New Motion Estimation Algorithm Using Adaptively Quantized Low Bit-Resolution Image and Its VLSI Architecture for MPEG2 Video Encoding

Seongsoo Lee, Jeong-Min Kim, and Soo-Ik Chae, *Member, IEEE*

***Abstract*—This paper describes a new motion estimation algorithm that is suitable for hardware implementation and substantially reduces the hardware cost by using a low bit-resolution image in the block matching. In the low bit-resolution image generation, adaptive quantization is employed to reduce the bit resolution of the pixel values, which is better than simple truncation of the least significant bits in preserving the dynamic range of the pixel values. The proposed algorithm consists of two search steps: in the low-resolution search, a set of candidate motion vectors is determined, and in the full-resolution search, the motion vector is found from these candidate motion vectors. The hardware cost of the proposed algorithm is 1/17 times of the full search algorithm, while its peak signal-to-noise ratio is better than that of the 4 : 1 alternate subsampling for the search range of $\pm 32 \times \pm 32$. A VLSI architecture of the proposed algorithm is also described, which can concurrently perform two prediction modes of the MPEG2 video standard with the search range of $(-32.0,-32.0)$–$(+31.5,+31.5)$. We fabricated a MPEG2 motion estimator with a 0.5-μm triple-metal CMOS technology. The VLSI chip includes 110 K gates of random logic and 90 K bits of SRAM in a die size of 11.5 mm $\times$ 12.5 mm. The full functionality of the fabricated chip was confirmed with an MPEG2 encoder chip.**

***Index Terms*— Adaptive quantization block matching, low-resolution search, motion estimation, VLSI.**

I. Introduction

MOTION estimation, which eliminates the temporal redundancy of the image sequences, is widely used in the video coding algorithm [1], [2]. In general, there are two major types of motion estimation algorithms: the block-matching algorithm [3] and the pel-recursive algorithm [4]. The former seems to be better in both performance and hardware cost [5]. For each reference block in the current frame, the block-matching algorithm searches for a best matched block in the previous frame. The motion vector is defined as a displacement between the reference block and the best matched block.

The full search algorithm [3] exhaustively matches all possible candidate blocks to find a motion vector. Regarded as the optimum solution for obtaining a high compression ratio [6], this method suffers from a tremendous hardware cost. It becomes impractical for hardware implementation if the search range is large [7]. Therefore, many fast algorithms have been proposed [8]–[15] to reduce this hardware cost. Among them, some algorithms [13]–[15] reduce the bit resolution of the pixel values by truncating the least significant bits (LSB) of the pixel values because the matching operation for low bit-resolution images requires less hardware than that for full bit-resolution images.

Although the bit resolution can be easily reduced with simple LSB truncation, this approach suffers from performance degradation because the dynamic range of the pixel values decreases. Therefore, we proposed the low-resolution quantization motion estimation (LRQME) algorithm [16] that is suitable for VLSI implementation. In the proposed algorithm, we employed adaptive quantization to reduce performance degradation and a novel block-matching criterion to reduce the hardware cost.

In this paper, we propose a hardware architecture for the proposed algorithm. It employs a novel processing element (PE) architecture that efficiently reduces the amount of hardware. In Section II, the proposed algorithm is briefly described, and is compared with the conventional block-matching algorithms. In Section III, the hardware architecture for the proposed algorithm is presented in detail. The VLSI design of the proposed architecture is also described. Finally, Section IV concludes this paper.

II. Low-Resolution Quantization Motion Estimation (LRQME)

A. Adaptive Quantization

In the block-matching algorithms using LSB truncation [13]–[15], severe performance degradation is observed, although they do not require additional hardware for bit-resolution reduction. To overcome this drawback, we employed adaptive quantization in the bit-resolution reduction. For each pixel in both the reference block and the search window, the reference block mean is subtracted from its value, and the result is quantized into a 2-bit code. In preserving the dynamic range of the pixel value, this approach is better than the LSB truncation methods. The scheme of the low resolution image generation is shown in Fig. 1.

We selected 2-bit resolution based on the tradeoff between the performance and the hardware cost, whose simulation results are described in Section II-D. For each reference block,

Manuscript received May 16, 1997; revised April 20, 1998. This paper was recommended by Associate Editor K. N. Ngan.

The authors are with the School of Electrical Engineering and Inter-University Semiconductor Research Center, Seoul National University, Kwanak-ku, 151-742, Seoul, Korea.

Publisher Item Identifier S 1051-8215(98)07877-X.

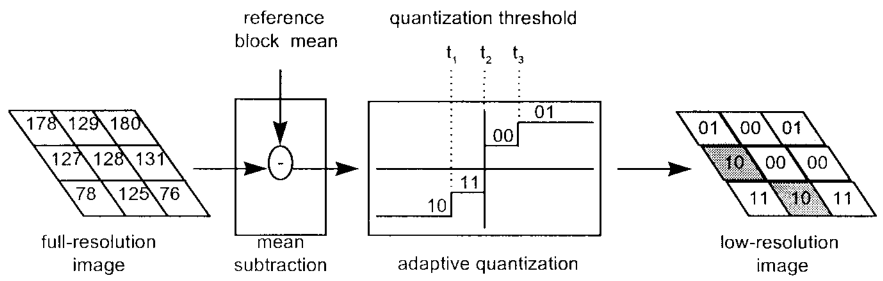


Fig. 1. Generation of low-resolution images.

TABLE I
AVERAGE CORRELATION COEFFICIENTS BETWEEN THE ORIGINAL AND THE REDUCED-BIT IMAGES

Sequences	CIF					CCIR601				
	Car Phone	Claire	Foreman	Miss America	average	Flower Garden	Football	Popple	Table Tennis	average
LSB truncation to 2-bit resolution	0.56	0.30	0.67	0.27	0.45	0.69	0.60	0.65	0.68	0.66
adaptive quantization to 2-bit resolution	0.86	0.72	0.86	0.77	0.80	0.87	0.89	0.83	0.88	0.87

TABLE II
NUMBER OF SEARCH POSITIONS THAT HAS THE MINIMUM VALUE OF THE BLOCK-MATCHING CRITERION

CIF			CCIR601		
number of search positions that has the minimum value of the block-matching criterion	LSB truncation to 2-bit resolution	adaptive quantization to 2-bit resolution	number of search positions that has the minimum value of the block-matching criterion	LSB truncation to 2-bit resolution	adaptive quantization to 2-bit resolution
1	47.78 %	91.96 %	1	75.52 %	92.91 %
2~32	10.81 %	7.97 %	2~512	17.48 %	7.09 %
33~64	0.83 %	0.06 %	513~1024	1.23 %	0 %
65~96	0.90 %	0 %	1025~1536	1.36 %	0 %
97~128	1.65 %	0 %	1537~2048	1.47 %	0 %
129~160	2.17 %	0 %	2049~2560	0.99 %	0 %
161~192	2.27 %	0 %	2561~3072	0.78 %	0 %
193~224	2.28 %	0 %	3073~3584	0.51 %	0 %
225~255	3.28 %	0 %	3585~4095	0.47 %	0 %
block-matching criterion is constant	28.03 %	0.01 %	block-matching criterion is constant	0.19 %	0 %

the quantization thresholds t_1, t_2, and t_3 are determined adaptively with (1) to minimize the average quantization error:

$$t_3 = -t_1 = \frac{3}{2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}|p(i,j) - \overline{p}|, \qquad t_2 = 0 \qquad (1)$$

where $p(i,j)$ is the pixel value of the reference block, $\overline{p}$ is the reference block mean, and the reference block size is $N \times N$. The thresholds in (1) are determined based on performance and hardware cost through experimental search.

