LOW-POWER PARALLEL TREE ARCHITECTURE FOR FULL SEARCH BLOCK-MATCHING MOTION ESTIMATION

Siou-Shen Lin, Po-Chih Tseng, and Liang-Gee Chen

DSP/IC Design Lab, Graduate Institute of Electronics Engineering,
Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan
E-Mail:{sslin, pctseng, lgchen}@video.ee.ntu.edu.tw

ABSTRACT

In this paper, a novel low-power parallel tree architecture is proposed for full search block-matching motion estimation. The parallel tree architecture exploits the spatial data correlations between parallel candidate block searches for data sharing, which effectively eliminates huge amount of data access bandwidth while consumes fewer hardware resources compared with array-based architectures. Combining with adaptive parallel partial distortion elimination algorithm, the required average clock cycle count for each macroblock search can be greatly reduced to below 50% to achieve low-power operation. Besides, this architecture can also eliminate redundant computation without pipeline latency and excess power consumption caused by register shifting and redundant memory accessing in array-based architectures. The proposed architecture is suitable for high-end real-time portable video encoding system, which desires high-quality video but low-power consumption.

1. INTRODUCTION

Motion estimation is the fundamental technique of video compression, which effectively reduces the temporal redundancy among video sequences. In order to achieve better video quality, full search block-matching algorithm (FSBMA) is usually adopted for motion estimation. The FSBMA determines the motion vector by identifying the macroblock with minimum distortion from a set of all possible candidate blocks in the search window, and therefore enables to achieve the optimal search result. However, it takes huge amount of computation to perform full search of all possible candidate blocks. Due to the huge amount of computation, motion estimation is the most power demanding kernel in a video encoding system, which usually consumes more than 50% power consumption of whole system [1].

For the portable system, lowering power consumption is one of the most critical design issues. In order to reduce computational complexity and thus lower power consumption, many fast algorithms were proposed in the literature. One category of fast algorithms proposed to effectively reduce the computation amount by fewer search positions or fewer matching samples, but it could only achieve sub-optimal search result. Another category proposed to reduce the computation amount without sacrificing the quality of motion estimation, which is referred as fast full search, such as the partial distortion elimination (PDE) [2] [3]. The principle of PDE is to stop the accumulation of sum of absolute difference (SAD) to reduce redundant computation when the temporal accumulated SAD value is larger than recent minimum one.

Due to the regular data flow of FSBMA motion estimation, various array-based [4] [5] [6] [7] [8] [9]and tree-based [10] architectures were proposed. However, most of them did not exploit fast full search algorithms to further reduce the redundant computation. Although some array-based architectures [11] [12] were proposed to exploit PDE and could achieve approximate 50% average skipping ratios, there were several problems in these architectures, such as pipeline latency and excess power consumption caused by register shifting and redundant memory accessing.

In this paper, a novel parallel tree architecture is proposed for FSBMA motion estimation. The parallel tree architecture exploits the spatial data correlations between parallel candidate block searches for data sharing, which effectively eliminates huge amount of data access bandwidth while consumes fewer hardware resources compared with array-based architectures. Combining with adaptive parallel PDE algorithm, the required average clock cycle count for each macroblock search can be greatly reduced to below 50% without problems found in array-based architectures. The organization of this paper is as follows. Section 2 presents the proposed parallel tree architecture, and section 3 gives the hardware implementation results. In section 4, the comparison results with previous arts are listed, and finally, a brief conclusion is given in section 5.

2. PROPOSED ARCHITECTURE

2.1. FSBMA Motion Estimation

The FSBMA motion estimation is a four nested-loops calculation in the macroblock level, where p denotes the search range and N denotes the block size.

```
Loop1 : For j = -p to p - 1
Loop2 :     For i = -p to p - 1
Loop3 :         For l = 0 to N- 1
Loop4 :             For k = 0 to N-1
                        SAD(i,j)=SAD(i,j) + |C(k,l) - R(k+i,l+j)|
                    End (Loop4)
                End (Loop3)
            End (Loop2)
        End (Loop1)
```

These four nested-loops are generally decomposed with different methods to derive various architectures. Most array-based and

This work was supported in part by MOE Program for Promoting Academic Excellence of Universities under the grant number 89E-FA06-2-4-8, in part by National Science Council, Republic of China, under the grant number 91-2215-E-002-035, and in part by MediaTek Inc.

