A Flexible Data-Interlacing Architecture for Full-Search Block-Matching Algorithm

Yeong-Kang Lai, *Student Member, IEEE,* Liang-Gee Chen, *Senior Member, IEEE,* and Yung-Pin Lee
Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan, R.O.C.

Abstract

This paper describes a data-interlacing architecture with two-dimensional (2-D) data-reuse for full-search block-matching algorithm. Based on some cascading strategies, the same chips can be flexibly cascaded for different block sizes, search ranges, and pixel rates. In addition, the cascading chips can efficiently reuse data to decrease external memory accesses and achieve a high throughput rate. Our results demonstrate that the architecture with 2-D data-reuse is a flexible, low-pin-counts, high-throughput, and cascadable solution for full search block-matching algorithm.

1 Introduction

The block-matching algorithm (BMA) for motion estimation is nowadays used in various applications. It removes the temporal redundancy within frame sequences, and thus provides these coding systems with significant bit-rate reduction. A straightforward method, the full-search block-matching algorithm (FBMA), is widely used because it gives the optimal performance and the low control overhead. A number of VLSI motion estimators based on the FBMA have been reported previously. Most of them are based on the array processors because of inherent massive parallelism and high speed requirement of motion estimation. In [1]-[2], they propose a one-dimensional semi-systolic architecture to perform the FBMA. In [3]-[4], two-dimensional systolic arrays combined with on-chip line buffers for implementing the FBMA are presented. The architecture in [3] uses a large amount of register elements to store current block data and search area data. In [5]-[6], the procedure of mapping the FBMA onto systolic arrays is described.

This paper presents a data-interlacing VLSI architecture with 2-D data-reuse to implement the FBMA. The architecture allows serial data inputs to save the pin counts and performs parallel processing to achieve high-throughput requirement. In addition, it is adaptable to the dimensional change of the current block and the search area by cascading the same chips. Moreover, it is simple, regular, and modular and thus is suitable for VLSI implementation.

1063-6862/97 $10.00 © 1997 IEEE

2 The Data-Interlacing VLSI Architecture

In the FBMA, the mean absolute difference (MAD) is calculated for each candidate location (u, v) to find the best match:

$$\mathrm{MAD}(u,v) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} |a(i,j) - b(i+u, j+v)|$$

where $a(i,j)$ is the pixel data in the current block of a size $N \times N$, and $b(i+u, j+v)$ is the pixel data within the search area of previous frame. (u, v) represents the candidate displacement vector, and the values of u and v are limited to between $-p$ to $p-1$. Fig. 1 shows a simple example to illustrate the operation of the proposed architecture. For the case of $N = 4$ and $p = 2$, the search area size is 7×7. Thus, there are 16 ($= 4p^2$) candidate blocks within the search area to perform the block-matching of the current block. Referring to Fig. 2 and Table 1, a detailed data-flow is given for this example. The proposed architecture is based on a one-dimensional PE array and two data-interlacing shift-register arrays. Let each of the PEs compute a fixed candidate block. Thus, the architecture uses 16 ($= 4p^2$) PEs to perform parallel processing. E0~E15 registers and O0~O15 registers are parallel-in parallel-out shift registers. m_i and m_j are the horizontal and vertical components of the estimated motion vector, respectively. MMAD is the final minimum MAD for the current block.

In the first 16 cycles, $t = -15$ to $t = 0$, the even column pixel data (p_{even}) and the odd column pixel data (p_{odd}) within the search area are stored in E0~E15 registers and O0~O15 registers, respectively, as shown in Fig. 2(a). The PEs and CMP are also properly initialized at $t = 0$. After this initialization stage, the current block pixels (c) are sequentially input and broadcasted to all PEs according to the block scan mode and the column-scan order [3]. The search area data sequences, p_{even} and p_{odd}, are also sequentially shifted into the E-registers and the O-registers in a column-scan order at each cycle, respectively. Every 4 ($= N$) cycles, the PEs alternately select data either in the E-registers or in the O-registers by multiplexers to calculate the absolute pixel differences and accumulate the results. After 16 ($= N \times N$) cycles, the PEs will contain the results of accumulated block differences for all the possible candidate blocks within the search area. These results are loaded in parallel into the latches and then sent to the CMP one by one for comparisons. The CMP is a comparator which compares the results in the latches. Then, it outputs the optimum motion vector and the corresponding MAD of the current block. During the comparisons, the PEs continue to perform the block-matching operations of the next current block. The data-flow in Table 1 is continuous for the subsequent blocks within a slice. It does not consume extra cycles to fill up the pipeline operations.

