A Hardware-Efficient H.264/AVC Motion-Estimation Design for High-Definition Video

Yu-Kun Lin, Chia-Chun Lin, Tzu-Yun Kuo, and Tian-Sheuan Chang, *Senior Member, IEEE*

***Abstract*—Motion estimation (ME) in high-definition H.264 video coding presents a significant design challenge for memory bandwidth, latency, and cost because of its large search range and various modes. To conquer this problem, this paper presents a low-latency and hardware-efficient ME design with three design techniques. The first technique on integer-pel ME (IME) adopts parallel instead of serial multiresolution search so that we can process 1080 p @ 60 fps videos with $\pm$128 search range within just 256 cycles, 5.95-KB buffers, and 213.7K gates. The second technique on fractional-pel ME (FME) uses a single-iteration six-point search to reduce the cycle count by half with similar gate count and negligible quality loss. The third technique applies a mode-filtering approach to further reduce the bandwidth and cycles and share the buffer of IME and FME. The final ME implementation with 0.13-μm process can support processing of 1080 p @ 60 fps with just 128.8 MHz, 282.6 K gates, and 8.54-KB buffer, which saves 60% gate count, and 68.9% SRAM buffers when compared with the previous design.**

***Index Terms*—Digital circuits, high-definition television (HDTV), H.264, motion estimation (ME), video coding, video signal processing.**

I. Introduction

THE latest video coding standard, MPEG-4 AVC/H.264 video coding [1], provides better coding efficiency than others and is widely adopted in various multimedia devices, ranging from mobile phones to high-definition television (HDTV), in which the variable block size integer-pel motion estimation (IME) and its improved fractional-pel ME (FME) not only contribute a lot for coding efficiency but also dominate the computational loading of the whole encoding process. Thus, various VLSI realizations of ME have been proposed to speed up the process [2]–[11]. However, most of them are only applicable for standard-definition TV (SDTV) size or below. For HD video applications that require a large search range up to [−128, 127] or even larger, direct extension with previous approaches will consume too large area cost, buffers, bandwidth, and cycles. To support a large search range, many fast IME algorithms have been proposed [12]–[17]. However, most of them are not suitable for hardware implementation because of the irregular data flow. Besides, most of these approaches only consider IME or FME only without exploiting their relationship, which may result in extra computational cost.

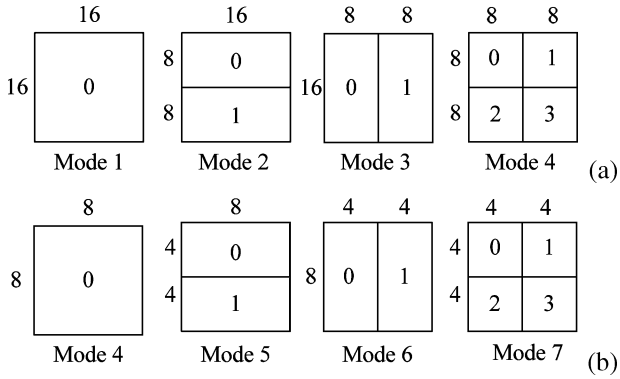

Fig. 1. (a) Four block sizes for a 16 × 16 macroblock. (b) Four subblock sizes for an 8 × 8 subblocks.

To solve the above problems, this paper presents an efficient ME architecture suitable for HD videos by various design techniques, including a parallel multiresolution ME (PMRME) for large search range IME, a single-iteration FME (SIFME) to achieve the lower cycle count, and a mode-filtering algorithm to jointly reduce the IME and FME computations. The cycles are reduced by hardware parallelism and algorithm modification (PMRME and SIFME). Furthermore, we lower requirements of bandwidth and buffer by reusing data within IME as well as between IME and FME (mode filtering). The video quality loss is low by exploiting unequal distribution of motion vectors (MVs). With these approaches, we can save more than half of the area, bandwidth, and buffer costs when compared with previous designs.

The remainder of this paper is organized as follows. We first overview the ME of H.264 and review the previous approaches in Section II. Then, we present the proposed approaches and the simulation performance in Section III. Section IV shows the corresponding architecture. The implementation results and comparisons are listed in Section V. Finally, a conclusion is made in Section VI.

II. Review of ME in H.264 and Previous Approaches

A. ME in H.264

ME in H.264 contains two parts, IME and FME. IME has seven kinds of block size partition, as shown in Fig. 1. To determine the best partition mode, it first checks the mode 16 × 16 to mode 8 × 8. If mode 8 × 8 is chosen as the best mode, the modes with smaller block size as in Fig. 1(b) will be checked. Thus, 41 MVs will be generated by IME and refined by FME to

Manuscript received June 11, 2007; revised September 3, 2007. First published February 2, 2008; last published July 10, 2008 (projected). This work was supported in part by the National Science Council, Taiwan, R.O.C., under Grant NSC-95-2220-E-009-005. This paper was recommended by Associate Editor J. R. (a.k.a. Rong-Jian) Chen.

The authors are with the Institute of Electronics, National Chiao-Tung University, HsinChu 300, Taiwan, R.O.C. (e-mail: yklin@twins.ee.nctu.edu.tw; tschang@twins.ee.nctu.edu.tw).

Digital Object Identifier 10.1109/TCSI.2008.916681

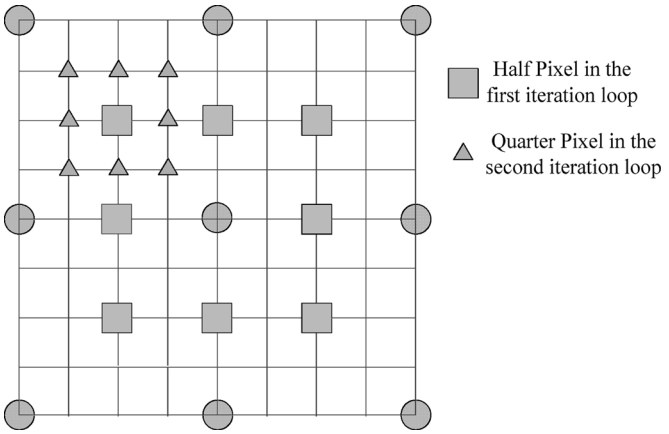

Fig. 2. FME algorithm in reference software [20].

find the best mode. FME will refine these MVs to quarter-pixel precision.

For details, readers can refer to [1].

B. Review of Previous Approaches

For fast IME, various approaches have been proposed [7]–[17], but few can be readily applicable to large search range as used in HDTV. The large search-range requirement will result in longer execution cycles as well as large buffer and high memory access. Previous designs with $[-63, +64]$ search range [18], [19] use the full search method and thus occupies a large area cost. To solve these problems, one promising approach is the multiresolution ME such as that proposed in [7]. In [7], they use three hierarchical levels for searching and refining the MV from the coarse level to the finest level. However, the MV found in the higher level needs to be further refined in the lower level. This implies that the search is a sequential process that will increase the cycle count and thus decrease the hardware utilization and throughput. Besides, a full search range sized buffer is still needed because of the dependency between the three hierarchical levels. Then, the required bandwidth is still too large because of poor data reuse of the refinement process. In [8], a modified three-step algorithm is used to decrease the search points for low power, but still consumes large area cost and memory. The work in [9] also used the subsampling techniques to reduce the hardware cost; however, the two-stage architecture results in a longer cycle count. In [11], the two-stage flow and the irregular search range cause the difficulty of external data transfer.

