

Semi-Systolic Array Based Motion Estimation Processor Design*

Mei-Cheng Lu

Chen-Yi Lee

Dept. of Electronics Engineering, National Chiao Tung University,
1001, University Road, Hsinchu 300, Taiwan, ROC
Tel: 886-35-712121 ext.54149; Email: cylee@cc.nctu.edu.tw

ABSTRACT

This paper presents a new VLSI architecture for full-search block matching algorithm. The proposed architecture has two specific features: (1) it has a processor element (PE) array which provides sufficient computational power, where PE's work in a semi-systolic style and (2) it contains stream memory banks which provide scheduled data flow to reduce idle operations within PE array. By exploiting broadcasting and local data communications, hardware efficiency of the proposed architecture can be up to 100%, which outperforms those systolic-array solutions found in the literature.

1. INTRODUCTION

Many architectural solutions for implementing block matching algorithm (BMA) can be found in the literature [1,2,3,4,5,6]. Most of the solutions are focused on the data flow within processor element (PE) array. Therefore systolic array approach has been highly exploited in VLSI implementation. However, this approach causes some problem in data flow outside PE array. In other words, too much overhead on memory bandwidth is requested to provide a scheduled data sequence in order to meet the need of PE array. Therefore large number of I/O pins is needed, resulting in higher packaging cost. In addition, due to pipeline filling at the boundary of search area, hardware efficiency, which can be expressed by $E_h = (\frac{2P+1}{2P+N})^2$, is degraded a lot. For example, if $P=N=8$, only 50% of process are working on the candidate blocks. Although, in [5,6], the authors proposed a snake-like data stream format which can reduce the I/O bandwidth problem, the hardware efficiency still remains very low.

In this paper, we propose a semi-systolic array to improve the low efficiency problem as found in systolic array solutions. Instead of local connections of search data flow, we use a global distribution of search data connected to each PE row (or column). The partial sum is locally connected. With this style, it has been proved that hardware efficiency up to 100% can be achieved if a dedicated memory management unit is supported. Section 2 describes how the general BMA

algorithms can be mapped onto the proposed semi-systolic array or SSA architecture. Section 3 presents the memory management strategy in order to offer the scheduled data sequence so that 100% efficiency can be achieved in PE array. Finally a demonstrator design of motion estimation processor for $N=P=16$ is described.

2. MAPPING BMA ONTO SEMI-SYSTOLIC ARRAY ARCHITECTURE

The basic structure of the SSA is shown in Figure 1. In this structure, the connections are divided into two types— one is broadcasting or global distribution type and the other is local type. For broadcasting type, input data is fed in from the stream memory and connected to all PEs of the same column (row). For local type, results obtained from the higher (left) PEs are pumped into next lower (right) PEs for further processing.

To illustrate how full search motion estimation can be mapped onto SSA architecture, we use an example of a 3×3 reference block with search area of 7×7 .

First we assume that reference data have been stored in each PE, then search data are pumped out from the stream memory and broadcast to PEs which perform absolute mean calculation and partial sum accumulation. With the latency of 6 cycles, the first distortion comes out from the bottom right cell (ACC). Then the distortion values of the rest candidates are obtained sequentially. However, when boundary is detected, all PEs become idle since data of the next row (column) have to be filled to the pipeline.

This low efficiency can be overcome by preloading data on next row before boundary is detected. As shown in Figure 2(a), when the distortion calculation is done on the boundary, data of the next row should be pumped into PE array at the next cycle. The mask region indicates that these data should be simultaneously pumped into the PE-array. Figure 2(b) illustrates the process of distortion calculation at boundary. At cycle 5, boundary is detected. Summation of the distortion values of different candidates are still performed at $PE_{1,j}$, $j=1, 2, 3$. At cycle 6, the boundary block still needs the boundary data (label 16) which should be pumped into PE2 and PE3. However, data of the next row (i.e. label 21) should be pumped into PE1. At cycle 7, data items (label 17 and 22) are needed. Distortion calculation of the candidate on the boundary is achieved now in PE3. In the meantime, dis-

*Work supported by the National Science Council of Taiwan, ROC, under Grant NSC84-2213-E009-115

