A Fast Algorithm and Its VLSI Architecture for Fractional Motion Estimation for H.264/MPEG-4 AVC Video Coding

Yu-Jen Wang, Chao-Chung Cheng, and Tian-Sheuan Chang, *Member, IEEE*

***Abstract*—This paper presents a fast algorithm and its VLSI architecture for H.264 fractional motion estimation. Motivated by the high correlation of cost between neighboring fractional pel position, the proposed algorithm efficiently explores the neighborhood position around the minimum one and thus skips other unlikely ones. Thus, the proposed search pattern and early termination under constant quantization parameter can reduce about 50% of computation complexity compared to that in reference software but only with 0.1–0.2 dB peak signal-to-noise ratio degradation and less than 2% of bit rate increase. The VLSI architecture of the proposed algorithm thus can save 40% of area cost due to only half of the processing elements and save 14% of searching time when compared with the previous design.**

***Index Terms*—H.264/AVC, motion estimation, video coding.**

I. Introduction

IN RECENT years, multimedia application has become more flexible and powerful with the development of semiconductors, digital signal processing, and communication technology, in which the latest video standard [1], known as H.264 and also MPEG-4 Part 10 Advanced Video Coding, is regarded as the next generation video compression standard. Among its coding tools, motion estimation (ME) is the most important part in exploiting the temporal redundancy between successive frames and is also the most time consuming part in the coding framework. ME is conducted in two parts: the integer-pel ME and the fractional pel ME with quarter-pel precision around the best integer-pel ME position. ME occupies 60%–90% of computational time of the whole encoder from the simplest configuration to the complex configuration, respectively, in which the fractional pel ME occupies significant amount of the computation time of the whole ME process. Thus, the fast implementation is important for real time video applications.

Many fast integer ME algorithms such as the three-step search [2], and new diamond search [3] are proposed to reduce the complexity of integer ME, but few discuss the fractional ME [4]–[10]. However, the fractional ME has a strong impact on the peak signal-to-noise ratio (PSNR) quality, about 2-3 dB significant improvement, and also has measurable high computational complexity compared to the integer ME due to complex sub-pels interpolation process. Thus, this paper proposes a fast algorithm and its architecture to speed up the H.264/AVC fractional ME. The proposed algorithm only needs eight or nine search points instead of 49 search points in the full search method and 17 search points in the algorithm of reference software. The proposed algorithm is also suitable for hardware design that has smaller area cost and shorter refinement time compared with the previous design.

The rest of the paper is organized as follows. In Section II, we first review the past approaches. In Section III, we analyze the statistics of the final fractional pel position. Then, in Section IV, we propose the fast fractional ME algorithm. The VLSI architecture and implementation result are listed in Section VI and VII. Finally, conclusions are made in Section VIII.

Manuscript received November 16, 2005; revised June 20, 2006 and September 15, 2006. This workwas supported in part by the National Science Council, Taiwan, R.O.C., under Grant NSC-93-2200-E-009-028. This paper was recommended by Associate Editor O. C. Au.

Y.-J. Wang is with M-Star, Inc., Hsinchu 302, Taiwan, R.O.C.

C.-C. Cheng is with the Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 10617, Taiwan, R.O.C.

T.-S. Chang is with the Department of Electronics Engineering, National Chiao Tung University, Hsinchu 300, Taiwan, R.O.C. (e-mail: tschang@twins.ee.nctu.edu.tw).

Digital Object Identifier 10.1109/TCSVT.2007.894050

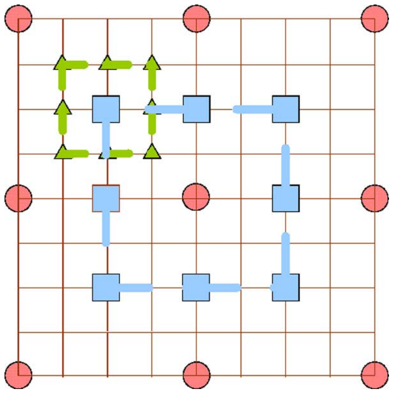


Fig. 1. Search algorithm in reference software.

