A HIGH-THROUGHPUT, FLEXIBLE VLSI ARCHITECTURE FOR MOTION ESTIMATION

Chin-Liang Wang, Ker-Min Chen, and Jin-Min Hsiung
Department of Electrical Engineering, National Tsing Hua University
Hsinchu, Taiwan 300, Republic of China

ABSTRACT

This paper presents a new systolic VLSI architecture to realize the full-search block matching algorithm for motion estimation. The architecture has an efficiency of 100 percent and a throughput of one motion vector per n^2 cycles, where n×n is the reference block size. As compared to existing VLSI motion estimators with the same efficiency and throughput, the proposed one not only gains advantages in the flexibility of changing the reference block size and the tracking range, but also employs no additional control circuitry to determine the motion vectors. These features make it useful for a wide range of applications.

I. INTRODUCTION

The motion estimation using the full-search block matching algorithm (FBMA) has been widely used in image/video signal processing applications, such as motion compensated interframe coding [1] and signal interpolation [2]. Since the FBMA-based motion estimation involves a rather high computational load, high-speed VLSI structures are necessary to meet the real-time requirements. Moreover, the reference block size and the tracking range used in motion estimation are dependent on applications; therefore, it is worth designing a VLSI motion estimator with a flexibility in change of these two parameters.

A number of VLSI architectures have been proposed for realizing the FBMA-based motion estimation (see, for example, [3]-[7]). Each of them has its own distinct features which make it suitable for some specific applications. For example, the architecture in [3] has a flexibility in change of the reference block size (n×n) and the tracking range or maximum displacement (m), but its processing speed is not fast enough for high-speed applications such as high-definition television (HDTV). In contrast, the architectures in [4] and [5] reach higher throughput rates without the flexibility.

In this paper, we present a new systolic architecture to realize the FBMA for motion estimation. The architecture is highly regular, modular, and thus well suited to VLSI implementation. It is 100 percent efficient in hardware utilization and yields results at a rate of one motion vector per n^2 cycles. As compared to existing VLSI motion estimators with the same efficiency and throughput performance, the proposed one has the following advantages:

1) It can easily be reconfigured for various reference block sizes (n×n, 2n×2n, 4n×4n, ...) via simple control.
2) It can be used for various tracking ranges. If the maximum displacement allowed in the basic chip is m, then a larger tracking range of m'=2m, 4m, 8m, ... can be obtained by interconnecting an appropriate number of basic chips.
3) It does not need additional control circuitry to determine the motion vectors

II. A VLSI MOTION ESTIMATOR

Let the reference block be of size n×n and the search area be of size (n+2m-1)×(n+2m-1), where the displacement allowed is from -m to m-1 in the horizontal and vertical directions. Then the FBMA using the mean-absolute difference (MAD) criterion involves the following computations for each block matching process:

$$MAD(i,j) = \sum_{x=0}^{n-1}\sum_{y=0}^{n-1}\left|C(x,y) - P(i+x, j+y)\right|, \quad -m \le i,j \le m-1 \qquad (1)$$

$$(i^*, j^*) = Min^{-1}[MAD(i,j)] \qquad (2)$$

where $\{C(x,y)\}$ represent the pixel values of the reference block in the current frame and $\{P(x+i,y+j)\}$ represent the pixel values of the candidate block with displacement vector (i,j) within a search area in the previous frame. It is easy to see from (1) that there are $4m^2$ MAD's to be computed for each reference block. The desired motion vector (i^*,j^*) is the displacement vector of the candidate block that has the minimum MAD.

A. The Module of MAD Calculator

For the case of n=4 and m=2, the size of the search area is 7×7 and there are 16 distance measures for each reference block. The pixels of the reference block are represented by $\{C(x, y) \mid 0 \le x, y \le 3\}$ and those of the search area are represented by $\{P(k, l) \mid -2 \le k, l \le 4\}$, as shown in Figs.1 (a) and (b). Fig.1(c) describes a 2-D systolic array with the required input data flow for the MAD computation. It consists of 2m×2m TYPE-1 cells, where each cell calculates one of the 16 MAD's required in a block matching process. Fig.2 shows the function of the TYPE-1 cell. The pixel data of the reference block and those of a candidate block in the search area are fed to the TYPE-1 cell in serial form. Therefore, the TYPE-1 cell will produce one MAD every n^2 cycles. Let the data enter the cell at the 1st cycle. Then, at the n^2 cycle, the MAD is generated and sent to the next cell via the multiplexer (MUX). To control the operation of the MUX, a pattern CTRL of one 1 followed by n^2-1 0's is fed to each TYPE-1 cell. Note that this pattern must be synchronized with the reference block data $C(x,y)$, as shown in Fig.1(c). Each vertical stream of $P(x+i-2,y+j-2)$ is pipelinedly shifted downwards through the 2-D array, while the current block data $C(x,y)$ are serially fed to each row of the 2-D array, where the sequence of a given row is staggered by n+1 cycles relative to that of the row above.

