SIFME: A SINGLE ITERATION FRACTIONAL-PEL MOTION ESTIMATION ALGORITHM AND ARCHITECTURE FOR HDTV SIZED H.264 VIDEO CODING

Tzu-Yun Kuo, Yu-Kun Lin, and Tian-Sheuan Chang
Institute of Electronics
National Chiao-Tung University
HsinChu, Taiwan
{tykuo,yklin,tschang}@twins.ee.nctu.edu.tw

ABSTRACT

This paper presents a set of fast algorithm and VLSI architecture for HDTV-sized H.264 fractional motion estimation. To solve the long computational latency in HD-sized application, we propose to use the single iteration algorithm with only six search points. This single iteration method halves the cycle count of two iteration methods in previous approaches. Moreover, we propose to use 4x4 Hadamard instead of 8x8 Hadamard as cost function for H.264 high profiles without significant video quality loss. By these techniques, the resulted architecture can save 20% of area and provide over 40% of throughput improvement than the previous work, and is able to support HDTV applications.

Index Terms— H.264**,** motion estimation, high profile.

1. INTRODUCTION

The H.264/AVC video compression standard[1], jointly developed by ITU-T and ISO/IEC, provides better compression and is widely adopted in various video applications. In which, motion estimation (ME) contributes a lot in compression efficiency and also on the computation time. Thus, many fast algorithms and hardware architectures are proposed for integer pixel motion estimation (IME) to meet real-time requirement. With the computation reduction of IME, the fractional pixel motion estimation (FME) now occupies 45% of the run-time in inter prediction and thus needs speedup as well.

Many fast FME algorithms are proposed to speed up the process such as the center based fractional pixel search (CBFPS)[3], the quadratic prediction based fractional ME algorithm[4], and the five candidates algorithm[5]. However, some algorithms [3][4] are software-oriented and exhibit irregular data flow and thus are not suitable for hardware design. Our previous work [5] is more suitable for hardware and can reduce the processing unit from nine to five to save hardware cost. However, from the hardware viewpoint it still suffers from long computation cycles as others. That is because it still takes two iterative search loops, one on half-pels and one on quarter-pels [6]. Thus, fast algorithms only reduce the processing element but do not reduce the cycle count in the hardware implementation. This problem will pose a strict limit on the HDTV sized applications since FME will take a lot of cycles and dominate the whole pipelining cycle time. Besides, all of these algorithms and designs do not consider the costly 8x8 SATD (sum of absolute transformed difference) computations in the high profile of H.264.

To solve above problems, this paper presents a single iteration fast FME algorithm and its architecture suitable for HDTV and high profile applications. The proposed algorithm can complete the quarter-pixel precision motion search by only examining six search points in one search step instead of 17 search points in two search steps in the reference software[2]. Thus, we can reduce the number of SATD units since we only search 6 candidates. Besides, the cycle count is also halved by using only one search step. Furthermore, to avoid the costly 8x8 SATD computations with 8x8 Hadamard transform, we use the 4x4 Hadamard transform units. Thus, we can achieve smaller area and fewer cycle counts at the same time.

The rest of the paper is organized as follows. In Section 2, we introduce previous fractional-pel motion estimation algorithms and their drawbacks. In Section 3, we propose our single iteration fractional-pel motion estimation algorithm (SIFME) to overcome their drawbacks. In Section 4, the corresponding hardware architecture for this algorithm is demonstrated, and the reduction of hardware cost and latency is introduced. Section 5 presents the results. Finally, we conclude the paper in Section 6.

2. OVERVIEW OF RELATED WORK

2.1 FME in the reference software

Fig. 1 shows the search method applied in the reference software[2] with two search steps. In the first search step, it calculates the cost for each half pixel around the central integer pixel position (the rectangular point in the figure) and chooses the candidate with minimum cost as the search center for further refinement. In the second step, further 8 candidates (the triangular points) around the previous search

center in quarter pixel position are examined. Thus, total 17 search points are needed for fractional ME. Although this algorithm is suitable for hardware[6], it has two drawbacks. First, the search points in each step are still too many and thus results in nine processing units (PUs) for hardware implementation. The second drawback is that it needs two iterative search loops of interpolation and Hadamard transform to calculate the SATD cost.