II. Previous Approaches

Fig. 1 shows the search method applied in the reference software [9]. It first recalculates the cost of integer pixel position and half-pel positions (the rectangular points) and further refine the motion vector with the quarter-pel precision (the triangular points). Thus, total 17 search points are needed for fractional ME. Note that the recalculation of best integer-pel position is required due to different matching criteria in fractional ME [sum of absolute transformed difference (SATD) instead of sum of absolute difference (SAD)]. In [4], a gradient-based search algorithm is brought that uses the distribution features of ME error surface. In [7], the number of the search points can be reduced

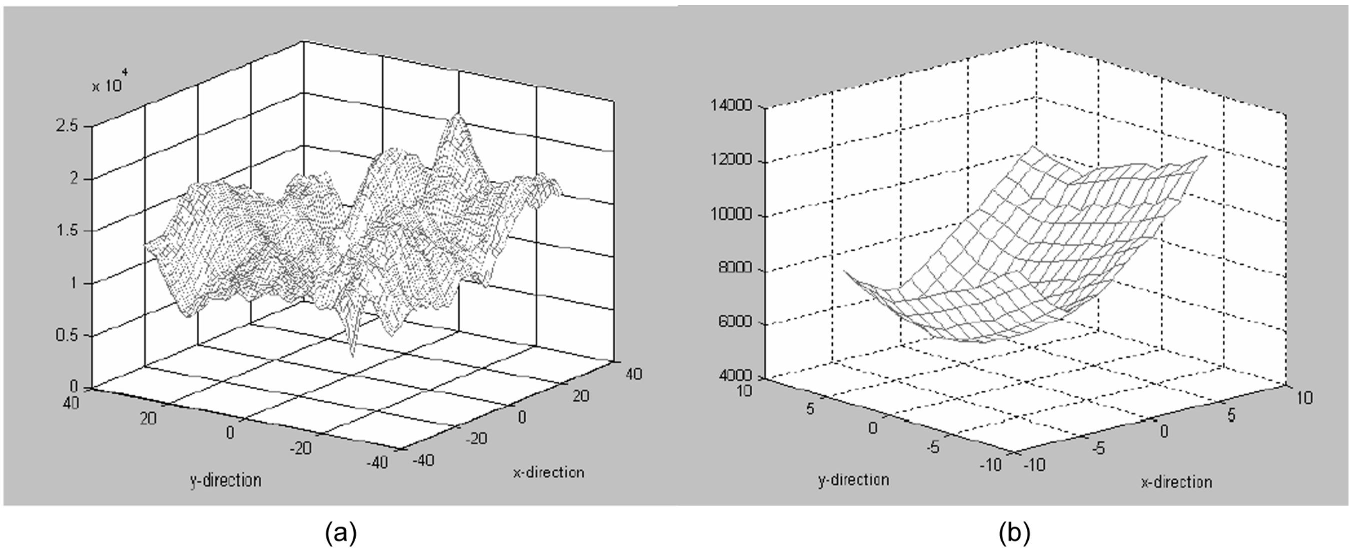


Fig. 2. Statistics from [10]. Error surface of (a) integer-pel ME (search range: 32) and (b) fractional pel ME (1/8-pel case).

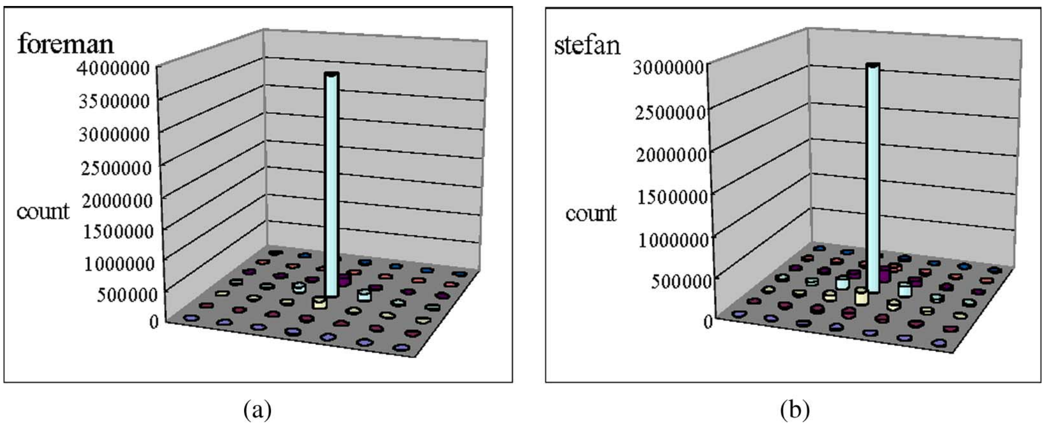


Fig. 3. Distribution of the fractional ME. (a) Foreman. (b) Stefan.

