A Novel Low-Power Full-Search Block-Matching Motion-Estimation Design for H.263+

Jun-Fu Shen, Tu-Chih Wang, and Liang-Gee Chen

***Abstract*—In this paper, a low-power full-search block matching (FSBM) motion-estimation design for ITU-T recommendation H.263+ standard was proposed. New motion-estimation modes in H.263+ can be fully supported by our architecture. Unlike most previously presented motion-estimation chips, this design can deal with 8×8 and 16×16 block size with different searching ranges. Basically, the proposed architecture is composed of an integer pixel unit with 64 processing elements, and a half-pixel unit with interpolation, a control unit, and data registers. In order to minimize power consumption, gated-clock and dual-supply voltages are used. This design has been realized by TSMC 0.6 μm SPTM CMOS technology. The power consumption is 423.8 mW at 60 MHz and the throughput is 36 fps in CIF format.**

***Index Terms*—DSP architecture, motion estimation, video coding.**

I. Introduction

In THE last decade, multimedia applications have become more and more popular. In order to reduce the bandwidth of multimedia data transmission, various video and audio standards were proposed. H.263 Version 2, known as H.263+, was proposed by ITU-T to alleviate the bandwidth problem of digital video transmission. Compared to H.263 Version 1, there are additional 16 negotiable advanced modes in H.263+. The new motion-estimation modes, such as advanced prediction (AP) mode and reduced resolution updated (RRU) mode, provide significant coding gain in H.263+. Although the coding efficiency is better in these modes, more computations are needed. For real-time application, either dedicated hardware or a fast algorithm should be applied. Nowadays, there are many fast algorithms for block-matching motion estimation, such as a three-step hierarchical search [2], 1-D full search [3], 2-D logarithmic search [4], clustering search [5], and sub-sampled search [6] … etc., but the degradation of the picture quality and the irregular data flow make it less attractive in system view. On the other hand, due to the amazing progress of IC fabrication technique, a full-search algorithm could be efficiently implemented in a video system. With an array processing technique, the high computation demand of the full-search algorithm could be met. The optimal performance and regular data flow bring benefits to high-quality video and low-memory addressing complexity. Furthermore, a simple buffer strategy could be used to reduce the external memory transfer.

Manuscript received June 29, 1998; revised March 23, 2001. This paper was recommended by Associate Editor R. Chandramouli.

The authors are with DSP/IC Lab, Department of Electrical Engineering, National Taiwan University, Taipei, 106 Taiwan (e-mail: sor@video.ee.ntu.edu.tw; eric@video.ee.ntu.edu.tw; lgchen@video.ee.ntu.edu.tw).

Publisher Item Identifier S 1051-8215(01)05286-7.

The issue of low power has become more and more important in recent years due to the widespread application of portable devices. For higher computation, causing much more power consumption, the power-consumption issue of the motion estimator should not be ignored in video codecs. To prolong the using period between battery recharges, it is a trend to develop the low-power motion-estimation architecture.

This paper is organized as follows. New H.263+ algorithms related to this design are described in Section II. The architecture overview is then described in Section III. The architectures of integer-pixel precision unit (IU) and half-pixel precision unit (HU) are depicted in Sections IV and V. Section VI shows the chip implementation and comparison with other design. Finally, a conclusion is given in Section VII.

II. New Motion-Estimation Modes in H263+

In H.263+ standard, there are 16 advanced modes improving the picture quality and reducing transmission bit rate, and also five advanced modes related to motion estimation, i.e., AP mode, PB-frame mode, improved PB-frame mode, unrestricted motion vector (UMV) mode, and RRU mode. UMV mode and PB-frame mode can be easily implemented by reordering input data and have little influence on the core architecture. The only two advanced modes that affect motion-estimation architecture are AP mode and RRU mode.

A. AP Mode

In AP mode, there are three features shown as follows:

1) motion vectors are allowed to cross picture boundaries;
2) four motion vectors per macroblock (8×8 block size);
3) overlapped motion compensation for the luminance block.

In AP mode, not only the mean absolute error (MAE) in the macroblock, but also the MAE in each block, is used to determine the final motion vector. Thus, those motion-estimation cores which can not deal with different block sizes are not suitable for this new standard.

B. PB Frame Mode

A PB frame consists of two pictures, i.e., a P-picture and a B-picture, being coded as one unit. A P-picture is predicted from the previously decoded P-picture and a B-picture is predicted both from the previously decoded P-picture and the P-picture currently being decoded. The second picture is called a B-picture, since parts of this picture may be bidirectionally predicted similar to the B-frame in MPEG standard. The main advantage of using PB-frame mode is that the frame rate can be double,

