270 MHz Full HD H.264/AVC High Profile Encoder with Shared Multibank Memory-Based Fast Motion Estimation

Sukho Lee, Seongmo Park, and Jongwon Park

We present a full HD (1080p) H.264/AVC High Profile hardware encoder based on fast motion estimation (ME). Most processing cycles are occupied with ME and use external memory access to fetch samples, which degrades the performance of the encoder. A novel approach to fast ME which uses shared multibank memory can solve these problems. The proposed pixel subsampling ME algorithm is suitable for fast motion vector searches for high-quality resolution images. The proposed algorithm achieves an 87.5% reduction of computational complexity compared with the full search algorithm in the JM reference software, while sustaining the video quality without any conspicuous PSNR loss. The usage amount of shared multibank memory between the coarse ME and fine ME blocks is 93.6%, which saves external memory access cycles and speeds up ME. It is feasible to perform the algorithm at a 270 MHz clock speed for 30 frame/s real-time full HD encoding. Its total gate count is 872k, and internal SRAM size is 41.8 kB.

Keywords: H.264/AVC, High Profile, encoder, fast motion estimation, shared multibank memory, rate distortion (RD).

Manuscript received Apr. 21, 2009; revised June 29, 2009; accepted July 14, 2009.

This work was supported by the IT R&D program of MKE/IITA, Rep. of Korea (2007-S-026-02, Multi-format multimedia SoC based on MPCore Platform).

Sukho Lee (phone: +82 42 860 6171, email: shlee99@etri.re.kr) and Seongmo Park (email: smpark@etri.re.kr) are with the Convergence Components & Materials Research Laboratory, ETRI, Daejeon, Rep. of Korea.

Jongwon Park (email: jwpark@cnu.ac.kr) is with the Department of Information Communications Engineering, Chungnam National University, Daejeon, Rep. of Korea.

doi:10.4218/etrij.09.1209.0007

I. Introduction

H.264/AVC High Profile [1] is known to achieve a significant improvement in rate-distortion efficiency compared with existing standards and is designed for broadcast TV over cable/DSL/satellite, IP set-tops, and HD-DVD/Bluray-DVD recorders [2]. There are essential coding tools for High Profile H.264/AVC encoding: inter-prediction, intra-prediction, deblocking, and entropy coding tools. Among them, motion estimation (ME) for inter-prediction has high computational complexity because it thoroughly searches the motion vector (MV) for all possible candidate macroblocks (MBs). Therefore, it generally chooses the performance of the encoder, particularly when implemented in hardware. It is also more complex in cases of high resolution video such as full HD.

1. Motivation of This Work

Recently, a variety of algorithms have been employed to reduce the computational complexity of inter-prediction or ME. A fast inter-mode decision algorithm [3] uses a thresholding method and early stop of the inter-mode determination. Another fast MB mode decision algorithm is described in [4], while [5] presents fast inter-mode selection using a hierarchical decision process. Their common approach quickly decides the inter-mode using termination algorithms based on temporal correlation through statistical analysis. The fast ME proposed in [6] also adopts a similar early termination strategy according to the threshold value of minimum cost (MCOST). The complex optimizing ME in [7] demonstrates the complexity control by MB partitions with variable block sizes, and fast full-pel ME for variable block sizes is described in [8]. Also, computational complexity management for real-time

H.264/AVC encoding is presented in [9] and [10]. A novel processing element (PE) design for ME architecture is introduced in [11]. However, because most of the previous works focus on fast inter-mode decisions and size reduction, they have potential drawbacks in that they do not always conduct accurate ME compared with the full search algorithm and are not suitable for various images because they are based on statistical data. Besides, they suffer from quality loss in some cases. To solve these problems, we have contrived a novel approach for full-search-based fast ME and its hardware implementation. The proposed pixel subsampling ME algorithm is suitable for ME for high-quality resolutions and sustains video quality without noticeable PSNR loss. Also, it reduces computational complexity more than prior works. The designed ME block has internal shared multibank memory between coarse ME (MEC) and fine ME (MEF) subblocks, which reduces the number of data fetch cycles from the external memory.

2. Brief Introduction of ME

ME is carried out in a video encoder and has a significant effect on encoder performance. A good prediction reference choice minimizes the energy in the motion-compensated residual, which in turn maximizes the compression performance. ME, therefore, aims to find a match to the current block or region that minimizes the energy in the motion compensated residual (the difference between the current block and the reference area) [12]. For an $M \times N$ block, the sum of the absolute difference (SAD) to estimate the MV is calculated by

$$SAD(c, r(m)) = \sum_{x=1}^{M} \sum_{y=1}^{N} | c(x, y) - r(x - m_x, y - m_y) | , \quad (1)$$

where c is the original current MB, and r is the predicted reference MB at the position designated by candidate MV m in the reference picture considered. Finally, a predicted MV that has a minimum SAD is expressed as

$$MV(m_x, m_y) = \min(SAD(c, r(m)). \quad (2)$$

Full search ME, which calculates the SAD and searches the MV for all candidate positions from the top-left of the window (position [$-X$, $-Y$]) to the bottom-right (position [X, Y]) through a raster search order, is computationally intensive and not applicable to power-limited and real-time applications [12]. Thus, the choice of its algorithm depends on the platform and on whether the ME is block-based or region-based.

Section II analyzes the experimental results of exploring optimal parameters to implement the proposed H.264/AVC High Profile encoder and describes our proposed searching algorithm in detail. The entire proposed pipeline architecture and ME structure for inter-prediction are presented in section III. Section IV shows the implementation results. A performance comparison of rate distortion (RD) is described in section V. Finally, conclusions are drawn in section VI.

II. Proposed Algorithm and Complexity Management

We have thoroughly experimented on various conditions using the JM13.2 reference software [13] in order to seek optimal parameters for the hardware implementation of the High Profile H.264/AVC encoder. On the basis of these results, the presented hardware encoder was designed. Because most processing cycles are occupied in ME for inter-prediction, we have thought out the pixel subsampling ME algorithm and its architecture to mitigate the computational complexity of hardware.

1. Block-Based ME

Full search ME involves calculating SAD in (1) at each point in the search window, that is, $\pm S$ samples around the current MB position (0, 0). It is guaranteed to find the minimum SAD in the search window, but it is computationally intensive since the energy measure must be calculated at every $(2S+1)^2$ location.

In full search ME, the first search location is at the top-left of the window (position [$-S$, $-S$]), and the search proceeds in raster order until all positions have been evaluated. Many fast search algorithms have been proposed, such as logarithmic search and hierarchical search. In each case, the performance of the algorithm can be evaluated by a comparison with full search [12].

2. Prior Three-Step Search vs. Full Search Algorithm

In the three-step search presented in our prior work [14], ME is carried out by skipping two pixel (2-pel) units so that there is occasionally a possibility of searching an inaccurate MV in the horizontal and vertical directions (X/Y-direction) if an image includes a higher frequency component. For this reason, it is necessary to search an accurate MV using fewer steps. However, because this kind of small step ME increases hardware cost, various methods are considered: a search method using a 1-pel unit search in the X direction and a 2-pel unit search in the Y direction, or using 1-pel in both the X and Y directions. In the latter case, the window size of Y is half that of X because movement in the Y direction is generally small in most video images, whereas movement in the X direction is large in the human visual system.