3 Flexible Block Size and Search Area Size

Since different motion-compensation schemes may use different block sizes and require various search area sizes, it is desirable to have motion estimation chip flexible enough for use in different system applications. By a simple control, we can cascade the same motion estimation chips to operate a variety of current block size and search area size. Assume that there are $N \times N$ PEs in a motion estimation chip. If the block size is $N \times N$ and the search area is $-N$ to $N-1$, i.e., $p = N$, there are $2N \times 2N$ candidate blocks within the search area. We partition the search area into 4 sub-search areas. Each of the sub-search areas contains $N \times N$ candidate blocks. Then, each of the chips is designated to handle one sub-search area. By connecting the last outputs of the E and O registers of one chip to the corresponding inputs of another, the proposed architecture can be easily cascaded to handle a large search area. Fig. 3 shows the connection of 4 chips. The chip A processes the sub-search area A, and the chip B processes the sub-search area B, and so on. These four motion estimation chips can work in parallel to estimate the motion vectors within each sub-search area. Then, the results are shifted out of chip for final comparison. Since the speed requirement to do the final comparison is not critical, a slow microprocessor is sufficient to do this job.

The proposed architecture can also handle the block-matching with flexible block sizes. Considering the case for the block size of $2N \times 2N$ with the search area of $p = N/2$, it can be performed by cascading two motion estimation chips. The connection is shown in Fig. 4. The current block is divided into two $N \times 2N$ sub-blocks. The Chip A operates the block-matching of the upper $N \times 2N$ sub-block, and the Chip B operates the block-matching of the lower $N \times 2N$ sub-block. Finally, the partial MAD results at the two corresponding search positions of the two sub-blocks are added and compared in the ADD/CMP to get the motion vector with the minimum MAD. The operations in the ADD/CMP can also be executed by a slow microprocessor without any idle cycle.

4 Performance Analysis

The comparison of our proposed architecture with the other architectures for the FBMA is presented in Table 2 and Table 3. These tables show the compared features for the two cases of ($N = 16, p = 8$) and ($N = 16, p = 16$). Since there are two types of data dependency: 1) the overlap among the adjacent candidate blocks within the search area and 2) the overlap between the search areas of adjacent current blocks. The proposed architecture fully exploits the two-dimensional data-reuse to perform parallel processing. This leads to the significant reduction in I/O bandwidth and saves the pin counts. In these two tables, the number of input data pins and the total number of data accesses per block includes current block data and the corresponding search area data. Fewer data accesses imply the lower demand for I/O bandwidth and pin counts. Furthermore, since the data-flow is continuous, the PEs are 100% busy all the time. Compared to the previously proposed FBMA architectures, this architecture achieves the highest throughput. From the viewpoint of VLSI implementation, the proposed architecture is simple, modular, regular, and cascadable.

References

[1] K. M. Yang, M. T. Sun, and L. Wu, " A family of VLSI designs for the motion compensation block-matching algorithm," *IEEE Transactions on Circuit and System for Video Technology.*, vol. 36, no. 10, pp.1317-1325, Oct. 1989.

[2] Seung Hyun Nam and Moon Key Lee , "Flexible VLSI architecture of motion estimator for video image application ," *IEEE Transactions on Circuits and Systems-II.*, vol. 43 no. 6, pp 467-470, Jun. 1996.

Table 1: Data flow for full-search block matching algorithm ($N = 4, p = 2$).

