A Fast VLSI Architecture for Full-Search Variable Block Size Motion Estimation in MPEG-4 AVC/H.264

Minho Kim Ingu Hwang Soo-Ik Chae

School of Electrical Engineering
Seoul National University
Seoul 151-742, Korea
Tel : +82-2-880-5457
Fax : +82-2-888-1691
E-mail : {mhkim,ingu,chae}@sdgroup.snu.ac.kr

Abstract— We describe a fast VLSI architecture for full-search motion estimation for the blocks with 7 different sizes in MPEG-4 AVC/H.264. The proposed variable block size motion estimation (VBSME) architecture consists of a 16x16 PE array, an adder tree and comparators to find all 41 motion vectors and their minimum SADs for the blocks of 16x16, 16x8, 8x16, 8x8, 8x4, 4x8 and 4x4. It employs a 2-D datapath and its control of the search area data is simple and regular. The proposed VBSME can achieve 100% PE utilization by employing a preload register and a search data buffer inside each PE and allow real-time processing of 4CIF(704x576) video with 15 fps at 100 Mhz for a search range of [-32~+31].

I. INTRODUCTION

Nowadays, many international video compression standards such as ITU-T H.261, H.263, MPEG-1, -2, and -4 adopts motions estimation technique to reduce temporal redundancy between a current frame and its reference frames. The emerging MPEG-4 AVC/ITU-T H.264 standard supports motion estimation for the block of 7 different sizes: 16x16, 16x8, 8x16, 8x8, 8x4, 4x8, and 4x4 to improve coding performance, but its computational complexity becomes substantially higher. Thus, a fast architecture that can support motion estimation for all the 7 block sizes is essential for high-end real-time applications.

Many full-search motion estimation architectures were proposed so far, but there were only a few architectures that support motion estimation for variable block size [1-4]. Algorithms in [1] and [2] described 1-D array architectures, and that in [3] is about a 16x16 PE array for the MPEG-4 standard which supports ME only for the blocks of 16x16, 8x8 and 4x4. The algorithm in [4] described a 2-D array architecture with 1-D partial data reuse and 1-D data broadcasting. Note that we limit our discussion only for the full-search ME algorithm in this paper.

An 1-D array architecture is simple, easy to control, and occupy smaller area than a 2-D array architecture but searches only one row or column of the block at a time, so it is slower than a 2-D array architecture which computes the sum of absolute differences (SAD) of a search point at every cycle. Therefore, a 2-D array architecture is more suitable for high-end real-time video processing.

In the full-search 2-D array architecture, a processing element (PE) array computes the SADs between a pixel of the current frame and a pixel of one of its reference at every cycle, so the pixel data should be changed every cycle. According to the data flows, the 2-D array architectures can be divided into 3 classes: (1) The current frame pixel data are moving while the reference frame pixel data are fixed, (2) While the current frame pixel data are fixed, the reference frame pixel data are changed, (3) Both current and reference frame pixel data are changed.

In the VBSME architectures, the SADs of 4x4 blocks are first computed on a 16x16 macroblock, and the SADs with blocks of larger size are calculated by summing up the SADs of 4x4 blocks. For a 2-D array VBSME architecture, therefore, the class (1) is not suitable because the positions of the partition are changed if the current frame pixel data moves every cycle, which makes it difficult to sum up the SADs of the block of smaller size for those of larger size. The class (2) is also not suitable because it is hard to satisfy both high PE utilization and simple datapath requirements at the same time when the search area data moves both horizontally and vertically. In this paper we propose a fast architecture of the class (3) because it can achieve both high PE utilization and simple datapath.

II. PROPOSED ARCHITECTURE

Fig. 1 shows the block diagram of the proposed architecture, which consists of 4 basic blocks. The processing element array computes sixteen 4x4 SADs of a 16x16 macroblock. The adder & comparator block sums up the 4x4 SADs to form the SADs for 7 different block sizes and finds the minimum distortions and corresponding motion vectors. The search area SRAM contains the reference frame pixel data within a given search range to reduce I/O memory

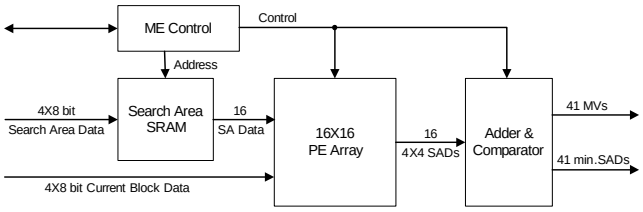

Fig. 1. Block diagram of the VBSME architecture

bandwidth. The ME control block generates the addresses of the search area SRAMs and the control signals to other blocks.