In this paper, 40 frames of four CIF (352 pels $\times$ 288 pels, 30 frames/s, 8 bits/pixel) sequences and 40 frames of four CCIR601 (720 pels $\times$ 480 pels, 30 frames/s, 8 bits/pixel) sequences are used in all simulations. The size of the reference block is 16 $\times$ 16. The search ranges are $\pm 32 \times \pm 32$ in the case of CCIR601 sequences and $\pm 8 \times \pm 8$ in the case of CIF sequences. From the correlation coefficients between the original and the reduced-bit pixel values, shown in Table I, we concluded that the adaptive quantization method is more efficient than the LSB truncation method.

If the minimum value of the block-matching criterion occurs on two or more search positions, only one of these positions should be determined as the motion vector, which often results in a suboptimal solution. If all pixels in the search window are reduced into the same value in the bit-resolution reduction, the block-matching criterion is constant over the whole search range. In this extreme case, the correct motion vector cannot be found properly. Hereafter, we will refer to it as a tie-score problem.

Table II shows the number of search positions that have the minimum value of the block-matching criterion for each reference block. To make a fair comparison, the full search algorithm and 2-bit sum of absolute difference (SAD) are used

TABLE III
PROBABILITY TO FIND THE CORRECT MOTION VECTOR

Sequences	40 frames of CIF (352x288, 30Hz) Search Range = ±8x ±8				40 frames of CCIR601 (720x480, 30Hz) Search Range = ±32x32			
	Car Phone	Claire	Foreman	Miss America	Flower Garden	Football	Popple	Table Tennis
LSB truncation to 2-bit resolution	34.6%	21.0%	48.2%	20.2%	60.1%	21.5%	23.0%	70.2%
adaptive quantization to 2-bit resolution	83.1%	92.9%	91.7%	64.6%	86.4%	53.7%	40.3%	88.6%

in two methods of reducing bit resolution. The LSB truncation method suffers more than the adaptive quantization from the tie-score problem. Note that in the LSB truncation method, the block-matching criterion is constant over the whole search range for 28% of the reference block in the CIF sequences. This is because the CIF sequences have large background regions without edges and complex patterns where, with high possibility, all pixels in the search window are reduced into the same value in the bit-resolution reduction.

Table III shows the probability of finding the correct motion vector in two methods of reducing bit resolution so that the motion vector obtained with each method is equal to that of 8-bit SAD. As in Table II, the full search algorithm and the 2-bit SAD are used for both the LSB truncation method and the adaptive quantization method. From Table III, the adaptive quantization method has lower probability to be trapped in the local minima than the LSB truncation method because the correct motion vector is the global minimum of the error surface.

B. Block-Matching Criterion

The performance and required hardware amount of a motion estimation algorithm largely depend on its block-matching criterion. The SAD [3] is widely used in most of the conventional motion estimation algorithms because of its efficiency and simplicity, but its hardware cost is rather high for VLSI implementation if the search range is large [7]. Therefore, several block-matching criteria have been proposed to reduce this hardware cost, which are defined as follows.

1) *Pel Difference Classification (PDC) [11]:*

$$\mathrm{PDC}(u,v) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} T_{u,v}(i,j)$$

$$T_{u,v}(i,j) = 1, \qquad \text{if } |p_k(i,j) - p_{k-1}(i+u,j+v)| \le \text{Threshold, 0 otherwise.}$$

2) *Bit Truncation (BT) [13], [14]:*

$$\mathrm{BT}_m(u,v) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} |p_k(i,j)_{7:m} - p_{k-1}(i+u,j+v)_{7:m}|.$$

3) *Bit Exclusive-OR (BXOR) [15]:*

$$\mathrm{BXOR}_m(u,v) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} XOR \cdot [p_k(i,j)_{7:m}, p_{k-1}(i+u,j+v)_{7:m}]$$

where $p_k(x,y)$ is a pixel value in the kth frame, m is the number of truncated LSB bits, $p_k(x,y)_{7:m}$ is the (8-m) most significant bits of a pixel value in the kth frame, $XOR(x,y)$ is the bitwise exclusive-OR of x and y, and the reference block size is $N \times N$.

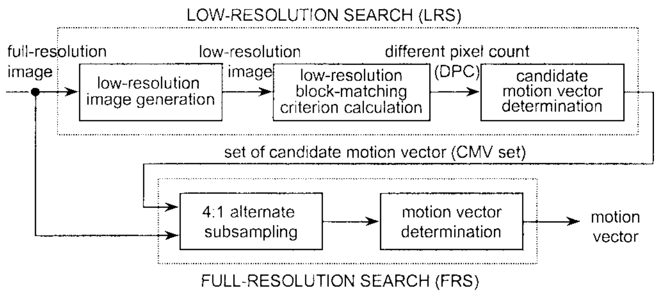

LOW-RESOLUTION SEARCH (LRS)
full-resolution image
low-resolution image generation
low-resolution image
low-resolution block-matching criterion calculation
different pixel count (DPC)
candidate motion vector determination
set of candidate motion vector (CMV set)
4:1 alternate subsampling
motion vector determination
motion vector
FULL-RESOLUTION SEARCH (FRS)

Fig. 2. Block diagram of the LRQME algorithm.

In this paper, the different pixel count (DPC) defined in (2) is proposed as a block-matching criterion for low bit-resolution images. It is equal to the number of pixels whose quantized codes are unmatched in a block. It is a special case of the PDC because the PDC with zero threshold is the number of matched pixels

$$\mathrm{DPC}(u,v) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} \overline{\delta}[\hat{p}_k(i,j), \hat{p}_{k-1}(i+u,j+v)] \quad (2)$$

where $\hat{p}_k(x,y)$ is the 2-bit quantized code of a pixel value in the kth frame, $\overline{\delta}(x,y) = 1$ when $x \neq y$ and 0 otherwise, and the reference block size is $N \times N$.

The DPC can be implemented by fewer logic gates than the BT and the PDC by employing a novel processing element (PE). In Section III-B, the processing element and the hardware reduction of the DPC are described in detail.

C. LRQME Algorithm

We proposed the low-resolution quantization motion estimation (LRQME) algorithm [16] which employs two search steps, namely, low-resolution search (LRS), and full-resolution search (FRS). The block diagram of the LRQME algorithm is illustrated in Fig. 2.

In a low-resolution search, a candidate motion vector set (CMV set) is determined by calculating the different pixel count. In a full-resolution search, the motion vector is found by calculating the sum of absolute difference on the positions of the CMV set.

In the hardware realization, the LRS and the FRS are pipelined by every two rows of search positions, which is described in Section III-A. For this reason, four CMV's are found in every two rows of search positions, and 128 CMV's are found for each reference block when the search range is $\pm32 \times \pm32$. Simulation results for the number of the CMV's are shown in Section II-D. To reduce the hardware cost further, a 4 : 1 alternate subsampling algorithm [8] is used in the FRS. Low-pass filtering is not used in any algorithms whose simulation results are presented in this paper.

