An Efficient VLSI Computation Reduction Scheme in H.264/AVC Motion Estimation

Shikai Zuo, Mingjiang Wang, Liyi Xiao
Microelectronics Center, Key Laboratory of the Technology of the Internet of Things
Harbin Institute of Technology
Harbin, Heilongjiang
China
zuoshikai@yahoo.cn, http://www.hitsz.edu.cn

Abstract: - The variable block sizes motion estimation in H.264 is key technique to remove inter-frame redundancy. This technique not only requires huge memory bandwidth but also its computation complexity is higher. Therefore, this paper proposes one efficient sub-pixel search algorithm for reducing computation complexity and bandwidth utilization, and a novel VLSI architecture for this algorithm which simplifies variable block sizes motion estimation. The proposed method is efficient compared with those of existing methods which have negative effects on compression, with respect to chip area, operation frequency, and throughput rate. The proposed sub-pixel search architecture decreases the numbers of search pixels of full pixels motion estimation by around 70% and the chip area by around 40% than the others search algorithm. Besides, an optimized motion estimation MV prediction algorithm is used to remove data dependency, and optimization storage policies are used to save hardware resources. The proposed sub-pixel search architecture can work at 200 MHz with 530k gate count, which supports high-definition television 1920×1080 format.

Key-Words: - sub-pixel search, systolic array, H.264 encoder, motion estimation, 1080P, HDTV, VLSI

1 Introduction

The H.264 Advanced Video Codec is an ITU standard for encoding and decoding video with a target coding efficiency twice that of H.263 and with comparable quality [1]. An increasing number of services and growing popularity of HDTV are creating much more need for higher coding efficiency. An important coding tool of H.264 is the variable block size matching algorithm for the ME (Motion Estimation) which is a part of the prediction step [2]. Because of the use of variable block-matching motion estimation (Variable Block Sizes Motion Estimation, VBSME), multiple reference frame motion compensation (Motion Compensation, MC) and Lagrange rate-distortion optimization (Rate Distortion Optimization, RDO) and other advanced coding techniques, making the integer pixel motion estimation (IME) and fractional pixel motion estimation (FME) consisting of inter-frame motion estimation process takes up more than 70% of the entire encoder encoding computing time [3]. So the integer pixel motion estimation is the bottleneck of H.264 encoder hardware implementation [4]. The key problem of the H.264 encoder for HDTV is that the bandwidth of memories access is limited.

Many efficient techniques have been used to reduce the complexity, for example Full-Search motion estimation, UMHexagon search, NTSS search etc by JM software. At the same time many hardware architecture have been proposed by some researchers. The authors researched full-pixel search motion estimation hardware implementation includes Anchao Tsai, Mohammed Sayed and Weifeng He etc. In paper [5], the authors present an efficient architecture design based on the search point reduction for HDTV variable block size ME of H.264/AVC. The hardware architecture is implemented with the 2-D systolic array and it successfully increases the coding speed at the expense of hardware cost. The 2-D systolic array successfully reduces the data reuse for pixel SAD computation, but it increased the number of control circuits for its complexity. In paper [6], the authors researched parallel-pipelined architecture based on full search block matching algorithm, proposed an architecture consisting of two main parts: the SAD computing part and the SAD comparing part with pipeline registers between them and a control unit to control their operation. Using the techniques of pipeline circuit and reducing supply voltage reduce the power consumption and simplify the control circuit. But the drawback of this technique is that the data reuse rate is low for reading data from storage. The full-search algorithm exhaustively computes all candidate blocks to find the best match within a particular window [7]-[9]. Therefore, this technique has enormous complexity. In order to reduce the motion estimation complexity, many fast searching algorithms are presented [10]-[13], but they have not perfect solutions. In paper [10] and [11], the authors use fast ME algorithm called HMDS to reduce bandwidth, but the hardware implementation of HMDS algorithms need more logic circuits. In paper [12] and [13], the modified three step search (TSS) algorithm is used to reduce the computational cost and the memory access in the motion estimation part. Those fast ME algorithm can dramatically reduce the search points, but the efficiency of VLSI architecture is decreased because of the lack of regularity. So the most VLSI implementations of motion estimations adopt full-search mode for regular designs [14] [15].However, such full-search chips are not suitable for portable systems due to more bandwidth and power consumption for HDTV.

This paper proposes an efficient sub-pixel search algorithm for variable block-matching motion estimation. The efficiency of the sub-pixel search cooperated with a simplified predicted MV is verified for H.264/AVC encoders. We found that the sub-pixel search ME can reduce hardware consumption around 40% compared to JM reference software with negligible video quality loss. To realize the sub-pixel search algorithm, the VBSME architecture is designed by using a 1-D systolic array. Thus, the proposed architecture can compute the optimal MV more efficiently than the existing ones found in the literature. The proposed VBSME architecture with input memory array includes the sum computation of absolute difference (SAD) and Lagrangian cost function. Simulation results demonstrate that the proposed scheme has better coding performance than conventional architectures.

The remainder of this paper is organized as follows. Section II introduces the proposed SPR algorithm. Section III presents the proposed very large-scale integrated VLSI architecture for VBSME implementation using SPR method. Section IV, presents the experimental results using several video sequences, in order to verify the effectiveness of the proposed SPR algorithm. Conclusions are finally drawn in Section V.