The dataflows of the search area data and the current block data are shown in Fig. 2(a) and 2(b) respectively. In order to operate all the 16x16 PEs every cycle, 16 pixel data from the search area must be supplied to the PEs at every cycle. We decided to shift the search area data only in the horizontal direction (right to left) and to supply the 16 search area data to the rightmost column of the 16x16 PEs from the search area SRAM. In the vertical direction, the current block data move down and wrap around.

For a search range of [-16~+15], the data sequence in the PE array is presented in TABLE I where C(x, y) is the pixel data in the current block and R(x, y) is the pixel data in the search area. During the clocks from 0 to 31, y coordinates of the search area pixels are fixed and only x coordinates are changed. When computations for one row are finished, the current block data shift down and wrap around by one row position and the initial search area data is loaded from the search data buffer (SDB) to the reference block register (RBR). Then the PE array starts to find the second row in the search area from the clock 32 without stall.

A PE consists of a absolute difference computing unit, a current block register (CBR), a RBR, a preload register (PR), a SDB and two multiplexers as shown in Fig. 3(a). At every clock, the absolute difference between the CBR and the RBR is computed. The SDB stores data in the initial search area data. If the current macroblock is the region 4 in Fig. 3(b), the shaded macroblocks 0, 3 and 6 will be the initial search area. Each SDB stores one pixel from each shaded macroblock. The search area SRAM stores the data in the remaining regions 1, 2, 4, 5, 7 and 8. When starting the computation for a new row, the multiplexer in the PE selects the initial data from the SDB so that we can compute the SADs without stall. Otherwise, the data from RBin are selected. The regions 1, 4 and 7, which will be the initial search area in the next macroblock, are stored in the SDB as soon as the computation for the regions 0, 3, and 6 is finished.

In addition, a preload register is put inside a PE to achieve macroblock pipelining [5]. When the current block is the region 4, the data of the next current block, which is the region 5, are transferred to the preload register in each PE. They are loaded to the CBR in all PEs immediately before starting computation for the next macroblock.

As the current block data move down in the PE array, the position of 4x4 sub-blocks also moves in the proposed architecture. To compute the 4x4 SADs correctly, we add a selector inside each 4x16 PE array to match results of 4x1 SADs with adder inputs. Four 4x16 PEs are connected to arrange the 16x16 PE array.

While computing for the second row of the search range, all the rows except the first row in the PE array are provided with the same search area data just like when the first row of the search range is computed. The search area data of the first row is supplied with the new data separated by 16 rows from the one previously supplied. Likewise, when changing the row position in a search area, we just change one address among 16 on-chip SRAM addresses, where the new address is pointing the row below by 16 rows. Therefore, each on-chip SRAM has only to store every 16th row of the search area data. That is, k-th (k=0,1,2,…,15) SRAM contains

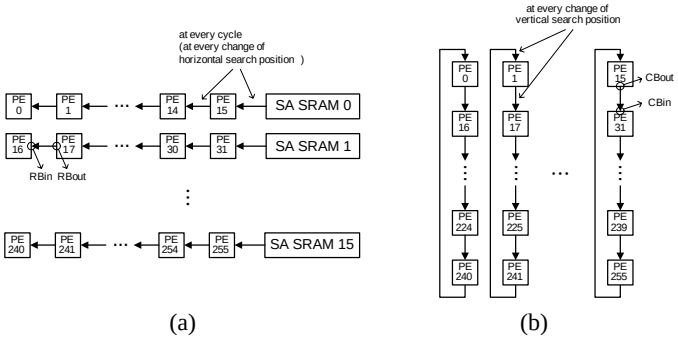

Fig. 2. (a) Dataflow of the search area data (b) Dataflow of the current block data

