A LOW COST SINGLE-PASS FRACTIONAL MOTION ESTIMATION ARCHITECTURE USING BIT CLIPPING FOR H.264 VIDEO CODEC

Giwon Kim, Jaemoon Kim, Chong-Min Kyung

Dept. of EECS at KAIST
{gwkim, jmkim}@vslab.kaist.ac.kr, kyung@ee.kaist.ac.kr

ABSTRACT

As the video resolution increases, high computational complexity of the fractional motion estimation (FME) introduces difficulty to meet real-time constraints in a video coding. In this paper, we proposed a single-pass FME algorithm and its architecture with low hardware cost and negligible loss of the image quality. The proposed algorithm directly searches only surroundings of both the predicted fractional motion vector and the search center. To reduce the hardware cost of processing units in the proposed FME architecture, *bit clipping* scheme is applied to processing units reducing the hardware cost by 25%. Experimental results show that the proposed algorithm provides almost the same rate-distortion performance as the full-search algorithm. The result of hardware implementation shows that a quad full high definition video (4096×2160) can be processed in real time (24 frame/sec) using 134k gates when the operating frequency is 250MHz. Compared with the recent work supporting quad full high definition video [8], the proposed FME architecture has shown 70% reduction of the hardware cost.

Keywords— H.264/AVC, Fractional Motion Estimation, Quad Full High Definition, Video Coding

1. INTRODUCTION

Variable block size motion estimation (VBSME) with quarter-pixel accuracy which consists of integer motion estimation (IME) and fractional motion estimation (FME) achieves substantially higher compression efficiency in video coding. IME performs the search of the integer motion vector for 41 sub-blocks (IMV) in coarse resolution, and FME refines each of the 41 integer motion vectors into the quarter-pixel accuracy. FME is divided into two search passes: fractional motion vector (FMV) search with the half-pixel accuracy around the best IMV (first pass) and FMV search with the quarter-pixel accuracy around the best half-pixel FMV (second pass). The improvement of PSNR due to FME with quarter-pixel accuracy is significant, i.e., up to 4 dB [1].

H.264/AVC adopts seven block modes for each macroblock (MB), namely 16×16, 16×8, 8×16, 8×8, 8×4, 4×8, and 4×4. Because FME refines these various block modes with the quarter-pixel accuracy, FME occupies over 45% of the computational complexity of the total encoding process [2]. FME architecture based on the full-search [1], which refines each block mode sequentially, cannot meet the real-time requirement of the high resolution such as HD1080p or QFHD. For example, the architecture proposed in [1] must operate at no less than 405 MHz and 1.4 GHz for HD1080p and QFHD video coding, respectively. Because operating frequency is 100MHz in [1], it is difficult to meet these conditions. Many FME architectures proposed to overcome this problem have high hardware cost. To achieve real-time video coding in the high video resolution, parallel architectures are adopted for FME operation [3] [4]. However, parallel FME architectures are very expensive; they occupy 43.6% of the whole H.264/AVC encoder in [3].

To reduce the increased hardware cost of FME, single-pass FME algorithms have been proposed [5] [6] [8]. In [5], the scheme, which searches directly six quarter pixels, reduces the hardware cost as the number of PUs is reduced to six. To allow real-time operation in the HD1080p, additional mode reduction scheme is utilized in [6]. After the scheme divides seven block modes into two groups, FME is processed only for the two best modes of each group. In [8], the use of two-tap finite impulse response (FIR) filter instead of six-tap FIR filter in the generation of half pixels leads to high throughput, while the image quality is degraded despite the increased hardware to handle all 49 quarter pixels in the FME search range.

In this paper, we propose a single-pass fractional motion estimation (SPFME) algorithm and its hardware architecture with low hardware cost and negligible loss of image quality. The proposed SPFME algorithm, to increase the throughput with negligible loss of image quality, directly searches ten quarter pixels as candidates of the best fractional motion vectors. A bit clipping scheme is employed to reduce the hardware cost of the proposed FME architecture. Since the difference between current pixel value and reference pixel value is small, the proposed bit clipping scheme reduces the size of each processing unit without degrading the image quality.

This paper is organized as follows. Section 2 proposes the SPFME algorithm and bit clipping scheme. Section 3 explains SPFME architecture. In Section 4, rate-distortion performance of the proposed algorithm and the result of hardware implementation are given, followed by the conclusion in Section 5.

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MEST) (No.2010-0000823).