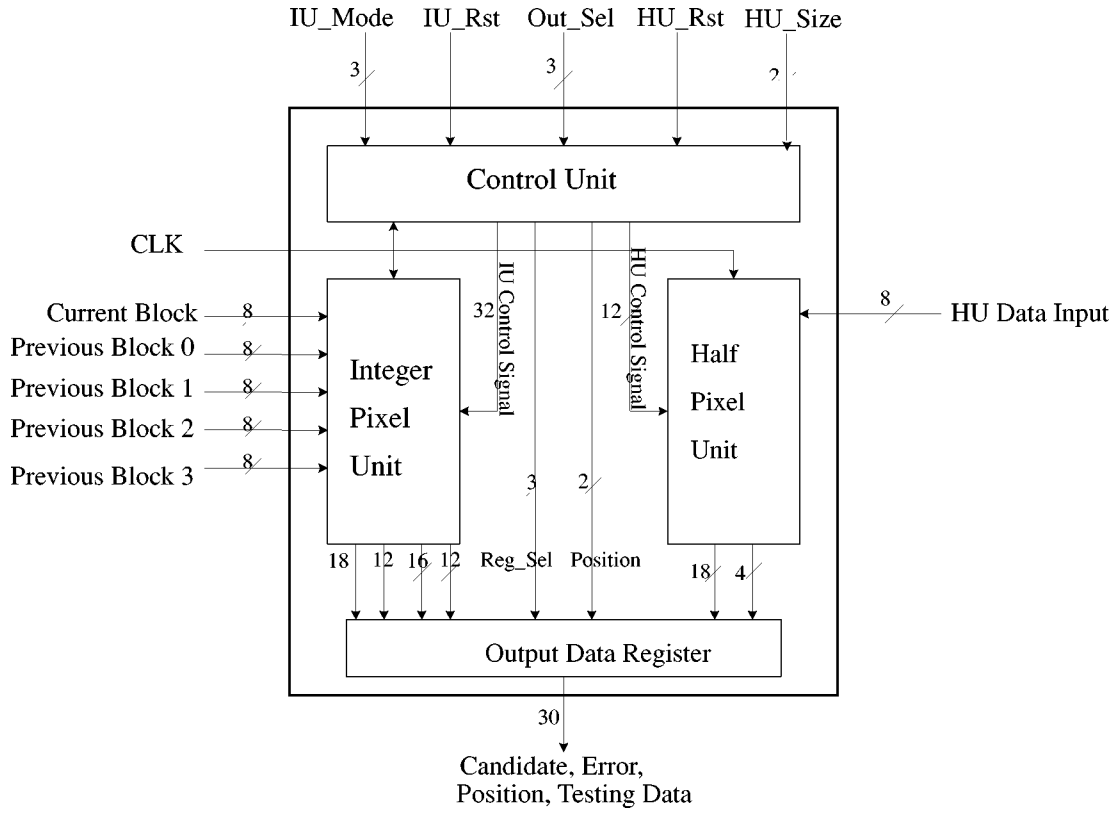

Fig. 1. Architecture overview.

without increasing the bit rate much because of coding these two pictures in one unit.

C. RRU Mode

In RRU mode, the size of the macroblock and block are twice as wide and high as the usual size. The searching range is also doubled in width and height. Therefore, in RRU mode, the macroblock size is 32×32, the block size is 16×16, and the searching range becomes 64×64. To reduce the bit rate in each picture, the DCT data in a 16×16 block is sub-sampled in 8×8 block size on a reduced-resolution version of each picture. Although the prediction error is sub-sampled, the picture quality still maintains sufficiently due to full resolution in the motion-compensation process.

In order to realize RRU mode, motion-estimation design must support 32×32 block size and larger search range. The computation complexity with 64×64 search range is 16 times larger than normal motion estimation and is the most computation exhaustive among these modes. Computation demand in this design could be obtained by analyzing this mode.

III. Architecture Overview

Fig. 1 shows the architecture of this design. The IU and HU are used to calculate the MAE in integer-pixel precision and half-pixel precision individually. Considering the tradeoff between speed and area, 64 is the number of processing elements (PEs) chosen. With 64 PEs, the clock rate is about 60 MHz in RRU mode for real-time application, and the chip area could be controlled within a reasonable range.

TABLE I
Clock Cycles to Generate a Motion Vector in IU and HU

Mode	Normal mode	AP mode	RRU mode
Integer unit	4096	1024	65536
Half unit	2304/k	576/k	9216/k

The PE number in the HU could be determined by analyzing the throughput demand of the HU. Table I shows the numbers of clock cycles generating a motion vector in the IU and HU blocks. The number of PEs in the IU is set to 64, and the one in HU is set k. It should be as small as possible and meet the requirement that HU be faster than IU. From Table I, it is obvious that one PE is enough.

In order to provide sufficient data and achieve 100% PE utilization, there are four input ports for previous block data and one input port for current block data in IU. In HU, previous block data and current block data could share the same port due to the low requirement of input bandwidth. In the output block, we use an identical 30-bit register to output the motion vector and the minimum absolute error for IU and HU sequentially to reduce the number of output pads.

IV. Architecture for the IU

A 1-D and 2-D mixed architecture is used in the IU. Unlike general 1-D full-search architecture, 64 PEs are not mapping to some rows of searching range, but a fixed 8×8 square searching range. We call this a 1-D and 2-D mixed architecture. Fig. 2 is the architecture of the IU. Current block data is propagated through the 64 8-bit shift registers on the right side, and previous