We first compare the existing hierarchical search algorithm with the full search algorithm by various anchor values using

JM13.2 reference software. To judge the algorithm itself, the test conditions and remaining default settings in JM13.2 are used. Table 1 shows the main test conditions. An iterative experiment using various test conditions is performed for CIF to 1080p images. Tables 2 and 3 demonstrate the results of the performance comparison for two search modes.

We have deliberated on SearchRange as well as NumberReferenceFrames and NumberBFrames because its parameters particularly affect PNSR loss for high quality sequences. As shown in Table 2, average ΔY-PSNR loss of SearchRange 16 compared to 128 is –0.14 dB on JM13.2 full search condition. In Table 3, ΔY-PSNR between NumberReferenceFrames 1 and 3 is –0.20 dB on JM13.2 full search condition. If the encoder supports three reference frames as shown Table 3, the internal SRAM is increased by three times (9 kB), and ME processing cycles also increase by three times (1,536 cycle/MB) compared to ME with one reference frame in case of B frame. It needs a 519 MHz clock frequency (an additional 1,024 cycle/MB is added to a 1,100 cycle/MB in the proposed design) for 30 frame/s full HD encoding. Actually, because it is true that ME with multiple reference frames increases coding efficiency even if it needs more processing cycles and higher clock frequency, the proposed design can be configured to support multiple reference frames by appending an extra SRAM and merely adding a multiple reference ME stage to the top scheduler. Additionally, ΔY-PSNR between B frames number one and three is −0.04 dB. However, because the proposed High Profile H.264 encoder is a dedicated hardware for real-time encoding, ME with a wide search range and multiple reference frames is too computationally intensive for many practical applications such as computation or power-limited applications. Therefore, we have prudently chosen a search range and reference frame numbers and considered the optimized parameters for hardware implementation because there is not a serious PNSR loss. Optimal parameters for hardware complexity are adopted on the basis on these test results except for SearchMode. This is because the full search method consumes a great deal of processing cycle, which is not appropriate for real-time applications, and the prior three-step search method incurs more quality loss.

Table 1. Main test conditions in JM13.2 reference software.

Parameters	Anchor value
DisableSubpelME	0 (quarter-pel)
SearchRange	16
NumberReferenceFrames	1
InterSearchSize	InterSearch16×16
NumberBFrames	1
DirectModeType	1
RDOptimization	0
SearchMode	0 (fast full search)
RateControlEnable	1

3. Proposed Pixel Subsampling ME Algorithm and Complex Management

As shown in Tables 2 and 3, the prior three-step search [14] deteriorates ΔPSNR by more than 1.0 dB in the CITY and MOBCAL images, which have a greater number of high-frequency and complex components. To resolve this problem, we contrived four candidate methods for ME and finally selected one novel approach considering the hardware complexity.

Table 4 shows the four proposed search methods. Actually, the search range of method 1 is –16 to +15 in the horizontal direction and –8 to +7 in the vertical direction. Its search steps are (1-pel, 1-pel) in the x and y directions, respectively, and the subsampling ratio is (1, 2) as expressed in Table 4.

For an (x, y) position of a pixel in an $M \times N$ block in the search window, let x' be the subsampled position designated by candidate MV m_x toward the x direction, and let y' be the subsampled position designated by the candidate MV m_y toward the y direction. Then,

$$x' = \begin{cases} 2x-1, & m_x \pm \text{odd position} \\ 2x, & m_x \pm \text{even position,} \end{cases}$$

$$y' = \begin{cases} 2y-1, & m_y \pm \text{odd position} \\ 2y, & m_y \pm \text{even position.} \end{cases}$$

The SADs for the four kinds of methods are calculated as follows.

For method 1,

$$SAD(c, r(m)) = \sum_{x=1}^{M} \sum_{y=1}^{N/2} | c(x, 2y-1) - r(x - m_x, y' - m_y) | . \quad (3)$$

For method 2,

$$SAD(c, r(m)) = \sum_{x=1}^{M/2} \sum_{y=1}^{N/2} c(2x-1, 2y-1) - r(x' - m_x y' - m_y). \quad (4)$$

For method 3,

$$SAD(c, r(m)) = \sum_{x=1}^{M/2} \sum_{y=1}^{N/2} | c(2x-1, 2y-1) - r(x' - m_x, (2y-1) - m_y) | . \quad (5)$$

Table 2. Performance comparison of SearchRange on a JM13.2 reference encoder for IBPBP sequences.