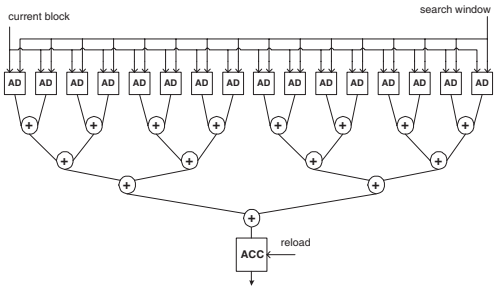

Fig. 1. A 1/16-cut subtree architecture (16×1PE)

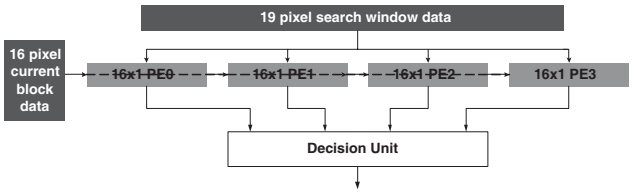

Fig. 2. Four 16×1PEs organized in parallel form

tree-based architectures process loop3 or loop4 as well in parallel while some process loop1 or loop2 as well in parallel. Regarding to fast full search, [11] and [12] proposed to exploit PDE in loop3 or loop4 as well. When the accumulated SAD is lager than recent minimum one, the processing elements (PEs) are disabled to avoid redundant computation. However, there were few discussions about the exploitation of PDE in loop1 or loop2 as well. In proposed parallel tree architecture, it exploits PDE both in loop4 and loop2.

2.2. Parallel tree architecture

The proposed parallel tree architecture is based on the tree architecture [10]. As shown in Fig. 1, a 1/16-cut subtree is presented and denoted as "16×1PE". Each AD in the 16×1PE calculates the absolute difference value between the current block data and the candidate block data in search window, and the following adder tree and accumulator (ACC) accumulate these 16 absolute difference values. A SAD value is generated in the ACC every 16 clock cycles for one candidate block search. In other words, the loop4 are processed in parallel. By exploiting PDE, the clock cycles for each candidate block search can be reduced further.

In proposed parallel tree architecture, the loop2 is also processed in parallel with PDE. As shown in Fig. 2, the 16×1PE can be organized in parallel form, and the number of 16×1PE is scalable according to target system specification. In the following,

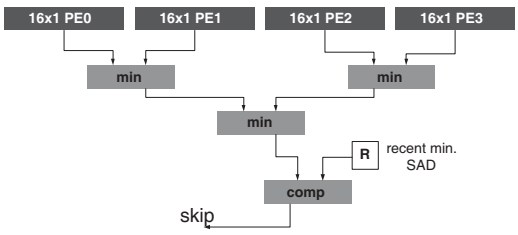

Fig. 3. Minimum comparing tree in decision unit

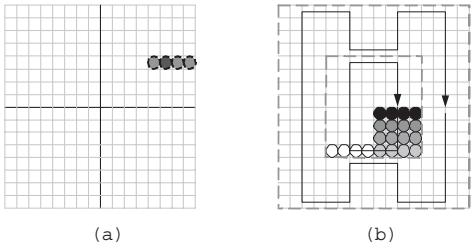

Fig. 4. (a) Four parallel predicted positions (b) The modified spiral scanning order for four candidate block searches

four 16×1PEs in parallel is taken for example. As shown in Fig. 2, 19 pixels of the search window data and 16 pixels of the current block data are broadcasted to the four 16×1PEs. Each of the four 16×1PEs receives the corresponding 16 pixels of the broadcasted 19 pixels as its input data. For example, pixel 1 to pixel 16 are sent to the 16×1PE0, pixel 2 to pixel 17 are sent to the 16×1PE1, and so on. That is to say, the four candidate block searches are computed in parallel, and the search window data are shared horizontally by the four 16×1PEs. Besides, the four accumulated SAD values for four candidate block searches are delivered to the decision unit, which is a minimum comparing tree as shown in Fig. 3. The minimum comparing tree decides the smallest SAD value of the four accumulated SAD values, and the comp compares this smallest SAD value with recent minimum one stored in the register R. If this smallest SAD value is larger than recent minimum one, the SAD value in R will remain the same and the computation for the four candidate block searches will be skipped. Without PDE, it takes 4096 (32×32 / 4×16) clock cycles to calculate a 16×16 block for search range from -16 to +15. The clock cycles can be further reduced by exploiting PDE.