The outline of the LRQME algorithm is as follows.

Low-Resolution Search:

1) For each reference block, the block mean and the quantization thresholds are calculated.
2) The block mean is subtracted from each pixel in both the reference block and the search window, and the result is quantized into 2-bit resolution.
3) For every search position, its DPC is calculated for low-resolution images.
4) Four search positions with smaller DPC are selected as the CMV's for every two rows of search positions.

Full-Resolution Search:

1) For each CMV, a 4 : 1 alternate subsampling search is performed, and the SAD is calculated using full-resolution images.
2) The search position with minimum SAD is determined as the motion vector.

D. Simulation Results

The amount of hardware required for the DPC calculation is substantially smaller than that for the SAD calculation. Therefore, the number of operations or the number of search positions is not a good measure of the hardware cost. In fact, the total gate count of the motion estimator with the same throughput is a better measure, but it is difficult to estimate the total gate count unless the design of the motion estimator is completed.

However, the total gate count of all processing elements is easy to calculate, and is a good measure of the algorithm complexity for hardware implementation. We define the hardware cost as (total NAND-equivalent gate count of all processing elements) $\times$ (required cycle time per reference block). The second term is used to equalize the throughput of the algorithm. The peak signal-to-noise ratio (PSNR) is used as the performance of the algorithm. In the hardware cost of the proposed algorithm, the cost of all preprocessing steps such as quantization threshold determination, mean subtraction, and quantization is included.

The simulation results used to determine the bit resolution of the quantization and the number of CMV's are shown in Figs. 3 and 4. 2-bit resolution and 128 CMV's were selected based on the simulation results.

Six motion estimation algorithms, such as the full search (FS) algorithm [3], the 4 : 1 alternate subsampling (4 : 1AS) algorithm [8], the one-dimensional full search (1DFS) algorithm [10], the bit truncation (BT) algorithm [13], [14], the bit exclusive-OR (BXOR) algorithm [15], and the proposed algorithm (LRQME) are compared in Table IV and Figs. 5–7.

Table IV shows their PSNR performances, which are based on the block difference. A comparison of their hardware costs and the average PSNR performances is shown in Fig. 5.

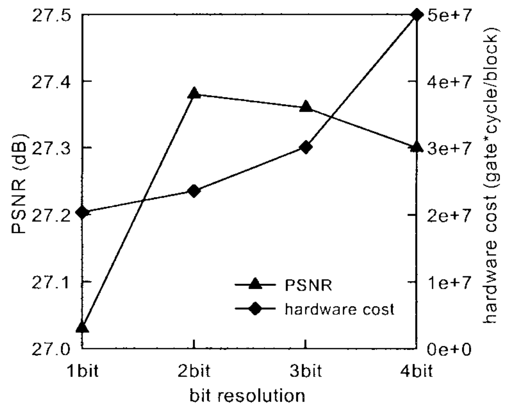


Fig. 3. Hardware cost and PSNR versus bit resolution.

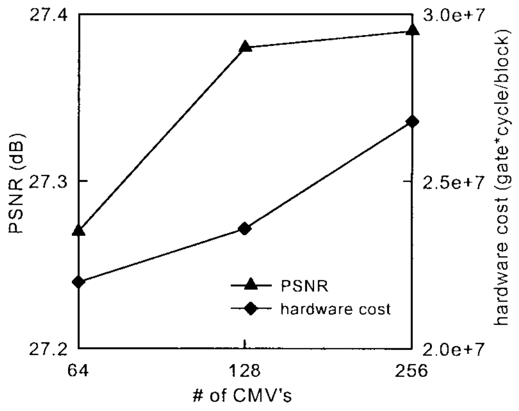


Fig. 4. Hardware cost and PSNR versus number of CMV's.

Note that the proposed algorithm is superior to four other fast algorithms in both the hardware costs and the PSNR performances. Compared with the full search algorithm, the hardware cost of the proposed algorithm is 1/17 and 1/10 times while its PSNR degradation is less than 0.37 and 0.12 dB in the case of CCIR601 and CIF sequences, respectively.

Figs. 6 and 7 show the PSNR performances in the case of the MPEG2 [2] P-prediction and H.261 [1] encoding, respectively. Note that the PSNR degradation of the proposed algorithm is smaller than those of four other fast algorithms for all bit rates.

III. The Proposed Architecture

A. Overall Architecture

Fig. 8 shows the block diagram of the proposed architecture. It consists of four main blocks with five internal buffers: the preprocessing unit (PPU), the low-resolution search (LRS) unit, the full-resolution search (FRS) unit, and the half-pel search (HPS) unit.

The PPU calculates the reference block mean and determines quantization thresholds. The LRS unit generates a low-resolution image by quantization, calculates the DPC in the whole search range (4096 search positions), and determines the CMV set (128 search positions). The FRS unit determines four search positions using the 4 : 1 alternate subsampling algorithm on the position in the CMV set. It performs the two

TABLE IV
PSNR PERFORMANCES

Sequences	CIF					CCIR601				
	Car Phone	Claire	Foreman	Miss America	average	Flower Garden	Football	Popple	Table Tennis	average
FS	33.68 dB	42.67 dB	32.52 dB	39.18 dB	37.01 dB	27.03 dB	25.11 dB	30.01 dB	28.26 dB	27.60 dB
4:1AS	33.61 dB	42.60 dB	32.46 dB	39.06 dB	36.93 dB	26.96 dB	24.65 dB	29.77 dB	28.04 dB	27.35 dB
1DFS	32.93 dB	42.48 dB	31.87 dB	38.98 dB	36.57 dB	26.66 dB	23.67 dB	29.76 dB	27.29 dB	26.84 dB
BT	31.45 dB	35.20 dB	30.85 dB	36.74 dB	33.56 dB	26.66 dB	23.82 dB	28.40 dB	27.54 dB	26.47 dB
BXOR	31.20 dB	35.17 dB	30.66 dB	36.69 dB	33.43 dB	25.93 dB	23.37 dB	28.38 dB	27.32 dB	26.25 dB
LRQME	33.56 dB	42.64 dB	32.46 dB	39.09 dB	36.94 dB	26.93 dB	24.74 dB	29.78 dB	28.07 dB	27.38 dB