Condition	Sequences	Full search with SR 16 in JM13.2					Full search with SR 128 in JM13.2					PSNR loss
Search Range	Contents	Bitrate (kbps)	PSNR (Y)	PSNR (U)	PSNR (V)	Encoding time (s)	Bitrate (kbps)	PSNR (Y)	PSNR (U)	PSNR (V)	Encoding time (s)	ΔY-PSNR (dB)
128	CITY (4CIF)	1048.10	32.71	41.46	43.90	177.42	1045.81	32.72	41.40	43.73	2879.48	-0.01
	CREW (4CIF)	1077.98	34.82	40.59	40.32	179.05	1071.60	34.39	39.89	39.55	2684.17	0.43
	HARBOUR (4CIF)	1259.63	29.86	40.12	41.72	177.36	1054.61	29.33	40.00	41.62	2735.07	0.53
	ICE (4CIF)	1033.36	38.93	45.67	46.31	176.86	1034.74	39.47	45.74	46.19	2418.96	-0.54
	SOCCOR (D1)	1093.63	31.26	40.79	42.63	176.37	1046.20	32.13	41.26	43.01	3245.25	-0.87
	MOBCAL (720p)	2110.75	32.17	35.67	39.07	393.69	2100.44	32.14	35.63	39.04	6299.73	0.03
	PEDESTRIAN (1080p)	5226.43	38.76	43.18	44.52	897.21	5229.07	38.96	42.89	44.07	11592.08	-0.20
	BLUE_SKY (1080p)	5219.81	40.04	40.01	41.38	288.61	5239.58	40.10	40.05	41.43	10643.22	-0.06
	RIVERBED (1080p)	6373.62	29.41	37.66	40.16	393.94	6023.97	30.32	37.63	40.07	19790.34	-0.91
	RUSH_HOUR (1080p)	6373.62	29.41	37.66	40.16	393.94	5239.58	40.10	40.05	41.43	10643.22	-0.07
	STATION2 (1080p)	5253.38	40.62	44.60	44.55	302.88	5249.43	40.63	44.58	44.54	13489.31	-0.01
	SUNFLOWER (1080p)	5192.30	42.55	43.89	44.43	272.47	5189.37	42.63	43.97	44.48	9978.29	-0.08
	Average ΔY-PSNR loss											-0.14
Condition	Sequences	Three step search					Full search in JM13.2					PSNR loss
Search Range	Contents	Bitrate (kbps)	PSNR (Y)	PSNR (U)	PSNR (V)	Encoding time (s)	Bitrate (kbps)	PSNR (Y)	PSNR (U)	PSNR (V)	Encoding time (s)	ΔY-PSNR (dB)
64	CITY (4CIF)	1063.44	31.40	41.02	43.39	157.67	1047.17	32.67	41.45	43.86	1950.51	-1.27
	CREW (4CIF)	1084.18	34.58	40.46	40.11	155.17	1080.19	34.80	40.52	40.25	1951.03	-0.22
	HARBOUR (4CIF)	1550.24	29.78	40.17	41.86	156.05	1267.38	29.87	40.17	41.64	1950.75	-0.09
	ICE (4CIF)	1033.22	39.06	45.80	46.36	145.31	1033.39	39.34	45.94	46.65	1951.72	-0.28
	SOCCOR (D1)	1068.94	31.39	41.11	43.05	153.10	1063.36	32.03	41.31	43.24	1950.14	-0.64
	MOBCAL (720p)	2172.98	31.00	35.27	38.66	354.77	2108.61	32.11	35.65	39.06	4454.91	-1.11
	PEDESTRIAN (1080p)	5213.73	39.01	43.35	44.78	780.95	5218.96	39.15	43.46	44.89	10111.78	-0.14
	Average ΔY-PSNR loss											-0.54
16	CITY (4CIF)	1054.00	31.64	41.09	43.51	56.19	1048.10	32.71	41.46	43.90	177.42	-1.07
	CREW (4CIF)	1079.65	34.68	40.51	40.21	56.23	1077.98	34.82	40.59	40.32	179.05	-0.14
	HARBOUR (4CIF)	1446.95	29.81	40.21	41.84	58.02	1259.63	29.86	40.12	41.72	177.36	-0.05
	ICE (4CIF)	1033.62	38.74	45.50	46.19	57.07	1033.36	38.93	45.67	46.31	176.86	-0.19
	SOCCOR (D1)	1129.12	30.72	40.60	42.46	57.31	1093.63	31.26	40.79	42.63	176.37	-0.54
	MOBCAL (720p)	2149.74	31.20	35.33	38.72	131.18	2110.75	32.17	35.67	39.07	393.69	-0.97
	PEDESTRIAN (1080p)	5226.25	38.64	43.10	44.46	300.22	5226.43	38.76	43.18	44.52	897.21	-0.12
	Average ΔY-PSNR loss											-0.44

For method 4,

$$SAD(c,r(m))=\sum_{x=1}^{M}\sum_{y=1}^{N/2}|c(x,2y-1)-r(x'-m_x,(2y-1)-m_y)|. \quad (6)$$

The subsampling position of pixels for SAD calculation is dependent on the position of candidate MV (m_x, m_y) as shown Fig. 1.

The four search methods were tested using JM13.2 software, and the performance of each method was compared with the full search algorithm in JM. Table 5 shows the results. Method 1 has the least average ΔPSNR loss compared with a full search, and it comes closest to the full search algorithm. In particular, it achieves an outstanding result compared with that of the prior hierarchical search, which is degraded by more than 1 dB in such images as the CITY sequence as shown Tables 2 and 3.

However, since this method requires more processing cycles than methods 2 and 3, method 2 is the most appropriate for hardware implementation, though it incurs a little more PSNR

Table 3. Performance comparison of NumberReferenceFrames on a JM13.2 reference encoder for IBPBP sequences.

Condition	Sequences	Three-step search					Full search in JM13.2					PSNR loss
Number Reference Frames	Sequence	Bitrate (kbps)	PSNR (Y)	PSNR (U)	PSNR (V)	Encoding time (s)	Bitrate (kbps)	PSNR (Y)	PSNR (U)	PSNR (V)	Encoding time (s)	ΔY-PSNR (dB)
1	CITY (4CIF)	1054.00	31.64	41.09	43.51	56.19	1048.10	32.71	41.46	43.90	177.42	-1.07
	CREW (4CIF)	1079.65	34.68	40.51	40.21	56.23	1077.98	34.82	40.59	40.32	179.05	-0.14
	HARBOUR (4CIF)	1446.95	29.81	40.21	41.84	58.02	1259.63	29.86	40.12	41.72	177.36	-0.05
	ICE (4CIF)	1033.62	38.74	45.50	46.19	57.07	1033.36	38.93	45.67	46.31	176.86	-0.19
	SOCCOR (D1)	1129.12	30.72	40.60	42.46	57.31	1093.63	31.26	40.79	42.63	176.37	-0.54
	MOBCAL (720p)	2149.74	31.20	35.33	38.72	131.18	2110.75	32.17	35.67	39.07	393.69	-0.97
	PEDESTRIAN (1080p)	5226.25	38.64	43.10	44.46	300.22	5226.43	38.76	43.18	44.52	897.21	-0.12
	Average ΔY-PSNR loss											-0.44
3	CITY (4CIF)	1050.33	32.38	41.37	43.74	79.81	1047.52	32.95	41.55	43.95	352.89	-0.57
	CREW (4CIF)	1078.02	35.02	40.70	40.44	80.06	1079.53	35.12	40.78	40.52	352.83	-0.10
	HARBOUR (4CIF)	1310.01	29.91	40.17	41.88	79.51	1224.31	29.94	40.12	41.76	353.61	-0.03
	ICE (4CIF)	1032.69	39.02	45.65	46.41	78.97	1032.96	39.25	45.80	46.53	363.47	-0.23
	SOCCOR (D1)	1098.64	31.09	40.78	42.63	79.37	1088.32	31.49	40.92	42.77	347.75	-0.40
	MOBCAL (720p)	2107.86	32.45	35.72	39.19	176.57	2092.78	32.69	35.77	39.24	785.61	-0.24
	PEDESTRIAN (1080p)	5232.46	38.90	43.28	44.61	402.57	5229.62	38.98	43.32	44.67	1791.97	-0.08
	Average ΔY-PSNR loss											-0.24

Table 4. Proposed search methods for integer-pel ME.

	Method 1 (X,Y)	Method 2 (X,Y)	Method 3 (X,Y)	Method 4 (X,Y)	Full search
Search range	–S ~ +(S–1), +S/2 ~ –(S–1)/2	–S ~ +(S–1), +S/2 ~ – (S–1)/2	–S ~ +(S–1), –S ~ +(S–1),	–S ~ +(S–1), –S ~ +(S–1),	–S ~ +(S–1), –S ~ +(S–1),
Search step	(1-pel, 1-pel)	(1-pel, 1-pel)	(1-pel, 2-pel)	(1-pel, 2-pel)	(1-pel, 1-pel)
Subsampling ratio of MB[1]	(1, 2)	(2, 2)	(2, 2)	(1, 2)	(1, 1)
Complexity[2] (cycle)	S^2MN	$S^2MN/2$	$S^2MN/2$	S^2MN	$4S^2MN$

1. The subsampling ratio of an MB is the sampling ratio in block matching. For example, a (1, 2) subsampling ratio calculates SAD for only a *Y* directional 2:1 subsampled 16×8 MB instead of a normal 16×16 MB, and a (2, 2) subsampling ratio calculates SAD for a bidirectional 2:1 subsampled 8×8 MB. This helps reduce the number of processing cycles for SAD calculation, where the search range is 16 and expresses two's complement format when implemented in hardware.
2. Complexity indicates the processing cycles when the designated method is implemented in hardware, where *S* is the search range and *M*×*N* is the searched block size.

