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.

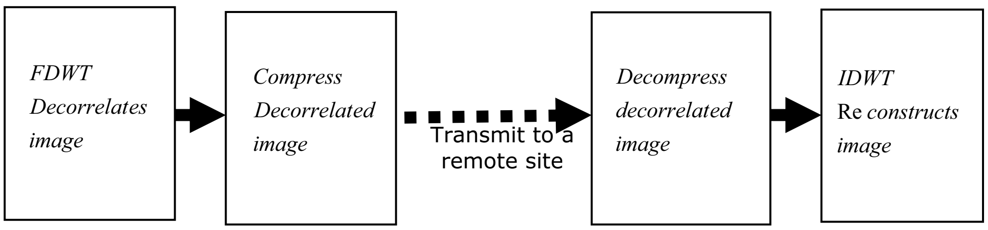

FDWT Decorrelates image
Compress Decorrelated image
Transmit to a remote site
Decompress decorrelated image
IDWT Re constructs image

Fig. 1. A simplified Compression System

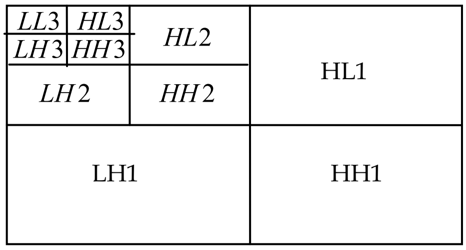


Fig. 2. Subband decomposition of an *NxM* image into 3 levels.

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

$$\begin{aligned} step1: X(2n) &= Y(2n) - \left\lfloor \frac{Y(2n-1)+Y(2n+1)+2}{4} \right\rfloor \\ step2: X(2n+1) &= Y(2n+1) + \left\lfloor \frac{X(2n)+X(2n+2)}{2} \right\rfloor \end{aligned} \tag{1}$$

9/7 synthesis algorithm

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

$$\text{Step 5: } X(2n) = Y''(2n) - \beta(Y''(2n-1) + Y''(2n+1))$$
$$\text{Step 6: } X(2n+1) = Y''(2n+1) - \alpha(X(2n) + X(2n+2))$$

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.

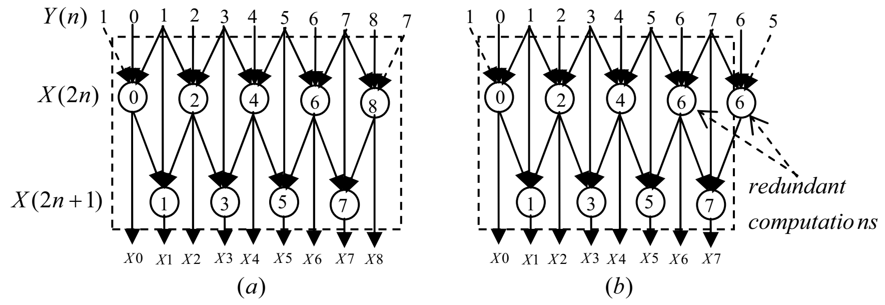


Fig. 3. 5/3 synthesis algorithm's DDGs for (a) odd and (b) even length signals

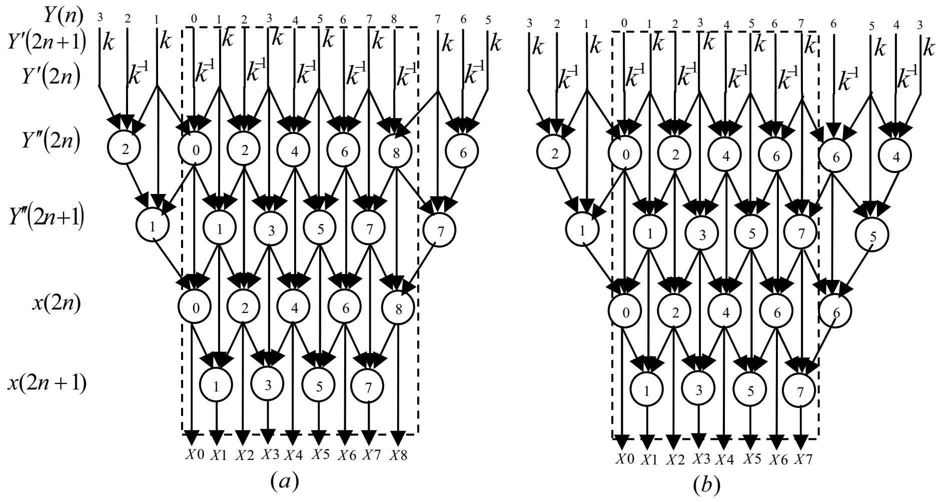


Fig. 4. 9/7 synthesis algorithm's DDGs for (a) odd and (b) even length signals

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.