Cycle Time	Data Sequence	PE0	PE1	.	PE14	PE15
0× 16+0	a(0,0)	a(0,0)-b(0,0)	a(0,0)-b(1,0)	.	a(0,0)-b(2,3)	a(0,0)-b(3,3)
0× 16+1	a(1,0)	a(1,0)-b(1,0)	a(1,0)-b(2,0)	.	a(1,0)-b(3,3)	a(1,0)-b(4,3)
...	...	...	...	.	...	...
...	...	...	...	.	...	...
0×16+14	a(2,3)	a(2,3)-b(2,3)	a(2,3)-b(3,3)	.	a(2,3)-b(4,6)	a(2,3)-b(5,6)
0×16+15	a(3,3)	a(3,3)-b(3,3)	a(3,3)-b(4,3)	.	a(3,3)-b(5,6)	a(3,3)-b(6,6)
1× 16+0	a(0,0)	a(0,0)-b(0,4)	a(0,0)-b(1,4)	.	a(0,0)-b(2,7)	a(0,0)-b(3,7)
1× 16+1	a(1,0)	a(1,0)-b(1,4)	a(1,0)-b(2,4)	.	a(1,0)-b(3,7)	a(1,0)-b(4,7)
...	...	...	...	.	...	...
...	...	...	...	.	...	...
1×16+14	a(2,3)	a(2,3)-b(2,7)	a(2,3)-b(3,7)	.	a(2,3)-b(4,10)	a(2,3)-b(5,10)
1×16+15	a(3,3)	a(3,3)-b(3,7)	a(3,3)-b(4,7)	.	a(3,3)-b(5,10)	a(3,3)-b(6,10)
...	...	...	...	.	...	...
...	...	...	...	.	...	...
NEXT						
...	...	...	...	.	...	...
...	...	...	...	.	...	...

[3] Luc De Vos and Michael Stegherr, , "Parameterizable VLSI architectures for the full-search block-matching algorithm ," *IEEE Transactions on Circuit and System.*, vol. 36 no. 10, pp 1309-1316, Oct. 1989.

[4] C. H.Hsieh and T. P. Lin , "VLSI architecture for block-matching motion estimation algorithm ," *IEEE Transactions on Circuit and System for Video Technology.*, vol. 2 no. 4, pp 169-175, Jun. 1992.

[5] T. Komarek and P. Pirsh, , "Array architectures for block matching algorithms ," *IEEE Transactions on Circuit and System.*, vol. 36 no. 10, pp 1301-1308, Oct. 1989.

[6] Hangu Yeo and Yu-Hen Hu , "A novel modular systolic array architecture for full-search block matching motion estimation ," *IEEE Transactions on Circuit and System for Video Technology.*, vol. 5 no. 5, pp 407-416, Oct. 1995.

Table 2: Block Size: $N = 16$, Search Area Size: $p = 8$

Architecture	[1]	[2]	[3]	[4]	[5]	[6]	proposed architecture
No. of input data pins	24	24	24	16	136	24	24
No. of PEs	16	16	256	256	256	256	256
Clock cycles/block	4096	4096	256	961	496	256	256
Throughput(blocks/cycles)	$\frac{1}{4096}$	$\frac{1}{4096}$	$\frac{1}{256}$	$\frac{1}{961}$	$\frac{1}{496}$	$\frac{1}{256}$	$\frac{1}{256}$
PE utilization	100%	100%	100%	26.6%	100%	100%	100%
No. of data accesses	12288	12032	752	1217	8192	768	752
Cascadable ?	Yes	Yes	No	No	No	Yes	Yes

Table 3: Block Size: $N = 16$, Search Area Size: $p = 16$

Architecture	[1]	[2]	[3]	[4]	[5]	[6]	proposed architecture
No. of input data pins	48	32	24	16	136	72	40
No. of PEs	64	64	256	256	256	1024	1024
Clock cycles/block	4096	4096	1024	2209	1504	256	256
Throughput(blocks/cycles)	$\frac{1}{4096}$	$\frac{1}{4096}$	$\frac{1}{1024}$	$\frac{1}{2209}$	$\frac{1}{1504}$	$\frac{1}{256}$	$\frac{1}{256}$
PE utilization	100%	100%	100%	46.6%	100%	100%	100%
No. of data accesses	32768	32256	1008	2465	24320	2304	1248

