We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

6,900
Open access books available

185,000
International authors and editors

200M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities



WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com



2

High Performance Parallel Pipelined Lifting-based VLSI Architectures for Two-Dimensional Inverse Discrete Wavelet Transform

Ibrahim Saeed Koko and Herman Agustiawan
Electrical & Electronic Engineering Department
Universiti Teknologi PETRONAS, Tronoh
Malaysia

1. Introduction

Two-dimensional discrete wavelet transform (2-D DWT) has evolved as an effective and powerful tool in many applications especially in image processing and compression. This is mainly due to its better computational efficiency achieved by factoring wavelet transforms into lifting steps. Lifting scheme facilitates high speed and efficient implementation of wavelet transform and it attractive for both high throughput and low-power applications (Lan & Zheng, 2005).

DWT considered in this work is part of a compression system based on wavelet such as JPEG2000. Fig.1 shows a simplified compression system. In this system, the function of the 2-D FDWT is to decompose an *NxM* image into subbands as shown in Fig. 2 for 3-level decomposition. This process decorrelates the highly correlated pixels of the original image. That is the decorrelation process reduces the spatial correlation among the adjacent pixels of the original image such that they can be amenable to compression.

After transmitting to a remote site, the original image must be reconstructed from the decorrelated image. The task of the reconstruction and completely recovering the original image from the decorrelated image is performed by inverse discrete wavelet transform (IDWT).

The decorrelated image shown in Fig. 2 can be reconstructed by 2-D IDWT as follows. First, it reconstructs in the column direction subbands LL3 and LH3 column-by-column to recover L3 decomposition. Similarly, subbands HL3 and HH3 are reconstructed to obtain H3 decomposition. Then L3 and H3 decompositions are combined row-wise to reconstruct subband LL2. This process is repeated in each level until the whole image is reconstructed (Ibrahim & Herman, 2008).

The reconstruction process described above implies that the task of the reconstruction can be achieved by using 2 processors (Ibrahim & Herman, 2008). The first processor (the column-processor) computes column-wise to combine subbands LL and LH into L and subbands HL and HH into H, while the second processor (the row-processor) computes row-wise to

combine L and H into the next level subband. The decorrelated image shown in Fig. 2 is assumed to be residing in an external memory with the same format.
In this chapter, parallelism is explored to best meet real-time applications of 2-D DWT with demanding requirements. The single pipelined architecture developed by (Ibrahim & Herman, 2008) is extended to 2- and 4-parallel pipelined architectures for both 5/3 and 9/7 inverse algorithms to achieve speedup factors of 2 and 4, respectively. The advantage of the proposed architectures is that the total temporary line buffer (TLB) size does not increase from that of the single pipelined architecture when degree of parallelism is increased.

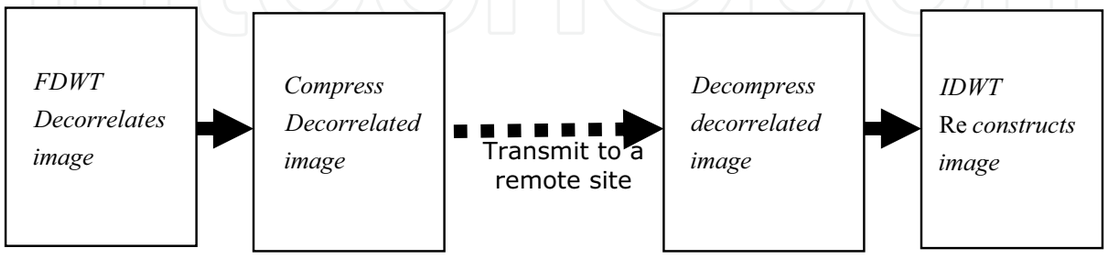


Fig. 1. A simplified Compression System

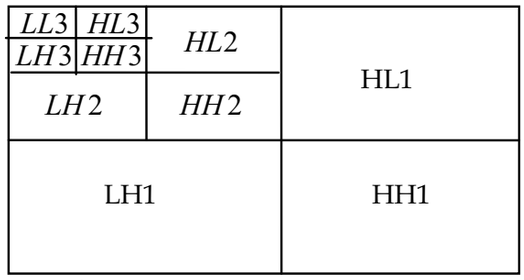


Fig. 2. Subband decomposition of an *NxM* image into 3 levels.

2. Lifting-based 5/3 and 9/7 synthesis algorithms and data dependency graphs

The 5/3 and the 9/7 inverse discrete wavelet transforms algorithms are defined by the JPEG2000 image compression standard as follow (Lan & Zheng, 2005):
5/3 synthesis algorithm

$$step1: X(2n) = Y(2n) - \left\lfloor \frac{Y(2n-1) + Y(2n+1) + 2}{4} \right\rfloor$$
$$step2: X(2n+1) = Y(2n+1) + \left\lfloor \frac{X(2n) + X(2n+2)}{2} \right\rfloor \quad (1)$$

9/7 synthesis algorithm

$$\text{Step 1: } Y'(2n) = 1/k \cdot Y(2n)$$
$$\text{Step 2: } Y'(2n+1) = k \cdot Y(2n+1)$$
$$\text{Step 3: } Y''(2n) = Y'(2n) - \delta(Y'(2n-1) + Y'(2n+1))$$
$$\text{Step 4: } Y''(2n+1) = Y'(2n+1) - \gamma(Y''(2n) + Y''(2n+2)) \quad (2)$$

Step 5: $X(2n) = Y''(2n) - \beta(Y''(2n-1) + Y''(2n+1))$

Step 6: $X(2n+1) = Y''(2n+1) - \alpha(X(2n) + X(2n+2))$

The data dependency graphs (DDGs) for 5/3 and 9/7 derived from the synthesis algorithms are shown in Figs. 3 and 4, respectively. The DDGs are very useful tools in architecture development and provide the information necessary for designer to develop more accurate architectures. The symmetric extension algorithm recommended by JPEG2000 is incorporated into the DDGs to handle the boundaries problems. The boundary treatment is necessary to keep number of wavelet coefficients the same as that of the original input Pixels and as a result prevent distortion from appearing at image boundaries. The boundary treatment is only applied at the beginning and ending of each row or column in an *NxM* image (Dillin et al., 2003). In DDGs, nodes circled with the same numbers are considered redundant computations, which will be computed once and used thereafter. Coefficients of the nodes circled with even numbers in the DDGs are low coefficients and that circled with odd numbers are high coefficients.

3. Scan methods

The hardware complexity and hence the memory required for 2-D DWT architecture in general depends on the scan method adopted for scanning external memory. Therefore, the scan method shown in Fig. 5 is proposed for both 5/3 and 9/7 CPs. Fig. 5 (A) is formed for illustration purposes by merging together subbands LL and LH, where subband LL coefficients occupy even row and subband LH coefficients occupy odd rows, while Fig. 5 (B) is formed by merging subband HL and HH together.

According to the scan method shown in Fig. 5, CPs of both 5/3 and 9/7 should scan external memory column-by-column. However, to allow the RP, which operates on data generated by the CP, to work in parallel with the CP as earlier as possible, the A's (LL+LH) first two columns coefficients are interleaved in execution with the B's (HL+HH) first two columns coefficients, in the first run. In all subsequent runs, two columns are interleaved, one from A with another from B.

Interleaving of 4 columns in the first run takes place as follows. First, coefficients LL0,0, LH0,0 from the first column of (*A*) are scanned. Second, coefficients HL0,0 and HH0,0 from the first column of *B* are scanned, then LL0,1 and LH0,1 from the second column of *A* followed by HL0,1 and HH0,1 from the second column of *B* are scanned. The scanning process then returns to the first Fig. 3. 5/3 synthesis algorithm's DDGs for (a) odd and (b) even length signals returns to the first column of *A* to repeat the process and so on.

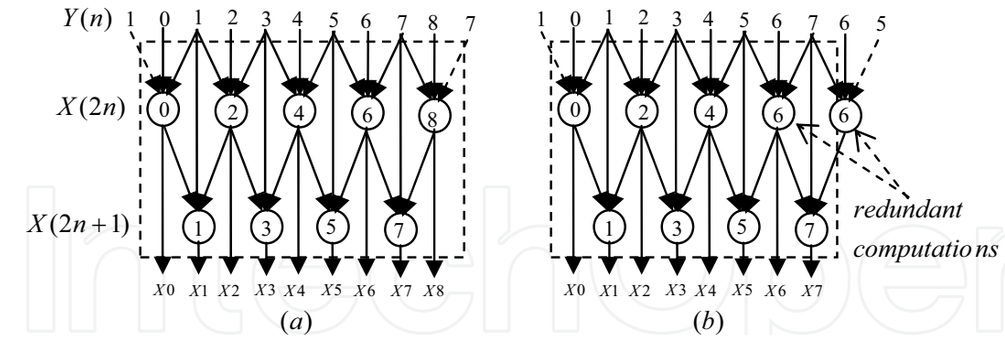


Fig. 3. 5/3 synthesis algorithm's DDGs for (a) odd and (b) even length signals

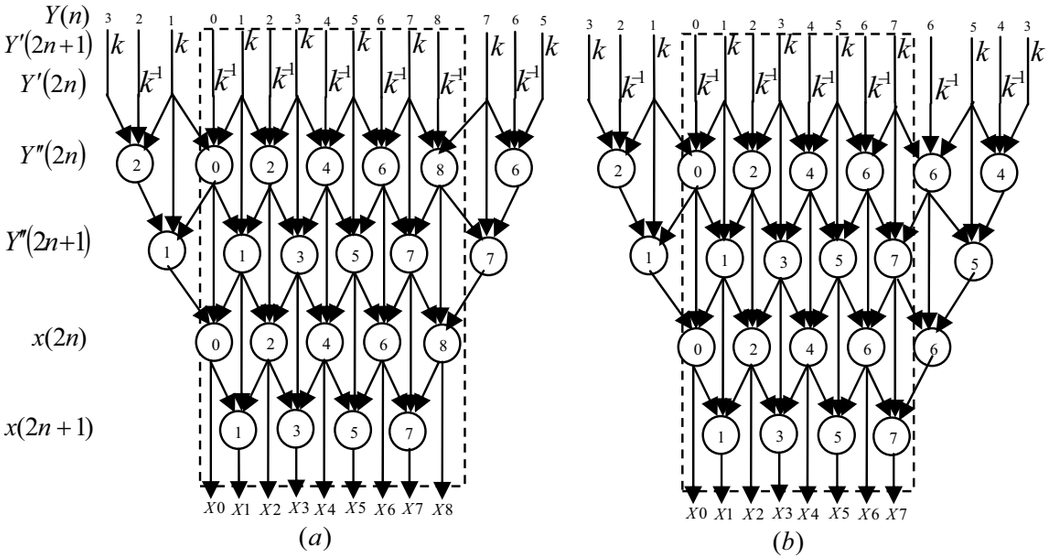


Fig. 4. 9/7 synthesis algorithm's DDGs for (a) odd and (b) even length signals

The advantage of interleaving process not only it speedups the computations by allowing the two processors to work in parallel earlier during the computations, but also reduces the internal memory requirement between CP and RP to a few registers.

The scan method illustrated in Fig. 5 for 5/3 and 9/7 CP along with the DDGs suggest that the RP should scan coefficients generated by CP according to the scan method illustrated in Fig. 6. This figure is formed for illustration purposes by merging L and H decompositions even though they are actually separate. In Fig. 6, L's coefficients occupy even columns, while H's coefficients occupy odd columns. In the first run, the RP's scan method shown in Fig. 6 requires considering the first four columns for scanning as follows. First, coefficients L0,0 and H0,0 from row 0 followed by L1,0 and H1,0 from row 1 are scanned. Then the scan returns to row 0 and scans coefficients L0,1 and H0,1 followed by L1,1 and H1,1. This process is repeated as shown in Fig. 6 until the first run completes.

In the second run, coefficients of columns 4 and 5 are considered for scanning by RP as shown in Fig. 6, whereas in the third run, coefficients of columns 6 and 7 are considered for scanning and so on.

According to the 9/7 DDGs, the RP's scan method shown in Fig. 6 will generate only one low output coefficient each time it processes 4 coefficients in the first run, whereas 5/3 RP will generate two output coefficients. However, in all subsequent runs, both 9/7 and 5/3 RPs generate two output coefficients.