There is also a feature of overlap between the search areas for two consecutive reference blocks. The first data of the next reference block and the first data P(2,1) of the corresponding search area will reach the (-2,1) TYPE-1 cell n^2 cycles after the first data of the current reference block enters the (-2,1) TYPE-1

0-7803-2431-5/95 $4.00 © 1995 IEEE

cell. It is easy to see that the candidate MAD's are generated at a rate of one MAD per n^2 cycles for each TYPE-1 cell. All the cells are busy all the time, so the efficiency is 100 percent. The MAD's will emerge from the array of Fig.1(c) in a format as shown in Fig.3.

B. The Module of Comparator

To determine the minimum MAD and its associated index (i.e., the motion vector), the array shown in Fig.3 can be used. This array is composed of 2m TYPE-2 cells and one TYPE-3 cell. The functions of these two basic cells are shown in Figs. 4 and 5. The column of TYPE-2 cells is used to pass all candidate MAD's produced by TYPE-1 cells. Also, it labels each MAD with a vector index (i,j), where the i component is stored as a constant for each TYPE-2 cell and the j component is generated by a counter. Note that the output sequence of a given row of the MAD calculator is n+1 cycles ahead compared with that of the row below. Therefore, if $n \geq 2m$ is satisfied, the 2m MAD's entering from a row can be moved down through the column of TYPE-2 cells without overlapping to another 2m MAD's from the row below. To make each TYPE-2 cell pass the 2m MAD's of the corresponding row properly, we need a control signal SW that becomes 1 for 2m cycles whenever CTRL=1 occurs at the cell and becomes 0 otherwise. If SW=1, the TYPE-2 cell passes MAD's with the associated vector indices from the corresponding row via the MUX; otherwise, it passes MAD's with the associated vector indices from the preceding TYPE-2 cells. The TYPE-3 cell is used for comparing the candidate MAD's to find the minimum one and the corresponding motion vector. We can see that one motion vector will be generated every n^2 cycles. It is worth mentioning that, unlike existing systems, the proposed architecture does not need additional control circuitry to determine the motion vector.

C. I/O Schemes

In order to arrange the data flow P(x,y) of the search area shown in Fig.1(c), the circuit shown in Fig.6(b) is employed, where n=4 and m=2. The search area data are divided into two portions, namely P1 and P2. They are the results of interlaced scanning of the search area as shown in Fig.6(a). To make P1 and P2 form the required data flow shown in Fig.6(b), a control pattern CS of alternating n 1's and n 0's is used. This arrangement of data flow has a constraint of $n \geq 2m-1$, which is included in the constraint given by TYPE-2 cells that $n \geq 2m$. To produce the data flow C(x,y) of the reference block shown in Fig.1(c), one shift registers (SR) of length n+1 can be placed between adjacent rows for delaying the sequence of reference block data. With these schemes, the system will have a reasonable number of I/O pins.

D. Flexibility In the Reference Block Size and Tracking Range

The size of the array for MAD calculation is 2m×2m, which is apparently independent of the reference block size n. Thus, for different reference block sizes, we need only to change the length of shift registers and adjust the search area input format. That is, for a given m, the architecture is flexible in change of n under the constraint of $n \geq 2m$ when single-chip implementation is employed. To simplify the design, we use a programmable shift register that can easily be programmed for the following reference block sizes: n×n, 2n×2n, 4n×4n, and so on.

Assume that the proposed motion estimator with a reference block size n and a maximum displacement m is realized as a basic chip. Then we may obtain a larger tracking range of m'=2m, 4m, 8m, ... by interconnecting an appropriate number of basic chips to form the complete system. Note that n<m' is allowed in this case. Fig.7 shows an example that realizes a motion estimator with m=2 using four basic chips with m=1 each. For multichip implementation, the TYPE-3 cell should be modified as shown in Fig.8 such that it can handle comparison of MAD's between adjacent chips in the horizontal direction. Also, an additional TYPE-4 cell (shown in Fig.9) is used for handling comparison of MAD's between adjacent chips in the vertical direction.