by predictor which is incorporating the motion vector information of the adjacent blocks. Our design also explores the features of the error surface but in a different way to predict the fluctuation. In [10], it uses a three-step approach by using the diamond search in the first and third step. To compensate the quality loss, they add a second step by searching another one to four half-pels. Thus, the total search points will be from 9 to 13 instead of 8 to 9 in our algorithm. However, from the hardware viewpoint, each step will take the similar cycles to complete regardless of the number of search points due to hardware parallelism. Thus, the three-step nature will result in a extra one-half of computation cycles than our two-step approach. If forcing the algorithm into a two-step design by combining the first two steps, the required hardware will be nine processing elements (PEs) as in [6], which needs an extra 40% of area cost.

III. Analysis of Fractional Pel Motion Vector

It is generally believed that the fast ME algorithm works best if the error surface inside the search window is unimodal. As shown in Fig. 2, the error surface of the integer-pel ME is not unimodal due to the large search window and complexity of video content. So the ME search would easily be trapped into a local minimum. On the other hand, since the sub-pels are generated from the interpolation of integer-pels, the correlation inside a fractional pel search window is much higher than that of the integer-pel search window. Thus, the unimodal error surface will be valid in most cases of the fractional pels. So the matching error decreases monotonically as the search point moves closer to the global minimum.

In the full search method, every fractional pel around the original integer-pel is treated equal. However, with the valid unimodal error surface assumption, a fast algorithm can work well if every candidate of the sub-pel refinement has different occurring probabilities. Fig. 3 shows the distribution of the fractional motion vector around the best integer motion vector. It is obvious that more than 90% of the fractional motion vector is at the search center in all kinds of video content. However, we still cannot just avoid the fractional part even though there is a huge density diagram near the bias search center. The small error drift of the fractional part in the motion vector will lead to a significant bit rate increase.

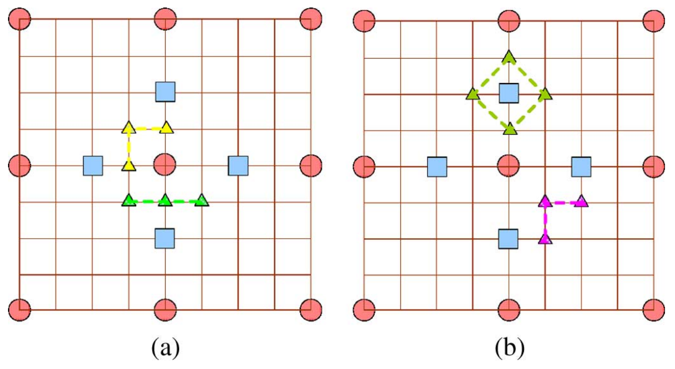


Fig. 4. Proposed search algorithm (different symbols mean different search patterns). (a) Case 1 and 2. (b) Case 3 and 4.

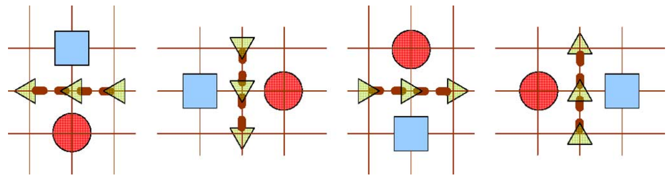


Fig. 5. Refined positions in case 1.

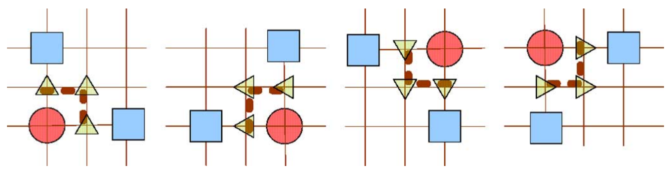


Fig. 6. Refined positions in case 2.