2 Proposed Sub-pixel Search Algorithms

In video encoding systems, the motion estimation (ME) can remove most inter-frame redundancy, so a high compression ratio can be realized. Among various motion estimation algorithms, fast full-search algorithm is usually used because of its perfect effect and regular computation [16]. Thus, we propose an efficient sub-pixel search algorithm for variable block-matching motion estimation, in order to reduce the bandwidth and power consumption.

2.1 VBSME Algorithm

In a typical VBSME, each frame of a video sequence is divided into a fixed number of no overlapped square blocks [17]. The search for the best matching block is compared with the previous frame search area. The sum of absolute differences (SAD) is used as the main metric. SAD is shown (1) as below:

$$SAD(dx,dy)=\sum_{m=x}^{x+N-1}\sum_{n=y}^{y+N-1}|I_k(m,n)-I_{k-1}(m+dx,n+dy)|$$

$$(MV_x,MV_y)=(dx,dy)|_{\min SAD(dx,dy)} \qquad (1)$$

In the H.264/AVC motion estimation, the current frame is divided into a number of small macroblocks(MBs) of size 16×16. The tree structure motion estimation method is used in the H.264 VBSME. As shown in figure 1, the luminance MB size of 16×16 may be divided into 16×8, 8×16, 8×8, and 8×8 block also may be divided into 4×8, 8×4, 4×4 blocks [18]. The tree structure motion estimation method in VBSME can adopt many combinations of variable block-sizes to match the shape of different objects in the video frame [19]. As a result, the coding efficiency of the tree structure motion compensation method is better than the previous approaches. Although it can achieve a higher compression ratio, it not only requires large computation complexity but also needs huge memory bandwidth for HDTV.

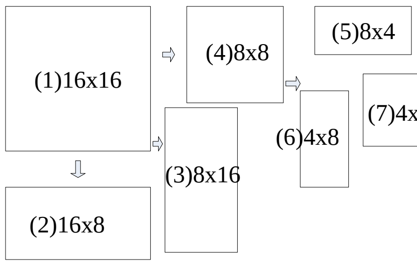

Fig.1. The tree structure motion estimation for H.264

2.2 Proposed Sub-pixel Search Algorithm

In order to reduce the bandwidth and power consumption, the paper proposed the Sub-pixel Search Algorithm base on similarity value of adjacent pielxs.

2.2.1 Sub-pixel Extraction

We compress the variable macroblocks data into a quarter of original MB of image according to similarity of adjacent pielxs. Simultaneously, we obtain sub-pixel(SuP) on the basis of regulation (2), as below:

$$SuP = MIN\left\lfloor \max(P_{(i,j)}, P_{(i,j+1)}), \max(P_{(i+1,j)}, P_{(i+1,j+1)}) \right\rfloor \text{--}(2)$$

Accordingly each frame size of M×N of a video sequence is compressed into to a quarter of original frame, and the number of each macroblock pixel ponits is reduced to a quarter of original MB. So the computing load is enormous reduction in VBSME. As shown in the figure 2.

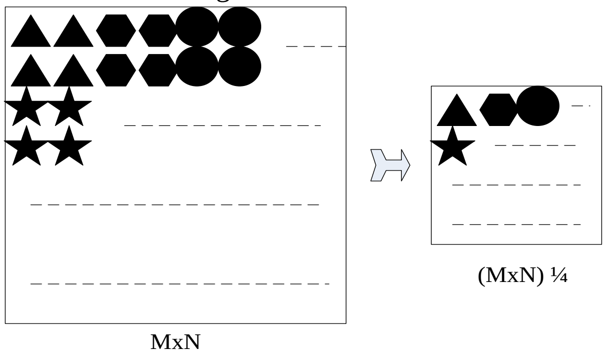

Fig.2. The sub-pixel extraction

2.2.2 Hierarchy search mode

We adopt hierarchy search mode to achieve sub-pixel search algorithm, which has two steps: First sub-pixel points search, then full-pixel search.

In the sub-pixel points search, the current MB and the corresponding reference frame use the sub-pixel data. In addition, the tree structure motion estimation method which has 7 partition block patterns can reuse the smaller blocks' SADs to calculate the larger blocks' SAD of an 8×8 block in parallel. According to the formula (2), each current MB size is changed to a quarter of the original MB. As shown in figure 3, we first calculate the a(2×2) block SAD, then calculate the b(4×2) and c(2×4) blocks by combinations of corresponding a(2×2) blocks. So we can achieve SAD of d(4×4), e(8×4), f(4×8), g(8×8) blocks by this method.

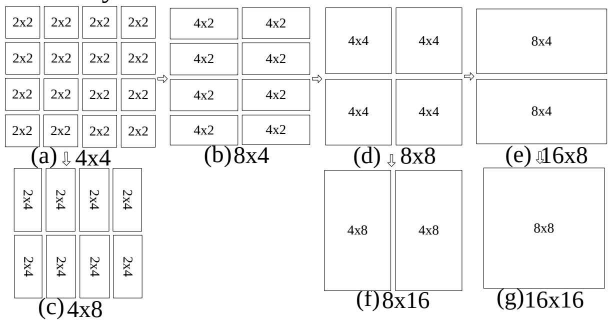

Fig.3. Different sub-blocks of a 16×16 MB for H.264/ AVC

In the full-pixel search, the current MB and the corresponding reference frame use the full-pixel data. In addition, this step only searches eight points surround the sub-pixel that computed in step one.

