Design of a Fast Multi-Reference Frame Integer Motion Estimator for H.264/AVC

Juwon Byun and Jaeseok Kim

Abstract—**This paper presents a fast multi-reference frame integer motion estimator for H.264/AVC. The proposed system uses the previously proposed fast multi-reference frame algorithm. The previously proposed algorithm executes a full search area motion estimation in reference frames 0 and 1. After that, the search areas of motion estimation in reference frames 2, 3 and 4 are minimized by a linear relationship between the motion vector and the distances from the current frame to the reference frames. For hardware implementation, the modified algorithm optimizes the search area, reduces the overlapping search area and modifies a division equation. Because the search area is reduced, the amount of computation is reduced by 58.7%. In experimental results, the modified algorithm shows an increase of bit-rate in 0.36% when compared with the five reference frame standard. The pipeline structure and the memory controller are also adopted for real-time video encoding. The proposed system is implemented using 0.13 um CMOS technology, and the gate count is 1089K with 6.50 KB of internal SRAM. It can encode a Full HD video (1920x1080P@30Hz) in real-time at a 135 MHz clock speed with 5 reference frames.**

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

Manuscript received Nov. 30, 2012; accepted May. 30, 2013
The authors are with the Department of Electrical and Electronic Engineering, Yonsei University, Seoul 120-749, Korea.
E-mail : jaekim@yonsei.ac.kr

I. INTRODUCTION

H. 264/AVC is the latest video coding standard of the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG). This standard provides much higher compression than earlier standards such as MPEG-2 or MPEG-4. The higher coding efficiency comes from several new features such as the multi-reference frame, variable sub block size, quarter-pixel accuracy motion vector, intra prediction, integer transformation, adaptive in-loop deblocking filter and enhanced entropy coding methods [1]. Although the coding efficiency of H.264/AVC is much higher than that of previous standards, these new features increase the computational complexity and make it difficult to implement in mobile devices [2-5].

Multi-reference frame motion estimation is one of the new features in H.264/AVC. It increases the performance of the video encoder by using up to five reference frames, but this feature greatly increases the computation quantity. The number of computations is linear relative to the number of reference frames, so H.264/AVC, which uses multi-reference frames, takes more time than earlier standards such as MPEG-2 or MPEG-4. In particular, the integer pixel motion estimation module is one of the most time-consuming blocks, consuming 74.25% of the execution time in the H.264/AVC encoder [6]. Because the number of computations required for multi-reference frame motion estimation is very large, implementation of a real-time multi-reference integer motion estimation encoder is difficult, and most of the previous integer motion estimators used only one reference frame or two reference frames [7-11].

In this paper we design a fast multi-reference frame

integer motion estimator for H.264/AVC. For this work, we use and modify the fast multi reference frame motion estimation algorithm that was proposed in our in previous work [12]. The previously proposed algorithm reduces the search area by using a linear feature of object motion and does not use early termination by threshold value. This feature gives us a fixed processing time schedule and a small sized hardware area. It makes hardware implementation easy.

The rest of this paper is organized as follows: Section 2 introduce the multi-reference frame motion estimation and the previously proposed algorithm. In Section 3, we modify the previously proposed algorithm for hardware implementation. Section 4 provides the experimental results. In Section 5, we design the fast multi reference integer motion estimator for H.264/AVC. Section 6 provides the implementation results and a comparison of the proposed system with previous systems. Finally, we conclude this paper in Section 7.

II. MULTI-REFERENCE FRAME MOTION ESTIMATION AND PREVIOUS FAST ALGORITHMS

1. The Multi-Reference Frame Motion Estimation

In the latest video standard, H.264/AVC, several features are added to the motion estimation module and they make the amount of computation required for motion estimation huge. In particular, multi-reference frame motion estimation, which uses up to five reference frames, provides a highly efficient coding rate, but requires a great deal of computations [13, 14]. To investigate the number of computation required for multi-reference frame motion estimation, we measured the motion estimation time and bit-rate. Fig. 1 shows the relationship between the bit-rate and processing time of motion estimation. The x-axis shows the number of reference frames, the dashed line shows the motion estimation time and the solid line shows the percentage increment of the bit-rate. This graph shows that the amount of computation for multi-reference frame motion estimation has a linear relationship with the number of reference frames. For this reason, the complexity of computations for multi-reference frame motion estimation that uses five reference frames is five times

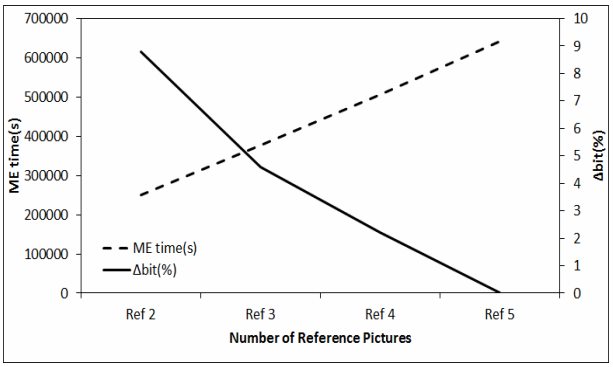

Fig. 1. Relationship between the number of reference frames, bit-rate and ME time.

more than that of single reference frame motion estimation. The motion estimation computation time is very large, and the bit-rate is low in motion estimation that uses five reference frames. Because of the computation complexity, the previous integer motion estimators use only one reference frame or two reference frames. To implement the multi reference frame motion estimator, the fast multi reference frame motion estimation algorithm is required and such fast multi-reference algorithms were developed.

2. Fast Multi-frame Motion Estimation Algorithm with Adaptive Search Strategies (FMASS) [16]

The algorithm of FMASS [16] uses two features of multi-reference frame motion estimation. The first feature is that the probability of selecting ref_3 and ref_4 is lower than 3%. The second feature is that ref_3 and ref_4 are more likely to be selected under conditions of vibration or smooth moving. To use these features, first, FMASS executes motion estimation in ref_0, ref_1, and ref_2. Then, if the values of the difference between the reference frame 0 motion vector and other reference frame motion vectors is smaller than certain threshold values and the matching cost of ref_0 is smaller than that of the other reference frames, the motion estimation of ref_3 and ref_4 will be skipped.

FMASS uses a feature such that ref_3 and ref_4 will be more likely to be selected under conditions of vibration or smooth moving, but it uses threshold values. It does not provide any gain regarding implementation on hardware.