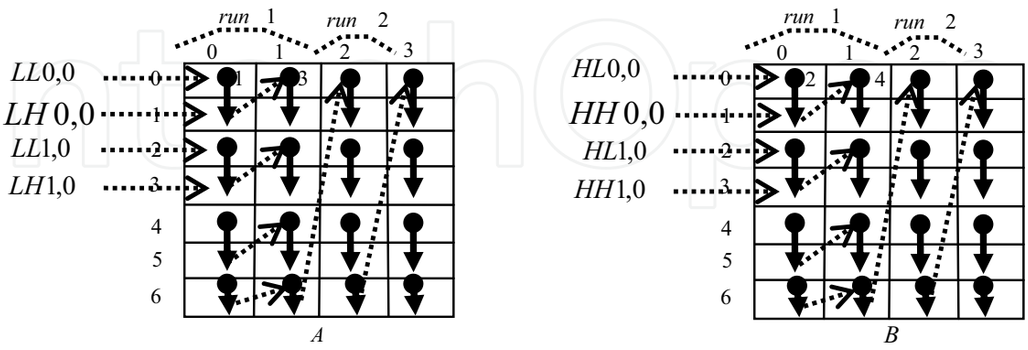


Fig. 5. 5/3 and 9/7 CPs scan method

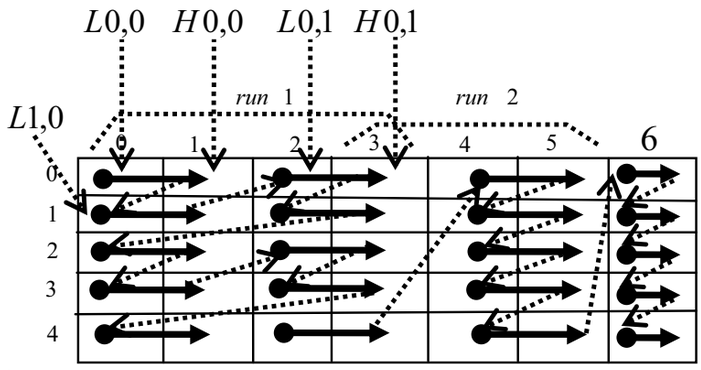


Fig. 6. 5/3 and 9/7 RPs scan method

4. Approach

In (Ibrahim & Herman, 2008), to ease the architecture development the strategy adopted was to divide the details of the development into two steps each having less information to handle. In the first step, the DDGs were looked at from the outside, which is specified by the dotted boxes in the DDGs, in terms of the input and output requirements. It can be observed that the DDGs for 5/3 and 9/7 are identical when they are looked at from outside, taking into consideration only the input and output requirements; but differ in the internal details. Based on this observation, the first level, called external architecture, which is identical for both 5/3 and 9/7, and consists of a column-processor (CP) and a row-processor (RP), was developed. In the second step, the internal details of the DDGs for 5/3 and 9/7 were considered separately for the development of processors' datapath architectures, since DDGs internally define and specify the internal structure of the processors

In this chapter, the first level, the external architectures for 2-parallel and 4-parallel are developed for both 5/3 and 9/7 inverse algorithms. Then the processors' datapath

architectures developed in (Ibrahim & Herman, 2008) are modified to fit into the two proposed parallel architectures' processors.

4.1 Proposed 2-parallel external architecture

Based on the scan methods shown in Figs. 5 and 6 and the DDGs for 5/3 and 9/7, the 2-parallel external architecture shown in Fig. 7 (a) is proposed for 5/3 and 9/7 and combined 5/3 and 9/7 for 2-D IDWT. The architecture consists of two *k*-stage pipelined column-processors (CPs) labeled CP1 and CP2 and two *k*-stage pipelined row-processors (RPs) labeled RP1 and RP2. The waveforms of the two clocks f_2 and $f_2/2$ used in the architecture are shown in Fig. 7 (b).

In general, the scan frequency f_l and hence the period $\tau_l = 1/f_l$ of the parallel architectures can be determined by the following algorithm when the required pixels I of an operation are scanned simultaneously in parallel. Suppose t_p and t_m are the processor and the external memory critical path delays, respectively.

$$\begin{aligned} &If\ t_p/l \cdot k \geq t_m \quad then \\ &\qquad \tau_l = t_p/l \cdot k \\ &else \quad \tau_l = t_m \end{aligned} \tag{3}$$

Where l = 2, 3, 4 ... denote 2, 3, and 4-parallel and t_p/k is the stage critical path delay of a k-stage pipelined processor. The clock frequency f_2 is determined from Eq(3) as

$$f_2 = 2k/t_p \tag{4}$$

The architecture scans the external memory with frequency f_2 and it operates with frequency $f_2/2$. Each time two coefficients are scanned through the two buses labeled *bus0* and *bus1*. The two new coefficients are loaded into CP1 or CP2 latches Rt0 and Rt1 every time clock $f_2/2$ makes a negative or a positive transition, respectively.

IntechOpen

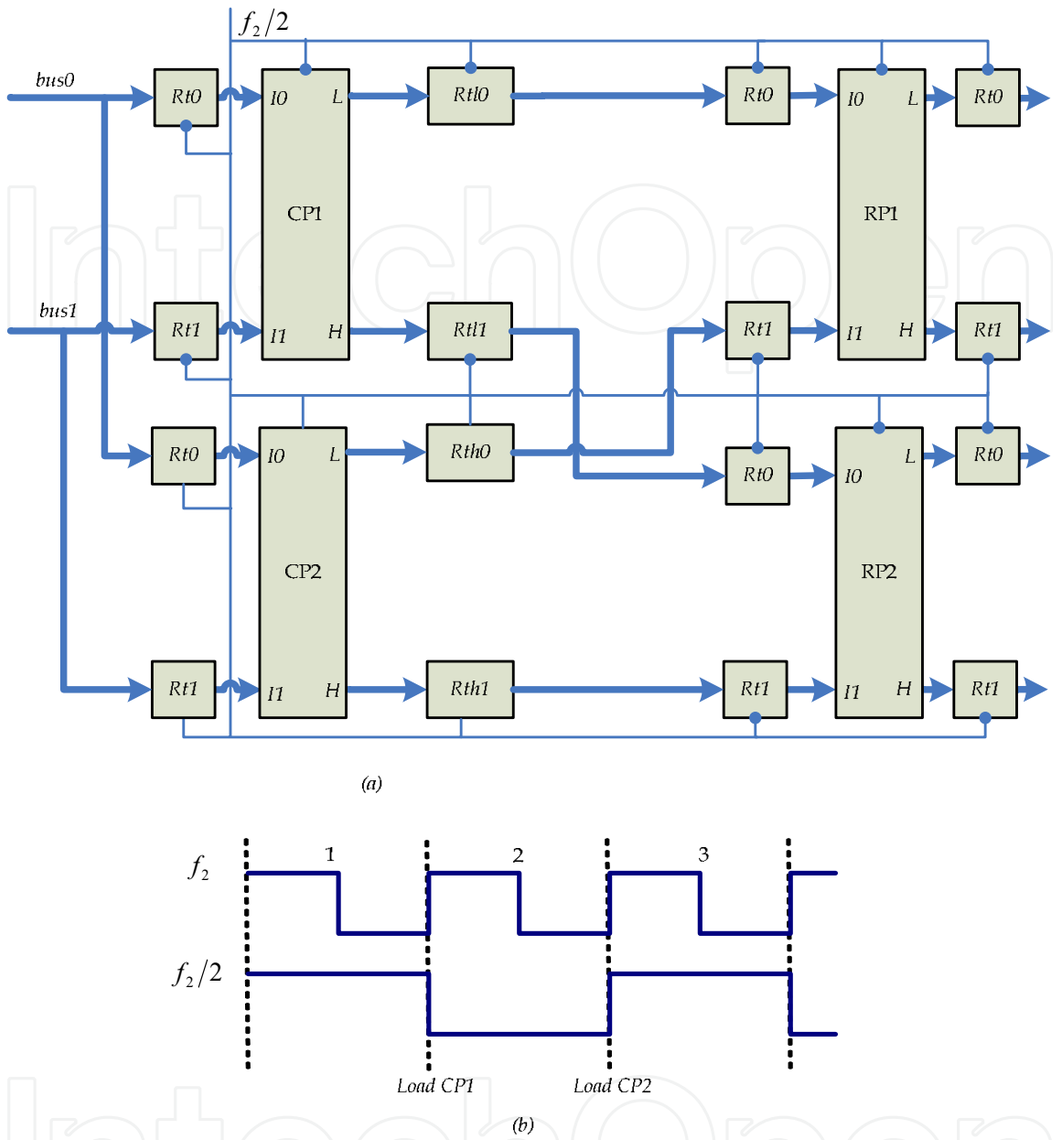


Fig. 7. (a) Proposed 2-parallel pipelined external architecture for 5/3 and 9/7 and combined (b) Waveform of the clocks.

On the other hand, both RP1 and RP2 latches Rt0 and Rt1 load simultaneously new data from CP1 and CP2 output latches each time clock $f_2/2$ makes a negative transition.

The dataflow for 5/3 2-parallel architecture is shown in Table 1, where CPs and RPs are assumed to be 4-stage pipelined processors. The dataflow for 9/7 2-parallel architecture is similar, in all runs, to the 5/3 dataflow except in the first, where RP1 and RP2 of the 9/7 architecture each would generate one output coefficient every other clock cycle, reference to clock $f_2/2$. The reason is that each 4 coefficients of a row processed in the first run by RP1 or RP2 of the 9/7 would require, according to the DDGs, two successive low coefficients

from the first level of the DDGs labeled $Y''(2n)$ in order to carry out node 1 computations in the second level labeled $Y''(2n+1)$. In Table 1, the output coefficients in Rt0 of both RP1 and RP2 represent the output coefficients of the 9/7 in the first run.

The strategy adopted for scheduling memory columns for CP1 and CP2 of the 5/3 and 9/7 2-parallel architectures, which are scanned according to the scan method shown in Fig. 5, is as follow. In the first run, both 5/3 and 9/7 2-parallel architectures are scheduled for executing 4 columns of memory, two from each (A) and (B) of Fig. 5. The first two columns of Fig. 5 (A) are executed in an interleaved manner by CP1, while the first two columns of Fig. 5 (B) are executed by CP2 also in an interleaved fashion as shown in the dataflow Table 1 In all other runs, 2 columns are scheduled for execution at a time. One column from (A) of Fig. 5 will be scheduled for execution by CP1, while another from (B) of Fig. 5 will be scheduled for CP2. However, if number of columns in (A) and (B) of Fig. 5 is not equal, then the last run will consist of only one column of (A). In that case, schedule the last column in CP1 only, but its output coefficients will be executed by both RP1 and RP2. The reason is that if the last column is scheduled for execution by both CP1 and CP2, they will yield more coefficients than that can be handled by both RP1 and RP2.

On the other hand, scheduling RP1 and RP2 of 5/3 and 9/7 2-parallel architectures occur according to scan method shown in Fig. 6. In this scheduling strategy, all rows of even and odd numbers in Fig. 6 will be scheduled for execution by RP1 and RP2, respectively. In the first run, 4 coefficients from each 2 successive rows will be scheduled for RP1 and RP2, whereas in all subsequent runs, two coefficients of each 2 successive rows will be scheduled for RP1 and RP2, as shown in Fig. 6. However, if number of columns in Fig. 6 is odd which occur when number of columns in (A) and (B) of Fig. 5 is not equal, then the last run would require scheduling one coefficient from each 2 successive rows to RP1 and RP2 as shown in column 6 of Fig. 6.

In general, all coefficients that belong to columns of even numbers in Fig. 6 will be generated by CP1 and that belong to columns of odd numbers will be generated by CP2. For example, in the first run, CP1will first generate two coefficients labeled L0,0 and L1,0 that belong to locations 0,0 and 1,0 in Fig. 6, while CP2 will generate coefficient H0,0 and H1,0 that belong to locations 0,1 and 1,1. Then coefficients in locations 0,0 and 0,1 are executed by RP1, while coefficients of locations 1,0 and 1,1 are executed by RP2. Second, CP1 will generate two coefficients for locations 0,2 and 1,2, while CP2 generates two coefficients for locations 0,3 and 1,3. Then coefficients in locations 0,2 and 0,3 are executed by RP1, while coefficients in locations 1,2 and 1,3 are executed by RP2. The same process is repeated in the next two rows and so on.

In the second run, first, CP1 generates coefficients of locations 0,4 and 1,4, whereas CP2 generates coefficients of locations 0,5 and 1,5 in Fig. 6. Then coefficients in locations 0,4 and 0,5 are executed by RP1, while coefficients in locations 1,4 and 1,5 are executed by RP2. This process is repeated until the run completes. However, in the even that the last run processes only one column of (A), CP1 would generate first coefficients of locations 0,m and 1,m where m refers to the last column. Then coefficients of location 0,m is passed to RP1, while coefficient of location 1,m is passed to RP2. In the second time, CP1 would generate coefficients of locations 2,m and 3,m. Then 2,m is passed to RP1 and 3,m to RP2 and so on.

In the following, the dataflow shown in Table 1 for 2-parallel pipelined architecture will be explained. The first run, which ends at cycle 16 in the table, requires scheduling four columns as follows. In the first clock cycle, reference to clock f_2 , coefficients LL0,0 and

LH0,0 from the first column of LL3 and LH3 in the external memory, respectively, are scanned and are loaded into CP1 latches Rt0 and Rt1 by the negative transition of clock $f_2/2$. The second clock cycle scans coefficients HL0,0 and HH0,0 from the first column of HL3 and HH3, respectively, through the buses labeled *bus0* and *bus*1 and loads them into CP2 latches Rt0 and Rt1 by the positive transition of clock $f_2/2$. In the third clock cycle, the scanning process returns to the second column of subbands LL3 and LH3 in the external memory and scans coefficients LL0,1 and LH0,1, respectively, and loads them into CP1 latches Rt0 and Rt1 by the negative transition of the clock $f_2/2$. The fourth clock cycle scans coefficients HL0,1 and HH0,1 from the second column of HL3 and HH3, respectively, and loads them into CP2 latches Rt0 and Rt1. The scanning process then returns to the first column in subbands LL3 and LH3 to repeat the process until the first run is completed.