2.2.3 Optimize predictions of motion vector

In H.264/AVC, predicted motion vectors of different block sizes have different values as described in [20]. However, it is very complex for hardware implementation, because the data dependency of the neighboring MBs makes the 1-D systolic array difficult to implement. In order to reduce the data dependency of the neighboring MBs, the author proposed a modified PMV strategy in paper [21]. This strategy is that the PMV values of all types of block size are calculated from the median of left MB, up MB, and right-up MB. However, we have to slightly modify this design for sub-pixel search implementation. The proposed PMV values of all types of block size are calculated from the median of left-down MB, up MB, and right-up MB. Therefore the predictions of motion vector should have common values for each MB and its sub-blocks.

3 Hardware Architecture Design

3.1 Overview of the Architecture in H.264/AVC

The MB is the basic unit of operation in the video frames processing of H.264 encoder, it includes a 16×16 luminance MB and a 8×8 chrominance [22]. The encoder is organized as a pipeline architecture processing the data on a MB basis. The pipeline has four stages. As shown in figure 4, the first stage consists of the sub-pixel motion estimation (ME), the second stage involves the integer ME and fraction ME, the third stage consists of the motion compensation (MC) and the intra prediction and the TQ and the Inverse TQ, the fourth stage involves the Deblocking Filter and Entropy code. At last the code stream data are output. The Sub-pixel ME and full-pixel ME module are discussed only in this paper.

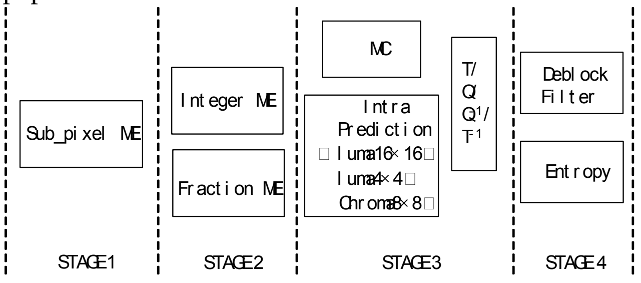

Fig.4. H. 264 encoder hardware architecture

3.2 Overview of the Architecture in Sub-pixel

To achieve the HDTV (1920 × 1080, 30Hz) video encoder Motion Estimation, a high efficient data reusable architecture is presented. Fig. 5 shows the block diagram of the proposed architecture which consists of the Sub-pixel ME and Integer ME module. Additionally, both of the modules are connected with pipeline registers. The Sub-pixel ME pipeline scheme is adopted to divide the proposed architecture into three parts for easy realization. The first part is the data input system which comprises two major modules. One is the Cur_Ram which stores the current data from memory. Another is Sub_Ref_Ram which stores the sub-pixels reference data from memory. The second part is the SAD compute array