3. Adaptive and Fast Multi-frame Selection Algorithm (AFMFSA) [17]

AFMFSA [17] consists of three skip methods. The first method uses a relationship between the current block and its neighboring blocks, which are located above and to the left side of the current block. If the current block is not located at object boundaries and its neighboring blocks have the same optimal reference frame, ref_n, its ultimate reference frame has a very large probability (equal to 95%) of being ref_n. The second method uses the magnitudes of the ref_0 and ref_1 motion vectors. If the magnitudes of the ref_0 and ref_1 motion vectors are very small or large, the motion estimation in ref_2, ref_3, and ref_4 can be skipped. The third method uses the Sum of Absolute Difference (SAD) value of the reference frames. If the magnitudes of the ref_0 and ref_1 motion vectors have suitable sizes, the motion estimation in ref_2 is executed. After the execution of a motion estimation in ref_2, AFMFSA classifies the current block into four types by using the SADs of ref_0, ref_1, and ref_2. If the SAD of ref_0 has a minimum value and that of ref_2 has a maximum value, the motion estimation of ref_3 and ref_4 can be skipped. If the SAD of ref_1 has a maximum value, the SAD of ref_3 determines an execution of motion estimation in ref_4. If the SAD of ref_0 has a maximum value and that of ref_2 has a minimum value, the motion estimation of ref_3 and ref_4 is executed.

AFMFSA determines on the execution of motion estimation for each reference frame at every reference frame. It gives non-fixed timing, and data dependency appears. These make it difficult to implement a pipeline structure or a parallel structure.

4. Fast Reference Frame Selection Algorithm (FRFSA) [18]

FRFSA [18] uses a temporal correlation. It classifies the reference regions into four types using the reference frame index of the anchor block. The anchor block is the macro block that has the same position in the previous encoded frame. Region0 represents the entire reference index of the anchor blocks that are ref_0. In the same manner, Region1 consists of ref_0 and ref_1, and Region2 (or Region3) is composed of ref_0, ref_1 and ref_2 (ref_0, ref_1, ref_2, ref_3 and ref_4).

To implement the motion estimator hardware for H.264/AVC, FRFSA requires additional memory that saves the reference frame indices of the previously encoded frame. It shows increases in hardware size and cost.

5. The Previously Proposed Algorithm [12]

As we explained in the previous section, previous fast multi-reference frame motion estimation algorithms, such as FMASS [16], AFMFSA [17], and FRFSA [18] have problems in that they use threshold values, have a large data dependency, and require the additional memory. It makes hardware implementation difficult and provides no gains for hardware implementation. To resolve these problems, we proposed a new fast multi-reference motion estimation algorithm. Our fast multi-reference algorithm uses a linear relationship between the motion vector and the distances from the current frame to the reference frames. To resolve the estimated motion vector, we find the center positions of the reduced search areas by using the motion vector of the previous reference frame and the Picture Order Counts (POC) of each frame. The center positions of the reduced search areas are solved for by using Eq. (1).

$$CP_{n,i} = MV_i \times \frac{POC_n - POC_{curr}}{POC_i - POC_{curr}} + P_{curr} \qquad (1)$$

where i=0 or 1, n = 2 or 3 or 4

MV_i is the motion vector in ref_i. POC_n and POC_{curr} are the POC of reference frame n (ref_n) and the POC of the current frame respectively. $CP_{n,i}$ is the center position in ref_n by MV_i. P_{curr} is the current position in the current picture. The proposed algorithm is described as follows:

1. Process motion estimation in ref_0.
2. Process motion estimation in ref_1.
3. Solve the center positions of the search area in ref_2, ref_3 and ref_4.
4. Set the two center positions of the reduced search areas in ref_2, ref_3 and ref_4.
5. Process motion estimation in ref_2 reduced search areas.
6. Process motion estimation in ref_3 reduced search areas.

7. Process motion estimation in ref_4 reduced search areas.

Fig. 2 shows the search area of the previously proposed algorithm. In ref_0 and ref_1, the search area is the full-size search area. In ref_2, ref_3, and ref_4, the search area of each frame consists of two reduced search areas and each reduced search area size is 1/16 of the full search area. One of the two reduced search areas is resolved by the motion vector in ref_0. The other is resolved by the motion vector in ref_1. Consequentially, the previously proposed algorithm uses 2.375 reference frames, and the number of computations is reduced by 52.5% as compared to the standard algorithm, which uses five reference frames and does not use a threshold value. It provides fixed timing for hardware implementation and gives large gains for hardware implementation, such as easy time scheduling and reduced calculations. In addition, there is no data dependency among ref_2, ref_3, and ref_4. Therefore, an implementation of a pipeline structure or parallel structure is possible.

III. ALGORITHM MODIFICATION

To design the hardware of the previously proposed fast multi-reference frame motion estimation algorithm, some modification of the previously proposed algorithm is required. The first modification point is the division of Eq. (1). This equation is solving for the center position of the reduced search areas in ref_2, ref_3, and ref_4. The second modification point is the optimization of the reduced search area size. Because the full search area size is increased to 64x64, the optimization of the reduced search area size is required. The third modification point is the overlap of the reduced search areas in ref_2, ref_3, and ref_4. The overlap of the search areas creates overhead in terms of computation and memory bandwidth. We modify the previously proposed algorithm to solve these three problems.

1. The Modification of the Central Point Solving Equation

The previously proposed fast multi-reference frame motion estimation algorithm uses Eq. (1) to solve for the reduced search areas in ref_2, ref_3, and ref_4. Eq. (1) has the division operation. This makes hardware implementation difficult. For this reason, the modification of Eq. (1) is required.

H.264/AVC standard uses the temporal direct mode. The temporal direct mode predicts the reference frame index and motion vector by using the motion information of the anchor block, which is the same position block as in the previously encoded frame. One reference frame of the current block is the same as that of the anchor block, and another reference frame is the anchor frame. Motion vectors of the current block are solved for by using the motion vectors and time distances of each frame. Fig. 3 shows the temporal direct mode in H.264/AVC standard. MV_{L0} is the L0 motion vector of the current block, and MV_{L1} is the L1 motion vector of the current block. MV_{Col} is the motion vector of the anchor block. tb is the distance from the reference frame to the current frame. td is the distance from the reference frame to the anchor frame. MV_{L0} and MV_{L1} are solved for by using MV_{Col}, tb, td, and Eq. (2).

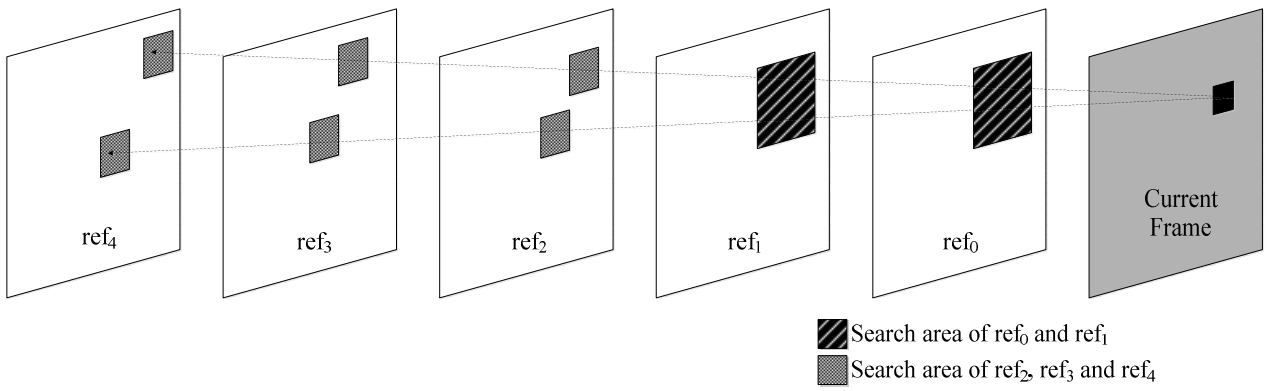

Fig. 2. Search area of the proposed algorithm.