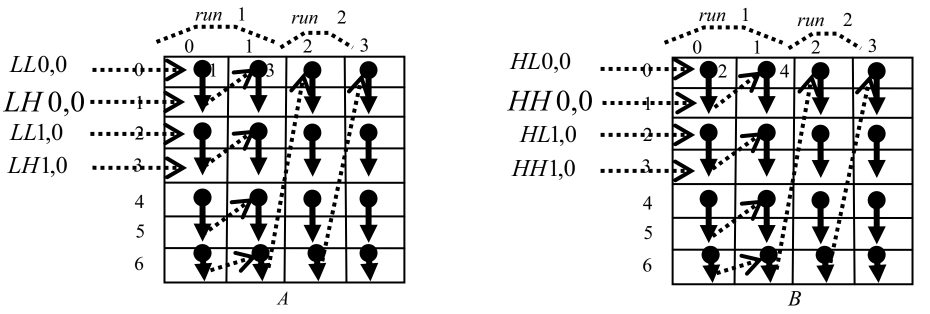


Fig. 5. 5/3 and 9/7 CPs scan method

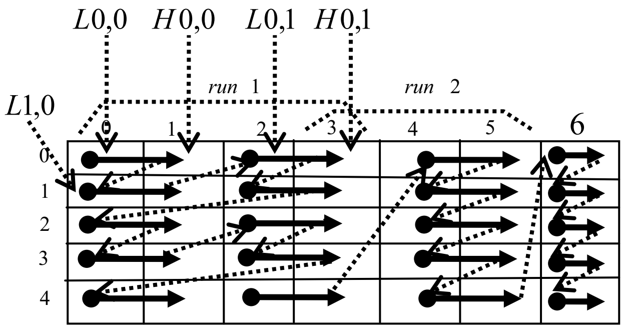


Fig. 6. 5/3 and 9/7 RPs scan method

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).

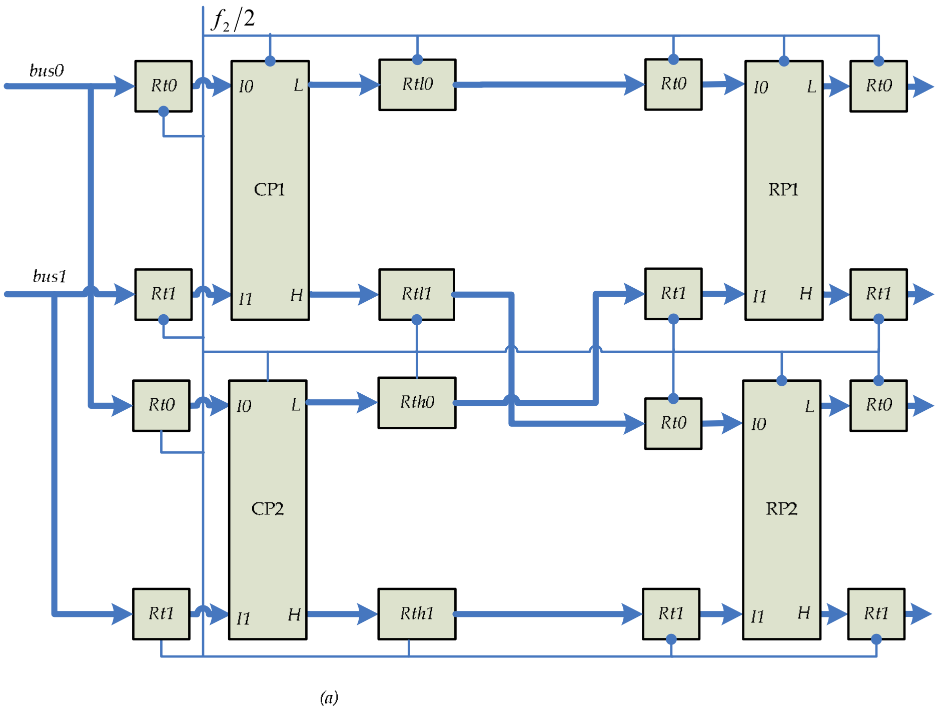
In general, the scan frequency f_l and hence the period $\tau_l = 1/f_l$ of the parallel architectures can be determined by the following algorithm when the required pixels I of an operation are scanned simultaneously in parallel. Suppose t_p and t_m are the processor and the external memory critical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

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

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.