In cycle 9, CP1 generates its first two output coefficients L0,0 and L1,0, which belong to L3 decomposition and loads them into its output latches Rtl0 and Rtl1, respectively, by the negative transition of clock $f_2/2$. In cycle 10, CP2 generates its first two output coefficients H0,0 and H1,0, which belong to H3 decomposition and loads them into its output latches Rth0 and Rth1, respectively, by the positive transition of clock $f_2/2$.

In cycle 11, contents of Rtl0 and Rth0 are transferred to RP1 input latches Rt0 and Rt1, respectively. The same clock cycle also transfers contents of Rtl1 and Rth1 to RP2 input latches Rt0 and Rt1, respectively, while the two coefficients L0,1 and L1,1 generated by CP1 during the cycle are loaded into Rtl0 and Rtl1, respectively, by the negative transition of clock $f_2/2$.

In cycle 21, both RP1 and RP2 each yield its first two output coefficients, which are loaded into their respective output latches by the negative transition of clock $f_2/2$. Contents of these output latches are then transferred to external memory where they are stored in the first 2 memory locations of each rows 0 and 1. The dataflow of the first run then proceeds as shown in Table 1. The second run begins at cycle 19 and yields its first 4 output coefficients at cycle 37.

	Ck f_2	CP	CP1 & CP2 input latches Rt0 Rt1	CP1 output latches Rtl0 Rtl1	CP2 output latches Rth0 Rth1	RP1 input latches Rt0 Rt1	RP2 input latches Rt0 Rt1	Output latches of RP1 RP2 Rt0 Rt1 Rt0 Rt1
RUN 1	1	1	LL0,0 LH0,0					
	2	2	HL0,0 HH0,0					
	3	1	LL0,1 LH0,1					
	4	2	HL0,1 HH0,1					
	5	1	LL1,0 LH1,0					
	6	2	HL1,0 HH1,0					
	7	1	LL1,1 LH1,1					
	8	2	HL1,1 HH1,1					
	9	1	LL2,0 LH2,0	L0,0 L1,0				
	10	2	HL2,0 HH2,0		H0,0 H1,0			
	11	1	LL2,1 LH2,1	L0,1 L1,1		L0,0 H0,0	L1,0 H1,0	
	12	2	HL2,1 HH2,1		H0,1 H1,1			
	13	1	LL3,0 LH3,0	L2,0 L3,0		L0,1 H0,1	L1,1 H1,1	
	14	2	HL3,0 HH3,0		H2,0 H3,0			
	15	1	LL3,1 LH3,1	L2,1 L3,1		L2,0 H2,0	L3,0 H3,0	
	16	2	HL3,1 HH3,1		H2,1 H3,1			
RUN 2	17	1	------ ------- -	L4,0 L5,0		L2,1 H2,1	L3,1 H3,1	
	18	2	------ -------		H4,0 H5,0			
	19	1	LL0,2 LH0,2	L4,1 L5,1		L4,0 H4,0	L5,0 H5,0	
	20	2	HL0,2 HH0,2		H4,1 H5,1			
	21	1	LL1,2 LH1,2	L6,0 L7,0		L4,1 H4,1	L5,1 H5,1	X0,0 X0,1 X1,0 X1,1
	22	2	HL1,2 HH1,2		H6,0 H7,0			
	23	1	LL2,2 LH2,2	L6,1 L7,1		L6,0 H6,0	L7,0 H7,0	X0,2 ---- X1,2 -----

	24	2	HL2,2 HH2,2		H6,1 H7,1			
	25	1	LL3,2 LH3,2	----- ------		L6,1 H6,1	L7,1 H7,1	X2,0 X2,1 X3,0 X3,1
	26	2	HL3,2 HH3,2		----- -----			
	27	1		L0,2 L1,2		----- ---- -	----- -- ----	X2,2 ---- X3,2 -----
	28	2			H0,2 H1,2			
	29	1		L2,2 L3,2		L0,2 H0,2	L1,2 H1,2	X4,0 X4,1 X5,0 X5,1
	30	2			H2,2 H3,2			
	31	1		L4,2 L5,2		L2,2 H2,2	L3,2 H3,2	X4,2 ---- X5,2 -----
	32	2			H4,2 H5,2			
	33	1		L6,2 L7,2		L4,2 H4,2	L5,2 H5,2	X6,0 X6,1 X7,0 X7,1
	34	2			H6,2 H7,2			
	35	1				L6,2 H6,2	L7,2 H7,2	X6,2 ---- X7,2 -----
	36	2						
	37	1						X0,3 X0,4 X1,3 X1,4
	38	2						
	39	1						X2,3 X2,4 X3,3 X3,4

Table 1. Dataflow for 2-parallel 5/3 architecture

4.2 Modified CPs and RPs for 5/3 and 9/7 2-parallel external architecture

Each CP of the 2-parallel external architecture is required to execute two columns in an interleave fashion in the first run and one column in all other runs. Therefore, the 5/3 processor datapath developed in (Ibrahim & Herman, 2008) should be modified as shown in Fig. 8 by adding one more stage between stages 2 and 3 for 5/3 2- parallel external architecture to allow interleaving of two columns as described in the dataflow Table 1. Through the two multiplexers labeled *mux* the processor controls between executing 2 columns and one column. Thus, in the first run, the two multiplexers' control signal labeled s is set 1 to allow interleaving in execution and 0 in all other runs. The modified 9-stage CP for 9/7 2-parallel external architecture can be obtained by cascading two copies of Fig. 8.

On the other hand, RP1 and RP2 of the proposed 2-parallel architecture for 5/3 and 9/7 are required to scan coefficients of H and L decompositions generated by CP1 and CP2 according to the scan method shown in Fig. 6. In this scan method, all rows of even numbers are executed by RP1 and all rows of odds numbers are executed by RP2. That is, while RP1 is executing row0 coefficients, RP2 will be executing row1 coefficients and so on. In addition, looking at the DDGs for 5/3 and 9/7 show that applying the scan methods in Fig.

6 would require inclusion of temporary line buffers (TLBs) in RP1 and RP2 of the proposed 2-parallel external architecture as follows. In the first run, the fourth input coefficient of each row in the DDGs and the output coefficients labeled X(2) in the 5/3 DDGs and that labeled Y"(2), Y"(1), and X(0) in the 9/7 DDGs, generated by considering 4 inputs coefficients in each row, should be stored in TLBs, since they are required in the next run's computations. Similarly, in the second run, the sixth input coefficient of each row and the output coefficients labeled X(4) in the 5/3 DDGs and that labeled Y"(4), Y"(3), and X(2) in the 9/7 DDGs generated by considering 2 inputs coefficients in each row, should be stored in TLBs. Accordingly, 5/3 would require addition of 2 TLBs each of size N, whereas 9/7 would require addition of 4 TLBs each of size N. However, since 2-parallel architecture consists of two RPs, each 5/3 RP will have 2 TLBs each of size N/2 and each 9/7 RP will have 4 TLBs each of size N/2 as shown in Fig. 9. Fig. 9 (a) represents the 5/3 modified RP, while both (a) and (b) represent the 9/7 modified RP for 2- parallel architecture.

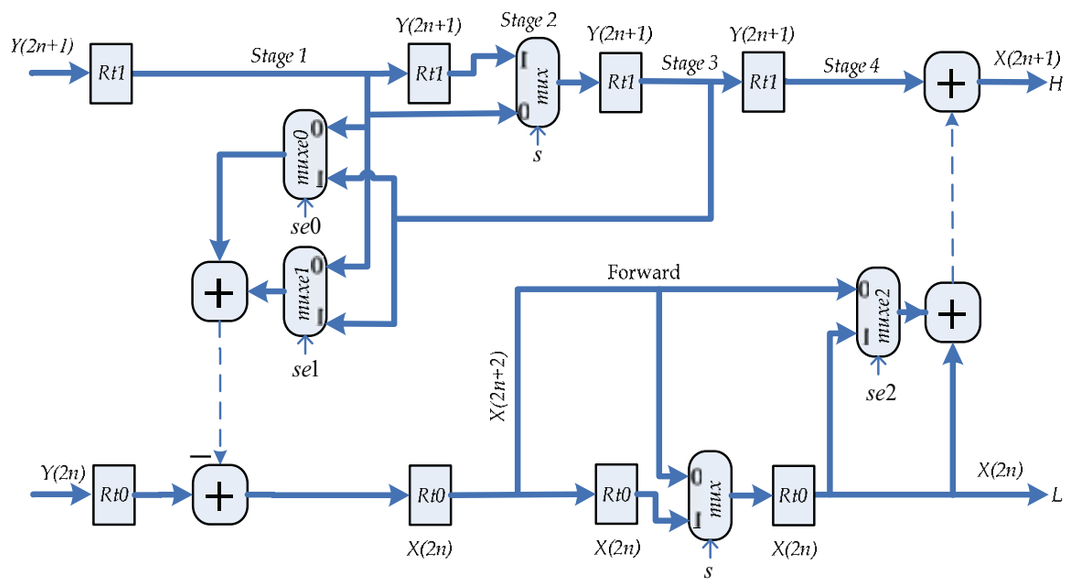


Fig. 8. Modified inverse 5/3 CP for 2-parallel External architecture

To have more insight into the two RPs operations, the dataflow for 5/3 RP1 is given in Table 2 for first and second runs. Note that stage 1 input coefficients in Table 2 are exactly the same input coefficients of RP1 in Table 1. In the first run, TLBs are only written, but in the second run and in all subsequent runs, TLBs are read in the first half cycle and written in the second half cycle. In the cycle 15, Table 2 shows that coefficients H0,1 is stored in the first location of TLB1, while coefficient H2,1 is stored in the second location in cycle 19 and so on. Run 2 starts at cycle 29. In cycle 30, the first location of TLB1, which contains coefficients H0,1 is read during the first half cycle of clock $f_2/2$ and is loaded into Rd1 by the positive transition of the clock, whereas coefficient H0,2 is written into the same location in the second half cycle. Then, the negative transition of clock $f_2/2$ transfers contents of Rd1 to Rt2 in stage 2.

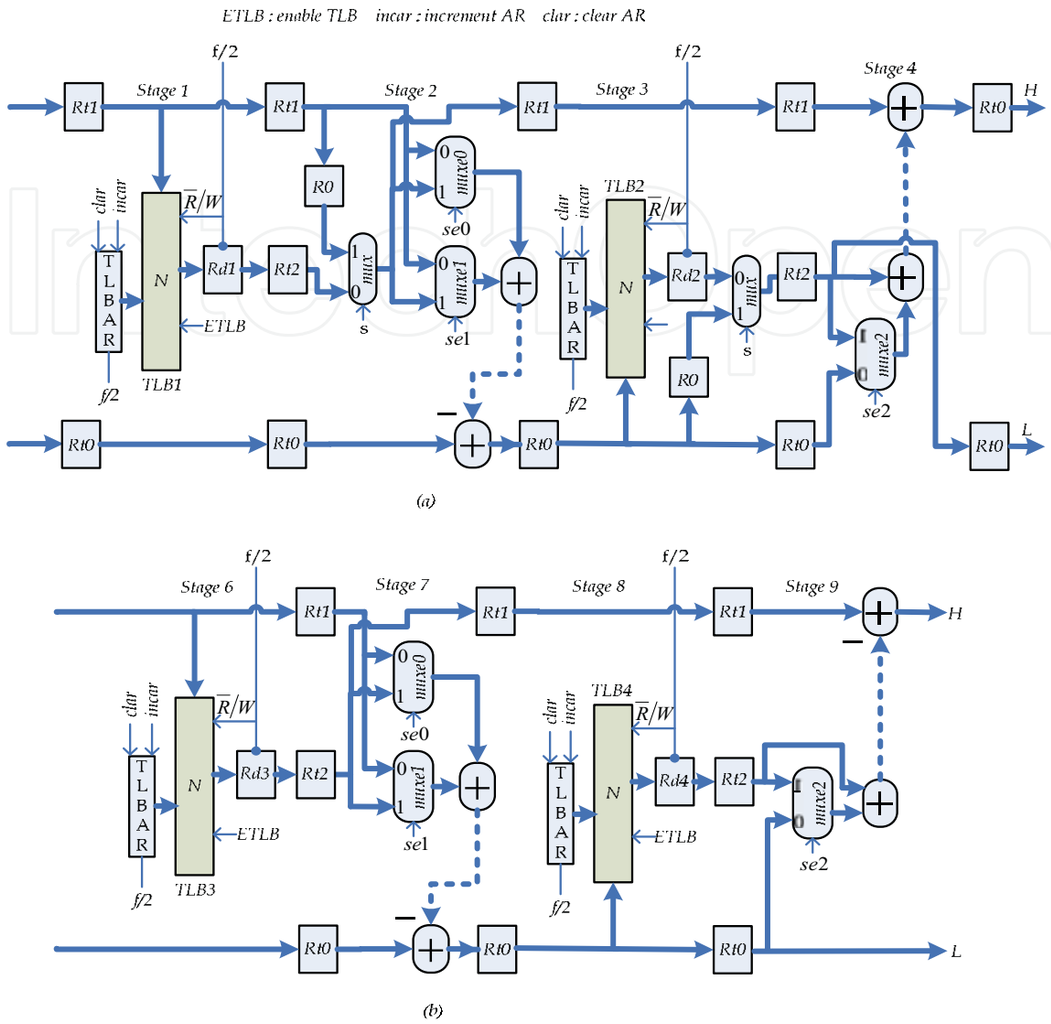


Fig. 9. Modified RP for 2-parallel architecture (a) 5/3 and (a) & (b) 9/7

	CK f_2	RP1 input latches STAGE 1 Rt0 Rt1 TLB1	STAGE 2 Rt0 Rt2 Rt1 R0	STAGE 3 Rt0 Rt1 R0 TLB2	STAGE 4 Rt0 Rt1 Rt2	RP1output latches Rt0 Rt1
RUN 1	11	L0,0 H0,0				----- -----
	13	L0,1 H0,1	L0,0 ---- H0,0			----- -----
	15	L2,0 H2,0 H0,1	L0,1 ---- H0,1 H0,0	X0,0 --- ----		----- -----
	17	L2,1 H2,1	L2,0 ---- H2,0 -----	X0,2 H0,0 X0,0	X0,0 ---- ----	----- -----
	19	L4,0 H4,0 H2,1	L2,1 ---- H2,1 H2,0	X2,0 ----- X0,2 X0,2	X0,2 H0,0 X0,0	----- -----