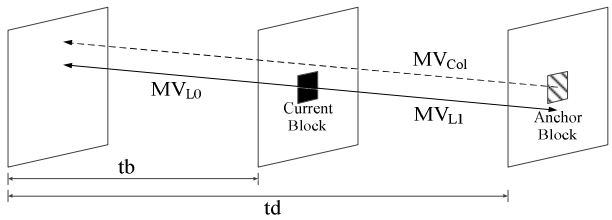

Fig. 3. The temporal direct mode in H.264/AVC standard.

$$MV_{L0} = MV_{Col} \times \frac{tb}{td}$$
$$MV_{L1} = MV_{L0} - MV_{Col} \tag{2}$$

We can show that Eq. (2) is very similar to Eq. (1). H.264/AVC standard modifies Eq. (2) in order to create Eq. (3) for solving the problems of the division operation. POC_{Anchor} is the POC of the anchor frame, and POC_{ref} is that of the reference frame. The division operation of Eq. (3) can be designed by using a reasonable amount of Read Only Memory (ROM), multipliers, and adders. Eq. (4) is a version of Eq. (1) that is modified by using the same method. It can also be implemented by using ROM, multipliers, and adders.

$$\begin{aligned} tb &= Clip(-128,127, POC_{curr} - POC_{ref}) \\ td &= Clip(-128,127, POC_{Anchor} - POC_{ref}) \\ tx &= (16384 + (td/2))/td \\ &DistScaleFactor \\ &= Clip(-1024,1023, (tb \times tx + 32) >> 6) \\ MV_{L0} &= (DistScaleFactor \times MV_{Col} + 128) >> 8 \end{aligned} \tag{3}$$

$$\begin{aligned} tb &= Clip(-128,127, POC_{curr} - POC_n) \\ td &= Clip(-128,127, POC_i - POC_n) \\ tx &= (16384 + (td/2))/td \\ &DistScaleFactor \\ &= Clip(-1024,1023, (tb \times tx + 32) >> 6) \\ CP_{n,i} &= (DistScaleFactor \times MV_i + 128) >> 8 + P_{curr} \end{aligned} \tag{4}$$

2. The Optimization of the Reduced Search Area Sizes

In previous work [12], the size of the full search area was 32x32. In this work, the full search area size is increased to 64x64 for higher performance of the designed motion estimator. Therefore, an adjustment of the reduced search area size is required. To find the optimum size of reduced search areas, the performance of 8x8 reduced search areas is compared with that of 16x16 reduced search areas. Table 1 displays the difference values of PSNR and bit-rate of 8x8 reduced search areas when compared with those of 16x16 reduced search areas. The quantization parameter is 25. Simulation results show that the PSNR drop is only 0.006 dB and bitrate is reduced to 0.04% rather. In other words, the performance drop is negligible when the size of reduced search areas is decreased. For this reason, the reduced search areas are decreased to 1/64.

3. The Removal of Overlap Operations

The previously proposed fast multi-reference frame motion estimation algorithm decreases the number of computations by reducing the search areas in ref_2, ref_3, and ref_4. The search areas of ref_0 and ref_1 are a full-sized search areas. In ref_2, ref_3, and ref_4, the search area of each frame consists of two reduced search areas, and each reduced search area size is 1/64 of the full search area. Because the number of search areas is two in ref_2, ref_3, and ref_4, search areas overlap in such cases. Table 2 shows the proportion that $CP_{n,0}$ is the same as $CP_{n,1}$.

Simulation results show that the same proportion $CP_{n,0}$ as $CP_{n,1}$ in ref_2, ref_3 and ref_4 is more than 38%. The result

Table 1. The difference values of PSNR and bit-rate of 8x8 reduced search areas compared with those of 16x16 reduced search areas

Sequence		△PSNR(dB)	△Bit (%)
QCIF	Akiyo	-0.001	0.40%
	Carphone	-0.038	0.14%
	Coastguard	-0.011	-0.32%
	Container	0.007	-0.20%
	Foreman	-0.041	0.41%
	Mobile	-0.007	-0.46%
	Mother-daughter	-0.002	-0.21%
CIF	Coastguard	-0.007	0.15%
	Container	0.001	0.10%
	Foreman	-0.007	0.81%
	Mobile	0.000	0.10%
	Mother-daughter	-0.011	0.07%
	News	0.003	0.20%
	Silent	-0.003	0.13%
4CIF	City	-0.003	0.21%
	Crew	-0.002	-0.18%
	Harbour	-0.001	0.08%
	Ice	-0.004	-0.15%
720P	Mobcal	-0.002	-0.19%
	Parkrun	0.000	-0.08%
	Shields	0.000	-0.12%
Average		-0.006	-0.04%

Table 2. the same proportion $CP_{n,0}$ as $CP_{n,1}$

Size	Sequence	The Same Portion of $CP_{n,0}$ as $CP_{n,1}$
QCIF	akiyo	87.14 %
	carphone	39.29 %
	coastguard	11.84 %
	container	85.88 %
	foreman	23.72 %
	mobile	27.59 %
	mother-daughter	71.11 %
CIF	coastguard	3.71 %
	container	79.57 %
	foreman	20.38 %
	mobile	11.18 %
	mother-daughter	66.00 %
	news	78.72 %
	silent	71.75 %
4CIF	city	3.66 %
	crew	15.93 %
	harbour	5.05 %
	ice	68.01 %
720p	mobcal	30.87 %
	parkrun	1.22 %
	shields	9.75 %
Average		38.68 %

indicates that overlapped search areas must not be neglected. Overlapped search areas cause unnecessary increases of motion estimation computation. It increases the power consumption of motion estimation system large. The power consumption is very important to design a system for mobile devices. For this reason, although the hardware time scheduling becomes more complex, we modify the proposed fast multi-reference frame motion estimation algorithm in order to reduce the number of computation by eliminating overlapping search areas.

