

Chapter 2 MPEG-4 Video Encoder

Hardware Design

2.1 MPEG-4 Video Encoder Algorithm

Overview

Before designing the hardware architecture, the algorithm and complexity of each MPEG-4 video encoder component are analyzed. Therefore, software model are used to evaluate the performance of our algorithm before starting to design the hardware architecture, and the software model will be used to verify our hardware design in the future. The MPEG-4 video encode flow diagram is shown in Fig. 2-1

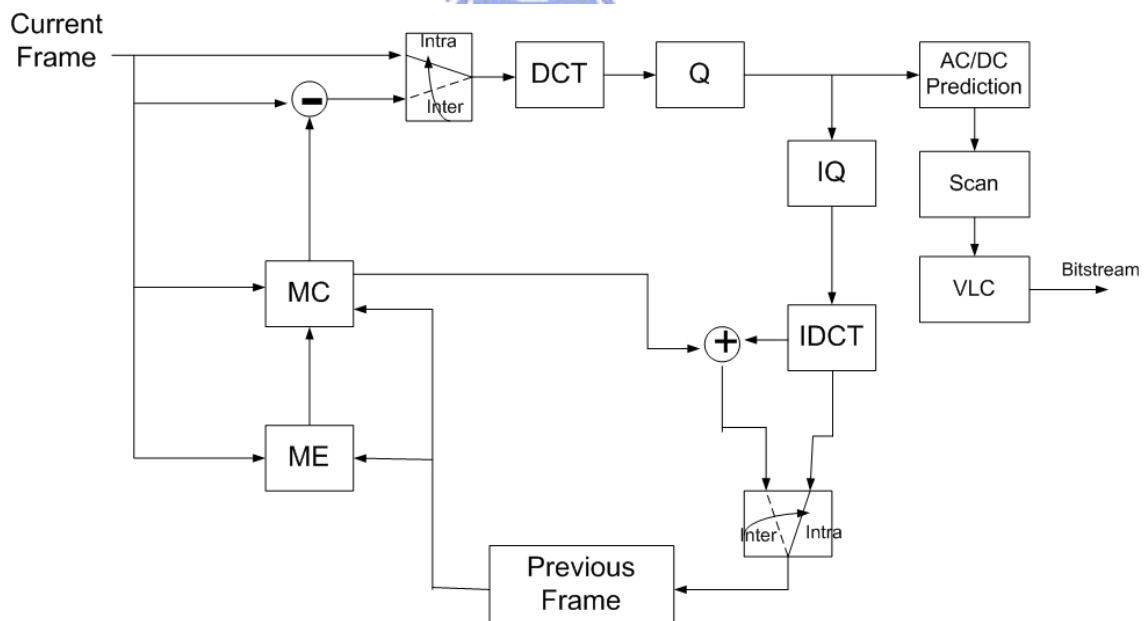


Figure 2-1 MPEG-4 video encode flow diagram

The first frame is called “I frame” which is sent into the DCT module to transfer the image data from the spatial domain to the frequency domain. The DCT algorithm is a loss-less and reversible mathematical transformation that converts a spatial amplitude representation of data into a frequency representation. The 2-D DCT equation is shown as below.

$$F(u,v) = \frac{2}{\sqrt{MN}} C(u)C(v) \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n) \cos\left[\frac{(2m+1)u\pi}{2M}\right] \cos\left[\frac{(2n+1)v\pi}{2N}\right],$$

where,

$$C(x) = \frac{1}{\sqrt{2}} \text{ when } x = 0 \text{ else} \\ = 1$$

M = Number of row in the input data set

N = Number of columns in the input data set

m = Row index in the time domain $0 \leq m \leq M-1$

n = Column index in the time domain $0 \leq n \leq N-1$

$f(m,n)$ = Time domain data

u = Row index in the frequency domain

v = Column index in the frequency domain

$F(u,v)$ = Frequency domain coefficient

One of the advantages of the DCT is its energy compaction property, that is, the signal energy is concentrated on a few components while most other components are zero or are negligibly small. The energy compaction property of the DCT is well suited for image compression, since as in most images, the energy is concentrated in the low to middle frequencies, and the human eye is more sensitive to the low and middle frequencies. From Fig. 2.2, we can see that most of the energy of the frequency domain is concentrated at the upper left corner.

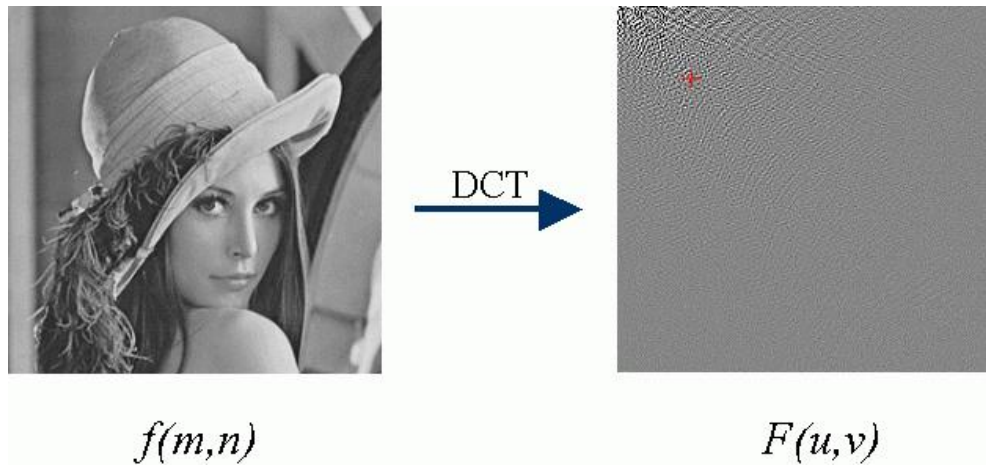


Figure 2-2 Lena image transferred from spatial domain to frequency domain

The DCT coefficient output will then be sent into the quantizer module. In Fig. 2-3, the $Q(x)$ is mapped from x , in this way we could reduce the expression bit numbers for x .

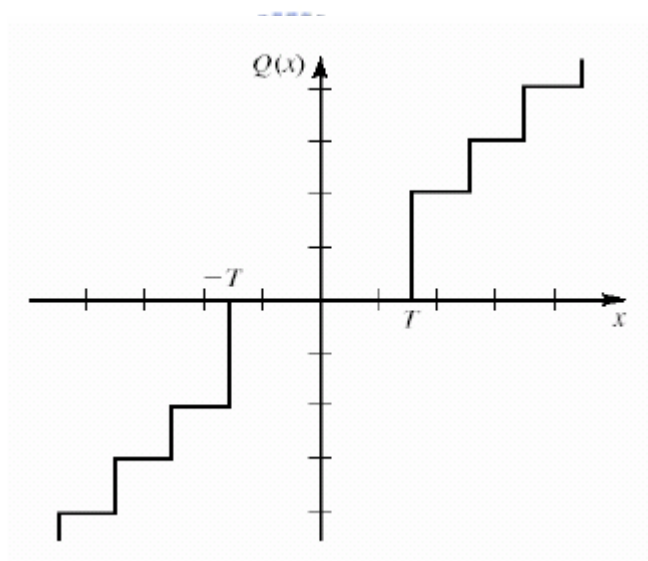


Figure 2-3 Quantization mapping

Fig. 2-4 is a macro block coded in intra mode, and Fig. 2-5 is the coefficient amplitude distribution of four blocks. We can see that the high relationship between neighboring DCT coefficient blocks. Therefore, the MPEG-4 uses the AC/DC prediction technique to reduce the expression amplitude of DC and AC values in I frame.

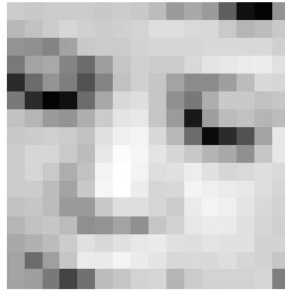


Figure 2-4 Macro block coded in intra mode

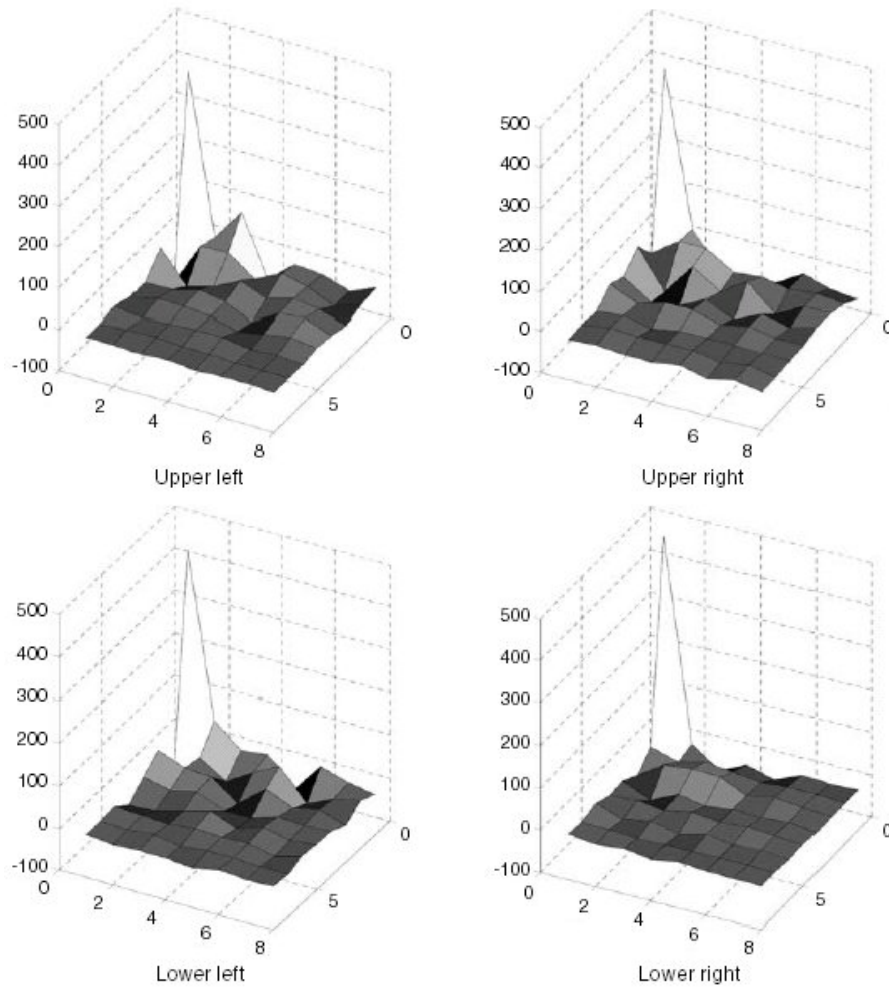


Figure 2-5 DCT coefficient (luma blocks)

The selection of AC/DC prediction direction is based on comparison of the horizontal and vertical DC differences around the block which is encoded currently. In Fig.2-6, the X is currently encoded block, and the DC prediction behavior is shown in Table 2-1.

Table 2-1 DC prediction pseudo code

IF ($|F_A[0][0]-F_B[0][0]| < |F_B[0][0]-F_C[0][0]|$)

Predict from block C

Else

Predict from block A

Where $F[0][0]$ is the DC value of a block.

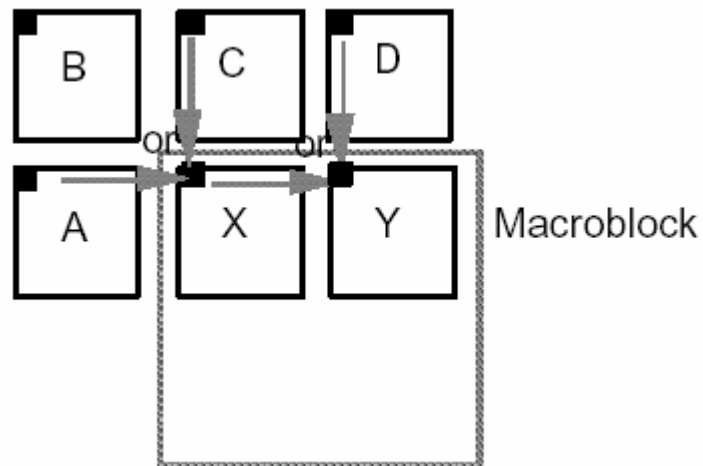


Figure 2-6 DC Prediction

After the direction of DC prediction is decided, the direction of AC prediction is the same as the direction of DC prediction which is shown in Fig. 2-7.

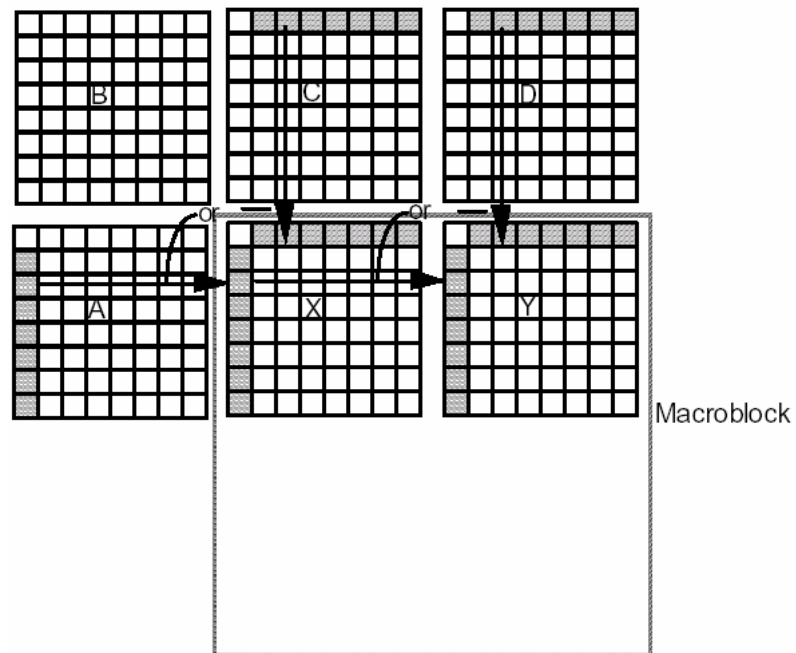


Figure 2-7 AC Prediction

The scan order supported by the MPEG-4 is shown in Fig. 2-8. After the re-ordering scan, the non-zero coefficient will be gathered in the head of scan sequences and this will help to improve the back-end entropy coding efficiency.

0	1	2	3	10	11	12	13
4	5	8	9	17	16	15	14
6	7	19	18	26	27	28	29
20	21	24	25	30	31	32	33
22	23	34	35	42	43	44	45
36	37	40	41	46	47	48	49
38	39	50	51	56	57	58	59
52	53	54	55	60	61	62	63

Alternate-Horizontal scan

0	4	6	20	22	36	38	52
1	5	7	21	23	37	39	53
2	8	19	24	34	40	50	54
3	9	18	25	35	41	51	55
10	17	26	30	42	46	56	60
11	16	27	31	43	47	57	61
12	15	28	32	44	48	58	62
13	14	29	33	45	49	59	63

Alternate-Vertical scan

0	1	5	6	14	15	27	28
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	49	57	58	62	63

Zigzag scan

Figure 2-8 VLC Scan Order

The VLC is a look-up table operation, it will look for a code word according to the 3-D run length coding (Run, Level, Last). In addition, the encoder will reconstruct the quantized coefficient by the inverse quantizer and the inverse DCT and then store it in the previous

frame memory.