For fast FME, most approaches follow the two-step approaches as in the reference software [20] shown in Fig. 2, which needs a total of 17 search points for FME. Although this algorithm is suitable for hardware [21], it has two drawbacks. First, the nine search points in each step result in area-costly 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.

To speed up FME, many fast FME [22]–[27] algorithms are proposed to speed up the process. However, these algorithms [22]–[25] are software-oriented with irregular data flow and thus are not suitable for hardware design. Our previous work [26] is more suitable for hardware and can reduce the PU from nine to five to save hardware cost. However, all of these algorithms suffer from long computation cycles due to the two iterative search loops, one on half-pels and one on quarter-pels. On the other hand, single iteration algorithms like [28] and [29] exhibit bad performance due to poor interpolation accuracy. The design in [30] increases the throughput by the cost of large area and memory bandwidth. In summary, the hardware implementation of these fast algorithms only reduces the processing element but does not reduce the total cycle count or degrade the quality a great deal. This problem will pose a strict limit on the HD video applications since FME will take more cycles than IME does and thus dominate the whole pipelining cycle time.

III. Proposed Hardware-Based Approaches

A. Overview of the Approaches

Our approach includes three parts, including PMRME, SIFME, and mode filtering, to trade off the minimum quality loss while reducing the area cost and memory bandwidth a great deal.

First, the proposed PMRME adopts the parallel multiresolution search method. We tweak the multiresolution search with two hardware design techniques to solve these problems mentioned above. First, we parallel search all resolution levels instead of serial one. This brings the benefit of a low cycle count, which is crucial to HD video. Second, we adjust the search center of different levels for data reuse purpose. In the three resolution levels, the first level without data subsampling covers the search range for the most frequently occurring MVs. Thus, this level has the search center at the MV predictor (MVP). On the other hand, the last two levels with data subsampling cover the large search range to find the rarely occurred large MV. These two levels have good data reuse by fixing the searching center at (0,0) as in other ME designs. The concept behind our algorithm is the unequal distribution of MV that most of them are near the MVP. Thus, a full search around the MVP can find most of the final MVs while the rest can be found in the coarse search. With the above parallel method and search center arrangement, we can save latency and at least 92.4% of memory buffer when compared with the direct approach [18]. In addition, data within two out of three memory buffers are highly reused, and thus PMRME can save about 63.7% of memory bandwidth. Thus, we can have a low-area-cost IME design with low buffer and bandwidth requirements.

Second, with the reduction of IME cycles, FME now occupies a significant part of the run-time in inter prediction and thus needs speedup as well. The latency of FME is determined by the number of MVs or modes to be tested and the cycle count of each block mode.

To reduce the cycle count of each block mode, we propose SIFME that 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 [20]. Thus, we can reduce the number of PUs since we only search six candidates. Besides, the cycle count is also halved by using only one search step.

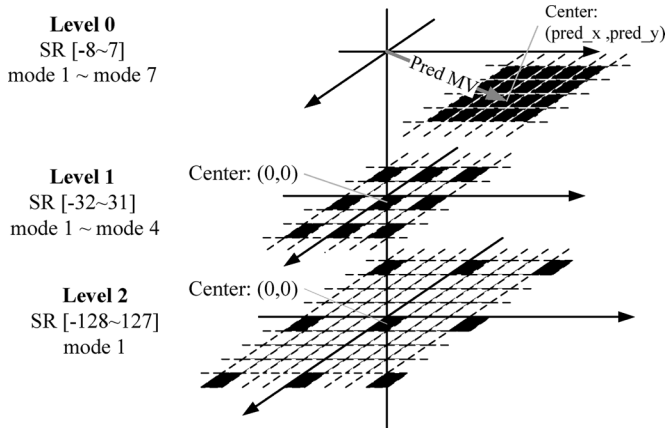

Fig. 3. Three-level new multiresolution algorithm.

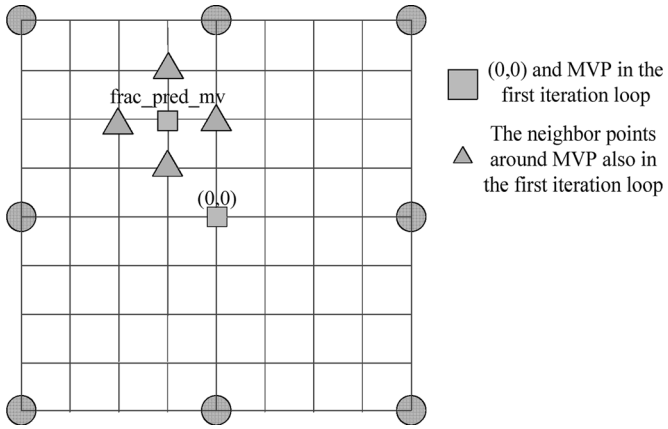

Fig. 4. Proposed SIFME on two square points (0, 0) and $\mathrm{frac\_pred\_mv}$ and four triangle points around $\mathrm{frac\_pred\_mv}$ in one quarter-pel distance.

To reduce the number of MVs or modes to be tested in FME, we propose a mode-filtering approach such that the IME stage only sends the two best modes to the FME stage. Therefore, the FME stage just needs to check at most 18 MVs instead of 41 MVs in [18]. This brings the benefit of 56.4% cycle count reduction.

With the above considerations, we can have a ME design with low latency, cost, and memory bandwidth.

B. PMRME

PMRME includes three levels, and all of them are independent of each other, as illustrated in Fig. 3.

In the coarsest level, level 2, the search range (SR) is the largest $[-128 \sim 127]$ and is centered on the original point (0, 0). This enables the regular memory reuse between successive MB processing as used in most ME designs [31]. This level uses the 16:1 sampling, and thus we only choose the 16×16 mode (mode 1 in Fig. 1) since other modes will contain too fewer pixels for SAD calculation and may result in poor mode decision.

In level 1, the SR is reduced to $[-32 \sim +31]$ and is centered on (0, 0) for memory reuse. This level uses the 4:1 sampling and thus we only choose the 16×16 to 8×8 mode (mode 1 to 4 in Fig. 1) for the same reason as in level 2.

In the finest level, level 0, the SR is set to $[-8 \sim +7]$. However, unlike the other two levels with (0, 0) center, we choose the MVP as the center due to its higher probability to find the final MV here. Thus, we do not subsample data in this level and thus enable search for all variable block-size modes.

In the three parallel levels, level 2 provides a large search range for high motion blocks with coarse precision. It is useful for very high motion blocks and can find a sufficient though rough MV candidate. Also, level 1 can provide a medium search range but a finer MV precision. With these two large search levels, the motion search algorithm of level 0 can converge to the true MV quickly by effects of MVP. If only level 0 is used, it is difficult to trace the high motion blocks because the MVP cannot follow up the real motion effectively in this case. The performance and quality analysis of PMRME can be found in our preliminary work [32].

