A NOVEL HYBRID MOTION ESTIMATOR SUPPORTING DIAMOND SEARCH AND FAST FULL SEARCH

Wei-Min Chao, Chih-Wei Hsu, Yung-Chi Chang, and Liang-Gee Chen

DSP/IC Design Lab,
Department of Electrical Engineering, National Taiwan University
Email: {hydra, jeromn, watchman, lgchen}@video.ee.ntu.edu.tw

ABSTRACT

In this paper, we present a novel hardware architecture supporting diamond search and fast full search for block matching motion estimation. It can handle irregular data flow of these fast algorithms without pipeline bubbles, and reduce computation of duplicated search positions. The proposed architecture needs preprocessing with small amount of computational power while performing fast full search and is suitable for the platform-based video coding system. While the diamond search mode can be applied for real-time requirements, we can choose the fast full search mode, which adapts processing cycles to picture contents and preserves the same quality of FSBM, for applications of high picture quality or compression ratio. It needs only 9K gates and one additional memory of search range size and is more cost-effective than the conventional systolic array architecture.

1. INTRODUCTION

Motion estimation is the key technique of video coding and can reduce the temporal redundancies of sequences to make compression efficient. In all algorithms of motion estimation, full search block matching (FSBM) algorithm is well known and commonly used in the video coding system because of its good performance and regularity. But due to its high computational power, dedicated hardware is usually employed. To meet real-time requirements, the systolic array architecture is widely adopted for FSBM and it needs large number of processing elements for parallel processing.

Recently many fast algorithms are proposed to speed up FSBM and preserve the same result. It is separated into two categories and called fast full search (FFS). One is the successively elimination algorithm (SEA) [1][2] and the other one is partial distortion elimination (PDE) [3][4]. The SEA uses the triangle inequality to eliminate useless search positions before calculating the sum of absolute difference (SAD). The PDE stops calculating SAD if the accumulating value is larger than the current minimum SAD. These algorithms can reduces 80% computational power of original FSMA but it also introduces irregular data flow and the computational complexity is still too large. Diamond search (DS) [5] has been proven to have sub-optimized result and just have 1.56% computational power compared to FSBM. For MPEG-4 series or H.263 series DS is a good choice to satisfy real-time applications and retains acceptable image quality at affordable cost. Due to its irregular data flow, it is also not suitable implemented by the systolic array architecture.

We proposed a new hardware architecture supporting DS and FFS dynamically for applications with different purposes. For real-time applications we choose the diamond search mode. For applications with the requirement of high picture quality we choose the fast full search mode and it can achieve the same processing power with the systolic array architecture of 64PEs.

The organization of this paper is as follows. In section 2, we review the algorithms adopted for hardware implementation. In section 3, the computationally adaptive motion estimator is presented. In section 4 the experiment results and discussions are represented. Finally, section 5 concludes this paper.

2. HARDWARE CONSIDERATIONS OF DS AND FFS

2.1 Architecture for irregular search positions

The irregular search positions of DS and FFS make it difficult to map into the systolic array architecture and therefore we use the tree architecture [6] and the interleaved memory organization to support random access according to search positions. The first advantage of this scheme is that it won't generate some pipeline bubbles after skipping some search positions. The second is that the bit-width of the adder-tree matches the inputs and is not the worse case as the systolic array. Therefore, the architecture is suitable for these fast algorithms with irregular data-flow and more cost-effective than the systolic array.

2.2 Skip duplicated search positions of DS

While performing DS, each SAD of the search positions in the current diamond patterns should be calculated and find the minimum one as the center of the next diamond pattern. There are four duplicated search positions for horizontal or vertical moving cases and three duplicated search positions for diagonal moving cases as shown in Figure 1. The SAD of these duplicated positions calculated in the current diamond pattern can be avoided in the next diamond pattern. For hardware implementation it costs a lot to store all these duplicated information while diamond patterns moves. We proposed the ROM-based solution to efficiently skip the duplicated positions among