In the “P frame”, current frame will be input into the motion estimation to compare with the previous frame and derived a motion vector. In this way, the motion compensation could subtract the predicted macro block from the current macro block. Then, the error residue is input into texture coding module. Fig. 2-9 shows the concept of motion estimation.

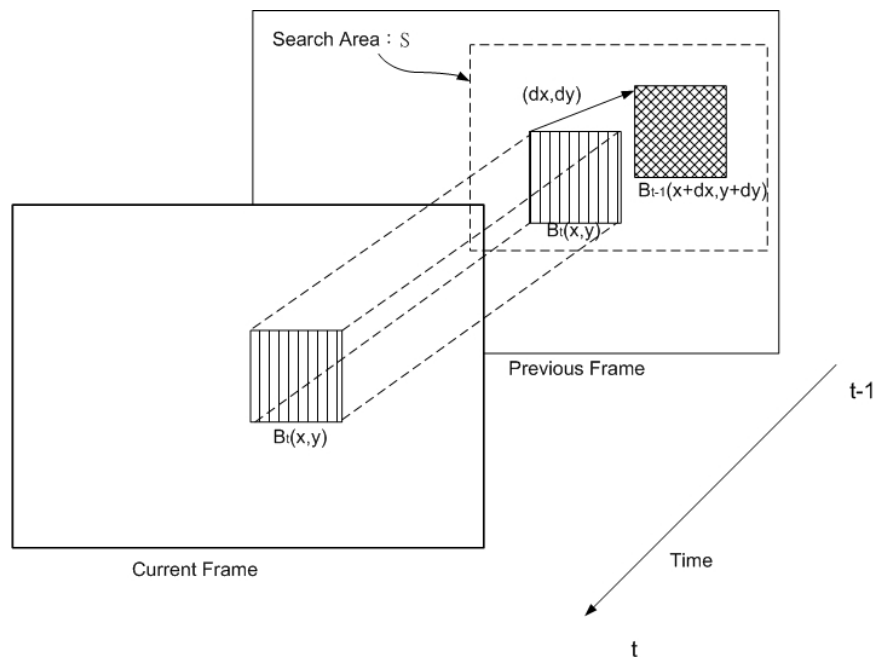


Figure 2-9 Motion Estimation Concept

Fig. 2-10 shows a moving car and Fig. 2-11 is derived by subtracting previous frame from current frame directly without motion estimation and plus an offset value 128. Fig. 2-12 shows the motion vector distribution after doing a full search of search range = 16. Fig. 2-13 shows the error residue distribution which is derived by subtracting previous frame from current frame according to the motion vector and plus an offset value 128. We can see that the car and background contour in Fig. 2-13 is unobvious contrast with Fig. 2-11, so that the motion estimation and the motion compensation subtraction could decrease the image differences to be close to zero. This will improve the coding efficiency in advance.



Figure 2-10 Moving car



Figure 2-11 Error residue without Motion estimation

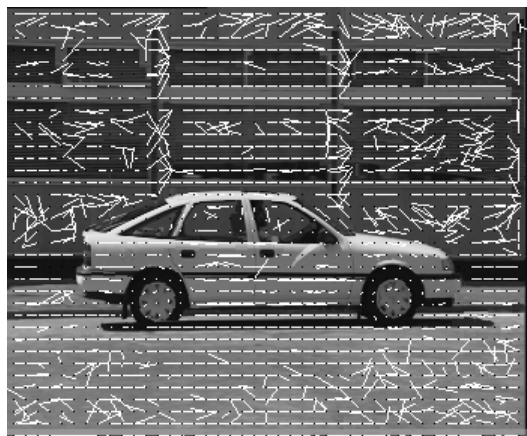


Figure 2-12 Motion vector distribution

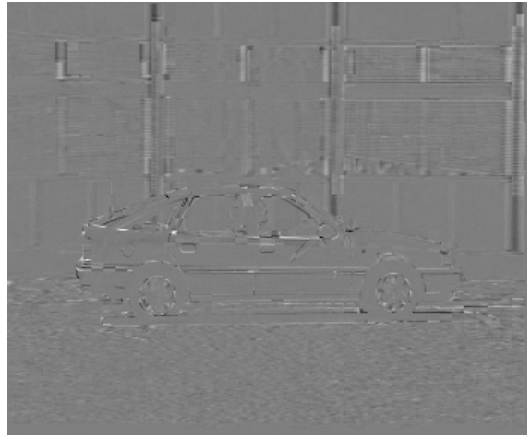


Figure 2-13 Error residues with motion estimation

2.2 Motion Estimation Architecture Design

2.2.1 Algorithm Analysis and Design

The motion estimation is an important part in the MPEG-4 video encoder. It dominates the encoded image quality, the compression ratio, the computation time and it requires the largest hardware resource in the whole encoder. In all algorithms of the motion estimation, full search block matching algorithm (FSBMA) is well known and has been developed for fast implementation [5]-[7] because of its good image quality and regularity data flow in hardware design. Fig. 2-14 shows a 16x16 current macro block doing a full search of search range = 16, and the pseudo code is shown in Table 2-2.

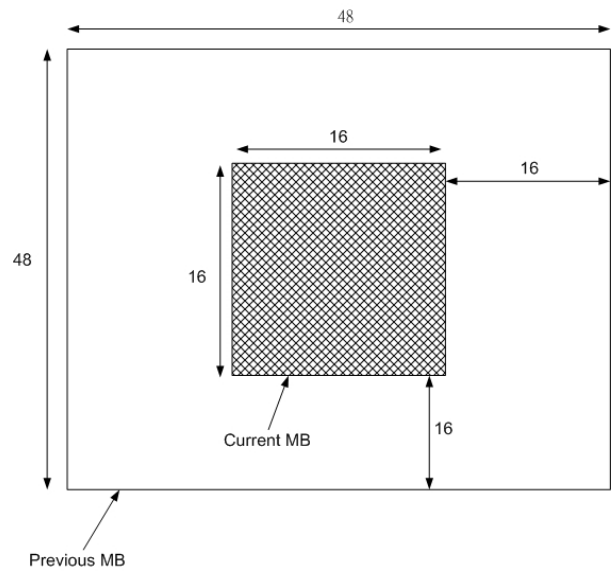
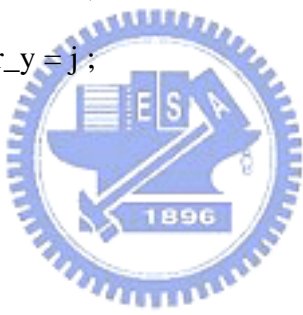


Figure 2-14 Search area and current MB of full search



Table 2-2 Full search pseudo code

<pre> begin min_SAD=Large_Number; for i= -16 to 16 for j= -16 to 16 { cur_SAD=SAD(i,j) ; if cur_SAD < min_SAD { min_SAD = cur_SAD ; motion_vector_x = i ; motion_vector_y = j ; } } } end </pre>	
$SAD(i, j) = \sum_{n_1=0}^{15} \sum_{n_2=0}^{15} B(n_1, n_2, t) - B(n_1 - i, n_2 - j, t - 1) , B(n_1, n_2, t) \text{ is the pixel value when time} = t \text{ and position} = (n_1, n_2)$	

In Table 2-2, the full search algorithm is a four levels loop operation and whose operation require 3.3 giga times per second for subtracting to get absolute value for 30Hz CIF image size processing. Due to the high computational complexity, the full search algorithm has a significant problem in real-time applications. To resolve this problem, we proposed a hierarchical motion estimation algorithm (HMEA) which is based on full search algorithm and could reduce the computation complexity and retain good image quality.

Our proposed HMEA consists of three resolution levels of images :

Level0 : It is the original image resolution.

Level1 : It is the image which downsampled from Level0, the downsampling filter is expressed as following :

$$p_k^1(i, j) = \lfloor [p_k^0(2i, 2j) + p_k^0(2i+1, 2j) + p_k^0(2i, 2j+1) + p_k^0(2i+1, 2j+1)] / 4 \rfloor ,$$

where $p_k^1(i, j)$ means the Level1 pixel value at the position (i, j) of the k th frame

$p_k^0(i, j)$ means the Level0 pixel value at the position (i, j) of the k th frame

Level2 : It is the image which downsampled from Level1 ,the downsampling filter is expressed as following :

$$p_k^2(i, j) = \lfloor [p_k^1(2i, 2j) + p_k^1(2i+1, 2j) + p_k^1(2i, 2j+1) + p_k^1(2i+1, 2j+1)] / 4 \rfloor ,$$

where $p_k^2(i, j)$ means the Level2 pixel value at the position (i, j) of the k th frame

$p_k^1(i, j)$ means the Level1 pixel value at the position (i, j) of the k th frame

Our experiments show that downsizing the image without low pass filter will degrade 2dB PSNR on average contrast with downsizing image with low pass-filter, and the experimental results are shown in Table 2-3. That's the reason we adopt low pass filter process in our image downsizing procedure.

Table 2-3 PSNR comparison of different downsizing strategies

Video Sequence	Downsizing without low pass filter	Downsizing with low pass filter
	PSNR	PSNR
News	31.06	32.88
Foreman	29.33	31.29
Flower garden	23.16	24.59
Table tennis	28.29	31.63

After the downsampling procedure, the macro block size in Level0, Level1, Level2 are

16x16, 8x8 and 4x4 which are shown in Fig. 2-15.

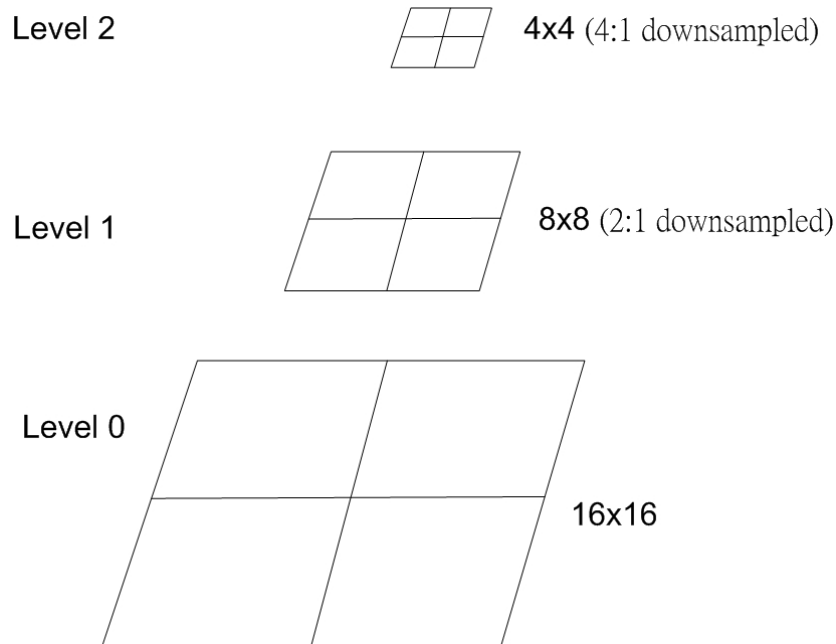


Figure 2-15 Downsampling procedure

The HMEA is based on a multi-resolution frame structure mentioned above. The overall procedure is shown in Fig. 2-16.

Search at Level2 : The Level2_Cur is a 4x4 block which downsampled 2 times from the original resolution macro block and the Level2_Pre is a 12x12 previous frame search area. How many candidates should be chosen on Level2 is evaluated, and the results are shown in Table 2-4. Table 2-4 includes different types of the video sequences, “News” is an almost static video sequence, “Foreman” is a middle motion video sequence, “Flower garden” is a video sequence whose whole scene is moving and contains many tiny objects in the scene, and “Table tennis” is a big motion video sequence. From all the sequences, we can see that the effect of 3rd candidate does not have big impact on the improvement of PSNR. On the contrary, too many candidates in Level2 might cause local minimum problem to direct the next level search to a wrong motion vector, so that we choose two candidates in the Level2.

For search range=16 case, Level2_Cur do a full search of search range = 4 on the Level2_Pre and get two motion vectors with SADs. These two motion vectors will be the start point of next level full search.

Table 2-4 PSNR comparison of different candidate number in Level2

Video sequence	1 candidate	2 candidates	3 candidates	Full Search
News	32.56	32.88	33.17	33.46
Foreman	30.29	31.29	31.60	32.06
Flower garden	24.37	24.59	25.46	25.81
Table tennis	28.55	31.63	31.74	32.07

Search at Level1 : The Level1_Cur is a 8x8 block which downsampled 1 time from original resolution macro block. Level1_Pre is a 12x12 previous frame search area. Level1_Cur do 2 times of full search of search range = 2 on the Level1_Pre to refine the Level2 search results and get one motion vector with SAD. This motion vector will be the start point of next level search.

Search at Level0 : The Level0_Cur is a 16x16 block which is the original resolution macro block. Level0_Pre is a 20x20 previous frame search area. Level0_Cur do a full search of search range = 2 to refine the result of Level1 on the Level0_Pre.

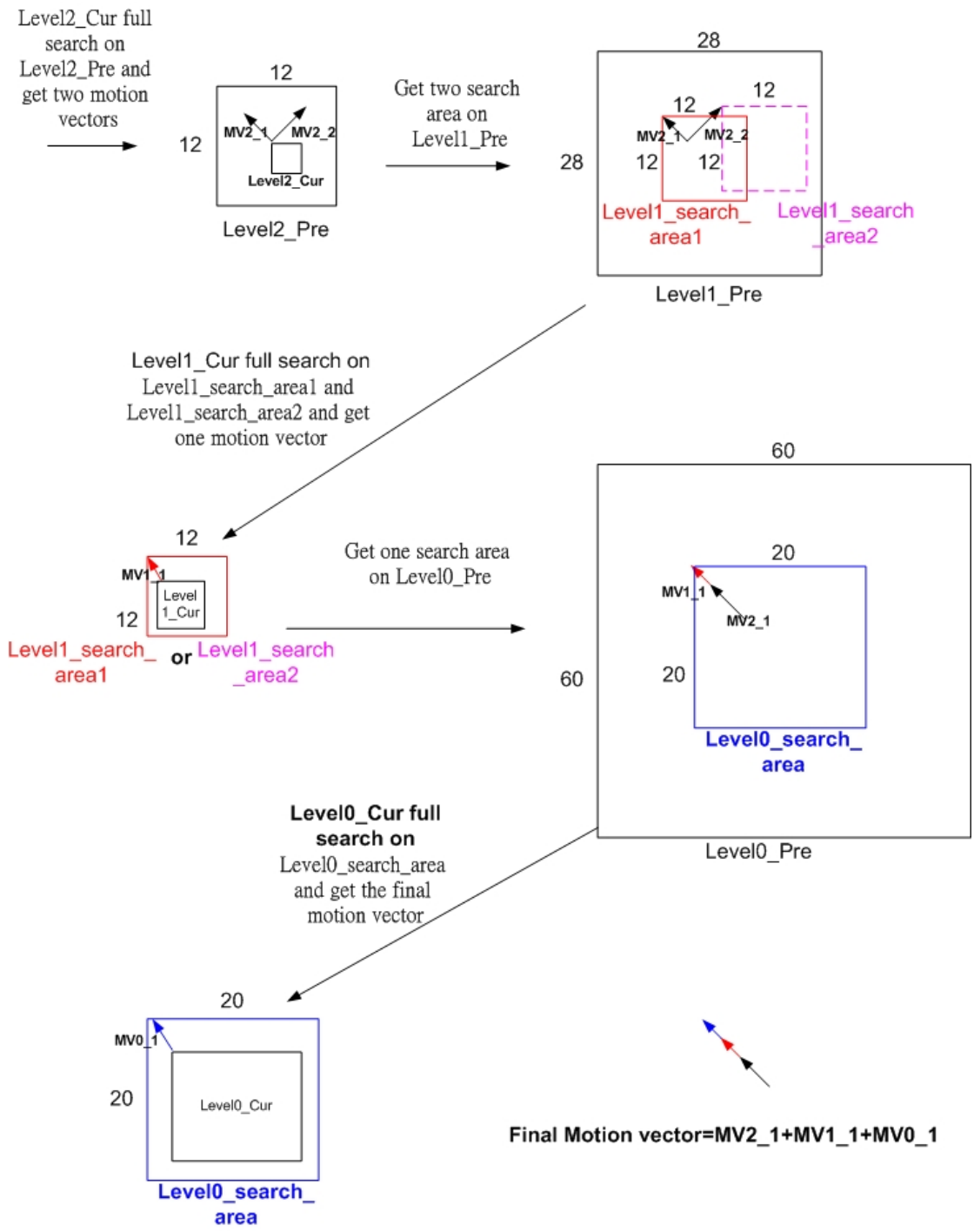


Figure 2-16 Hierarchical search procedure

Before hardware design, software mode are used to simulate the HMEA and other motion estimation algorithm such as full search, diamond search (radius is one pixel) and three step search behavior. The results are shown in Table 2-5, and the PSNR of HMEA is very close to Full search algorithm.

