ARCHITECTURE FOR HIERARCHICAL BLOCK MOTION ESTIMATION USING VARIABLE BLOCK SIZES

Teahyung Lee and David V. Anderson

School of Electrical and Computer Engineering, Georgia Institute of Technology
Atlanta, Georgia 30332–0250 Email: taehyung,dva@ece.gatech.edu

ABSTRACT

In this paper, we propose a new architecture for hierarchical block motion estimation (HBME) algorithms using variable block sizes. The binary tree architecture (BTA) is well suited to HBME with constant block size because of the interdependence between computations in different levels of search. However, HBME algorithms based on variable block sizes can decrease the processing element (PE) utilization of BTA. The modified binary tree architecture (MBTA) presented here improves the PE utilization and area efficiency of BTA with low overhead for HBME algorithms using variable block sizes. We describe the architecture of MBTA and present the comparison of the performance with BTA for wavelet-based multi-resolution motion estimation (MRME) algorithms.

1. INTRODUCTION

Motion estimation is a key block in video coding schemes because it can reduce bit-rates dramatically by removing temporal redundancies. The block-matching algorithm (BMA) is used as a motion estimation method in most video coding systems. BMA finds a block that is most similar to a current block within a pre-defined search area (SA) in a reference frame. The full search BMA (FSBMA) has been widely researched because of its high performance and low control overhead [1].

Hierarchical block motion estimation was developed to reduce the high computational complexity and maintain comparable performance to the FSBMA. In HBME, the search for motion is performed in a hierarchical way, where the size of the block and/or the search area can vary depending on the hierarchy level. Each subsequent level in HBME is dependent on the motion vector of the previous level. Therefore, low-latency architecture is required in HBME in order to finish the current level before the beginning of the next level of motion estimation. The binary tree architecture (BTA) is one of the most suitable architectures for this requirement.

Previous work regarding architectures for HBME are reported in [2]-[3]. Jehng et al. [2] and Gupta and Chakrabarti [4] presented architectures using BTA for constant computational block sizes. Lee et al. [3] proposed a 1-D systolic array processor using the largest common block size among variable block sizes in hierarchical motion estimation algorithm.

In this paper, we suggest a modified binary tree architecture (MBTA) for hierarchical block motion estimation algorithms using variable block sizes. This architecture achieves low-latency by using the characteristic of BTA and performs motion estimation for different block sizes without decreasing the processing element (PE) utilization. MBTA can execute the block matching operations for several candidate blocks of small size in a low-resolution image concurrently if the block size is small enough compared to the number of PE's.

The rest of the paper is organized as follows. A brief introduction of HBME is presented in section 2. The binary tree architecture is described in section 3. The proposed architecture is explained in section 4. In section 5, we show and compare the performance of the proposed architecture with BTA and conclusions are presented in section 6.

2. HIERARCHICAL BLOCK MOTION ESTIMATION (HBME) ALGORITHMS

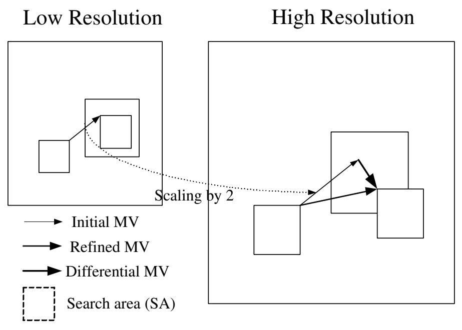

Fig. 1. Multi-resolution motion estimation.

Hierarchical block motion estimation has been developed to reduce the computational complexity and maintain good performance compared to the full search block-matching algorithm. In hierarchical block motion estimation, the size of the block and/or the search area varies depending on the level of hierarchy. At lower levels of hierarchy, larger block sizes are used to estimate the broad motion of the image, while at higher levels of hierarchy, smaller block sizes are employed to estimate the detailed motion of the image. However, the block size in computation for motion estimation can be the same. Therefore, the HBME algorithms can be divided into two categories depending on computational block sizes at different levels, constant block size and variable block size. The efficient architecture of the constant block size algorithms are more appealing because fixed block size helps to increase the processing element utilization and simplify data-flow. The three-step hierarchical search (3-SHS) is considered as one of the best constant block size algorithms. Some well-defined architectures for the constant block size algorithms are presented in [2], [4],and [5]. In the variable block size case, a promising solution is the multi-resolution motion estimation (MRME) algorithm. In conventional MRME schemes, motion vectors (MV's) are first estimated at the lowest resolution. This is reasonable since most of the image energy is preserved at this resolution. And then MV's are refined at other finer resolutions depending on the corresponding initial MV's at lower resolutions (see Fig. 1). Each lower resolution image is subsampled by 2 from the previous higher resolution image. A few hardware architectures have been proposed for MRME-type algorithms in [3] and [6].