	21	L4,1 H4,1	L4,0 ---- H4,0 -----	X2,2 H2,0 X2,0	X2,0 ----- X0,2	X0,0 X0,1
	23	L6,0 H6,0 H4,1	L4,1 ---- H4,1 H4,0	X4,0 ----- X2,2 X2,2	X2,2 H2,0 X2,0	X0,2 -----
	25	L6,1 H6,1	L6,0 ---- H6,0 -----	X4,2 H4,0 X4,0	X4,0 ----- X2,2	X2,0 X2,1
RUN 2	27	----- ----- H6,1	L6,1 ---- H6,1 H6,0	X6,0 ----- X4,2 X4,2	X4,2 H4,0 X4,0	X2,2 -----
	29	L0,2 H0,2 ---- -	----- ----- -----	X6,2 H6,0 X6,0	X6,0 ----- X4,2	X4,0 X4,1
	31	L2,2 H2,2 H0,2	L0,2 H0,1 H0,2 -----	---- ----- X6,2 X6,2	X6,2 H6,0 X6,0	X4,2 -----
	33	L4,2 H4,2 H2,2	L2,2 H2,1 H2,2 -----	X0,4 H0,1 ----- X6,2	---- ---- X6,2	X6,0 X6,1
	35	L6,2 H6,2 H4,2	L4,2 H4,1 H4,2 -----	X2,4 H2,1 X0,4 X0,4	X0,4 H0,1 X0,2	X6,2 -----
	37	---- ----- H6,2	L6,2 H6,1 H6,2 -----	X4,4 H4,1 ----- X2,4	X2,4 H2,1 X2,2	X0,3 X0,4
	39	---- -----	----- ----- ------ - -----	X6,4 H6,1 ----- X4,4	X4,4 H4,1 X4,2	X2,3 X2,4
	41	---- -----	----- ----- ------ - -----	------ ----- ----- X6,4	X6,4 H6,1 X6,2	X4,3 X4,4
	43	---- -----	----- ----- ------ - -----	----- ----- ------ - -----	----- ----- ------	X6,3 X6,4

Table 2. Dataflow of the 5/3 RP1 (Fig. 9a)

In Fig. 9 (a), the control signal, s, of the two multiplexers' labeled mux is set 1 during run 1 to pass R0 of both stages 2 and 3, whereas in all other runs, it is set 0 to pass coefficients stored in TLB1 and TLB2.

4.3 Proposed 4-parallel external architecture

To further increase speed of computations twice as that of the 2-parallel architecture, the 2-parallel architecture is extended to 4-parallel architecture as shown in Fig. 10 (a). This architecture is valid for 5/3, 9/7, and combined 5/3 and 9/7. It consists of 4 *k*-stage pipelined CPs and 4 *k*-stage pipelined RPs. The waveforms of the 3 clocks f_4, f_{4a}, and f_{4b} used in the architecture are shown in Fig. 10 (b). The frequency of clock f_4 is determined from Eq(3) as

$$f_4 = 4k/t_p \tag{5}$$

The architecture scans the external memory with frequency f_4 and it operates with frequency f_{4a} and f_{4b}. Every time clock f_{4a} makes a negative transition CP1 loads into its input latches Rt0 and Rt1 two new coefficients scanned from external memory through the buses labeled *bus0* and *bus1*, whereas CP3 loads every time clock f_{4a} makes a positive transition. CP2 and CP4 loads every time clock f_{4b} makes a negative and a positive transition, respectively, as indicated in Fig. 10 (b). On the other hand, both RP1 and RP2 load simultaneously new data into their input latches Rt0 and Rt1 each time clock f_{4a} makes a negative transition, whereas RP3 and RP4 loads each time clock f_{4b} makes a negative transition.

The dataflow for 4-parallel 5/3 external architecture is given in Table 3, where CPs and RPs are assumed to be 3- and 4-stage pipelined processors, respectively. The dataflow table for

4-parallel 9/7 external architecture is similar in all runs to the 5/3 dataflow except in the first run, where RPs of the 9/7 architecture, specifically RP3 and RP4 generate a pattern of output coefficients different from that of the 5/3. RP3 and RP4 of the 9/7 architecture would generate every clock cycle, reference to clock f_{4b}, two output coefficients as follows. Suppose, at cycle number n the first two coefficients X(0,0) and X(1,0) generated by RP3 and RP4, respectively, are loaded into output latch Rt0 of both processors. Then, in cycle n+1, RP3 and RP4 generate coefficients X(2,0) and X(3,0) followed by coefficients X(4,0) and X(5,0) in cycle n+1 and so on. Note that these output coefficients are the coefficients generated by RP1 and RP2 in Table 3.

The strategy used for scheduling memory columns for CPs of the 5/3 and 9/7 4-parallel architecture, which resemble the one adopted for 2-parallel architecture, is as follow. In the first run, both 5/3 and 9/7 4-parallel architecture will be scheduled to execute 4 columns of memory, two from (A) and the other two from (B), both of Fig. 5. Each CP will be assigned to execute one column of memory coefficients as illustrated in the first run of the dataflow shown in Table 3, whereas in all subsequent runs, 2 columns at a time will scheduled for execution by 4 CPs. One column from Fig. 5 (A) will be assigned to both CP1 and CP3, while the other from Fig. 5 (B) will be assigned to both CP2 and CP4 as shown in the second run of Table 3. However, if number of columns in (A) and (B) of Fig. 5 is not equal, then the last run will consist of only one column of (A). In that case, schedule the last column's coefficients in both CP1 and CP3 as shown in the third run of Table 3, since an attempt to execute the last column using 4 CPs would result in more coefficients been generated than that can be handled by the 4 RPs.

On the other hand, scheduling rows coefficients for RPs, which take place according to scan method shown in Fig. 6, can be understood by examining the dataflow shown in Table 3. At cycle 13, CP1 generates its first two output coefficients labeled L0,0 and L1,0, which correspond to locations 0,0 and 1,0 in Fig. 6, respectively. In cycle 14, CP2 generates its first two output coefficients H0,0 and H1,0, which correspond to locations 0,1 and 1,1 in Fig. 6, respectively. In cycle 15, CP3 generate its first two coefficients L0,1 and L1,1, which correspond to locations 0,2 and 1,2 in Fig. 6, respectively. In cycle 16, CP4 generates its first two output coefficients H0,1 and H1,0 which correspond to locations 0,3 and 1,3 in Fig. 6. Note that L0,0, H0,0, L0,1, and H0,1 represents the first 4 coefficients of row 0 in Fig. 6, whereas L1,0, H1,0, LL1,1 and H1,1, represent the first 4 coefficients of row1.

In cycle 17 and 18, the first two rows coefficients are scheduled for RPs as shown in Table 3, while CPs generating coefficients of the next two rows, row2 and row3. Table 3 shows that the first 4 coefficients of row 0 are scheduled for execution by RP1 and RP3, while the first 4 coefficients of row 1 are scheduled for RP2 and RP4. In addition, note that all coefficients generated by CP4, which belong to column 3 in Fig. 6, are required in the second run's computations, according to the DDGs. Therefore, this would require inclusion of a TLB of size $N/4$ in each of the 4 RPs to store these coefficients. The second run, however, requires these coefficients to be stored in the 4 TLBs in a certain way as follows. Coefficients H0,1 and H1,1 generated by CP4 in cycle 16 should be stored in the first location of TLB of RP1 and RP2, respectively. These two coefficients would be passed to their respective TLB through the input latches of RP1 and RP2 labeled Rt2, as shown in cycle 17 of Table 3, whereas, coefficients H2,1 and H3,1 generated by CP4 at cycle 20 should be stored in the first location of TLB of RP3 and RP4, respectively. These two coefficients are passed to their respective TLB for storage through the input latches of RP3 and RP4 labeled Rt1, as shown in cycle 22

of Table 3. Similarly, coefficients H4,1 and H5,1 generated by CP4 at cycle 24 should be stored in the second location of TLB of RP1 and RP2, respectively, and so on. Note that these TLBs are labeled TLB1 in Fig. 12.

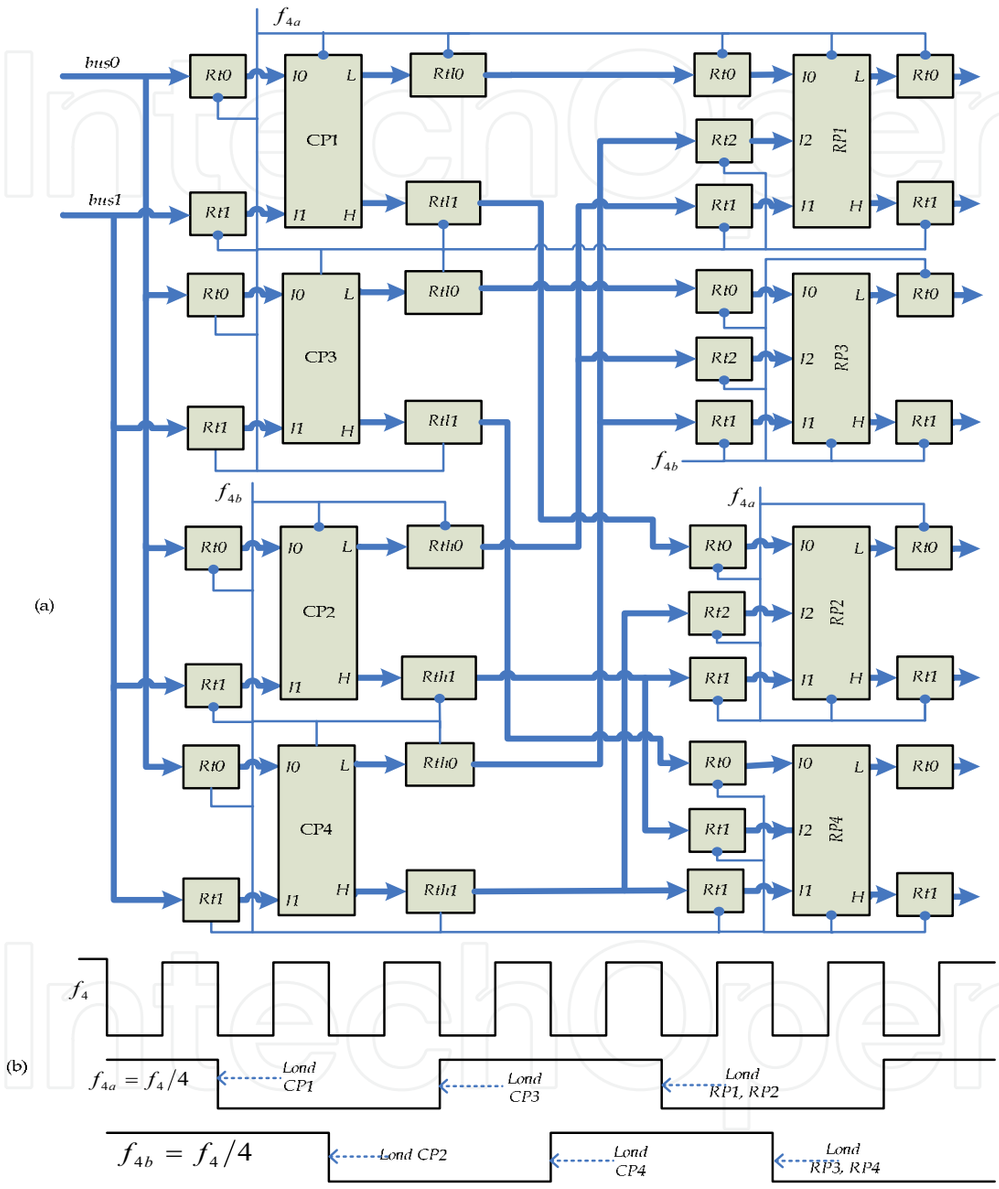


Fig. 10. (a) Proposed 2-D IDWT 4-parallel pipelined external architecture for 5/3 and 9/7 and combined (b) Waveforms of the clocks