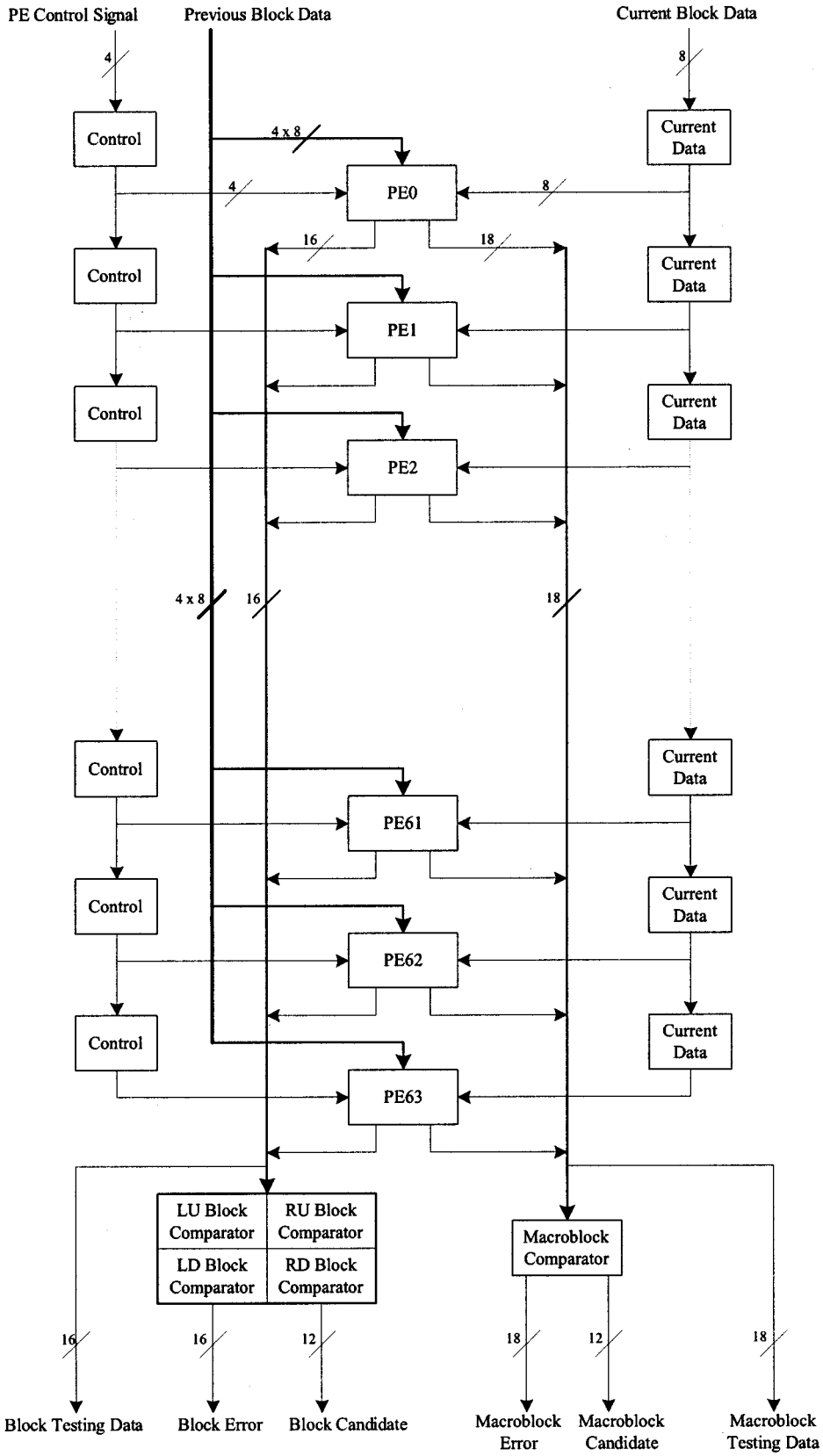

Fig. 2. IU architecture.

block data broadcasts to each PE from the four 8-bit buses. The MAE in each PE is transmitted to the macroblock comparator and the block comparator through the 18- and 16-bit bus.

Fig. 3 shows the IU data flow for a 8×8 block size and 8×8 searching range. Four previous block data are needed at the same time to provide PE processing. The processing of each