Fig. 4 shows the overlapping search area and reduced search areas in ref_2, ref_3, and ref_4 by using MV_0 and MV_1. $ref_{n,i}$ is the reduced search area of ref_n by MV_i. If $CP_{n,0}$ is the same as $CP_{n,1}$, the motion estimation of the search area by MV_1 is skipped. This decreases the use of unnecessary memory bandwidth. If $CP_{n,0}$ is not the same as $CP_{n,1}$ and if the absolute lengths of both dx and dy are shorter than R, the motion estimation of these search points is skipped in order to remove the overlapping computation. The difference of computation quantity between maximum and minimum is much less than that of previous fast algorithms. Because of this reason, the modified algorithm has more advantages than previous fast algorithms when designing a hardware architecture. The proposed modified multi-reference frame motion estimation algorithm is described as follows:

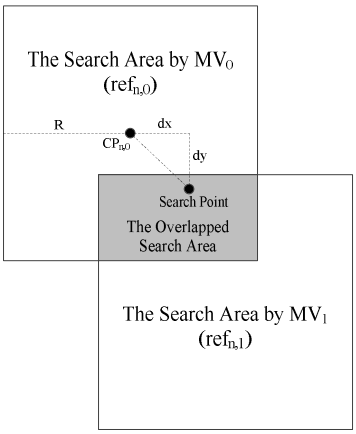

Fig. 4. The overlapped search area in ref_2, ref_3 and ref_4.

1. Process motion estimation in ref_0 and ref_1.
2. Solve the center positions of the search area in ref_2, ref_3, and ref_4.
3. Set the two center positions of the reduced search areas in ref_2, ref_3, and ref_4.
4. Process motion estimation at all search points of $ref_{2,0}$.
5. If $CP_{2,0}$ is the same as $CP_{2,1}$, the motion estimation of $ref_{2,1}$ is skipped. Otherwise, process motion estimation at the search points of $ref_{2,1}$ where dx and dy are longer than R.
6. Process motion estimation at all search points of $ref_{3,0}$.
7. If $CP_{3,0}$ is the same as $CP_{3,1}$, the motion estimation of $ref_{3,1}$ is skipped. Otherwise, process the motion estimation at search points of $ref_{3,1}$ where dx and dy are longer than R.
8. Process motion estimation at all search points of $ref_{4,0}$.
9. If $CP_{4,0}$ is the same as $CP_{4,1}$, the motion estimation of $ref_{4,1}$ is skipped. Otherwise, the process motion estimation at search points of $ref_{4,1}$ where dx and dy are longer than R.

IV. SIMULATION RESULTS

In Section 3, we modified the previously proposed fast multi-reference motion estimation algorithm. To compare the modified algorithm with the previously proposed algorithm and the previous fast algorithms, we simulate some sequences that have various sizes. We use JM v9.6 reference software. The quantization parameter

Table 3. Comparison of the modified algorithm with the previously proposed algorithm m and previous fast algorithms

		FMASS [16]			AFMFSA [17]			FRFSA [18]			The Modified Algorithm		
		# Ref. Frame	Δ PSNR (dB)	ΔBit (%)	# Ref. Frame	Δ PSNR (dB)	ΔBit (%)	# Ref. Frame	Δ PSNR (dB)	ΔBit (%)	# Ref. Frame	Δ PSNR (dB)	ΔBit (%)
QCIF	Akiyo	3.014	-0.091	1.06%	1.306	-0.133	3.79%	1.002	-0.133	3.72%	2.050	-0.046	0.13%
	Carphone	3.223	-0.054	0.72%	2.600	-0.087	2.23%	1.300	-0.248	5.24%	2.062	0.001	0.20%
	Coastguard	3.181	0.006	-1.39%	2.347	-0.031	-0.50%	1.040	-0.066	-1.12%	2.066	-0.018	-0.72%
	Container	3.017	-0.058	1.87%	1.786	-0.083	2.90%	1.356	-0.059	3.03%	2.050	-0.009	0.07%
	Foreman	3.224	-0.006	0.68%	2.938	-0.021	1.27%	1.030	-0.215	2.29%	2.065	0.042	1.97%
	Mobile	3.121	-0.072	-1.24%	2.510	-0.112	1.66%	3.835	-0.033	-0.79%	2.062	-0.030	-0.36%
	Mother-daughter	3.048	-0.022	0.06%	1.414	-0.060	1.67%	1.095	-0.066	2.24%	2.053	-0.055	0.37%
CIF	Coastguard	3.329	0.005	-1.05%	2.125	-0.022	0.12%	1.097	-0.044	-0.07%	2.076	-0.018	-0.01%
	Container	3.032	-0.015	0.65%	1.929	-0.021	0.71%	1.319	-0.018	0.86%	2.051	-0.001	0.29%
	Foreman	3.387	-0.027	0.55%	2.234	-0.061	1.47%	1.086	-0.177	2.96%	2.071	-0.018	1.73%
	Mobile	3.207	-0.051	0.46%	2.314	-0.138	3.71%	2.748	-0.088	1.11%	2.066	0.005	0.18%
	Mother-daughter	3.080	-0.022	0.59%	1.311	-0.071	2.30%	1.063	-0.063	2.81%	2.055	0.002	1.34%
	News	3.067	0.009	0.14%	1.311	-0.029	0.81%	1.040	-0.040	0.89%	2.052	-0.005	-0.01%
	Silent	3.133	-0.019	0.60%	1.381	-0.058	3.17%	1.040	-0.077	3.96%	2.056	-0.020	1.60%
4CIF	City	3.442	0.003	0.21%	1.462	-0.041	2.11%	1.024	-0.049	2.01%	2.080	-0.011	1.09%
	Crew	3.525	-0.006	0.05%	1.946	-0.034	1.01%	1.055	-0.047	2.43%	2.077	-0.011	0.96%
	Harbour	3.500	-0.009	-0.17%	2.041	-0.048	1.51%	1.324	-0.060	1.11%	2.080	-0.009	0.49%
	Ice	3.261	-0.015	-0.27%	1.418	-0.067	1.53%	1.040	-0.071	1.55%	2.060	-0.024	0.95%
720P	Mobcal	3.133	-0.094	7.92%	1.652	-0.182	13.16%	1.659	-0.242	20.01%	2.062	-0.002	0.03%
	Parkrun	3.279	-0.006	-0.15%	1.211	-0.037	1.15%	1.096	-0.044	1.29%	2.076	-0.001	-0.09%
	Shields	3.381	-0.016	0.55%	1.281	-0.078	3.63%	1.045	-0.085	3.47%	2.075	0.000	-0.04%
Total Average		3.218 -35.6%	-0.027	0.56%	1.834 -63.3%	-0.067	2.35%	1.307 -73.1%	-0.092	2.81%	2.064 -58.7%	-0.011	0.48%

