AN EFFICIENT ARRAY ARCHITECTURE WITH DATA-RINGS FOR 3-STEP HIERARCHICAL SEARCH BLOCK MATCHING ALGORITHM

Yeong-Kang Lai, Liang-Gee Chen, and Jun-Fu Shen

Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan, R. O. C. China

ABSTRACT

This paper describes an efficient 9-cells array architecture with data-rings for the 3-step hierarchical search block-matching algorithm. With the efficient data-rings and memory organization, the regular raster-scanned data flow and comparator-tree structure can be used to simplify control scheme and reduce latency, respectively. In addition, we utilize a three-half-search-area scheme to reduce external memory access and interconnection. The results demonstrate that the array architecture with the data-rings gives short latency and low input ports. It also provides a high normalized throughput solution for the 3SHS.

1. INTRODUCTION

Among various video compression techniques, the motion-compensated hybrid coding is the most commonly adopted by several standards . 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. However, it requires a large amount of computation and high memory bandwidth. Many fast search strategies have been developed to reduce the extremely high computational complexity of the optimal full search (FS) procedure. Among all these search strategies, the 3-step hierarchical search (3SHS) is considered as one of the best algorithms and is recommended by CCITT RM8 and MPEG SM3. In spite of the good performance and significant complexity reduction provided by the 3SHS, almost all reported BMA architectures and VLSI implementations focus on the FS because of its regular data flow and low control overhead [1]-[3]. The main architectural problems which prevent 3SHS BMA from being widely used in real-time systems are the variable distances between candidate locations and the sequential execution between steps. They will result in unpredictable data access. Besides, these problems complicate the control scheme, lower the efficiency of computation kernel, and make the data-reuse difficult. On the other hand, the latency of each step needs to be taken into account for high-speed applications. This imposes an additional limitation on architecture design. Recently, some researchers have reported various architectures for 3SHS BMA [4]-[6]. Based on the work published by Yang *el al.* [1], Dutta and Wolf [5] use multiple memory banks and a multistage interconnection network to map different search strategies on the architecture. Yeo and Hu [6] utilize search area data dependency to propose three types of architectures. the 3SHS BMA.

In this paper, we propose an efficient 9-cells array architecture with data-rings to implement 3SHS BMA. This architecture not only simplifies the control scheme with a regular raster-scanned data flow, but shortens the latency by using a comparator-tree structure. It makes the architecture achieve high normalized throughput. In addition, the techniques to reduce external memory access and interconnection can be applied to the architecture.

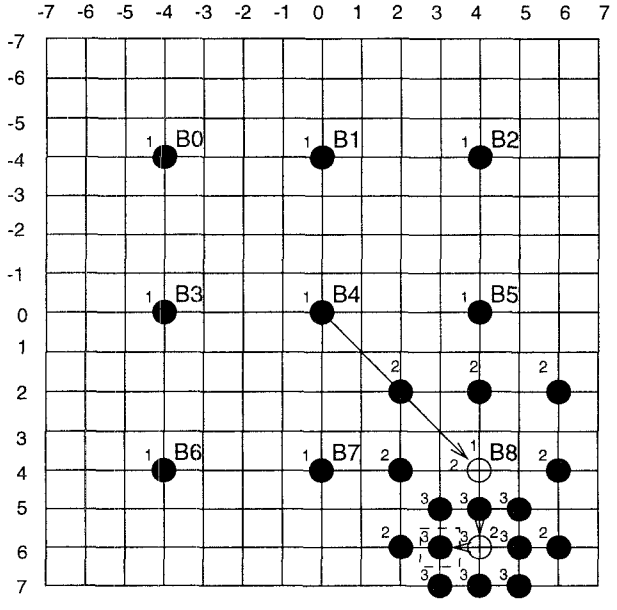

Figure 1. The illustration of three-step hierarchical search.