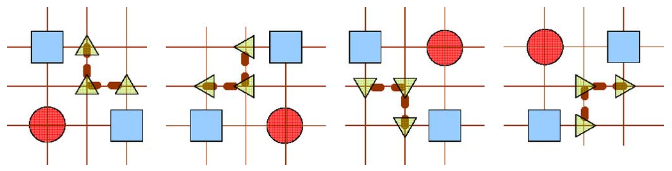


Fig. 7. Refined positions in case 3.

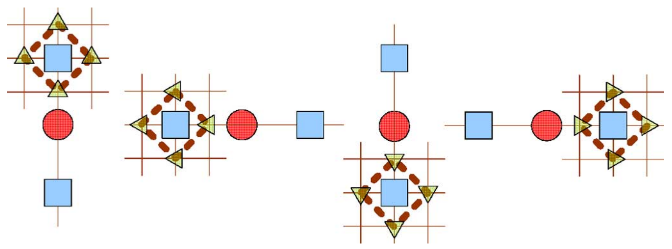


Fig. 8. Refined position in case 4.

IV. Proposed Fast Algorithm

A. Search Pattern

The proposed algorithm uses two-step search processes, as shown in Fig. 4.

Step 1) Calculate the cost of the best integer-pel position and four search points in half-pel positions (rectangular points).

Step 2) Depending on the best three search positions, the search pattern of the second step is adaptively selected as shown in Fig. 5–8. The algorithm will bias the search pattern to the search center if the minimum cost point is at the integer-pel, as shown in Fig. 4(a), in which case1 or case2 is chosen as the second step patterns according to whether the second and third best positions are neighboring or not. Otherwise, we will bias the search pattern away from the search center, as shown in Fig. 4(b), in which case3 or case4 is chosen as the second step patterns according to whether the first and second positions are neighboring or not. The details of each case are shown below.

Case 1) When the minimum cost point falls on the search center, and the second and third best search positions are not neighboring each other, we will choose the three search points between them, as shown in Fig. 5.

Case 2) When the minimum cost falls on the search center and the second best search position is neighboring the third one, we will choose the "L" shape pattern as shown in Fig. 6.

Case 3) When the best two search positions are at the four end points and neighboring each other, we will search the three candidates in the "L" shape between the best two as shown in Fig. 7 with search center bias.

Case 4) When the best two search positions are at the four end points and do not neighbor each other, we will search the four candidates around the best search point as shown in Fig. 8.

The concept of the algorithm is described below. From the previous analysis, we find that the fractional motion position will be close to the center of integer search point with very high probability. To match such statistics, we will bias our second step search near the center. Moreover, with the valid unimodal error surface assumption [10], we can just examine the neighborhood position around the points with low cost value and thus skip other unlikely positions. We use the fixed half-pel search pattern for half-pel and adapt quarter-pel search patterns. In every fractional-pel refinement, only triangle points in the same set will be visited.

With the above steps, our algorithm just needs 8 (for case 1$\sim$3) or 9 points (for case 4). Thus, compared with 17 points in the reference software, our algorithm significantly reduces the required search points by over 50%, especially for the second step refinement. In addition, the algorithm is also suitable to be implemented in hardware architecture due to the great decrease of the search point candidates, and thus reducing the hardware processing elements.

V. Early Termination

We also apply the early termination technique to every single search point in each step. The problem for early termination is how to define the threshold. The matching error considered as

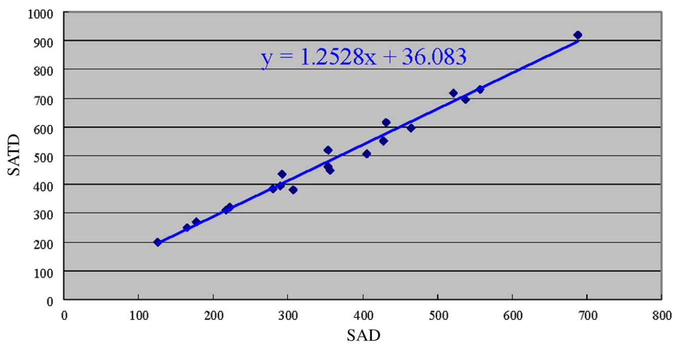


Fig. 9. Relationship between best SAD in integer part and best SATD in fractional part.