III. COMPARISON WITH EXISTING SYSTEMS

From the criteria given by Komarek and Pirsch [4], a motion estimator can work in real time if and only if

$$f_C \geq f_P \times C_P \equiv f_{C(\min)} \tag{3}$$

where f_C, f_P, and C_P represent the system clock rate, the pixel rate, and the average number of clock cycles per pixel, respectively. Assume that the pixel rate required is 80 MHz, each reference block is of size 32×32, and the displacement allowed is from -16 to +15 in the horizontal and vertical directions. Then we can compare the proposed architecture with some existing systems in terms of the throughput, C_P, efficiency, flexibility in changing m and n, and $f_{C(min)}$. The comparison results are listed in Table I. We can observe from this table that the proposed architecture is better than the others in terms of the features compared.

IV. CONCLUSIONS

In this paper, we have proposed a new systolic implementation of motion estimators based on the full-search block matching algorithm. The system possesses a high degree of regularity and modularity, and is thus well suited to VLSI implementation. It has an efficiency of 100 percent and yields one motion vector every n^2 cycles, where n×n is the reference block size. As compared to the previous VLSI motion estimators with the same efficiency and throughput, the proposed one gains improvements in the flexibility that various block sizes (n×n, 2n×2n, 4n×4n, ...) can be chosen via simple control and motion estimation for various tracking ranges (m, 2m, 4m, ...) can be realized by cascading an approximate number of basic chips. In addition, it does not require additional control circuitry to determine the motion vectors. A prototype chip of the proposed motion estimator for n=8 and m=4 has been implemented in a 0.8 μm CMOS technology. The chip requires a die size of about 7×7 mm^2 and is able to operate at a clock rate up to 100 MHz. Such area-time performance makes it attractive for use in high-speed applications such as HDTV.

ACKNOWLEDGMENT

This work was supported by the National Science Council of the Republic of China under Grant NSC 83-0404-E007-012.

REFERENCES

[1] 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.

[2] A. Furukawa, T. Koga, and K. Linuma, "Motion-adaptive interpolation for videoconference pictures," in Proc. ICC'84, vol. 2, May 1984, pp. 707-710.

[3] K. M. Yang, M. T. Sun, and L. Wu, "A family of VLSI designs for the motion compensation block-matching algorithm," IEEE Trans. Circuits Syst., vol. 36, pp. 1317-1325, Oct. 1989.

[4] T. Komarek and P. Pirsch, "Array architectures for block matching algorithms," IEEE Trans. Circuits Syst., vol. 36, pp. 1301-1308, Oct. 1989.

[5] L. D. Vos and M. Stegherr, "Parameterizable VLSI architectures for the full-search block-matching algorithm," IEEE Trans. Circuits Syst., vol. 36, pp. 1309-1316, Oct. 1989.

[6] A. Artieri and F. Jutand, "A versatile and powerful chip for real time motion estimation," in Proc. Int. Symp. Circuits Syst., 1989, pp. 2463-2456.

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

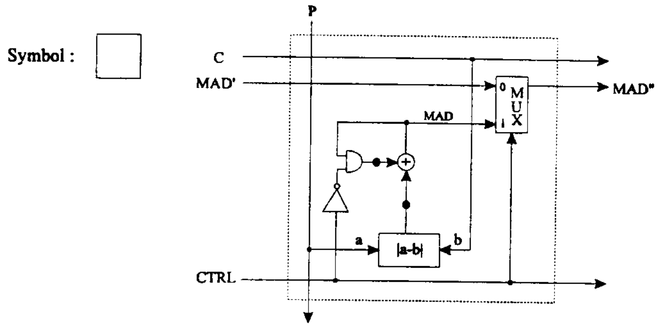


Fig.2. TYPE-1 cell.

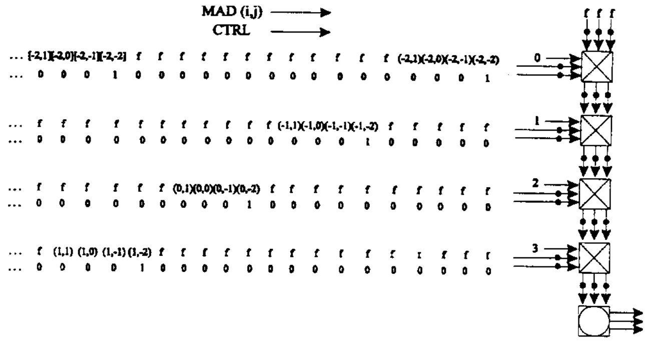


Fig.3. A systolic array for determining the minimum MAD.