TABLE I
PREDICTION ACCURACY OF MVS COMPARED WITH THE FULL-SEARCH FME ALGORITHM

720p size, 300 frame, IPPP, RDO off, SR=±32				
QP	mobile calendar	shields	park run	Stockholm
10	58.62%	48.77%	64.65%	73.49%
16	65.68%	55.27%	66.65%	76.07%
22	77.74%	66.78%	67%	78.01%
28	87.46%	87.61%	72.84%	84.86%
34	91.31%	92.34%	80.83%	91.2%
40	92.65%	93.71%	85.92%	94.3%
Avg.	78.91%	74.08%	72.94%	82.9%

TABLE II
SEARCH POINT COMPARISONS FOR DIFFERENT ALGORITHMS

	search point
JM [20]	17
[23]	6+multiple diamond search (Total <=11)
[22]	6 + multiple diamond search
[26]	8~9
proposed	6

C. SIFME: Single Iteration Fractional ME Algorithm

Inspired by the unequal distribution of MVs, we propose SIFME that searches six candidates in only one step without a refined search, as shown in Fig. 4. The candidate with the lowest cost will be selected as the best one.

It first calculates the fractional predicted MV ($\mathrm{frac\_pred\_mv}$) as

$$\mathrm{frac\_pred\_mv} = (\mathrm{pred\_mv} - \mathrm{mv})\%\beta \tag{1}$$

where $\mathrm{pred\_mv}$ (MVP) here is defined as the fractional pixel unit, mv is the integer pixel MV after IME process and is in fractional-pel units, % is the mode operation, β is 4 in 1/4-pel case and is 8 in 1/8-pel case, and $\mathrm{frac\_pred\_mv}$ is the predicted fractional MV and indicates only fractional position.

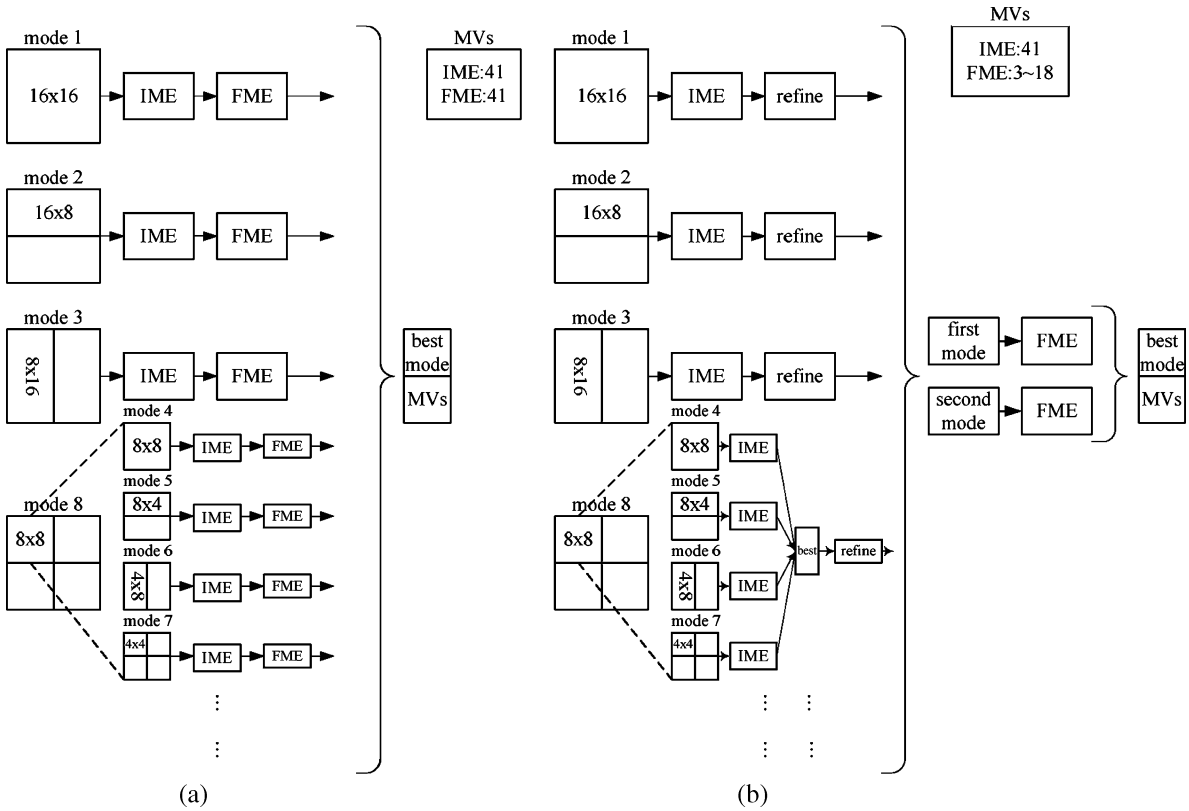

Fig. 5. (a) Original flow of IME and FME in reference software [20]. (b) Our proposed mode-filtering algorithm.

The six candidates include (0, 0), frac_pred_mv from (1), and four diamond points around frac_pred_mv. (0, 0) is included for low texture and low motion sequences. Other search points are placed around frac_pred_mv since the best fractional MV is more probable around frac_pred_mv than around (0, 0).

Table I shows the prediction correctness compared with the algorithm in the reference software. The prediction accuracy is defined as if the fractional MV by the proposed approach is the same as that by the full search algorithm of the reference software. We use four 720-p-sized test sequences with 300 frames under different QPs. The reference software is JM9.0 [20]. This result shows that it has more than 70% prediction accuracy in average though the proposed one has ignored more than 64 % search points.

Table II shows the search point comparisons with other algorithms. The proposed algorithm searches the fewest points compared with other search algorithms. Besides, our approach does not need the second step search and saves the additional interpolation time, which is very suitable for hardware designs.

The other preliminary performance analysis of SIFME can be found in our preliminary work [33].

D. Mode Filtering

Fig. 5(a) presents the general flow of IME and FME in the reference software that IME sends the MV to FME for refinement. After all possible modes and MVs are generated, the best mode and its MV are chosen in the final step of FME. Thus, the IME and FME module both processes 41 MVs.

To reduce the complexity, we select the two best modes instead of all modes for FME refinement, as shown in Fig. 5(b). One mode is chosen from mode 1 to mode 3 in Fig. 1, and the other mode is selected from mode 1 to mode 7. With this, only 3 to 18 MVs instead of 41 MVs are computed in FME, which saves 56.4%–73.2% computing cycles. In [21], a similar concept but more complex procedure has been proposed. Our method can achieve better quality and lower cycle count than that in [21] because we only select two instead of three candidates and only the best candidate for the 8×8 and subblock case is considered in the final best mode selection. Besides, the method also increases the overall ME pipelining efficiency because it can reduce the cycle count of FME to be similar to that of the IME stage.

E. Video Quality Analysis

Table III presents the simulation results for different algorithm combinations: PMRME, mode filtering, and SIFME. In these results, we also include the bit truncation in this design to reduce the hardware cost. The simulation environments are as following: no rate-distortion optimization (RDO), the sequence type is IPPP, and the search range is $[-128, 127]$. All of the simulation results are compared with those of the default full-search algorithm in JM9.0 [20]. The result in this table only shows the average performance under different QPs. The test sequences are all 720-p resolution including Stockholm, parkrun, mobile calendar, and shields. The frame rate is 30, and 100 frames are coded.