2.3. Low-power operation by exploiting PDE

The skipping ratio of PDE is mainly based on two fundamentals: the initial SAD value and the scanning order. From the result in [13], it was shown that better skipping ratio can be obtained by getting the initial SAD value from the predicted position and by scanning in spiral order. The predicted position is estimated by the median of motion vectors of three neighbor blocks.

In order to exploit PDE in proposed architecture, the predicted position and the spiral scanning order are required to be slightly modified. As shown in Fig. 4(a), the predicted position is (5,-4). Since four candidate blocks are computed in parallel, the positions (4,-4), (6,-4), and (7,-4) are computed as well. In other words, if the predicted position is (px,py),then (px%4,py), (px%4+1,py), (px%4+2,py), and (px%4+3,py) are all computed in parallel. This parallel predicted positions scheme not only fully utilizes the parallel tree architecture, but also enables to get a better initial SAD value for PDE. After getting initial SAD value from parallel predicted positions, the candidate blocks are scanned in a modified spiral order as shown in Fig. 4(b).

Since the number of 16×1PEs is scalable, skipping ratios of PDE are simulated for one 16×1PE, four 16×1PEs, and sixteen 16×1PEs. According to the simulation results by hardware C, it can be shown that the skipping ratios of PDE in parallel form are still attractive because of the spatial correlations between parallel candidate blocks. The skipping ratios are 71.01%, 67.75% and 64.05% in average for the parallelism one, four and sixteen, re-

Table 1. Comparison results between different parallelisms and test sequences. For CIF format, block size of 16×16, search range from -16 to +15.

	parallelism		
sequence	1	4	16
coastguard	68.04%	64.95%	63.47%
foreman	66.82%	62.54%	57.21%
mobile	70.02%	66.39%	63.25%
stefan	62.67%	59.21%	54.92%
table	69.89%	66.25%	62.54%
wether	88.64%	87.13%	82.91%
avg.	**71.01%**	**67.75%**	**64.05%**
avg. cycle / MB	**4750**	**1321**	**368**
cycle w.o. PDE /MB	16384	4096	1024
I/O bits	256	280	376
avg. bandwidth	**1216000**	**369880**	**138368**
avg. bandwidth w.o. PDE	4194304	1146880	385024

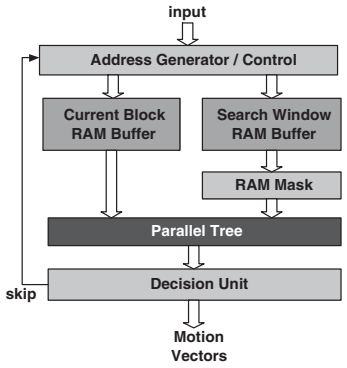

Fig. 5. Block diagram of proposed motion estimation engine

spectively. These comparison results are listed in Table 1. Taking the computation cycles into account, although the skipping ratio of parallelism one is the highest, the average cycles per block are also the highest due to the lowest parallelism. Besides, since the skipping ratio depends on the characteristics of test sequence, the computation cycles of the worst case, namely the cycles without PDE, is also listed in Table 1 as the reference. Moreover, consider the memory access bandwidth per block in depth. The search window data memory I/O are 16 pixels, 19 pixels, and 31 pixels for the parallelism one, four, and sixteen respectively. Assume the average memory access bandwidth of the search window data and the current block data to be the product of the average cycles and the I/O bit-width, then the average memory access bandwidth for parallelism one would be the highest one. It can be shown in Table 1 that the memory access bandwidth becomes lower as parallelism higher.

3. HARDWARE IMPLEMENTATION

As shown in Fig. 5, there are six modules in proposed motion estimation engine. The address generator controls two RAM buffers. Current block RAM buffers and search window RAM buffers receive data from the system to reduce the access of external main memory, and level C data reuse scheme [14] is adopted for search window data updating. The search window data are multiplexed by the RAM mask to the parallel tree. The SAD values are accumulated in the parallel tree, and the skip signal and the best-matched motion vector are sent out by the decision unit. During the skipped cycles, the parallel tree, the RAM buffers, and the decision unit are disabled using the technique of clock gating to achieve low-power consumption.