2. THE PROPOSED VLSI ARCHITECTURE

To reduce the huge computational load resulting from the massive number of candidate locations, the 3SHS algorithm searches for the best motion vector in a coarse-to-fine manner. Fig. 1 illustrates the procedure of the 3SHS with an example of motion vector $(3, 6)$. The first step demands nine sparsely located candidates to be evaluated and the one with a minimal mean absolute difference (MAD) is picked out. The candidate blocks are labeled as B0 $\sim$ B8. In the second step, the search focuses on the area centered at the winner of the previous step, but the distances between candidate locations are shortened by half. In a similar manner, the step three is to compare MAD's of nine locations around

0-7803-3583-X/97 $10.00 ©1997 IEEE

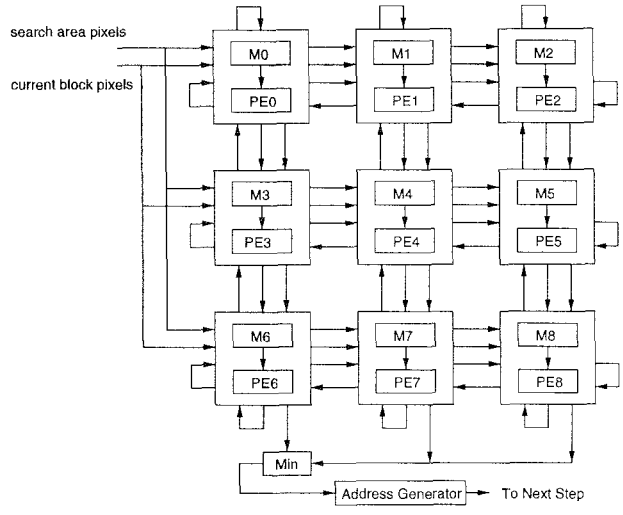

Figure 2. 9-cells architecture with data-rings for 3SHS.

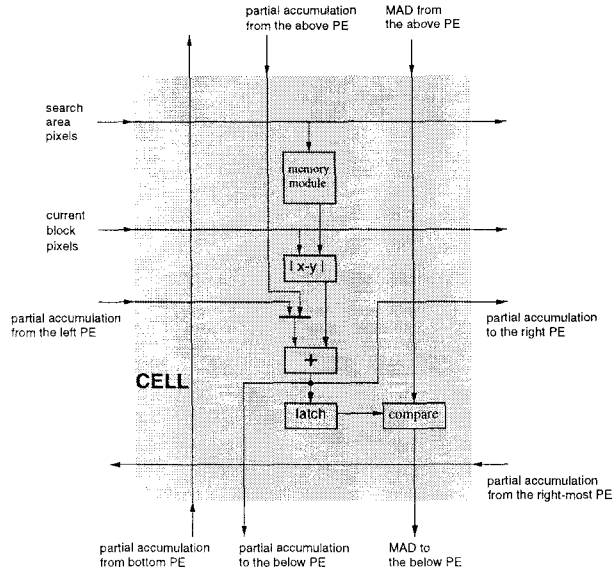

Figure 3. Architecture of the cell.

the winner of the second step and then give the final motion vector.

Considering the cost of control scheme, latency and data access, we propose the 9-cells array architecture with data-rings, as shown in Fig. 2. It can evaluate the 9 candidate locations in parallel and accumulate the absolute pixel difference of each candidate block-matching sequentially. Each cell is composed of a memory module and a processing element (PE). The PE consists of an absolute difference unit, an accumulator, a final-result latch, and a comparator. The comparators are connected into a tree structure. Fig. 3 shows the architecture of the cell. The current block pixels are sequentially input and broadcast to all PE's in raster-scanned order. For example, a (3×3)-pixel block is shown below:

$$\begin{matrix} a(0,0) & a(0,1) & a(0,2) \\ a(1,0) & a(1,1) & a(1,2) \\ a(2,0) & a(2,1) & a(2,2) \end{matrix}$$

The current block pixels are read in order of a(0,0), a(0,1), a(0,2), a(1,0), a(1,1), a(1,2), a(2,0), a(2,1), and a(2,2).