TABLE III
PSNR AND BIT-RATE CHANGE FOR PROPOSED ALGORITHMS COMPARED WITH FULL SEARCH

	Frame size	720p			
QP		PMRME	PMRME +MODE FILTERING	PMRME +MODE FILTERING +Bit-Truncation(5 bits)	PMRME +MODE FILTERING +Bit-Truncation (5 bits) +SIFME
QP16	PSNR inc.(db)	-0.0025	-0.085	-0.075	-0.0975
	Bit rate inc. (%)	-1.02	-1.66	-0.66	1.08
	Sharing Rate of L0 Buffer (%)	n.a.	96.95	96.08	96.10
QP20	PSNR inc.(db)	0	-0.095	-0.0825	-0.117
	Bit rate inc. (%)	-0.49	-1.24	-0.013	1.80
	Sharing Rate of L0 Buffer (%)	n.a.	97.84	96.86	96.87
QP24	PSNR inc.(db)	-0.0075	-0.08	-0.0675	-0.1025
	Bit rate inc. (%)	-0.33	-1.11	0.32	2.27
	Sharing Rate of L0 Buffer (%)	n.a.	98.36	97.77	97.75
QP28	PSNR inc.(db)	-0.005	-0.0625	-0.0525	-0.0925
	Bit rate inc. (%)	0.20	-0.57	0.76	2.52
	Sharing Rate of L0 Buffer (%)	n.a.	98.78	98.21	98.19
QP32	PSNR inc.(db)	-0.01	-0.0525	-0.045	-0.09
	Bit rate inc. (%)	1.56	1.14	2.18	2.90
	Sharing Rate of L0 Buffer (%)	n.a.	99.00	98.30	98.31
Avg	PSNR inc.(db)	-0.005	-0.075	-0.0645	-0.1
	Bit rate inc. (%)	-0.017	-0.69	0.52	2.11
	Sharing Rate of L0 Buffer (%)	n.a.	98.18	97.44	97.44

Table III shows the PSNR change, bit-rate increasing, and "Sharing Rate of L0 (Level 0) buffer." The sharing rate of level 0 denotes the percentage that FME can directly reuse the level-0 search range buffer for computation to save memory bandwidth. This sharing occurs once the final MV is within the level-0 search range. In our design, the sharing rate is at least 90%, and the higher QP will have higher sharing rate and thus can save more power and bandwidth.

In this table, we can find that the performance of PMRME is almost the same with full search. The average PSNR drop is only 0.005 dB, and the bit rate is even decreasing when compared with full search. This is because the PMRME ignores smaller blocks in levels 1 and 2 and prefers larger block which results in a bit-rate decrease. As for mode filtering, the algorithm also prefers to select larger block size, so the decreasing bit rate is more obvious. Oppositely, the PSNR drop is a little more serious than using only PMRME. However, the worst quality drop is only 0.095 dB. While considering the bit-truncation technique, the influence on PSNR is only 0.064 dB, and the increasing bit rate is 0.52% in average. Finally, we combine all proposed techniques, the PSNR quality is almost the same, and the bit-rate quality drop is small, increasing to 2.11% on average

IV. PROPOSED ARCHITECTURE

A. Integrated Motion-Estimation Architecture

Fig. 6 shows the total block diagram of the full ME modules. It contains IME, FME, several memory buffers, and external data access interface. The whole flow is as described in Fig. 5(b).

To enable the data reuse between IME and FME, the IME module has three internal SRAMs for reference pixels storage. When the IME search of an MB is completed, its macroblock information is sent to FME. Moreover, the reference pixels in level-0 SRAM is also sent to FME. However, instead of moving

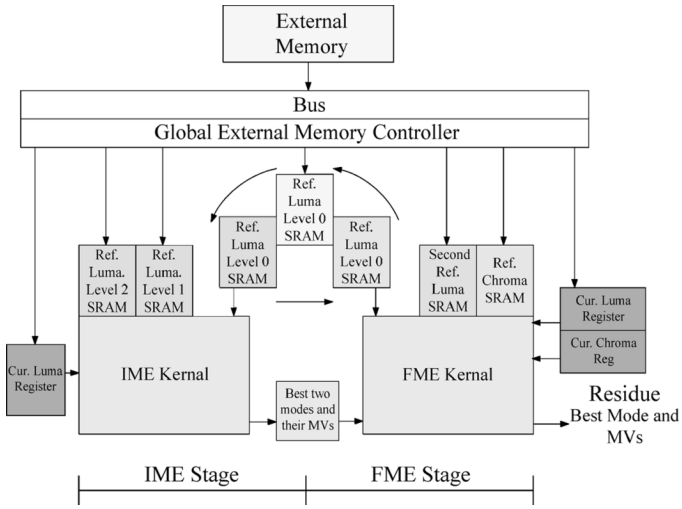

Fig. 6. Block diagram of IME and FME.

data, we use three SRAMs as the level-0 buffer and swap them with a ping-pong buffer concept. The three level-0 buffers includes one for IME level-0 reference, one for FME, and one for loading new data from external memory. Whenever the IME stage completes the coding of the first MB, the buffer for level-0 reference for the first MB is changed as the FME reference in the next stage. At the same time, the buffer for current FME reference is changed to load the data from the third MB from external memory for further use. The buffer that is now filled with the reference data for the second MB is switched for IME level-0 reference. With the above ping-pong buffers, we can share the level-0 data of IME with the FME, and no additional memory access time is necessary. Besides, the data in level-0 buffers can be reused by FME for more than 90% of MBs according to the sharing rate in Table III. With the above arrangement, all of these

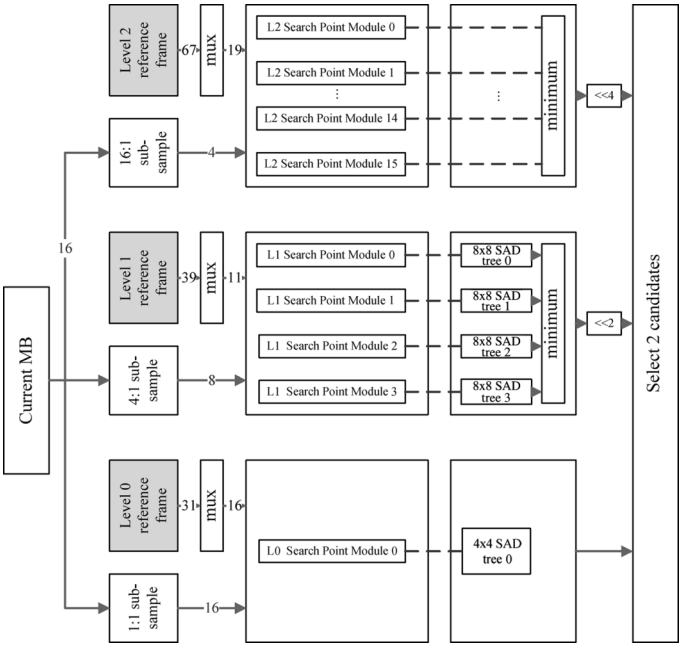

Fig. 7. Proposed architecture of IME stage.