Table 2-5 PSNR comparison between different algorithms

Video sequences	Full search	Proposed HMEA	Diamond search	Three step search
Foreman	32.06	31.29	30.80	30.85
Akiyo	42.95	42.82	42.92	42.8
Flower garden	25.81	24.59	25.68	24.58
News	33.46	32.88	32.54	32.7
Mobile	24.23	23.93	24.15	23.92
Table Tennis	32.07	31.63	30.66	30.5

2.2.2 Hierarchical Motion Estimation Hardware

Architecture Design

✧ Downsampling Unit

Four rows of Level0 are downsampled and two rows of Level1 and one row of Level2 are got each loop, and totally 72 loops are needed for one CIF image.

In a loop, four original resolution row data are input into the downsampling unit, the L0_row0, L0_row1, L0_row2, L0_row3 which are shown in Fig. 2-17, and Fig. 2-18 shows the data path to downsample Level0 image to Level1 image.

1. Input 1st row : At first, input the L0_row0 in Fig. 2-17 into the L0_even_data_path in Fig. 2-18. After processing the L0_row0, the P1+P2 and P3+P4 temporary values are stored in RAM1.
2. Input 2nd row : Second, input the L0_row1 into the L0_row1_data_path, the data flow

through the three stages pipeline registers and L1_row0 result are stored in the RAM2. At the same time, the data output from the 2nd stage pipeline register in the L0_row1_data_path, the L1_P11, the L1_P12, the L1_P13, and the L1_P14 will be input into the L1_row0_data_path and the temporary value of L1_P11+L1_P12 and L1_P13+L1_P14 are stored in RAM4.

3. Input 3rd row : Third, the L0_row2 is sent into the L0_even_data_path, and the P1+P2 and P3+P4 temporary values of the L0_row2 are stored in RAM1.
4. Input 4th row : Fourth, the L0_row3 is sent into the L0_row3_data_path. The same condition as “Input 2nd row”, the L1_row1 result will be stored in the RAM3, and the L1_P21, the L1_P22, the L1_P23 and the L1_P24 will be input into the L1_row1_data_path. Then, the L2_row result will be stored in the RAM5.
5. Finally, the RAM2, the RAM3, and the RAM5 contain the Level1 and Level2 result respectively.
6. Repeat step 1. ~ step 5. 71 times to finish one CIF image downsampling.

Using this architecture, only need $25344 + 3168 + 1584 = 30096$ cycles are needed to finish one CIF image downsampling. That is, 76 cycles per macro block on average.

L0_row0	P1	P2	P3	P4	P1	P2	P3	P4	P1	P2	P3	P4	P1	P2	P3	P4	P1	P2	P3	P4
L0_row1	P1	P2	P3	P4	P1	P2	P3	P4	P1	P2	P3	P4	P1	P2	P3	P4	P1	P2	P3	P4
L0_row2	P1	P2	P3	P4	P1	P2	P3	P4	P1	P2	P3	P4	P1	P2	P3	P4	P1	P2	P3	P4
L0_row3	P1	P2	P3	P4	P1	P2	P3	P4	P1	P2	P3	P4	P1	P2	P3	P4	P1	P2	P3	P4

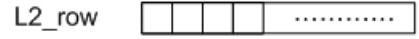
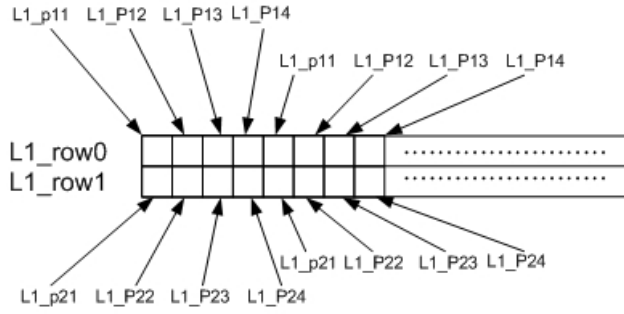


Figure 2-17 Downsample 4 Level0 rows

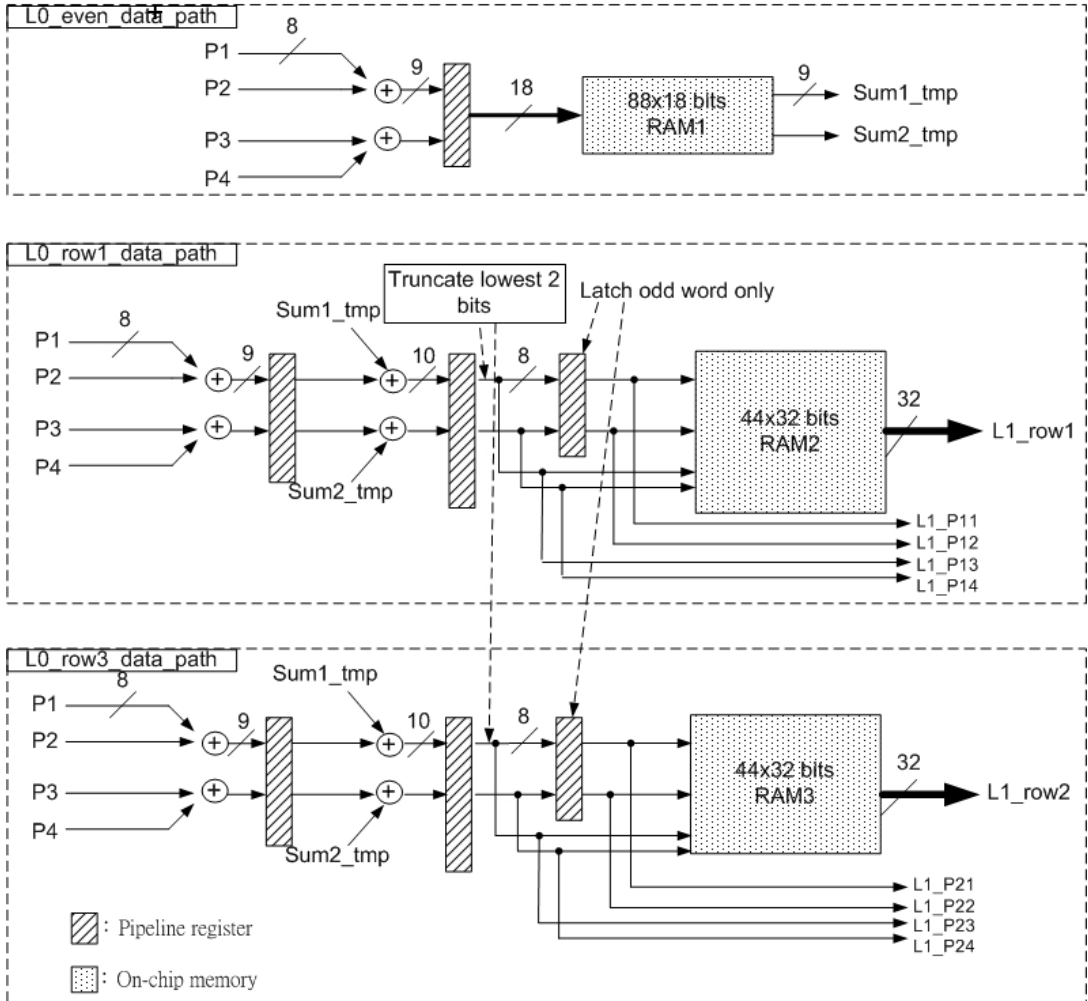


Figure 2-18 Level0 downsampling

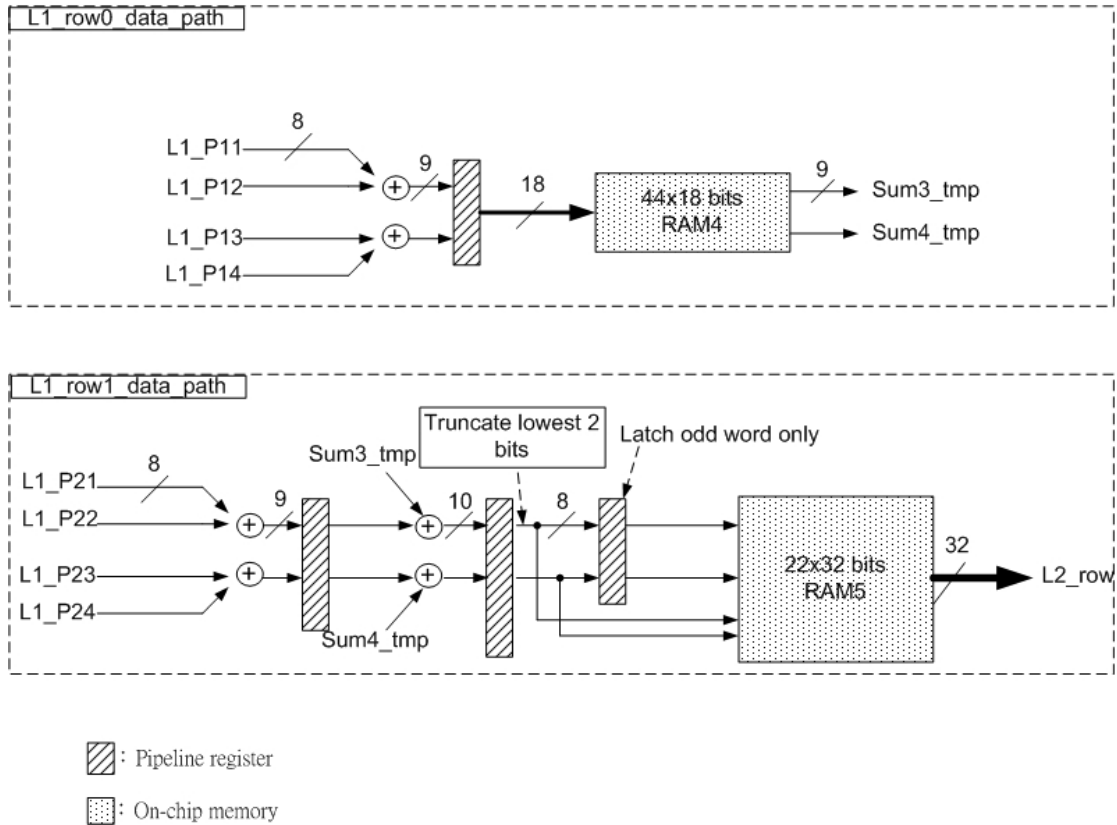


Figure 2-19 Level1 downsampling

✧ Motion Estimation Architecture

After downsampling process, the hierarchical motion estimation is started.

In [8], the BSU (Basic Search Unit) is a one dimension systolic processing element array. The data reuse is not good enough so that the current block and previous block data must be reloaded once again. This architecture needs 40 mega clock cycles to finish 30 CIF images motion estimation. If it adds other system overhead into the consideration (such as bus arbitration or software overhead), it will be difficult to achieve real-time encoding. Therefore, we propose an enhanced 2-D semi-systolic BSU (2DBSU) architecture to improve the data reuse capability and use two 2DBSU to improve the processing speed. Our proposed architecture decreases 80% bandwidth requirement and only needs 4.396 mega clock cycles to achieve 30 CIF images motion estimation.

Fig. 2-20 shows the 2DBSU architecture, this architecture could do a 4x4 current block full search whose search range = 2 and the basic data flow is shown in Fig. 2-21.