is 30, and the size of the search area is 64x64. With these setting values, the search area of ref_0 and ref_1 is one 64x64 square, and that of ref_2, ref_3 and ref_4 is two 8x8 squares. The QCIF (176x144)-sized sample sequences are ‘Akiyo,’ ‘Carphone,’ ‘Coastguard,’ ‘Container,’ ‘Foreman,’ ‘Mobile,’ and ‘Mother-daughter.’ The CIF (352x288)-sized sample sequences are ‘Coastguard,’ ‘Container,’ ‘Foreman,’ ‘Mobile,’ ‘Mother-daughter,’ ‘News,’ and ‘Silent’. The 4CIF (704x576)-sized sample sequences are ‘City,’ ‘Crew,’ ‘Harbor,’ and ‘Ice.’ The 720p (1280x720)-sized sample sequences are 'Mobcal,' 'Parkrun,' and 'Shields.' The number of encoded frames is 99 and CABAC is used for entropy coding. Table 3 shows the result of the simulation when compared with the modified algorithm, previous fast algorithms, and the standard, which uses five reference frames. △PSNR and △Bit refer to the difference values of PSNR and bit-rate when compared with the standard, which uses five reference frames. According to these results, the maximum PSNR drop of FMASS is 0.094 dB, and the maximum bit-rate increment is 7.92%. The number of reference frames is reduced by a maximum of 1.986, but reduced only by 1.475 in the extreme case. The maximum PSNR drop of AFMFSA is 0.182dB, and the maximum bit-rate increment is 13.16%. In particular, it has low performance in the ‘Akiyo’ QCIF sequence and the 'Mobcal' 720p sequence. AFMFSA uses 1.211 reference frames in the 'Parkrun' 720p sequence and uses 2.938 reference frames in the 'Foreman' QCIF sequence. The maximum bit-rate increment of FRFSA is 20.01% and the maximum PSNR drop is 0.248 dB. In particular, It has low performance in the ‘Mobcal’ 720P sequence. The bit-rate increment of the modified algorithm is less than 2% in every sequence, and the PSNR drop is a maximum of 0.046 dB which is much smaller than those of the previous fast algorithms. Fig. 5 compares the rate-distortion curves of various video sequences. The 720p-size sample sequences are ‘Mobcal’ and ‘Shields,’ the CIF-size sample sequence is ‘Mobile,’ and the QCIF-size sample sequence is ‘Container’. The performance of the modified algorithm is much better than previous fast algorithms. To compare the various quantization parameters, we use the rate-distortion curves, Bjontegaard Delta PSNR (BDPSNR) and Bjontegaard Delta Bit-rate (BD-bit-rate) [19]. BDPSNR and BD-bit-rate display performance by combining changes of PSNR

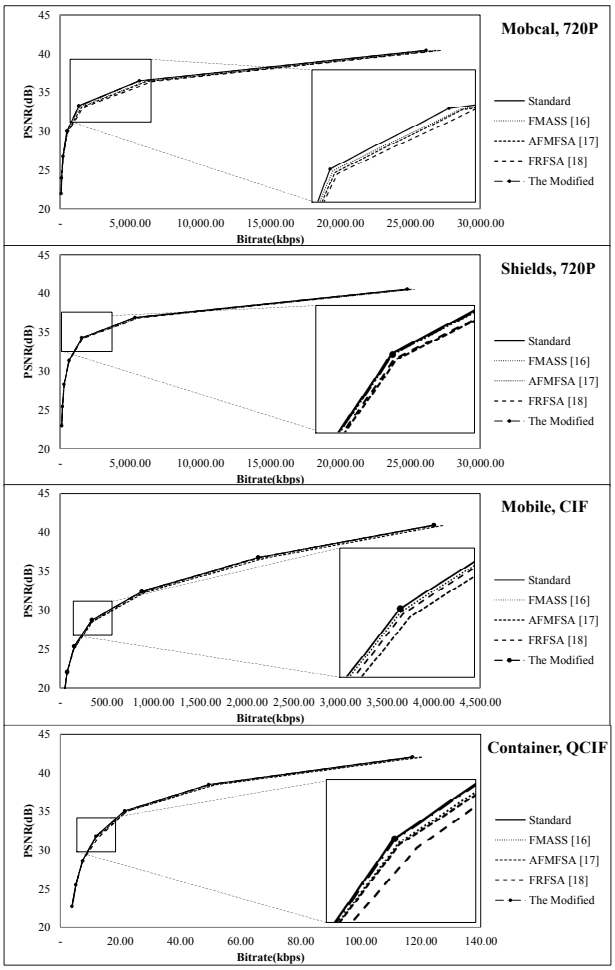

Fig. 5. Rate-distortion curve comparisons.

and bit-rate. We simulate using the same sample sequences in order to solve for BDPSNR and BD-bit-rate. The quantization parameter values are changed to 20, 25, 30, 35, 40, 45, and 50. Other simulation environments are the same as previously described. Table 4 and 5 represent the difference values of BDPSNR and BD-bit-rate when compared with the standard, which uses five reference frames. The results show that the modified algorithm always has a better performance than the previous algorithms, such as FMASS, AFMFSA, and FRFSA. In particular, the BDPSNR of the modified algorithm is always less than 0.035 dB when compared with the standard. This means that the performance of the modified algorithm is almost the same as that of the standard.

Table 4. Result of BDPSNR (dB)

QP	ΔFMASS [16]	ΔAFMFSA [17]	ΔFRFSA [18]	The Modified Algorithm
20	-0.009	-0.097	-0.110	-0.015
25	-0.043	-0.146	-0.184	-0.024
30	-0.055	-0.185	-0.232	-0.035
35	-0.027	-0.138	-0.188	-0.023
40	-0.012	-0.050	-0.082	-0.024
45	-0.018	-0.008	-0.014	-0.002
50	-0.002	-0.025	-0.032	-0.004
Avg.	-0.024	-0.103	-0.120	-0.018

Table 5. Result of BD-Bit-rate

QP	ΔFMASS [16]	ΔAFMFSA [17]	ΔFRFSA [18]	The Modified Algorithm
20	0.17%	1.93%	2.19%	0.30%
25	0.86%	2.93%	3.69%	0.48%
30	1.10%	3.70%	4.64%	0.70%
35	0.55%	2.76%	3.76%	0.47%
40	0.24%	1.01%	1.64%	0.47%
45	0.37%	0.16%	0.27%	0.04%
50	0.04%	0.50%	0.64%	0.08%
Avg.	0.48%	1.86%	2.40%	0.36%

V. ARCHITECTURE DESIGN

We designed an architecture that uses the modified fast multi-reference frame motion estimation algorithm for the H.264/AVC integer motion estimation system. As shown in the results of Section 3 above, we modified the previously proposed algorithm to design a hardware system.