Table 5. Performance comparison of candidate ME algorithms compared with full search algorithm in a JM13.2 reference encoder (the same test condition as in Table 1).

	Method 1			Proposed method 2			Method 3			Method 4		
Test sequences	PSNR (Y)	Bitrate (kbps)	ΔPSNR (dB)	PSNR (Y)	Bitrate (kbps)	ΔPSNR (dB)	PSNR (Y)	Bitrate (kbps)	ΔPSNR (dB)	PSNR (Y)	Bitrate (kbps)	ΔPSNR (dB)
CITY (4CIF)	32.61	1050.63	-0.10	32.57	1051.99	-0.14	32.07	1053.82	-0.64	32.16	1054.32	-0.55
CREW(4CIF)	34.67	1088.97	-0.15	34.64	1087.40	-0.18	34.75	1082.00	-0.07	34.77	1079.79	-0.05
HARBOUR (4CIF)	29.87	1257.20	0.01	29.86	1271.37	0.00	29.86	1278.98	0.00	29.86	1265.69	0.00
ICE (4CIF)	38.80	1033.48	-0.13	38.75	1033.45	-0.18	38.86	1034.07	-0.07	38.91	1033.37	-0.02
SOCCOR (D1)	31.04	1106.45	-0.22	30.98	1108.79	-0.28	30.91	1105.18	-0.35	30.94	1102.75	-0.32
MOBCAL (720p)	32.17	2106.50	0.00	31.95	2103.53	-0.22	31.76	2137.82	-0.41	31.86	2131.81	-0.31
PEDESTRIAN (1080p)	38.71	5227.88	-0.05	38.68	5229.36	-0.08	38.69	5232.97	-0.07	38.69	5227.03	-0.07
Average ΔPSNR loss	-0.09			-0.15			-0.23			-0.19		

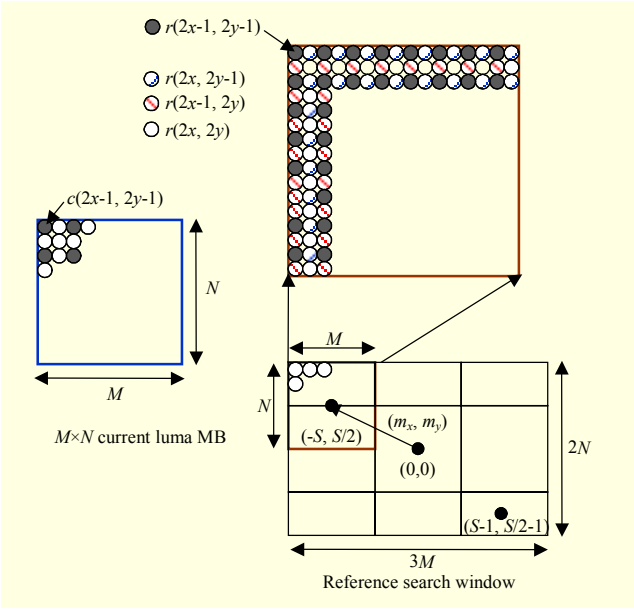


Fig. 1. Subsampled pixels for SAD calculation in method 2.

loss. The proposed pixel subsampling ME algorithm achieves an 87.5% reduction of computational complexity compared with the full search algorithm as shown in Table 4.

4. Fractional ME with Shared Memory

Subpixel ME requires the encoder to interpolate between integer sample positions in the reference frame. In the case of quarter-pixel ME, the best integer match is first found, and then the best half-pel position match in the immediate neighborhood is calculated. Finally, the best quarter-pel match around this half-pixel position is found [12]. It first searches the best half-pel MV for eight candidate positions around the previously calculated integer-pel and then finally searches the best quarter-pel MV around the half-pel in the same manner as in the half-pel search. Quarter-pel ME uses a six-tap finite impulse response (FIR) filter, and the six calculated samples are the neighboring samples of the position that the integer MV designates. Therefore, the reference window size used to search a fractional MV is $(3+3M+3)\times(3+2N+3)$ considering interpolation. If the estimated integer MV ranges between $X[-S+3, S-4]$ and $Y[-S/2+3, S/2-4]$, which expresses two's complement format in hardware, the sampling data to be considered for six-tap filtering for MEF wholly exists in the reference memory that was used for MEC. This is considerably effective because it fetches pixel data not from external frame memory, which requires further fetch cycles, but from internal MEC memory only. As previously stated, sharing MEC and MEF internal memory can reduce the number of fetch cycles and internal memory size. The hit ratio of the integer MVs existing within the previously mentioned range, which MEF can share with MEC reference memory, is explored. The average hit ratio is 93.9% for various test images as shown Fig. 2. The test conditions are the same as in Table 1, and FrameToBeEncoded is 300 frames. On the basis of these results, fast ME with a shared-memory-based hardware encoder is implemented.

III. Hardware Implementation

1. Proposed H.264/AVC High Profile Architecture

The proposed H.264/AVC High Profile hardware encoder has a four-stage MB level pipeline and consists of inter/intra blocks, deblocking, an entropy coder, rate control block, and FSM controller for the top schedule and pipeline management. It is connected with a 64-bit system/memory bus as shown Fig. 3. Most of all, the performance of the hardware encoder depends on not core processing cycles but on memory bandwidth, which is memory access cycles according to pixel data R/W between the external frame memory and the internal SRAM. To solve this problem, we used one-third of the

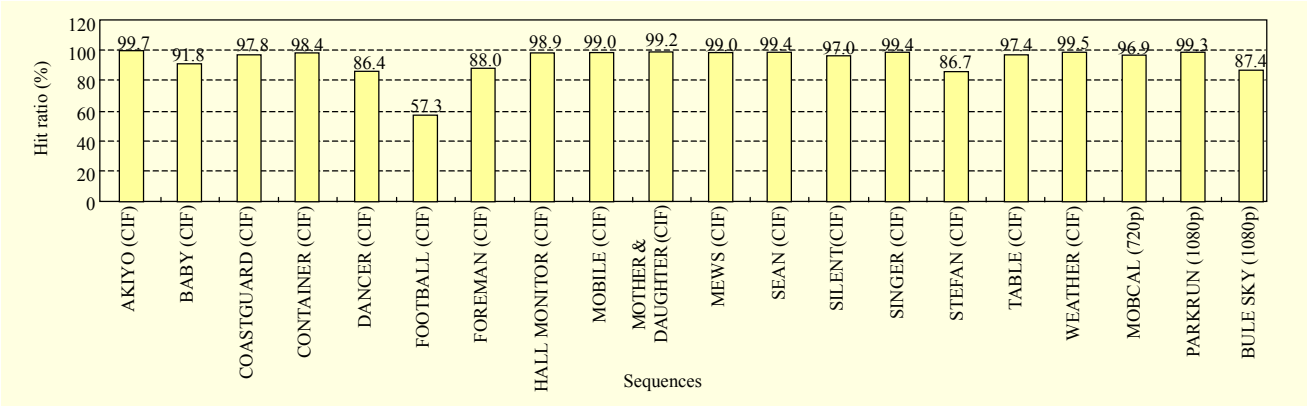


Fig. 2. Hit ratio of integer motion vectors existing within the range $X[-S+3, S-4]$ and $Y[-S/2+3, S/2-4]$ for various images.