	Port0	Port1	Port2	Port3	PE0	PE1	PE2	PE7	PE8	PE9	PE63
0	P(0,0)		P(8,7)	P(0,8)	C(0,0)-P(0,0)	C'(7,7)-P'(8,7)	C'(6,7)-P'(8,7)	C'(1,7)-P'(8,7)	C'(0,7)-P'(0,8)	C'(7,6)-P'(8,7)	C'(1,0)-P'(8,7)
1	P(1,0)		P(9,7)	P(1,8)	C(1,0)-P(1,0)	C(0,0)-P(1,0)	C'(7,7)-P'(9,7)	C'(2,7)-P'(9,7)	C'(1,7)-P'(1,8)	C'(0,7)-P'(1,8)	C'(2,0)-P'(9,7)
2	P(2,0)		P(10,7)	P(2,8)	C(2,0)-P(2,0)	C(1,0)-P(2,0)	C(0,0)-P(2,0)	C'(3,7)-P'(10,7)	C'(2,7)-P'(2,8)	C'(1,7)-P'(2,8)	C'(3,0)-P'(10,7)
3	P(3,0)		P(11,7)	P(3,8)	C(3,0)-P(3,0)	C(2,0)-P(3,0)	C(1,0)-P(3,0)	C'(4,7)-P'(11,7)	C'(3,7)-P'(3,8)	C'(2,7)-P'(3,8)	C'(4,0)-P'(11,7)
4	P(4,0)		P(12,7)	P(4,8)	C(4,0)-P(4,0)	C(3,0)-P(4,0)	C(2,0)-P(4,0)	C'(5,7)-P'(12,7)	C'(4,7)-P'(4,8)	C'(3,7)-P'(4,8)	C'(5,0)-P'(12,7)
5	P(5,0)		P(13,7)	P(5,8)	C(5,0)-P(5,0)	C(4,0)-P(5,0)	C(3,0)-P(5,0)	C'(6,7)-P'(13,7)	C'(5,7)-P'(5,8)	C'(4,7)-P'(5,8)	C'(6,0)-P'(13,7)
6	P(6,0)		P(14,7)	P(6,8)	C(6,0)-P(6,0)	C(5,0)-P(6,0)	C(4,0)-P(6,0)	C'(7,7)-P'(14,7)	C'(6,7)-P'(6,8)	C'(5,7)-P'(6,8)	C'(7,0)-P'(14,7)
7	P(7,0)			P(7,8)	C(7,0)-P(7,0)	C(6,0)-P(7,0)	C(5,0)-P(7,0)	C(0,0)-P(7,0))	C('7,7)-P'(7,8)	C'(6,7)-P'(7,8)	C'(0,1)-P'(7,8)
8	P(8,0)	P(0,1)	P(0,9)	P(8,8)	C(0,1)-P(0,1)	C(7,0)-P(8,0)	C(6,0)-P(8,0)	C(1,0)-P(8,0)	C(0,0)-P(0,1)	C'(7,7)-P'(8,8)	C'(1,1)-P'(8,8)
9	P(9,0)	P(1,1)	P(1,9)	P(9,8)	C(1,2)-P(1,1)	C(0,1)-P(1,1)	C(7,0)-P(9,0)	C(2,0)-P(9,0)	C(1,0)-P(1,1)	C(0,0)-P(1,1)	C'(2,1)-P'(9,8)
60	P(12,6)	P(4,7)		P(12,14)	C(4,7)-P(4,7)	C(3,7)-P(4,7)	C(2,7)-P(4,7)	C(5,6)-P(12,6)	C(4,6)-P(12,6)	C(3,6)-P(12,6)	C'(5,7)-P'(12,14)
61	P(13,6)	P(5,7)		P(13,14)	C(5,7)-P(5,7)	C(4,7)-P(5,7)	C(3,7)-P(4,7)	C(6,6)-P(13,6)	C(5,6)-P(13,6)	C(4,6)-P(13,6)	C'(6,7)-P'(13,14)
62	P(14,6)	P(6,7)		P(14,14)	C(6,7)-P(6,7)	C(5,7)-P(6,7)	C(4,7)-P(4,7)	C7,6)-P(14,6)	C(6,6)-P(6,7)	C(5,6)--P(6,7)	C'(7,7)-P'(14,14)
63		P(7,7)			C(7,7)-P(7,7)	C(6,7)-P(7,7)	C(5,7)-P(4,7)	C(0,7)-P(7,7)	C(7,6)-P(7,7)	C(6,6)-P(7,7)	C'(0,0)-P'(7,7)

Fig. 3. Dataflow in IU for a 8×8 block with 8×8 searching range. There is no switching cycle during block transition.

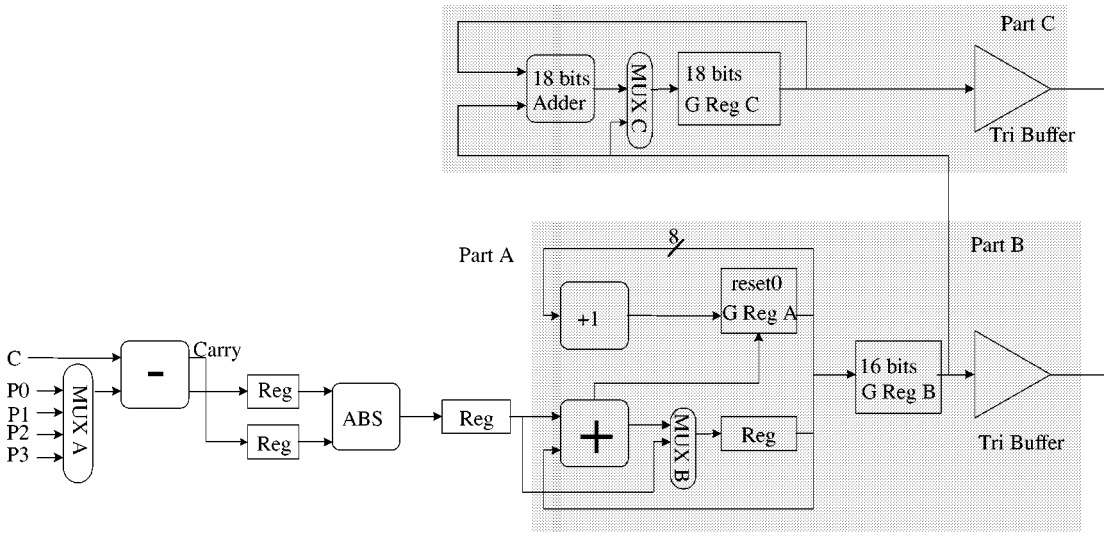

Fig. 4. PE architecture.

PE is skewed, and MAE calculation in each PE will finish one after another. For example, PE0 starts at cycle 0 and finishes at cycle 63 to search the candidate (0,0). PE1 starts at cycle 1 and finishes at cycle 64 to produce MAE of (1,0). PE8 searches the start point of the next row (0,1) from cycle 8 to cycle 71. There is no bubble cycle to switch between different current block and different previous block, which is well demonstrated in Fig. 3 (switching between c' to c and p' to p). This is an important property of this design to guarantee 100% utilization of the PE array.