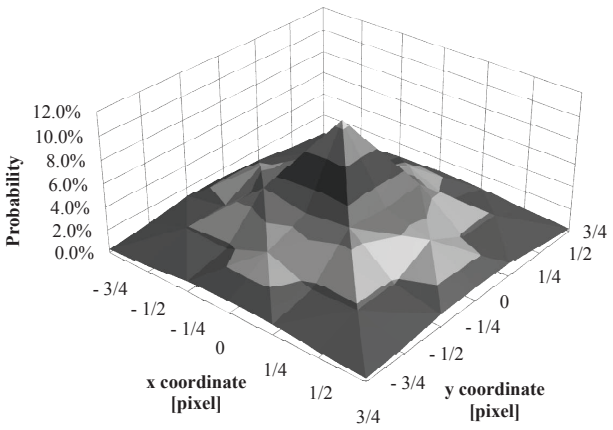

Fig. 1. The average probability distribution of the location of best fractional motion vector, when it is located at neither predicted fractional motion vector nor its four neighbors (top, bottom, left, and right) for the six video sequences: *Aspen*, *sunflower*, *RushFieldCuts*, *station2*, *pedestrian_area*, and *tractor*.

2. PROPOSED ALGORITHM

2.1. Single-Pass Fractional Motion Estimation (SPFME)

There are many fast FME algorithms based on prediction, including a hardware-friendly prediction scheme proposed in [7]. In H.264/AVC, the predicted motion vector ($pred\_mv$) is defined as the median of three neighboring motion vectors. The predicted fractional motion vector ($pred\_frac\_mv$) is extracted from $pred\_mv$ and the best integer motion vector (mv),

$$pred\_frac\_mv = (pred\_mv - mv) \ \ modulo \ \ 4 \qquad (1)$$

where $modulo$ 4 operation is applied to obtain the fractional component by removing the integer part. The basic idea of obtaining the $pred\_frac\_mv$ according to the equation (1) is based on the assumption that most of the best fractional motion vectors ($best\_frac\_mv$'s) lies on either $pred\_frac\_mv$ or its four neighbors (top, down, left and right).

To observe the probability distribution of the $best\_frac\_mv$ when it is located at neither $pred\_frac\_mv$ nor its four neighbors (top, down, left and right), we extracted statistics from six HD1080p video sequences, classified into three groups, over 100 frames when quantization parameter (QP) is 24. The first group includes *Aspen* and *sunflower* sequences with low motion contents. The second group includes *RushFieldCuts* and *station2* with moderate motion contents. The third group includes *pedestrian_area* and *tractor* with high motion contents. The average probability distribution for the six sequences is shown in Fig. 1, where x-axis and y-axis are in quarter-pixel resolution. It shows that the distribution of $best\_frac\_mv$, when it is located at neither $pred\_frac\_mv$ nor its four neighbors, is concentrated at the search center, $(0, 0)$, and its surroundings. Because the contents without motion like background occupy a substantial part of the total image, $best\_frac\_mv$ can be found at either the search center, $(0, 0)$, or its surroundings.

We then performed an experiment to observe the effect of searching $(0, 0)$ and its neighbors in addition to the $pred\_frac\_mv$ with its four neighbors on the image quality for the HD1080p sequences over 100 frames when QP is 24. Fig. 2 shows three different algorithms for fractional motion vector search. (a) searches $pred\_frac\_mv$, its four neighbors (top, down, left, and right), and $(0, 0)$ [5]. (b) is our scheme which additionally searches $(0, 0)$ and its four neighbors in a diagonal direction. (c) is full search scheme of the reference software [9]. In Table 1, a comparison of prediction ratio, which denotes the probability of the $best\_frac\_mv$ being found among the candidate MV locations and ΔPSNR of Fig. 2(a) and (b) in comparison with full search in Fig. 2(c) are shown. In (b) which shows the single-pass fractional motion estimation (SPFME) algorithm proposed in this paper, 10 candidates are directly searched. Average PSNR degradation of SPFME algorithm compared to the full search is negligible, i.e., 0.034dB, while the improvement of image quality of (b) in comparison with (a) is 0.092dB. Because $best\_frac\_mv$, when it is located at neither $pred\_frac\_mv$ nor its four neighbors, is concentrated at $(0, 0)$ and its surroundings as shown in Fig. 1, including $(0, 0)$ with its four neighboring quarter-pixels has proven to be quite effective. We select the four diagonal neighbors of $(0, 0)$ (left-up, right-up, left-down, and right-down) as the additional search candidates to avoid the duplication of search candidates when $pred\_frac\_mv$ is $(0, 0)$.