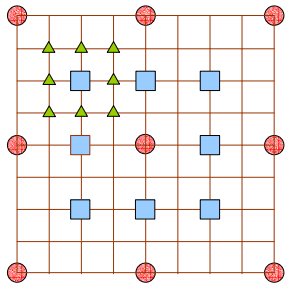

Fig. 1 FME algorithm in reference software

2.2 Center-biased FME

The center-biased FME[3] uses the information of predicted motion vector (*pred_mv*). It first calculates the fractional predicted motion vector(*frac_pred_mv*) :

$$frac\_pred\_mv = (pred\_mv - mv)\%\beta \quad (1)$$

where *pred_mv* here is defined as the fractional pixel unit. *mv* is the integer pixel motion vector after IME process, and *mv* is also in fractional pel unit. % is the mode operation, β=4 in 1/4-pel case and β=8 in 1/8-pel case. *frac_pred_mv* is the predicted fractional motion vector and indicates only fractional position. Then, it compares the cost at (0, 0) and *frac_pred_mv* and does the first diamond search around the lower cost one. After that, it refines around the best point until it is center-located. The concept behind [3] is that the probability of finding the motion vector around *frac_pred_mv* is higher than that around (0,0). However, this algorithm still needs at least two iterative loops and thus is not suitable for low latency hardware design.

3. SIFME: A SINGLE INTERATION FRACTIONAL MOTION ESTIMATION ALGORITHM

3.1 Proposed SIFME Algorithm

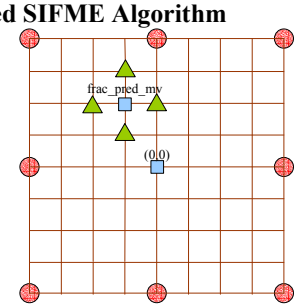

Fig. 2 The proposed SIFME algorithm flow on two square points, (0,0) and ***frac_pred_mv***, and four triangle point around ***frac_pred_mv*** in one quarter-pel distance.

Inspired by the center-biased FME, we modify it by searching six candidates in only one loop and no refined search as shown in Fig. 2. The six candidates includes (0, 0), *frac_pred_mv* and four diamond points around *frac_pred_mv*. (0, 0) is included for low texture and low motion sequences. More search points are placed around *frac_pred_mv* since the best fractional motion vector is more often around *frac_pred_mv* than around (0, 0).

3.2 Analysis of prediction accuracy and search point

Table 1 shows the prediction correctness compared with the algorithm in the reference software under different quantization parameter (QP). The prediction accuracy is defined as if the search fractional MV by the fast algorithm is the same as that in the full search algorithm of the reference software. This result shows that it can still have about 60~90 % of prediction accuracy though the proposed algorithm had ignored more than 88% search points.

Table 1 hit rate of motion vector (mvx and mvy) compared to the full search FME algorithm

CIF size, 300 frame, IPPP, ProfileIDC=100, RDO off				
QP	container	foreman	mobile	stefan
10	82.31%	61.80%	74.10%	62.20%
16	85.11%	68.60%	76.30%	70.70%
22	82.18%	70.97%	76.70%	75.70%
28	90.21%	78.90%	79.00%	79.40%
34	94.41%	86.40%	82.30%	82.83%
40	94.71%	91.10%	86.10%	85.40%

Table 2 shows the search point comparisons with other algorithms. The proposed algorithm needs the fewest search points compared with other search algorithms, 64% reduction compared to reference software. Besides, our algorithm does not need the second step search and saves the additional interpolation time in the second step. Table 3 shows the comparisons for hardware implementation. The proposed algorithm searches only six candidates and needs only six PUs. Besides, with one loop design, our design just takes about only half of cycles compared to that with reference software[6] and fast algorithm in [5].

Table 2 search point comparisons for different algorithms

	search point
JM [2]	17
[4]	6+multiple diamond search (Total <=11)
[3]	6 + multiple diamond search
[5]	8~9
proposed	6

Table 3 comparisons of number of processing unit(PU) and number of iterative search steps

	# of PU	# of iterative search step
[5]	5	2
[6]	9	2
proposed	6	1

3.3 SATD cost with 4x4 Hadamard for high profile

In high profile of H.264, residual of block size larger than 8x8 are passed through 8x8 integer transform rather than 4x4 one. Thus, in the reference software, it adopts 8x8 Hadamard transform for SATD calculation for block size larger than 8x8 [2]. Though Hadamard transform is greatly simplified, one 8x8 Hadamard transform unit still consumes about four times area than that of 4x4 one. For six PUs in our design, six 8x8 transform will be required and thus cost a lot of area cost.