The MAE outputs are then sent to macroblock comparator and block comparator cycle by cycle. Since there is only one PE completed at each cycle, two buses (18 bit, 16 bit) are enough.

A. PE Architecture

Unlike previous PE architectures, the PE cell proposed here has three additional features: low power consumption, supporting AP mode, and supporting RRU mode. Fig. 4 shows the PE architecture. The major difference from other PE

TABLE II
DATA FLOW IN THE PE FOR A 16 × 16 MACROBLOCK

Clock cycle	Block accumulator	Register B	Macroblock accumulator
0 - 63	accumulate LU block 1	-	-
64 - 127	accumulate LD block 1	store LU block 1	-
128 - 191	accumulate RU block 1	store LD block 1	LU
192 - 255	accumulate RD block 1	store RU block 1	LU+LD
256 - 319	accumulate LU block 2	store RD block 1	LU+LD+RU
320 - 383	accumulate LD block 2	store LU block 2	LU+LD+RU+RD

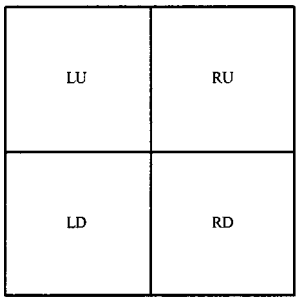

Fig. 5. Block order in a macroblock.

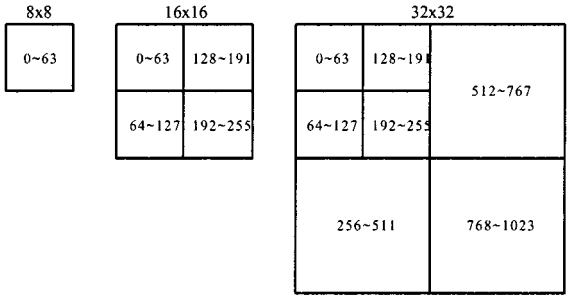

Fig. 6. Block processing order in different block size.

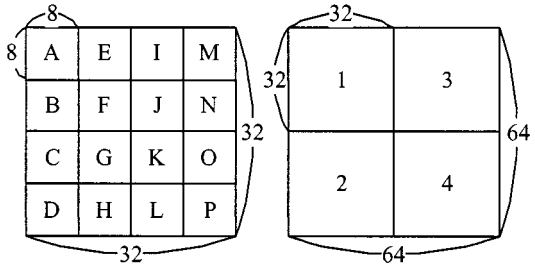

Fig. 7. Partition of a 64 × 64 searching range.

architectures proposed before is marked in the gray blocks. "G register" in Fig. 4 is a gated clock register. In part A, MUX A is used to select input from the four previous block data input buses, and input C is the current block data. Part A is almost the same as the PE architecture of previous design and has the function of abs(X-Y).

The function of the gray part B, which is called the "block accumulator," is to accumulate errors in an 8×8 block. To reduce the power consumption, gated clock control is used in the block accumulator. Because the absolute error of each pixel is always 8 bits, the 16-bit adder that accumulates the errors of each block could be replaced with an 8-bit adder and an 8-bit incrementor. The carry of the 8-bit adder is wired to the clock of register A, which could reduce the activity of the register. The other benefit from this configuration is that the delay path will be shorten from 16 bit adder delay to 8-bit adder delay plus register delay. Thus, the critical path in PE is shortened.

The gray part C is macroblock accumulator. It is used to accumulate 16×16 macroblock errors. We could get macroblock errors by adding four block errors together at the same search candidate and need not recalculate all the errors. In this design, the macroblock accumulator accumulates the errors from block accumulator every 64 cycles. Thus, the circuit activity is low in part C, and we could control register B and C with a 64 times lower frequency clock and use an 18-bit carry ripple adder to implement the adder. The critical path would not be longer due to the low clock rate, and the power needed can be reduced furhter. The detailed data flow is shown in Table II and Fig. 5.

The reason why we include the macroblock accumulator in PE architecture instead of using one accumulator with 64 18-bit registers to form a specific accumulator block is for consuming low power. If only one accumulator is used, the macroblock bus will swing every clock cycle and certainly results in much more power consumption. Although the area is slightly bigger, we believe this configuration is more suitable for low-power applications.

B. Variable Block Size and Searching Range

Variable block size could be achieved with the PE architecture mentioned above. The variable block size processing order is shown in Fig. 6. If we need MAE with 8×8 block size, the PE will send MAE in the block accumulator to the comparator. If we need MAE with 16×16 block size, the macroblock accumulator will be used. For 32×32 block size, it is divided into four 16×16 sub-blocks. Every 256 clock cycles, the block accumulator in each PE calculate the MAE of the 8×8 sub-block. After repeating four times (1024 cycles), the MAE of a 32×32 block could be figured out and sent to the comparator. The data flow is the same as the one of 16×16 block size, but the clock in the macroblock accumulator is 256 times longer than the one in the block accumulator.

To achieve variable searching range, a searching range partition strategy is applied. As discussed before, we could divide a 32×32 searching range into 16 8×8 sub-searching ranges, and the PE array could deal with one sub-searching range in 64 cycles. Similarly, 64×64 searching range could be divided into 64 8×8 sub-searching ranges as shown in Fig. 7. The processing

order is from 1 to 4 and A to P. Because these 64 PEs are mapping to an 8×8 2-D sub-searching range with one 8×8 block, we can easily achieve the variable searching range by dividing the whole searching into several 8×8 sub-searching ranges and map to this architecture. The data flow of each sub-searching range is the same as in Fig. 3, so there is no latency between each sub-searching range.