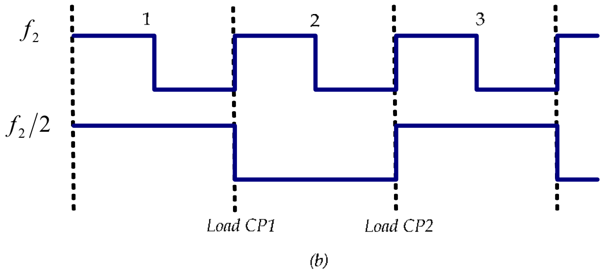


Fig. 7. (a) Proposed 2-parallel pipelined external architecture for 5/3 and 9/7 and combined (b) Waveform of the clocks.

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.

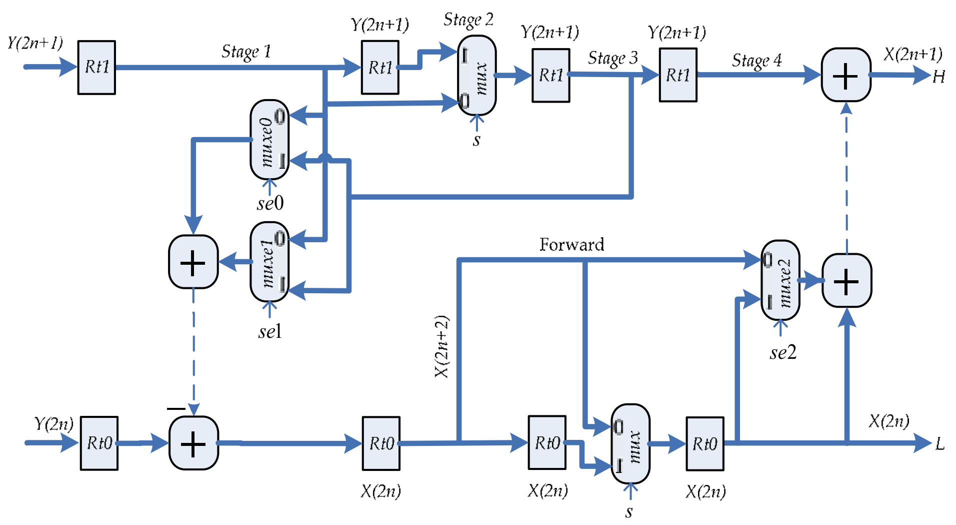


Fig. 8. Modified inverse 5/3 CP for 2-parallel External architecture

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.