	CK f_4	CP	CPs input Latches Rt0 Rt1	CPs 1 &3 Out latches Rtl0 Rtl1	CPs 2 & 4 Out latches Rth0 Rth1	RPs 1 & 3 input latches RP Rt0 Rt1 Rt2	RPs 2 & 4 input latches RP Rt0 Rt1 Rt2	RPs 1 & 3 Out latches Rt0 Rt1	RPs 2 & 4 Out latches Rt0 Rt1
RUN 1	1	1	LL0,0 LH0,0						
	2	2	HL0,0 HH0,0						
	3	3	LL0,1 LH0,1						
	4	4	HL0,1 HH0,1						
	5	1	LL1,0 LH1,0						
	6	2	HL1,0 HH1,0						
	7	3	LL1,1 LH1,1						
	8	4	HL1,1 HH1,1						
	9	1	LL2,0 LH2,0						
	10	2	HL2,0 HH2,0						
	11	3	LL2,1 LH2,1						
	12	4	HL2,1 HH2,1						
	13	1	LL3,0 LH3,0	L0,0 L1,0					
	14	2	HL3,0 HH3,0		H0,0 H1,0				
	15	3	LL3,1 LH3,1	L0,1 L1,1					
	16	4	HL3,1 HH3,1		H0,1 H1,1				
	17	1	LL4,0 ------	L2,0 L3,0		1 L0,0 H0,0 H0,1	2 L1,0 H1,0 H1,1		
	18	2	HL4,0 ------		H2,0 H3,0	3 L0,1 H0,1 H0,0	4 L1,1 H1,1 H1,0		
	19	3	LL4,1 ------	L2,1 L3,1					
	20	4	HL4,1 ------		H2,1 H3,1				
RUN 2	21	1	LL0,2 LH0,2	L4,0 L5,0		1 L2,0 H2,0 -----	2 L3,0 H3,0 -----		
	22	2	HL0,2 HH0,2		H4,0 H5,0	3 L2,1 H2,1 H2,0	4 L3,1 H3,1 H3,0		
	23	3	LL1,2	L4,1					

			LH1,2	L5,1					
	24	4	HL1,2 HH1,2		H4,1 H5,1				
	25	1	LL2,2 LH2,2	L6,0 L7,0		1 L4,0 H4,0 H4,1	2 L5,0 H5,0 H5,1		
	26	2	HL2,2 HH2,2		H6,0 H7,0	3 L4,1 H4,1 H4,0	4 L5,1 H5,1 H5,0		
	27	3	LL3,2 LH3,2	L6,1 L7,1					
	28	4	HL3,2 HH3,2		H6,1 H7,1				
	29	1	LL4,2 --- ---	L8,0 ---- -		1 L6,0 H6,0 -----	2 L7,0 H7,0 -----		
	30	2	HL4,2 -- ----		H8,0 --- --	3 L6,1 H6,1 H6,0	4 L7,1 H7,1 H7,0		
	31	3	------- --- ----	L8,1 ---- -					
	32	4	------- --- ----		H8,1 --- --				
RUN 3	33	1	LL0,3 LH0,3	L0,2 L1,2		1 L8,0 H8,0 H8,1	2 ----- --- - ----	X0,0 --- -	X1,0 --- --
	34	2	----- -- -----		H0,2 H1,2	3 L8,1 H8,1 H8,0	4 ----- --- - ----	X0,1 X0,2	X1,1 X1,2
	35	3	LL1,3 LH1,3	L2,2 L3,2					
	36	4	------ --- ----		H2,2 H3,2				
	37	1	LL2,3 LH2,3	L4,2 L5,2		1 L0,2 H0,2 -----	2 L1,2 H1,2 -----	X2,0 --- -	X3,0 --- --
	38	2	------- --- ----		H4,2 H5,2	3 L2,2 H2,2 -----	4 L3,2 H3,2 -----	X2,1 X2,2	X3,1 X3,2
	39	3	LL3,3 LH3,3	L6,2 L7,2					
	40	4	------- --- ----		H6,2 H7,2				
	41	1	LL4,3 --- ---	L8,2 ---- -		1 L4,2 H4,2 -----	2 L5,2 H5,2 -----	X4,0 --- -	X5,0 --- --
	42	2	------- --- ---		H8,2 --- --	3 L6,2 H6,2 -----	4 L7,2 H7,2 -----	X4,1 X4,2	X5,1 X5,2
	43	3	------- --- ----	----- --- --					
	44	4	------- --- ----		------ --- ---				
	45	1		L0,3 L1,3		1 L8,2 H8,2 -----	2 ----- --- - ----	X6,0 --- -	X7,0 --- --
	46	2			------ -- ----	3 ----- ---- - -----	4 ----- --- - ----	X6,1 X6,2	X7,1 X7,2
	47	3		L2,3 L3,3					
	48	4			------ -- ----				

	49	1		L4,3 L5,3		1 L0,3 ---- - -----	2 L1,3 --- -- -----	X8,0 --- -	----- --- --
	50	2			------ -- ----	3 L2,3 --- -- -----	4 L3,3 --- -- -----	X8,1 X8,2	----- --- --
	51	3		L6,3 L7,3					
	52	4			------ -- ----				
	53	1		L8,3 ---- -		1 L4,3 --- -- -----	2 L5,3 --- -- -----	X0,3 X0,4	X1,3 X1,4
	54	2			------ -- ----	3 L6,3 --- -- -----	4 L7,3 --- -- -----	X2,3 X2,4	X3,3 X3,4
	55	3		------ --- ---					
	56	4			----- --- ---				

Table 3. Dataflow for 4-parallel 5/3 architecture

At cycle 33, RP1 and RP2 yield their first output coefficients X0,0 and X1,0, respectively, which must be stored in the external memory locations 0,0 and 1,0, respectively. Note that indexes of each output coefficient indicate external memory location where the coefficient should be stored.

The second run, which requires scheduling two columns for execution by CPs, starts at cycle 21. In cycle 33, it generates its first two output coefficients L0,2 and L1,2, which belong to locations 0,4 and 1,4 in Fig. 6, respectively. In cycle 34, CP2 generates coefficients H0,2 and H1,2 which belong to locations 0,5 and 1,5 in Fig. 6. In cycle 35, CP3 generates coefficients L2,2 and L3,2, which belong to locations 2,4 and 3,4 in Fig. 6, whereas in cycle 36, CP4 generates coefficients H2,2 and H3,2 that belong to locations 2,5 and 3,5 in Fig. 6. From the above description it is clear that these 8 coefficients are distributed along 4 rows, 0 to 3 in Fig. 6 with each row having 2 coefficients. Table 3 shows that in cycle 37, the two coefficients of row 0, L0,2 and H0,2, and the two coefficients of row 1, L1,2 and H1,2 are scheduled for RP1 and RP2, respectively, while coefficients L4,2 and L5,2 generated by CP1 during the cycle are loaded into Rtl0 and Rtl1, respectively. In cycle 38, the two coefficients of row 2, L2,2 and H2,2, and that of row 3 L3,2 and H3,2 are scheduled for RP3 and RP4, respectively, while coefficients H4,2 and H5,2 generated by CP2 during the cycle are loaded into Rth0 and Rth1, respectively. In cycle 53, RP1 and RP2 generate the first output coefficients of the second run.

4.4 Column and row processors for 5/3 and 9/7 4-parallel external architecture

The 5/3 and the 9/7 processors datapath architectures proposed in (Ibrahim & Herman, 2008) were developed assuming the processors scan external memory either row by row or column by column. However, CPs and RPs of the 4-parallel architecture are required to scan external memory according to scan methods shown in Figs. 5 and 6, respectively. The 4-parallel architecture, in addition, introduces the requirement for interactions among the processors in order to accomplish their task. Therefore, the processors datapath architectures proposed in (Ibrahim & Herman, 2008) should be modified based on the scan methods and taking into consideration also the requirement for interactions so that they fit

into the 4-parallel's processors. Thus, in the following, the modified CPs will be developed first followed by RPs.

4.5 Modified CPs for 4-parallel architecture

The 4 CPs of the 4-parallel architecture each is required in the first run to execute one column at a time. That means the first run requires no modifications of the 5/3 and 9/7 datapath architectures proposed in (Ibrahim & Herman, 2008). However, in all subsequent runs, each two processors (CP1 and CP3 or CP2 and CP4) are assigned to execute one column together, which requires interactions between the two processors to accomplish the required task. Therefore, both CPs 1 and 3, similarly, CPs 2 and 4 should be modified as shown in Fig. 11 to allow interactions to take place. The two processors communicate or interact through the paths (buses) labeled *P1, P2, P3,* and *P4*. Fig. 11 shows modified 5/3 CPs 1 and 3 which is identical to CPs 2 and 4. Fig. 11 also represents the first 3 stages of both 9/7 CPs 1 and 3 and CPs 2 and 4 and the remaining stages are identical to stages 1 to 3. Note that since the first 3 stages of 5/3 and 9/3 are similar in structure, the 5/3 processor can be easily incorporated into 9/7 processor to obtain the combined 5/3 and 9/7 processor for 4-parallel architecture.

The control signal, *s* of the 4 multiplexers, labeled *mux* is set 0 in the first run to allow each processor to execute one column and 1 in all other runs to allow execution of one column by two processors.

In the following, the second run's dataflow of Fig. 11, which starts at cycle 21 in Table 3 will be described. In cycle 21, two coefficients LL0,2 and LH0,2, which correspond to the first two coefficients of the third column in Fig. 5 (A), are read from external memory and are loaded into CP1 first stage latches Rt0 and Rt1, respectively, by negative transition of clock f_{4a} to initiate the first operation. In cycle 23, the third and the fourth coefficients LL1,2 and LH1,2 of the third column are scanned from external memory and are loaded along with Rt1 in stage1 of CP1 into CP3 first stage latches Rt0, Rt1 and Rd3, respectively, by the positive transition of clock f_{4a}. Note that content of Rt1 in stage 1 of CP1 takes the path labeled *P*1 to Rd3.

In cycle 25, coefficients LL2,2 and LH2,2 are scanned from external memory and are loaded along with Rt1 in stage 1 of CP3 into CP1 first stage latches Rt0, Rt1, and Rd1, respectively, by the negative transition of clock f_{4a}, while coefficient X(0) calculated in stage 1 of CP1 and content of Rt1 are loaded into Rt0 and Rt1 of stage 2, respectively.

In cycle 27, coefficients LL3,2 and LH3,2 are scanned from external memory and are loaded, along with Rt1 in stage 1 of CP1, into CP3 first stage latches Rt0, Rt1, and Rd3, respectively, by the positive transition of clock f_{4a}, while coefficients X(2) calculated in stage 1 and content of Rt1 are transferred to Rt0 and Rt1 of the next stage, respectively.

In cycle 29, only one coefficient LL4,2 is scanned from external memory, which implies the third column of Fig. 5 (A) contains odd number of coefficients, and is loaded along with Rt1 in stage1 of CP3 into CP1 first stage latches Rt0 and Rd1, respectively, by the negative transition of clock f_{4a}, while coefficient X(4) calculated in stage 1 and content of Rt1 are transferred to Rt0 and Rt1 of stage 2 and that of Rt0 and Rt1 including Rt0 in stage 2 of CP3 to Rt0, Rt1, and Rt2 in stage 3 of CP1, respectively, to compute the first high coefficient labeled X(1) in 5/3 DDGs. Note that CP1 latches Rt0 and Rt2 of stage 3 now contain coefficients X(0) and X(2), while Rt1 contains coefficient LH0,2 (or Y(1) as labeled in the 5/3

DDGs). These 3 coefficients are required according to the DDGs to compute the high coefficient, X(1).