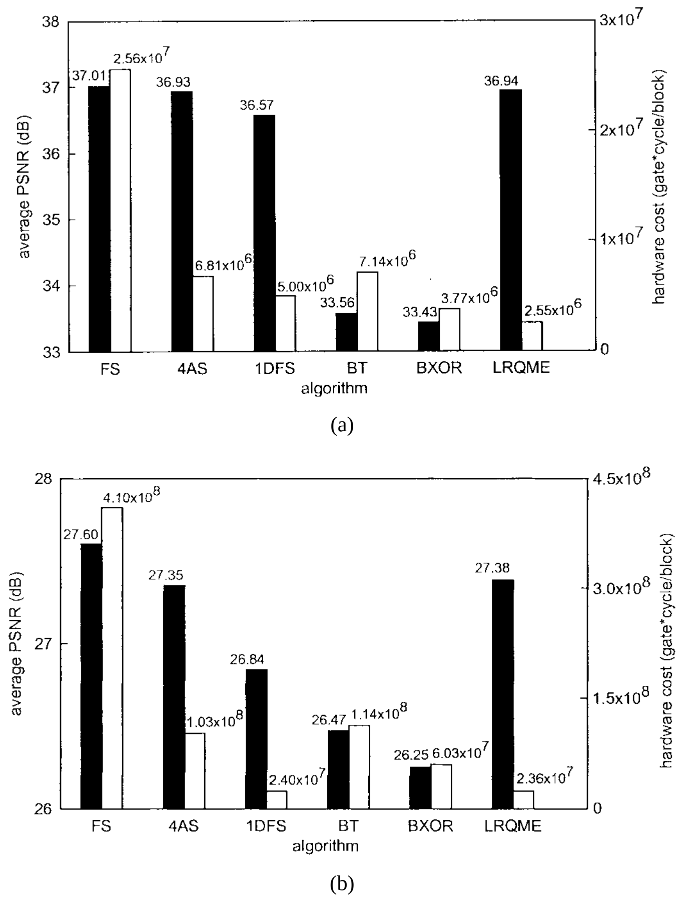


Fig. 5. Average PSNR and the hardware cost. (a) CIF and (b) CCIR 601.

prediction modes concurrently on these four search positions, and determines the integer-pel precision motion vector (IMV). The HPS unit determines the half-pel precision motion vector (HMV).

In general, memory access bottlenecks in motion estimation are a serious problem for real-time MPEG2 video encoding. To reduce this memory bandwidth, both the search window data and the reference block data are stored in the internal SRAM buffer for data reuse. There are five internal buffers in the proposed architecture: a previous search window (PSW) buffer, a previous reference block (PRB) buffer, a current search window (CSW) buffer, a current reference block (CRB) buffer, and a next reference block (NRB) buffer.

The search windows of the adjacent reference blocks overlap by 6144 (64 pels $\times$ 48 pels $\times$ 2 fields) pels. As a result, 7680 (80 pels $\times$ 48 pels $\times$ 2 fields) pels in the search window of the current reference block, and 1536 (16 pels $\times$ 48 pels $\times$ 2 fields) nonoverlapped pels in the search window of the next reference block, are stored in the CSW buffer. The widths and depths of the internal SRAM buffers are optimized for data transfer between the buffers and the processing units.

The CSW and CRB buffers are split into four banks because the 4 : 1 alternate subsampling algorithm is used in the FRS unit. Assuming that the cycle time of the frame memory is slower than that of the proposed architecture, the pixel data in the frame memory are accessed every two clock cycles and stored in the internal buffers.

The pipelining stage of the proposed architecture is illustrated in Fig. 9. The PPU, the LRS/FRS unit, and the HPS unit are pipelined for each reference block. In the second stage, the LRS unit and the FRS unit are pipelined for every two rows of search positions because the LRS unit determines four CMV's for every two row of search positions. When the LRS unit and the FRS unit are processing the Kth reference block, the $(K+1)$th reference block data are transferred into the NRB buffer and the nonoverlapped $(K+1)$th search window data are transferred into the CSW buffer.

B. Low-Resolution Search Unit

Fig. 10 shows the block diagram of the LRS unit. This consists of four main blocks: quantizers to generate the low-resolution images, low-resolution image (LRI) registers to store the low-resolution images, PE array to calculate the DPC, and comparators to determine the CMV set.

The quantizers (Fig. 11) subtract the reference block mean from the pixel values, and quantize the results to generate the low-resolution images. Twenty quantizers are required for 16 pixels in the search window and four pixels in the reference block.

The LRI registers are 2-bit word-width shift registers which provide low-resolution images to the PE array. Three LRI registers are employed in the LRS unit. Since two rows

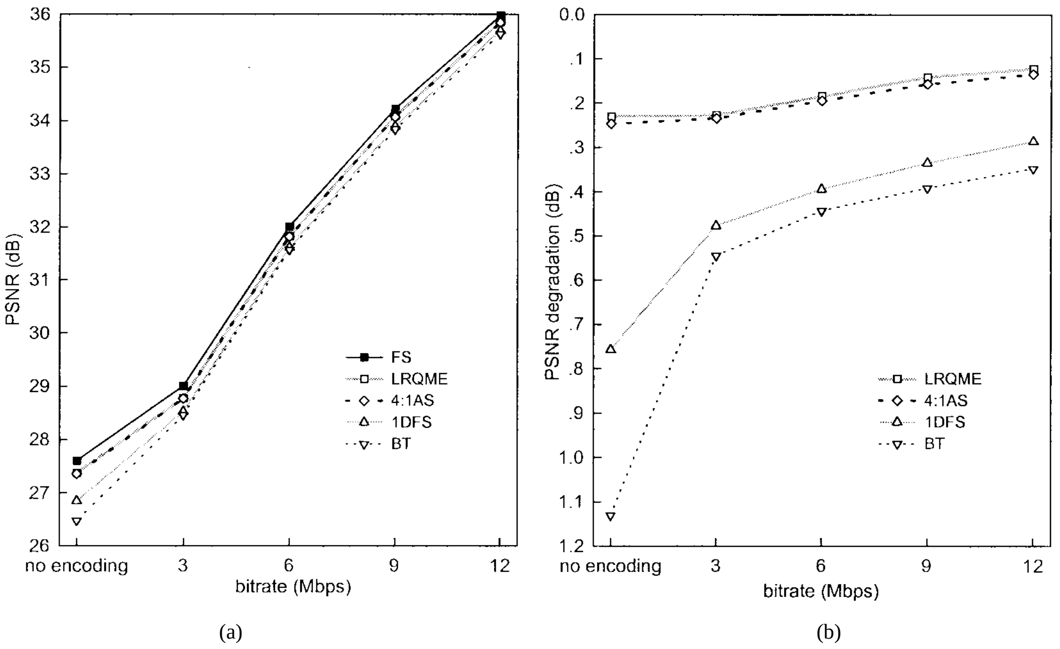


Fig. 6. PSNR versus bit rate in MPEG2 P-prediction encoding for CCIR 601 sequences: (a) PSNR and (b) PSNR degradation.