In the algorithm based on two passes shown in Fig. 2(c), it is difficult to achieve the HD-sufficient throughput, due to the large cycle count per MB for FME. The proposed SPFME algorithm reduces the cycle count per MB for FME into approximately half. To additionally suppress the hardware cost, the so-called *bit clipping* scheme is proposed as follows.

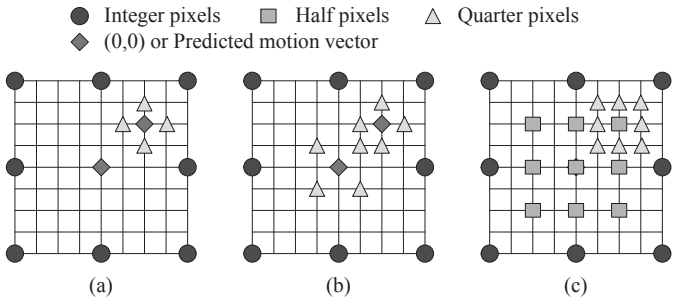

Fig. 2. Various search algorithms with candidate locations for the fractional motion vector are shown. (a) denotes Kuo's algorithm [5], with the predicted fractional motion vector, its four surroundings, and $(0, 0)$. (b) denotes the proposed SPFME algorithm, with the predicted fractional motion vector and its four surroundings, as well as $(0, 0)$ and its four surroundings. (c) is the full search algorithm in the reference software [9].

Table 1. Prediction ratio and ΔPSNR in Fig. 2 (a) and (b)

Sequences	Fig 2. (a)		Proposed Fig 2. (b)	
	Prediction ratio (%)	ΔPSNR (dB)	Prediction ratio (%)	ΔPSNR (dB)
sunflower	79.2	-0.130	83.4	-0.019
tractor	60.8	-0.122	68.0	-0.049
Average	**70.0**	**-0.126**	**75.7**	**-0.034**

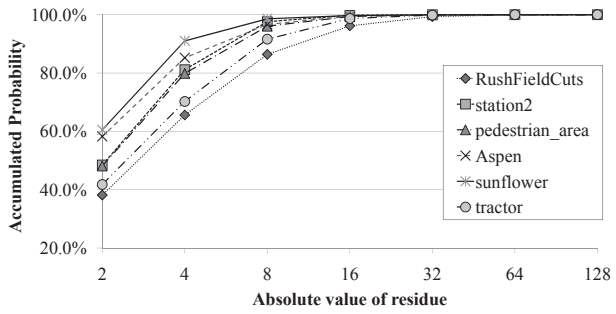

Fig. 3. Accumulated probability distribution of residues.

Table 2. ΔPSNR and Gate counts per PU of 3-bit clipping scheme

The number of clipped bits	ΔPSNR (dB)			Gate counts per PU
	Aspen	*station2*	*tractor*	
3-bit	-0.001	0.001	-0.002	3026
0-bit	0.000	0.000	0.000	4081

2.2. Bit Clipping Scheme

In [1], the group of processing units (PU's) occupies 44% of the gate count of the total FME hardware. In PU, the sum of absolute transformed difference (SATD), which is the evaluation criteria for the decision of the best fractional motion vector, is obtained from the residue. Therefore, the bit length of the residue directly affects the hardware cost in the FME architecture.

Generally each pixel of an image has a value ranging from 0 to 255 and is represented in eight bits. As the residue is the difference between current pixel value and reference pixel value ranging from -255 to 255, nine bits are normally required to represent the residue. However, the bit length of the residue can be reduced if the temporal correlation of pixels between current and reference frame is exploited. To observe the temporal correlation of pixels between previous and current frame, accumulated probability distribution of residue values was obtained from six HD1080p video sequences over 100 frames when QP is 24, as shown in Fig. 3, where x-axis denotes the absolute residue and y-axis denotes the accumulated probability. Fig. 3 shows that the probability of absolute residue being less than 32 is close to 100% for all six sequences, i.e., the maximum difference between current and reference pixel can be represented in 6 bits with almost no loss of information.

The effect of 3-bit clipping scheme on the R-D performance and hardware cost is shown in Table 2. 3-bit clipping confines all residues within $(-31, 31)$, while 0-bit clipping maintains the original residue. 3-bit clipping reduces the gate count per PU by 25% in comparison with 0-bit clipping scheme while ΔPSNR and Δrate are negligible. Table 2 shows that due to high temporal correlation between current and reference pixels, residue values are mostly limited to ± 31. This 3-bit clipping scheme helps suppress the hardware cost of the proposed SPFME.