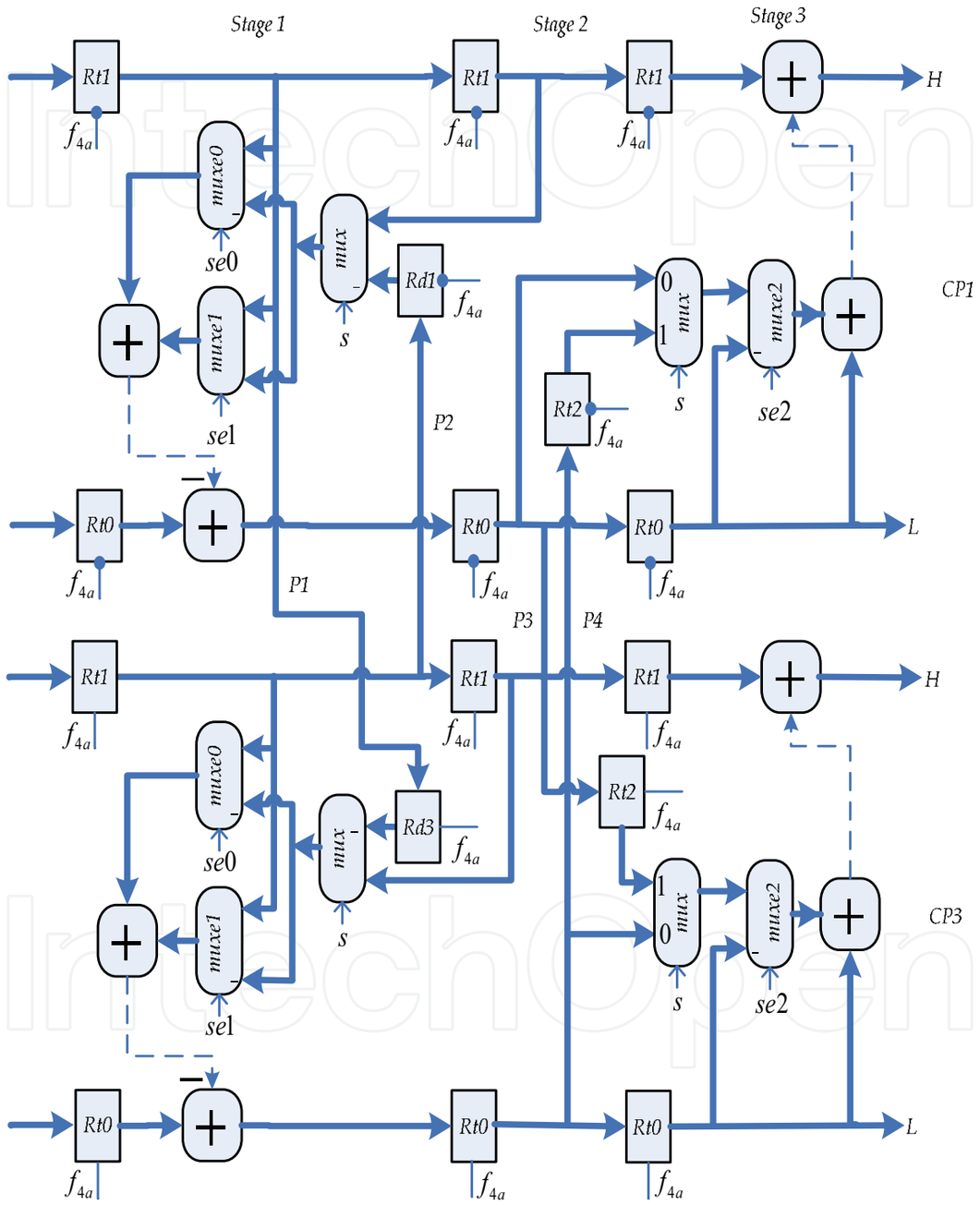


Fig. 11. Modified 5/3 CPs 1 & 3 for 4-parallel architecture

In cycle 31, the positive transition of clock f_{4a} transfers Rt0 and Rt1 in stage 2 of CP3 and Rt0 in stage 2 of CP1 to stage 3 latches Rt0, Rt1, and Rt2 of CP3, respectively, to compute the second high coefficient, X(3). While coefficient X(6) calculated in stage 1 of CP3 and coefficient in Rt1 are loaded into Rt0 and Rt1 of stage 2. As indicated in cycle 31 of Table 3, no coefficients are loaded into CP3 first stage latches.
In cycle 33, the negative transition of clock f_{4a} loads the first high coefficient, X(1) calculated in stage 3 of CP1 and Rt0, which contains X(0), into CP1 output latches Rt0 and Rt1, respectively. Coefficients X(0) and X(1) are labeled L0,2 and L1,2 in Table 3. The same negative transition of clock f_{4a} transfers contents of Rt0 and Rt1 in stage 2 of CP1 and Rt0 in stage 2 of CP3 to Rt0, Rt1, and Rt2 in stage 3 of CP1, respectively, to compute the third coefficient, X(3) and so on.

4.6 Modified RPs for 4-parallell architecture

In section 5.2, it has been pointed out the reasons for including TLBs in the two RPs of the 2-parallel architecture. For the same reasons, it is also necessary to include TLBs in the 4 RPs of the 4-parallel architecture, as shown in Figs. 12 (a) and (a,b) for 5/3 and 9/7, respectively. The processor datapath for both RP1 and RP3, which is also identical to the processor datapath of both RP2 and RP4, are drawn together in Figs.12 (a) and (a,b) for 5/3 and 9/7, respectively, because in the first run, both processors are required to execute together the 4 coefficients of each row. Which implies interactions between the two processors during the computations and that take place through the paths (buses) labeled *P1, P2, P3,* and *P4*. However, in all subsequent runs, according to the scan method shown in Fig. 6, each RP will be scheduled to execute each clock cycle two coefficients of a row as shown in cycles 37 and 38 of Table 3. The advantage of this organization is that the total size of the TLBs does not increase from that of the single pipelined architecture, when it is extended to 2- and 4-parallel architecture.
In the first run, all TLBs in Fig. 12 will be written only, whereas in all other runs, the same location of a TLB will be read in the first half cycle and written in the second half cycle with respect to clock f_{4a} or f_{4b}.
The control signal, *s* of the six multiplexers, labeled *mux* in Fig. 12, is set 0 in the first run to allow in the RP1, coefficient coming through path 0 of each multiplexer to be stored in its respective TLB, whereas in the RP3, it allows contents of Rt2 and Rd1 in stages 1 and 3, respectively, to be passed to the next stage. In all subsequent runs, *s* is set 1 to allow in the RP1, coefficient coming through path 1 of each multiplexer to be stored in the TLB, whereas in the RP3, it passes coefficients read from TLB1 and TLB2 to next stage.

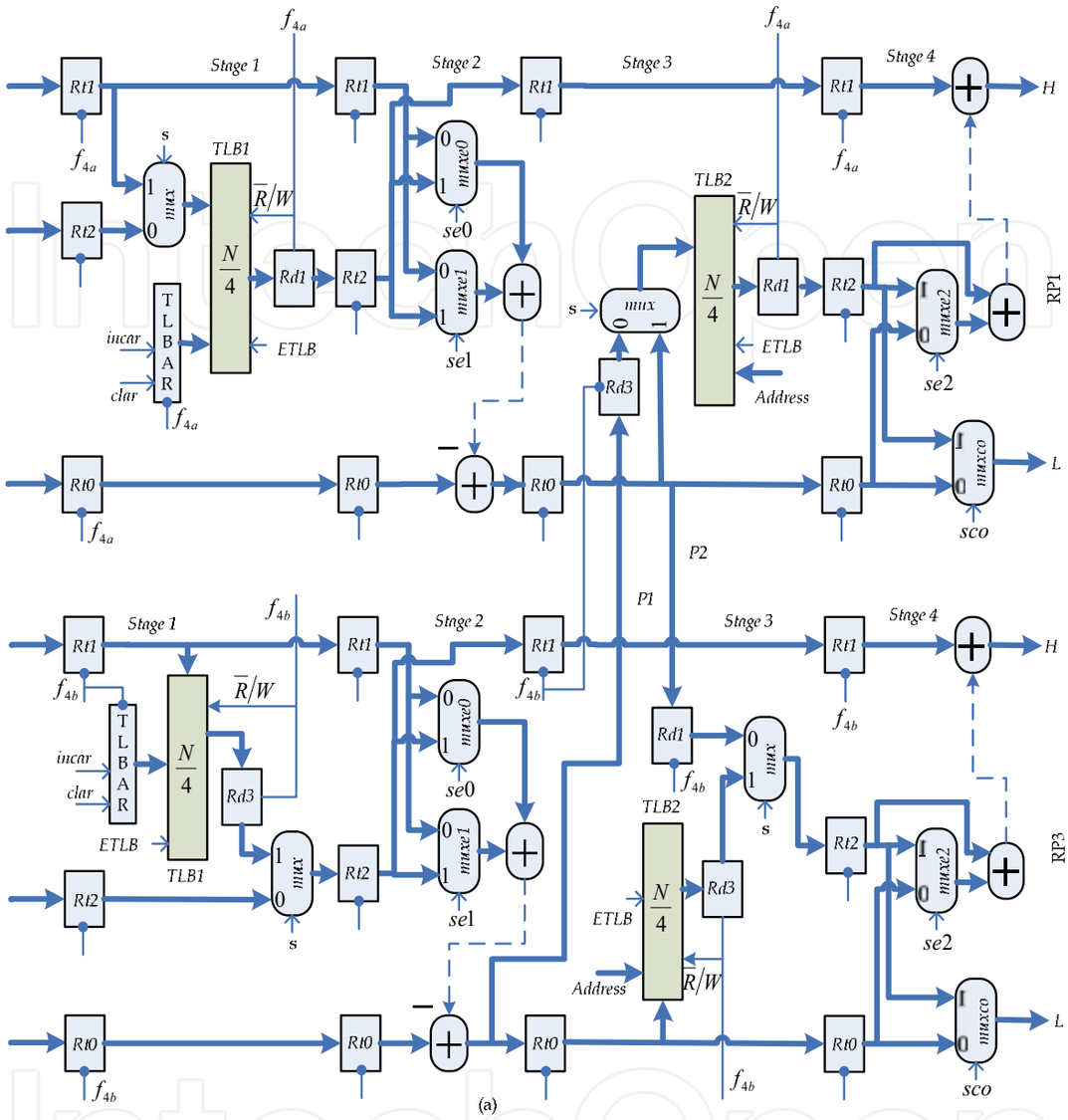


Fig. 12. (a) Modified 5/3 RPs 1 and 3 for 4-parallel External Architecture

In the following, the dataflow of the processor datapath architecture shown in Fig. 12 (a) will described in details starting from cycle 17 in Table 3. The detailed descriptions will enable us to fully understand the behavior of the processor. In cycle 17, the negative transition of clock f_{4a} (consider it the first cycle of f_{4a}) loads coefficients L0,0 and H0,0, which represent the first two coefficients of row 0 in Fig. 6, and H0,1 into RP1 first stage latches Rt0, Rt1, and Rt2, respectively. During the positive (high) pulse of clock f_{4a}, coefficient in Rt2 is stored in the first location of TLB1.

In cycle 18, Table 3 shows that the negative transition of clock f_{4b} (consider it the first cycle of f_{4b}) loads coefficients L0,1, H0,1, and H0,0 of row 0 into RP3 first stage latches Rt0, Rt1, and Rt2,

respectively. In cycle 21, the negative transition of the second cycle of clock f_{4a} transfers contents of RP1 latches Rt0 and Rt1 of the first stage to stage 2 latches Rt0 and Rt1, respectively, to compute the first low coefficient, X0,0, while loading two new coefficients L2,0 and H2,0, which are the first two coefficients of row 2 in Fig. 6, into RP1 first stage latches Rt0 and Rt1, respectively.

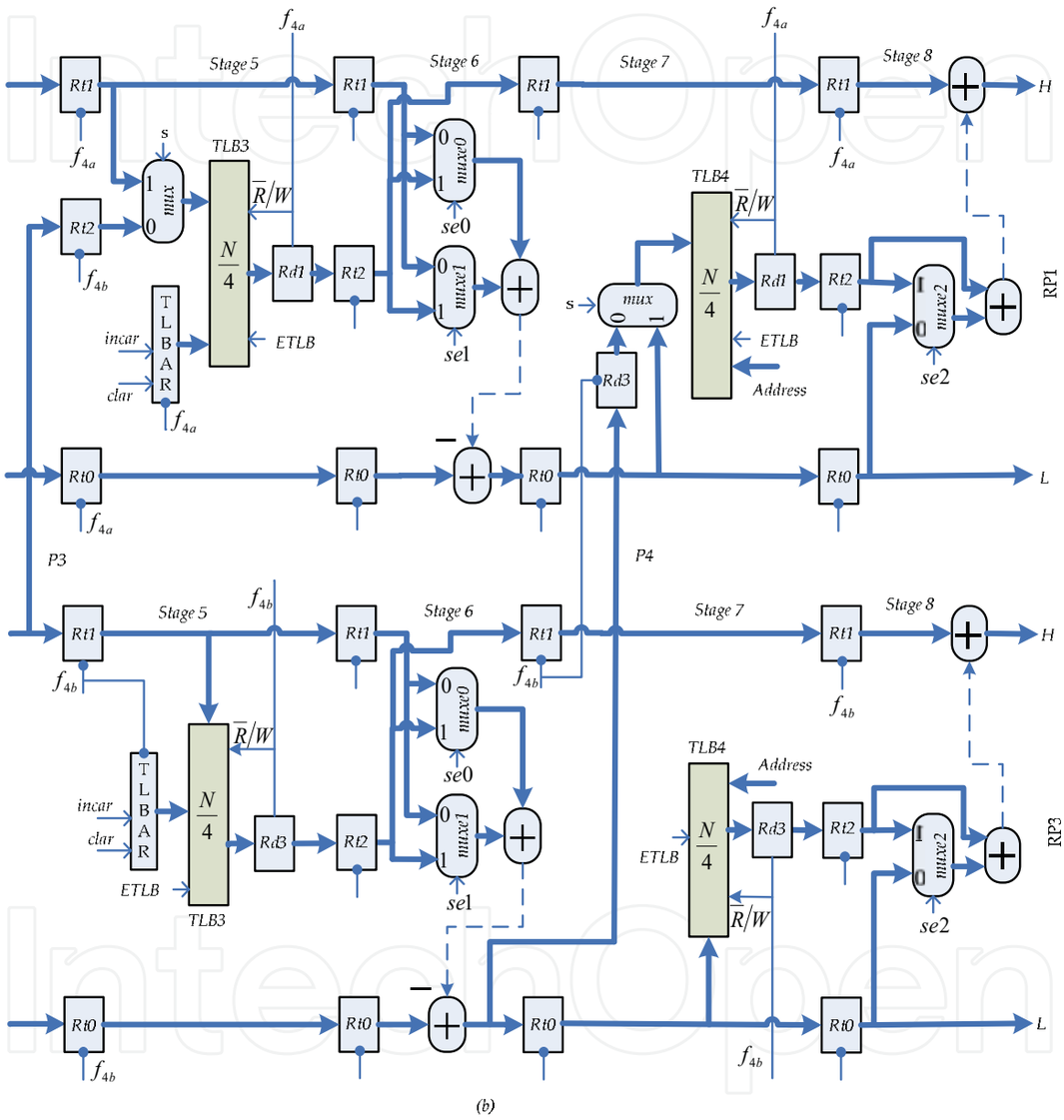


Fig. 12. (b) Modified 9/7 RPs 1 and 3 for 4-parallel External Architecture

During the second cycle of clock f_{4a} no data are stored in TLB1 of RP1.
In cycle 22, the negative transition of the second cycle of clock f_{4b} transfers contents of RP3 latches Rt0, Rt1 and Rt2 of the first stage to stage 2 latches Rt0, Rt1 and Rt2 to compute the second low coefficient of row 0, X0,2, while loading two new coefficients L2,1, H2,1, and

H2,0 into RP3 first stage latches Rt0, Rt1 and Rt2, respectively. During the second cycle of clock f_{4b}, the positive pulse stores content of Rt1 of the first stage in the first location of TLB1 of RP3.