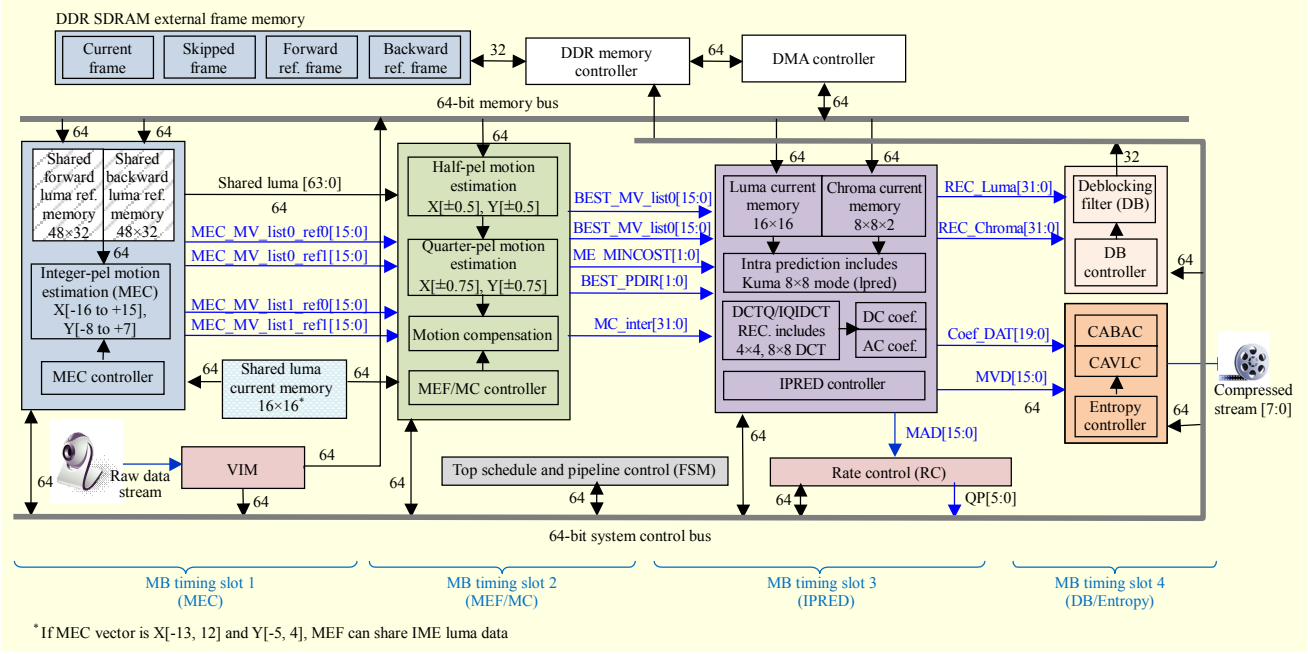


Fig. 3. Proposed H.264/AVC High Profile encoder hardware architecture.

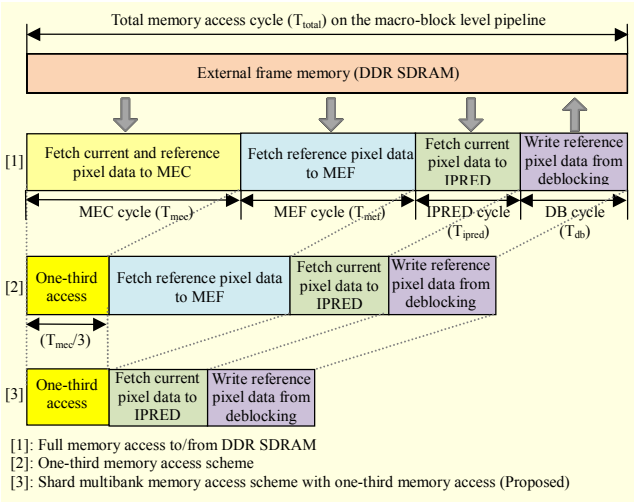


Fig. 4. Memory bandwidth reduction of the proposed encoder.

memory access schemes as in our previous works [14], [15].

The MEC stage searches integer MVs for two bi-directions (B-slice), where two motion-compensated reference areas are obtained from list 0 and list 1 pictures [12], respectively, and where the picture from list 0 is a forward picture and that from list 1 is a backward picture at this stage. When in interlace mode, each of the two reference pictures for list 0 and list 1 are used for ME and motion compensation (MC). Next, the MEF stage searches in detail the MVs up to the quarter-pel level and compensates the luma and chroma blocks with the predictive MV. As stated in section II.4, because MEF searches a fine MV in the neighborhood of the MEC-searched integer MV, the previously MEC-fetched reference pixel data in the shared memory can be reused if its vector is within the $X[-S+3, S-4]$ and $Y[-S/2+3, S/2-4]$ range. Therefore, MEF fetches new reference samples from the external memory only if MEC MV does not exist within the shared search range. To further reduce MEF internal SRAM access cycles, MEF does not store the fetched reference pixel data directly to the internal SRAM in the raw. After the reference pixel data for the search region is fetched from the shared SRAM or external DDR SDRAM, the MEF processes six-tap interpolation by 2-D FIR filtering immediately and then stores its result only to the internal MEF SRAM. These strategies limit unnecessary internal SRAM access cycles for fetch and save all of the ME processing cycles as well as limiting internal SRAM size. Figure 4 shows the memory bandwidth reduction of the proposed encoder with two memory access strategies. Though the pure MB processing cycle of each MB is within 512 cycles, it suffers from a limit of external memory bandwidth and CAS latency of the DDR/SDRAM. Therefore, there is a performance drop on the pipeline. These issues remain to be solved in future works. The overall performance of the proposed encoder is 1,100 cycle/MB, and it can encode a 30 frame/s full HD image at 270 MHz.