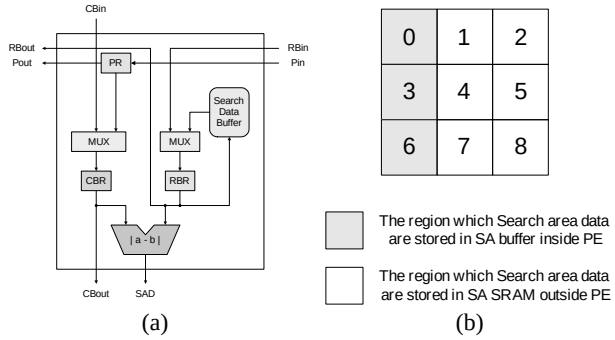

Fig. 3. (a) The PE structure (b) The region which the search area data are stored (when search range is [-16~+15])

TABLE I
DATA SEQUENCE OF THE CURRENT BLOCK DATA AND SEARCH AREA DATA (WHEN SEARCH RANGE IS [-16~+15])

Clock	1st row		⋯		2nd row		⋯		⋯	16th row		⋯	
	PE0	PE1		PE15	PE16	PE17		PE31		PE240	PE241		PE255
0	C(0,0)-R(-16,-16)	C(1,0)-R(-15,-16)		C(15,0)-R(-1,-16)	C(0,1)-R(-16,-15)	C(1,1)-R(-15,-15)		C(15,1)-R(-1,-15)		C(0,15)-R(-16,-1)	C(1,15)-R(-15,-1)		C(15,15)-R(-1,-1)
1	C(0,0)-R(-15,-16)	C(1,0)-R(-14,-16)		C(15,0)-R(0,-16)	C(0,1)-R(-15,-15)	C(1,1)-R(-14,-15)		C(15,1)-R(0,-15)		C(0,15)-R(-15,-1)	C(1,15)-R(-14,-1)		C(15,15)-R(0,-1)
⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯
31	C(0,0)-R(15,-16)	C(1,0)-R(16,-16)		C(15,0)-R(30,-16)	C(0,1)-R(15,-15)	C(1,1)-R(16,-15)		C(15,1)-R(30,-15)		C(0,15)-R(15,-1)	C(1,15)-R(16,-1)		C(15,15)-R(30,-1)
32	C(0,15)-R(-16,0)	C(1,15)-R(-15,0)		C(15,15)-R(-1,0)	C(0,0)-R(-16,-15)	C(1,0)-R(-15,-15)		C(15,0)-R(-1,-15)		C(0,14)-R(-16,-1)	C(1,14)-R(-15,-1)		C(15,15)-R(-1,-1)
33	C(0,15)-R(-15,0)	C(1,15)-R(-14,0)		C(15,15)-R(0,0)	C(0,0)-R(-15,-15)	C(1,0)-R(-14,-15)		C(15,0)-R(0,-15)		C(0,14)-R(-15,-1)	C(1,14)-R(-14,-1)		C(15,15)-R(0,-1)
⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯
63	C(0,15)-R(15,0)	C(1,15)-R(16,0)		C(15,15)-R(30,0)	C(0,0)-R(15,-15)	C(1,0)-R(16,-15)		C(15,0)-R(30,-15)		C(0,14)-R(15,-1)	C(1,14)-R(16,-1)		C(15,15)-R(30,-1)
⋯					⋯					⋯			
992	C(0,1)-R(-16,16)	C(1,1)-R(-15,16)		C(15,1)-R(-1,16)	C(0,2)-R(-16,17)	C(1,2)-R(-15,17)		C(15,2)-R(-1,17)		C(0,0)-R(-16,15)	C(1,0)-R(-15,15)		C(15,0)-R(-1,15)
993	C(0,1)-R(-15,16)	C(1,1)-R(-14,16)		C(15,1)-R(0,16)	C(0,2)-R(-15,17)	C(1,2)-R(-14,17)		C(15,2)-R(0,17)		C(0,0)-R(-15,15)	C(1,0)-R(-14,15)		C(15,0)-R(0,15)
⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯
1023	C(0,1)-R(15,16)	C(1,1)-R(16,16)		C(15,1)-R(30,16)	C(0,2)-R(15,17)	C(1,2)-R(16,17)		C(15,2)-R(30,17)		C(0,0)-R(15,15)	C(1,0)-R(16,15)		C(15,0)-R(30,15)