3. HARDWARE IMPLEMENTATION

FME hardware with high throughput can be achieved using a high degree of parallelism of the interpolation unit. However, it leads to the increase of cost in the whole FME hardware due to the increase of interpolation unit. To design FME hardware with high throughput and minimal hardware cost, the relation between throughput and the hardware cost according to the degree of parallelism of interpolation unit needs to be considered. For a block consisting of n×n integer pixels, n is called *pixel_width* which is defined as the number of integer pixels in a horizontal direction simultaneously processed (interpolated) in a clock cycle. Fig. 4 shows the block diagram of half-pixel interpolation unit and additional hardware elements for the *pixel_width* of 4, 8, and 16. To generate a half-pixel, horizontal finite impulse response (FIR) filter uses the values of six integer-pixels. Vertical FIR filter, which consists of six registers (denoted as squares) and a 6-tap FIR (denoted as a circle), generates a half-pixel from the five half-pixels extracted from horizontal FIR filter or from the six integer-pixels. As the *pixel_width* increases, the half-pixel interpolation unit requires additional hardware elements, i.e., horizontal and vertical FIR filters, as shown in Fig. 4. When the *pixel_width* is four, five horizontal FIR filters and eleven vertical FIR filters are required as denoted by white circles and white vertical bars, respectively. Table 3 shows the number of horizontal and vertical FIR's required for various *pixel_width*s. The increasing the *pixel_width* leads to an increase of throughput due to parallelism. However, with the increase of the number of pixels (n) to be processed, the number of horizontal and vertical FIR filters are also increasing (n+1 and 2n+3, respectively).

Table 3 also shows the gate count of FME architecture and the latency for FME processing according to the *pixel_width* when the operating frequency is 250MHz. In our FME architecture, 'block mode culling' is used. It has four block modes, i.e., 16×16, 16×8, 8×16, and 8×8 mode instead of seven block modes of H.264/AVC. Table 4 shows the effect of 'block mode culling' on PSNR for various video resolutions when QP is 24. As the video resolution increases, the degradation of PSNR reduces. Although the reduction of PSNR is significant in low video resolution, the degradation of PSNR is negligible in the high resolution. Therefore, we choose four block modes to improve the throughput of FME architecture.

In Table 3, as the *pixel_width* increases from 4 to 8, the cycle count per MB is reduced to half, while the gate count of the FME architecture roughly doubles. When the *pixel_width* increases from 8 to 16, gate count roughly doubles, but the cycle count per MB is not reduced to half due to lower utilization of interpolation unit of 8×16 and 8×8 block mode. Each block mode is independently processed because it has a different integer motion vector. In 8×16 and 8×8 block mode, nine horizontal half-pixels are generated, while interpolation unit has 17 horizontal FIR filters when *pixel_width* is 16. Therefore, these two block modes do not need to use eight horizontal FIR filters. As shown in Table 3, real-time encoding of 1080p video sequence is supported when the *pixel_width* is 4, while QFHD resolution cannot be supported in real time. When the *pixel_width* is 8 or 16, it enables real-time encoding in both QFHD and HD1080p resolution. We chose 8, i.e., the eight-pixel interpolation scheme as a trade-off between the cycle count per MB and the gate count of the FME architecture to reduce the hardware cost and support real-time encoding in QFHD.