1. Previous Systems

The Huang's system [9] uses five reference frames, but the Chen's system [8], the Youn's system [10] and the Kao's system [11] use only one or two reference frames and require an additional control to use the modified algorithm. The Huang's system uses only 16x16 or 8x8 block sizes. The Chen's system uses a four step search algorithm. The Youn's system uses down-sampling and pixel-truncation, which reduce the complexity of hardware. However, these cause a large performance drop. To compare the modified algorithm with the previous systems, we make software simulators of the previous systems. All simulators are based on JM

Table 6. Result of the BDPSNR (dB)

QP	Δ Chen's [9]	Δ Huang's [9]	Δ Youn's [10]	ΔEX-Youn's [10]	Δ Kao's [11]	Δ Proposed
20	-1.026	-0.162	-0.851	-0.168	-0.167	-0.015
25	-1.163	-0.146	-1.004	-0.233	-0.229	-0.024
30	-0.956	-0.092	-0.862	-0.220	-0.220	-0.035
35	-0.624	-0.020	-0.574	-0.107	-0.118	-0.023
40	-0.345	-0.010	-0.330	-0.042	-0.041	-0.024
45	-0.168	0.022	-0.145	-0.052	-0.031	-0.002
50	-0.029	0.010	-0.035	0.075	-0.081	-0.004
Avg.	-0.616	-0.057	-0.543	-0.107	-0.127	-0.018

Table 7. Result of BD-Bit-rate

QP	Δ Chen's [9]	Δ Huang's [9]	Δ Youn's [10]	ΔEX-Youn's [10]	Δ Kao's [11]	Δ Proposed
20	20.52%	3.24%	17.02%	3.36%	3.34%	0.30%
25	23.27%	2.91%	20.08%	4.66%	4.59%	0.48%
30	19.12%	1.83%	17.25%	4.41%	4.40%	0.70%
35	12.47%	0.41%	11.48%	2.14%	2.36%	0.47%
40	6.91%	0.19%	6.61%	0.85%	0.82%	0.47%
45	3.37%	-0.44%	2.89%	1.04%	0.62%	0.04%
50	0.57%	-0.19%	0.69%	-1.50%	1.62%	0.08%
Avg.	12.32%	1.14%	10.86%	2.14%	2.53%	0.36%

v9.6 reference software. Table 6 and 7 represent the difference values of BDPSNR and BD-bit-rate when compared with the standard, which uses five reference frames and the 64x64 search area. The EX-Youn's system is the extension version of the Youn's system which uses 2 reference frames. Simulation environments are the same as previously described. The results show that the proposed system, which uses the modified algorithm, usually has a better performance than the other systems.

2. The Overall Architecture

Fig. 6 shows our overall architecture for the proposed system. Our proposed fast algorithm processes a full search in all reference frames. The search area size of ref_0 and ref_1 is full size, and that of ref_2, ref_3 and ref_4 is reduced size. For Full HD (1920x1080P) video encoding, our architecture consists of search area memory, a current macro block memory, 16 processing units (PU), a SAD summation block, a comparison block, and a center point generator. The current macro block memory saves the current macro block pixels, the size of which is 256 (16x16) bytes. Because the search area of the proposed system is 64x64, the size of one line memory is 79 bytes and the number of line memories is 79.

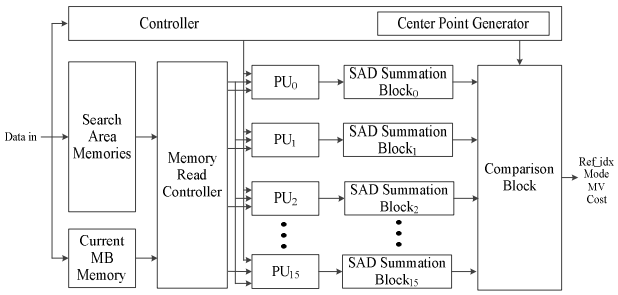

Fig. 6. The overall architecture of the proposed system.

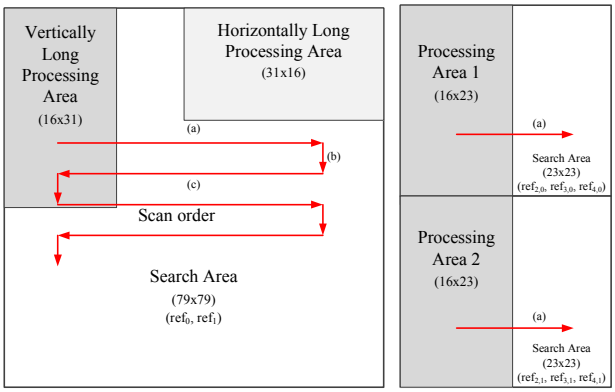

Fig. 7. The scan order of search area memories.

3. The Memory Read Controller

To minimize the time to read the memory, we proposed the processing area which has a shape of vertically long rectangular. The processing area is the region of search area to be calculated immediately. For reducing external memory bandwidth, the motion estimation of the two reduced search areas is performed at the same time in ref_2, ref_3 and ref_4. Fig. 7 shows the scan order of search area memories. The scan direction of the reduced search area is only (a), but that of the full search area can be (a), (b) and (c). If the processing area is horizontally long rectangular same as most cases, the size of the processing area is 31x16. In this case, each line memory output has to be only one byte at one clock cycle when the scan order is the direction (a) or (c) but the last search area line memory output has to be 31

bytes at one clock cycle when scan order is the direction (b). The almost CMOS technology does not supports that the bit-width of the SRAM is 31 bytes. But our system uses the vertically long rectangular read area. It makes the line memory output to be only 1 byte when the scan order is the direction (a) or (c), too. Additionally, the last search area line memory output is only 16 bytes at one clock cycle when scan order is the direction (b).

4. Processing Units (PU)

The PU calculates the 4x4 unit SAD values of the 16x16 macro block. One PU calculates 16 4x4 unit SAD values between the current block pixels and the reference block pixels. For real-time full HD video encoding, we use 16 PU in the proposed integer motion estimation system.

5. SAD Summation Blocks and the Comparison Block

The SAD summation blocks calculate the SAD values of each mode by using 4x4 unit SAD values, which are calculated in the PU. The inputs of one SAD summation block are 16 4x4 SAD values, and its outputs are 41 SAD values for the 7 block size in H.264/AVC: one 16x16, two 16x8, two 8x16, four 8x8, eight 8x4, eight 4x8, and sixteen 4x4. For real-time full HD video encoding, we use 16 SAD summation blocks in the proposed integer motion estimation system.

The comparison block calculates the cost value of each mode and determines the final mode and determines the final mode, which has a minimum cost value. The comparison block consists of 41 cost calculators and 41 comparison units.

6. The Center Point Generator

To design the center point generator, we modify Eq. (1) in Section 2. The modified Eq. (4) can be implemented by using ROM, several multipliers, and adders. Additionally, because Eq. (4) is similar to Eq. (3), the proposed reduced search area center position generator can be shared with the temporal direct mode predictor. Fig. 8 is the proposed search area center position generator. It consists of two multipliers, three adders, and ROM that has 128 bit-depth and 15 bit-width.