2. Coarse ME Architecture

MEC for searching an integer MV has the most processing cycles among other processing blocks and significantly affects the performance of the entire encoder. Basically, a single/dual

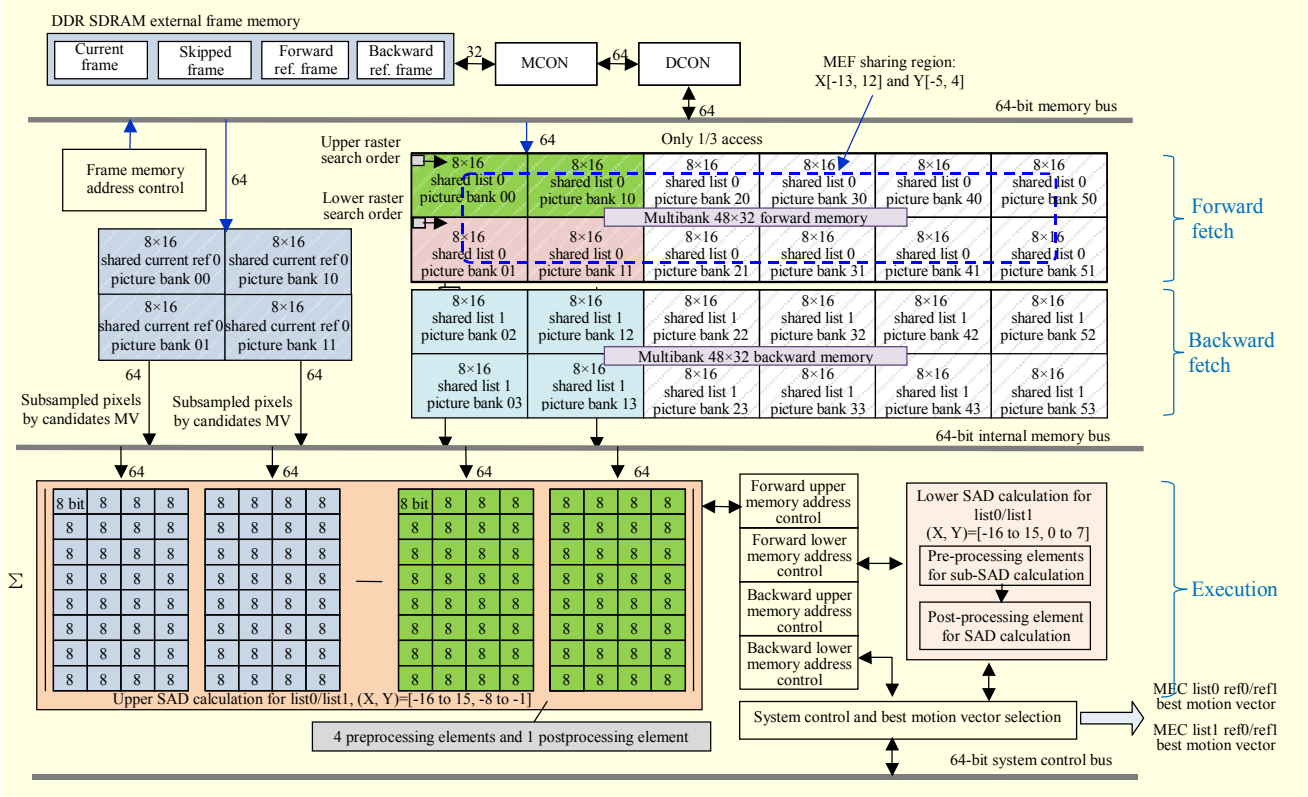


Fig. 5. Coarse motion estimation with shared multibank architecture.

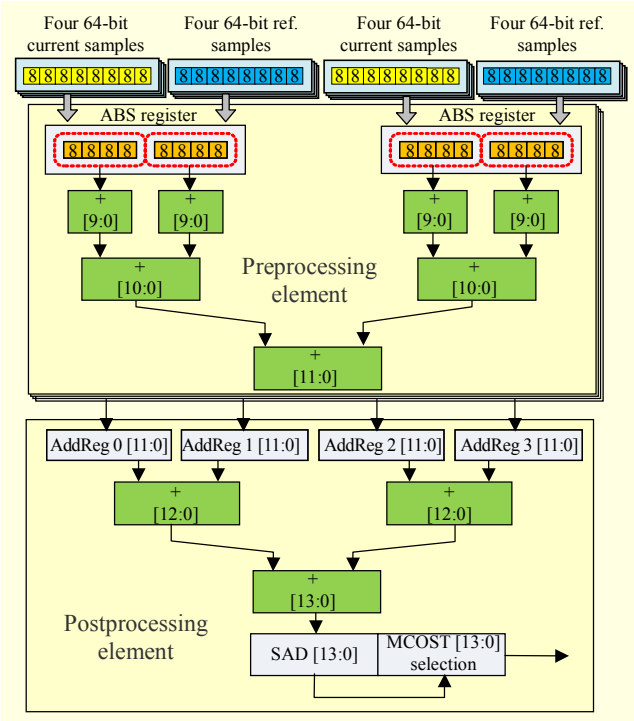


Fig. 6. High-throughput processing element.

port SRAM can read fast only one or two words within one cycle when one or two read addresses are given at the positive clock edge. However, multiwords of more than two words cannot be accessed at the same time at the SRAM because of its limited in-output port number. To solve the problem of throughput limit, a large SRAM is split into small SRAM blocks, and row slices and column banks are allocated to each partitioned SRAM as if it is a SDRAM as shown in Fig. 5. This multibank memory architecture can fetch multiwords randomly at one clock cycle wherever there is pixel data. Moreover, multibank memory is appropriate for our proposed one-third memory access scheme because the one-third bank change is needed at each MB cycle, and it has eight times the processing efficiency of a non-multibank memory architecture. Simultaneously fetched pixel data is transferred to eight preprocessing elements, and MCOST is finally calculated at one postprocessing element as shown in Fig. 6. The MEC automatically controls the bank change of memory at every MB cycle. Figure 7 shows the PE scheduling for multibank memory. The scheduling of the high-throughput PE consists of a five stage pipeline, including fetch, ABS, add, SAD, and MCOST stages. After pipeline throughput, an SAD for one candidate position is calculated within one cycle. Actually, the proposed pixel subsampling ME algorithm has a complexity of $S^2MN/2$ cycles (32,768 cycles in this case) as previously shown in Table 4.