Fig. 5 shows the number of cycles required for the horizon-

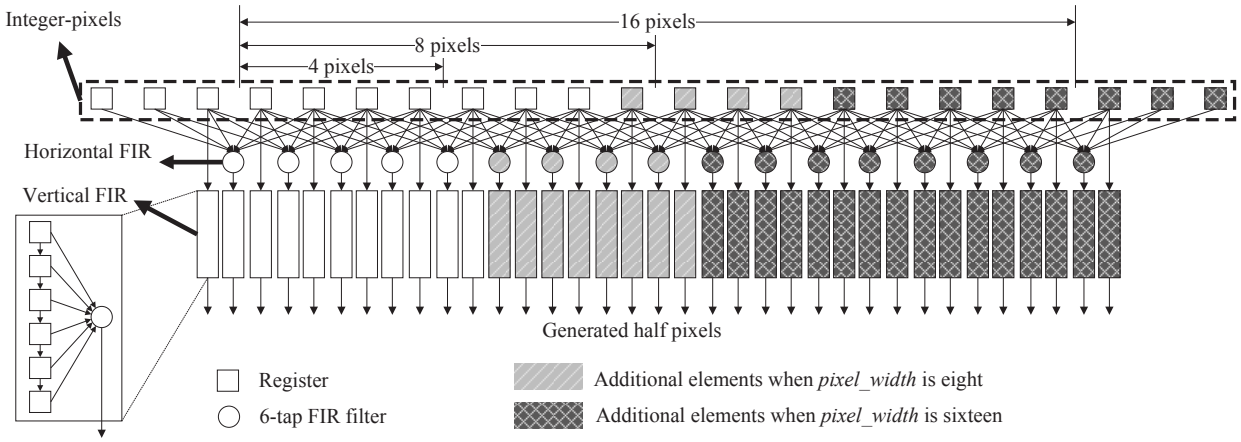

Fig. 4. The block diagram of half-pixel interpolation unit and additional hardware elements according to increment of *pixel_width*.

Table 3. The number of horizontal and vertical FIR's, gate count of FME architecture, latency for FME processing, and support of HD1080p (30 frame/sec) and QFHD (24 frame/sec) according to the *pixel_width*. Maximum latency for HD1080p and QFHD video resolution is 1054 cycles/MB and 301 cycles/MB, respectively.

pixel_width (n)	The number of horizontal FIR (n+1)	The number of vertical FIR (2n+3)	gate count (k)	latency (cycles/MB)	HD1080p support	QFHD support
4	5	11	69	512	Yes	No
8	9	19	134	256	Yes	Yes
16	17	35	267	150	Yes	Yes

Table 4. Effect of 'block mode culling' on PSNR for various video resolutions, as compared to the case of 'no culling'

	ΔPSNR (dB)			
Sequences	QCIF	CIF	HD720p	HD1080p
Aspen	-0.172	-0.123	-0.062	-0.056
sunflower	-0.077	-0.087	-0.042	-0.029
RushFieldCuts	-0.161	-0.176	-0.146	-0.108
station2	-0.132	-0.122	-0.055	-0.013
pedestrian_area	-0.223	-0.153	-0.033	-0.019
tractor	-0.227	-0.185	-0.026	-0.021
Average	**-0.165**	**-0.141**	**-0.061**	**-0.048**

tal and vertical interpolation of one MB to produce half-pixels for each block mode. In Fig. 5(a), with the *pixel_width* set at 8, generation of half-pixels of half of the 16×16 block, i.e., 8×16 block is performed in the vertical direction for 22 cycles. The identical procedure is repeated twice. As a result, 44 cycles are required to execute the whole 16×16 block. The procedure of 8×16 block mode is performed in a similar procedure as the 16×16 block mode, which takes 44 cycles. On the other hand, 16×8 and 8×8 block modes are performed in the vertical direction for 14 cycles in Fig. 5(b). As the identical procedure is repeated four times, 56 cycles are required. After the best block mode having the least SATD value is selected among the four block modes, MC is processed for the selected best block mode in the FME. As the worst case corresponding to 8×16 and 8×8 block mode requires 56 cycles, total required cycle count per MB is $44 + 56 + 44 + 56 + 56 = 256\ \ cycles$.

Fig. 6 shows the proposed FME architecture based on earlier-mentioned SPFME algorithm. "Half-pixel interpolation unit", which consists of horizontal and vertical six-tap FIR filters shown in Fig. 4, generates half-pixels from reference pixels. According to the predicted fractional motion

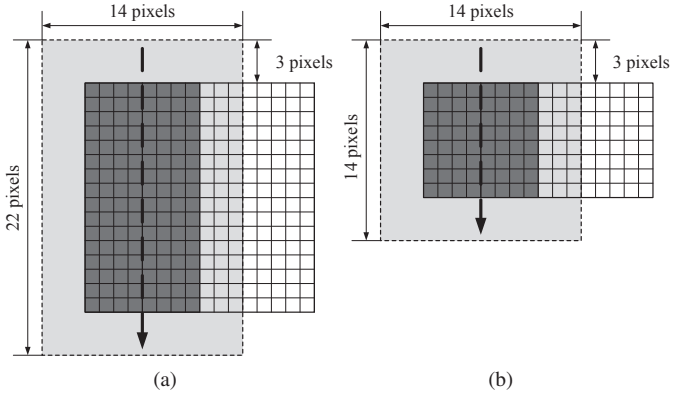

Fig. 5. The number of required cycles per macroblock (MB) in each mode. (a) 16×16 and 8×16 block mode requiring 22×2 = 44 cycles, (b) 16×8 and 8×8 block mode requiring (14×2)×2 = 56 cycles. Dark-shaded region denotes the current sub-macroblock, while the interpolation window denoted as light-shaded region represents a group of integer pixels used to generate fractional pixels. Arrow denotes the direction of interpolation.