TABLE IV
MEMORY AND BANDWIDTH REQUIREMENT EQUATION FOR EACH LEVEL. THE $\mathrm{MB_{SIZE}}$ IS 16. BESIDES, $\mathrm{SR_{L0}}$, $\mathrm{SR_{L1}}$, AND $\mathrm{SR_{L2}}$ ARE 16, 64, AND 256 IN RESPECTIVELY

Memory cost	buffer size	BW(per MB)
Level 0	$(\mathrm{SR_{L0}}+\mathrm{MB_{size}}+5)$ * $(\mathrm{SR_{L0}}+\mathrm{MB_{size}}+5)$*8	$(\mathrm{SR_{L0}}+\mathrm{MB_{size}}+5)$ * $(\mathrm{SR_{L0}}+\mathrm{MB_{size}}+5)$*8
Level 1	$(\mathrm{SR_{L1}}/2+\mathrm{MB_{size}}/2-1)$ * $(\mathrm{SR_{L1}}/2+\mathrm{MB_{size}}/2)$ *$(\mathrm{Pixel\_Depth_{L1}})$	$(\mathrm{SR_{L1}}/2+\mathrm{MB_{size}}/2-1)$ * $(\mathrm{SR_{L1}}/2+\mathrm{MB_{size}}/2)$* $(16/(64+16))$ *8
Level 2	$(\mathrm{SR_{L2}}/4+\mathrm{MB_{size}}/4-1)$ * $(\mathrm{SR_{L2}}/4+\mathrm{MB_{size}}/4)$ *$(\mathrm{Pixel\_Depth_{L2}})$	$(\mathrm{SR_{L2}}/4+\mathrm{MB_{size}}/4-1)$ * $(\mathrm{SR_{L2}}/4+\mathrm{MB_{size}}/4)$ *$(16/(256+16))$ *8
Direct design	$(\mathrm{SR}+\mathrm{MB_{size}}-1)$ $(\mathrm{SR}+\mathrm{MB_{size}})$ *8	$(\mathrm{SR}+\mathrm{MB_{size}}-1)$ $(\mathrm{SR}+\mathrm{MB_{size}})$ *$(16/(256+16))$ *8

data can be reused as much as possible and reduce the bandwidth a great deal.

B. Architecture for PMRME

Fig. 7 shows the proposed IME architecture. All three levels can be computed in parallel. A 16 × 16 current block is shared by three levels. The memory size and bandwidth for three reference frame buffers are listed in Tables IV and V. The bit width of the memory buffer of levels 1 and 2 are truncated while that of level 0 is not. The reason for this is that the level-0 data can be reused by the following FME hardware if the best MV falls in level 0. Table IV presents the equation of buffer size and the memory access requirement for each level and direct implementation [18]. The $\mathrm{MB_{size}}$ in the table is 16. In addition, $\mathrm{SR_{L0}}$, $\mathrm{SR_{L1}}$, and $\mathrm{SR_{L2}}$ are, respectively, 16, 64, and 256. We should note that the buffer size for direct implementation is the search range size. As for level 0, the buffer size is a little larger than the search range because it includes the neighboring pixels for FME interpolation. However, in the case of levels 1 and 2, the memory size is only one-fourth and one-sixteen of their search

TABLE V
MEMORY AND BANDWIDTH REQUIREMENT IS FOR DIFFERENT FRAME SIZE. THE SAVING IS COMPARED WITH THE DIRECT DESIGN [18]. THE MAXIMUM SEARCH RANGE IS [−128, 127]

Memory cost	for 720p		for 1080p	
	buffer size	BW(per MB)	buffer size	BW(per MB)
Level 0 (Kbyte)	1.369	1.369	1.369	1.369
Level 1 (Kbyte)	0.975	0.312	1.170	0.312
Level 2 (Kbyte)	2.8475	0.268	3.417	0.268
Total (Kbytes)	5.1915	1.949	5.956	1.572
Direct design	73.712	4.336	73.712	4.336
Saving (%)	92.95	55	91.91	55

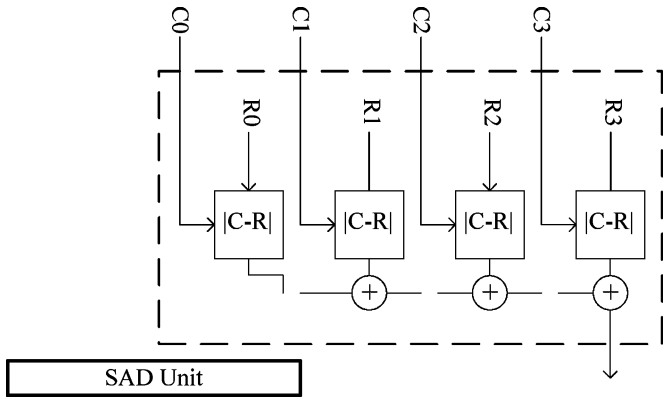

Fig. 8. Basic 4p-SAD unit can accumulate the SAD of four pixels.

range by the subsampling techniques. In addition, the bit-truncation technique also reduces 25%–37.5% of buffer size if two or three bits are truncated. As for the memory bandwidth, by the level-C data-reuse scheme in [31], the direct implementation needs to update $(\mathrm{SR}+\mathrm{MB_{size}}-1)*16$ pixels. Therefore, the larger search range results in the lower proportion of update rate. Thus, only $16/(64+16)=20\%$ data in level-1 SRAM should be updated when the coding MB changes with above approach. As for level 2, only $16/(256+16)=5.88\%$ data should be updated. Table V shows the real buffer size and memory bandwidth requirement for 720-p and 1080-p video. The proposed algorithm can save over 91.91% buffer in the 720-p case and 55% of bandwidth in the 1080-p case by subsampling and bit-truncation when comparing to [18] that also uses level C data-reuse scheme. If the bus width is 128 bits, it only needs 121 cycles per MB to transfer the required data from external memory to SRAM.

In this architecture, all computations are decomposed as the combinations of 4 × 4 blocks. The basic processing unit is the four-pixel SAD (4p-SAD) unit, which can process the SAD of four pixels, as depicted in Fig. 8. With this, every level can be easily implemented by regularly composed SAD units. As Fig. 7 presents, L0 (level 0) has one search point module which can process a search point within one cycle so that the level 0 with search range [−8, +7] can finish the full search within 256 cycles. In the same manner, levels 1 and 2 have four and 16 search point modules, respectively, which means that levels 1 and 2 can process four and 16 search points in parallel. Therefore, levels 1 and 2 can process 1024 and 4096 search points, respectively, within 256 cycles by the parallelization techniques.