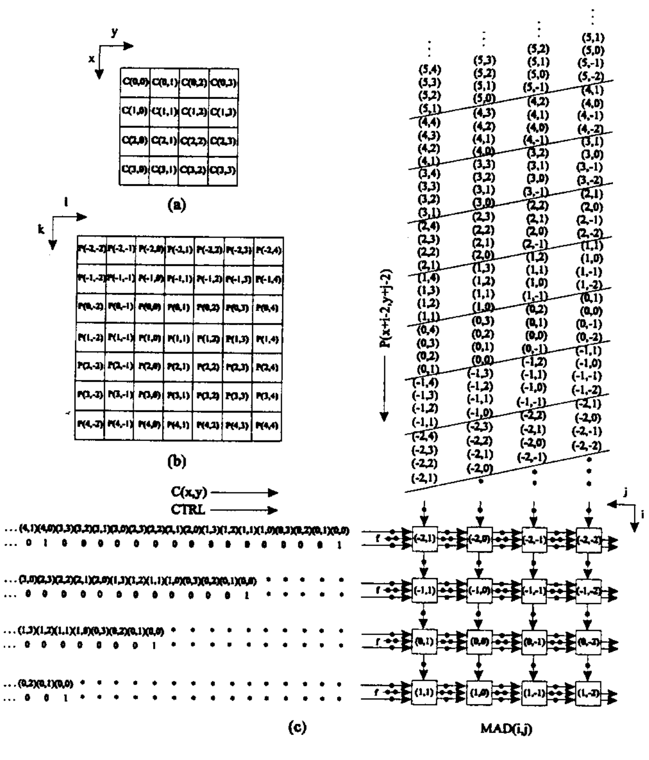
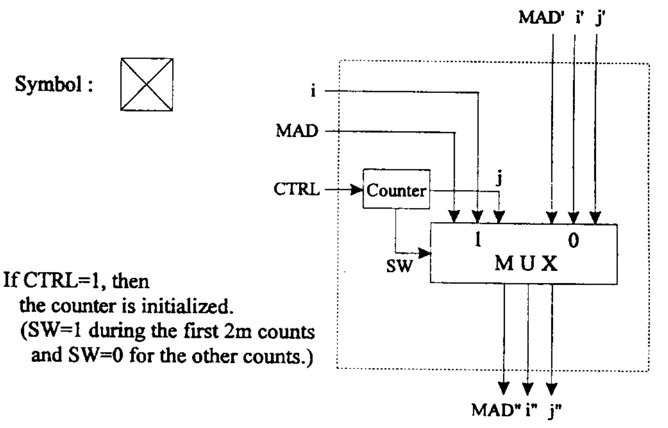


Fig.1. A systolic array for computing MAD's.



Symbol :
MAD' i' j'
i
MAD
CTRL
Counter
j
SW
1
0
MUX
If CTRL=1, then the counter is initialized. (SW=1 during the first 2m counts and SW=0 for the other counts.)
MAD" i" j"

Fig.4. TYPE-2 cell.

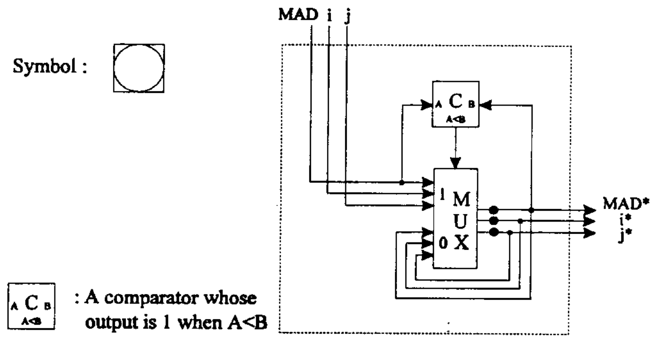


Fig.5. TYPE-3 cell.