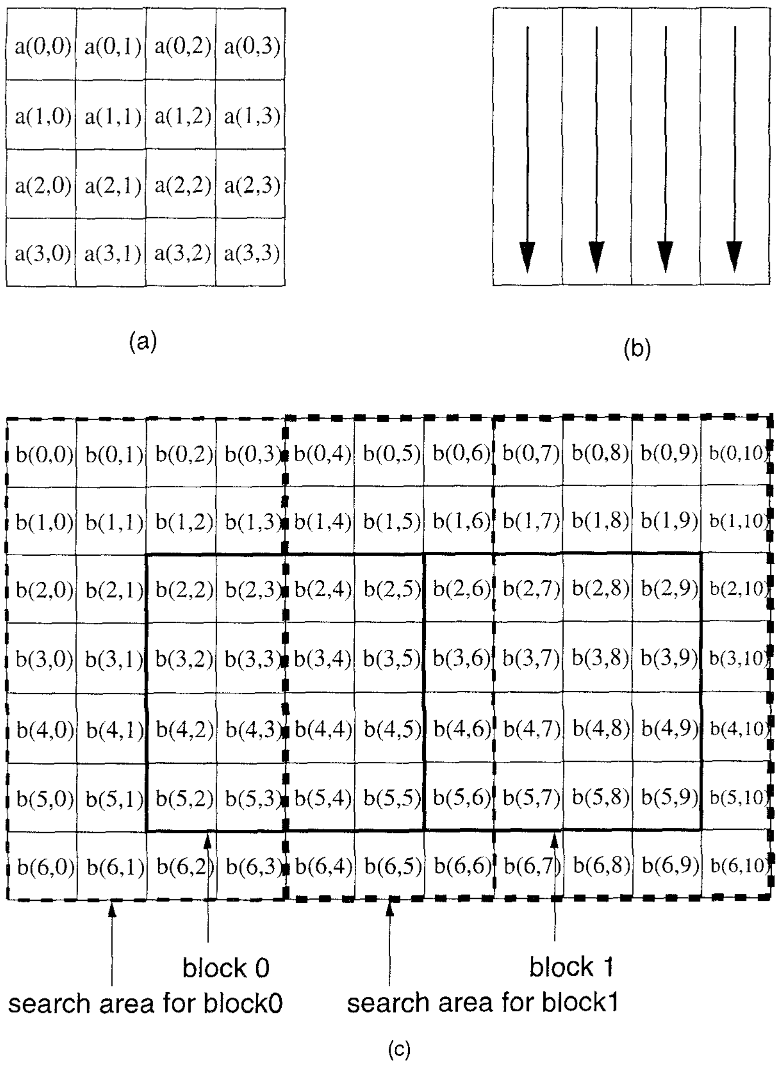
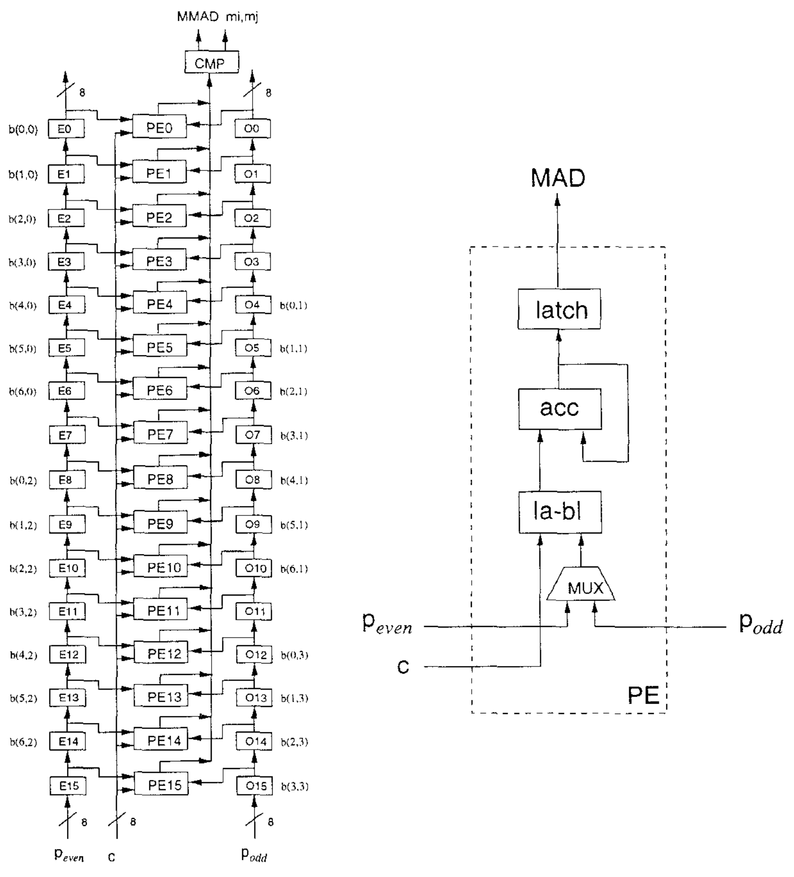
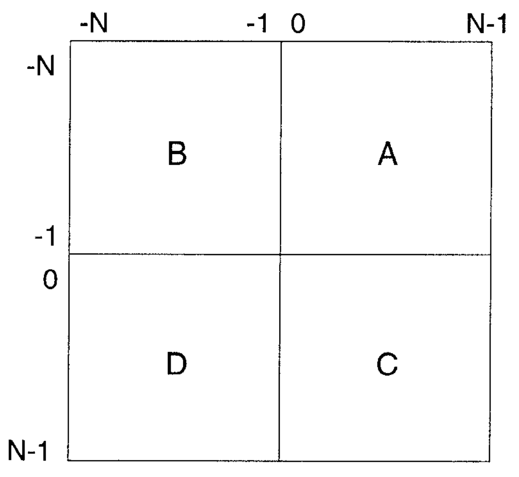