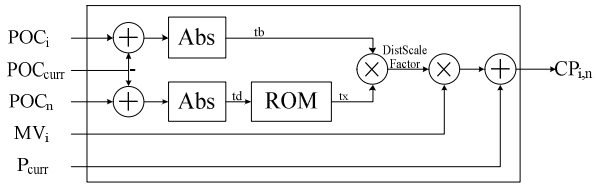

Fig. 8. The center point generator.

It is designed to solve for the motion vector in the temporal direct mode.

7. The Pipeline Process

For the pipeline process, the proposed system is divided into six stages, which are memory reading, PU, SAD summation 1, SAD summation 2, solving cost 1 and solving cost 2. All stages are designed to use a one clock cycle. The CU uses two clock cycles when the calculations of the all search area are finished. Fig. 9 shows the pipeline process of the proposed system. It requires 263 clock cycles to process motion estimation of macro block in one reference frame. Because the maximum number of the proposed algorithm's reference frames is 2.0938, motion estimation with the modified algorithm uses 550 clock cycles. The minimum required clock of Full HD (1920x1080P@30Hz) video encoding is 135 MHz.

VI. SYNTHESIS RESULTS

The proposed system is implemented in Verilog HDL and synthesized by using 0.13 um CMOS technology. Table 8 contains the synthesis results of the proposed system. The gate count is 1,089K with 6,497 bytes of SRAM. The proposed system uses five reference frames. The search area of ref_0 and ref_1 is 64x64. In ref_2, ref_3, and ref_4, the search area of each frame consists of two reduced search areas, and each reduced search area size is 8x8. The search algorithm in each search area is a full search algorithm. The operation frequency is 135 MHz when encoding a Full-HD (1920x1080p@30Hz) sized video. Table 9 contains the comparison of the proposed system with previous integer motion estimation systems. The performance of the Youn's system which uses 5 reference frames is almost same with that of the standard which uses 5 reference frames. However, the controller

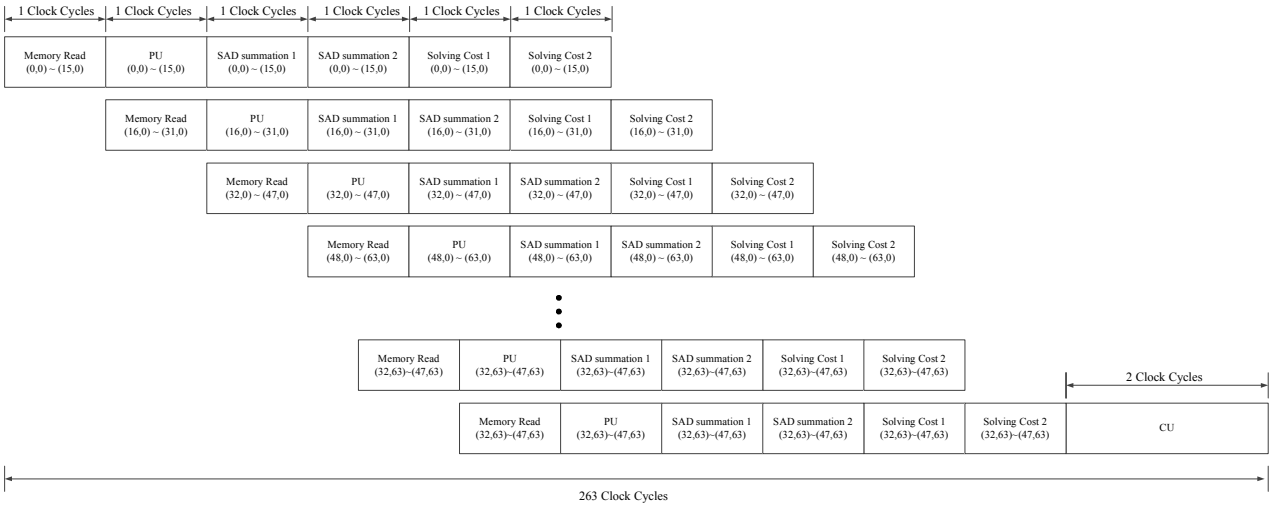

Fig. 9. The pipeline process of the proposed system.

Table 8. Synthesis results

Technology	0.13um CMOS
Gate count	1089K
SRAM	6.5 KB
Video Specification	1920x1080P@30Hz
Operating Frequency	135MHz
Search Area	64x64
Reference Frames	5

Table 9. The comparison of the proposed system with previous integer motion estimation systems

	Chen's [8]	Huang's [9]	Youn's [10]	EX-Youn's [10]	Kao's [11]	Proposed
Process (um)	0.18 CMOS	0.18 CMOS	0.18 CMOS	0.18 CMOS	0.18 CMOS	0.13 CMOS
Gate count	708K	94.3K	498K	≈ 900K	1449K	1089K
SRAM	8KB	-	1.44KB	≈ 2.88KB	2.97KB	6.50KB
Operating Frequency (CIF@30Hz)	13.5MHz	25 MHz	6.75 MHz	6.75MHz	6.35 MHz	6.53 MHz
Search Algorithm	4 Step Search	Full Search	Full Search	Full Search	Full Search	Full Search
Block Size	16x16 to 4x4	16x16 to 8x8	16x16 to 4x4	16x16 to 4x4	16x16 to 4x4	16x16 to 4x4
Search Area	64x32	128x64	128x64	128x64	64x64	64x64 (ref_0, ref_1) Two 8x8 (ref_2, ref_3, ref_4)
Reference Frames	1	5	1	2	2	5
Full HD (1920x1080p@30Hz)	No	No	Yes (138MHz with 1 reference frame)	Yes (138MHz with 2 reference frame)	Yes (130MHz with 2 reference frames)	Yes (135MHz with 5 reference frames)
BD-Bit-rate	+12.32 %	+1.14 %	+10.86 %	+2.14 %	+2.53 %	+0.36 %

part is only hardware resource which can be shared when extend reference frames to 5 without increasing the operation clock in design of motion estimator. Therefore, the gate count of the Youn's system which uses 5 reference frames is 4 times more than that of the Youn's system which uses 1 reference frame. Otherwise, the

operation clock is 5 times more than that of the Youn's system. In other words, the Youn's system which uses 5 reference frames needs more than 2000K gate count or faster than 500 MHz operation clock. Because of that, we except the Youn's system which uses 5 reference frames from the comparison group. Instead, because the gate count of the EX-Youn's system which uses 2 reference frames is almost same with our system, it is added to the comparison group. The gate count and SRAM size of the EX-Youn's system are estimated. Because the shared hardware resource is very small, the gate count of the EX-Youn's system is more than 900K and the SRAM size of the EX-Youn's system is approximately 2.88 KB. The proposed system supports Full HD real time encoding with 5 reference frames and has the highest performance.Consequently, the efficient pipeline structure of the proposed system and the proposed fast multi-reference motion estimation algorithm make it possible to encode a Full-HD video with 5 reference frames, high perfor- mance, and acceptable operation clock speed in real time.