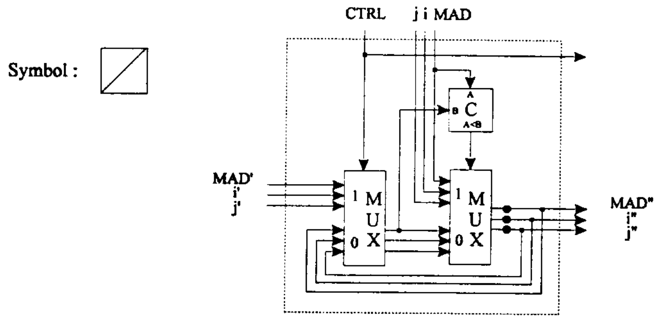


Fig.8. Modified TYPE-3 cell.

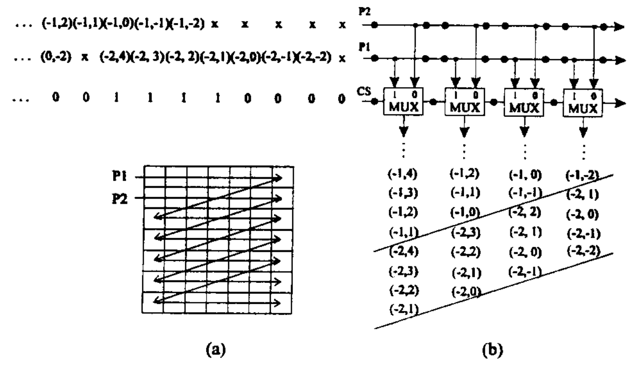


Fig.6. (a) The scanning pattern of the search area.
(b) The data flow arrangement of the search area.

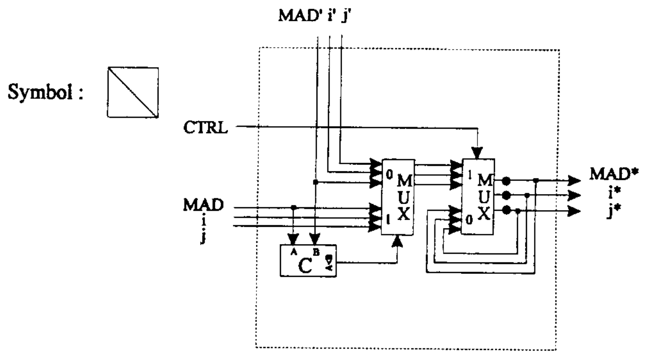


Fig.9. TYPE-4 cell.

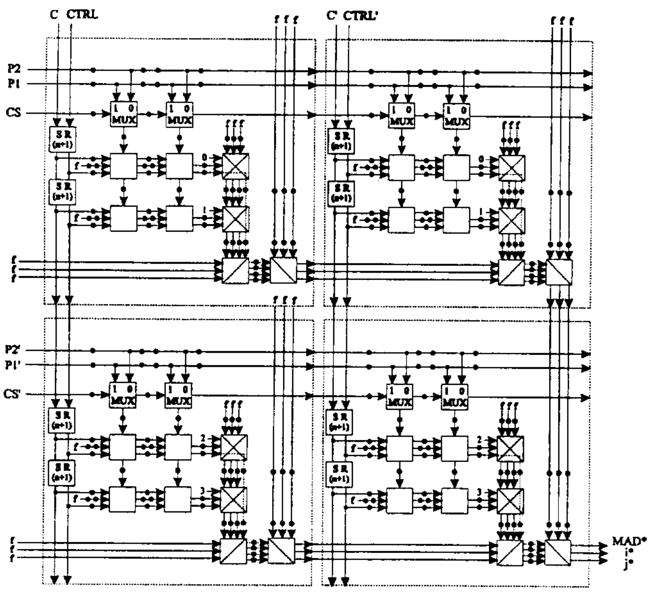


Fig.7. Realization of a motion estimator with m=2
using four basic chips with m=1 each.

TABLE I
Comparison of Different VLSI Motion Estimators

	Yang, Sun, and Wu [3]	Komarek and Pirsch (AS2) [4]	Vos and Stegherr [5]	Artieri and Jutand [6]	Hsieh and Lin [7]	Proposed
Throughput	1/32768	1/1024	1/1024	1/2048	1/4096	1/1024
Cp	32	1	1	2	4	1
Efficiency	100%	100%	100%	100%	50%	100%
Flexibility in m	Yes	No	No	Yes	No	Yes
Flexibility in n	Yes	No	No	Yes	Yes	Yes
fc,min (Hz)	2560M	80M	80M	160M	320M	80M