a(0,0) a(0,1) a(0,2) a(0,3)
a(1,0) a(1,1) a(1,2) a(1,3)
a(2,0) a(2,1) a(2,2) a(2,3)
a(3,0) a(3,1) a(3,2) a(3,3)
(a)
(b)
b(0,0) b(0,1) b(0,2) b(0,3) b(0,4) b(0,5) b(0,6) b(0,7) b(0,8) b(0,9) b(0,10)
b(1,0) b(1,1) b(1,2) b(1,3) b(1,4) b(1,5) b(1,6) b(1,7) b(1,8) b(1,9) b(1,10)
b(2,0) b(2,1) b(2,2) b(2,3) b(2,4) b(2,5) b(2,6) b(2,7) b(2,8) b(2,9) b(2,10)
b(3,0) b(3,1) b(3,2) b(3,3) b(3,4) b(3,5) b(3,6) b(3,7) b(3,8) b(3,9) b(3,10)
b(4,0) b(4,1) b(4,2) b(4,3) b(4,4) b(4,5) b(4,6) b(4,7) b(4,8) b(4,9) b(4,10)
b(5,0) b(5,1) b(5,2) b(5,3) b(5,4) b(5,5) b(5,6) b(5,7) b(5,8) b(5,9) b(5,10)
b(6,0) b(6,1) b(6,2) b(6,3) b(6,4) b(6,5) b(6,6) b(6,7) b(6,8) b(6,9) b(6,10)
block 0
search area for block0
block 1
search area for block1
(c)

Figure 1: An example to illustrate the operation of the architecture. (a) 4×4 current block. (b) The column-scan order of current block. (c) 7×7 search area.

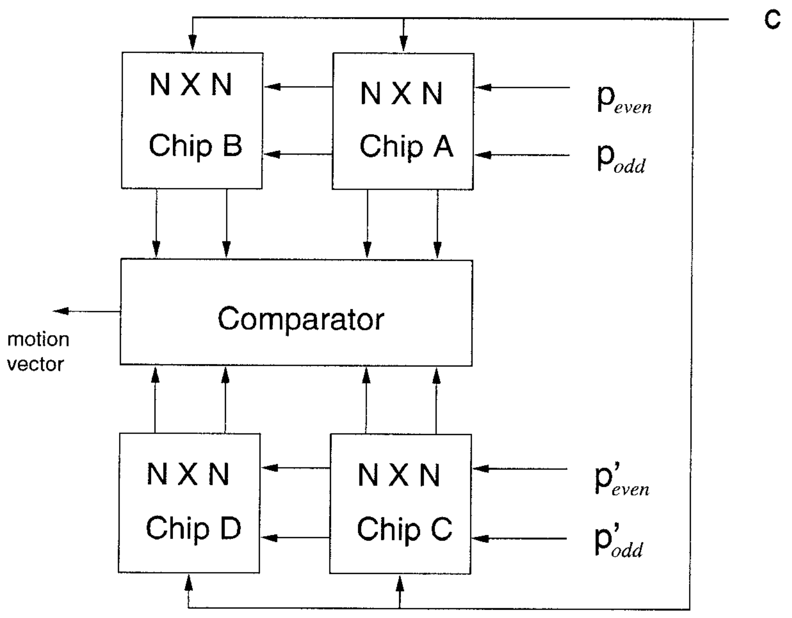


(a) (b)

Figure 2: (a) The proposed data-interlacing architecture with 2-D data-reuse ($N = 4, p = 2$). (b) Architecture of PE.