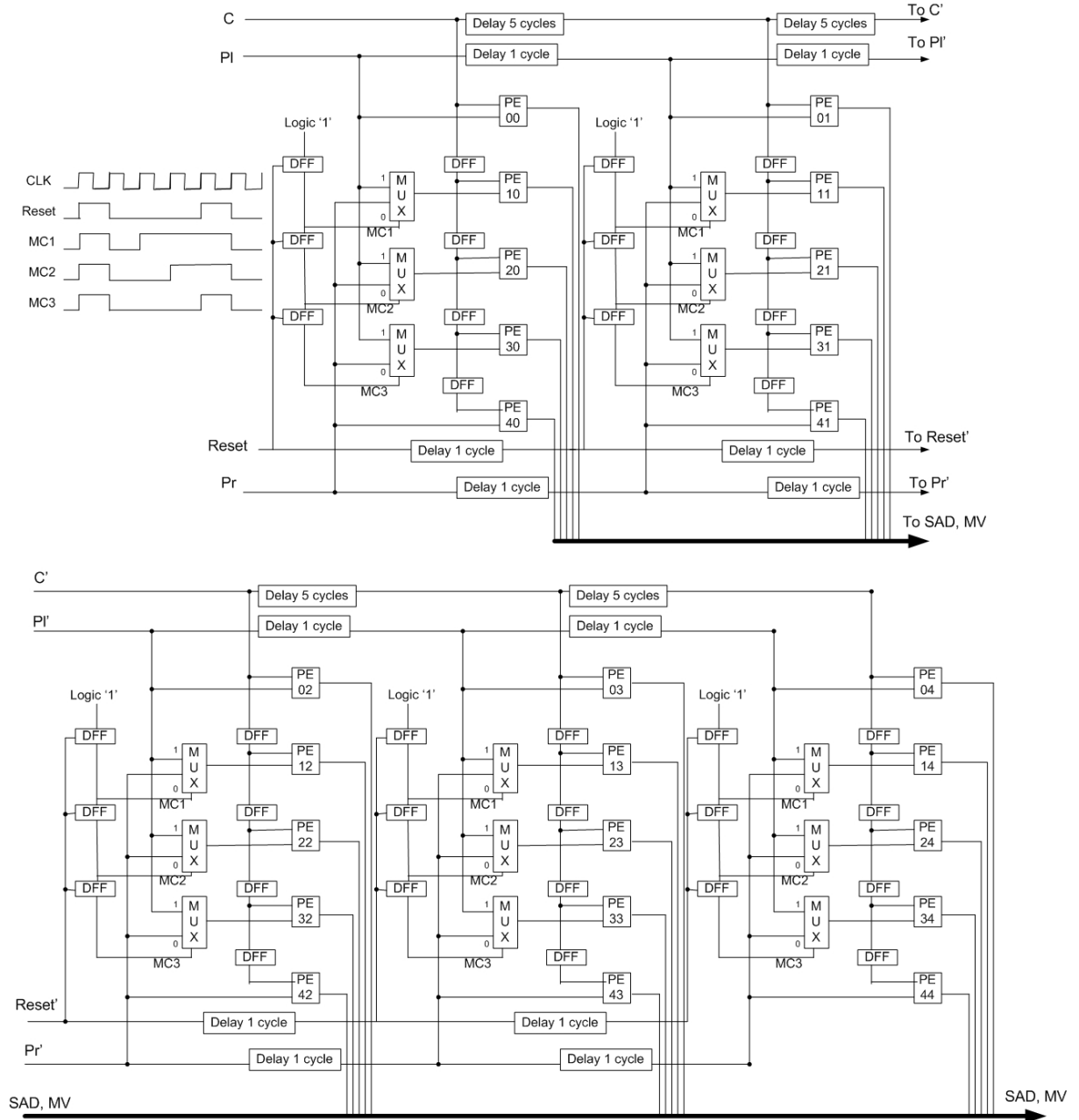


Figure 2-20 2DBSU architecture

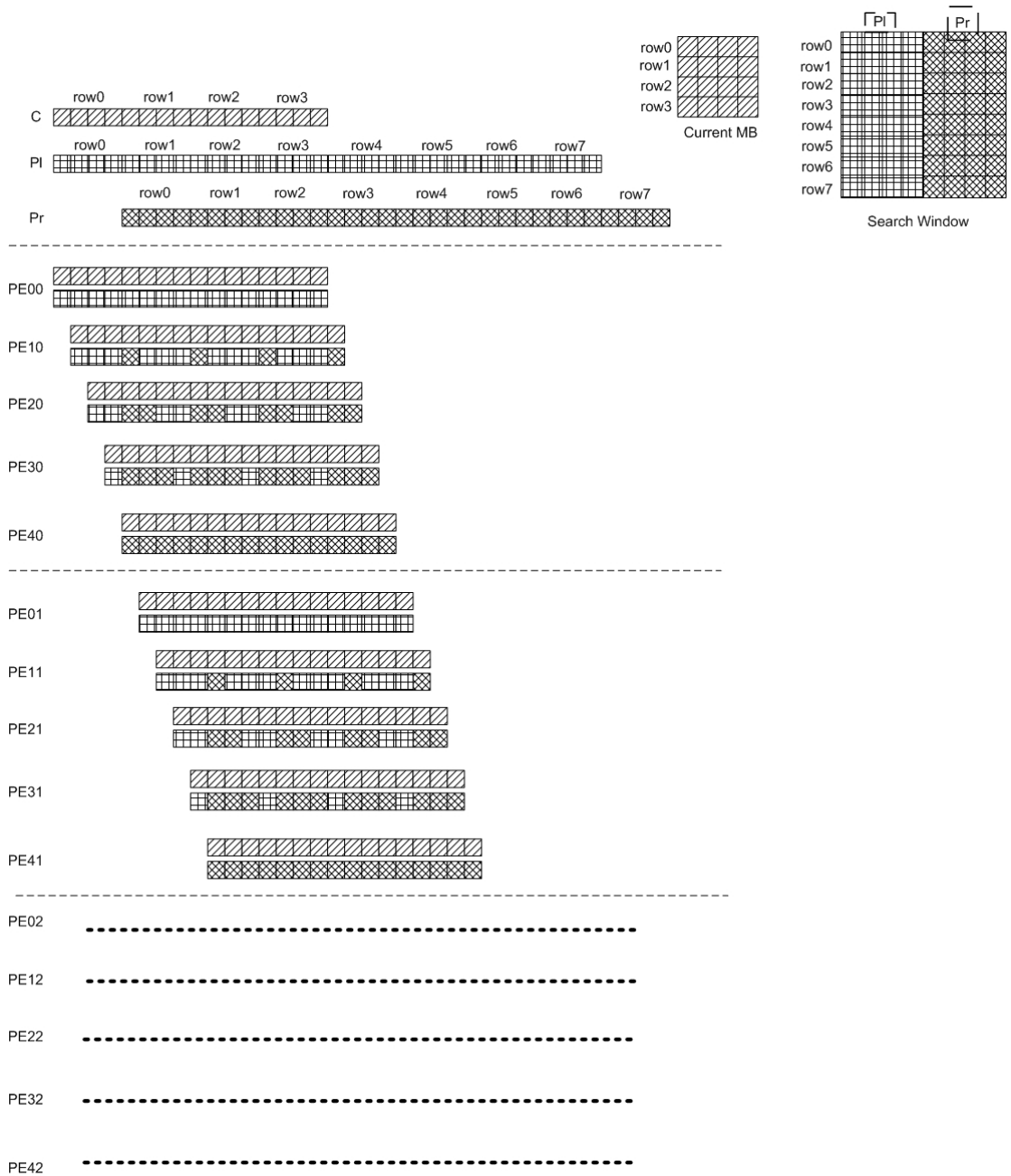


Figure 2-21 2DBSU basic data flow

In Fig. 2-21, the current MB is a 4x4 block and the search window is a 12x12 block, the search window is partitioned into two parts, the left part and the right part (that is “PI” and “Pr” in Fig. 2-21), then input the data into the 2DBSU architecture from the C, the PI and the Pr input ports as shown in Fig. 2-20. The PE00 in 2DBSU will accumulate the SAD of the search position (-2, -2), and the PE10 will accumulate the SAD of the search position (-1, -2), in the similar way, the other PEs will accumulate the SADs of the other search position. Based

on this 2DBSU architecture and the basic data flow, we develop a motion estimation architecture which is shown in Fig. 2-22. Two 2DBSU are used to accelerate the motion estimation and improve the data-reuse capability. This architecture is commonly used among different levels and could extend search range without adding additional hardware.

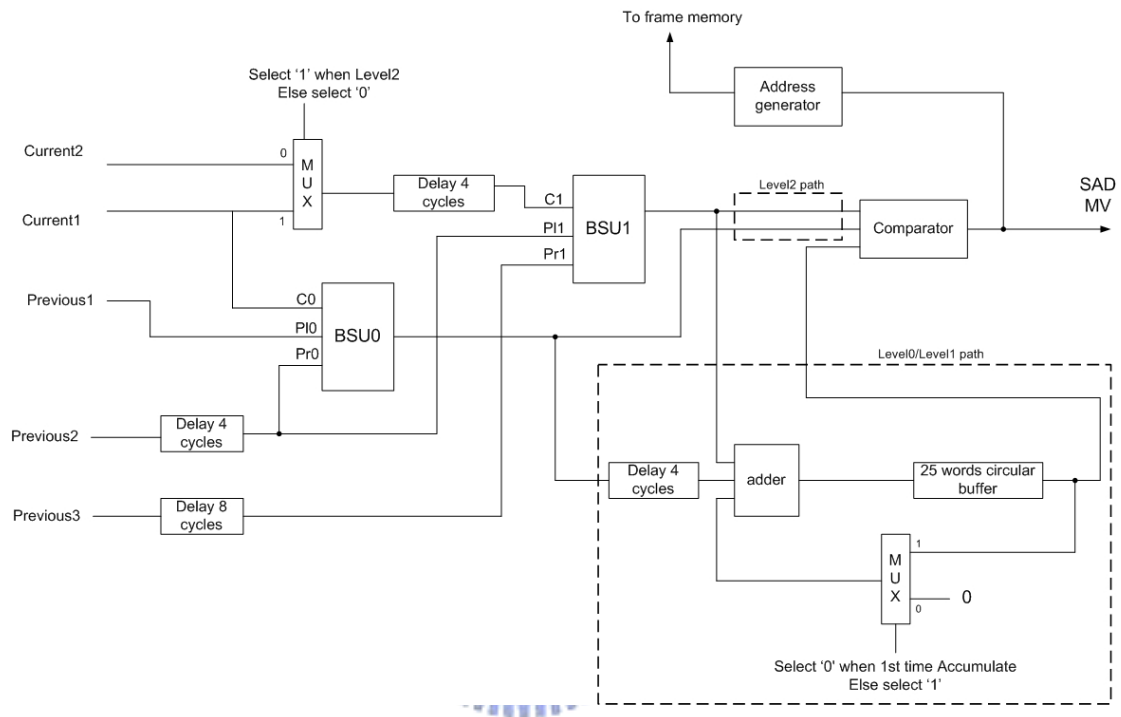


Figure 2-22 Motion unit architecture

1. In Level2 search: the current MB is a 4x4 block and the search window is a 12x12 block, the search window is partitioned into two parts, one part is input into the BSU0 and the other is input into BSU1. The data partition and data flow is shown in Fig. 2-23. The C0 is fed the current MB twice and the C1 is fed the same data as the C0 fed but delayed 4 cycles. And the PI0 is fed the left part data of search window row by row. The Pr0 and the PI1 are fed the middle part of search window four cycles later than the PI0. The Pr1 is fed the right part of search window four cycles later than the PI1. After 17 cycles, the 1st Level2 SAD will be output with relative motion vector from BSU0 and the other SADs and motion vectors will also be output from the BSU0 and the BSU1 continuously. The

SADs output from the BSU0 and the BSU1 will be input into the comparator. The minimum SAD and its corresponding motion vector will be retained in the comparator until the Level2 search is finished. After 56 cycles the two candidate motion vectors are found.

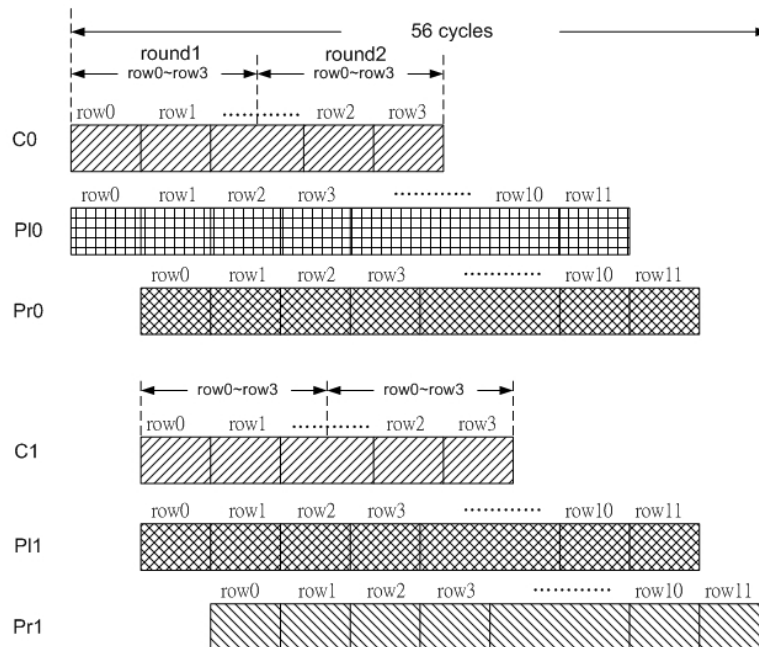
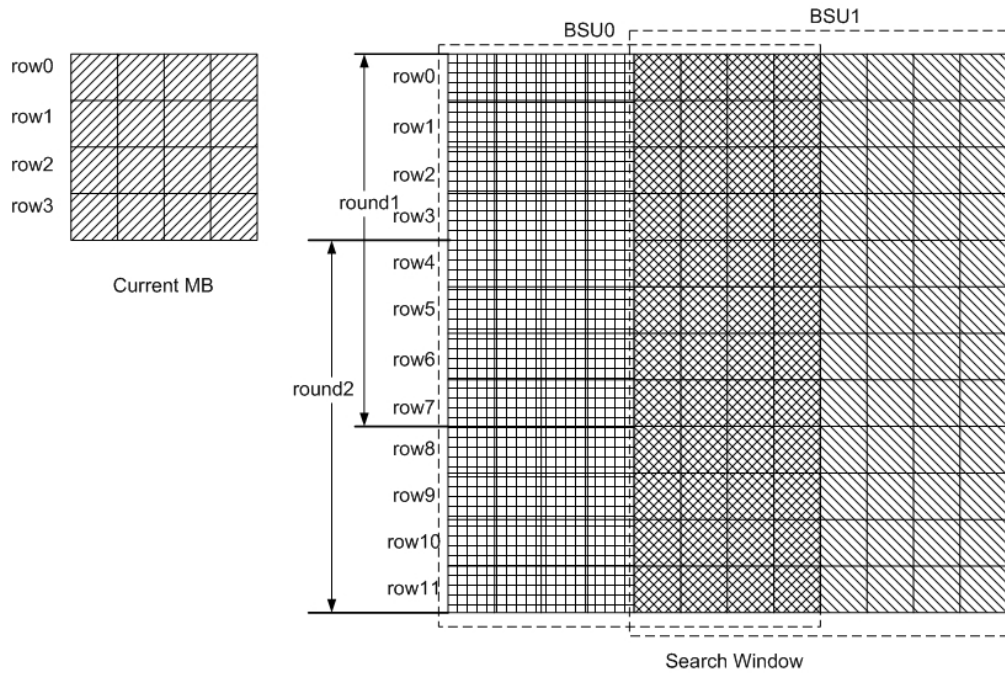


Figure 2-23 Level2 data flow

2. In Level1 search: the two candidate motion vectors found in Level2 are used as the starting point and do a full search of search range = 2 to refine the motion vector. In Level1, the current MB is an 8x8 block and the search window is a 12x12 block. The current block is partitioned into four parts (the left-upper, left-bottom, right-upper and right-bottom) and the search window are partitioned into three parts as shown in Fig. 2-24. Then, the left two parts (left-upper and left-bottom) of the current MB are input row by row into the C0, and the right two parts (right-upper and right-bottom) of the current MB are input row by row into the C1 four cycles later than the C0. In the search window, the condition is the same as searching at Level2, the P10 is fed with the left part data of search window row by row. The Pr0 and the P11 are fed with the middle part of search window four cycles later than the P10, and the Pr1 is fed with the right part of search window four cycles later than the P11. After 17 cycles, the 25 partial SADs of left-upper and right-upper part are begin to be output from BSU0 and BSU1 sequentially, and these partial SADs are sent into the “Level0/Level1 path” in Fig. 2-22 to accumulate with the partial SADs of left-bottom and right-bottom later. After 56 cycles, the 1st SAD of Level1 with the relative motion vector will be output from the “25 words circular buffer”, and the rest of the other 24 SADs with the motion vectors are also output sequentially. The Level1 SADs will be sent into the comparator and the minimum SAD with the corresponding motion vector will be retained in the comparator. After 56+25 = 81 cycles, the first time Level1 search is finished. Because Level2 has two candidates, Level1 search must be done two times according to these two motion vector candidates, and decide which motion vector is proper one. The Level1 search totally need 162 cycles per MB.