This paper presents a hardware design for the multi-reference integer motion estimation system by using a modified fast multi-reference motion estimation algorithm. The previously proposed algorithm uses a linear relationship between the motion vector and the distances from the current frame to the reference frames along the time axis. The modified algorithm reduces the overlapping search area and the size of the reduced search area, and modifies a division equation. The modified algorithm executes full search area motion estimation in ref_0 and ref_1. In ref_2, ref_3, and ref_4, motion estimation is executed in two squares of 1/64 of the search area surrounding of each center position. Because the search area is reduced, the amount of computation is reduced by 58.7%. In terms of the experimental results, the modified algorithm shows a 0.36% bit-rate increase when compared with the five reference frame standard. The simulation result shows that the performance of the proposed system is better than that of FMASS [16], AFMFSA [17], FRFSA [18], the Huang's system [9], the Youn's system [10], the extension version of the Youn's system, and the Kao's system [11].

We propose an efficient pipeline structure and memory read controller. The proposed fast multi-reference integer motion estimation system uses 0.13 um CMOS technology, the gate count of which is 1089K with 6.50 KB SRAM. It can encode a full HD video with 5 reference frames in real time at a 135 MHz clock speed.

ACKNOWLEDGMENTS

This work was supported by the IT R&D program of MOTIE/KEIT. [10035389, Research on high speed and low power wireless communication SoC for high resolution video information mining]

REFERENCES

[1] J. V. Team, Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification. ITU-T Rec. H.264 and ISO/IEC 14496-10 AVC.

[2] S. H. Lee and H. J. Lee, "A Pipelined Hardware Architecture of an H.264 Deblocking Filter with an Efficient Data Distribution", *Journal of Semiconductor Technology and Science*, vol. 6, no. 4, pp. 227-233, Dec. 2006.

[3] L. Yang, K. Yu, J. Li, and S. Li, "An effective variable block - size early termination algorithm for H.264 video coding", *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 15, no. 6, pp. 784-788, June 2005.

[4] N. Hirai, T. Song, Y. Liu, and T.Shimamoto, "A Novel Spiral-Type Motion Estimation Architecture for H.264/AVC", *Journal of Semiconductor Technology and Science*, vol. 10, no. 1, pp. 37-44, Mar. 2010.

[5] W. Lee, Y. Jung, S. Lee, and J. Kim, "High speed intra prediction scheme for H.264/AVC" , *IEEE Trans. Consumer Electronics*, vol. 53, issue 4, pp1577-1582, Nov. 2007.

[6] T. C. Chen, S. Y. Chien, Y. W. Huang, C H. Tsai, C. Y. Chen, T. W. Chen, and L. G. Chen, "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, June 2006.

[7] J. Miyakoshi, Y. Kuroda, M. Miyama, K. Imamura, H. Hashimoto, and M. Yoshimoto, "A sub-mW MPEG-4 motion estimation processor core for mobile video application," *in Proceedings of the*

Custom Integrated Circuits Conference 2003, San Jose, The U.S.A., pp. 181−184, Sept. 2003.

[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. on Circuits and Systems for Video Technology*, vol. 17, no. 5, pp. 568–576, May 2007.

[9] Y. Huang, Z. Liu, Y. Song, S. Goto, and T. Ikenaga, "Parallel improved HDTV720p targeted propagate partial SAD architecture for variable block size motion estimation in H.264/AVC", *IEICE Trans. Fundamentals*, vol.E91-A, no.4, pp.987-997, April 2008.

[10] J. S. Youn and J. R. Choi, "Implementation of parallel integer motion estimation method by using reference blocks shared for HD video encoding", *IEICE Electronics Express*, vol. 8, no. 17, pp.1380-1386, September 2011.

[11] C. Y. Kao and Y. L. Lin, "A memory-efficient and highly parallel architecture for variable block size integer motion estimation in H.264/AVC", *IEEE Trans. Very Large Scale Integration Systems*, vol. 18, no. 6, pp. 866-874, June 2010.

[12] J. Byun, J. Choi, and J. Kim, "A fast multi-reference frame motion estimation algorithm", *IEEE Trans. Consumer Electronics*, vol 56, issue 3, pp.1911-1917, August 2010.

[13] T. Wiegand, X. Zhang, and B. Girod, "Block-based hybrid video coding using motion-compensated long-term memory prediction," *Proc. Picture Coding Symp.* Berlin, The Germany, Sept. 1997.

[14] T. Wiegand, X. Zhang, and B. Girod, "Long-term memory motion-compensated prediction," *IEEE Trans. Circuit and Systems for Video Technology*, vol. 9, pp. 70-84, February 1999.

[15] Y. W. Huang, B. Y. Hsieh, T. C. Wang, S. Y. Chien, S. Y. Ma, C. F. Shen, and L. G.. Chen, "Analysis and reduction of reference frames for motion estimation in MPEG-4 AVC/JVT/H.264", *ICASSP'03*, vol. 2, pp.809-812, Hong Kong, The China, Apr. 2003.

[16] X. Li, Li, E.Q., and Y. K. Chen, "Fast multi-frame motion estimation algorithm with adaptive search strategies in H.264", *ICASSP'04*, vol. 3, pp 369-372, Quebec, The Canada, May 2004.

[17] L. Shen, Z. Liu, Z. Zhang, X. Shi, "An adaptive and fast multiframe selection algorithm for H.264 video coding", *IEEE Signal Processing Letters*, vol. 14, no. 11, PP 836-839, Nov. 2007.

[18] Kangjun Lee, Gwangil Jeon, and Jechang Jeong, "Fast reference frame selection algorithm for H.264/AVC", *IEEE Trans. Consumer Electronics*, vol 55, issue 2, pp.773-779, August 2009.

[19] Bjontegaard, G.: Recommended Simulation Condition for H.26L, ITU-T Q6/SG16, Doc. #VCEG-L38, 9-12 Jan, 2001.

Juwon Byun received B.S. and M.S. degrees in electrical and electronic engineering from Yonsei University, Seoul, Korea, in 2007 and 2009, respectively, and is currently pursuing his Ph.D. degree. His research interests include algorithms and SoC design for video encoder/decoder.

Jaeseok Kim received a B.S. degree in electronic engineering from Yonsei University, Seoul, Korea in 1977, M.S. degree in electrical and electronic engineering from KAIST, Daejon, Korea in 1979, and Ph.D. degree in electronic engineering from RPI, NY, USA in 1988. From 1988 to 1993, he was a member of the technical staff at AT&T Bell Labs, USA. He was Director of the VLSI Architecture Design Lab of ETRI from 1993 to 1996. He is currently a professor in the electrical and electronic engineering department at Yonsei University, Seoul, Korea. His current research interests include communication IC design, high performance Digital Signal Processor VLSI design, and CAD S/W.