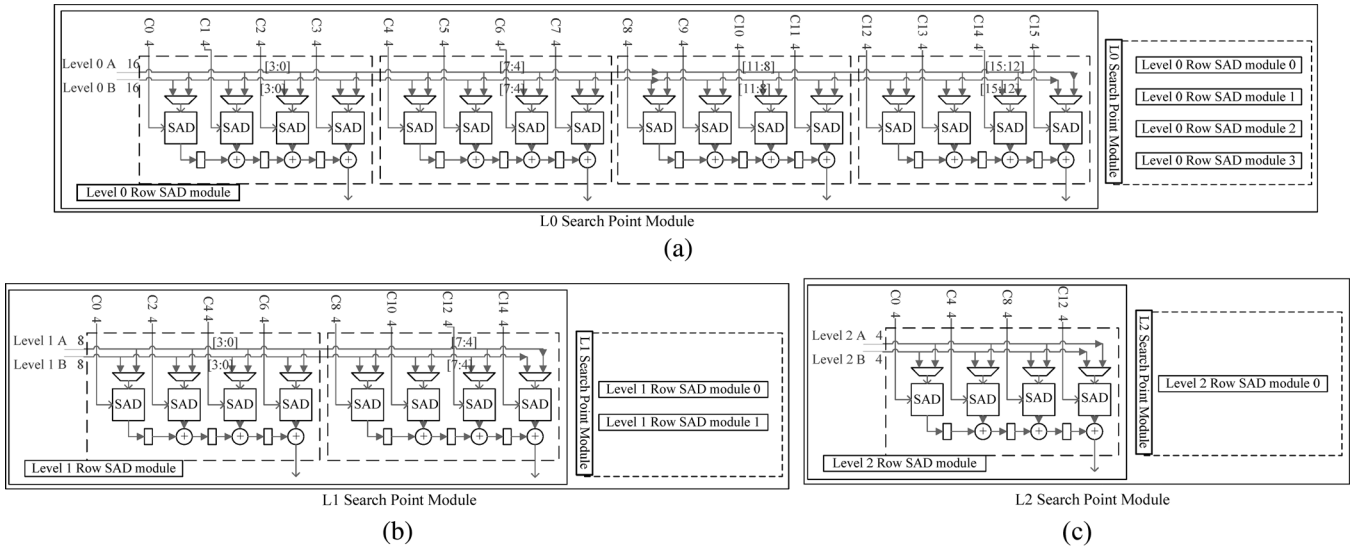

Fig. 9. SAD calculation unit used for different levels. The modules can process a search point of a 16 × 16 MB within one cycle. (a) L0 (level 0) search point module. (b) L1 (level 1) search point module. (c) L2 (level 2) search point module.

Fig. 9 shows the detailed architecture of each level. Fig. 9(a) shows the "L0 search point module," which consists of four row SAD modules. Each row SAD module contains 16 4p-SAD units. Thus, the L0 search point module includes 64 4p-SAD units in total to generate the total SAD cost of a 16 × 16 MB. As for level 1 with 2:1 subsampling, the number of search point is 1024. Furthermore, since the current buffer for level 1 is also subsampled, only 64 pixels are compared in current MB. Therefore, L1 (Level 1) search point modules in Fig. 9(b) only needs 16 4p-SAD units, which is quarter of that in level 0. In order to keep the cycle count of level 1 as the same as that of level 0, we use four L1 search point modules. Thus, four search points in level 1 can be processed in parallel with the same current block. With above arrangement, the total hardware cost of level 1 is the same as that of level 0, 64 4p-SAD units. Similar design considerations are also applied to level 2. Thus, in level 2, the L2 (Level 2) search point module in Fig. 9(c) only needs 4 4p-SAD units so that we use 16 L2 search point modules to compute 16 search points in parallel. In summary, all these levels have 64 4p-SAD units respectively to balance the computation cycle of each level to be the same 256 cycles.

The SADs generated from the SAD modules are further summed up by the summation trees to generate the SAD of different block size.

C. Architecture for SIFME

Fig. 10 shows the proposed FME hardware architecture. The input data are first interpolated by the interpolation unit for half and quarter pixels of one 4 × 4 block. Then, these data are computed with the current block data by six 4 × 4 block PUs. Each PU is in charge of residual generation and 4 × 4 Hadamard transform. All larger sized blocks are decomposed into a 4 × 4 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 the SB_buffer.

Table VI shows the comparisons for hardware implementation. The proposed algorithm searches only six candidates and

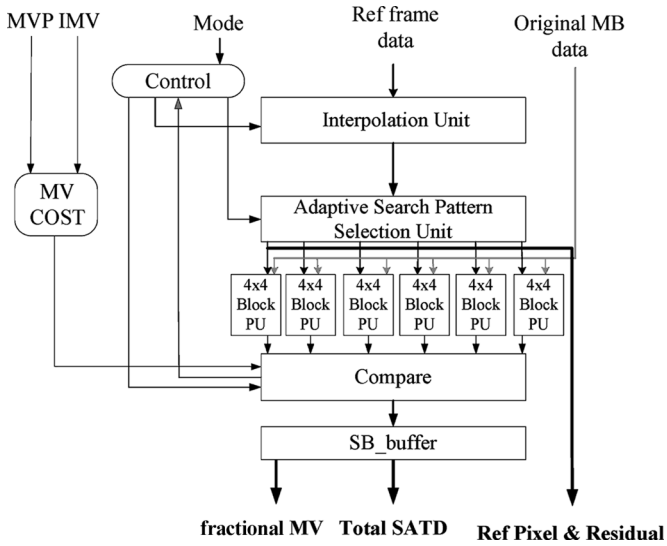

Fig. 10. Proposed hardware architecture of FME.

TABLE VI
COMPARISON OF NUMBER OF PUS AND NUMBER OF ITERATIVE SEARCH STEPS

	# of PU	# of iterative search step
[26]	5	2
[21]	9	2
proposed	6	1

needs only six PUs. In addition, with one loop design, our design just takes approximately half the number of cycles compared with the other designs [21], [26].

V. IMPLEMENTATION RESULT AND COMPARISONS

The proposed design has been implemented by Verilog and synthesized by the 0.13-μm CMOS process. Table VII shows the total hardware cost of our IME design and comparison to other designs. Our design can provide the largest search range

TABLE VII
COMPARISON OF THE IME PART WITH PREVIOUS DESIGNS

	[8]	[2]	[6]	[7]	[19]	[9]	[5]	Ours
Max. Supporting Resolution	CIF@30fps	4CIF@15fps	4CIF@15fps	720x480@30fps	720p@30fps	720p@30fps	720p@60fps	1080p@60fps
Search Algorithm	4-Step	Full	Full	Multi-resolution	Full	Subsampling	Full	Multi-resolution
Quality loss (dB)	About 0.1	0	0	0.4	0	0.083	0	0.065
PE (SAD Module)	256	16	256	64/320	1024	32	256	192
Max. Search Range	H:±32 V: ±16	H:±32 V: ±32	H:±64 V: ±64	H:±64 V: ±64	H:±64 V: ±32	H:±32 V: ±32	H:±16 V: ±16	H: ±128 V: ±128
Gate Count (K)	131.2	61	154	n.a	330.2	47.9+4k bit buffer	176	155.8 for 720p 213.7 for 1080p
Memory (Kbyte)	8	n.a.	7.5	n.a	26	2.75	41.6	5.19 for 720p 5.95 for 1080p
Operating Freq. (MHz)	40 (13.3 for CIF)	294 for 4CIF	100 for 4CIF	16 for 720x480	n.a	105 for 720p	55.6 for 720p	27.6 for 720p 124.4for 1080p
Latency (Cycle)	n.a.	4096	1024	375	n.a	972	258	256
CMOS Tech.	0.18μm	0.13μm	0.18μm	n.a	0.18μm	0.18μm	0.18μm	0.13μm