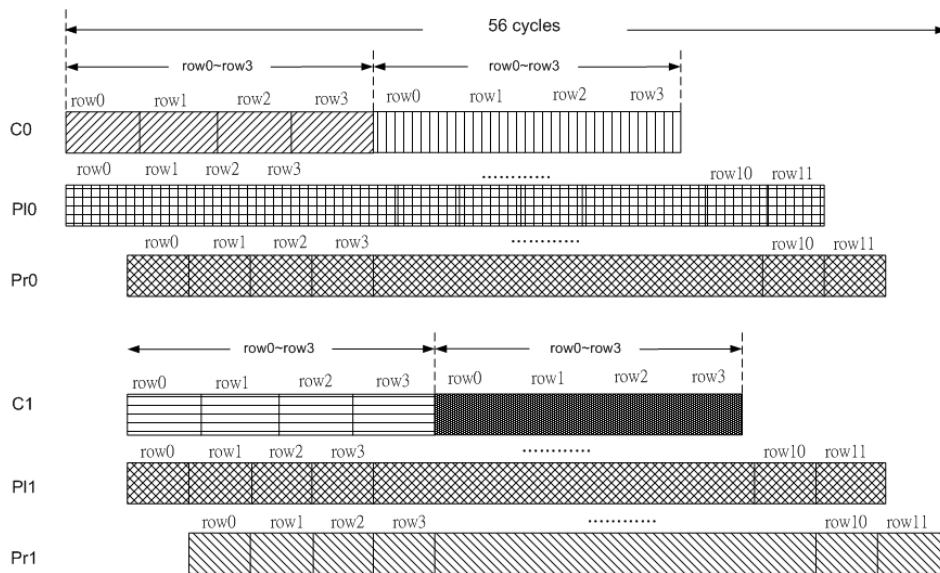
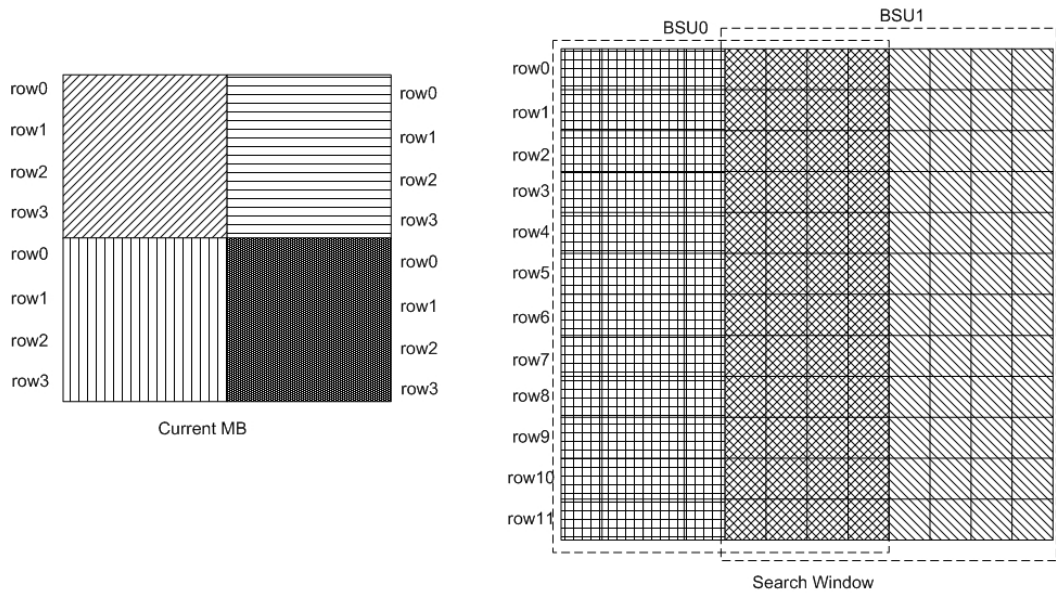


Figure 2-24 Level1 data flow

3. In Level0 search: The most proper motion vector found in Level1 is the search start point in Level0 search. A full search whose search range = 2 is done to refine the motion vector found in Level1. In Level0, the current MB is a 16x16 block and the search window is a 20x20 block. The current block is partitioned into sixteen parts and the search window into five parts as shown in Fig. 2-25.

(1) In the first round: The “LL” part is input into the C0, and the “LR” part is input into

the C1 four cycles later than the “LL” part. The part1 of search window is input into the P10, the part 2 is input into the Pr0 and the P11 four cycles later than the part1, the part3 is input into the Pr1 four cycles later than the part2. At the 17th cycle, the 1st partial SAD with relative motion vector is input into the “25 word circular buffer” as shown in Fig. 2-22 to accumulate with other partial SADs of the rest parts later. After 88 cycles, the 25 sums of partial SADs of the “LL” and the “LR” are stored in the “25 word circular buffer”.

- (2) In the second round: The second round is started at the 89th cycle, the “RL” is begin to be input into the C0, and the “RR” is input into the C1 four cycles later than the “RL”. In the search window, the part3 is input into the P10 and the part4 is input into the Pr0 and P11 four cycles later than the part3. The part5 is input into the Pr1 four cycles later than the part4. The 25 partial SADs are input into the “Level0/Level1 path” and accumulate with the 25 sums of partial SADs gotten in the 1st round. At the 152th cycle, the 1st Level0 SAD with relative motion vector is output from the “25 word circular buffer” and input into the comparator. The minimum SAD and its corresponding motion vector will be retained in the comparator. At the 176th cycle, the 1st motion vector is output from the “25 word circular buffer”, and the Level0 search totally need $176+25 = 201$ cycles.

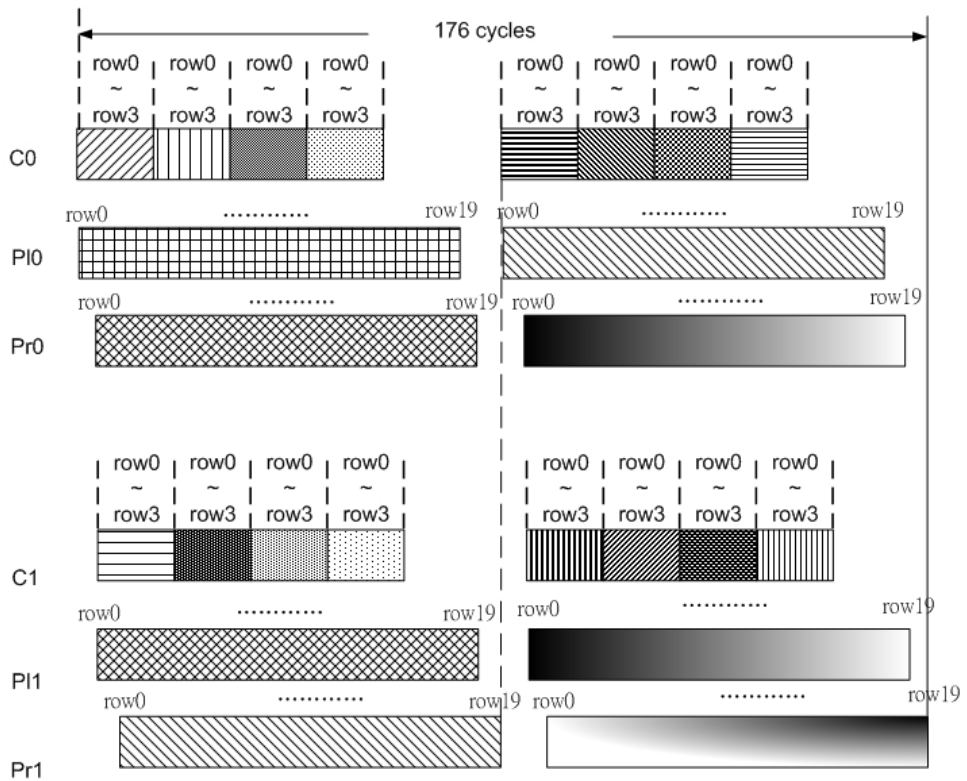
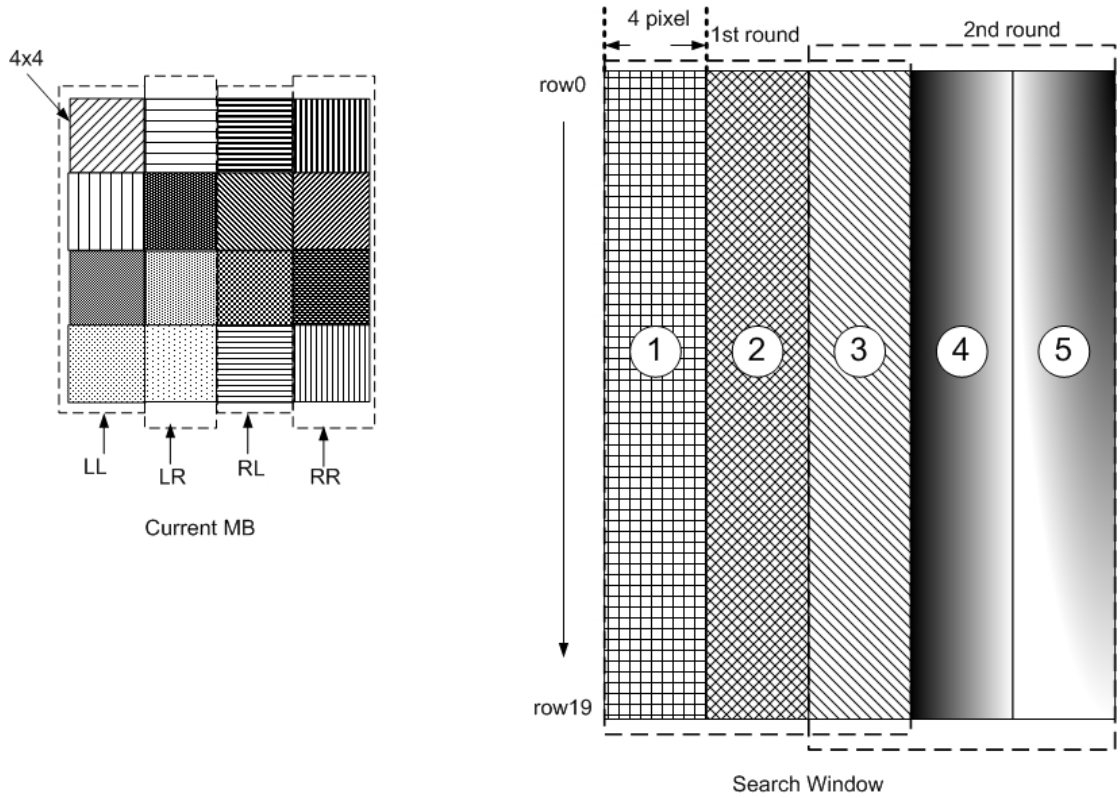


Figure 2-25 Level0 data flow

In Table.2-5, the image quality of HMEA is very close to full search. In addition to

quality, another important character of the motion estimation is its processing speed. The processing speed and the ASIC implementation cost of the HMEA are compared in Table 2-6. The [8]-[10] use the similar algorithm as HMEA. In Table 2-6, HMEA only needs 495 cycles to find one motion vector. The 495 cycles include the downsampling cycles needed per MB. The on-chip memory usage also includes the memory required in downsampling unit. The HMEA costs an acceptable ASIC area and needs the fewest cycle to accomplish the motion estimation computation.

Table 2-6 Performance comparison with other architectures (search range = 16)

Architecture	Cycles per MV	Required cycles for 30 CIF	Gate counts	On-chip memory usage
Proposed HMEA (Hierarchical)	495	5.88 mega clock cycles	59K gates	1393 bytes
Kun-Bin Lee [9] (Subsampling)	615	7.31 mega clock cycles	24.8K gates	2623 bytes
Seongsoo [10] (Low Bit-Resolution)	1320	52.27 mega clock cycles	110K gates	90K bytes
Jae Hun Lee [8] (Hierarchical)	2640	40 mega clock cycles	25K gates	288 bytes

◇ Motion unit integration

The motion compensation subtracts and compensation add will be described in the following, and the whole motion unit integration will also be explained.

The motion unit contains the motion estimation, the motion compensation add, the motion compensation subtract and the downsampling. The architecture is shown in Fig. 2-26.

The “MC current MB” and “MC previous MB” are on-chip memory. The “MC controller” is a finite state machine which could latch off-chip pixel data according to motion vector, and the “MC interpolator” is a bilinear interpolator which could do the pixel interpolation according to the motion vector. The “MC downsampling” is a downsampling circuit to downsample the MB level data.

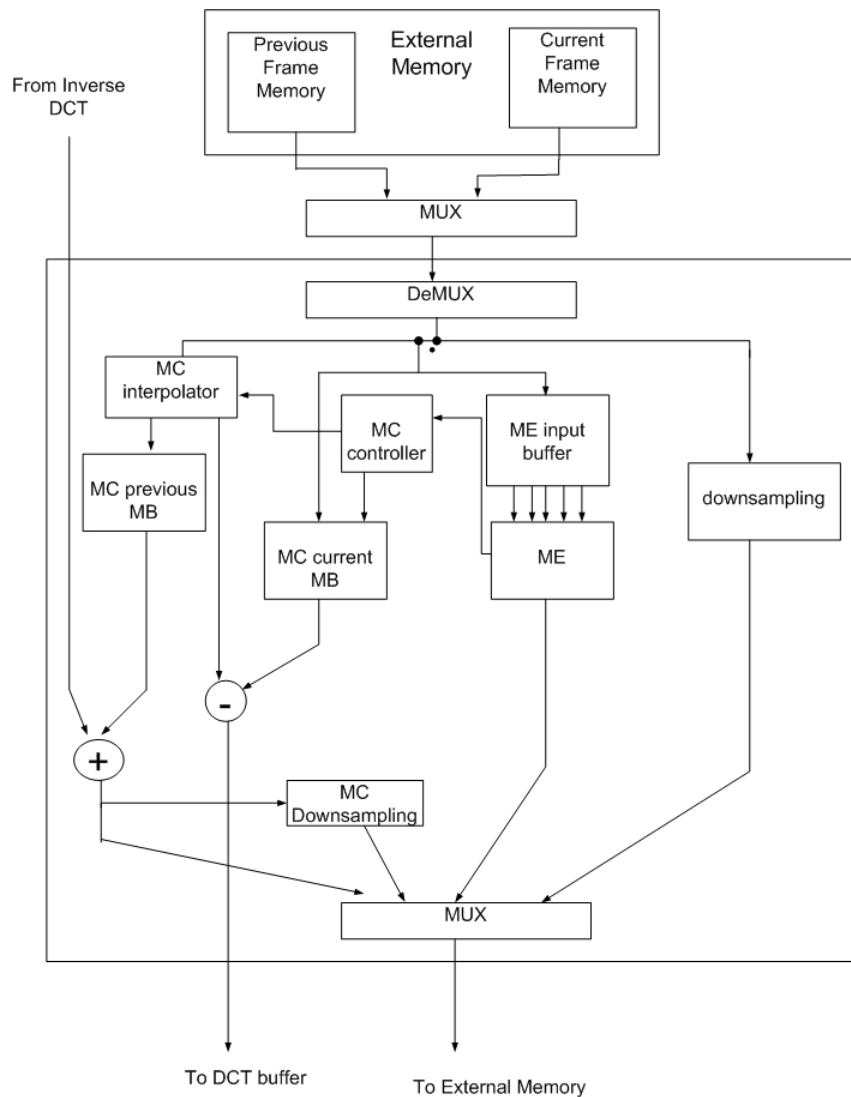


Figure 2-26 Motion unit integration

When the motion estimation (ME) input buffer is fed with the current MB, the “MC current MB” is also fed with the same data. Therefore, the bandwidth can be saved to avoid

re-loading current MB again.