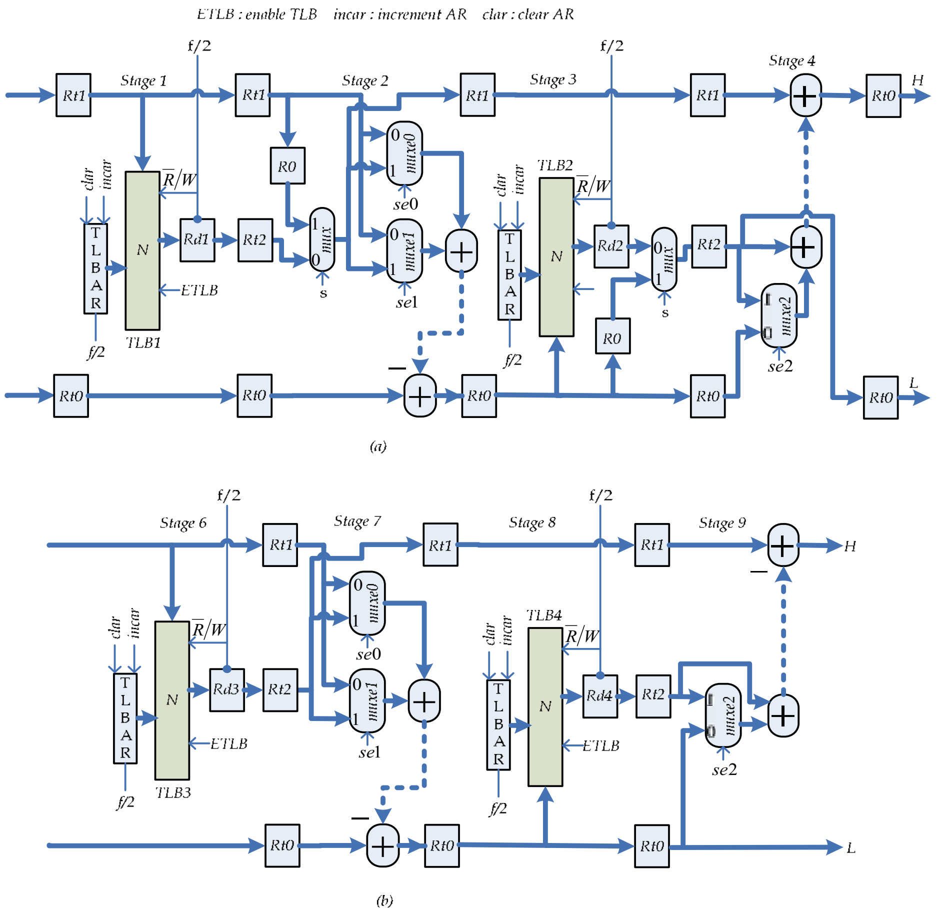


Fig. 9. Modified RP for 2-parallel architecture (a) 5/3 and (a) & (b) 9/7

	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.

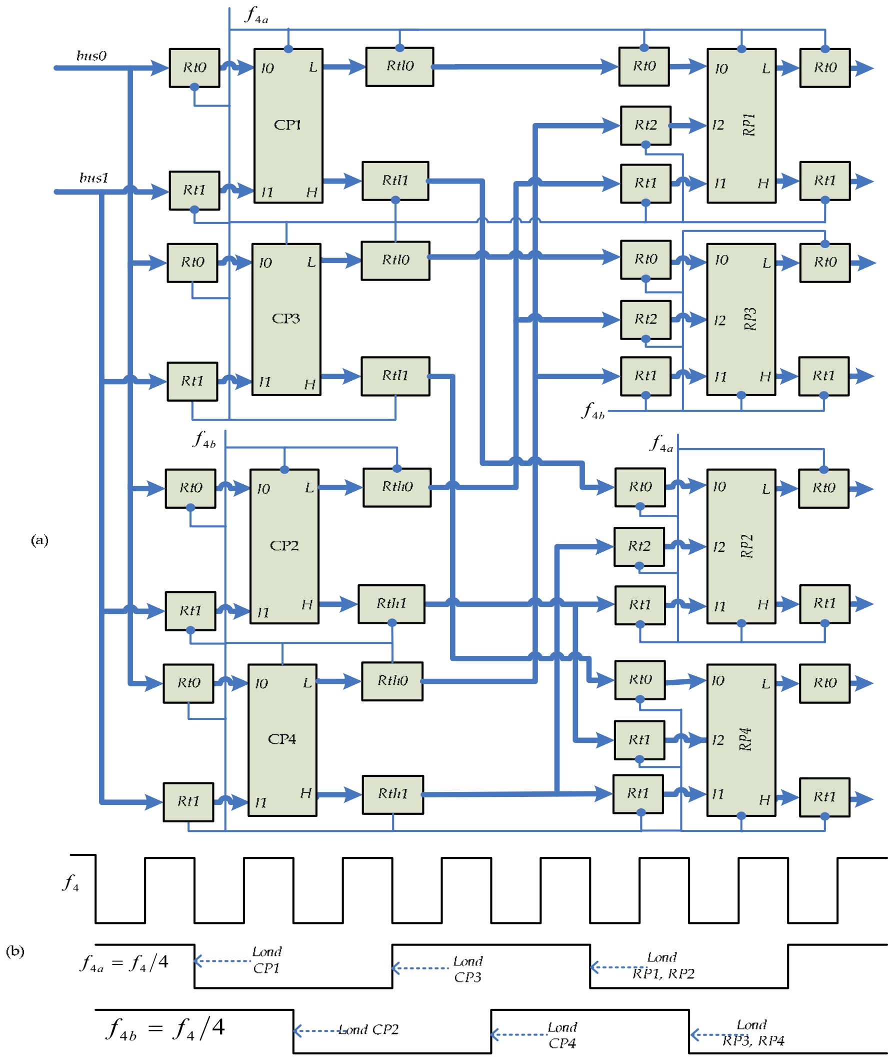


Fig. 10. (a) Proposed 2-D IDWT 4-parallel pipelined external architecture for 5/3 and 9/7 and combined (b) Waveforms of the clocks

	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					