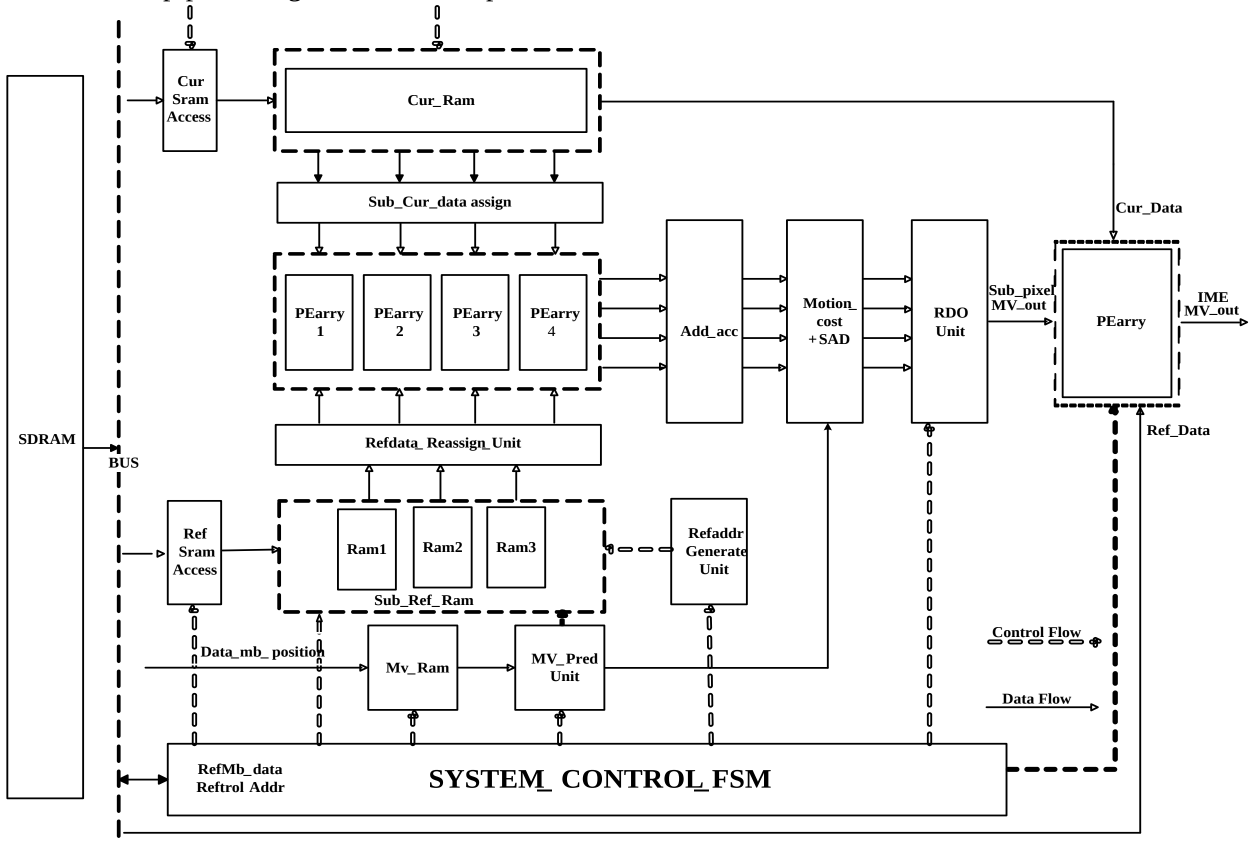

Fig.5 The architecture of H.264 VBSME

which includes PEarrays 1-4 and Add_acc. The third part is the Motion cost and RDO Unit. The System Control Unit, which schedules the pipeline operations, completes the encoder architecture. The pipeline processes each MB as follows: First The luma pixel of the reference video is input to the Sub_Ref_Ram by computing sub-pixel. The luma component of the current MB is input to the Cur_Ram according to current MB Coordinate. At the same time, the Refadder Generate Unit compute the predicted motion vector (PMV) by using the MV of left MB, up MB, and right-up MB. Then the luma pixel data is loaded to the Ram1 Ram2 process element arrays (PEarry) compute the SAD of current pixel and reference pixel. Firstly, the Add_acc tree is used to get the sub-macroblock SAD of variable block-matching. Secondly, the third part computes each block cost including the Motion cost and the corresponding SAD. At last, the best matching block MV is selected by the RDO Unit and output to the Full-PE which is used to search the full pixel point.

3.3 1-D Processing Element Array

To increase process speed of motion estimation and reduce the data reuse for pixel SAD compute, the 1-D systolic array is chosen as the basic architecture of a PEA to compute the SAD of a 2x2 block. As shown in Fig.6 The motion search engine is formed by four PEA arrays which are formed by 16

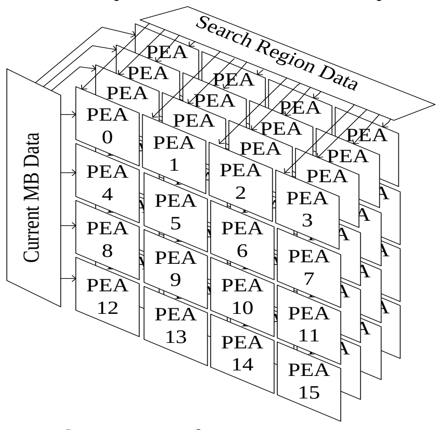

Fig.6 The architecture of PEarrays 1-4

PEAs. So this process engine can obtain the SAD of four macroblocks and its sub-macroblocks at one clock. Fig.7 shows the PEA structure, which is formed by 4 PEs of 16 PEAs. So this structure needs 256 PEs to perform the entire computation. Fig.8 shows the PE structure, which includes the search pixel data register (SRD-REG) and current pixel data register (CMD-REG) and the MV input unit. Because adjacent PEA arrays unit need the same search data and current pixel data, the SRD-REG and CMD-REG can be reused by adjacent PEA-arrays. The SRD-REG is used to hold the corresponding search region data and to change in next pulse. The CMD-REG stores the current data, and it is refreshed once when the Sub-pixel ME compute next MB MV. Firstly, the SAU is responsible for calculating the absolute difference between current block pixel and search area pixel. Then the SRD-REG is propagated to the next PE search data register (NSR), when the current block SAD is computed completely.

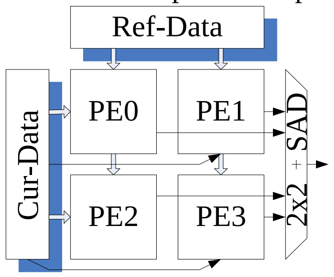

Fig.7 The architecture of PEA

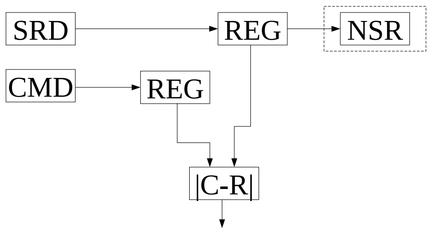

Fig.8 The architecture of PE

3.4 Data Flow of PEA arrays

The ME engine needs search pixel data (SRD) and current pixel data (CMD), which are read from the Sub_Ref_RAM and Cur_RAM. As shown in fig.9, the current_MB sub-pixel register of PEA arrays is assigned in left-to-right order when encoding one MB. There are 64 CMD-REG per sub-pixel MB, which is partitioned into 8 columns. For example, the first column C(i,0) is assigned firstly, next is C(i,1),...the last is C(i,7). The data is loaded in CMD-REG registers within 8 clocks cycles.

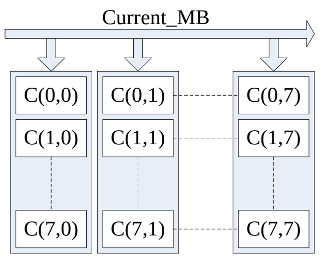

Fig.9. CMD register

We designed the architecture as shown in fig.6 for increasing reusability for the search region data. As shown in fig.10, 88 SRD registers marked as S(0,0), S(0,1)...S(10,7) are used to store the search region pixel data. The operating process of VBSME engine is as following: firstly the SRD data is loaded in RAM1, RAM2 and RAM3. Then the S(0,0), S(1,0), S(2,0)...S(10,0) are loaded in first cycle, in the next cycle the data stored in first column register are shifted to the second column, and next cycle all data in the SRD register are shifted to next column. By this way, the search region data is refreshed in proper timing.