SATD is used in fractional ME and SAD in integer ME. SATD is the results after SAD goes through a 2-D Hadamard transform. The threshold value (SATD) used in fractional ME can be estimated from the integer-pels matching error (SAD). We experiment from several test sequences and get the formula listed in Fig. 9.

In most of situations, we can use the above approximated formula to predict the threshold. However, direct linear prediction may lead to a too large threshold and arise too much imprecision when the SAD is getting larger. To solve the problem and avoid second- or high-order approximation, we adopt adaptive linear prediction threshold. We have found that while the quantization parameter (QP) is getting larger, the average best SAD from integer ME is getting bigger. To achieve the shorter searching time without significant performance loss, we increase the threshold associating to the current QP. The final prediction formulas are

```
if $(\mathrm{SAD} > 1000)$
    $\{\mathrm{threshold} = \mathrm{SAD} * 0.75 + (\mathrm{QP} - 28) * 16 + 375 + 36;\}$
else if $(\mathrm{SAD} > 500)$
    $\{\mathrm{threshold} = \mathrm{SAD} * 1 + (\mathrm{QP} - 28) * 16 + 125 + 36;\}$
else
    $\{\mathrm{threshold} = \mathrm{SAD} * 1.25 + (\mathrm{QP} - 28) * 16 + 36;\}$
```

Every coefficient used in the formula could be calculated by add and shift, and the summation of constants could be combined easily. A constant with the value of 36 is obtained in the formula listed in Fig. 9. Constants with the value of 375 and 125 are used to maintain the continuity of the adaptive prediction curve.

By applying the early termination technique, we can improve searching speed about 8.5%–14%. While the QP is getting larger, we may get a bigger threshold and lead to shorter searching time.

VI. Hardware Design

A. Hardware Consideration

Algorithms suitable for hardware designs should be regular. Thus, it is more difficult to implement an irregular search pattern used in most of the software-based fast algorithms. For example, in the integer ME part, the memory access cannot support jumping access. In the fractional ME part, the search window is smaller so that access to any point in the search window is possible. But an irregular search pattern will certainly result in more overhead circuits. In our design, even though there are four cases, it is still fixed. We believe that the overhead circuits of our design will be smaller than other algorithms, such as the gradient-based algorithm [4] that has a higher variation probability in the search window.

Furthermore, 4×4 block decomposition and vertical integration are proposed in [6]. All block types can be decomposed by 4×4 block, and the SATD of each element is accumulated to get the final cost. For the data reusability, vertical integration is one of the ways to reduce the encoding time to reduce redundant interpolating operations in the overlapped area of an adjacent interpolation window. But the overhead, such as the more complex timing control circuit, will be introduced.

B. Algorithm Modification for Hardware

The algorithm implemented is slightly different from the fast algorithm mentioned before. The main difference is when to apply the early termination technique. Early termination at each search point is only reasonable in the sequential processing like those on CPU or DSP. However, in the case of hardware design, the available resources allow us to use parallel processing units for all the search points in the same step, and thus we only terminate the second step process if the requirement is met. With this modification, the count of early termination occurrences decreases from 56%–28% in average, but is still significant according to our simulation results.

The total encoding time of the above modification can be calculated as follows. First, the encoding time is the same in each refinement step since every search point is calculated in parallel. Let us assume the total time without early termination is $\boldsymbol{T}$, and $\boldsymbol{t}$ as the total time with step stop early termination. We can find the following relationship:

$$\boldsymbol{t} = \boldsymbol{T} * (\mathbf{1} - \mathbf{0.28}) + \mathbf{0.5}\boldsymbol{T} * \mathbf{0.28} = \mathbf{0.86}\boldsymbol{T}.$$

Thus, we can save 14% of search time in FME module.

C. Architecture

Fig. 10 shows the proposed architecture for fast FME module. The core procedure of FME includes interpolation, residual generation, and Hadamard transform.