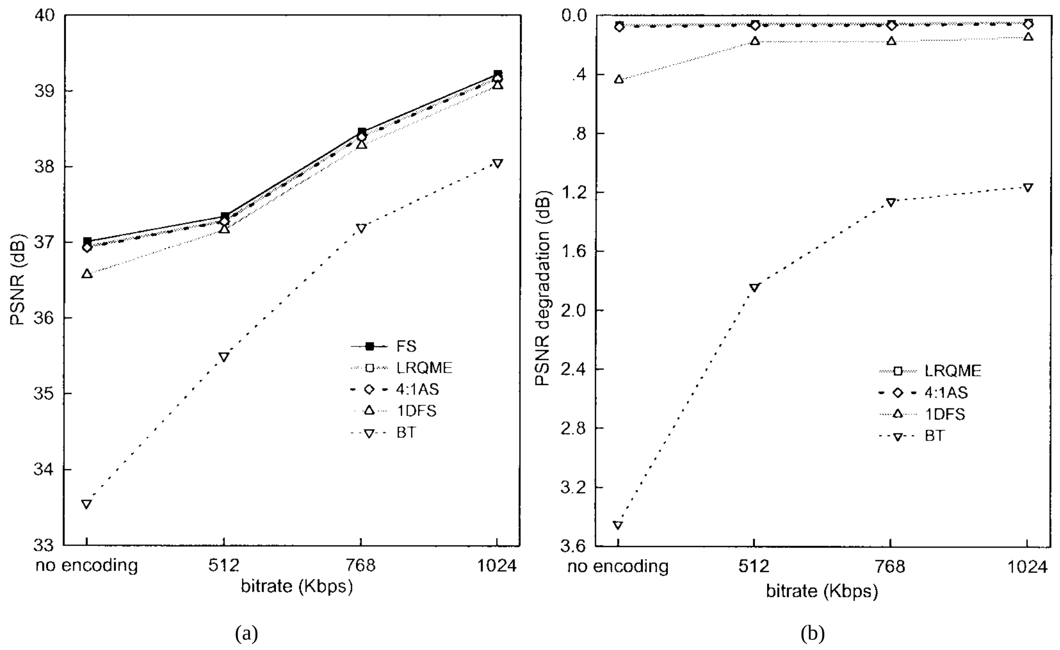


Fig. 7. PSNR versus bit rate in H.261 encoding for CIF sequences: (a) PSNR and (b) PSNR degradation.

of search positions are processed simultaneously, 79 $\times$ 17 quantized codes of the search window data are stored in LRI register 1. 16 $\times$ 16 quantized codes of the reference block data are stored in LRI register 3. After finishing the computation for the Kth, $(K+1)$th rows of search positions, the LRS unit begins processing the $(K+2)$th, $(K+3)$th rows of search positions. The search window data for the $(K+2)$th $(K+3)$th rows of search positions are retrieved from the CSW buffer, and are stored in LRI register 2 while the LRS unit processes the Kth and $(K+1)$th rows of search positions. Since the 79 $\times$ 15 quantized codes of search window data overlap between the Kth, $(K+1)$th rows of search positions and the $(K+2)$, $(K+3)$th rows of search positions, LRI register 2 stores 79 $\times$ 2 quantized codes of search window data.

Fig. 12 shows the block diagram of the LRS PE. It processes one column of the reference block at a time, i.e., 16 pixels, whereas conventional full search architecture processes only one pixel. It has four advantages in hardware reduction described below.

1) Use of Simple Logic Gates Instead of Full Adders: The PE of a conventional full search architecture [18] requires an 8-bit full adder and a 16-bit accumulator for *one pixel comparison* (24 bits of full adders in total), while the PE of the proposed architecture requires 19 bits of full adders and

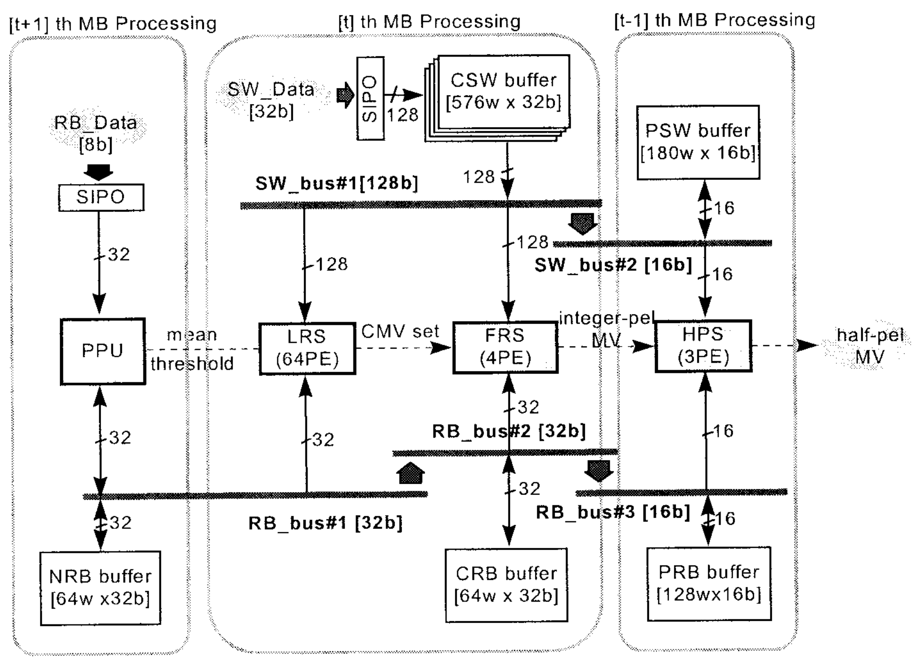


Fig. 8. Block diagram of the proposed architecture.

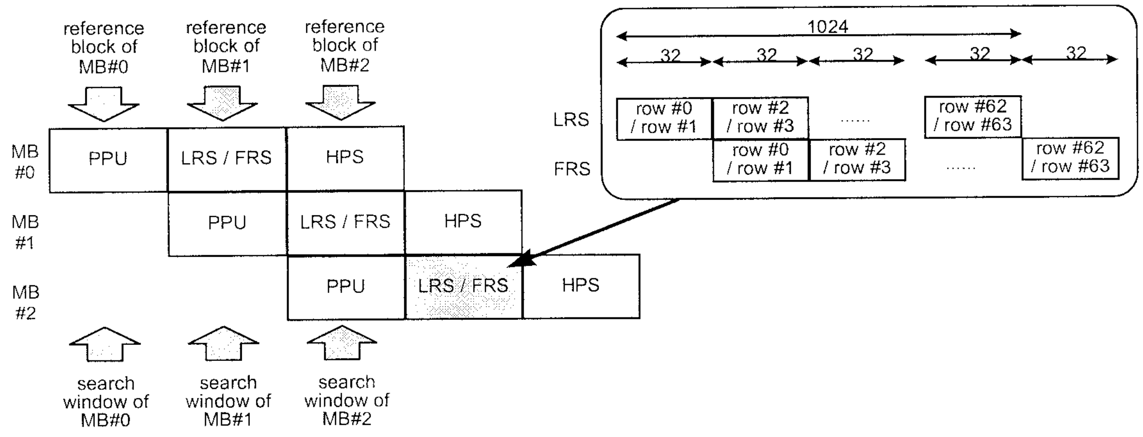


Fig. 9. Pipelining diagram of the proposed architecture.

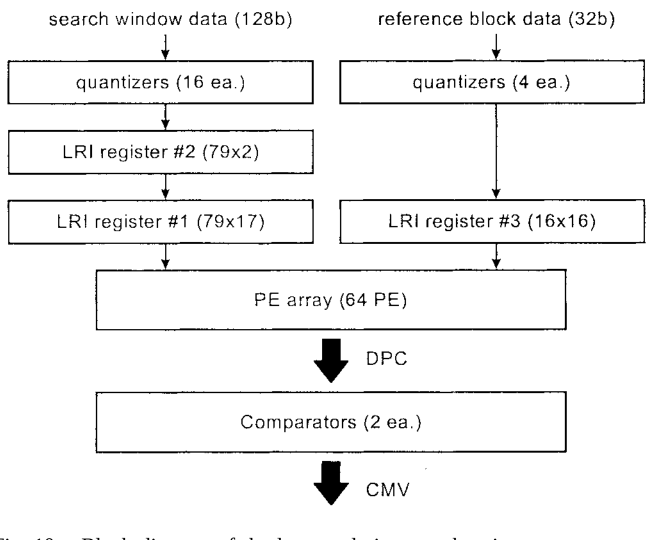


Fig. 10. Block diagram of the low-resolution search unit.