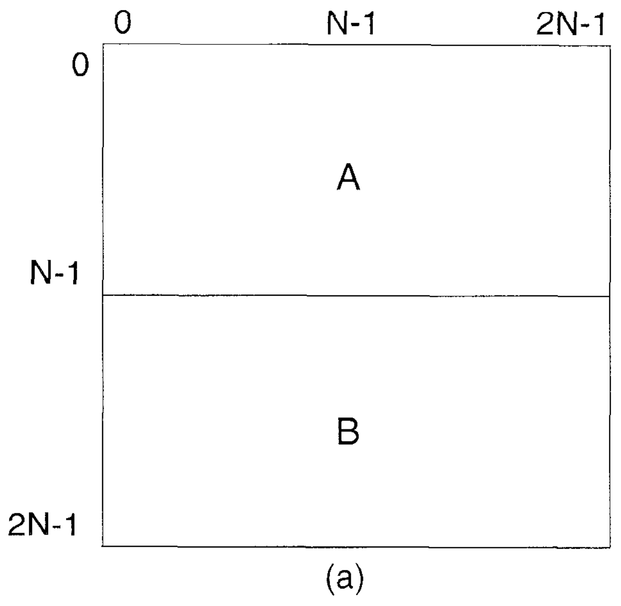


(a)



(b)

Figure 3: Realization of a motion estimator for the $N \times N$ current block and the search area of $-N$ to $N-1$ pixels. (a) 4 sub-search areas. (b) An architecture for cascading 4 chips (Each chip is designated to handle one sub-search area).



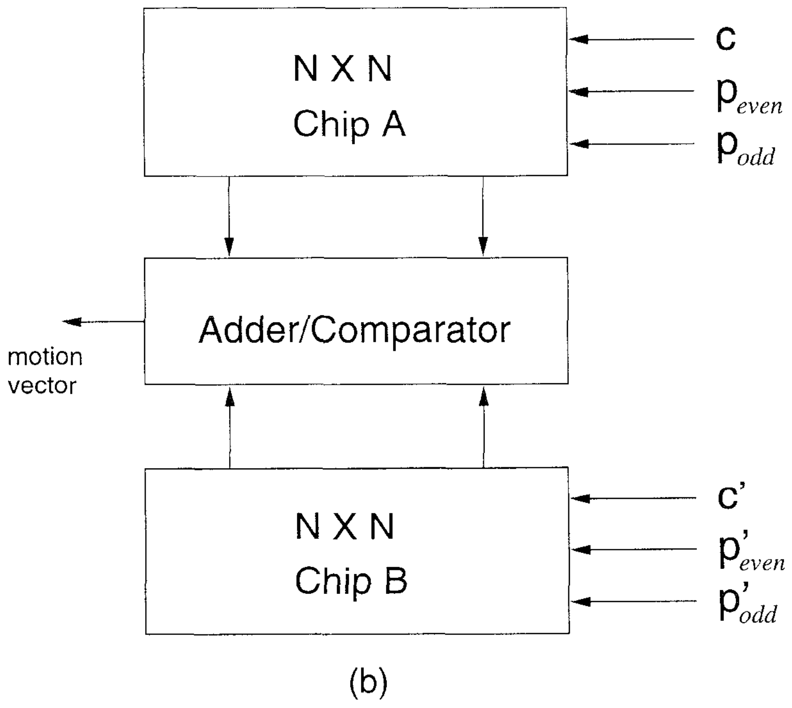


Figure 4: Realization of a motion estimator for the $2N \times 2N$ current block and the search area of $-N/2$ to $N/2 - 1$ pixels.(a) 2 sub-blocks. (b) An architecture for cascading two chips (Each chip is designated to handle the block-matching of one sub-block).