However, because this number of cycles is still not

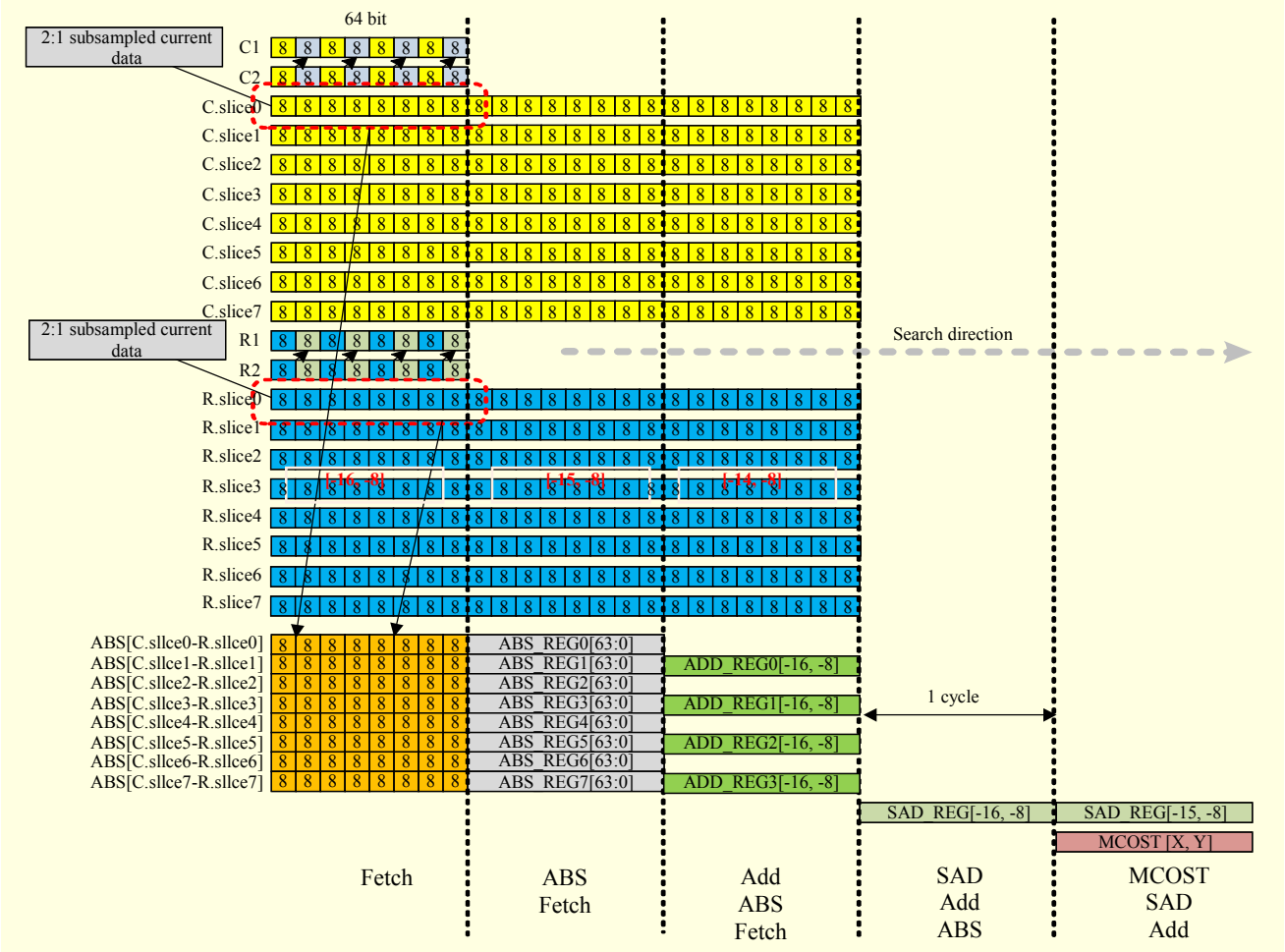
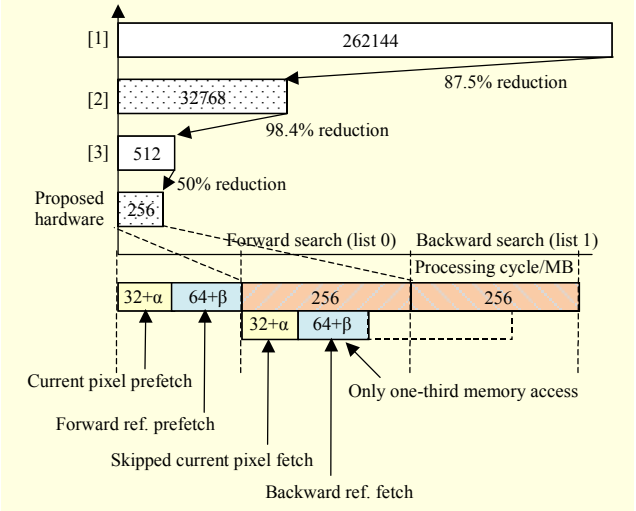


Fig. 7. High-throughput PE scheduling for multibank memory.



[1]: Full search algorithm
[2]: Proposed algorithm
[3]: Proposed algorithm with high-throughput PE, where the proposed hardware has a high-throughput PE and two SAD engines, and α and β are additional cycles according to the CAS latency of DDR/SDRAM.

Fig. 8. Processing cycle reduction for the presented MEC.

reasonable for real-time encoding on a hardware platform, two SAD engines are added to further enhance the throughput. As previously shown in Fig. 5, the upper SAD subblock calculates from samples at position *Y*[-8, 1], and the lower one calculates from samples at position *Y*[0, 7]. The MEC then selects the best one among the upper and lower SADs.

Consequently, the entire processing cycle reduces to S^2 cycles (256 cycles) in the P-slice and $2S^2$ (512 cycles) in the B-slice. During a forward search, the MEC first fetches backward reference samples for list 1 ME. Backward ME for the B-slice is successively executed after the forward ME. Figure 8 depicts the processing cycle reduction for the presented MEC.

IV. Implementation Results

The novel pixel subsampling ME-algorithm-based H.264/AVC High Profile encoder reference software was developed in C language, and the hardware encoder was designed using Verilog RTL language. The developed

reference model was used to verify the implemented encoder for various test images from CIF to full HD. To accelerate hardware verification, the designed encoder was prototyped on a Virtex 5 LX330 FPGA and was cosimulated using diverse test vectors. Its LUT usage was 81%, the estimated ASIC gate count was 872k, and internal memory size was 41.8 kB. The power consumption estimation of this encoder is 324.9 mW at 270 MHz. Power estimation was carried out with 65 nm CMOS technology by the Synopsys Power Compiler.

V. Rate-Distortion (RD) Performance

Figure 9 shows the RD performance for High Profile IBPBP sequences and CA-BAC. The test sequences for this were encoded using developed reference software with QP factors ranging from 24 to 36 in steps of 4. Here, its main test conditions are equivalent to those in Table 1, and the Baseline Profile uses IPPPP sequences and CA-VLC. Consequently, in the case of High Profile encoding, there is an average degradation of around 0.15 dB compared with the JM13.2 full search algorithm for various test images including the previous ones.

