lifting (5, 3) and lifting (9, 7) filters, respectively. The number of buffers and multipliers required for (9, 7) filter is double than that for (5, 3). HUE of block (*B1*) is 100%, but block (*B2*) has some interleaved clock cycles that results into 75% HUE of all over RMA.

Hardware and time complexity of the proposed RMA for lifting (9, 7) are compared with architectures proposed by [10] and [16] in Table 12. The proposed RMA uses a total of 16 multipliers, out of which block (*B1*) uses 10 and block (*B3*) uses 6. Numbers of adder utilized by blocks (*B1*) and (*B3*) are 16 and 8, respectively. $4N$ line buffers are used by block (*B1*), $8N$ are consumed by block (*B3*), and $0.5N$ are used in storing intermediate-level coefficient, which resulted in a total of $12.5N$ line buffers in the design of RMA. The proposed RMA is simple and reuses same processing element to compute predict and update values. The proposed architecture requires less multipliers, adders, and computing time as compared to [16]. Liao et al. [10] require the lowest number of multipliers and adders because it uses only one 2-D DWT block, but require more complex control path as compared to the proposed RMA.

4.2 FPGA Implementation

The proposed FMA for lifting (5, 3) and (9, 7) filters is implemented on Xilinx Virtex-5 XC5VLX110T FPGA target device (Xilinx, Inc., San Jose, CA, USA), and the results are reported in Table 13. The maximum utilization frequency (MUF) for FMA is very high as a result of very low critical path delay, i.e., $Ta + Ts$ and $Ta + Tm$ for lifting (5, 3) and (9, 7) filters, respectively.

The proposed PMA for five-level lifting (5, 3) is synthesized using Xilinx ISE 10.1 for Xilinx Virtex-5 XC5VLX110T FPGA target device, and the results are reported in Table 14. Sixteen-bit word length is used in the proposed scheme for higher-level decomposition without overflow. Since the pipelined processor element is used in the design, 537-MHz frequency of operation is obtained. The CLB count reported in [15] is less than that in the proposed scheme because of folded processor element, but this approach leads to lower throughput. Throughput is calculated by (7). The proposed PMA utilizes 1,178 slices and provides throughput rate of 4,080 frames/s for lifting (5, 3) for 512 × 512 frame resolution. This is almost five times more than [15]. Power is estimated using Xilinx X-power at a 100-MHz frequency and reported in Table 14. It is apparent from this comparison that the proposed scheme consumes lower power as compared

Table 14 FPGA synthesis results of the proposed PMA for lifting (5, 3) 2-D DWT

Architecture	Power for $j = 1$ (mW)			Frequency	CLB slice count		Throughput frames/s
	Dynamic	Quiescent	Total		For $j = 1$	For $j = 5$	
Aziz [15]	33.85	1,186.94	1,220.79	221.44	206	1,052	835
PMA without clk gating	29.6	980.8	1,010.6	539	412	1,329	4,080
PMA with clk gating	28	980.8	1,008.2	539	342	1,178	4,080

Virtex-5 XC5VLX110T FPGA at 100 MHz. Image size 512 × 512.

Table 15 Comparison of FPGA synthesis results for RMA: image size 256 × 256 and $j = 3$

Arch.	Liao [10]	Xiong [16]	RMA (5, 3)	RMA (9, 7)
FPGA	Virtex-4	Virtex-4	Virtex-4	Virtex-4
Device	4VFX100FF1152-12	4VFX100FF1152-12	4VFX100FF1152-12	4VFX100FF1152-12
LE/slices	1,180	2,532	1,040	1,822

LE, logic elements or cells.

to [15] because of dual-scan technique and lower line buffers are used in design implementation.

$$\text{Throughput} = F_{\max}/\text{Total clocks required for a frame transform} \tag{7}$$

The proposed RMA is synthesized for lifting (5, 3) and (9, 7) filters using Xilinx ISE 10.1 for Xilinx Virtex-4 XC4VFX100 FPGA target device, and the results are reported in Table 15. The RMA implementation uses 1,822 (4%) and 1,040 (2%) slices for lifting (9, 7) and (5, 3) filters, respectively. The output of FMA using lifting (9, 7) filter with $j = 1$ and RMA using lifting (9, 7) filter with $j = 3$ is shown in Figure 20. Output of PMA for lifting (5, 3) filter with $j = 1$ to $j = 5$ levels for input image Cameraman (256 × 256) is shown in Figure 21.