The memory modules store the search area pixels for prediction. Memory interleaving technique is used to provide a solution to parallel data access. The search area pixels are interleaved to these 9 memory modules. Fig. 4 illustrates the distribution of search area pixels to 9 memory modules. The label $(0 \sim 8)$ for each pixel indicates the index of the memory module that stores the pixel. However, if each PE is fixed to compute a certain candidate, the 9 PE's can simultaneously read 9 individual candidate pixels from the 9 memory modules. For example, PE0 performs all the operations required to find the MAD of candidate location B0 (Fig. 1 shows the labels of the 9 candidate locations). This approach is straightforward, but it suffers high interconnection and switching overhead. To overcome this drawback, we rearrange the data flow from memory modules to PE's. Let 9 PE's alternatively perform the operations of 9 candidate locations. Since the PE's are connected in horizontal rings and vertical rings, it makes partial-accumulation data shift and accumulate between the two adjacent PE's in a horizontal or vertical ringed fashion. Therefore, each PE may alternatively calculate partial-accumulations of different candidate locations at different cycles. By using the ringed transfer of the partial-accumulation, each PE only connects to one memory module. This eliminates the complicated interconnection and the switching circuitry between the memory modules and the PE's.

Table 1 and Table 2 illustrate the whole data flow of each step by showing every PE's corresponding candidate locations (B0 $\sim$ B8) at different clock cycles. Referring to Fig. 2 and Fig. 3, the operation of the architecture for the block-size of 16×16 and search range of -7 to +7 can be explained as follows. At each cycle, the current block pixels a(.) are sequentially broadcast to all PE's in raster-scanned order. The candidate pixels required by each PE are read into the PE's in parallel. These pixels operate with a(.). Then, each PE shifts the partial-accumulation to the adjacent PE in horizontal ringed direction to accumulate the next partial-accumulation of the the same candidate location. After every 16 cycles, each partial-accumulation shifts and accumulates in a vertical ringed direction. After 256 clock cycles, the final accumulation-result of each candidate is produced in the final-result latch of each PE. Finally, the 9 latched MAD's can then be sent to the comparator-tree to get a minimum MAD. It only necessitates four clock cycles to get the minimum MAD. Then, the architecture takes one clock cycle to decide the next step again and repeats the above operation. At $t = (256 + 4 + 1) \times 3 = 783$, all the possible candidate blocks within search area have been compared. The motion vector for the current block is generated.

To avoid the extremely high memory bandwidth and large input ports, we utilize the original three half-search-area (HSA) sheme (see Fig. 6) [4]. When executing the 3SHS of block 0, the PE's access HSA A and B. During performing the 3SHS of block 0, the right-half pixels of the search area (HSA C) for block 1 is being written into the 9 memory modules. In performing the 3SHS of the next block (block 1), the PE's access HSA B and C. Meanwhile, the pixels of HSA D are input to the 9 memory modules. In this cyclic manner, the HSA pixels can be easily accessed from system memory during the current block-matching by only one input port.