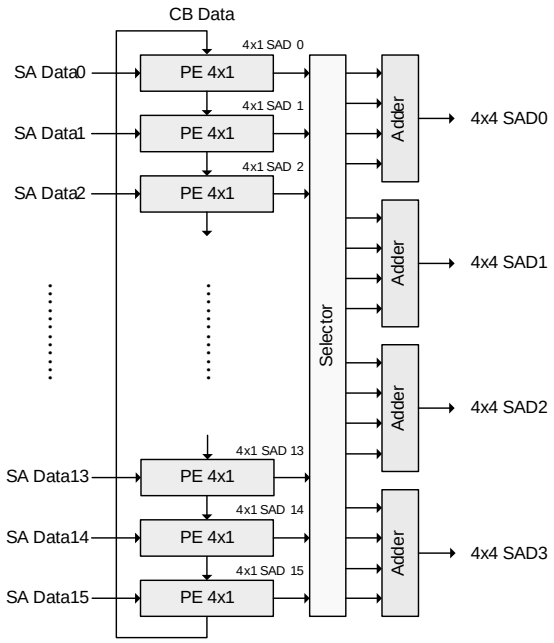

Fig. 4. The architecture of the 4x16 PE array

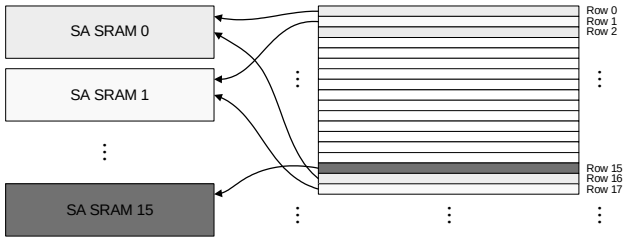

Fig. 5. Data transfer of reference frame data to search area SRAM

(16i+k)-th row data in the search area (i: nonnegative integer) as shown in Fig. 5. It can be done easily by manipulating address generation.

Sixteen 4x4 SADs, which are the outputs of a 16x16 PE array, are inputted to an adder & comparator block. Before adding them up, the 16 4x4 SADs are stored in the temporal registers and an adder tree sums them up to produce 8x4, 4x8, 8x8, 16x8, 8x16 and 16x16 SADs. Comparators compare total 41 SADs and save the 41 minimums with their corresponding motion vectors. They can be used at the rate-distortion optimization stage to find the best block mode.

It takes only one cycle to compute the absolute differences for each search position so finding 4x4 SADs for the search range of [-16~+15] can be performed in 1024 cycles as shown in TABLE I. If we add one more cycle for the adder tree delay to obtain the SADs of larger sizes, the total number of cycles required to finish the computation becomes 1025 for one macroblock. Therefore, for the search range of [-32~+31], the number of the clock cycles required is 64x64 + 1 = 4097.

III. IMPLEMENTATION

The proposed architecture was implemented with a VHDL description and synthesized by Synopsys Design Compiler

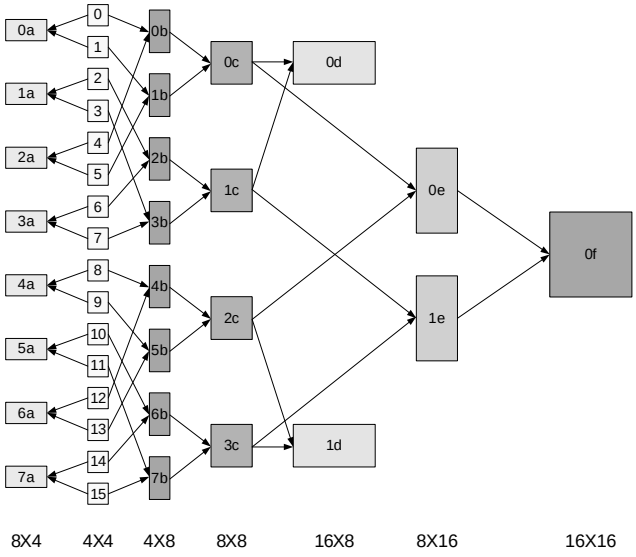

Fig. 6. Block diagram of the adder tree

with TSMC 0.18um standard cell library. Our design has a search range of [-32~+31], and it requires 10Kb on-chip memory as the search data buffer inside a PE and 40Kb as the search area SRAM and an additional 10 Kb as the search area SRAM for the next search area so the total 60Kb on-chip memory is required. If we search [-16~+15] range, only 24Kb on-chip memory is required.