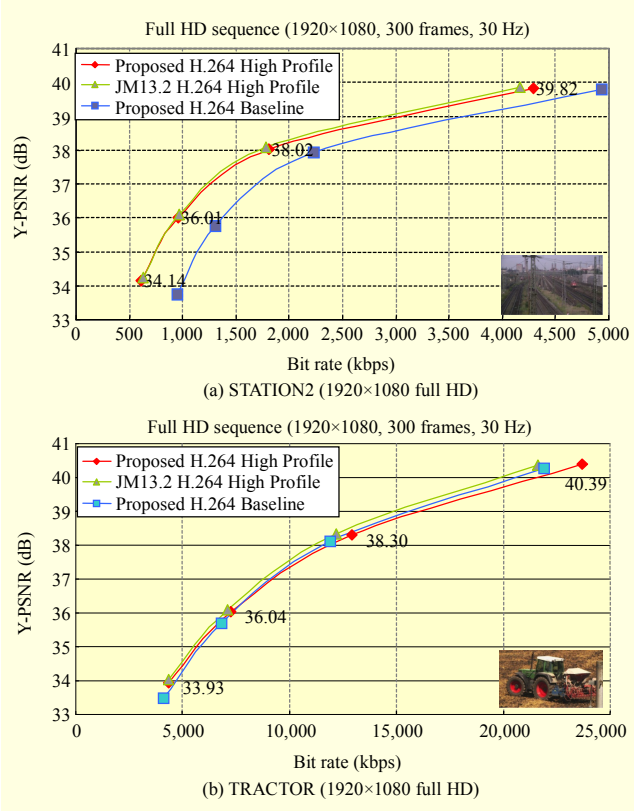


Fig. 9. RD performance of IBPBP sequences: (a) average ΔPSNR = -0.07 dB and (b) average ΔPSNR = -0.03 dB compared to the JM13.2 full search in the case of High Profile encoding.

VI. Conclusion

The developed H.264/AVC High Profile hardware encoder has a fast ME block with shared multibank memory. The proposed search algorithm is a pixel subsampling ME algorithm that achieves an 87.5% reduction of computational complexity compared with the full search algorithm. Although there is an average degradation of around 0.15 dB compared with JM reference software, it is acceptable considering the hardware complexity. On average, 93.9% of the pixel data in the shared memory can be shared and reused together with MEC and MEF to calculate SAD, reducing the number of data fetching and MB processing cycles. The shared multibank memory and high throughput PE significantly raise the performance of the encoder, and enable the encoding of a full HD image in real-time at 30 frame/s. While there are still remaining issues regarding memory bandwidth and the size of the rate control block, these are left to be solved in future works.

References

[1] ISO/IEC 14496-10, "Information Technology: Coding of Audio/Visual Objects-Part 10: Advance Video Coding," Sept. 2008.

[2] J.B. Lee and H. Kalva, *The VC-1 and H.264 Video Compression Standards for Broadband Video Services*, Springer Science, 2008, pp.78.

[3] B.G. Kim, J.H. Kim, and C.S. Cho, "Fast Inter Mode Decision Algorithm Based on Macroblock Tracking in H.264/AVC Video," *ETRI J.*, vol. 29, no. 6, Dec. 2007, pp. 736-744.

[4] G.F.F. Escibano et al., "A Fast MB Mode Decision Algorithm for MPEG-2 to H.264 P-Frame Transcoding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 2, Feb. 2008, pp. 172-185.

[5] A.C.W. Yu, G.R. Martin, and H. Park, "Fast Inter-Mode Selection in the H.264/AVC Standard Using a Hierarchical Decision Process," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 2, Feb. 2008, pp. 186-195.

[6] X. Xu and Y. He, "Improvements on Fast Motion Estimation Strategy for H.264/AVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 3, Mar. 2008, pp. 285-293.

[7] H.F. Ates and Y. Altunbasak, "Rate-Distortion and Complexity Optimized Motion Estimation for H.264 Video Coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 2, Feb. 2008, pp. 159-171.

[8] C.M Mak, C.K. Fong, and W.K Cham, "Fast Motion Estimation for H.264/AVC in Walsh-Hardamard Domain," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 6, June 2008, pp. 735-745.

[9] C.S. Kannangara, I.E. Richardson, and A.J. Miller, "Computational Complexity Management of a Real-Time

H.264/AVC Encoder," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 9, Sept. 2008, pp. 1191-1200

[10] S. Saponara et al., "Dynamic Control of Motion Estimation Search Parameters for Low Complex H.264 Video Coding," *IEEE Trans. Consum. Electron.*, vol. 52, no. 1, Feb. 2006, pp. 232-239.

[11] S. Lopez et al., "Grouped Approach for the Design of H.264/AVC Motion Estimation Architectures," *ETRI J.*, vol. 30, no. 6, Feb. 2008, pp. 862-864.

[12] I.E.G. Richardson, *H.264 and MPEG-4 Video Compression*, John Wiley & Sons Ltd., 2003.

[13] Joint Model (JM) – H.264/AVC Reference Software, http://iphome.hhi.de/suehring/tml/download/.

[14] S.H. Lee et al., "A 40 MHz Dedicated Hardware H.264/AVC Video Encoder with the Reducing Memory Access Scheme," *IEEE Int. Symp. Consum. Electron.*, ISCE.2008.4559542, Apr. 2008, pp. 1-4.

[15] S.M. Park et al., "A MPEG-4 Video Codec Chip with Low Power Scheme for Mobile Application," *IEICE Trans. Fundam. Electron., Commun. and Comput. Sci.*, vol. E86-A, no. 6, June 2003, pp. 1353-1363.

Sukho Lee received the BS degree in electronics engineering from Kyunghee University, Seoul, Korea, in 1995. From 1995 to 1997, he was with the Samsung Electronics, Yongin, Korea, where he worked on ASIC design and DSP design. He received the MS degree in electronics engineering at Korea University, Seoul Korea, in 1999. In 1999, he joined ETRI, Daejeon, Korea, and participated in SoC design development. He is currently a senior researcher and has been engaged in research on SoC design, image compression algorithms, and SoC architecture design. He is currently working toward the PhD degree with the Department of Information Communications Engineering, Chungnam National University, Korea. His main research interests are video coding, image compression, and low-power SoC architecture design.

Seongmo Park received the BS, MS, and PhD degrees in electronics engineering from Kyungpook National University, Daegu, Korea, in 1985, 1987, and 2006, respectively. From 1987 to 1992, he was with the LG semiconductor company, Gumi, Korea, where he worked on ASIC design and Mask ROM design. In 1992, he was with ETRI, Daejeon, Korea and participated in SoC design development. He is currently engaged in research on SoC design, image compression algorithms, and SoC architecture design. He is now a team leader of the multimedia processor design team and a professor with the University of Science and Technology. He has published over 30 technical papers in international journals and conference proceedings. His main research interests are video coding, image compression, multiprocessor design and low-power SoC architecture design.

Jongwon Park received the BE degree in electronics engineering from Chungnam National University (CNU), Korea, in 1979. He received the MS degree in computer science from Korea Advanced Institute of Science and Technology (KAIST), Korea, in 1981. From 1981 to 1993, he was with the Department of Computer Science of CNU. He worked towards the PhD degree in computer science at KAIST from 1988 to 1991. In 1992, Dr. Park was a visiting associate professor with the Department of Electrical Engineering of the University of Texas, Dallas, USA. Since 1994, he has been with the Department of Information Communications Engineering at the CNU, where he is currently a professor. His main research interests include parallel computer architecture, image processing and computer vision, pattern recognition and parallel memory systems, as well as image processing systems and computer vision. He is a senior member of the IEEE and the IEEE Computer Society.