0-7803-7448-7/02/$17.00 ©2002 IEEE

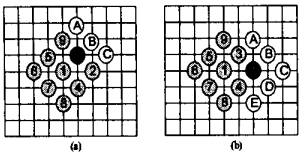

Figure 1. (a) Diagonally duplicated positions
(b) Horizontally duplicated positions

Table 1. Skipping methods for foreman sequence in the CIF format of 30fps at 512Kbps

Methods for duplicated positions	Candidates per frame	Skipping ratio	Hardware overhead
No skipping	8857	0	No
Skipping all duplicated positions	6693	24.43%	RAM of search range size
ROM-based	6711	24.23%	ROM (8 cases) 4-bits registers

successively moving patterns. For example in Figure 1(b) we suppose that the position 2 has the minimum SAD in the diamond pattern of the positions 1-9 and therefore not only the coordinate (u,v) of the position 2 but also the location (id) against the diamond pattern should be stored. Then in the next diamond pattern of nine locations (1-4 and A-E) this information is used to look up the table and choose valid ones (A-E). The table contains only eight cases due to eight locations of a diamond pattern except the center one and therefore is very small. Although it can't avoid all situations that the locations duplicate after two more moving of diamond patterns, it seldom occurs and is smaller than 0.3%. In Table 1 we use the ROM-based method and 24.23% of search positions is skipped with only a little of hardware overhead.

2.3 Computation reduction by the SEA and PDE methods

While performing FFS, the SEA and PDE methods avoid large useless computations but also cause irregular data flow and some hardware cost for calculating SEA criteria. The first problem can be solved by the tree architecture. For the second one the calculation of the SEA criteria is divided into hardware and software parts and it is suitable for platform-based video coding system.

Let C(x,y) is the pixel intensity of coordinate (x,y) in the current MB and R(u+x,v+y) is the pixel intensity of coordinate (u+x,v+y) in the search window where the coordinate (u,v) means the search position. From inequality of (1), it guarantees that SAD(u,v) is larger than the current minimum SAD if |SUM-SEA(u,v)| is larger than that. Therefore this criteria is used to decide whether to skip the search position (u,v) and avoid useless computations afterwards. SUM of (2) is the sum of pixels in the current MB and will be calculated once. SEA(u,v) of (3) is the sum of pixels in the candidate MB(u,v) and can be separated into horizontal calculation and then vertical calculation of (4) at the frame level. For sequences of CIF at 30 fps, the operations are only 24.3 MIPS and the data bandwidth is 12.16 Mbytes per second. Due to the features for calculating SEA criteria and affordable computation the processor is a good choice to do this task. Therefore, we suppose that it can be calculated by the processor platform first and transferred into the motion estimator. The penalty of this scheme is that we need an additional memory

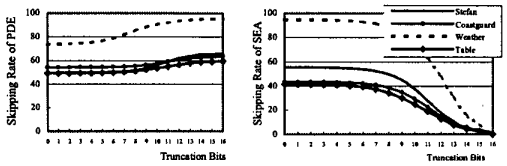

Figure 2. Skipping rate of SEA and PDE after truncating

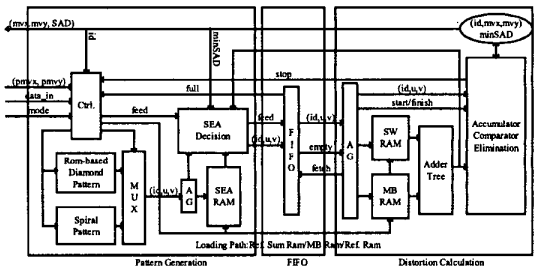

Figure 3. Architecture of hybrid motion estimator

$$SAD(u,v)=\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}\left|C(x,y)-R(x+u,y+v)\right|\geq\left|SUM-SEA(u,v)\right| \quad (1)$$

$$SUM=\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}C(x,y) \quad (2)$$

$$SEA(u,v)=\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}R(u+x,v+y) \quad (3)$$