Table 1. Data flow for step 1 and step 3

Cycle time	Data Sequence	PE0	PE1	PE2	.	PE7	PE8
0X16+0	a(0,0)	B0	B1	B2	.	B7	B8
0X16+1	a(0,1)	B2	B0	B1	.	B6	B7
0X16+2	a(0,2)	B1	B2	B0	.	B8	B6
0X16+3	a(0,3)	B0	B1	B2	.	B7	B8
0X16+4	a(0,4)	B2	B0	B1	.	B6	B7
0X16+5	a(0,5)	B1	B2	B0	.	B8	B6
0X16+6	a(0,6)	B0	B1	B2	.	B7	B8
0X16+7	a(0,7)	B2	B0	B1	.	B6	B7
0X16+8	a(0,8)	B1	B2	B0	.	B8	B6
0X16+9	a(0,9)	B0	B1	B2	.	B8	B8
0X16+10	a(0,10)	B2	B0	B1	.	B6	B7
0X16+11	a(0,11)	B1	B2	B0	.	B8	B6
0X16+12	a(0,12)	B0	B1	B2	.	B7	B8
0X16+13	a(0,13)	B2	B0	B1	.	B6	B7
0X16+14	a(0,14)	B1	B2	B0	.	B8	B6
0X16+15	a(0,15)	B0	B1	B2	.	B7	B8
1X16+0	a(1,0)	B6	B7	B8	.	B4	B5
1X16+1	a(1,1)	B8	B6	B7	.	B3	B4
...	...	...	...	...	.	...	...
...	...	...	...	...	.	...	...
1X16+14	a(1,14)	B7	B8	B6	.	B5	B3
1X16+15	a(1,15)	B6	B7	B8	.	B4	B5
...	...	...	...	...	.	...	...
...	...	...	...	...	.	...	...
...	...	...	...	...	.	...	...
...	...	...	...	...	.	...	...
14X16+0	a(14,0)	B3	B4	B5	.	B1	B2
14X16+1	a(14,1)	B5	B3	B4	.	B0	B1
...	...	...	...	...	.	...	...
14X16+14	a(14,14)	B4	B5	B3	.	B2	B0
14X16+15	a(14,15)	B3	B4	B5	.	B1	B2
15X16+0	a(15,0)	B0	B1	B2	.	B7	B8
15X16+1	a(15,1)	B2	B0	B1	.	B6	B7
...	...	...	...	...	.	...	...
...	...	...	...	...	.	...	...
15X16+14	a(15,14)	B1	B2	B0	.	B8	B6
15X16+15	a(15,15)	B0	B1	B2	.	B7	B8

Table 2. Data flow for step 2

Cycle time	Data Sequence	PE0	PE1	PE2	.	PE7	PE8
0X16+0	a(0,0)	B0	B2	B1	.	B5	B4
0X16+1	a(0,1)	B1	B0	B2	.	B3	B5
0X16+2	a(0,2)	B2	B1	B0	.	B4	B3
0X16+3	a(0,3)	B0	B2	B1	.	B5	B4
0X16+4	a(0,4)	B1	B0	B2	.	B3	B5
0X16+5	a(0,5)	B2	B1	B0	.	B4	B3
0X16+6	a(0,6)	B0	B2	B1	.	B5	B4
0X16+7	a(0,7)	B1	B0	B2	.	B3	B5
0X16+8	a(0,8)	B2	B1	B0	.	B4	B3
0X16+9	a(0,9)	B0	B2	B1	.	B5	B4
0X16+10	a(0,10)	B1	B0	B2	.	B3	B5
0X16+11	a(0,11)	B2	B1	B0	.	B4	B3
0X16+12	a(0,12)	B0	B2	B1	.	B5	B4
0X16+13	a(0,13)	B1	B0	B2	.	B3	B5
0X16+14	a(0,14)	B2	B1	B0	.	B4	B3
0X16+15	a(0,15)	B0	B2	B1	.	B5	B4
1X16+0	a(1,0)	B3	B5	B4	.	B8	B7
1X16+1	a(1,1)	B4	B3	B5	.	B6	B8
...	...	...	...	...	.	...	...
...	...	...	...	...	.	...	...
1X16+14	a(1,14)	B5	B4	B3	.	B7	B6
1X16+15	a(1,15)	B3	B5	B4	.	B8	B7
...	...	...	...	...	.	...	...
...	...	...	...	...	.	...	...
...	...	...	...	...	.	...	...
...	...	...	...	...	.	...	...
14X16+0	a(14,0)	B6	B8	B7	.	B2	B1
14X16+1	a(14,1)	B7	B6	B8	.	B0	B2
...	...	...	...	...	.	...	...
14X16+14	a(14,14)	B8	B7	B6	.	B1	B0
14X16+15	a(14,15)	B6	B8	B7	.	B2	B1
15X16+0	a(15,0)	B0	B2	B1	.	B5	B4
15X16+1	a(15,1)	B1	B0	B2	.	B3	B5
...	...	...	...	...	.	...	...
...	...	...	...	...	.	...	...
15X16+14	a(15,14)	B2	B1	B0	.	B4	B3
15X16+15	a(15,15)	B0	B2	B1	.	B5	B4