TABLE VIII
COMPARISON OF THE FME PART WITH PREVIOUS DESIGNS

	[21]	[26]	[27]	[30]	[28]	Ours
Max. Supporting Resolution	720x576@30fps	720p@30fps	720p@30fps	1080p@30fps	3200x2400@30fps	1080p@60fps
Algorithm	17 candidates 2-iteration interpolation	8 candidates 2-iteration interpolation	25 candidates 1-iteration interpolation	17 candidates 2-iteration interpolation	Math-model No interpolation	6 candidates 1-iteration interpolation
Gate Count (K)	79.3	48	117.2	188.45	56.53	52.8 for 720p 68.9 for 1080p
Latency (Cycle)	1648	2000	1000	790	110	264(Best) 432(Worst)
Operating Freq. (MHz)	100	100	108 for 720p	285	100	28.5 for 720p 128.3 for 1080p
Throughput (Kilo-MBs/sec)	49	50	108	250	909	486(Best)
Quality Drop (dB)	0.1	0.09	0.012	n.a.	0.15	0.04
CMOS Tech.	0.18μm	0.18μm	0.13μm	0.18μm	0.13μm	0.13μm

capability (High Profile Level 2) but just needs similar hardware cost and smaller buffers. In addition, our design has the shortest latency so that our design can achieve 1080-p@60-fps specification with only 124-MHz operating frequency only. In comparison, the designs in [5], [6], and [19] have larger area cost and long latency due to the full-search architecture. Though designs in [7] and [8] use fast algorithms to reduce the latency, they still needs large area cost and buffer. As for [9], their throughput is only one-fourth of ours though it uses fast algorithm. The proposed IME design can achieve low latency with low buffer cost and similar area cost and, thus, is suitable for HD applications.

Table VIII shows the hardware comparison of FME designs. Our design can achieve high throughput with low area cost and negligible quality loss. Its hardware cost is only slightly larger than our previous work [26], but with six times throughput. As for [27], the area cost is doubled compared with our work because it needs to search 25 points in an iteration. In comparison, though the design in [28] has higher throughput than ours, their quality drop is more than 0.15 dB due to imprecise mathematical modeling. In addition, such mathematical modeling design still needs additional hardware to calculate the final residuals, which is not included in the gate count report. When comparing with other designs for 1080 p @ 30 fps [30], our FME design only needs 36.5% of gate count and 54.6% of cycles due to our hardware-based approaches.

Table IX shows the total hardware cost of our ME design and comparison to the integrated designs [11], [18]. Comparing with [18], our design can save at least 30% of area costs and 50% of memory costs in the IME part. As for the FME part, we save 82.8% of area cost due to fewer PUs and reduce memory by 81.2%. In summary, the total area and memory saving is 60% and 68.9%, respectively. As for the throughput, our design is sufficient for HD video applications. Our design improves throughput by 75% when comparing with that in [18]. If comparing with the other integrated design [11] using fast algorithms in IME, our design still saves 12.3% of area. As for the cycle count, our design also has 75.5% of throughput improvement than [11]. Due to the high throughput, only 28.5 MHz is

TABLE IX
HARDWARE COST COMPARISON FOR COMPLETE H.264 ME ACCELERATOR WITH PREVIOUS WORKS

	[18]	[11]	Ours	Saving with [18]	Saving with [11]
Max Supporting Resolution	720p@30fps	720p@30fps	1080p@60fps		
Search Range	H: ±64 V: ±32	H: ±96 V: ±96	H: ±128 V: ±128		
Quality Loss (dB)	0	n.a.	0.1		
IME Gate Count (K)	305.2	n.a	155.8 for 720p 213.7 for 1080p	48.9% for 720p 30% for 1080p	
FME Gate Count (K)	401.8	n.a	52.8 for 720p 68.9 for 1080p	86.8% for 720p 82.8% for 1080p	
Total Gate Count (K)	707	238	208.6 for 720p 282.6 for 1080p	70.4% for 720p 60% for 1080p	12.3% for 720p
IME Memory (Kbyte)	13.71	n.a	5.19 for 720p 5.95 for 1080p	62.1% for 720p 56.6% for 720p	n.a
FME Memory (Kbyte)	13.82	n.a	2.59	81.2%	n.a
Total Memory (Kbyte)	27.53	n.a	7.78 for 720p 8.54 for 1080p	71.7% for 720p 68.9% for 1080p	n.a
Latency for IME Stage (Cycle)	1024	1079	256	75%	75.5%(Best)
Latency for FME Stage (Cycle)	1648		264(Best) 432(Worst)	83.9%(Best)	
Freq. (MHz)	120 (108 for 720p)	117 for 720p	28.5 for 720p 128.8 for 1080p	73.6% for 720p	75.6% for 720p
CMOS Tech.	0.18μm	0.18μm	0.13μm		

enough for a 720-p sequence with 30 frames per second and 128.8 MHz for 1080-p sequences with 60 frames per second. To satisfy such a high throughput requirement, the external bus bandwidth should be set to 128 bits.

VI. CONCLUSION

In this paper, we propose a highly data-reused ME design with low cost and latency for HD video. This design maximizes the most concerned data reuse by sharing data within IME as well as between IME and FME, while minimizing the computation and latency by parallel multiresolution IME and single-iteration FME. The final design can easily support processing for 1080-p sequences with just 128.8 MHz and 282.6 K gates and saves 60% of gate count and 68.9% of SRAM buffers compared with the previous design. The presented design also can be easily scaled to other smaller size video with search range adjustment.

ACKNOWLEDGMENT

The authors would like to thank the National Chip Implementation Center (CIC) for providing cell-based design tools and cell library.

REFERENCES

[1] ITU-T Recommendation and International Standard of Joint Video Specification. ITU-T Rec. H.264/ ISO/ IEC 14496-10 AVC, Mar. 2005.

[2] S. Y. Yap and J. V. McCanny, "A VLSI architecture for variable block size video motion estimation," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 51, no. 7, pp. 384–389, Jul. 2004.

[3] C. M. Ou, C. F. Le, and W. J. Hwang, "An efficient VLSI architecture for H.264 variable block size motion estimation," *IEEE Trans. Consum. Electron.*, vol. 51, no. 4, pp. 1291–1299, Nov. 2005.

[4] C. Wei and M. Z. Gang, "A novel VLSI architecture for VBSME in MPEG-4 AVC/H.264," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2005, vol. 2, pp. 1794–1797.

[5] Z. Zheng, H. Sang, W. Huang, and X. Shen, "High data reuse VLSI architecture for H.264 motion estimation," in *Proc. Int. Conf. Commun. Technol.*, Nov. 2006, pp. 1–4.

[6] M. Kim, I. Hwang, and S. I. Chae, "A fast vlsi architecture for full search variable block size motion estimation in MPEG-4 AVC/H.264," in *Proc. Asia South Pacific Design Autom. Conf.*, Jan. 2005, vol. 1, pp. 631–634.

[7] J. H. Lee and N. S. Lee, "Variable block size motion estimation algorithm and its hardware architecture for H.264/AVC," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2004, vol. 3, pp. 741–744.

[8] T. C. Chen, Y. H. Chen, S. F. Tsai, S. Y. Chien, and L. G. Chen, "Fast algorithm and architecture design of low-power integer motion estimation for H.264/AVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 5, pp. 568–577, May 2007.