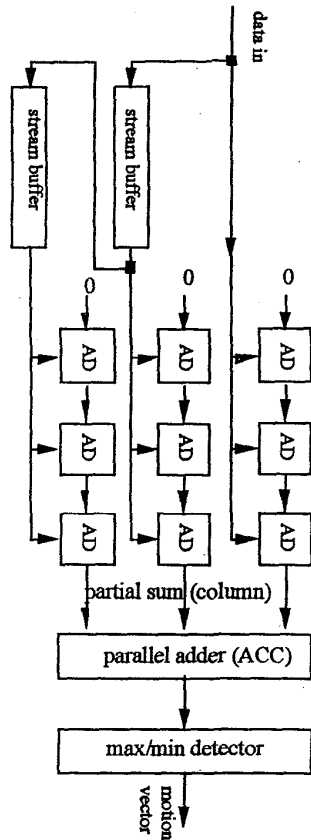


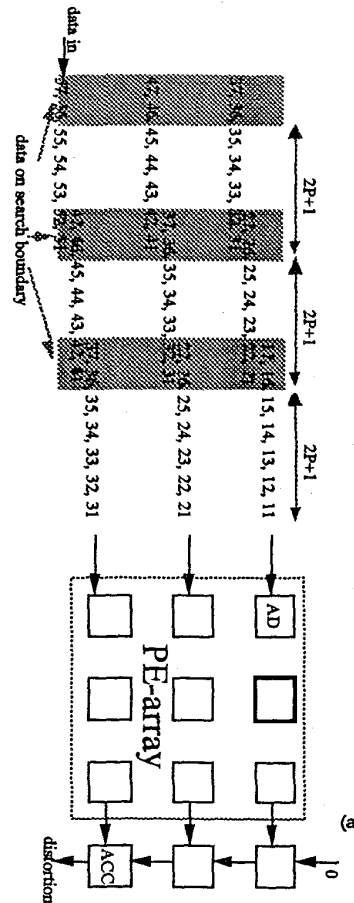
Fig. 1. Illustration of the semi-systolic array architecture.

tortion of the candidates of the next row are also performed in PE2 and PE1. At cycle 8, only the data items from the same row are needed. Therefore the BMA can be mapped onto the semi-systolic array, where 100% hardware efficiency within PE-array can be achieved.

3. MEMORY MANAGEMENT

When candidate blocks are not within the boundary area, only one single data stream is needed for all PEs on the same row. However when boundary criteria is detected, two data streams are needed. This implies that a two-read-port memory is needed. In addition, the data items fetched from the current stream memory have to be loaded into next stream memory. Therefore a two-write-port memory is also needed. As a result from these read/write considerations, it is necessary to provide a 2-port memory with size of $(N-1) \times (2P+1)$ for non-boundary data and a 4-port memory with size of $(N-1) \times (N-1)$ for boundary data. However, since storage space is only activated once at a certain time interval, the 4-port memory devices can be reduced to 2-port memory with the constraint that these $(N-1) \times (N-1)$ boundary data should use different read/write ports as those for non-boundary data.

We still have to consider the problem of data initialization since this problem may cause idle operations within the PE ar-



at cycle:	output from each PE (only the 1st row is shown)		
	PE1	PE2	PE3
5	(15-11),	(15-12)+(14-11),	(15-13)+(14-12)+(13-11)
6	(21-11),	(16-12)+(15-11),	(16-13)+(15-12)+(14-11)
7	(22-11),	(22-12)+(21-11),	(17-13)+(16-12)+(15-11)
8	(23-11),	(23-12)+(22-11),	(23-13)+(22-12)+(21-11)

Fig. 2. (a) Improve hardware efficiency using multiple inputs so that search data can be filled in the PE array and (b) shows an example how hardware efficiency can be improved by parallel ports when boundary is detected.

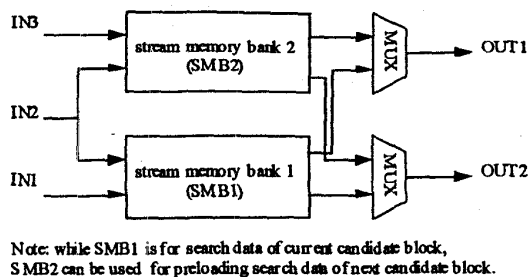


Fig. 3. Organization of the stream memory banks. Note that these two identical banks are working interleavely to reach 100% efficiency.