To solve this area problem, we propose to use 4x4 Hadamard for all SATD calculation disregarding of the block size. Table 4 shows the simulation results of our algorithm with different SATD strategy. All the data are compared with the reference software. As shown in the table, the results with 4x4 and 8x8 Hadamard transform are similar except for low motion sequences like container at high QP situations. That is quite acceptable since the bit rate at that condition is quite low and any increase will be large in terms of that bit rate. As the 4x4 transform unit only consumes 25% area cost of 8x8 one, we choose to calculate SATD by 4x4 Hadamard transform that has similar performance and saves about 75% of area cost in PU and 60% of area cost in the total FME module.

Table 4 simulation results of SIFME with different SATD methods when compared to the reference software

CIF size, 300 frame, IPPP, ProfileIDC=100, RDO off						
SIFME with 4x4 Hadamard transform						
	Container		foreman		Stefan	
QP	ΔPSN R(dB)	Δbit rate	ΔPSN R(dB)	Δbit rate	ΔPSN R(dB)	Δbit rate
10	-0.03	-0.75%	-0.05	0.04%	-0.04	0%
16	0	-0.28%	-0.07	1.03%	-0.05	0.30%
22	-0.03	-0.37%	-0.09	0.89%	-0.06	0.50%
28	0.03	0.46%	-0.09	1.50%	-0.07	1.24%
34	0.04	2.11%	-0.12	1.35%	-0.10	1.57%
40	-0.03	4.36%	-0.08	-0.36%	-0.13	1.02%
SIFME with 8x8 Hadamard transform						
	container		foreman		stefan	
QP	ΔPSN R(dB)	Δbit rate	ΔPSN R(dB)	Δbit rate	ΔPSN R(dB)	Δbit rate
10	-0.02	-0.19%	-0.05	0.41%	-0.03	0.31%
16	-0.03	-0.35%	-0.06	1.33%	-0.04	0.68%
22	-0.01	-0.25%	-0.07	1.42%	-0.06	1.06%
28	0.03	0.19%	-0.09	1.22%	-0.07	1.86%
34	-0.02	0.72%	-0.12	-0.54%	-0.08	1.97%
40	0.02	2.53%	-0.11	-2.77%	-0.11	-0.04%

4. HARDWARE ARCHITECTURE

Fig. 3 shows the proposed hardware architecture. The input data are first interpolated by the interpolation unit for half and quarter pixels of one 4x4 block. Then these data are computed with the current block data with the six 4x4 block PUs. Each PU is in charge of residual generation and 4x4 Hadamard transform. All larger sized block are decomposed into 4x4 block for processing. Then the residual cost combining with MV cost is sent to the Compare unit to find the best one and stored in SB_buffer.

The total cycle count is 1002 cycles for one MB processing. With the single step algorithm, the total cycle count is just 40% of that in [6].

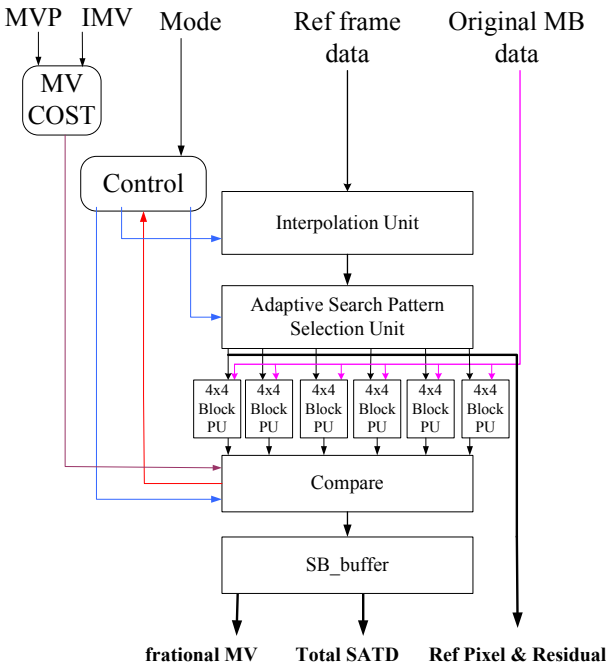

Fig. 3 The proposed hardware architecture

5. SIMULATION & SYNTHESIS RESULTS

Table 5 shows the simulation results of the proposed SIFME with 4x4 Hadamard transform algorithm compared with that of reference software for 720p sequences. As our hardware architecture is used for high profile and HDTV size video, we care more about the performance on 720p size sequences rather than that on CIF size sequences. Comparing the results of Table 4 and Table 5, we can find that our algorithm has better performance on large size sequences than CIF size sequences, which matches our goal. We can also find that our algorithm greatly reduces computation time of FME. The proposed algorithm can speedup the FME part by up to 4 times compared to the reference software. The reason is due to the reduction of search candidates, and 4x4 instead of 8x8 Hadamard transform.