When the ME is finished, the “MC controller” will latch the previous MB from the “External Memory” according to the motion vector got from the ME. Because the UV image is half size of the Y image, the “MC interpolator” must interpolate the UV data if the motion vector is odd. After interpolation, the previous MB data will be stored in the “MC previous MB” temporarily and be subtracted from the “MC current MB” later. The subtracted difference will be sent into DCT buffer.

In the same time slot (please refer to section 2.4 system scheduling), the data from inverse DCT will add with the data in the other “MC previous MB” (Considering the whole MPEG-4 encoder system pipeline scheduling, the “MC previous MB” is a ping-pong buffer mode. That is, when “MC controller” is writing data into the buffer1 in the “MC previous MB” the IDCT could read out the buffer2 in the “MC previous MB” to do reconstruct operation. In the next MB, the “MC controller” will write into the buffer2 and the IDCT will read out the buffer1.) .

Since the HMEA needs Level2 and Level1 reconstructed frame, we design a “MB downsampling” circuit which could do a MB level downsampling without wasting any extra time (we use the interval which IDCT does not output data to the motion unit). In this way, the “downsampling” circuit just has to downsample the current frame in each “P frame” processing time and do not need to downsample the reference frame. The MB relation is shown in Fig. 2-27, and the “MB downsampling” circuit is shown in Fig. 2-28. When the IDCT input 16 pixels into the motion unit, we could get 4 pixels of Level1, and two pixels of Level2. When the “MC downsampling” gets 4 Level1 pixels, it will write out to the “External Memory”. Finally, when the IDCT finish outputing Y1~Y4, U and V data, the Level1 macro block all write out to “External Memory”. The “Level2 data register1 ~ register4” have stored the Level2 macro block, and the “MC downsampling” could write out to the “External Memory”. Therefore, the reference frame of the next P frame is downsampled without

wasting extra time.

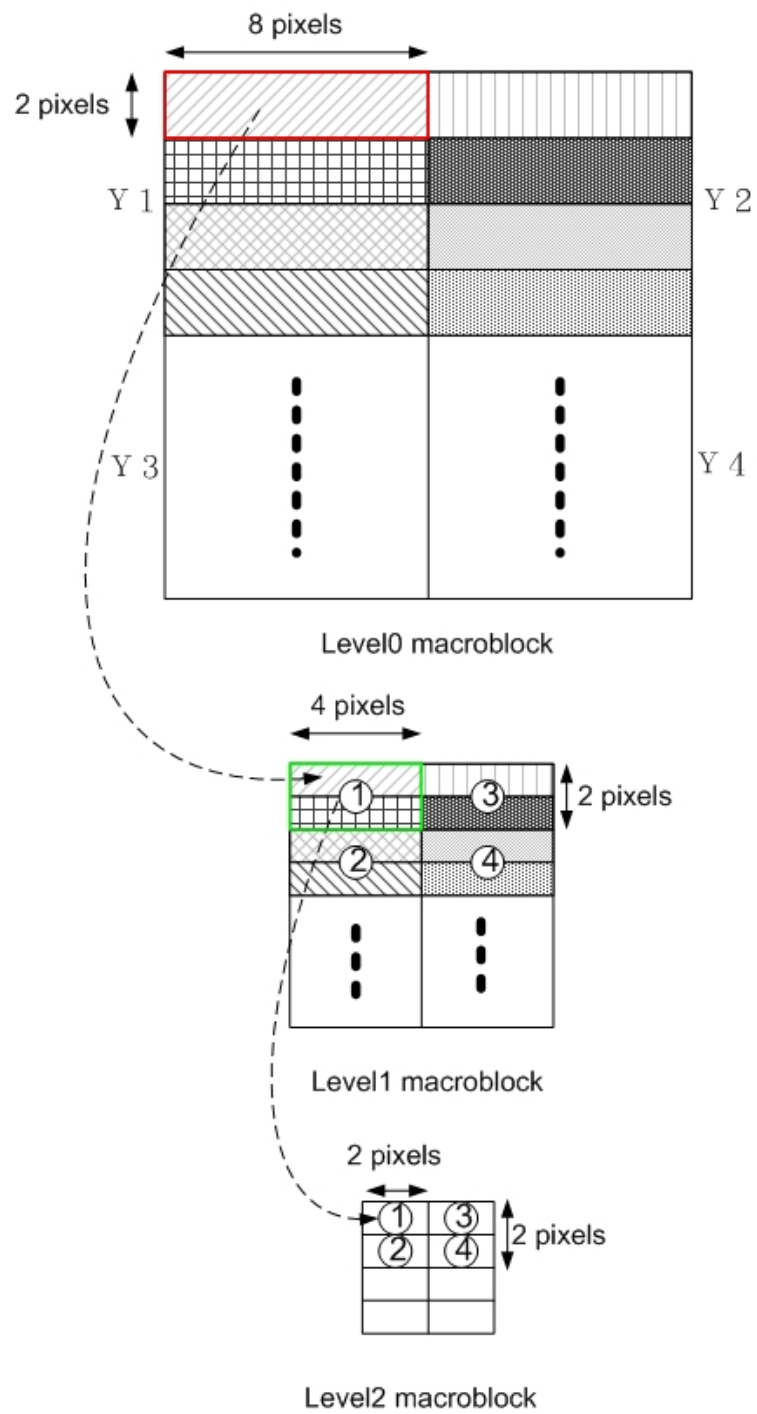


Figure 2-27 Relationship of different level MB

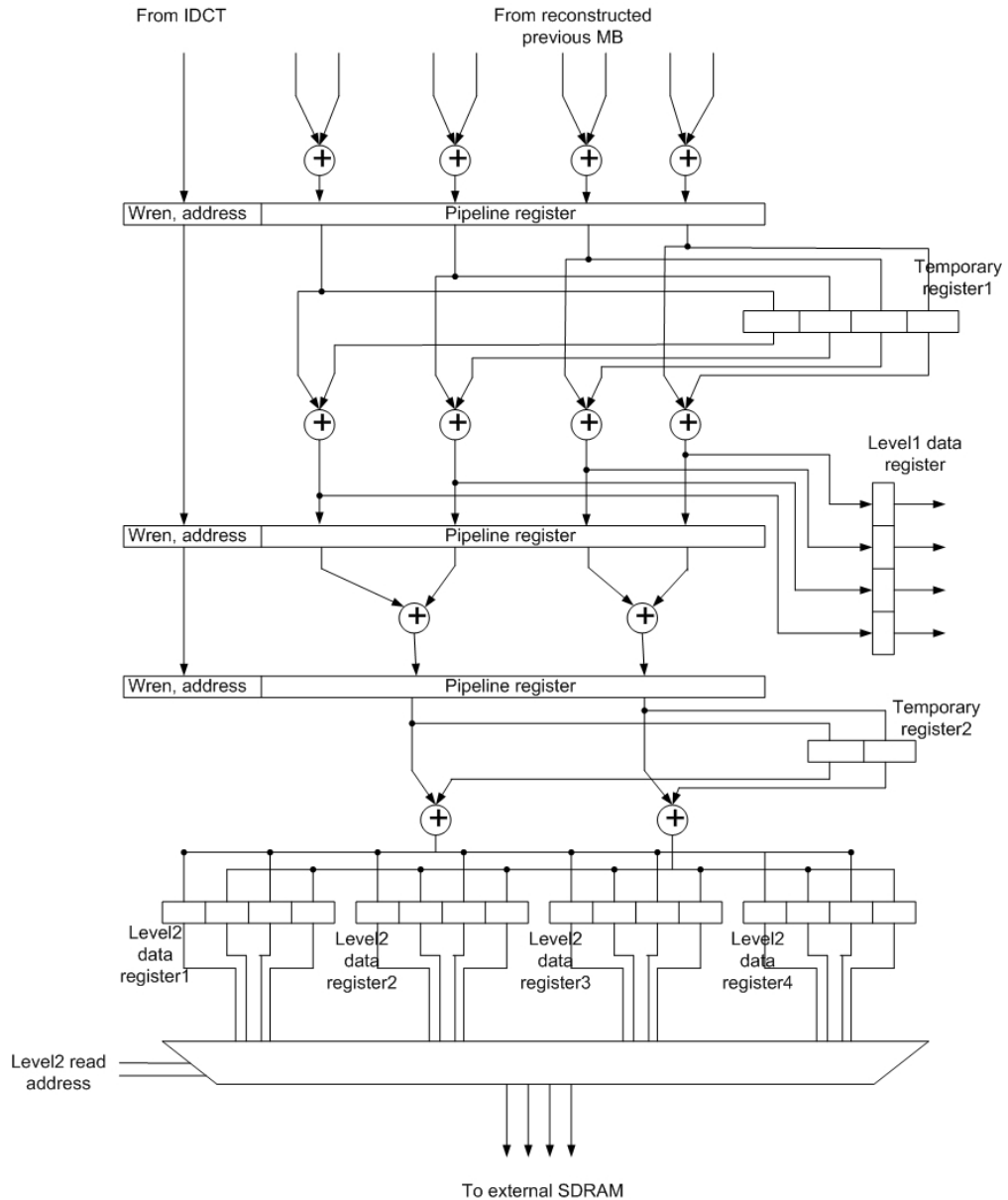


Figure 2-28 MB downsampling circuit

✧ Off-chip memory Organization

In the off-chip memory organization design, two main parts of off-chip memory are used to store two frames, “Mem1” and “Mem2”. For the I frame, the frame source is always the “Mem1”, and the IDCT throughs the reconstructed frame to the “Mem2”. For the 1st P frame, the current frame is stored in the “Mem1”, and the reference frame is stored in the “Mem2”. The reconstructed frame is written to the “Mem1”. And for the 2nd P frame, the current frame

is stored in the “Mem2”, and the reference frame is stored in the “Mem1”. In this way, the frame memory status is switch mutually until arrive the next I frame. When it arrives the next I frame, the “Mem1” is switched back to current frame. The switch condition is shown in Fig. 2-29.

Frame Number and Frame Type	Mem1 status	Mem2 status
I Frame	Current Frame	Reference Frame
1st P Frame	Current Frame	Reference Frame
2nd P Frame	Reference Frame	Current Frame
3rd P Frame	Current Frame	Reference Frame
4th P Frame	Reference Frame	Current Frame
5th P Frame	Current Frame	Reference Frame
⋮	⋮	⋮
I Frame	Current Frame	Reference Frame
1st P Frame	Current Frame	Reference Frame
2nd P Frame	Reference Frame	Current Frame
3rd P Frame	Current Frame	Reference Frame
⋮	⋮	⋮

Figure 2-29 Memory organization

2.3 Variable Length Coding Architecture

Design

2.3.1 VLC Hardware Architecture Design

The MPEG-4 bit stream syntax hierarchy is shown in Fig. 2-30. The video object

sequence delivers the complete MPEG-4 visual scene, which may contain 2-D or 3-D natural or synthetic objects. The video object consists of video object layers, the video object layer can be of arbitrary shape corresponding to an object or background of the scene. The group of video object plane is an optional level which is used for random access or error-resilience. The video object plane is a snapshot of a video object at a particular moment. The macro block is a 16x16 block in a frame. The block is an 8x8 block in a macro block. The computational complexity is dramatically increased in the macro block and block level. The bit stream processing is partitioned into two parts. The hardware is responsible for macro block and block level bit stream processing, and the software or firmware is responsible for the other higher levels. The hardware/software bit stream work partition is shown in Fig. 2-31

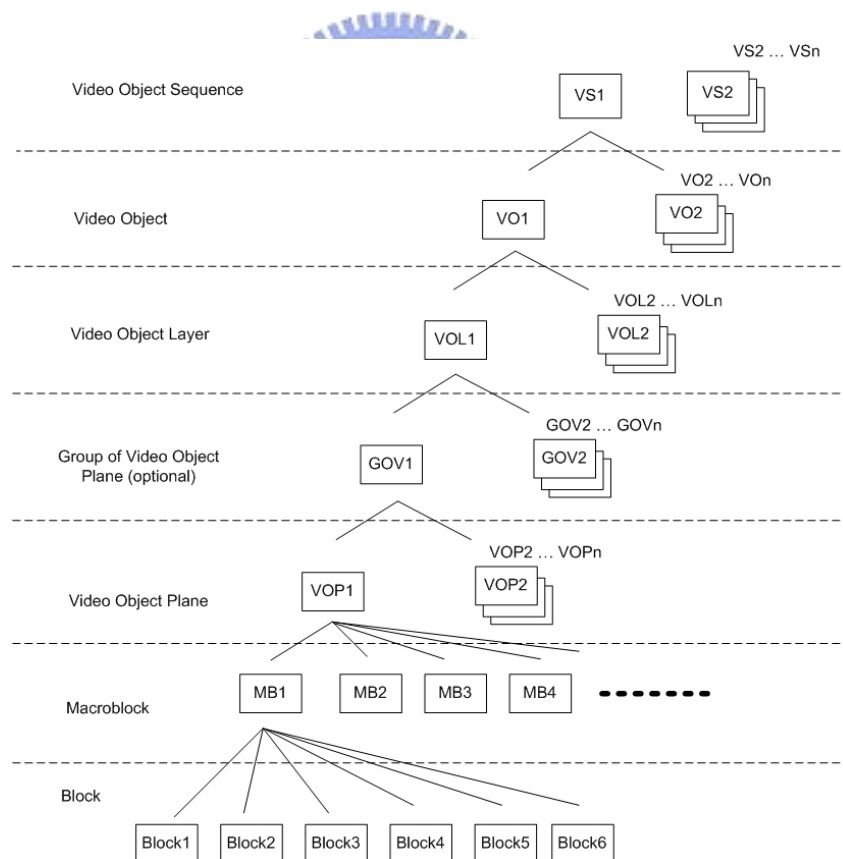


Figure 2-30 MPEG-4 bitstream syntax hierarchy

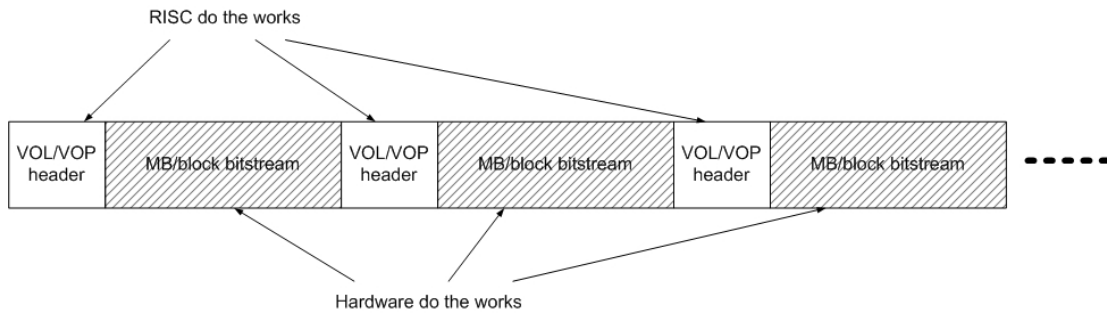


Figure 2-31 Hardware/Software bitstream partition

The VLC hardware architecture is shown in Fig. 2-32. Considering the overall encoder system scheduling, the VLC module has a ping-pong buffer, so that the VLC could achieve pipeline work with motion unit and texture coding. In the ping-pong buffer, we design a re-order mapping mechanism to support the three kinds of scan order in I frame (please refer to the section 2.1 “MPEG-4 Video Encoder Algorithm Overview” and Fig. 2-7). When the DC predict from vertical, the alternate-horizontal scan is used. When the DC predict from horizontal, the alternate-vertical scan is applied. When the I frame predict flag is zero or in P frame mode, the zig-zag scan is used. On average, the AC/DC prediction could improve 14% of I frame coding efficiency. In order to use an efficient method to implement these three kinds of scan order without wasting additional hardware cost. The alternate-horizontal and alternate-vertical scan have some regular relationship in the memory addressing between each other. The texture coding write in one of the buffers of VLC component, and the write order is shown in Fig. 2-33. In next “time slot” (please refer to the section 2.4 “System Scheduling”), the RLC module sends a sequence of number from 0 to 63 to read out the qcoefficient. In the “Qcoefficient RAM”, a simple case statement is used to re-mapping the read address. The alternate- horizontal and alternate-vertical read address is simply exchanged the lowest three bits and the highest three bits with each other (which is shown in Fig. 2-34). So only the zig-zag scan and alternate-horizontal scan re-order mapping implementations are needed and three kinds of scan order can be achieved.