ray and hence 100% efficiency cannot be achieved. In the previous discussion, we first assumed that reference data ($N \times N$) and part of search data $(N-1) \times (2P+N)$ are preloaded into PE array and stream memory banks respectively. However, at $(2P+1)^2$ cycle, distortion calculation for the last candidate block will be finished except the latency inherent in pipeline delay. If we have to fill in the stream memory banks, then all PEs are idle for at least $(N-1) \times (2P+N)$ cycles. Therefore we propose to use interleaving stream memory banks, i.e. there are two identical stream memory bank units which are working on different search area as shown in Figure 3. While stream memory bank 1 (SMB1) is for current search area data, SMB2 can be used to preload part of the search data, i.e. $(N-1) \times (2P+N)$, for calculating next motion vector. Thus once distortion calculation for the last candidate block of current candidate block is done, we can immediately switch from SMB1 to SMB2 for calculating next motion vector if next reference data are already available.

Moreover to ensure that reference data are already available when motion estimation is activated for another reference block, we need one $N \times N$ shift register array (SRA). However, this SRA is also needed for matching partial sum sequence to get final distortion value. Since one SRA cannot be shared for delay management and storing reference data, we use two SRA's which are interleaved as shown in Figure 4.

Based on this organization, the total memory space needed is $2(N-1) \times (N+2P) + 2(N \times N)$, where the former part is for search data and the latter part is for both delay management and reference data.

4. THE DEMONSTRATOR DESIGN

Floorplan of the ME processor is shown in Figure 5. Based on the proposed architecture, the area for a motion estimation processor with $N=P=16$ is about $9.5 \times 7.2 \text{ mm}^2$ for a $0.8 \mu\text{m}$ CMOS double metal technology [7]. The critical path has been limited to 10ns to meet the requirements of MPEG2 Main profile at main level [8]. This chip has been assembled and under fabrication currently. Figure 6 shows the final layout of the ME processor.

Table 1 shows the performance comparison with other available architectures. The reference size of 8×8 is selected as a platform for comparison, where search area is 23×23 and PE-

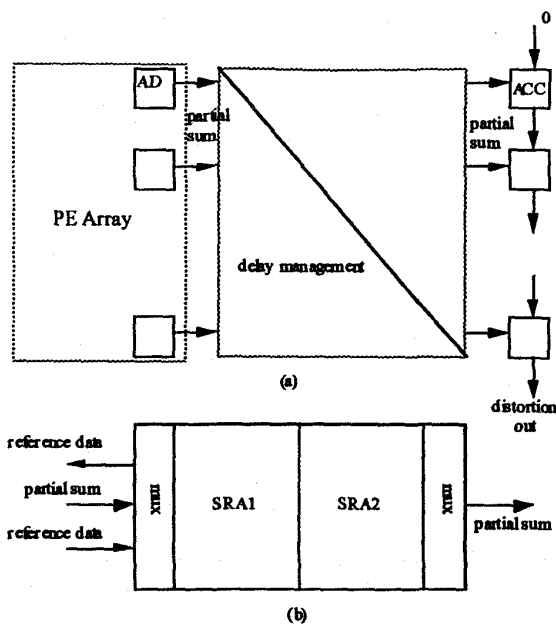


Fig. 4. The shift register array is allocated on the other side of PE array for two purposes: (a) delay matching and (b) preloading reference data.

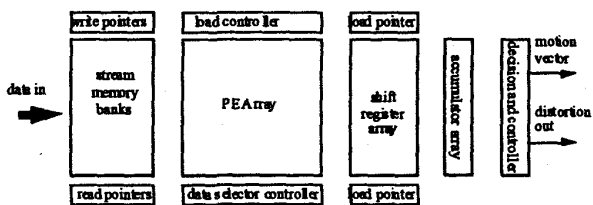


Fig. 5. Block diagram of the motion estimation processor with $N=P=16$.

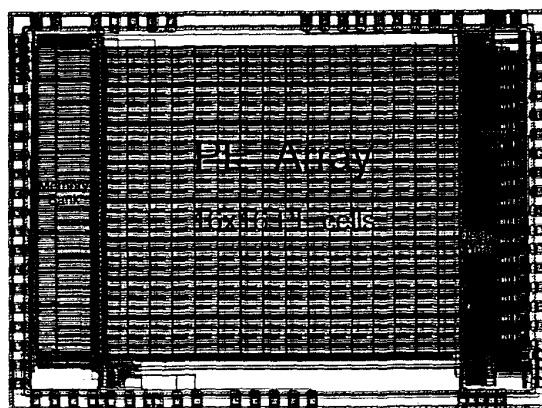


Fig. 6. Chip plot of the SSA-based motion estimation processor design with $N=P=16$.