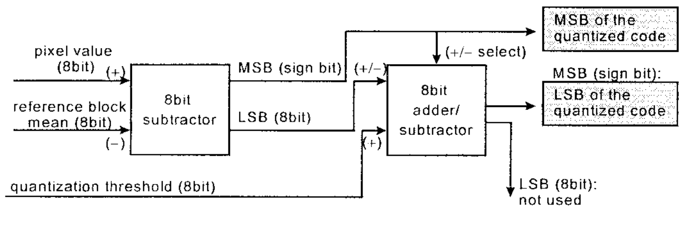


Fig. 11. Adaptive quantizer.

additional simple logic gates for *16 pixel comparison*. Thus, 1/4 reduction in bit resolution produces a reduction in gate count of greater than 1/16.

2) Use of Ripple Carry Adders Instead of Fast, Area-Consuming Aadders: The maximum bits of the full adder in the LRS PE do not exceed 8 bits, allowing the use of ripple carry adders to reduce the amount of hardware, while the PE of a conventional full search architecture requires a fast 16-bit adder such as a carry select adder in the accumulator.

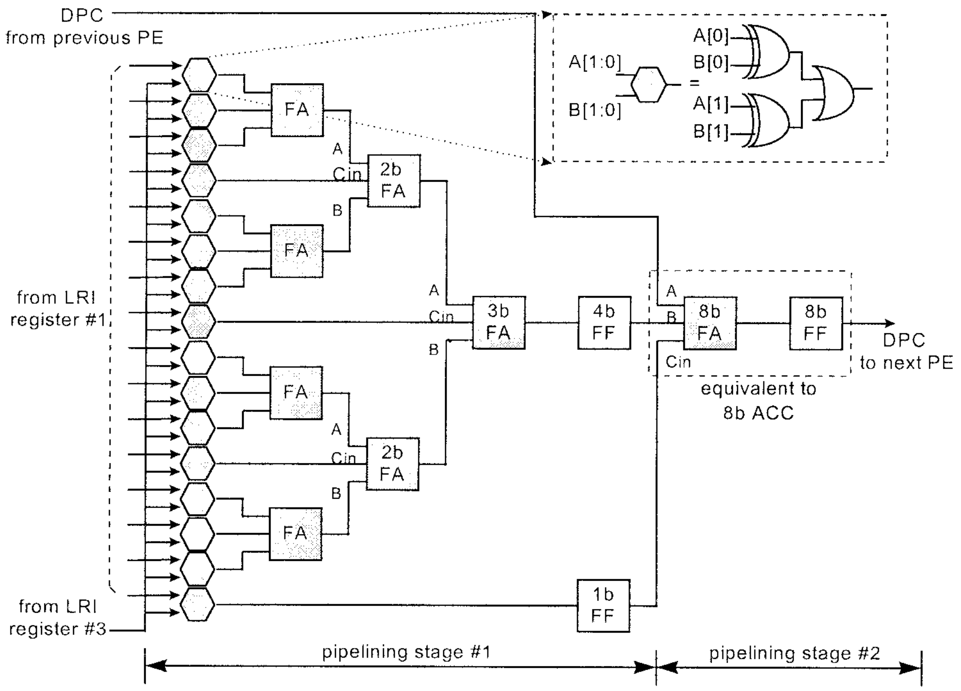


Fig. 12. Block diagram of the LRS PE.

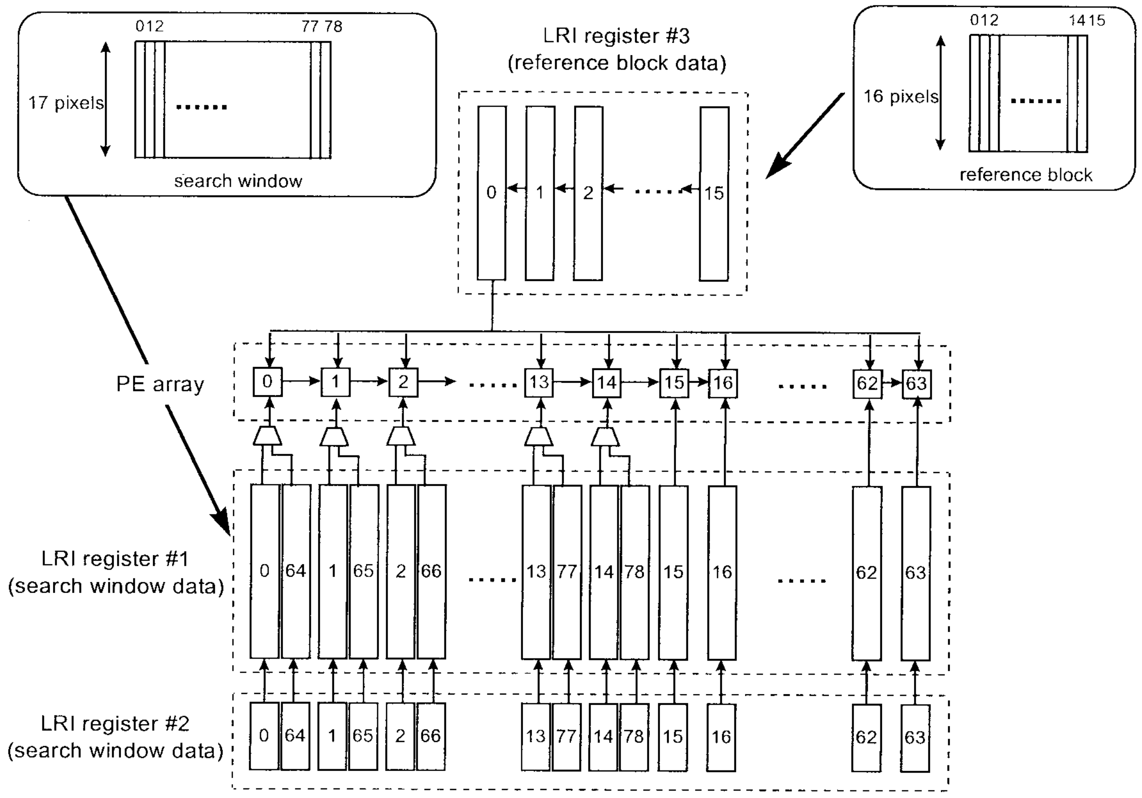


Fig. 13. Block diagram of the LRS PE array and the LRI registers.

3) Reduction in Number of Accumulators: The PE of a conventional full search architecture requires an accumulator for one pixel comparison, while the LRS PE requires the same number of accumulators for *16 pixel comparison* by using an adder tree, as shown in Fig. 12. Therefore, using the LRS PE saves 15 accumulators.