C. H.263+ Advanced Mode Support

As mentioned before, the architecture can support the AP, PB, and RRU modes in H.263+ [1]. In AP mode, we use the MAE data in the block accumulator and macroblock accumulator to find the final motion vector. It could be treated as combining an 8×8 block size and 16×16 block size at the same time. In normal mode, only the MAE data in macroblock accumulator is used. RRU mode could be realized by setting block size to 32 and searching range to 64. This scheme is discussed in Section IV-B.

In PB mode, the searching range is smaller than 32×32. A 5×5 searching range is applied here. Without modifying the PE array structure, MAE in the same row could be calculated at the same time. By using control unit and gated logic, the forward five PEs are set to active and the other 59 PEs are halted to reduce the power consumption.

D. Control-Signal Flow Design

There are some methodologies to design the control unit. We can place a counter in each PE to ndividually with some global control signals such as reset, AP mode indicator, and RRU mode indicator. Or we can use very complex structure to design the controller and just send the control signals to each PE by a very wide bus (about $5 \times 64\,320$ bits). But these approaches are not optimal here because there are some relations on the control signal between nearby PEs. Thus, we use another approach to design the control unit. As shown in Fig. 2, there are 64 4-bit shift registers placed in the left side to store the control signal. The penalty of this approach is only the additional 64 4-bit registers. No counter or wide bus is needed. Referring to the data flow discussed in Section II, 64 PEs are working sequentially during 64 clock cycles. So the control signal only needs to be sent to first PE. This approach also reduces chip area because the global bus is omitted.

This approach works correctly for control signals to the accumulators in each PE. But there is a problem in control signals that select previous block data buses. Table III is the original relation between a control signal and previous block data buses ordering. With this setting, the previous data control signal will not follow the rule to shift. To solve this problem, we modified the buses arrangement of previous block data, which is shown in Table IV. By this modification, the modified control signal data flow could be implemented with a shift register. The complexity of the control signal is thus simplified.

TABLE III
ORIGINAL BLOCK DATA BUSES ORDERING

Control signals	00	01	10	11
PE0 - PE7	Bus 0	Bus 1	Bus 2	Bus 3
PE8 - PE15	Bus 0	Bus 1	Bus 2	Bus 3
PE16 - PE23	Bus 0	Bus 1	Bus 2	Bus 3
PE24 - PE31	Bus 0	Bus 1	Bus 2	Bus 3
PE32 - PE39	Bus 0	Bus 1	Bus 2	Bus 3
PE40 - PE47	Bus 0	Bus 1	Bus 2	Bus 3
PE48 - PE55	Bus 0	Bus 1	Bus 2	Bus 3
PE56 - PE63	Bus 0	Bus 1	Bus 2	Bus 3

TABLE IV
MODIFIED BLOCK DATA BUSES ORDERING

Control signals	00	01	10	11
PE0 - PE7	Bus 0	Bus 0	Bus 2	Bus 3
PE8 - PE15	Bus 1	Bus 1	Bus 3	Bus 2
PE16 - PE23	Bus 0	Bus 0	Bus 2	Bus 3
PE24 - PE31	Bus 1	Bus 1	Bus 3	Bus 2
PE32 - PE39	Bus 0	Bus 0	Bus 2	Bus 3
PE40 - PE47	Bus 1	Bus 1	Bus 3	Bus 2
PE48 - PE55	Bus 0	Bus 0	Bus 2	Bus 3
PE56 - PE63	Bus 1	Bus 1	Bus 3	Bus 2

V. HU ARCHITECTURE

The HU is composed of an interpolation pixel generator, a processing element, and a comparator. The detailed block diagram is shown in Fig. 8. The input of current block data and previous block data share the same input port because of the low demand of input bandwidth. Fig. 9 is the architecture of the interpolation pixel generator. In this architecture,only two adders and seven registers are used. The gated clock register is marked as "G" in Fig. 9. The register A is used to store current block data. The registers B, C, and D are used to store the previous block data during each interpolation cycle. An interpolation cycle is the period of producing eight surrounding half pixels of integer-pixel. If the block size is 8×8, we need 64 interpolation cycles to finish a block matching in HU.

Fig. 10 is the input pixels and the generated half-pixels during an interpolation cycle. P0 to P8 are previous block integer-pixels. H0 to H7 are half pixels generated surrounding P0. And T1 to T4 are temporary pixels produced during the interpolation cycle. Table V shows the data flow in the interpolation pixel generator. The input data stream is C, P0, P1, ... , P8, the output data stream is H0, H1, H2, ..., H7, and the current block pixel C is stored in register A during the interpolation cycle. From Table V, we could conclude that one interpolation cycle equals 11 clock cycles. Thus, $11 \times 8 \times 8704$ clock cycles are needed for an 8×8 block half-pixel search. This number is still smaller than the clock cycles in IU. As discussed in Section II, there is only one PE in the HU. In order to calculate nine candidates during a block matching, we use nine shift registers to store partial absolute errors.