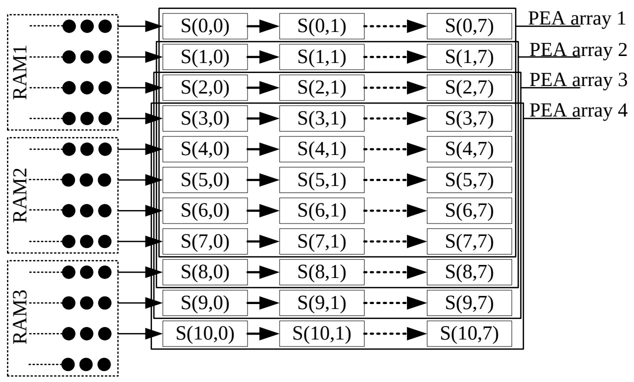

Fig.10. SRD register

3.5 Merge Module

The advantage of the Sub-pixel Search Algorithm is that each block SAD of variable block sizes partitions can be computed at the same time. By this means the computation complexity is simplified for variable block-matching. As shown in fig.11, PEA0, PEA1,…PEA15 are SADs with each block of 2x2 partition. The merging block of PEA0 and PEA1 is a SAD of 2x4 partition, the merging block of PEA0 and PEA4 is a SAD of 4x2 partition, the merging block of PEA0, PEA1, PEA4, PEA5 is a 4x4 partition. The SAD of 8x4, 4x8, 8x8 partition are calculated in the same way. A carry look-ahead adder is used to compute the SAD for avoid the long critical path.

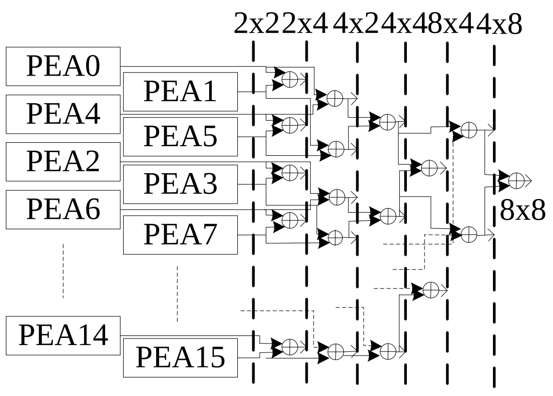

Fig.11 Architecture of the SAD

3.6 Sub-pixel search memory organize

As shown in Fig.12, the proposed hexagon search region replaces the rectangle region for reducing the search points in VBSME. The rectangle region is used in previous design. For example, the rectangle search region (128x64) is used in paper [23] and another paper [24] uses the square search region (65x65). There are 2112 pixels when the hexagon search region is adopted. The search pixels are reduced 31.5% from the square search region. So this method tremendously increases the processing speed without affecting the performance of video coding.

In scheduling, each PE SRD must propagate its processed data to the neighboring PE. The interconnections between neighbor PEs and data holding memory are required. In order to simplify the design of SRD memory, we proposed to use three RAMs to store the SRD. As shown in Fig.10, each RAM could store 4 lines of pixels. According to the maximal line of 72 pixels, we used three 2k (72 × 4 × 8bits) on-chip RAMs for the SRD to store each strip data at different instants. Consequently, we divided the search region into 14 independent regions as illustrated in Fig.12. Each independent region is called a strip. The search directions are from left to right and from top to bottom as shown in map.

Just as shown in Fig.13, we need nine steps to fetch the entire SRD data from memories. In the first step, the data of 0th, 1th and 2th strips will be mapped to RAM1 RAM2 and RAM3. In this step, the SAD of 1th strips MB are calculated and output the results. The following strips are required when motion search reaches the next stage. In step 2, the data of 3rd strip will be mapped to RAM1 and the reading order for input motion estimation engine is changed to RAM2 RAM3 RAM1 from top to bottom. By using this method, the SAD of each pixel point is calculated in the search region.

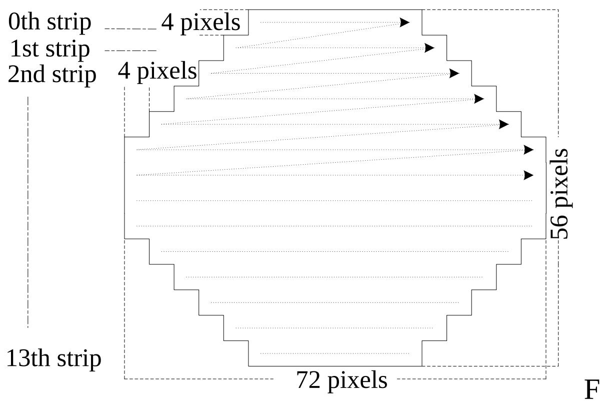

Fig.12 The search region

	Step 1	Step 2	Step 3		Step 11
RAM1	0th Strip	3rd Strip			13th Strip
RAM2	1st Strip		4th Strip		
RAM3	2nd Strip				
Order	1/2/3	2/3/1	3/1/2		2/3/1

Fig.13 Nine steps of fetching SRD memory

The Table.1 shown the scheduling of motion estimation engine, all of search points are completed in 812 cycles.