4) Full Utilization of the Adder Tree: The DPC is the sum of 1-bit value, so we can fully utilize the adder tree as shown

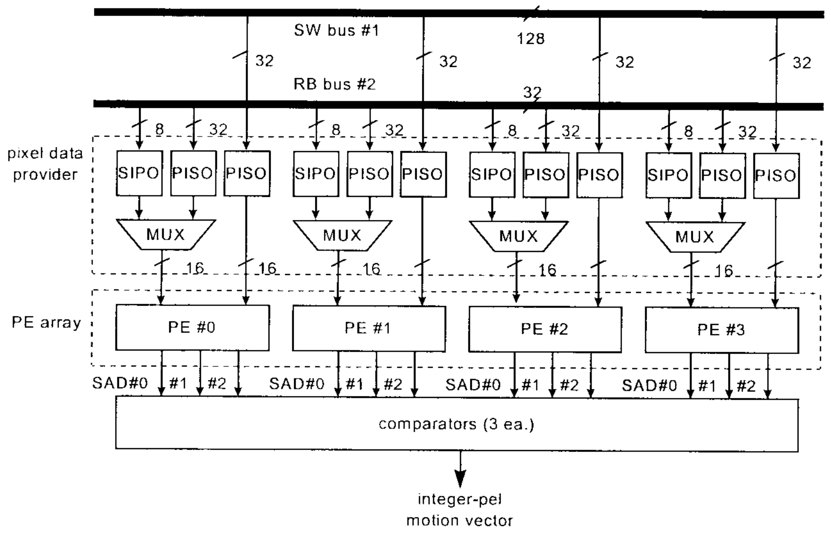


Fig. 14. Block diagram of the full-resolution search unit.

in Fig. 12, which results in a smaller adder tree. Note that the adder tree in the LRS PE utilizes carry-in inputs of the adders, while conventional adder trees do not.

LRI registers 1 and 2 are split into 79 columns, and LRI register 3 is split into 16 columns. Both are connected to PE array, as shown in Fig. 13. The reference block data in LRI register 3 are circulated and broadcast over all PE's, column by column. To avoid shifting the search window data stored in LRI register 1, the partial DPC's in each PE are shifted toward the next PE and accumulated. This reduces the interconnections of LRI register 1. Note that in the first 15 PE's (PE0–PE14), two columns of the search window data are multiplexed into one PE.

C. Full-Resolution Search Unit

The FRS unit determines the integer-pel precision motion vector (IMV) by performing a 4 : 1 alternate subsampling algorithm on the positions in the CMV set. The full-resolution search consists of two steps. In the first step, four candidates are selected in the CMV set with 8 $\times$ 8 SAD comparison. In the second step, the IMV is determined among the four candidates with 16 $\times$ 16 SAD comparison.

Fig. 14 shows the block diagram of the FRS unit. It consists of three main blocks: pixel data provider, PE array, and comparators. The FRS unit has a complicated pixel data provider because the CSW and CRB buffers are split into four banks, and the grouping of the required reference block pixels, in 8 $\times$ 8 and 16 $\times$ 16 pixel comparisons, is different. The PISO (parallel-in serial-out converter) and the SIPO (serial-in parallel-out converter) are required in pixel data provider because the widths of the data from the CSW and CRB buffers are different from the width of the input data into the PE array. Fig. 15 illustrates the block diagram of the FRS PE. It is similar to the PE of conventional full search architecture except that the FRS PE processes two pixels every cycle.

In the proposed architecture, the two prediction modes (frame and field prediction in frame picture, field and 16 $\times$ 8 MC prediction in field picture) are concurrently searched in the 16 $\times$ 16 pixel comparison. Thus, the FRS PE has two accumulators and one adder to calculate the three SAD's of the two prediction modes.

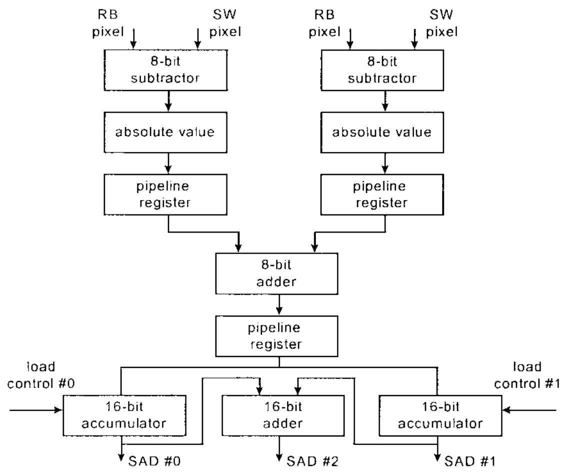


Fig. 15. Block diagram of the FRS PE.

Fig. 16 illustrates the operations of the FRS PE. In 8 $\times$ 8 pixel comparison, an accumulator is activated because only one 8 $\times$ 8 SAD is needed. In 16 $\times$ 16 pixel comparison, two 16 $\times$ 8 SAD's are needed for top and bottom field prediction (in frame prediction) or upper and lower 16 $\times$ 8 block prediction (in field prediction). The 16 $\times$ 16 SAD is the sum of two 16 $\times$ 8 SAD's.

D. VLSI Implementation

The proposed motion estimator was implemented in a VLSI circuit with a 0.5-μm triple-metal CMOS standard-cell technology. It contains 110 K gates of random logic and 90 K bits of SRAM in a die size of 11.5 mm $\times$ 12.5 mm. A microphotograph of the proposed motion estimator is shown in Fig. 17. The full functionality of the fabricated chip was confirmed with an MPEG2 encoder chip.

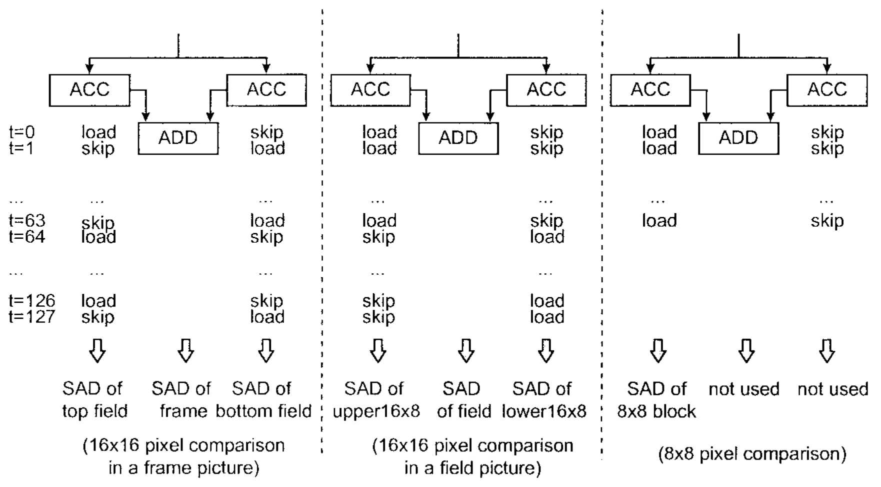


Fig. 16. Operations of the FRS PE.