VI. IMPLEMENTATION AND COMPARISON

To reduce the design period and meet the low-power consumption requirement, we use full-custom and cell-based hybrid design flow. Full-custom design flow is used to design the PE array because of its large area, high power consumption, and

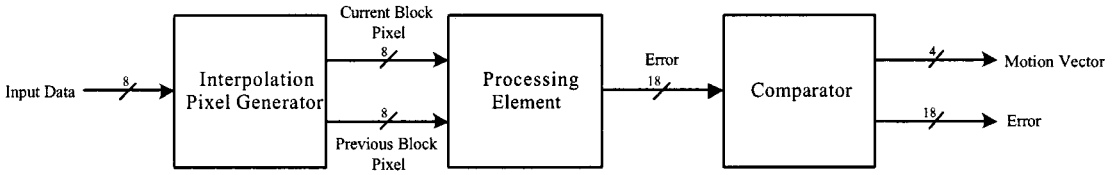

Fig. 8. Architecture of HU.

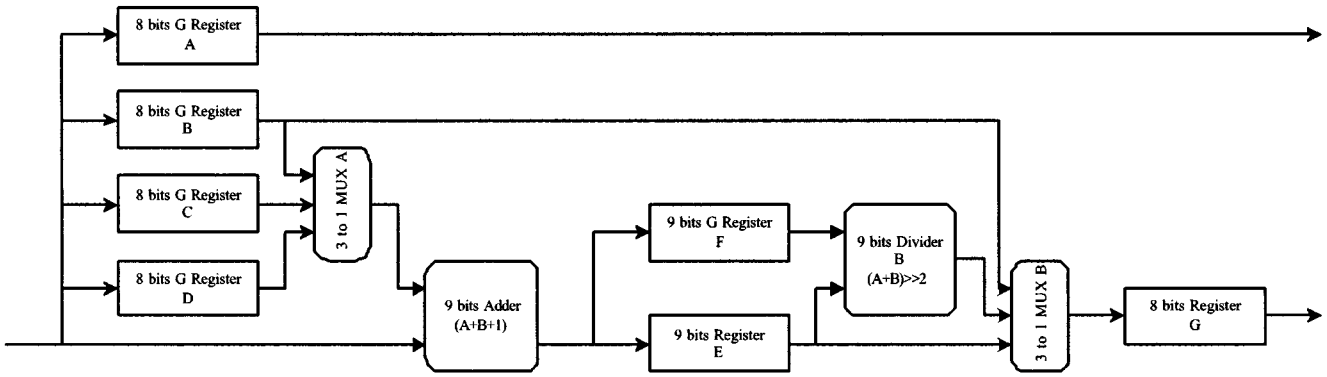

Fig. 9. Architecture of the interpolation pixel generator.

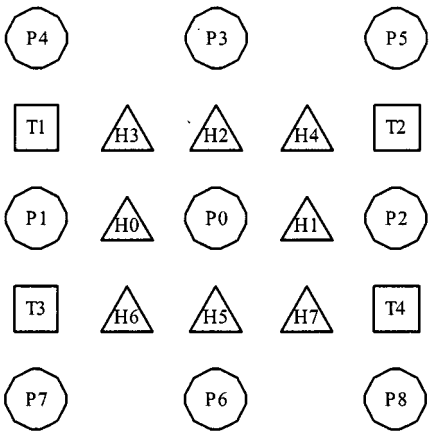

Fig. 10. Input pixels and the half-pixel positions.

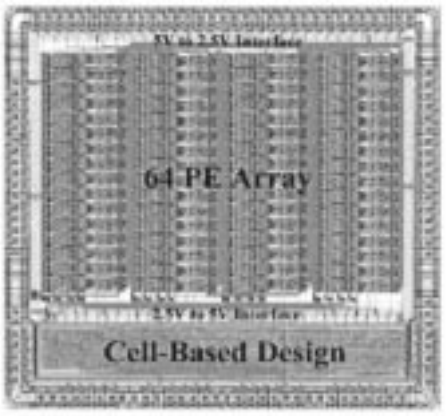

Fig. 11. Physical layout view.

TABLE V
DATA FLOW IN THE INTERPOLATION PIXEL GENERATOR

Input	A	B	C	D	E	F	G
C							
P0	C						
P1	C	P0					
P2	C	P0	P1		H0		P0
P3	C	P0	P1	P2	H1		H0
P4	C	P0	P1	P2	H2	H2	H1
P5	C	P0	P1	P2	T0	H2	H2
P6	C	P0	P1	P2	T1	H2	H3
P7	C		P1	P2	H5	H5	H4
P8	C			P2	T2	H5	H5
	C				T3	H5	H6
C'	C						H7

TABLE VI
CHIP SPECIFICATION

Parameters of the design	Proposed chip for H.263+
PE number	64
Frame size	176×144 & 352×288
Searching range	16×16 & 32×32
Block size	8×8, 16×16 & 32×32
Frame rate	30 to 120 frames/sec
Technology	TSMC 0.6um CMOS SPTM
Number of transistors	267208
Number of I/O pads	113
Core size ($\mu m \times \mu m$)	6702.8×6366.2
Die size ($\mu m \times \mu m$)	6344.8×6008.2
Package	120 CQFP
Clock rate	60MHz
Supply voltage	2.5V & 5V
Power	423.8mW