3. PERFORMANCE ANALYSIS AND VLSI IMPLEMENTATION

The comparison of our proposed architecture with the other architectures for 3SHS BMA is presented in Table 3. The programmable architecture in [5] needs many clock cycles and input data ports to execute 3SHS BMA. It degrades the system throughput, since the architecture breaks cycles in order to remove memory conflicts and network-path conflicts. Type II and Type III architectures in [6] have high throughput rate. However, they demand many PE's and huge input data ports to satisfy high input bandwidth requirement. Our proposed architecture gets a better balance between the throughput and the PE count. With the regular data flow and the efficient data-rings, the high normalized throughput ($\frac{Throughput}{No.\ of\ the\ used\ PE's}$) can be realized. On the other hand, we use the three-half-search-area scheme to reduce input bandwidth and get low input data ports.

The 9-cells array architecture has been designed using COMPASS 0.6 μm CMOS technology. It is capable of processing MPEG-2/MP@ML (720 pixels $\times$ 480 pixels, 30 frames/s) video in real-time in a single chip when the block size is 16 $\times$ 16 and the search area range is -7 to +7. Table 4 summarizes the characteristics of the chip.

4. CONCLUSION

An array architecture with data-rings for the 3SHS BMA has been described in this paper. Due to the regular raster-scanned data flow and the efficient data-ringed architecture, the partial-accumulation data can transfer in a ringed manner. The final MAD's of the 9 candidate locations are computed out simultaneously. Then, they are compared by a comparator-tree to shorten the latency. The interleaved memory module provides every PE with its required data simultaneously without introducing complicated interconnection and switching circuitry. These advantages make the architecture achieve high normalized throughput and low input data ports. Furthermore, since it effectively utilizes the key features of hierarchical BMA to manage data, schemes with different search ranges and block sizes can be handled without affecting the hardware structure. The proposed architecture can also be applied to the implementation of the other hierarchical BMA.

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] Y. S. Jehng, L. G. Chen, and T. D. Chiueh, " An efficient and simple VLSI tree architecture for motion estimation algorithm," *IEEE Transactions on Signal Processing.*, vol. 41, no. 2, Feb. 1993.

[3] Luc De Vos and Matthias Schobinger , "VLSI architecture for a flexible block matching processor ," *IEEE Transactions on Circuit and System for Video Technology.*, vol. 5 no. 5, pp 417-428, Oct. 1995.

[4] H. M. Jong, L. G. Chen, and T. D. Chiueh, " Parallel architecture for 3-step hierarchical search block-matching algorithm," *IEEE Transactions on Circuit and System for Video Technology.*, vol. 4, no. 4 pp. 407-416, 1994.

[5] Santanu Dutta and Wayne Wolf , "A flexible parallel architecture adapted to block-matching motion-estimation algorithms ," *IEEE Transactions on Circuit and System for Video Technology.*, vol. 6 no. 1, pp 74-86, Feb. 1996.

[6] Hangu Yeo and Yu-Hen Hu , "A novel architecture and processor-level design based on a new matching criterion for video compression ," *IEEE Workshop on VLSI Signal Processing.*, pp 448-457, Nov. 1996.

Table 3. Comparison between the proposed architecture and the other architectures

Architecture	proposed architecture	architecture in [5]	architecture in [6]		
			Type I	Type II	Type III
Input data ports	16	136	40	132	416
No. of the used PE's	9	9	9	36	144
Clock cycles per block matching	783	1280	798	237	129
Throughput	$\frac{1}{783}$	$\frac{1}{1280}$	$\frac{1}{798}$	$\frac{1}{237}$	$\frac{1}{129}$
$\frac{Throughput}{No.\ of\ the\ used\ PE's}$	$\frac{1}{7047}$	$\frac{1}{11520}$	$\frac{1}{7182}$	$\frac{1}{8532}$	$\frac{1}{18576}$