The 4×4 block processing unit (PU) has four times parallelization of horizontal adjacent pixels and is in charge of residual generation and Hadamard transform. The architecture of PU is shown in Fig. 11 that contains four PEs, 2-D Hadamard transform decomposed by two 1-D Hadamard transforms, and a transpose register array. It processes 4×4 element blocks decomposed from subblocks in sequential order and continually processes four pixels in each cycle without any latency.

Five 4×4 block PUs around the refinement center process five candidates simultaneously. Four horizontal adjacent pixels from the original macroblock (MB) are broadcasted to every PU at the same time and the reference subpixels are provided by the interpolation unit. The interpolation unit by the 6-tap 2-D finite impulse response (FIR) filter is divided into two directional (horizontal and vertical) 1-D FIR filters. First, we interpolate the horizontal half pixels by five FIR filters from ten adjacent integer pixels. These five intermediate values and six integer

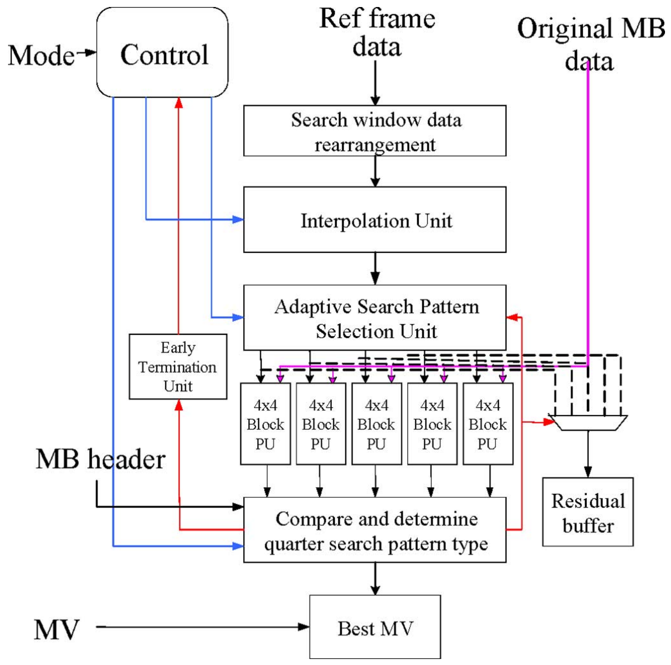


Fig. 10. Block diagram of fast FME hardware.

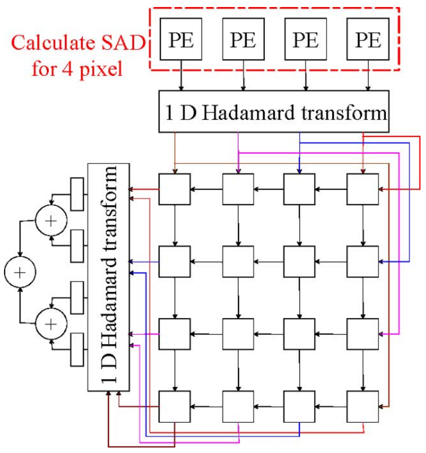


Fig. 11. 4×4 block PU.

pixels are stored and shifted cycle by cycle in the interpolation buffer. We use the same way to interpolate the vertical half pixels with 11 FIR filters. In our algorithm, since we will not visit the entire position in the whole refinement window, some redundant interpolations appear in certain pixels in the quarter precision. To avoid redundant interpolate operation, we remove those redundant bilinear filters, which are from 106 (no positions skipped) to 68 (no positions redundant). Thus, the 36% of bilinear filters that each includes an adder and a shifter can be saved by using the proposed algorithm.

Because of the irregular search pattern used in the second step, the adaptive selection should be done before the pixels are sent into PU—one is the overhead by applying fast FME algorithm and the others are the early termination unit and comparison unit. In the former one, the way to predict threshold is the same but different in check time. In the later one, we should know not only the best position but also the second and third places.