5 Conclusions

In this paper, we have proposed high-performance FMA, PMA and RMA with dual-pixel scanning method for computing multi-level 2-D DWT. The architectures are compared on the basis of resources utilized and speed. Micro-pipelining is employed in predict/update processor element to reduce the critical path to $Ta + Ts$ and $Ta + Tm$ for lifting (5, 3) and (9, 7) filters, respectively. Optimized single-level 2-D DWT blocks (*B1*), (*B2*), and (*B3*) are proposed to design multi-level architecture. The proposed FMA for lifting (5, 3) and lifting (9, 7) uses only $2N$ and $4N$ line buffers, respectively. The proposed PMA is simple, regular, modular, and can be cascaded for n-level decomposition. The PMA for lifting (5, 3) has a critical path delay of $Ta + Ts$. Moreover, it requires only $4.8N$ line buffers for five-level decomposition, thus reduces line buffer approximately 50% than other similar designs. The proposed RMA uses 16 multipliers and 24 adders for n-level decomposition. Moreover, a requirement of line buffers is independent of level of decomposition. The proposed architectures are implemented on Xilinx Virtex family devices. The proposed FMA and PMA operate with frequency of 537 MHz, which is sufficient to handle 518 full-HD frames with $1,920 \times 1,080$ resolution. The proposed PMA, when implemented on FPGA for five-level DWT, utilizes 1,178 slices and provides a throughput rate of 4,080 frames (512 × 512) per second, which is almost five times than that of the existing design. The proposed RMA uses unique buffer management and only single processing element for computing predict and update to save area and power. The Xilinx Virtex-4 implementation of RMA uses 1,822 (4%) and 1,040 (2%) slices for lifting (9, 7) and (5, 3) filters, respectively.

FPGA implementation of the proposed schemes show higher operating frequency, low latency, and lower power as compared to other architectures with the same specifications. The proposed designs can be used for

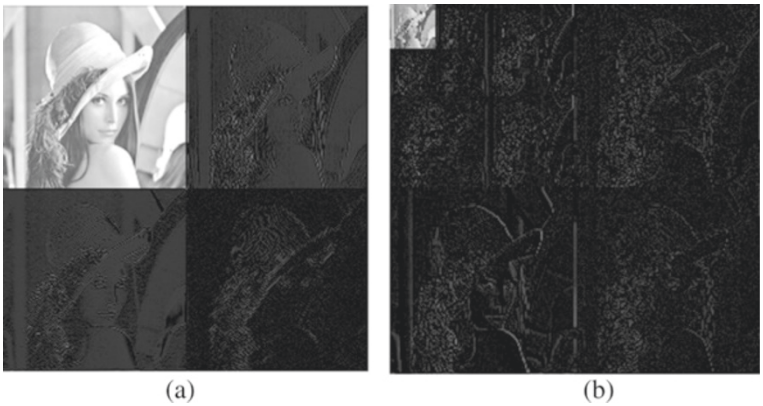

Figure 20 Multi-level output for input image size 256 × 256. (a) FMA lifting (9, 7) and $j = 1$. **(b)** RMA lifting (9, 7) and $j = 3$.

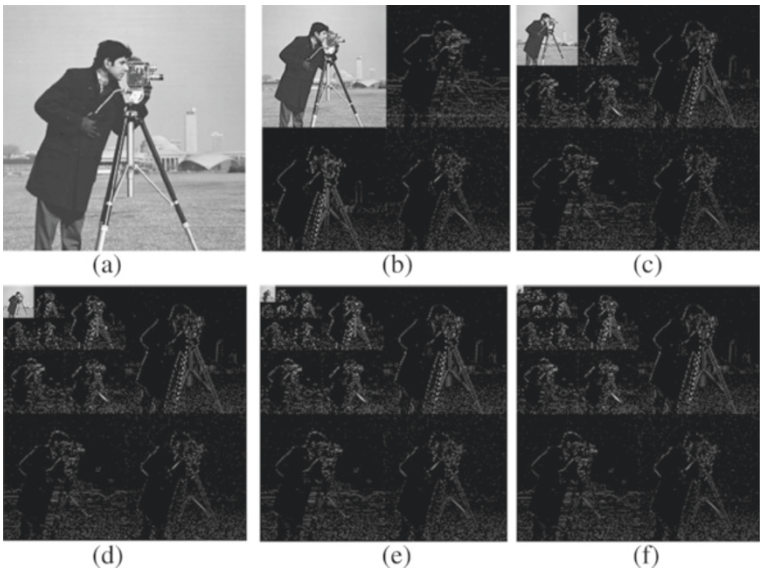

Figure 21 PMA output. (a) Original image. **(b)** One-level, **(c)** two-level, **(d)** three-level, **(e)** four-level, and **(f)** five-level lifting (5, 3) 2-D DWT decomposition of image size 256 × 256.

practical applications with power, area, and speed constrains. The proposed architectures are suitable for high-speed real-time systems such as image de-noising, on-line video streaming, watermarking, compression, and multi-resolution analysis.

Competing interests
The authors declare that they have no competing interests.

Author details
[1]Department of Electrical Engineering, Indian Institute of Technology Bombay, Powai, Mumbai 400076, India. [2]Electronics Engineering Department, S. V. National Institute of Technology, Surat, Gujarat 395007, India.