[9] C. L. Su, W. S. Wang, Y. L. Chen, Y. C. Wang, C. W. Chen, J. I. Guo, and S. Y. Tseng, "A low complexity high quality interger motion estimation architecture design for H.264/AVC," in *Proc IEEE Asia Pacific Conf. Circuits Syst.*, Dec. 2006, pp. 398–401.

[10] Y. L. Xi, C. Y. Hao, Y. Y. Fan, and H. Q. Hu, "A fast block-matching algorithm based on adaptive search area and its VLSI architecture for H.264/AVC," *Signal Process.: Image Commun.*, vol. 21, no. 8, pp. 626–646, Sep. 2006.

[11] L. Zhang and W. Gao, "Reusable architecture and complexity-controllable algorithm for the integer/fractional motion estimation of H.264," *IEEE Trans. Consum. Electron.*, vol. 53, no. 2, pp. 749–756, May 2007.

[12] T. H. Tsai and Y. N. Pan, "A novel 3-D predict hexagon search algorithm for fast block motion estimation on H.264 video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 12, pp. 1542–1547, Dec. 2006.

[13] C. H. Kuo, M. Shen, and C. C. J. Kuo, "Fast motion search with efficient inter-prediction mode decision for H.264," *J. Visual Comm. Image Represent.*, vol. 17, no. 2, pp. 217–242, Apr. 2006.

[14] N. A. Khan, S. Masud, and A. A. Ahmad, "Variable block size motion estimation algorithm for real-time H.264 video encoding," *Signal Process.: Image Commun.*, vol. 21, no. 4, pp. 306–315, Apr. 2006.

[15] Y. K. Tu, J. F. Yang, M. T. Sun, and Y. T. Tsai, "Fast variable-size block motion estimation for efficient H.264/AVC encoding," *Signal Process.: Image Commun.*, vol. 20, no. 7, pp. 595–623, Aug. 2005.

[16] Z. Chen, J. Xu, Y. He, and J. Zheng, "Fast integer-pel and fractional-pel motion estimation for H.264/AVC," *J. Visual Commun. Image Represent.*, vol. 17, no. 2, pp. 264–290, Apr. 2006.

[17] Z. Zhou, J. Xin, and M. T. Sun, "Fast motion estimation and inter-mode decision for H.264/MPEG-4 AVC encoding," *J. Visual Commun. Image Represent.*, vol. 17, no. 2, pp. 243–263, Apr. 2006.

[18] T. C. Chen *et al.*, "Analysis and architecture design of an HDTV720p 30 frames/s H.264/AVC encoder," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 6, pp. 673–688, Jun. 2006.
[19] C. Y. Chen *et al.*, "Analysis and architecture design of variable block size motion estimation for H.264/AVC," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 53, no. 2, pp. 578–593, Feb. 2006.
[20] Joint Video Team Reference Software. ver. JM9.0, ITU-T.
[21] T. C. Chen, Y. W. Huang, and L. G. Chen, "Fully utilized and reusable architecture for fractional motion estimation of H.264/AVC," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, May 2004, vol. 4, pp. 9–12.
[22] L. Yang, K. Yu, J. Li, and S. Li, "Prediction-based directional fractional pixel motion estimation for H.264 video coding," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, May 2005, vol. 2, pp. 901–904.
[23] J. F. Chang and J. J. Leou, "A quadratic prediction based fractional-pixel motion estimation algorithm for H.264," *Proc. IEEE Int. Symp. Multimedia*, pp. 491–498, Dec. 2005.
[24] H. Chao and J. Lu, "A high accurate predictor based fractional pixel search for H.264," in *Proc. IEEE Int. Conf. Image Process.*, Sep. 2006, pp. 2365–2368.
[25] L. Shen, Z. Zhang, Z. Liu, and W. Zhang, "An adaptive and fast fractional pixel search algorithm in H.264," *Signal Process.*, vol. 87, no. 11, pp. 2629–2639, Nov. 2007.
[26] Y. J. Wang, C.-C. Cheng, and T.-S. Chang, "A fast fractional-pel motion estimation algorithm for H.264/AVC," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2006, pp. 3974–3977.
[27] C. L. Su, W. S. Yang, Y. Li, C. W. Chen, J. I. Guo, and S. Y. Tseng, "Low complexity high quality fractional motion estimation algorithm and architecture design for H.264/AVC," in *Proc IEEE Asia Pacific Conf. Circuits Syst.*, Dec. 2006, pp. 578–581.
[28] C. Y. Kao, H. C. Kuo, and Y. L. Lin, "High performance fractional motion estimation and mode decision for H.264/AVC," in *Proc. IEEE Int. Conf. Multimedia Expo*, Jul. 2006, pp. 1241–1244.
[29] J. W. Suh and J. Jeong, "Fast sub-pixel motion estimation techniques having lower computation complexity," *IEEE Trans. Consum. Electron.*, vol. 50, no. 4, pp. 968–973, Aug. 2004.
[30] C. Yang, S. Goto, and T. Ikenaga, "High performance VLSI architecture of fractional motion estimation in H.264 for HDTV," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2006, pp. 2605–2608.
[31] J. C. Tuan, T. S. Chang, and C. W. Jen, "On the data reuse and memory bandwidth analysis for full-search block-matching VLSI architecture," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 1, pp. 61–72, Jan. 2002.
[32] C. C. Lin, Y. K. Lin, and T. S. Chang, "PMRME: A parallel multi-resolution motion estimation algorithm and architecture for HDTV sized H.264 video coding," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Apr. 2007, vol. 2, pp. 285–288.
[33] T. Y. Kuo, Y. K. Lin, and T. S. Chang, "SIFME: A single iteration fractional-pel motion estimation algorithm and architecture for HDTV sized H.264 video coding," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Apr. 2007, vol. 1, pp. 1185–1188.

Yu-Kun Lin was born in Kaohsiung, Taiwan, R.O.C., in 1979. He received the B.S and M.S degrees in electrical engineering from National Taiwan University, Taipei, Taiwan, in 2000 and 2002, respectively. He is currently working toward the Ph.D. degree at the Institute of Electronics, National Chiao-Tung University, Hsinchu, Taiwan.

His research interests include video and image processing, digital signal processing, and VLSI architecture design.

Chia-Chun Lin received the B.S. and M.S. degrees in electronics engineering from National Chiao-Tung University, Hsinchu, Taiwan, R.O.C., in 2005 and 2007, respectively.

He is currently serving in the Army of R.O.C. as a Second Lieutenant. His major research interests include digital signal processing, multimedia video systems, and computer architecture.

Tzu-Yun Kuo received the B.S. and M.S. degrees in electronics engineering from National Chiao-Tung University, Hsinchu, Taiwan, R.O.C., in 2005 and 2007, respectively.

After graduation, he joined Alpha Imaging Technology Corporation, Inc., Jhubei City, Hsinchu County. His major research interests include H.264/AVC video coding, digital signal processing, and associated VLSI architecture design.

Tian-Sheuan Chang (S'93–M'06–SM'07) 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 Associate Professor. During 2000 to 2004, he was with Global Unichip Corporation, Hsinchu. His research interests include IP and SOC design, VLSI signal processing, and computer architecture.