Table 5 PSNR & bit rate comparison for different 720p sequences and QPs. Speed up is only the performance in fractional ME part

720p, 100 frames, IPPP, ProfileIDC=100, RDO off												
	mobcal			parkrun			shields			stockholm		
QP	ΔPSNR (dB)	Δbit rate	speed up	ΔPSNR (dB)	Δbit rate	speed up	ΔPSNR (dB)	Δbit rate	speed up	ΔPSNR (dB)	Δbit rate	speed up
10	-0.04	-0.77%	4.0	-0.02	-0.77%	3.9	-0.04	-0.42%	3.6	-0.04	0.05%	3.8
16	-0.04	-1.07%	3.6	-0.04	-0.99%	3.7	-0.08	-1.27%	3.7	-0.08	-0.86%	3.6
22	-0.01	-1.08%	4.0	-0.05	-1.42%	3.9	-0.04	-1.54%	3.9	-0.05	-1.50%	3.7
28	-0.01	-0.36%	3.9	-0.04	-0.63%	3.9	-0.02	-0.36%	3.6	-0.02	-0.71%	3.8
34	-0.05	3.20%	3.9	-0.05	-0.14%	3.8	-0.03	0.30%	3.6	-0.01	-1.87%	3.7
40	-0.06	4.28%	3.7	-0.04	-0.70%	4.1	-0.01	-7.05%	3.5	0	-8.86%	3.7

The proposed FME architecture is synthesized with 0.18um technology at 70MHz and the details of each block are listed in Table 6. It can be found that the six PUs occupy the largest area, and thus reducing the number of PUs can significantly reduce the area cost. Compared to [6] as shown in Table 7, we save 20% of area cost due to fewer number of PUs. Besides, our design can process 71.3k MB/sec in 70MHz clock rate, which is sufficient for HD sized applications. We can achieve more than 40% of throughput improvement with lower clock rate than that in [6] because we only take one search step. Most of all, our hardware architecture is designed for H.264/AVC high profile that is not supported in [6].

6. CONCLUSION

In this paper, we propose a simple single iteration fast FME and its architecture for HD-sized and high profile applications. With the presented techniques, we can save 20% of area cost and provides 40% higher of throughput compared with previous approaches. Therefore, with processing rate of 71.3K MBs/sec, the design is suitable for the HDTV applications or other cases which needs fast fractional motion estimation.

Table 6 synthesis result of the proposed architecture

	Gate Count
Control	303
MV_COST	1278
Interpolation unit	19049
Selection unit	10525
4x4 Block PU	25807
Compare unit	3016
SB_buffer	2089
Total	62260

Table 7 comparison between the proposed architecture and other hardware architecture

	[6]	proposed
Δbit rate	0	-0.24%
ΔPSNR(dB)	0	-0.0438
clock	100MHz	70MHz
Gate count	79372	62260
MB/sec	50k	71.3k
Cycle/MB	1648	1002

7. REFERENCES

[1] *Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification* (ITU-T Rec. H.264/ ISO/ IEC 14496-10 AVC), Mar. 2003.

[2] Joint Video Team Reference Software JM9.8.

[3] Libo Yang, Keman Yu, Jiang Li, and Shipeng Li, "Prediction-based Directional Fractional Pixel Motion Estimation for H.264 Video Coding", *in Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP2005)*, Vol. 2, pp.901 – pp. 904, 2005.

[4] Jing-Fu Chang, Jin-Jang Leou, "A Quadratic Prediction Based Fractional-Pixel Motion Estimation Algorithm for H.264," *in Proc. Seventh IEEE International Symposium on Multimedia (ISM'05)* pp. 491-498, 2005.

[5] Yu-Jen Wang, Chao-Chung Cheng, and Tian-Sheuan Chang, "A Fast Fractional Pel Motion Estimation Algorithm for H.264/AVC", *in Proc. International Conference on Circuit and System (ISCAS)* 2006

[6] Tung-Chien Chen, Yu-Wen Huang, and Liang-Gee Chen, "Fully utilized and reusable architecture for fractional motion estimation of H.264/AVC" *in Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP2005)*, vol. 4. pp.9-12, 2004.