Received: 26 January 2014 Accepted: 29 September 2014
Published: 4 October 2014

References

1. I Daubecies, W Sweldens, Factoring wavelet transforms into lifting steps. J. Fourier Anal. Appl. **4**, 247–269 (1998)
2. J-M Jou, Y-H Shiau, C-C Liu, Efficient VLSI architectures for the bi-orthogonal wavelet transform by filter bank and lifting scheme, in *Proc. IEEE International Symposium on Circuits Systems (ISCAS)*, vol. 2 (Sydney, New South Wales, 06–09 May 2001), pp. 529–532
3. B Wu, C Lin, A high-performance and memory-efficient pipeline architecture for the 5/3 and 9/7 discrete wavelet transform of JPEG2000 codec. IEEE Trans. Circuits Syst. Video Technol. **15**(12), 1615–1628 (2005)
4. Y-K Lai, L-F Chen, Y-C Shih, A high-performance and memory-efficient VLSI architecture with parallel scanning method for 2-D lifting-based discrete wavelet transform. IEEE Trans. Consum. Electron. **55**(2), 400–407 (2009)
5. W Zhang, Z Jiang, Z Gao, Y Liu, An efficient VLSI architecture for lifting-based discrete wavelet transform. IEEE Trans. Circuits Syst. II. **59**(3), 158–162 (2012)
6. C-H Hsia, J-S Chiang, J-M Guo, Memory-efficient hardware architecture of 2-D dual-mode lifting-based discrete wavelet transform. IEEE Trans. Circuits Syst. Video Technol. **25**(4), 671–683 (2013)
7. A Darji, S Agrawal, A Oza, V Sinha, A Verma, SN Merchant, A Chandorkar, Dual-scan parallel flipping architecture for a lifting-based 2-D discrete wavelet transform. IEEE Trans. Circuits Syst. II, Exp. Briefs. **61**(6), 433–437 (2014)
8. K Andra, C Chakrabarti, T Acharya, A VLSI architecture for lifting-based forward and inverse wavelet transform. IEEE Trans. Signal Process. **50**(4), 966–977 (2002)
9. C-T Huang, P-C Tseng, L-G Chen, Efficient VLSI architectures of lifting-based discrete wavelet transform by systematic design method, in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 5 (Scottsdale, Arizona, 26–29 May 2002), pp. 565–568
10. H Liao, MK Mandal, BF Cockburn, Efficient architectures for 1-D and 2-D lifting-based wavelet transforms. IEEE Trans. Signal Process. **52**(5), 1315–1326 (2004)
11. P-Y Chen, VLSI implementation for one-dimensional multilevel lifting-based wavelet transform. IEEE Trans. Comput. **53**(4), 386–398 (2004)
12. BK Mohanty, PK Meher, VLSI architecture for high-speed / low-power implementation of multilevel lifting, DWT, in *Proceedings of the IEEE Asia Pacific Conference on Circuits and Systems* (Singapore, 4–7 Dec 2006), pp. 458–461
13. C Xiong, J Tian, J Liu, Efficient architectures for two-dimensional discrete wavelet transform using lifting scheme. IEEE Trans. Image Process. **16**(3), 607–614 (2007)
14. BK Mohanty, PK Meher, Memory efficient modular VLSI architecture for highthroughput and low-latency implementation of multilevel lifting 2-D DWT. IEEE Trans. Signal Process. **59**(5), 2072–2084 (2011)
15. SM Aziz, DM Pham, Efficient parallel architecture for multi-level forward discrete wavelet transform processors. J. Comp. Elect. Eng. **38**, 1325–1335 (2012)
16. C-Y Xiong, J-W Tian, J Liu, Efficient high-speed/low-power line-based architecture for two-dimensional discrete wavelet transform using lifting scheme. IEEE Trans. Circuits Syst. Video Technol. **16**(2), 309–316 (2006)
17. C-H Hsia, J-M Guo, J-S Chiang, Improved low-complexity algorithm for 2-D integer lifting-based discrete wavelet transform using symmetric mask-based scheme. IEEE Trans. Circuits Syst. Video Technol. **19**(8), 1202–1208 (2009)
18. AK Al-Sulaifanie, A Ahmadi, M Zwolinski, Very large scale integration architecture for integer wavelet transform. J. IET Comp. Digital Tech. **4**(6), 471–483 (2010)

19. Y Hu, CC Jong, A memory-efficient high-throughput architecture for lifting-based multi-level 2-D DWT. IEEE Trans. Signal Process. **61**(20), 4975–4987 (2013)
20. ME Angelopoulou, K Masselos, PY Cheung, Y Andreopoulos, Implementation and comparison of the 5/3 lifting 2-D discrete wavelet transform computation schedules on FPGAs. J. Signal Process. Syst. **51**(1), 3–21 (2008)
21. BK Mohanty, PK Meher, Memory-efficient high-speed convolution-based generic structure for multilevel 2-D DWT. IEEE Trans. Circuits Syst. Video Technol. **23**(2), 353–363 (2013)
22. S Barua, JE Carletta, KA Kotteri, AE Bell, An efficient architecture for lifting-based two-dimensional discrete wavelet transforms. J. Integration, VLSI J. **38**(3), 341–352 (2005)
23. H Varshney, M Hasan, S Jain, Energy efficient novel architectures for the lifting-based discrete wavelet transform. IET Image Process. **1**(3), 305–310 (2007)

doi:10.1186/1687-5281-2014-47
Cite this article as: Darji *et al.*: **High-performance hardware architectures for multi-level lifting-based discrete wavelet transform.** *EURASIP Journal on Image and Video Processing* 2014 **2014**:47.