Table.1 Clock Distribution

CLK	SYSTEM_CONTROL	Position of MB
0~15	LOAD DATA	
16~47	Process 1st Strip	MB(15,0)~MB(31,3)
48~55	LOAD DATA	
56~95	Process 2nd Strip	MB(11,4)~MB(51,7)
96~103	LOAD DATA	
104~151	Process 3rd Strip	MB(7,8)~MB(55,11)
152~159	LOAD DATA	
160~215	Process 4th Strip	MB(3,12)~MB(59,15)
216~223	LOAD DATA	
224~287	Process 5th Strip	MB(0,16)~MB(63,19)
289~296	LOAD DATA	
297~360	Process 6nd Strip	MB(0,20)~MB(63,23)
361~368	LOAD DATA	
369~432	Process 7rd Strip	MB(0,24)~MB(63,27)
433~440	LOAD DATA	
441~504	Process 8th Strip	MB(0,28)~MB(63,31)
505~512	LOAD DATA	
513~568	Process 9th Strip	MB(3,32)~MB(59,35)
569~576	LOAD DATA	
577~624	Process 10th Strip	MB(7,36)~MB(55,39)
625~732	LOAD DATA	
733~772	Process 11th Strip	MB(11,40)~MB(51,43)
773~780	LOAD DATA	
781~812	Process 12th Strip	MB(15,44)~MB(47,47)

3.7 Proposed MV cost

The MV cost represents the required bits for the difference of PMV and current motion vector. The architecture of MV cost adopted by predecessors is look-up table, for example in paper [5]. However this method will use many ROM for look-up table, especially when the search region is larger. The whole architecture of MV cost is show in Fig. 14. We use the multiplexer to generate the data bits of absolute PMV. Then the results are enlarged by two times. Finally we obtain the MV cost by adding 1 to the results of second step. The experimental results show that the circuit is more rational and valid.

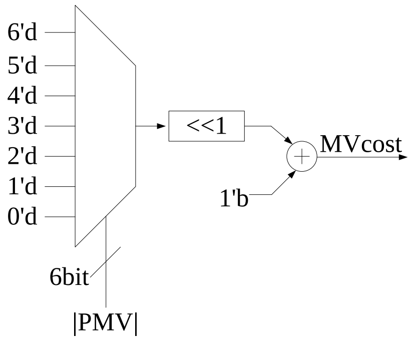

Fig.14 The architecture of MV cost

3.8 Full pixel search

In the first stage, the best matching point of sub-pixel is obtained by the blocks of PEarrays 1-4, Add_acc, Motion cost and RDO Unit. For example we obtain the best matching point in located the circle marked as 4(start) in fig.15. Then the search region data is mapped into the RAM. Then, we compute the SAD of full pixel search block by order 0,1,2,5,4,3,6,7,8, to increase efficiency, for the search data can be reused by adjacent PE arrays. Additionly, the MV cost of each point still uses the same results from the first stage motion cost block, for the MV cost keeps almost unchanged in the small size range.

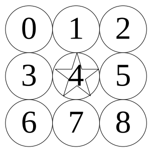

Fig. 15 The map of full pixel search

The architecture of full-pixels search shown in fig.16 can compute SAD of one bock of 4x4 in one clock. Based on the demand of HDTV encode, the same architecture numbers of 2 are added in the stage 2. When the stage one ends, the Ref_Data RAM loads the corresponding data of reference pixle according to the best match coordinate.

According to the H.264 implementation principle, we firstly need to obtain the best partition mode in 8x8,16X8,8X16 and 16x16. If the partition is 8x8 mode, the other sub block partition SAD will be computed to select the best partition. The order of partition blcok process is 8x8, 16x8, 8x16 for reduce the steps of process.

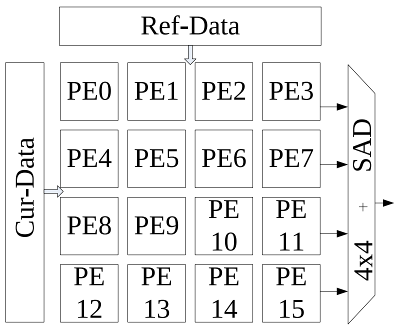

Fig.16 The architecture of full pixel search

4 Results and Discussion

The algorithm presented in this paper is implemented with C program. It is compared with the fast search algorithm of JM16.2 in the same condition. The setting of initial conditions: (1)group of picture structure is IPPP; (2)Hadamard transform used; (3)MV search range 128× 96; (4)number of reference frames equal to 1. (5)RDO ON; (6)quarter pixel is used for MV resolution; (7)fast search integer pixel ME; (8) context-adaptive variable-length coding is enabled.

4.1 Results of the sub-pixels search algorithm

A series of video sequences were used in our simulations. These sequences cover a wide range of motion content with two different formats including QCIF and HDTV. In addition, each video test in three QP includes 28, 32 and 36.

The simulation result of the sub-pixel algorithm shown in Table 3 depicts the performance with different sequences compared to JM16.2. The (+) sign in BD-BR and (−) sign in BDPSNR indicate the coding loss. The performance of PSNR and bit-rate in Table 2 does not significantly degrade which is a good trade off between performance and hardware cost. Additionally, the BD-BR is greater for CIF than HDTV video, for the larger the size of video, the less MB is divided into sub-block and the better effect of sub-pixel search algorithm is. The proposed algorithm can save the search points by around 70% in ME and yields an acceptable performance, which is deemed worthy for hardware implementation.

4.2 Results of the VLSI Architecture

The proposed architecture was synthesized based on the Synopsys Design Compiler with SMIC 0 .13μm CMOS technology and Artisan Memory compiler. The circuit can operate at frequencies reaching 200 MHz, subsequently allowing the VBSME processing with a search range of 128 × 96. The total clock cycles required to process an MB is 812 clock cycle.