In cycle 25, the negative transition of the third cycle of clock f_{4a} loads coefficient X0,0 computed in stage 2 of RP1, into Rt0 of stage 3 and transfers contents of Rt1 and Rt0 of stage 1 into Rt1 and Rt0 of stage 2 in order to compute the first low coefficient, X2,0 of row 2 labeled X(0) in the 5/3 DDGs, while loading new coefficients L4,0, H4,0, and H4,1 of row 4 in Fig. 6, into RP1 first stage latches Rt0, Rt1 and Rt2, respectively. During the third cycle of clock f_{4a}, the positive pulse stores Rt2 of the first stage in the second location of TLB1 of RP1.

In cycle 26, the negative transition of the third cycle of clock f_{4b} loads coefficient X0,2 calculated in stage 2 of RP3 into both Rt0 in stage 3 of RP3 and Rd3 in stage 3 of RP1, while content of Rt2 in stage 2 of RP3 is transferred to Rt1 of stage 3 and that of Rt0 in stage 3 of RP1 to Rd1 in stage 3 of RP3. The same negative transition of clock f_{4b} also transfers Rt0, Rt1, and Rt2 of stage 1 into Rt0, Rt1, and Rt2 of stage 2, respectively, to compute the second low coefficient, X2,2 of row 2, while loading new coefficients L4,1, H4,1, and H4,0 of row 4 into RP3 first stage latches Rt0, Rt1, and Rt2, respectively. During the third cycle of clock f_{4b}, coefficient in Rt1 is not stored in TLB1 of RP3. It is important to note that Rd3 in stage 3 of RP1, which holds coefficient X0,2, will be stored in the first location of TLB2 by the positive pulse of the third cycle of clock f_{4a}.

In cycle 29, the fourth cycle's negative transition of clock f_{4a} transfers Rt0 in stage 3 of RP1, which contains X0,0, to Rt0 of stage 4, while X2,0 calculated in stage 2 is loaded into Rt0 of stage 3. The same negative transition of clock f_{4a} transfer Rt1 and Rt0 of stage 1 to Rt1 and Rt0 of stage 2, respectively, to compute the first low coefficient of row 4, X4,0, and loads new coefficients L6,0 and H6,0 into Rt0 and Rt1 of stage 1, respectively. During the fourth cycle of clock f_{4a}, coefficient in Rt2 of stage 1 is not stored in TLB1 of RP1.

In cycle 30,the negative transition of the fourth cycle of clock f_{4b} transfers contents of Rt0, Rt1, and Rd1 in stage 3 of RP3 to Rt0, Rt1, and Rt2 of the next stage, respectively, to compute the first high coefficient, X0,1 of row 0. While coefficient X2,2 calculated in stage 2 of RP3 is transferred to both Rt0 in stage3 of RP3 and Rd3 in stage 3 of RP1 and coefficient X2,0 in Rt0, in stage 3 of RP1, is loaded into Rd1 in stage 3 of RP3, whereas Rt2 of stage 2 is transferred to Rt1 in stage 3 of RP3. The same negative transition of clock f_{4b} transfers contents of Rt0, Rt1, and Rt2 of stage 1 to Rt0, Rt1, and Rt2 of stage 2 to compute the second low coefficient, X4,0 of row 4, while loading new coefficients L6,1, H6,1, and H6,0 of row 6 into RP3 first stage latches Rt0, Rt1, and Rt2, respectively. During the fourth cycle's positive pulse of clock f_{4b}, content of Rt1 in stage 1 of RP3 will be stored in the second location of TLB1 and content of Rt0, X2,2, in stage 3 of RP3 will be stored in the first location of TLB2, while content of Rd3, X2,2, in stage 3 of RP1 will not be stored in TLB2 of RP1.

In cycle 33, the negative transition of the fifth cycle of clock f_{4a} transfers content of Rt0, X0,0, in stage 4 of RP1 to RP1 output latch Rt0, as first output coefficient and Rt0 of stage 3 holding coefficient X2,0 to Rt0 of stage 4, while coefficient X4,0 calculated in stage 2 is loaded into Rt0 of stage 3. The same negative transition of clock f_{4a} transfers contents of Rt0 and Rt1 of stage 1 to Rt0 and Rt1 of stage 2 to compute the first low coefficient, X6,0 of row 6, while loading new coefficients L8,0, H8,0, and H8,1 of row 8 into RP1 first stage latches Rt0, Rt1, and Rt2, respectively. During the fifth cycle of clock f_{4a}, the positive pulse of the clock stores content of Rt2 in stage 1 of RP1 into the third location of TLB1.

In cycle 34, the negative transition of the fifth cycle of clock f_{4b} transfers content of Rt0, X0,2, and the high coefficient, X0,1 computed in stage 4 of RP3 to RP3 output latches Rt0 and Rt1, respectively, while loading contents of Rt0, Rt1, and Rd1 of stage 3 into Rt0, Rt1, and Rt2 of stage 4 to compute the first high coefficient of row 2, X2,1. Furthermore, the same negative transition of clock f_{4b} also transfers coefficient X4,2 calculated in stage 2 of RP3 to both Rt0 in stage 3 of RP3 and Rd3 in stage 3 of RP1. It also transfers coefficient X4,0 in Rt0, in stage 3 of RP1, and content of Rt2 in stage 2 of RP3 to stage 3 of RP3 latches Rd1 and Rt1, respectively, and contents of Rt0, Rt1, and Rt2 of stage 1 to Rt0, Rt1, and Rt2 in stage 3 of RP3, while loading new coefficients L8,1, H8,1, and H8,0 into Rt0, Rt1, and Rt2 of stage 1. Content of Rd3, X4,2 in stage 3 of RP1 will be stored in the second location of TLB2 by the positive pulse of the fifth cycle of clock f_{4a}.

In cycle 37, the second run of the RPs begins when the negative transition of the sixth cycle of clock f_{4a} loads two new coefficients L0,2 and H0,2, which are the fifth and sixth coefficients of row 0, into first stage latches Rt0 and Rt1 of RP1, respectively. During the first half (low pulse) of cycle 6, the first location of TLB1 will be read and loaded into Rd1 by the positive transition of clock f_{4a}, whereas during the second half cycle, content of Rt1 will be written in the first location of TLB1.

In cycle 38, the negative transition of the sixth cycle of clock f_{4b} loads two new coefficients L2,2 and H2,2, the fifth and sixth coefficients of row 2, into RP3 first stage latches Rt0 and Rt1, respectively. The first half cycle of clock f_{4b} reads the first location of TLB1 and loads it into Rd3 by the positive transition of the clock, whereas the second half cycle writes content of Rt1 in the same location of TLB1.

In cycle 41, the negative transition of the seventh cycle of clock f_{4a} transfers contents of Rt0, Rt1, and Rd1 of stage 1 to stage 2 of RP1 latches Rt0, Rt1, and Rt2, respectively, to compute the third low coefficient, X0,4 of row 0, while loading two new coefficients L4,2 and H4,2 of row 4 into RP1 first stage latches Rt0 and Rt1 respectively.

Note that during run 2 all RPs execute independently with no interactions among them. In addition, in the first run, if the first coefficient generated by stage 2 of RP3 is stored in TLB2 of RP1, then the second coefficient should be stored in TLB2 of RP3 and so on. Similarly, TLB1, TLB3, and TLB4 of both RP1 and RP3 are handled. Furthermore, during the whole period of run 1, the control signals of the three extension multiplexers labeled *muxe0, muxe1,* and *muxe2* in RP1 should be set 0, according to Table 4 (Ibrahim & Herman, 2008), whereas those in RP3 should be set normal as shown in the second line of Table 4, since RP3 will execute normal computations during the period. However, in the second run and in all subsequent runs except the last run, the extension multiplexers control signals in all RPs are set normal. Moreover, the multiplexers labeled *muxco* in stage 4 is only needed in the combined 5/3 and 9/7 architecture, otherwise, it can be eliminated and Rt2 output can be connected directly to Rt0 input of the next stage in case of 9/7, whereas in 5/3, Rt0 is connected directly to output latch Rt0. In the combined architecture, signal *sco* of *muxco* is set 0 if the architecture is to perform 5/3; otherwise, it is set 1 if the architecture is to perform 9/7.

It is very important to note that when the RP executes its last set of input coefficients, according to 9/7 DDGs for odd and even signals shown in Fig. 4 it will not yield all required output coefficients as expected by the last run. For example, in the DDGs for odd length signals shown in Fig. 4 (a), when the last input coefficient labeled Y8 is applied to RP it will yield output coefficients X5 and X6. To get the last remaining two coefficients X7 and X8, the

RP must execute another run, which will be the last run in order to compute the remaining two output coefficients. Similarly, when the last two input coefficients labeled Y6 and Y7 in the DDG for even length signals shown in Fig. 4 (b) are applied to 9/7 RP it will yield output coefficients X3 and X4. To obtain the remaining output coefficients X5, X6, and X7, two more runs should be executed by RP according to the DDG. The first run will yield X5 and X6, whereas the last run will yield X7. The details of the computations that take place during each of these runs can be determined by examining the specific area of the DDGs.

	se0	*se1*	*se2*
First	0	0	0
Normal	0	1	0
Last	1	1	0

a) Odd length signals

	se0	*se1*	*se2*
First	0	0	0
Normal	0	1	0
Last	0	1	0

b) Even length signals

Table 4. Extension's control signals

5. Performance Evaluation

In order to evaluate performance of the two proposed parallel pipelined architectures in terms of speedup, throughput, and power as compared with single pipelined architecture proposed in (Ibrahim & Herman, 2008) consider the following. Assume subbands HH, HL, LH, and LL of each level are equal in size. The dataflow table for single pipelined architecture (Ibrahim & Herman, 2008) shows that $\rho_1 = 20$ clock cycles are needed to yield the first output coefficient. Then, the total number of output coefficients in the first run of the J[th] level reconstruction can be estimated as

$$N/2^{J-1} \tag{6}$$

and the total number of cycles in the first run is given by

$$2N/2^{J-1} \tag{7}$$

The total time, $T1$, required to yield n pairs of output coefficients for the J[th] level reconstruction on the single pipelined architecture can be estimated as

$$\begin{aligned} T1 &= \left(\rho_1 + 2N/2^{J-1} + 2\left(n - 1/2\, N/2^{J-1}\right)\right)\tau_1 \\ &= \left(\rho_1 + N/2^{J-1} + 2n\right)t_p/2k \end{aligned} \tag{8}$$

On the other hand, the dataflow for 2-parallel pipelined architecture shows that $\rho_2 = 21$ clock cycles are required to yield the first 2 pairs of output coefficients. The total number of paired output coefficients in the first run of the J[th] level reconstruction on the 2-parallel architecture can be estimated as

$$3/2\,N/2^{J-1} \tag{9}$$

and the total number of 2-paired output coefficients is given by

$$3/4\,N/2^{J-1} \tag{10}$$

While, the total number of cycles in the first run is

$$2\,N/2^{J-1} \tag{11}$$

Note that the total number of paired output coefficients of the first run in each level of reconstruction starting from the first level can be written as

$$3/2\,N,\ 3/2\,N/2,\ 3/2\,N/4,\ldots\ldots\ldots\ldots,3/2\,N/2^{J-1}$$

where the last term is Eq (9).

The total time, *T2*, required to yield *n* pairs of output coefficients for the J[th] level reconstruction of an *NxM* image on the 2-parallel architecture can be estimated as

$$T2 = \left(\rho_2 + 2N/2^{J-1} + 2(n/2 - 3/4\,N/2^{J-1})\right)\tau_2 \tag{12}$$

$$T2 = \left(\rho_2 + 1/2\,N/2^{J-1} + n\right)t_p/2k \tag{13}$$

The term $2\left(n/2 - 3/4\,N/2^{J-1}\right)$ in (12) represents the total number of cycles of run 2 and all subsequent runs.

The speedup factor, *S2,* is then given by

$$S2 = \frac{T1}{T2} = \frac{\left(\rho_1 + N/2^{J-1} + 2n\right)t_p/2k}{\left(\rho_2 + 1/2\,N/2^{J-1} + n\right)t_p/2k}$$

For large *n*, the above equation reduces to

$$S2 = \frac{2(1/2\,N/2^{J-1} + n)}{(1/2\,N/2^{J-1} + n)} = 2 \tag{14}$$

Eq(14) implies that the 2-parallel architecture is 2 times faster than the single pipelined architecture.