TABLE I
SIMULATION RESULT WHEN $\mathrm{QP} = 28$. SPEED-UP IS ONLY FOR THE FRACTIONAL ME PART. ALL RESULTS ARE RELATIVE TO THE REFERENCE SOFTWARE. RATE DISTORTION OPTIMIZATION IS OFF

QP = 28		stefan	mobile	foreman	coastguard	news
Original fast algorithm	△PSNR	-0.09	-0.11	-0.07	-0.04	-0.06
	△bit rate(%)	2.2843	2.36407	1.780915	1.070159	2.23494
	speedup	2.34227	2.24167	2.361651	2.283373	2.24787
Hardware modification	△PSNR	-0.07	-0.07	-0.05	-0.03	-0.1
	△bit rate(%)	2.35577	2.73152	2.057679	1.3326	1.62256
	speedup	2.23192	2.14755	2.256368	2.19521	2.12451

VII. ANALYSIS AND IMPLEMENTATION RESULTS

A. *Performance Analysis and Comparison*

Table I shows the simulation results under constant QP that is the worst case among our simulation for $\mathrm{QP} = 28$ to 40. We integrate our algorithm into the reference software and use the full search algorithm for integer ME for fair comparison. For the original fast algorithm case, it greatly reduces computational complexity but only leads to a small amount of quality loss. The main reason for the quality loss is the that coverage of our search window is not big enough. Thus, some positions cannot be arrived by our fast algorithm. However, it is still acceptable since the loss is still small. For the fast algorithm modified for hardware design, we get slower encoding speed but smaller PSNR drop due to the decreased probability for early termination.

To compare our proposed original algorithm to other fast fractional ME algorithms (2SS in [8] and FSIP in [4]), we turn on the rate control option as other fast algorithms and list the results in Table II. We can find that our proposed algorithm is the most accurate and fastest one. In lower bit rates, the QP for every frame is usually very large. Thus, due to the QP weighted factor used in early termination, we may get the larger threshold and result in better speedup. In addition, with a larger threshold, the resulted prediction error will not be observed because the quantization error in bigger QPs is too large.

B. *Implementation Result*

The proposed FME architecture for H.264 is implemented by Verilog and synthesized in UMC 0.18-μm technology at 100 MHz. The details of every part are listed in Table III. The latency per MB can be calculated as follows if all 41 modes do the FME:

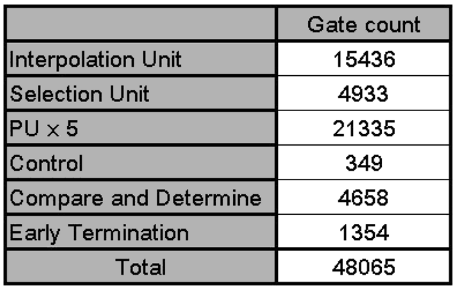
$$\begin{aligned}\text{Latency per MB} = {} & [21 \times 16 + 21 \times 2 \times 8 + 29 \times 8 \\ & + 29 \times 2 \times 4 + 29 \times 4 \times 2 \\ & + 45 \times 2 \times 2 + 45 \times 4] + [17 \times 16] \\ = {} & 2000 \text{ cycles}.\end{aligned}$$

For such case, our design can process 50k MB/s in 100 MHz and is sufficient to support SDTV format in 30 Hz for one reference frame. When compared with other hardware designs [6] with the same algorithm as the reference software, our design has slight quality loss but is 14% faster and 40% smaller using the same technology at the same operating frequency.

TABLE II
COMPARISON BETWEEN DIFFERENT FAST ALGORITHMS FOR FRACTIONAL ME. SPEEDUP IS ONLY FOR THE FRACTIONAL ME PART. ALL RESULTS ARE RELATIVE TO THE REFERENCE SOFTWARE. RATE DISTORTION OPTIMIZATION IS OFF

Rate control enable		64kbps			128kbps			256kbps			512kbps		
		2SS	FSIP	**Our**	2SS	FSIP	**Our**	2SS	FSIP	**Our**	2SS	FSIP	**Our**
Foreman	△PSNR	-0.17	-0.19	**-0.02**	-0.17	-0.17	**-0.02**	-0.1	-0.1	**-0.04**	-0.13	-0.14	**-0.09**
	Speedup	2	2.62	**4.52**	2	2.55	**4.52**	2	2.85	**3.91**	2	2.7	**3.52**
Coastguard	△PSNR	-0.05	-0.07	**-0.01**	-0.02	-0.02	**-0.01**	-0.03	-0.04	**-0.02**	-0.05	-0.05	**-0.05**
	Speedup	2	3	**3.53**	2	2.89	**3.53**	2	3.01	**3.18**	2	2.82	**2.83**