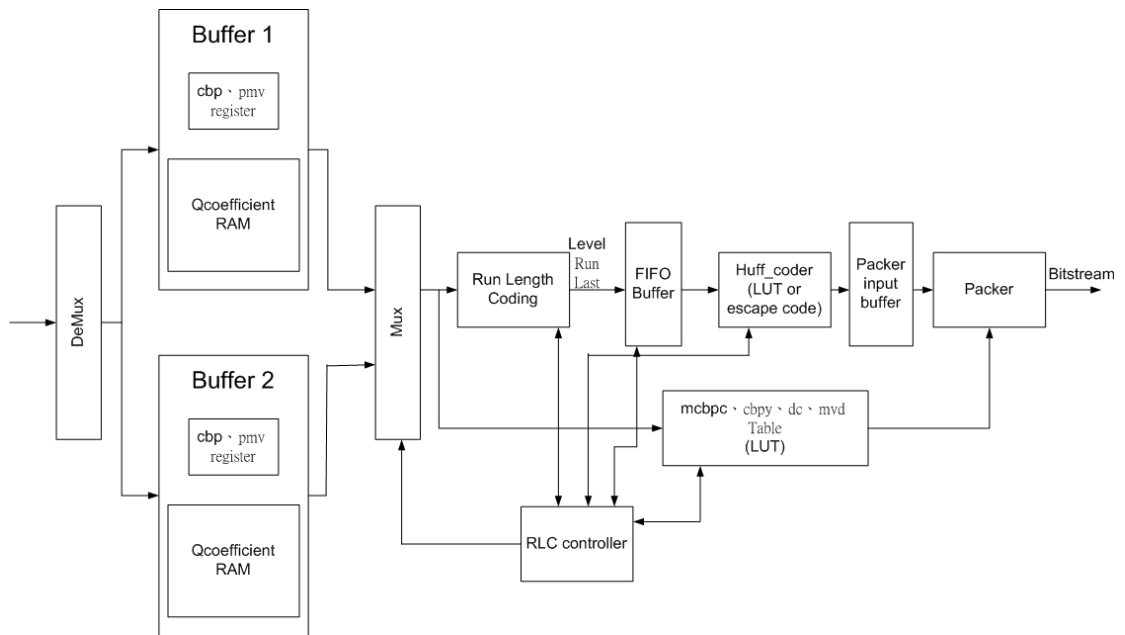


Figure 2-32 VLC architecture

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63

Figure 2-33 Texture coding write-in order

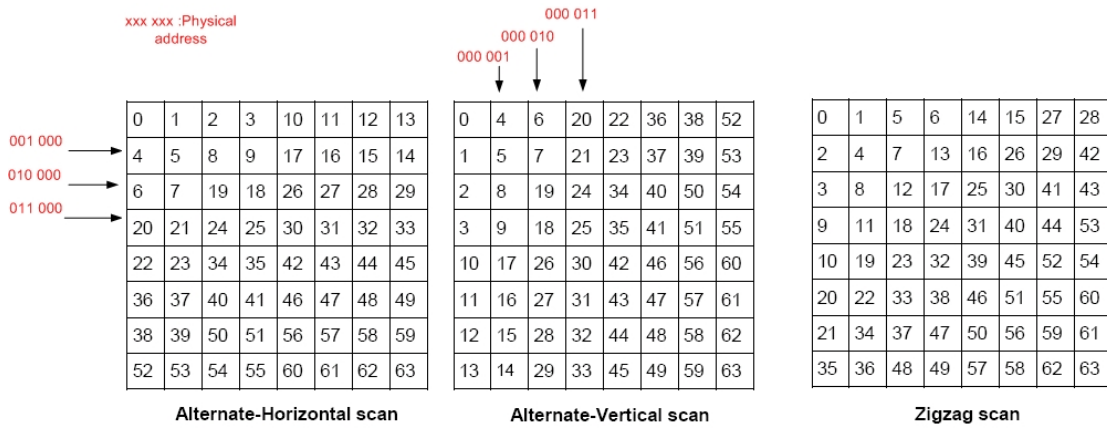


Figure 2-34 VLC memory read-out order

The “RLC controller” is a finite state machine, which will read out the qcoefficient from VLC ping-pong buffer into the “Run Length Coding” module and control the DC、cbp、mvd look up table operation. The “Run Length Coding” will generate a 3-D symbol that is the (Run, Length, Last).

Run : This number represents how many zeros has been encountered in the scan order before each non-zero term arrived.

Level : This number represents the absolute number of the non-zero term.

Last : This bit represents if this symbol is the last one in a block.

Because the “Last” parameter can not be decided until encounter the last non-zero term or last qcoefficient in a block, a specific FIFO buffer is adopted between the “Run Length Coding” and “Coefficient Table”. This specific FIFO architecture is shown in Fig. 2-35.

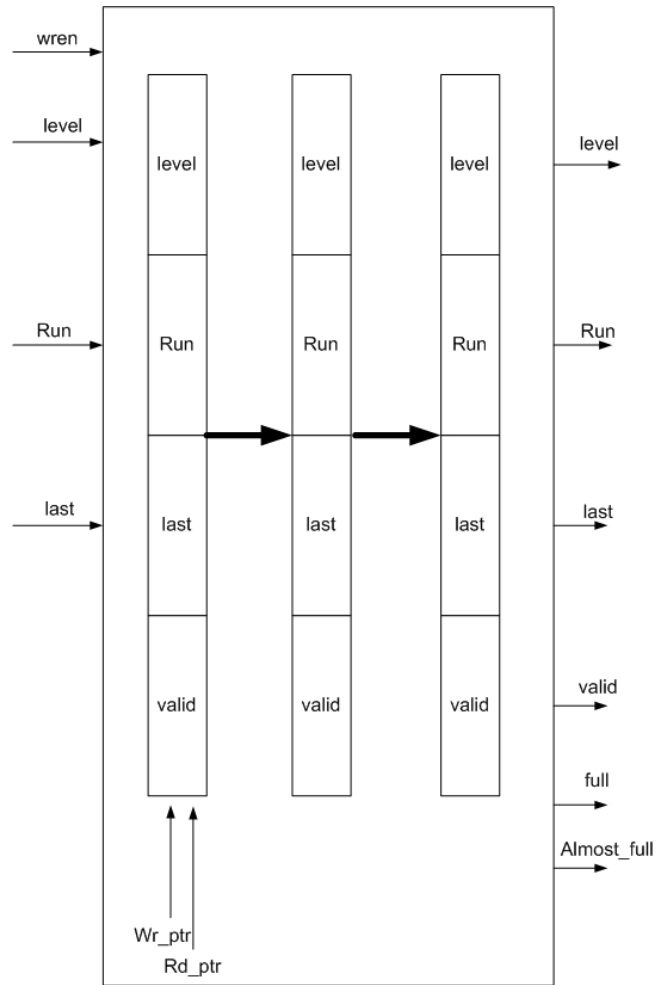


Figure 2-35 RLC FIFO architecture

The “full”, “almost_full” are signals to indicate “RLC controller” whether stop read out the qcoefficient or not. And the valid output is a signal to indicate the “Huff_coder” to fetch the (Level, Run, Last) symbol on the output port of the FIFO when there are more than two or equivalent to two successive (Level, Run) symbols in the FIFO register. When the “RLC” encounters the last qcoefficient, the “last” input port will be set to HIGH to set the “Last” register of the last symbol.

The “Huff_coder” is mainly doing a look up table (LUT) for one symbol. The MPEG-4 use 3-D LUT, that is, there are 3 parameters to map one symbol to a unique code word. When the symbol does not exist in the specific table, it will be encoded by escape mode. The MPEG-4 standard code table is analyzed and the arrangement of the distribution of encoding

mode which is shown in Fig. 2-36. The three kinds of escape mode encoding method is described as following :

Escape type1 mode : When the code word is not in the normal mode table (Fig. 2-36 black color area). Looking up table for a LMAX [1] and calculate the $Level^+$, $Level^+ = sign(Level) * (abs(Level) - LMAX)$ and use the (Level⁺, Run, Last) to do LUT. When a symbol is on the cyan color area of Fig. 2-36, the escape type1 mode is used. And the bit stream is in the form of 0000011 0 *code word* .

Escape type2 mode : When the code word is not exist in the code table when using escape type1 mode. Looking up table for a RMAX [1], and calculate the Run^+ , $Run^+ = Run - (RMAX + 1)$ and use the (Level, Run⁺, Last) to do LUT. When a symbol is on the red color area of Fig. 2-36, the escape type2 mode is used. And the bit stream is in the form of 0000011 10 *code word* .

Escape type3 mode : When the code word is not exist in the code table when using escape type2 mode, using the escape type3 mode to encode this symbol. The code word is in the form

ESCAPE	Follow code	last	run		level	
--------	-------------	------	-----	--	-------	--

of 0000011 11 x xxxxxx 1 xxxxxxxxxxxx 1 . When a symbol is on the white color area of Fig. 2-36, the escape type3 mode is used.

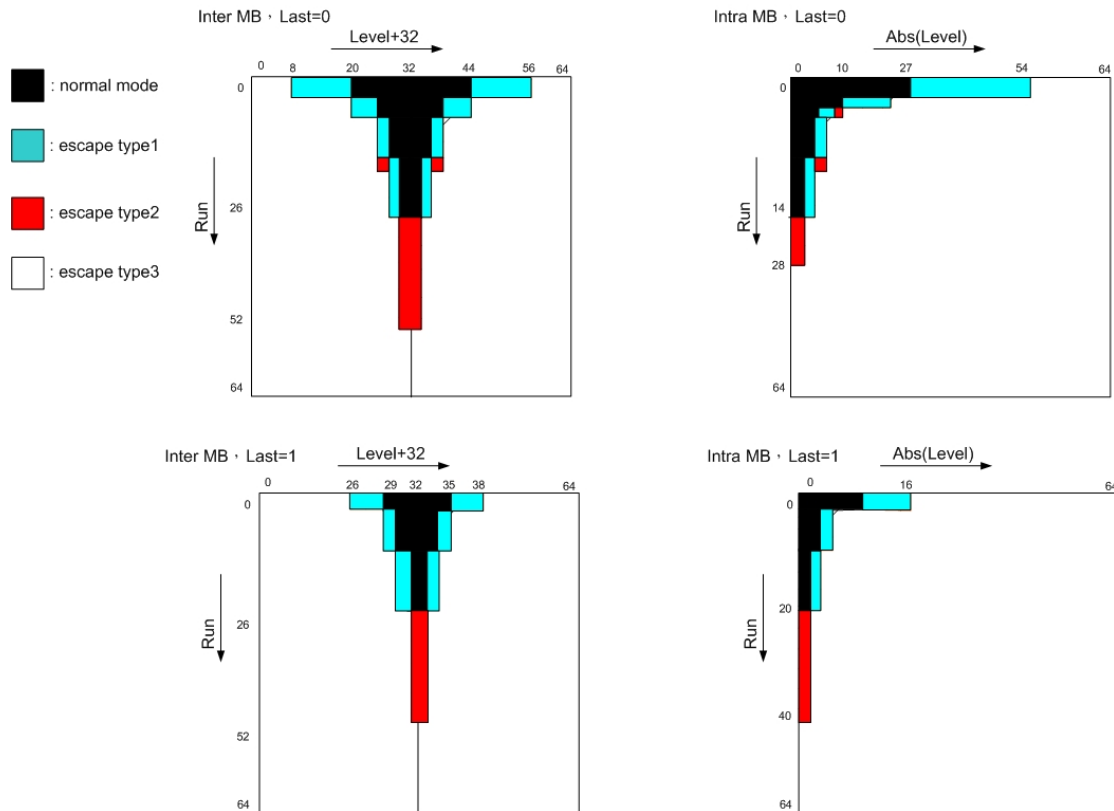


Figure 2-36 Encoding mode distribution

The “Huff_coder” will output one stream word to the “packer”. The packer is responsible for packaging bit stream into 32-bits packet and writes out to the external memory. Due to the irregular length of code word, a FIFO buffer is used between the “Huff_coder” and the “Packer”. The architecture of “Packer” is shown in Fig. 2-37. When the “Packer” read in one code word, the code word will be stored in the “D1” register and the length of this code word will be stored in the “D2” register. Considering the area and timing issues, two 16-bits barrel shifters are adopted instead of two 32-bits barrel shifters. The “Barrel shifter 1” and the “Barrel shifter 2” are both right shift window, which would locate the code word according to the code length and threw the located code word to “D3” register and the “D4/D5” register. The “D3” register contains the effective residue bits which hasn’t put into the “D4/D5” register, and the “D6” register stores the effective residue bit numbers in the “D3”. The VLC hardware implementation results are shown in Table 2-7.