$$SEAH(u,v)=SEAH(u-1,v)-R(u-1,v)+R(u-1+N,v),SEAH(0,v)=\sum_{i=0}^{N-1}R(i,v)$$
$$SEA(u,v)=SEA(u,v-1)-SEAH(u,v-1)+SEAH(u,v-1+N),SEA(u,0)=\sum_{i=0}^{N-1}SEAH(u,i) \quad (4)$$

$$SAD(u,v)\geq\left|(SUM-N0)-(SEA(u,v)-N1)-(N1-N0)\right|=\left|SUM'-SEA'(u,v)-(N1-N0)\right| \quad (5)$$
$$0\leq N0,N1\leq 2^{K}$$

$$SAD(u,v)\geq\left|SUM'-SEA'(u,v)\right|-2^{K} \quad (6)$$

of frame size in external RAM and an additional SEA memory of search range size in our architecture. To balance the performance of SEA and hardware cost, the least significant k bits of SEA(u,v) and SUM are truncated into SEA' and SUM' in (5). To guarantee the same result of FSBM, the absolute difference of SUM' and SEA' is subtracted by 2^K further in (6). From our simulation results in Figure 2, we choose the upper 8 bits of U and V for saving half size of SEA memory and it only drops 5% of the SEA skipping rate.

Besides, if the smaller minimum SAD of the candidate MBs can be found earlier, more useless computations can be avoided. A good initial guess is required to find the minimum SAD earlier and it not only improves the skipping ratio of SEA but also that of PDE. From the assumption that the motion vectors (MVs) of the neighbor MBs can usually be used to predict the MV of the current MB correctly and there is a high probability that MVs are closer to zero motion, the predictor is chosen as the first search position and others are generated in the spiral order from the origin of the search window [6].

3. HARDWARE ARCHITECTURE

3.1. Overview of the hybrid motion estimator

Figure 3 depicts the proposed hardware architecture of hybrid motion estimator supporting DS and FFS. This architecture mainly includes three processing stages and three buffers to store current MB, the search window and SEA criteria. Before performing motion estimation, the video coding system transfers data from external memory into these three buffers and the adder tree accumulates the sum of the pixels in the current MB and saves it in the SEA decision module meanwhile. To speed up the data loading, the search window buffer and the SEA buffer can be loaded using column-by-column data-reuse scheme [8]. After starting the motion estimation, the pattern generation (PG) stage generates the valid candidate positions. Then these positions are passed through the FIFO stage and fetched by the distortion calculation (DC) stage. The DC stage is responsible for calculating SAD of candidate positions and finds the minimum one. In the following sections we describe the detailed flows of these stages.

3.2 Pattern generation stage

While performing DS, this stage uses the ROM-based diamond pattern to generate valid search positions. After SAD of all search positions in the current diamond pattern are calculated, the controller feedbacks the id information of the minimum one to decide the center of the next diamond pattern and valid search positions.

While performing FFS, this stage generates search positions in the spiral scan order in Figure 4 and then uses SEA criteria pre-calculated by processor to filter useless positions off. In the following we describe how to generate the spiral pattern by hardware implementation. Suppose that the coordinate (u,v) is the start point of spiral pattern, G_k is the set of (k,k), N is the sequence of $\{G_1,G_2,G_3,G_4,...\}=\{1,1,2,2,3,3,4,4,...\}$, and n is the number following the element of the sequence N. Observing the regularity of the spiral pattern, it outputs n positions at the same direction and then turns to another direction. While k is the odd number, decrement u (v) if the current position is located in the first (second) part of G. While k is the even number, increment u (v) if the position is located in the first (second) part of G. Following this rule, we can generate the spiral pattern by using two counters, two comparators, two adders, and a simple state machine. Besides the good initial guess for finding the smaller SAD earlier, the other advantage is that the search range can be changed dynamically depending on the features of the sequences by controlling the counters.