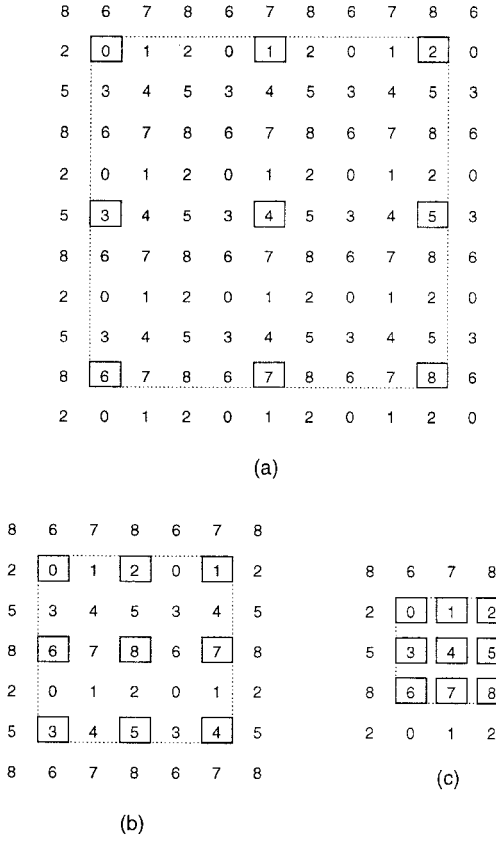

Figure 4. The distribution of search area pixels to 9 memory modules and the memory interleaving instances of accessed data (marked by small frames) for (a) step 1; (b) step 2; (c) step 3.

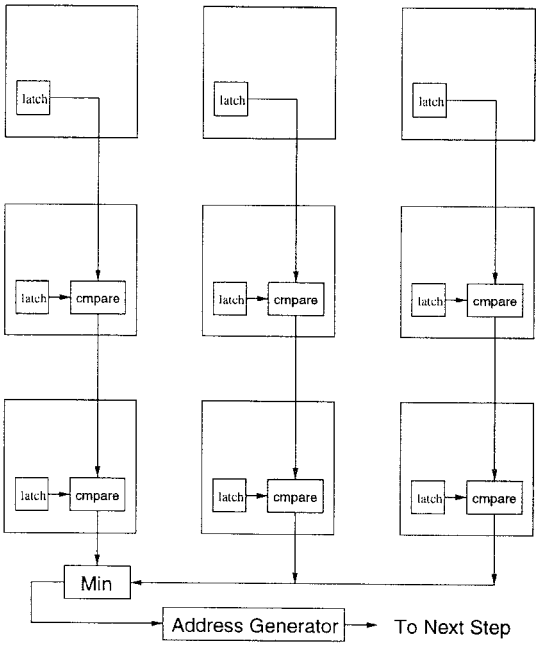

Figure 5. Extraction of minimum MAD.

Table 4. Chip Features

Technology	COMPASS 0.6 μm CMOS
Die Size	6.67 mm × 4.69 mm
Number of Transistors	45636
Number of Pins	26 pins
Clock rate	50 MHz

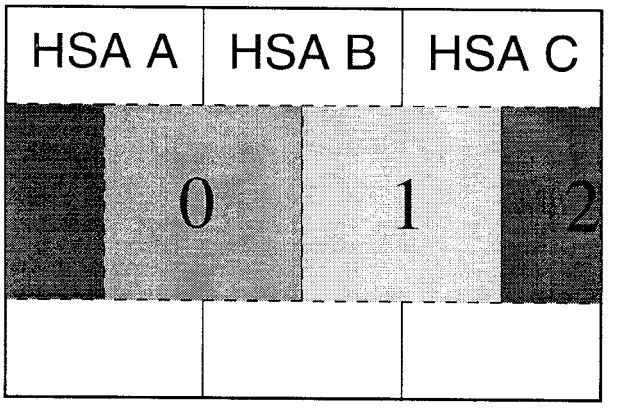

Figure 6. Data re-use in proposed design.