vector, $pred\_frac\_mv$, decided by (1), the generated half-pixels are adaptively selected to interpolate 10 quarter-pixels ($pred\_frac\_mv$, $(0,0)$, and their four neighbors, respectively). The quarter-pixels are generated by averaging the value of two neighboring half-pixels. Because quarter-pixels are directly obtained from reference pixels (integer-pixels), FME with single-pass can be achieved.

The bit-clipped processing unit (BCPU) generates residues and extracts SATD for each candidate. Ten BCPUs are divided into two groups. One group of BCPUs generates residues and extracts SATDs for $pred\_frac\_mv$ and its four neighbors. Another group is utilized for residues and SATDs of $(0,0)$ and their four neighbors. Fig. 7(a) shows the details of the BCPU

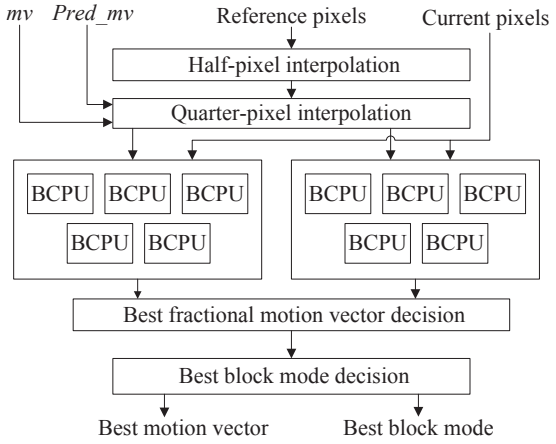

Fig. 6. Proposed FME architecture based on the SPFME algorithm.

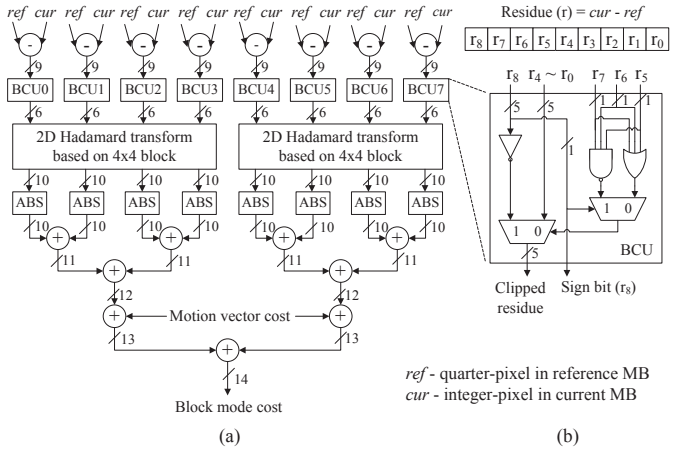

Fig. 7. (a) Proposed bit-clipped processing unit (BCPU). ABS represents the operation taking the absolute value. *ref* denotes the quarter-pixel in reference MB and *cur* denotes the integer-pixel in current MB. The number next to slash denotes the required number of bits. (b) Bit clipping unit (BCU) architecture.

which is based on two 4×4 blocks to accommodate 8×8 block because block modes below 8×8 block are culled in our design. In BCPU, residues are obtained from the difference between candidates in reference MB and integer-pixels in current MB. To reduce the hardware cost of the BCPU, we applied the earlier-mentioned bit clipping scheme to the BCPU. The bit clipping unit (BCU), which confines the nine-bit residue [-255, 255] within six-bit residue [-31, 31], can be represented as the simple architecture. In Fig. 7(b), the clipped residue is selected between the subordinate five bits of nine-bit residue (from r_4 to r_0) and 31 according to the value obtained from combination of r_7, r_6, and r_5. Due to bit clipping scheme, the reduced bit-width in BCPU leads to the reduction of hardware cost.

The "best fractional motion vector decision unit" selects the candidate with the minimum value among ten block mode costs obtained from BCPUs. After motion vector with minimum cost is decided for each mode, "best block mode decision unit" selects the block mode with the minimum value among the accumulated block mode costs as the best block mode. Finally, MC for the selected best block mode is performed.