3.3 Distortion calculation stage

This stage calculates the distortion between the current block and the candidate block of the specified position. Due to irregular candidate positions from the previous stage, the adder-tree structure and the interleaved memory organization in Figure 5 and 6 are used. From reasonable point of view, 64 bits width of eight banks memory are adopted and it can process eight pixels per cycle. To support block size of 8x8 and 16x16 and remain the same controlling scheme, the order of SAD calculation is shown in Figure 7. While computing partial distortion of eight pixels, the ACE module accumulates it for the SAD of the current position, compares it with the current minimum one, and eliminates this position if it is larger than that. If the process continues for the last cycle of SAD calculation, the current minimum SAD will be replaced with this one. By this PDE method, the processing cycles of the distortion calculation decrease further.

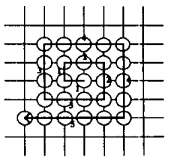

Figure 4. Spiral pattern of dynamic range

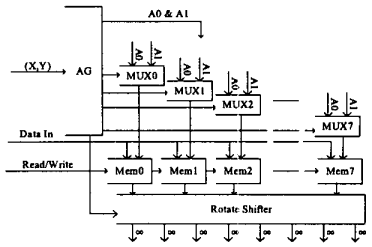

Figure 5. Interleaved memory organization of the search range RAM

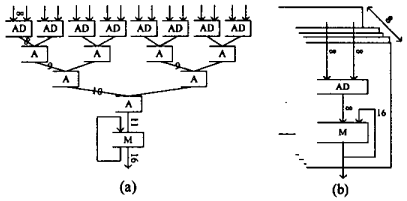

Figure 6. (a) Adder tree with suitable bit width
(b) Systolic Array with the worse case bit width

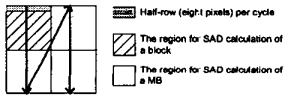

Figure 7. Scan order for SAD calculation

3.4 FIFO stage

The PG stage processes one position per cycle and decides whether it should be filtered off or not. The DC stage processes one position less than 32 cycles for a MB or 8 cycles for a block. At most time the PG stage waits for DC stage and the hardware utilization is poor. A FIFO is inserted between the PG stage and the DC stage to improve the hardware utilization. The PG stage will stop only when the FIFO is full. It will improve more if the FIFO has more elements, but the hardware cost will increase at the same time. From our simulation we adopt the FIFO of five elements and improve 5% performance.

4. EXPERIMENT RESULTS AND DISCUSSION

In Table 2 we simulate three modes of our motion estimator on five different sequences. The sequences are in the CIF (352x288)

format and each of them contains 300 frames. The search range is -16 to 15 along both axes. The first mode performs FFS and guarantees the same results with FSBM. The second mode also performs the fast full search but it stops before or at 4208 cycles for each MB to meet the real-time requirements. By using this scheme of 'cycle-cut' at 4208, it guarantees to complete in 4208 cycles (2879 cycles on the average) and PSNR only drops 0.0106 dB on the average and drops 0.981 dB in the worst case. The third mode performs the DS and it needs only one-tenth cycles of the fast full search mode and the PSNR drops 0.7194 dB on the average.

The SEA criteria provide the measurement of how much computational power to perform SAD calculation. In Table 2, a lot of positions will be excluded by SEA criteria in the case of small motion or detailed texture of sequences, and therefore it only consumes lesser cycles for valid search positions. If the motion of video sequences is fast or texture of pictures is similar, the skipping rate of SEA will decrease and it will consume more cycles to perform fast full search. In the latter case, all SAD in the search range will be almost the same and therefore it has better compression ratio to choose the MV with smaller SAD closer to the zero motion rather than the MV with minimum one. So we use the simple cycle-cut method and guarantee to complete the motion estimation at or before the specified time.

Our motion estimator targets for sequences in the CIF format at 30 fps and the search range is −16~15. While working at 50MHz it costs 9K gates and 28Kbits memory. For real-time applications, motion estimation should complete in 4208 cycles for a MB on the average and in 1666K cycles for a frame at most. Compared to the 1-D systolic array architecture with 64 PEs, which costs 60K gates to provide the same computational power, only one-sixth gates and more 8K bits memory are required in the proposed design. Preprocessing for the SEA criteria can be achieved by the processor when the proposed motion estimator is integrated with the platform-based video coding system.