Table 2. Key features of implementation result

Technology	TSMC 0.25 um CMOS 1P5M Process	
Parallelism	4	16
Gate counts (K)	25	75
Frequency (MHz)	50	20
Voltage (V)	2.5	2.5
On chip RAM (Kbits)	20.48	20.48
Power (mW)	99.11	46.52
Power w.o. PDE (mW)	254.65	112.29

Table 2 lists the gate-level implementation results under different parallelism for CIF 30fps, block size of 16×16, and search range from -16 to +15. The average power consumptions are 99.11 mW and 46.52 mW for parallelism four and parallelism sixteen respectively, estimated by Synopsys Power Compiler under TSMC 0.25um CMOS 1P5M process.

4. COMPARISON

There are several advantages in proposed parallel tree architecture compared with previous architectures which also exploit PDE. In [11], a 1-D array-based architecture was proposed. The data flow is regular due to the pipeline operation. For this reason, the scanning order is not in spiral order, good initial SAD value from the predicted position could not be obtained, the unnecessary memory accessing could not be eliminated, and the skipping ratio is not as good as proposed ones. There are 15 shift registers in the 1-D array. Although redundant computation is eliminated in the PEs, the data in the shift registers are still shifted and results in unavoidable power consumption. As shown in Table 3, it needs 285.88MHz and is not suitable for high-end real-time applications. In [12], a 2-D array-based architecture was presented using a large amount of shift registers. The data flow is also regular due to the pipeline operation, and thus, the problems in this architecture are similar to [11]. There are 691 shift registers which would result in a large amount of power consumption. According to the simulation result of Synopsys Power Compiler under TSMC 0.25um process at 25MHz, these 691 8-bit shift registers consume 180.07mW power consumption and 42K logic gate counts. Besides, when compared with conventional architectures without PDE, the superiority of the parallel tree architecture is more obvious as in Table 3. There are fewer PEs in the parallel tree architecture but the performance is still comparable to others.

To quantify the relationship between the PE number and the computation cycles, the product *P* of PE number and cycles is used. If the number of PE increases, then the cycles would decrease. For the architectures with 100% utilization, the product should be a fixed value which equals to 262144, such as [4], [7] [8] [9], and [10]. Normalizing each product with $P_{100\%}$, the normalized product *NP* can be estimated. Assume that the normalized efficiency ratio *NR* be the inverse of the *NP*.

$$NR = P_{100\%}/P$$

Table 3. The comparison results between various architectures for CIF 30fps, block size of 16×16, search range from -16 to +15

Architecture	Description	#PE	cycles / MB	SW I/O bits	Freq. MHz	PDE	avg. cycles after PDE	avg. skip ratio %	good initial value	spiral scanning	skip RAM accessing	unavoidable data shifting	NR
[4] Yang	1-D semi-systolic	32	8192	24	97.32	N	8192	N	N	N	N	Y	1.000
[5] AB1	1-D systolic	16	24064	256	285.88	N	24064	N	N	N	N	Y	0.681
[5] AB2	2-D systolic	256	1504	128	17.87	N	1504	N	N	N	N	Y	0.681
[6] Hsieh	2-D systolic	256	2209	8	26.24	N	2209	N	N	N	N	Y	0.464
[7] Yeo	2-D semi-systolic	1024	256	24	3.04	N	256	N	N	N	N	Y	1.000
[8] Lai	1-D semi-systolic	1024	256	24	3.04	N	256	N	N	N	N	Y	1.000
[9] SA	2-D systolic	256	1024	16	12.17	N	1024	N	N	N	N	Y	1.000
[9] SSA	2-D semi-systolic	256	1024	16	12.17	N	1024	N	N	N	N	Y	1.000
[10] Tree	tree	256	1024	2048	12.17	N	1024	N	N	N	N	N	1.000
[11] Sousa	1-D systolic	16	24064	8	285.88	Y	24064	50.00	N	N	N	Y	0.681
[12] DO	2-D systolic	256	2209	8	26.24	Y	2209	44.60	N	N	N	Y	0.464
ours : 4	**parallel tree**	**64**	**4096**	**152**	**48.66**	**Y**	**1321**	**67.75**	**Y**	**Y**	**Y**	**N**	**3.101**
ours : 16	**parallel tree**	**256**	**1024**	**248**	**12.17**	**Y**	**368**	**64.05**	**Y**	**Y**	**Y**	**N**	**2.783**

The NR represents the efficiency ratio compared with the architectures with 100% utilization under the same number of PE. It can be shown in Table 3 that the normalized efficiency ratios *NR* of the parallel tree architecture are the highest ones compared with others.