Similarly, the dataflow for the 4-parallel pipelined architecture shows that $\rho_4 = 33$ clock cycles are needed to yield the first two output coefficients. From the dataflow table of the 4-paralell architecture it can be estimated that both RP1 and RP2, in the first run of the J[th] level reconstruction, yield $(N/2^{J-1})/2$ pairs of output coefficients, whereas both RP3 and RP4 yield $N/2^{J-1}$ pairs of output coefficients, a total of $3/2\,N/2^{J-1}$ pairs of output coefficients in the first run. The total number of cycles in run 1 is then given by

$$4(N/2^{J-1})/2 \tag{15}$$

Thus, the total time, *T4*, required to yield *n* pairs of output coefficients for the J[th] level reconstruction of an *NxM* image on the 4-parallel architecture can be estimated as

$$T4 = \left(\rho_4 + 2N/2^{J-1} + 2\left(n - 3/2\,N/2^{J-1}\right)/2\right)\tau_4$$

$$T4 = \left(\rho_4 + 2N/2^{J-1} + \left(n - 3/2\,N/2^{J-1}\right)\right)t_p/4k \tag{16}$$

$$T4 = \left(\rho_4 + 1/2\,N/2^{J-1} + n\right)t_p/4k \tag{17}$$

The term $(n - 3/2\,N/2^{J-1})$ in (16) represents the total cycles of run 2 and all subsequent runs.

The speedup factor, *S4,* is then given by

$$S4 = \frac{T1}{T4} = \frac{\left(\rho_1 + N/2^{J-1} + 2n\right)t_p/2k}{\left(\rho_4 + 1/2\,N/2^{J-1} + n\right)t_p/4k}$$

For large *n* it reduces to

$$S4 = \frac{4(1/2\,N/2^{J-1} + n)}{(1/2\,N/2^{J-1} + n)} = 4 \tag{18}$$

Thus, the 4-parallel architecture is 4 times faster than the single pipelined architecture.
The throughput, *H,* which can be defined as number of output coefficients generated per unit time, can be written for each architecture as

$$H(\sin gle) = n/(\rho_1 + N/2^{J-1} + 2n)\,t_p/2k$$

The maximum throughput, H^{max}, occurs when *n* is very large ($n\to\infty$), thus,

$$\begin{aligned} H^{\max}(\sin gle) &= H(\sin gle)_{n\to\infty} \\ &= 2nkf_p/2n = kf_p \end{aligned} \tag{19}$$

$$H(2-parallel) = n/(\rho_2 + 1/2\,N/2^{J-1} + n)t_p/2k$$

$$\begin{aligned} H^{\max}(2-parallel) &= H(2-parallel)_{n\to\infty} \\ &= 2knf_p/n = 2kf_p \end{aligned} \tag{20}$$

$$H(4-parallel) = n/(\rho_4 + 1/2\,N/2^{J-1} + n)\,t_p/4k$$

$$\begin{aligned} H^{\max}(4-parallel) &= H(4-parallel)_{n\to\infty} \\ &= 4knf_p/n = 4kf_p \end{aligned} \tag{21}$$

Thus, the throughputs of the 2-parallel and the 4-parallel pipelined architectures have increased by factors of 2 and 4, respectively, as compared with the single pipelined architecture.
On the other hand, the power consumption of *l*-parallel pipelined architecture as compared with the single pipelined architecture can be obtained as follows. Let P_1 and P_l denote the power consumption of the single and *l*-parallel architectures without the external memory, and P_{m1} and P_{ml} denote the power consumption of the external memory for the single and *l*-parallel architectures, respectively. The power consumption of VLSI architecture can be estimated as

$$P = C_{total} \cdot V_0^2 \cdot f$$

where C_{total} denotes the total capacitance of the architecture, V_0 is the supply voltage, and *f* is the clock frequency. Then,

$$P_1 = C_{total} \cdot V_0^2 \cdot f_1/2\,,\quad P_l = l \cdot C_{total} \cdot V_0^2 \cdot f_l/l \qquad \text{and}$$

$$\begin{aligned} \frac{P_l}{P_1} &= \frac{l \cdot C_{total} \cdot V_0^2 \cdot f_l/l}{C_{total} \cdot V_0^2 \cdot f_1/2} = \frac{2f_l}{f_1} \\ &= 2\left(\frac{l \cdot k}{t_p}\right)\Big/2k/t_p = l \end{aligned} \tag{21}$$

While, P_{m1} and P_{ml} can be estimated as

$$P_{m1} = C^m_{total} \cdot V_0^2 \cdot f_1 \quad , \quad P_{ml} = 2 \cdot C^m_{total} \cdot V_0^2 \cdot f_l \, , \quad \text{and} \quad \frac{P_{ml}}{P_{m1}} = \frac{2 \cdot f_l}{f_1} = \frac{2 \cdot l \cdot k/t_p}{2k/t_p} = l \tag{22}$$

C^m_{total}, is the total capacitance of the external memory.

In summary, the above equations indicates that as degree of parallelism increases the speedup and the power consumption of the proposed architectures, without external memory, and the power consumption of the external memory increase by a factor of l, as compared with single pipelined architecture.

6. Comparisons

Table 5 provides comparison results of the proposed architectures with most recent architectures in the literature. The architecture proposed in (Lan & Zheng, 2005) achieves a critical path of one multiplier delay using very large number of pipelined registers, 52 registers. In addition, it requires a total line buffer of size 6N, which is a very expensive memory component, while the proposed architectures require only 4N. In (Rahul & Preeti, 2007), a critical path delay of Tm + Ta is achieved through optimal dataflow graph, but requires a total line buffer of size 10N.

In (wang et at., 2007), by rewriting the lifting-based DWT of the 9/7, the critical path delay of the pipelined architectures have been reduced to one multiplier delay but it requires a total line buffer of size 5.5N. In addition, it requires real floating-point multipliers with long delay that can not be implemented by using arithmetic shift method (Qing & Sheng, 2005). (Qing & Sheng, 2005) has illustrated that the multipliers used for scale factor k and coefficients α, β, γ, and δ of the 9/7 filter can be implemented in hardware using only two adders. Moreover, the fold architecture which uses one module to perform both the predictor and update steps in fact increases the hardware complexity, e.g., use of several multiplexers, and the control complexity. In addition, use of one module to perform both predictor and update steps implies both steps have to be sequenced and that would slow down the computation process.

In the 2-parallel architecture proposed in (Bao & Yong, 2007), writing results generated by CPs into MM (main memory) and then switching them out to external memory for next level decomposition is really a drawback, since external memory in real-time applications, e.g., in digital camera, is actually consist of charge-coupled devices which can only be scanned. In addition, it requires a total line buffer of size 5N for 5/3 and 7N for 9/7 while the proposed architectures require 2N and 4N for 5/3 and 9/7 respectively. The architecture requires also used of several FIFO buffers in the datapath, which are complex and very expensive memory components, while the proposed architectures require no such memory components.

The 2-parallel architecture proposed in (Cheng et al., 2006) requires a total line buffer of size 5.5N and use of two-port RAM to implement FIFOs, whereas, the proposed architectures require only use of single port RAM.

Architecture	Multi	Adders	Line Buff.	Computing time	Critical Path
Lan	12	12	6N	$2(1-4^{-j})NM$	Tm
Rahul	9	16	10N	$2(1-4^{-j})NM$	Tm + Ta
Wang	6	8	5.5N	$2(1-4^{-j})NM$	Tm
Ibrahim	10	16	4N	$2(1-4^{-j})NM$	Tm +2Ta
Cheng (2-parallel)	18	32	5.5N	$(1-4^{-j})NM$	Tm +2Ta
Bao (2-parallel)	24	32	7N	$(1-4^{-j})NM$	Tm +2Ta
Proposed (2-parallel)	18	32	4N	$(1-4^{-j})NM$	Tm +2Ta
Proposed (4-parallel)	36	64	4N	$1/2\ (1-4^{-j})NM$	Tm +2Ta

Tm: multiplier delay Ta: adder delay

Table 5. Comparison results of several 2-D 9/7 architectures

7. Conclusions

In this chapter, two high performance parallel VLSI architectures for 2-D IDWT are proposed that meet high-speed, low-power, and low memory requirements for real-time applications. The two parallel architectures achieve speedup factors of 2 and 4 as compared with single pipelined architecture.

- The advantages of the proposed parallel architectures :

i. Only require a total temporary line buffer (TLB) of size 2N and 4N in 5/3 and 9/7, respectively.
ii. The scan method adopted not only reduces the internal memory between CPs and RPs to a few registers, but also reduces the internal memory or TLB size in the CPs to minimum and allows RPs to work in parallel with CPs earlier during the computation.
iii. The proposed architectures are simple to control and their control algorithms can be immediately developed.

8. References

Bao-Feng, L. & Yong, D. (2007). "FIDP A novel architecture for lifting-based 2D DWT in JPEG2000," MMM (2), lecture note in computer science, vol. 4352, PP. 373-382, Springer, 2007.

Cheng-Yi, X.; Jin-Wen, T. & Jian, L. (2006). "Efficient high-speed/low-power line-based architecture for two-dimensional discrete wavelet transforms using lifting scheme," IEEE Trans. on Circuits & sys. For Video Tech. Vol.16, No. 2, February 2006, PP 309-316.

Dillin, G.; Georis B.; Legant J-D. & Cantineau, O. (2003). "Combined Line-based Architecture for the 5-3 and 9-7 Wavelet Transform of JPEG2000," IEEE Trans. on circuits and systems for video tech., Vol. 13, No. 9, Sep. 2003, PP. 944-950.

Ibrahim saeed koko & Herman Agustiawan, (2008). "Pipelined lifting-based VLSI architecture for two-dimensional inverse discrete wavelet transform," proceedings of the IEEE International Conference on Computer and Electrical Engineering, ICCEE 2008, Phuket Island, Thailand.

Lan, X. & Zheng N. (2005). "Low-Power and High-Speed VLSI Architecturefor Lifting-Based Forward and Inverse Wavelet Transform," IEEE tran. on consumer electronics, Vol. 51, No. 2, May 2005, PP. 379 –385.

Qing-ming Yi & Sheng-Li Xie, (2005). "Arithmetic shift method suitable for VLSI implementation to CDF 9/7 discrete wavelet transform based on lifting scheme," Proceedings of the Fourth Int. Conf. on Machine Learning and `Cybernetics, Guangzhou, August 2005, PP. 5241-5244.

Rahul. J. & Preeti R. (2007). "An efficient pipelined VLSI architecture for Lifting-based 2D-discrete wavelet transform," ISCAS, 2007 IEEE, PP. 1377-1380.

Wang, C.; Wu, Z.; Cao, P. & Li, J. (2007). "An efficient VLSI Architecture for lifting-based discrete wavelet transform," Mulltimedia and Epo, 2007 IEEE International conference, PP. 1575-1578.

IntechOpen



VLSI

Edited by Zhongfeng Wang

ISBN 978-953-307-049-0

Hard cover, 456 pages

Publisher InTech

Published online 01, February, 2010

Published in print edition February, 2010

The process of Integrated Circuits (IC) started its era of VLSI (Very Large Scale Integration) in 1970's when thousands of transistors were integrated into one single chip. Nowadays we are able to integrate more than a billion transistors on a single chip. However, the term "VLSI" is still being used, though there was some effort to coin a new term ULSI (Ultra-Large Scale Integration) for fine distinctions many years ago. VLSI technology has brought tremendous benefits to our everyday life since its occurrence. VLSI circuits are used everywhere, real applications include microprocessors in a personal computer or workstation, chips in a graphic card, digital camera or camcorder, chips in a cell phone or a portable computing device, and embedded processors in an automobile, et al. VLSI covers many phases of design and fabrication of integrated circuits. For a commercial chip design, it involves system definition, VLSI architecture design and optimization, RTL (register transfer language) coding, (pre- and post-synthesis) simulation and verification, synthesis, place and route, timing analyses and timing closure, and multi-step semiconductor device fabrication including wafer processing, die preparation, IC packaging and testing, et al. As the process technology scales down, hundreds or even thousands of millions of transistors are integrated into one single chip. Hence, more and more complicated systems can be integrated into a single chip, the so-called System-on-chip (SoC), which brings to VLSI engineers ever increasingly challenges to master techniques in various phases of VLSI design. For modern SoC design, practical applications are usually speed hungry. For instance, Ethernet standard has evolved from 10Mbps to 10Gbps. Now the specification for 100Mbps Ethernet is on the way. On the other hand, with the popularity of wireless and portable computing devices, low power consumption has become extremely critical. To meet these contradicting requirements, VLSI designers have to perform optimizations at all levels of design. This book is intended to cover a wide range of VLSI design topics. The book can be roughly partitioned into four parts. Part I is mainly focused on algorithmic level and architectural level VLSI design and optimization for image and video signal processing systems. Part II addresses VLSI design optimizations for cryptography and error correction coding. Part III discusses general SoC design techniques as well as other application-specific VLSI design optimizations. The last part will cover generic nano-scale circuit-level design techniques.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Ibrahim Saeed Koko and Herman Agustiawan (2010). High Performance Parallel Pipelined Lifting-based VLSI Architectures for Two-Dimensional Inverse Discrete Wavelet Transform, VLSI, Zhongfeng Wang (Ed.), ISBN: 978-953-307-049-0, InTech, Available from: http://www.intechopen.com/books/vlsi/high-performance-parallel-pipelined-lifting-based-vlsi-architectures-for-two-dimensional-inverse-dis

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen

IntechOpen

© 2010 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike-3.0 License, which permits use, distribution and reproduction for non-commercial purposes, provided the original is properly cited and derivative works building on this content are distributed under the same license.

IntechOpen

IntechOpen