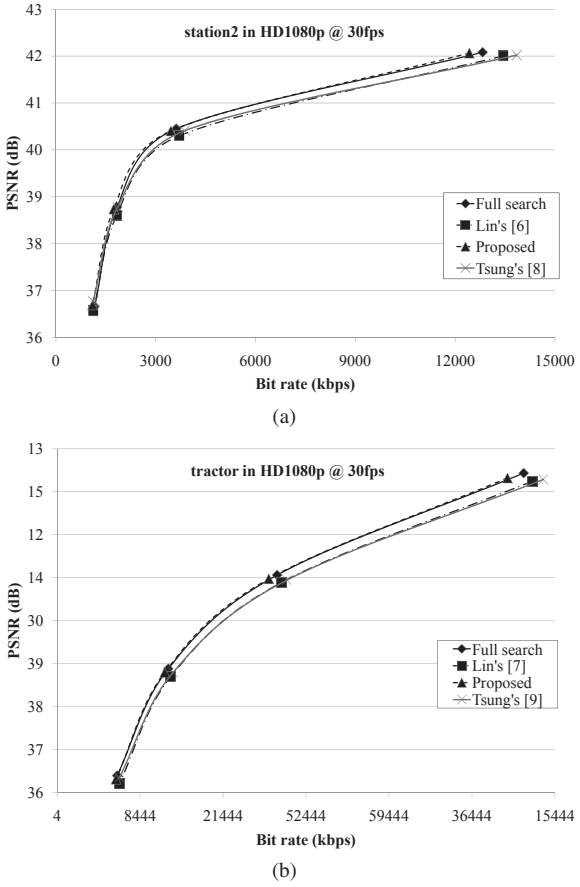

Fig. 8. R-D performance curves of conventional, [6], [8], and proposed FME algorithms. (a) *station2*. (b) *tractor*.

4. EXPERIMENT RESULT

4.1. Performance Comparison

The proposed algorithm as well as the algorithm in [6] and [8] was implemented in JM 14.0, H.264/AVC reference software [9]. The simulation environment is as follows:

1. main profile,
2. CABAC is enabled,
3. RDO is disabled,
4. motion vector search range is [±64, ±64],
5. the number of reference frames is one,
6. tested QP is from 20 to 32,
7. the total number of frames per QP is 100, and
8. IPPP GOP (Group of Pictures) structure is used.

Fig. 8 shows the comparison of R-D performance curve for *station2* and *tractor*. The difference of R-D curve between the proposed SPFME algorithm and full search in reference software is negligible, while the R-D curve of [6] and [8] shows substantial degradation. Table 5 shows the comparison of the algorithms with single pass [6] [8] in terms of the average ΔPSNR and the average Δrate for five HD1080p sequences when QP is from 20 to 32. The average ΔPSNR of the proposed SPFME algorithm is -0.062dB and the average Δrate is -3.29%. On the

Table 5. R-D performance of the proposed FME and FME from [6] and [8] for five HD1080p sequences when QP is from 20 to 32

	Aspen		*sunflower*		*station2*		*tractor*		*RushFieldCuts*		**average**	
	Δrate (%)	ΔPSNR (dB)	Δrate (%)	ΔPSNR (dB)	Δrate (%)	ΔPSNR (dB)	Δrate (%)	ΔPSNR (dB)	Δrate (%)	ΔPSNR (dB)	Δrate (%)	ΔPSNR (dB)
Proposed	-3.79	-0.062	-3.25	-0.044	-4.54	-0.016	-3.06	-0.093	-1.83	-0.094	**-3.29**	**-0.062**
Lin's [6]	-0.19	-0.135	-1.29	-0.190	0.89	-0.121	2.39	-0.187	3.56	-0.157	**1.07**	**-0.158**
Tsung's [8]	4.98	-0.071	9.39	-0.050	-3.15	0.005	4.16	-0.100	6.86	-0.076	**5.71**	**-0.058**

Table 6. Comparison of the proposed algorithm with previous works

	Lin's [6]	Tsung's [8]	Proposed
Max. resolution	HD1080p	QFHD	QFHD
CMOS tech. (μm)	0.13	0.09	0.13
Gate count	68.9k	448k	134k
Operating freq. (MHz)	128.3	280	250
Latency (cycles/MB)	432/264[1)]	156	256
Throughput (kMB/sec)	297/485[1)]	1659	977
TPUA (MB/sec/gate)	4.31/7.04[1)]	3.70	7.29
ΔPSNR (dB)	-0.157	-0.058	-0.062

1) The left and the right part of slash are the worst and the best case, respectively.

other hand, average ΔPSNR of [6] is -0.157dB, and Δrate is 1.07%. The average ΔPSNR of [8] is -0.058dB, and Δrate is 5.71%. Compared with the R-D performance in [6], the improvement of the proposed algorithm is driven from additional search of $(0, 0)$ and its surroundings. The R-D performance of the proposed algorithm is better than that of [8], because the generation of the fractional pixels is performed by 6-tap FIR filters compared to 2-tap FIR filters used in [8].