TABLE III
IMPLEMENTATION RESULT OF PROPOSED ARCHITECTURE



	Gate count
Interpolation Unit	15436
Selection Unit	4933
PU x 5	21335
Control	349
Compare and Determine	4658
Early Termination	1354
Total	48065

VIII. CONCLUSION

In this paper, we propose a fast sub-pel ME and VLSI architecture design for H.264/AVC. By taking advantage of the unimodal error surface, the proposed algorithm can significantly decrease more than 50% computational complexity and with only 0.1–0.2 dB PSNR degradation. The corresponding architecture can significantly decrease the total number of 4×4 block PU by reducing the candidates in the same step and speed up the search process by modified early termination technique. The resulting architecture achieves the slight video quality loss but nearly 40% area saving and 14% time saving when compared to the previous one.

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] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion-compensated interframe coding for video conferencing," in *Proc. NTC 81*, New Orleans, LA, Nov./Dec. 1981, pp. C9.6.1–C9.6.5.
[3] C. Zhu and K.-K. Ma, "A new diamond search algorithm. for fast block-matching motion estimation," *IEEE. Trans. Image Process.*, vol. 9, no. 2, pp. 287–290, Feb. 2000.
[4] H. M. Wong, O. C. Au, and A. Chang, "Fast subpixel inter_prediction - based on the texture direction analysis," in *Proc. ISCAS*, 2005, vol. 6, pp. 5477–5480.
[5] C.-C. Cheng, Y. J. Wang, and T.-S. Chang, "A fast fractional pel motion estimation algorithm for H.264/AVC," in *Proc. VLSI/CAD*, Taiwan, R.O.C., 2005, pp. 181–184.
[6] T. C. Chen, Y.-W. Huang, and L. G. Chen, "Fully utiliized and reusable architecture for fractional motion estimation of H.264/AVC," in *Proc. ICASSP*, 2004, vol. 4, pp. 9–12.
[7] Z. Wei, B. Jiang, X. Zhang, and Y. Chen, "A new full-pixel and sub-pixel motion vector search algorithm for fast block-matching motion estimation in H.264," in *Proc. Int. Conf. Image Graph.*, Dec. 2004, pp. 345–348.
[8] B. Zhou and J. Chen, "A fast two-step search algorithm for half pixel motion estimation," in *Proc. 10th IEEE ICECS 2003*, vol. 2, pp. 611–614.
[9] JM8.2. Reference Software of JVT.
[10] A. Tourapis and P. Topiwala, "Sub-pel ME for enhanced predictive zonal search," MPEG/JVT Meeting, Doc. JVT-Q079, Oct. 2005.
[11] Z. Chen, P. Zhou, and Y. He, "Fast motion estimation for JVT," MPEG/JVT Meeting, Doc. JVT G-016, Mar. 2003.

Yu-Jen Wang received the B.S. and M.S. degrees in electronics engineering from National Chiao-Tung University, Hsinchu, Taiwan, R.O.C., in 2004 and 2006, respectively.

After graduation, he joined M-Star, Inc., Hsinchu. His major research interests include H.264/AVC video coding, digital signal processing, and associated VLSI architecture design.

Chao-Chung Cheng received the B.S. and M.S. degrees in electronics engineering from National Chiao-Tung University, Hsinchu, Taiwan, R.O.C., in 2003, and 2005. He is currently working toward the Ph.D. degree in the Graduate Institute of Electronics Engineering, National Taiwan University, Taiwan, R.O.C.

His research interests include digital signal processing, video system design, and computer architecture.

Tian-Sheuan Chang (S'93–M'06) received the B.S., M.S., and Ph. D. degrees in electronics engineering from National Chiao-Tung University, Hsinchu, Taiwan, R.O.C., in 1993, 1995, and 1999, respectively.

He is currently with the Department of Electronics Engineering, National Chiao-Tung University, as an Assistant Professor. From 2000 to 2004, he worked at Global Unichip Corporation, Hsinchu, Taiwan. His research interests include IP and SOC design, VLSI signal processing, and computer architecture.