Table 3 compares the proposed method with previous works in hardware design of motion estimation. The search pixels processed per gate in one second (pixels/(s*gate)) represent the hardware efficiency. Considering the performance and efficiency of the proposed sub-pixels search algorithm, the two level architecture was used in the ME design. Each level used independent 1-D array architecture.

The proposed architecture has SAD and MV cost electric circuit, whereas papers [25], [23], [26] and [27] have only SAD circuit. The hardware efficiency of proposed architecture was improved more than 4 times comparing with other method. The proposed architecture processes an MB within 812 cycles. Therefore, the resulting values explain why the proposed method performs better than that in [26] and [27]. The proposed sub-pixel search ME intellectual property covered with full search range of 128 × 96 can process 1920 × 1080 video sequence at 200 MHz.

Table.2. Simulation results for different video size

Sequences		QP	Full search		Subpixel search		BD-BR	BD-PSNR
			kbits/s	PSNR	kbits/s	PSNR		
CIF	Bus	28	1657.92	35.09	1733.9	35.02	+4.58%	-0.07
		32	990.77	32.08	1032.43	32.01	+4.20%	-0.07
		36	589.30	29.42	616.85	29.33	+4.67%	-0.09
	foreman	28	665.57	37.34	679.2	37.27	+2.05%	-0.07
		32	409.2	35.1	418.03	35.05	+2.15%	-0.05
		36	260.2	32.84	262.9	32.81	+1.04%	-0.03
	mobile	28	2735.81	34.42	2811.89	34.32	+2.78%	-0.1
		32	1571.57	30.97	1603.92	30.87	+2.06%	-0.1
		36	870.93	28	875.52	27.89	+0.52%	-0.89
HDTV	factory	28	8505.65	38.3	8577.89	38.28	+0.85%	-0.02
		32	4936.37	35.88	4984.80	35.85	+0.98%	-0.03
		36	2914.99	33.84	2945.62	33.81	+1.05%	-0.03
	controlled_burn	28	57317.04	38.64	57318.19	38.64	+0.002%	0
		32	40121.28	36.04	40125.6	36.03	+0.01%	-0.01
		36	27213.78	33.84	27215.28	33.69	+0.005%	-0.05
	touchdown_pass	28	31609.78	38.77	31620.53	38.77	+0.03%	0
		32	19431.46	36.97	19436.11	36.97	+0.02%	0
		36	11280. 14	35.43	11287.20	35.43	+0.06%	0
	ducks_take_off	28	38879.23	34.69	39333.89	34.65	+1.16%	-0.04
		32	20508.00	32.58	20750.02	32.54	+0.82%	-0.03
		36	11260.61	30.53	11366.69	30.47	+0.93%	-0.06

Table.3 Comparison of the Proposed Architecture and Previous Works

Paper	[25]	[23]	[5]	[26]	[27]	Proposed
Function	SAD		SAD+Mvcost	SAD		SAD+Mvcost
Architecture	2-D			1-D		
Search range	33x33	128x64	48x32	16x16	16x16	128x96
Work frequency（MHz）	200	108	200	200	260	200
Process （ um ）	0.18	0.18	0.13	0.18	0.18	0.13
Area： Gate - count	160k	330k	330k	597k	210k	530k
Cycles/MB	1129	8270	1614	256	5216	812
Search pixels/s/gate	1206	324	997	335	1003	4730

V. Conclusion

This paper presentes an efficient motion estimation algorithm based on the fast search algorithm for HDTV variable block size ME of H.264/AVC. The proposed algorithm can effectively complete the motion estimation with a peak signal-to-noise ratio drop of less than 0.1 dB than that of the fast search algorithm, and a maximum video codec bit rates drop of less than 1.16% than that of fast search algorithm of HDTV. In addition, the accuracy of the proposed algorithm makes it highly promising for hardware design implementation. The hardware architecture is implemented with the 1-D systolic array and its related circuits are successfully simulated for the H.264/AVC. The chip can operate at 200 MHz with a gate count of 530 k, including the memory modules. The proposed method is efficient compared with those of existing methods with respect to chip area, operation frequency, and throughput rate with negative effects on compression. The proposed sub-pixel search architecture decreases the numbers of search pixels of full pixels motion estimation by around 70% and the chip area by around 40% than the previous full pixels search algorithm.

References:

[1] T. Wiegand, G. J. Sullivan, G. Bjontegard, Over view of the H.264/ AVC video coding standard, *IEEE Transactions on Circuits and Systems for video technology*, Vol.13, No.7, 2003, pp. 560-576.

[2] Advanced Video Coding for Generic Audiovisual Services, *H.264 Standard*, International Telecommunication Union, 2005.

[3] K. M. Yang, M. T. Sun, and L. Wu, A family of VLSI designs for the motion compensation block-matching algorithm, *IEEE Transactions on Circuits and Systems*, Vol.36, No.10, 1989, pp. 1317-1325.

[4] J. C. Tuan, T. S. Chang, and C. W. Jen, On the data reuse and memory bandwidth analysis for full-search block-matching VLSI architecture, *IEEE Transactions on Circuits and Systems for video technology*, Vol.12, No.1, 2003, pp. 61-72.

[5] An-Chao Tsai, K. Bharanitharan, Jhing-Fa Wang, Effective Search Point Reduction Algorithm and Its VLSI Design for HDTV H.264/AVC Variable Block Size Motion Estimation, *IEEE Transactions on Circuits and Systems for video technology*, Vol.22, No.7, 2012, pp. 981-988.