Our design contains total 154k gates. The gate counts of the blocks are 113.4k for the PE array, 22.1k for the ME control block, and 18.5k for the adder & comparator block.

The search range of our ME can be selected to be either [-16~+15] or [-32~+31]. When the search range is [-16~+15], it becomes 4 times faster than when the search range is [-32~+31]. TABLE II illustrates the summary of our ME design.

The comparison of our proposed architecture with other full-search VBSME architectures is presented in TABLE III. This table shows that our architecture is faster for the same search range, requires less on-chip memory, and has more flexible and wider search range.

TABLE II
DESIGN SPECIFICATION

Algorithm	Full Search
# of PE	16X16 (2-D array)
Search Range	32X32, 64X64 (flexible)
Block size	4X4, 4X8, 8X4, 8X8, 8X16, 16X8, 16X16
Process	TSMC 0.18um standard cell library
Gate Count	154k
On-chip memory	60 kbits
Max Freq.	100Mhz
Throughput (search range 64X64)	4CIF 15 fps

TABLE III
COMPARISON OF VBSME ARCHITECTURES

	[1]	[3]	[4]	Ours
# of PE	16	16X16	16X16	16X16
Search Range	32X32 16X16	64X64	48X32	64X64 32X32
Block size	7 kinds of block size	16X16, 8X8, 4X4	7 kinds of block size	7 kinds of block size
Process	0.13um	0.5um	0.35um	0.18um
Gate Count	108k	-	106k	154k
On-chip memory	-	96k bits	24k bits	60k bits
Max Freq.	100Mhz	100Mhz	66.67 Mhz	100Mhz
Throughput(blocks/sec) (search range 32X32)	5560	-	61218	97560
Throughput(blocks/sec) (search range 64X64)	-	23668	-	24408

IV. CONCLUSION

This paper presents a new fast VLSI architecture for VBSME in MPEG-4 AVC/H.264. It is based on 2-D array architecture and computes the SADs for the blocks of all the 7 different sizes. The search area data moves only horizontally, and the current block data moves only vertically to simplify the dataflow. It achieves 100% PE utilization and macroblock pipelining by employing 16 on-chip SRAMs and search data buffers inside each PE. For the search range of [-32~+31], our implementation allows variable block size motion estimation of 4CIF (704x576) video with 15 fps at 100 Mhz.

REFERENCES

[1] Swee Yeow Yap, John V. McCanny, "A VLSI architecture for advanced video coding motion estimation," *Proc. IEEE International Conference on Application-Specific Systems, Architectures, and Processors (ASAP'03)*, June 24-26, 2003.

[2] Cao Wei, Mao Zhi Gang, "A novel SAD computing hardware architecture for variable-size block motion estimation and its implementation with FPGA," *Proc. 5th international conference on ASIC*, Oct 21-24, 2003.

[3] P. M. Kuhn, A. Weisgerber, R. Poppenwimmer, and W. Stechele, "A flexible VLSI architecture for variable block size segment matching with luminance correction," *IEEE International conference on Application-specific Systems, Architectures, and Processors (ASAP 97)*, Zurich, 1997.

[4] Yu-Wen Huang, Tu-Chih Wang, Bing-Yu Hsieh, and Liang-Gee Chen, "Hardware architecture design for variable block size motion estimation in MPEG-4 AVC/JVT/ITU-T H.264," *Proc. IEEE International Symposium on Circuits and Systems(ISCAS 2003)*, Bangkok, Thailand, May 2003.

[5] Jen-Chieh Tuan, Tian-Sheuan Chang, "On the data reuse and memory bandwidth analysis for full-search block-matching VLSI architecture," *IEEE Transactions on circuits and systems for video technology*, vol.12, no.1, Jan. 2002.