TABLE VII
COMPARISON OF SOME FSBM MOTION-ESTIMATION CHIP

	S.K.Rao '93[11]	Ishihara '95[12]	A.Otani '95[13]	H.D.Lin '96[10]	G. Fujita '97[5]	This work [14]
Process	0.9um	0.5um	0.5um	0.5um	0.35um	0.6um
Supply Voltage	5V	3.3V	3.3V	3.3V	-	5V&2.5V
Core size(mm2)	23.6	147	105	8.36	1.34	42.67
Clock rate	45MHz	40MHz	80MHz	66MHz	15MHz	60MHz
Transistors	395k	850k	1500k	162k	48k	267k
Power	1.33W	2.37W	1.85W	300mW	-	424mW
Frame precision	Integer	Half	Integer	Integer	Half	Half
Block size	16	16	16	16	16,8	32,16,8
AP mode	No	No	No	No	Yes	Yes
PB mode	No	No	No	No	Yes	Yes
RRU mode	No	No	No	No	No	Yes

regularity. A 2.5-V supply voltage is applied in this part to reduce the power consumption. In other modules, such as the controller, HU, and comparators in the IU, cell-based design flow is used to reduce the design period. A 5-V supply voltage is used to meet cell library timing requirement. Because shift registers spend most power in our architecture, we use TSPC register proposed in [7], which consumes less power than a conventional register. A level converter circuit proposed in [8] is used on the boundary between 5- and 2.5-V parts. Compared to conventional style [9], this circuit avoids the dc path and consumes less power. This chip has been implemented with TSMC 0.6-μm SPTM technology and was tested by IMS200. The maximum working frequency is 66.7 MHz at 5- and 2.5-V supply voltage. Fig. 11 shows the physical layout view with the whole chip. The detailed description of the chip specification is listed in Table VI.

A brief comparison with some FSBM motion-estimation chips is given in Table VII. G. Fujita [5] designed for H.263 and H. D. Lin [10] designed for H.261. The clock rate shown here is the working frequency for real-time application. From Table VII, the design proposed here has superior performance on power, flexibility, and functions over the previously proposed design.

VII. CONCLUSION

In this paper, a low-power FSBM motion-estimation architecture that supports half-pixel precision, AP mode, PB mode, and RRU mode, was proposed. The complexity of control signals is also considered.The control unit is implemented using a much easier way by switching wires properly. Due to proper design for a PE cell, this architecture can deal with variable block sizes at the same time; the searching range could vary as well. The power consumption here is lower because of the TSPC circuit, gated clock, and dual-supply voltage techniques. This design has been implemented by TSMC 0.6-μm SPTM technology and shows even better performance and flexibility than many other FSBM motion-estimation chips previously.

REFERENCES

[1] *ITU-T Recommendation H.263 version 2*, Sept. 1997.
[2] M. Bierling, "Displacement estimation by hierarchical block-matching," in *SPIE Vis. Commun. and Image Processing*, 1988, pp. 942–951.
[3] 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.
[4] J. R. Jain and A. K. Jain, "Displacement measurement and its application in interframe image coding," *IEEE Trans. Commun.*, vol. 29, pp. 799–808, Dec. 1981.
[5] G. Fujita, T. Onoye, and I. Shirakawa, "A new motion estimation core dedicated to h.263 video coding," in *Proc. IEEE Circuits and Systems*, vol. 2, 1997, pp. 1161–1164.
[6] J. Chalidabhongse and C. C. Jay Kuo, "Fast motion vector estimation using multiresolution-spatio-temporal correlations," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, pp. 477–488, June 1997.
[7] J. Yuan and C. Svensson, "High-speed CMOS circuit technique," *IEEE J. Solid-State Circuits*, vol. 24, pp. 62–70, Feb. 1989.
[8] J. S. Caravella and J. H. Quigley, "Three volt to five volt interface circuit with device leakage limited dc power dissipation," in *Proc. IEEE ASIC Int. Conf. and Exhibit*, 1993, pp. 448–451.
[9] N. H. E. Weste and K. Eshraghian, *Principles of CMOS VLSI design—A Systems Perspective* . Reading, MA: Addison-Wesley, 1984.
[10] H. D. Lin, A. Anesko, and B. Petryna, "A 14-gops programmable motion estimator for h.26x video coding," *IEEE J. Solid-State Circuits*, vol. 31, pp. 1742–1750, Nov. 1996.
[11] S. K. Rao, M. Hatamian, M. T. Uyttendaele, S. Narayan, J. H. O'Neill, and G. A. Uvieghara, "A versatile and powerful chip for real time motion estimation," in *Proc. IEEE ISSCC*, 1993, pp. 32–33.
[12] K. Ishihara, S. Masuda, S. Hattori, H. Nishikawa, Y. Ajioka, T. Yamada, H. Amishiro, S. Uramoto, M. Yoshimoto, and T. Sumi, "A half-pel precision mpeg2 motion-estimation processor with concurrent three-vector search," *IEEE J. Solid-State Circuits*, vol. 30, pp. 1502–1509, Dec. 1995.
[13] A. Ohtani, Y. Matsumoto, M. Gion, H. Yoshida, T. Araki, A. Ubukata, M. Serizawa, K. Aoki, A. Sota, A. Nagata, and K. Aono, "A motion estimation processor for mpeg2 video real time encoding at wide search range," in *Proc. IEEE CICC*, 1995, pp. 405–408.
[14] J. F. Shen and L. G. Chen, "Low power full-search block-matching motion estimation chip for h.263+," in *Proc. IEEE Circuits and Systems*, vol. 4, 1999, pp. 299–302.