5. CONCLUSION

In this paper a new hybrid motion estimator supporting diamond search and fast full search is proposed. In the DS mode, it only takes 173.2K cycles for a P-frame of CIF format and can meet real-time requirements while working at 6 MHz. For the FFS mode, it dynamically adjusts the processing cycles to the picture contents and takes almost the same amount of cycles with 64PEs 1-D systolic array architecture. With the cycle-cut method, the motion estimation is completed within 4208 cycles while the PSNR drops only 0.0106 dB. The SEA criteria can be calculated recursively by the processor platform and it takes about 24.3 MIPS for CIF 30 fps. The future work is integration with a complete MPEG-4 video encoding system.

6. REFERENCES

[1] W. Li and E. Salari, "Successive elimination algorithm for motion estimation," *IEEE Trans. on Image Processing*, vol. 4, No.1, pp. 105-107. Jan. 1995.

[2] X.Q. Gao, C.J. Duanmu, and C.R. Zou, "A multilevel successive elimination algorithm for block matching motion estimation," *IEEE Trans. Image Processing*, vol. 9, No.3, pp.501-504. March 2000.

[3] S. Eckart and C. Fogg, "ISO/IEC MPEG-2 software video codec." *Proc. SPIE*, vol. 2419, pp. 100-118, 1995.

[4] J.N. Kim, and et al., "A fast motion estimation for software based real-time video coding," *IEEE Trans. Consumer Electronics*, vol. 45, No. 2, pp. 417-426, May 1999.

[5] S. Zhu and K.K. Ma, "A new diamond search algorithm for fast block-matching motion estimation," *IEEE Trans. Image Processing*, vol. 9, No. 2, pp. 287-290, Feb. 2000.

[6] Jens-R. Ohm,"Digitale Bildcodierung: Repraesentation, Kompression und Ubertragung von Bildsignalen," Springer Verlag, pp.303-316. 1995.

[7] Y.S. Jehng, L.G. Chen, and T.D. Chiueh,"An efficient and simple VLSI tree architecture for motion estimation algorithms," *IEEE Trans. Signal Processing*, vol. 41, No. 2, pp.889-900, Feb. 1993.

[8] M.Y. Hsu, H.C. Chang, and L.G. Chen, "Scalable module-based architecture for MPEG-4 BMA motion estimation," *IEEE ISCAS*, vol. 2, pp.245-248, 2001.

Table 2. Various modes for different sequences of CIF 30fps at 256Kbps with search range -16~15

Sequence		FFS mode		FFS mode with cycle-cut at 4208			DS mode		
		PSNR-Y	Cycles (k) per frame	PSNR-Y	PSNR drop	Cycles (k) per frame	PSNR-Y	PSNR drop	Cycles (k) per frame
Weather	Max	36.000	1753	36.017	0.316	1158	35.784	0.314	195
	Min	26.361	574	26.361	-0.585	563	26.361	-0.551	109
	Average	33.545	777	33.555	-0.010	703	33.231	0.314	126
Foreman	Max	34.054	3362	34.083	0.981	1529	33.849	2.480	225
	Min	29.717	1299	29.299	-0.377	940	29.434	-0.301	180
	Average	32.529	1813	32.516	0.013	1159	31.797	0.732	194
Hall	Max	36.359	1908	36.431	0.628	1139	36.236	0.906	204
	Min	32.167	1337	32.167	-0.528	913	32.167	-0.306	151
	Average	35.690	1415	35.750	-0.060	940	35.490	0.200	161
Table Tennis	Max	34.832	6320	34.855	0.565	1579	34.275	6.665	299
	Min	27.399	2139	27.526	-0.292	1231	26.805	0.000	156
	Average	31.950	3743	31.935	0.016	1405	29.803	2.147	198
Coastguard	Max	30.728	5144	30.699	0.378	1583	30.657	0.923	300
	Min	26.294	2990	26.299	-0.325	1416	26.263	-0.196	152
	Average	28.554	3909	28.527	0.027	1494	28.304	0.249	187