TABLE V
SUMMARY OF THE MOTION ESTIMATOR

Picture format	CCIR601 (720 pels × 480 pels, 30Hz)
Search range	horizontal: -32.0 ~ +31.5 vertical: -32.0 ~ +31.5 (frame picture) -16.0 ~ +15.5 (field picture)
Operating frequency	54MHz
Prediction mode	frame / field prediction (frame picture) field / 16×8MC prediction (field picture)
Estimated gate count	110 K gates (random logic) 90 K bits (SRAM)
Die size	11.5 ㎜ × 12.5 ㎜
Pin counts	106 pins (signal pins) 74 pins (power pins)
Technology	0.5-㎛ triple-metal CMOS standard-cell technology

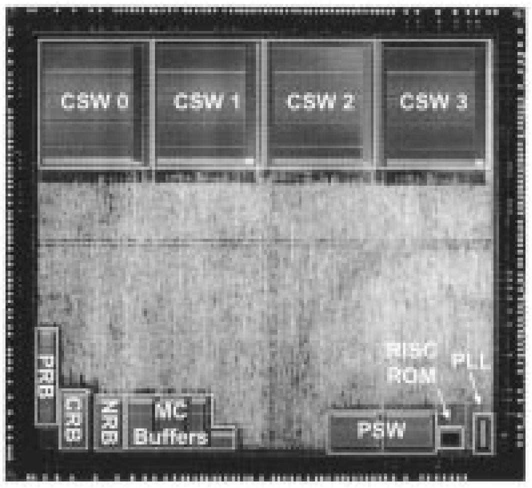


Fig. 17. Microphotograph of the proposed motion estimator.

The motion estimator concurrently performs two prediction modes for real-time MPEG2 motion estimation at 54 MHz. Dual-prime prediction mode is not supported. It includes a built-in RISC (reduced instruction set computer) controller and an 8-bit host bus for an external microcontroller. It also has a motion compensation module that performs interpolation for motion-compensated luminance and chrominance signals. A summary of the motion estimator is shown in Table V.

IV. Conclusions

In this paper, we propose a new motion estimation algorithm that uses adaptively quantized low bit-resolution image. Simulation results show that the proposed adaptive quantization method is more efficient than the LSB truncation method for bit-resolution reduction. Low- and full-resolution search steps are employed to reduce performance degradation. Simulation results show the PSNR performance as superior to 4 : 1 alternate subsampling, while the hardware cost is 1/10 and 1/17 of the full search algorithm when the search range is $\pm 8 \times \pm 8$ and $\pm 32 \times \pm 32$, respectively.

The hardware architecture of the proposed algorithm for real-time MPEG2 video encoding with a search range of $(-32.0, -32.0) \sim (+31.5, +31.5)$ is explained. The amount of hardware is reduced because the block matching is implemented by simple logic gates instead of full adders and the PE processes 16 pixels concurrently in the low-resolution search unit. The two prediction modes of MPEG2 can be performed concurrently. We have tested the functionality of a VLSI implementation of the proposed architecture, which was fabricated with a 0.5-μm triple-metal CMOS technology. Its gate count is about 110 K gates of random logic and 90 K bits of SRAM. We are trying to modify the proposed algorithm for motion estimation for a larger search window such as $\pm 128 \times \pm 128$.

References

[1] CCITT SG XV, "Recommendation H.261—Video codec for audiovisual services at $p * 64$ kbit/s," Aug. 1990.

[2] ISO/IEC JTC1/SC29/WG11 13818-1, "Coding of moving pictures and associated audio," Nov. 1994.

[3] J. R. Jain and A. K. Jain, "Displacement measurement and its application in interframe image coding," *IEEE Trans. Commun.*, vol. COM-29, pp. 1799–1808, Dec. 1981.

[4] A. Netravali and J. D. Robbins, "Motion compensated television coding: Part I," *Bell Syst. Tech. J.*, vol. 58, pp. 629–668, Apr. 1979.

[5] F. Dufaux and F. Moscheni, "Motion estimation techniques for digital TV: A review and a new contribution," *Proc. IEEE*, vol. 83, pp. 858–876, June 1995.

[6] H. Hsieh and T. P. Lin, "VLSI architecture for block-matching motion estimation algorithm," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 2, pp. 169–175, June 1992.

[7] T. Matsumura *et al.*, "A chip set architecture for programmable real-time MPEG2 video encoder," in *Proc. CICC*, 1995, pp. 393–396.

[8] M. J. Chen, L. G. Chen, and T. D. Chiueh, "One-dimensional full search motion estimation algorithm for video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 4, pp. 504–509, Oct. 1994.

[9] T. Koga, K. Linuma, A, Hirano, Y. Iijima, and T. Ishiguro, "Motion-compensated interframe coding for video conferencing," in *Proc. Nat. Television Conf.*, Dec. 1981, pp. G5.3.1–G5.3.5.

[10] B. Liu and A. Zaccarin, "New fast algorithms for the estimation of block motion vectors," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 3, pp. 148–157, Apr. 1993.

[11] H. Gharavi and M. Mills, "Block-matching motion estimation algorithm—New results," *IEEE Trans. Circuits Syst.*, vol. 37, pp. 649–651, May 1990.

[12] M. J. Chen, L. G. Chen, T. Z. Chiueh, and Y. P. Lee, "A new block-matching criterion for motion estimation and its implementation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 5, pp. 231–236, June 1995.

[13] Y. Chan and S. Y. Kung, "Multi-level pixel difference classification methods," in *Proc. ICIP*, 1995, pp. 252–255.

[14] Y. Baek, H. S. Oh, and H. K. Lee, "An efficient block-matching criterion for motion estimation and its VLSI implementation," *IEEE Trans. Consumer Electron.*, vol. 42, pp. 885–892, Nov. 1996.

[15] H. Yeo and Y. H. Hu, "A novel architecture and processor-level design based on a new matching criterion for video compression," in *Proc. Workshop VLSI Signal Processing IX*, 1996, pp. 448–457.

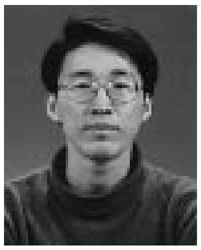
[16] S. Lee and S.-I. Chae, "Motion estimation algorithm exploiting low-resolution quantization," *Electron. Lett.*, vol. 32, pp. 647–648, Mar. 1996.

[17] N. Jayant and P. Noll, *Digital Coding of Waveforms*. Englewood Cliffs, NJ: Prentice-Hall, 1984, p. 131.

[18] K. M. Yang, M. T. Sun, and L. Wu, "A family of VLSI architectures for the full-search block-matching algorithm," *IEEE Trans. Circuits Syst.*, vol. 36, pp. 1317–1325, Oct. 1989.

Seongsoo Lee received the B.S., M.S., and Ph.D. degrees in electrical engineering from Seoul National University, Seoul, Korea, in 1991, 1993, and 1998, respectively.

His research interests are VLSI system design, motion estimation, rate control, and image/video coding.



Jeong-Min Kim received the B.S. and M.S. degrees in electrical engineering from Seoul National University, Seoul, Korea, in 1993 and 1995, respectively, where he is now working toward the Ph.D. degree.

His research interests are VLSI system design, digital signal processor and software–hardware codesign.

Soo-Ik Chae (S'84–M'87) received the B.S. and M.S. degree in electrical engineering from Seoul National University, Seoul, Korea, in 1976 and 1978, respectively, and the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA, in 1987.

He was an Instructor in the Electronics Department at the Korea Air Force Academy from 1978 to 1982. He also worked as a manager in the ASIC Design Group of ZyMOS Corporation and Daewoo Telecom from 1987 to 1990. He joined the Inter-University Semiconductor Research Center and the Department of Electronics Engineering at Seoul National University in 1990. He is currently an Associate Professor in the School of Electrical Engineering at Seoul National University. His research interests are VLSI architecture for video signal processing, VLSI system design, and ultralow-energy circuits.