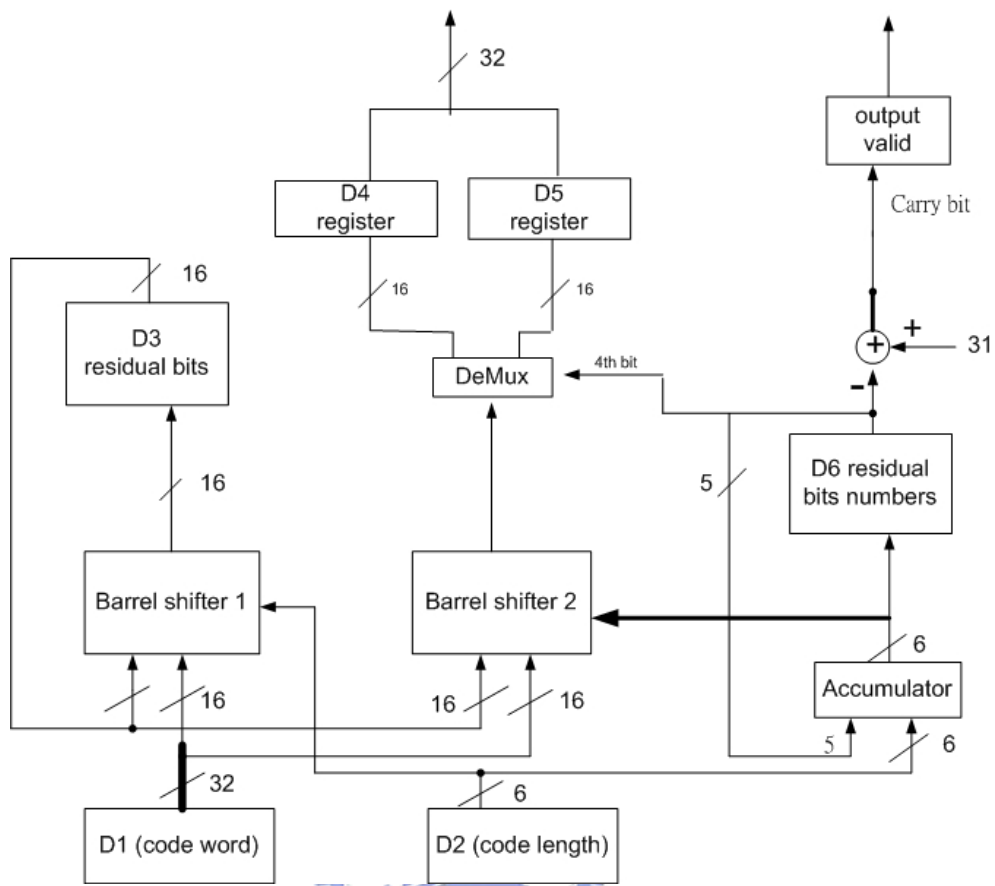


Figure 2-37 Packer architecture

Table 2-7 Variable length coding ASIC implementation result

Process	UMC 0.18 um
Max. Operation Frequency	85 MHz
Power	125 mW
Gate count	8860 gates

2.4 System Scheduling

After analyzing the clock cycles needed for processing one macro block, three stages pipeline scheduling is applied in the whole MPEG-4 video encoder system. The “Motion

Unit”, “Texture Coding” and “VLC” are three pipeline stages respectively. The whole encoder system architecture is shown in Fig. 2-38. The “controller” is a finite state machine which could control these three components and assurance that they could work together compatibly. Besides, the controller is also responsible for handling the data flow and multiplexer control in different frame type (I frame and P frame).

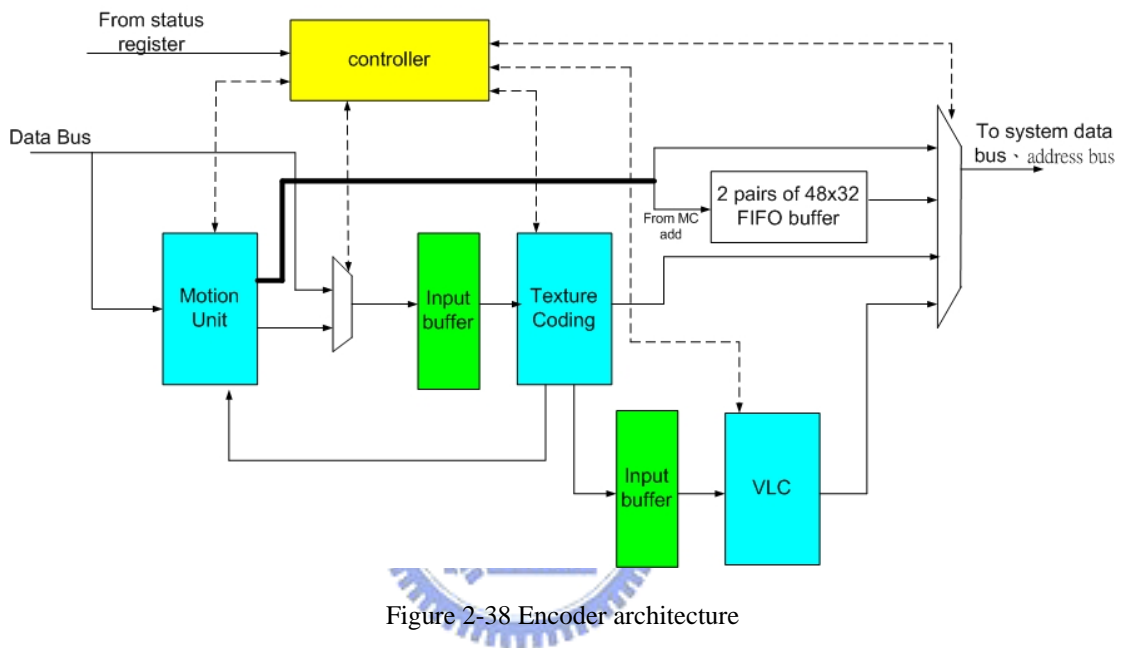


Figure 2-38 Encoder architecture

The controller finite state machine is shown in Fig. 2-39. When the controller jump out from the idle state, it will initialize the VLC internal register in the `init_vlc_st` state.

In I frame: The controller jump into a loop between the `i_text_en` state and the `i_text_vlc_en` state. After 396 loops, it jumps to the `dn_en` state to downsample the reconstructed frame. After finish downsampling, it writes `0x03` to the DMAC status register to start the DMAC to move bitstream from the SSRAM to the SDRAM at the `finish_frame_st` state. Then it goes back to idle state.

In P frame: The controller must downsampling the current frame first at the `dn_en` state. Then it jumps into the loop between the `p_ME_text_en` state and the `p_text_vlc_en` state. After 396 loops, it jumps to the `finish_frame_st` state to writes `0x03` to the DMAC status

register to start the DMAC to move bitstream from the SSRAM to the SDRAM and then goes back to the idle state.

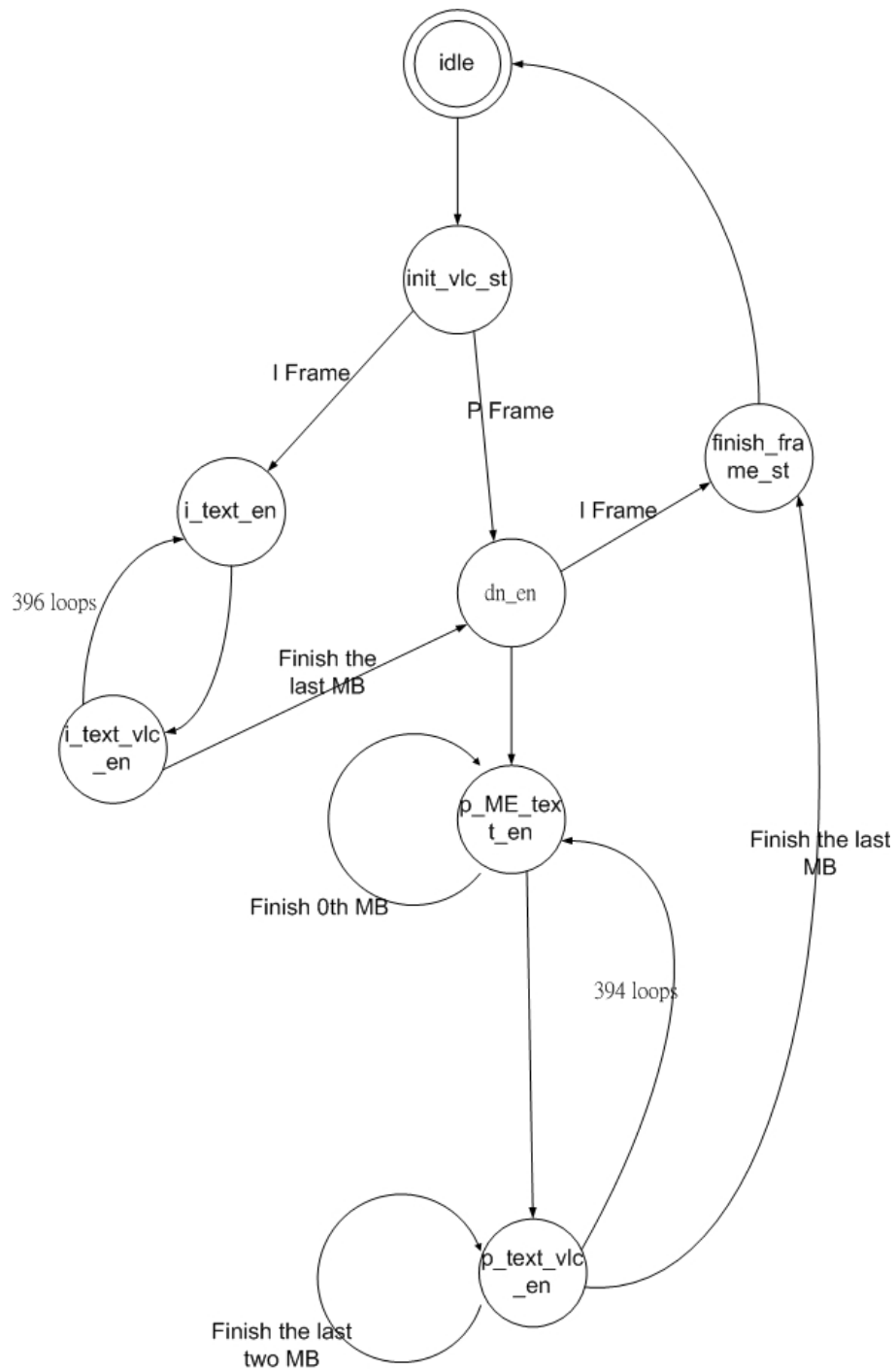


Figure 2-39 Encoder controller finite state machine

Fig. 2-40 shows the pipeline scheduling in the “Intra MB” and the “Inter MB” mode. The “IDCT” output timing is distributed in a whole time slot (the duration of two dotted line is

called “1 time slot”) uniformly. The “VLC” has highest priority to use the system bus, and the ME has second priority to use the system bus. Two pairs of FIFO buffer is adopted, the reconstructed frame is written into these two pairs of FIFO. When the system bus is free, the data in the two pairs of FIFO are written to external memory. In this scheduling, each time slot is 1200 cycles on average (the time slot length is depend on the cycle time which VLC occupies the bus). We suppose the SDRAM latency is 5 cycles at 20MHz (that is 250 ns latency time per word). Combining the downsampling cycle number, this MPEG-4 video encoder could achieve 30 CIF images encoding in 21 mega clock cycles.



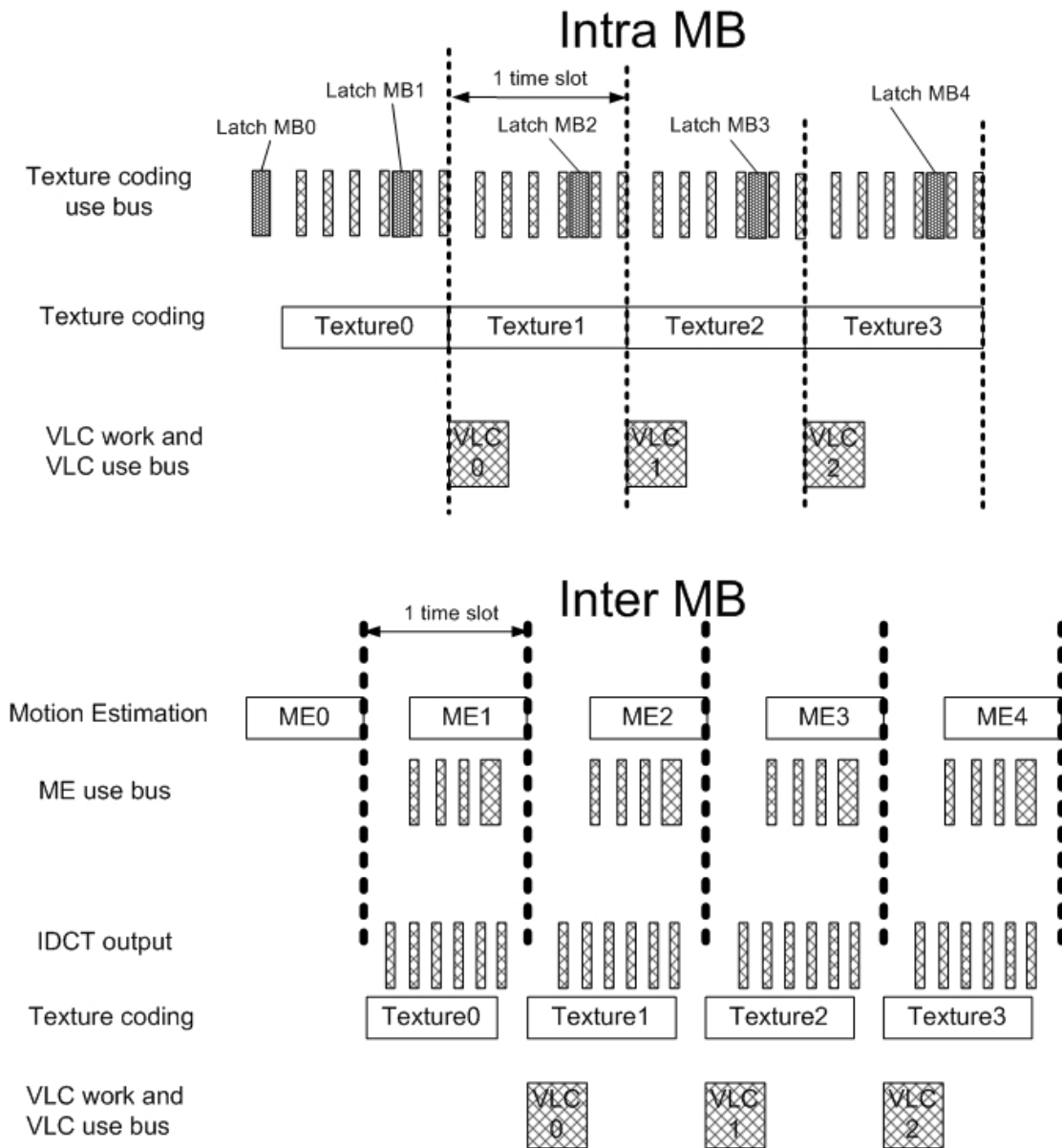


Figure 2-40 Encoder scheduling

2.5 Summary

In this chapter, an enhanced hierarchical motion estimation architecture is proposed, and MPEG-4 variable length coding hardware architecture is proposed to handle the irregular stream length smoothly. All modules pipeline work with each other in the MPEG-4 video encoder system such that it could achieve 30 CIF images encoding in 21 mega clock cycles.