[6] Mohammed Sayed and Wael Badawy, A Fully Parallel-Pipelined Architecture for Full-Search Block-Based Motion Estimation, *IEEE Transactions on Circuits and Systems*, Vol.30, No.11, 2002, pp. 24-27.

[7] Roger Porto, Luciano Agostini, Hardware Design of the H.264/AVC Variable Block Size Motion Estimation for Real-Time 1080HD Video Encoding, *IEEE Computer Society Annual Symposium on VLSI*, 2009, pp. 115-120.

[8] Weifeng He, Weiwei Chen, Zhigang Mao, AN EFFICIENT VLSI ARCHITEC TURE FOR EXTENDED VARIABLE BLOCK SIZES MOTION ESTIMATION, *IEEE Transactions on Circuits and Systems*, Vol.13, No.7, 2010, pp. 560-576.

[9]Shih-Chang Hsia, VLSI Implementation for Low-Complexity Full-Search Motion Estimation, *IEEE Transactions on Circuits and Systems for video technology*, Vol. 12, No. 7, 2002, pp. 613-619.

[10] Obianuju Ndili and Tokunbo Ogunfunmi, Algorithm and Architecture Co-Design of Hardware-Oriented, Modified Diamond Search for Fast Motion Estimation in H.264/AVC, *IEEE Transactions on Circuits and Systems for video technology*, Vol. 21, No.9, 2011, pp. 1214-1227.

[11] Yiqing HUANG, Qin LIU and Takeshi IKENAGA, VLSI Oriented Fast Motion Estimation Algorithm Based on Macroblock and Motion Feature Analysis, *IEEE Signal Processing*, 2009. pp.166-171.

[12] Tarek Darwish, Amit Viyas, Wael Badawy, Magdy Bayoumi, A LOW POWER VLSI PROTOTYPE FOR LOW BIT RATE VIDEO APPLICATIONS, *IEEE Transactions*, 2000. pp.159-167.

[13] Donglai Xu and John Bentley, VLSI- BASED PARALLEL ARCHITECTURE FOR BLOCK MATCHING MOTION ESTIMATION IN LOW BIT-RATE VIDEO CODING, *IEEE Transactions*, 2001.

[14] A.V. PARAMKUSAM and V.S.K. REDDY, An Optimal Fast Full Search Motion Estimation Algorithm In Video Coding, *IEEE Transactions*, 2011. pp.598-603.

[15] Xuan Jing and Lap-Pui Chau, Partial Distortion Search Algorithm Using Predictive Search Area for Fast Full-Search Motion Estimation, *IEEE Signal Processing Leters*, Vol.14, No.11, 2007, pp. 840-843.

[16] Zheng Zhaoqing, Sang Hongshi, Huang Weifengand and Shen Xubang, High Data Reuse VLSI Architecture for H.264 MotionEstimation, *IEEE*, 2006. pp.188-191

[17] L. Funucci, R. Saletti, L. Bertini and P. Moio, S. Suponara, High-Throughput, Low Complexity,

Parametrizable VLSI Architecture for Full Search Block Matching Algorithm for Advanced Multimedia Applications, *IEEE*, 1999.

[18] Cao Wei, Mao Zhi Gang, A Novel VLSI Architecture for VBSME in MPEG-4 AVC/H.264, *IEEE*. Vol.2, No.7, 2005, pp.1794-1797.

[19] Chien-Min Ou, Chian-Feng Le and Wen-Jyi Hwang, An Efficient VLSI Architecture for H.264 Variable Block Size Motion Estimation, *IEEE*. Vol.51, No.29, 2005, pp.1291 - 1299

[20]ITU-T, SERIES H: Audiovisual and Multimedia systems: Infrastructure of audiovisual services - Coding of moving video, *ITU-T Rec*. 2003.

[21] Bin Qi, Duoli Zhang, Yukun Song, Gaoming Du, Yong Zheng, Design and Implementation of a New Pipelined H.264 Encoder, *IEEE*, Vol.1, No.10, 2011, pp.130-133.

[22] Joint Video Team, Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification, *Geneva: ITU-T Rec H264 ISO/IEC 14496-10 AVC*, 2003.

[23] C. Y. Chen, S. Y. Chine, Y. W. Hung, T. C. Chen, T. C. Wang, and L. G. Chen, Analysis and architecture design of variable block-size motion estimation for H.264/AVC, *IEEE Transactions on Circuits and Systems for video technology*, Vol.53, No.2, 2006, pp. 578–593.

[24] L. Deng, W. Gao, M. Z. Hu, and Z. Z. Ji, An efficient hardware implementation for motion estimation of AVC standard, *IEEE Transactions Consumer Electron*, Vol.51, No.4, 2005, pp. 1360–1366.

[25] C. Wei, H. Hui, T. Jiarong, L. Jinmei, and M. Hao, A high-performance reconfigurable VLSI architecture for VBSME in H.264, *IEEE Transactions Consumer Electron*, Vol. 54, No. 3, 2008, pp. 1338–1345.

[26] J. Kim and T. Park, A novel VLSI architecture for full-search variable block-size motion estimation, *IEEE Transactions Consumer Electron*, Vol.55, No.2, 2009, pp.728–733.

[27] C. M. Ou, C. F. Le, and W. J. Hwang, An efficient VLSI architecture for H.264 variable block size motion estimation, *IEEE Transactions Consumer Electron*, Vol.51, No.4, 2005, pp. 1291–1299.