4.2. Implementation Result

The proposed SPFME architecture was implemented in Verilog, synthesized in TSMC 0.13um technology, and operated at 250MHz. Table 6 shows the comparison of the proposed architecture with previous works with single pass. Our design and [8] lead to the real-time video coding in QFHD video resolution due to the high throughput, while [6] supports until HD1080p video resolution. However, [8] obtains the high throughput at the expense of the significant hardware cost in comparison to our design and [6].

Throughput is computed as the ratio of the operating frequency to latency (operating frequency/latency). Thus, throughput can be increased as reducing latency at the expense of additional hardware for exploiting parallelism. We make throughput per unit area (TPUA) as the evaluation criteria to consider both hardware cost and throughput. The definition of TPUA is as follows.

$$TPUA = \frac{Throughput}{Gate\ \ count}\ \ (MB/sec/gate) \qquad (2)$$

In [6], TPUA has minimum and maximum value due to the variable latency of the selected block mode. Compared with [6] and [8] in terms of TPUA, our design can achieve the highest throughput with low hardware cost. With the proposed design, real-time constraints can be met in the video resolution of QFHD as well as HD1080p at the expense of low hardware cost.

5. CONCLUSION

In H.264/AVC coder, FME architecture with the low hardware cost and the high throughput is significantly important in the real-time video coding with high resolution. In this paper, we proposed FME algorithm and its architecture with the low hardware cost and the negligible loss of image quality. Our SPFME algorithm has the high throughput with negligible loss of image quality. We also proposed a bit clipping scheme for reducing the hardware cost of the processing unit (PU) by 25% in the FME architecture. In our design, the use of block mode over 8×8 leads to substantial reduction of cycle count for MB. Our FME design has 977KMB/sec throughput with 0.062dB PSNR degradation. With TSMC 0.13um, total gate count of the proposed architecture is 134K at 250MHz. As a result, our design can support QFHD video resolution with low hardware cost.

6. REFERENCES

[1] T. Chen, *et al.*, "Fully utilized and reusable architecture for fractional motion estimation of H.264/AVC," in *IEEE Proc. ICASSP*, vol. 4, pp. 9-12, May. 2004.

[2] T. Chen, *et al.*, "Analysis and design of macroblock pipelining for H.264/AVC VLSI architecture," in *IEEE Proc. ISCAS*, vol. 2, pp. 273-276, May. 2004.

[3] T. Chen, *et al.*, "Analysis and architecture design of HDTV720p 30frames/s H.264/AVC encoder," in *IEEE TCSVT*, vol. 16, no. 6, pp. 673-688, Jun. 2006.

[4] C. Wu, *et al.*, "A high performance three-engine architecture for H.264/AVC fractional motion estimation," in *IEEE Proc. ICME*, pp. 133-136, Jun. 2008.

[5] T. Kuo, *et al.*, "SIFME: a single iteration fractional-pel motion estimation algorithm and architecture for HDTV sized H.264 video coding," in *IEEE Proc. ICASSP*, vol. 1, pp. 1185-1188, Apr. 2007.

[6] Y. Lin, *et al.*, "A hardware-efficient H.264/AVC motion-estimation design for high-definition video," in *IEEE TCS I, Reg. Papers*, vol. 55, no. 6, pp. 1526-1535, Jul. 2008.

[7] L. Yang, *et al.*, "Prediction-based directional fractional pixel motion estimation for H.264 video coding," in *IEEE Proc. ICASSP*, vol. 2, pp. 901-904, Mar. 2005.

[8] P. Tsung, *et al.*, "Single-iteration full-search fractional motion estimation for quad full HD H.264/AVC encoding," in *IEEE Proc. ICME*, pp. 9-12, Jun. 2009.

[9] "Joint video team (JVT) reference software version 14.0," http://iphome.hhi.de/suehring/tml/download/old_jm/.