Table 1. Performance comparison of different VLSI architectures for motion estimation processor design.

company	cycle-count	memory size	hardware efficiency
SGS-Thomson[9]	432	576	59.30%
LSI Logic[10]	480	576	53.30%
Hsieh et. al[5]	576	105	44.40%
our approach	256	464	100%

count is 8X8. It can be found that our proposed architecture can produce a motion vector every 256 cycles, which is the minimum among the four architectures. Although, the memory size used in our proposal is about 4 times of [5], it is still less than the other two [9,10]. Besides, we have found that the PE array occupies more than 70% area in physical design. This implies that the memory size is not a key issue in VLSI implementation for our proposed architecture.

In addition, the SSA architecture also has the following features:

- simple control flow: the required control signals for each module are rather simple and can easily be derived. For example, the *load* used in PE array is only needed when the start of a new motion vector is requested.
- selection of different displacements (P): this can be done by adjusting the read/write pointers at the stream memory banks, where the cycle count for calculating each motion vector is $(2P+1)^2$. In our demonstrator design, 3 different displacement codes (4, 8, 16) are allowed.
- selection of different sizes of reference block: this can be done by adjusting the position of the input port of the stream memory banks.

5. CONCLUSION

In this paper, we have presented a novel VLSI architecture for optimally implementing full-search motion estimation algorithm. The proposed architecture mainly consists of (1) PE-array which is a semi-systolic array structure to offer computation power and (2) memory management unit which offers a scheduled data flow so that 100% hardware efficiency within the PE-array can be achieved. In addition, this proposed architecture is also flexible in selecting the sizes of reference and search blocks. The architecture has also been demonstrated by a full-search motion estimation processor for $N=P=16$. We are currently investigating the possibility of applying this novel architecture to other motion estimation algorithms, such as 3-step search and telescopic search methods.

Acknowledgement: The authors would like to thank their colleagues within the SI2 group of NCTU for many discussions and fruitful suggestions.

REFERENCES

[1] K. Komarek and P. Pirsch, "Array Architectures for Block Matching Algorithms", IEEE Trans. on Circuits and Systems, Vol. 36, No. 10, Oct. 1989, pp. 1301-1308.

[2] R.C. Kim and S.U. Lee, "A VLSI Architecture for a Pel Recursive Motion Estimation Algorithm", IEEE Trans. on Circuits and Systems, Vol. 36, No. 10, Oct. 1989, pp. 1291-1300.

[3] L. De Vos and M. Stegherr, "Parameterizable VLSI Architectures for the Full-Search Block Matching Algorithm", IEEE Trans. on Circuits and Systems, Vol. 36, No. 10, Oct. 1989, pp. 1309-1316.

[4] K.M. Yang, M.T. Sun, and L. Wu, "A Family of VLSI Designs for the Motion Compensated Block-Matching Algorithm", IEEE Trans. on Circuits and Systems, Vol. 36, No. 10, Oct. 1989, pp. 1317-1325.

[5] C.H. Hsieh and T.P. Lin, "VLSI Architecture for Block-Matching Motion Estimation Algorithm", IEEE Trans. on Circuits and Systems for Video Technology, Vol. 2, No. 2, June 1992, pp. 169-175.

[6] S.I. Uramoto, A. Takabatake, M. Suzuki, H. Sakurai, and M. Yoshimoto, "A Half-Pel Precision Motion Estimation Processor For NTSC-Resolution Video", In IEEE 1993 Custom Integrated Circuits Conference, San Diego, CA, May 9-12, pp. 11.2.1-11.2.4

[7] M.C. Lu, "Data Flow Oriented Parallel Architectures for Block Matching Motion Estimation Algorithm", NCTU/DEE Master Thesis, Hsinchu, Taiwan, June 1994.

[8] "International Organization for Standardization Coding of Moving Pictures and Associated Audio", ISO/IEC JTC1/SC29/WG11/N0702, March 25, 1994.

[9] "STV3220 Motion Estimation Processor", SGS-Thomson Microelectronics Data Book, May 1989.

[10] "L64720 Video Motion Estimation Processor", LSI LOGIC Data Book, Sep., 1991.