5. CONCLUSION

The parallel tree architecture exploits the spatial correlations between parallel candidate block searches and performs PDE to eliminate redundant computation for low-power operation. Horizontal data sharing effectively eliminates excess memory access bandwidth while consumes fewer hardware resources. The parallel tree architecture realizes high skipping ratio at low latency and low working frequency compared with conventional array-based and tree-based architectures. When the parallelism is sixteen, the power consumption estimated by the Synopsys Power Compiler is 46.52mW at 20MHz under TSMC 0.25um CMOS 1P5M process with 75K logic gate counts. Therefore, the parallel tree architecture enables to achieve high-quality and low-power full search motion estimation, which is the essential component for high-end real-time portable video encoding system.

6. REFERENCES

[1] K. Guttag, R. J. Gove, and J. R. Van Aken, "A single chip multiprocessor for multimedia: The mvp," *IEEE Computer Graphics and Applications*, vol. 12, no. 6, pp. 53–64, Nov. 1992.

[2] S. Eckart and C. Fogg, *ISO/IEC MPEG-2 Software Video Codec*, SPIE Digital Video Compression: Algorithms and Technologies, 1995.

[3] J. N. Kim and T. S. Choi, "A fast motion estimation for software based real-time video coding," *IEEE Transactions on Consumer Electronics*, vol. 45, no. 2, pp. 417–426, May 1999.

[4] K. M. Yang, M. T. Sun, and L. Wu, "A family of vlsi designs for the motion compensation block matching algorithm," *IEEE Transactions on Circuits and Systems*, vol. 36, no. 10, pp. 1317–1325, Oct. 1989.

[5] T. Komarek and P. Pirsch, "Array architectures for block matching algorithms," *IEEE Transactions on Circuits and Systems*, vol. 36, no. 2, pp. 1301–1308, Oct. 1989.

[6] C. H. Hsieh and T. P. Lin, "Vlsi architectures for block-matching motion estimation algorithms," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 2, no. 2, pp. 169–175, June 1992.

[7] H. Yeo and Y. H Hu, "A novel modular systolic array architecture for full-search block matching motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 5, no. 5, pp. 407–416, Oct. 1995.

[8] Y. K. Lai and L. G Chen, "A data-interlacing architecture with two-dimentional data-reuse for full-search block-matching algorithm," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 2, pp. 124–127, Apr. 1998.

[9] Y. H. Yeh and C. Y. Lee, "Cost-effective vlsi architectures and buffer size optimization for full-search block matching algorithms," *IEEE Transactions on VLSI Systems*, vol. 7, no. 3, pp. 345–358, Sept. 1999.

[10] Y. S. Jehng, L. G. Chen, and T. D. Chiueh, "An efficient and simple vlsi tree architecture for motion estimation algorithms," *IEEE Transactions on Signal Processing*, vol. 41, no. 2, pp. 889–900, Feb. 1993.

[11] L. Sousa and M. Roma, "Low-power array architectures for motion estimation," in *IEEE Workshop on Multimedia Signal Processing*, 1999.

[12] V. L. Do and K. Y. Yun, "A low-power architecture for full-search block-matching motion estimation," *IEEE Transactions on Circuit and System Video Technology*, vol. 8, no. 4, pp. 393–398, Aug. 1998.

[13] W. M. Chao, C. W. Hsu, Y. C. Chang, and L. G. Chen, "A novel hybrid motion estimator supporting diamond search and fast full ssearch," in *IEEE International Symposium on Circuits and Systems*, 2002.

[14] J. C. Tuan, T. S. Chang, and C. W. Jen, "On the data reuse and memory bandwidth analysis for full-search block-matching vlsi architecture," *IEEE Transactions on Circuit and System Video Technology*, vol. 12, no. 1, pp. 61–72, Jan. 2002.