Most of the hardware architectures for motion estimation algorithms have been implemented for a fixed block size. However, in order to implement the MRME architecture efficiently, processing elements have to execute the variable block sizes to find the best motion vector. Low latency for lower resolution images are required since there is inter-level dependency among different resolution levels. The computational complexity of MRME for a 16 × 16 block size at the finest resolution is derived below,

$$\begin{aligned} C_{mrme} &= C_{2\times 2} + C_{4\times 4} + C_{8\times 8} + C_{16\times 16} \\ &= \{K_3(w_3)^2\left(\frac{1}{2^3}\right)^2 + K_2(w_2)^2\left(\frac{1}{2^2}\right)^2 \\ &\quad + K_1(w_1)^2\left(\frac{1}{2}\right)^2 + K_0(w_0)^2(1)\ \} \\ &\quad \times (M \times 16^2) \times \left(\frac{W \cdot H}{16^2}\right) \cdot f\ (operations/s), \end{aligned} \tag{1}$$

where $C_{n\times n}$ is the searching complexity associated with $n \times n$ block size. $W \cdot H$ is the image size, $(w_i)^2$ is the search range at level i of resolution, M is the number of operations required for finding the sum of absolute difference, f is the image frame rate, and K_i the number of images at level i of resolution.

3. BINARY TREE ARCHITECTURE

The binary tree architecture (BTA), which is especially well suited for HBME algorithms with constant block size such as 3-SHS, was presented in [2]. This architecture achieves low latency and computation time. Each stage of the tree architecture can be regarded as a parallel pipeline stage with the data path length equal to the tree height as described in Fig. 2. The total latency in the BTA is $log_2^p + 2$, where p is the number of leaf processors. The tree architecture makes no assumption of regular data-flow for block matching operation within the SA. This makes the tree architecture suitable for HBME because it provides low-latency and irregular data flow is tolerated. The BTA is well matched to HBME, however, the processing element efficiency decreases when the block size is variable.

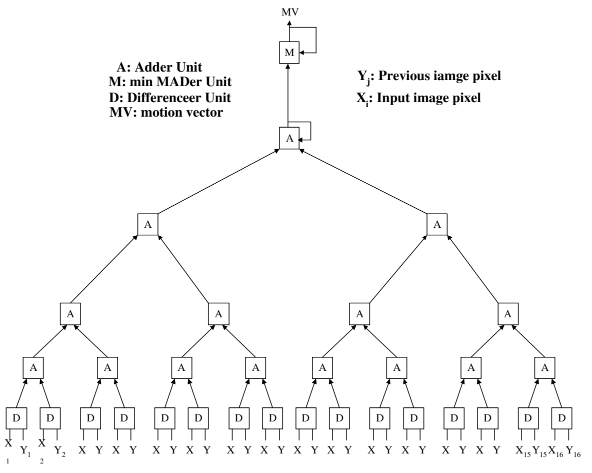

Fig. 2. Binary tree architecture for $N^2 = 16$, where N^2 is a block size.

4. THE PROPOSED ARCHITECTURE

In this section, we describe the proposed binary tree architecture and data shuffling network to overcome the decrease in PE efficiency when using BTA for variable block size.

4.1. Modified Binary Tree Architecture

To implement an HBME algorithm with interdependence between the hierarchy levels, BTA is a good solution except that it does not work well for variable block sizes. To maintain the characteristic of the binary tree architecture without the loss of PE utilization efficiency, parallel computations without data dependence could be exploited. In general this is the case if HBME does not require dependency within a level of hierarchy. In such a scenario, we can perform parallel computation within the search range (SA) of the current block with the flexible data-flow adaptation within a level of hierarchy. For the variable block size algorithms, the height of the

BTA should be changed. We propose the modified binary tree structure to make the BTA height flexible (see Fig. 3). Efficient utilization of MUX's and DEMUX's within the BTA is enough to achieve this functionality. Therefore, the overhead for the MBTA is low. Figure 3 shows the MBTA with a 2×2 block as the smallest block size. The latency of the MBTA varies depending on the block size as opposed to the BTA, which has the same latency regardless of the block size. The latency of the MBTA is $log_2^p + 2$ for a full tree and $log_2^p + 1$ for a block size smaller than the number of leaf processors, p. In addition, we can reduce the number of cycles to perform the HBME by using the idle leaf processors in smaller block sizes.

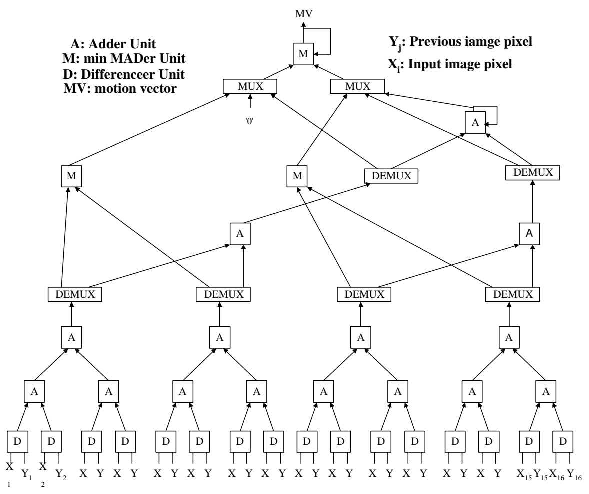

Fig. 3. Modified binary tree architecture for $N^2 = 16$, where N^2 is the largest block size.

4.2. Data Shuffling Network

The parallel computation in the MBTA and BTA needs appropriate data transfer from memory to leaf processors . We need as many memory modules as the number of leaf processors to maximize processor utilization. An image pixel data distribution among memory modules is denoted in Fig. 4. Four parallel processing of block matching operation is possible for a 2×2 block size in Fig. 4. As we can see in Fig. 5, which is a data shuffling network based on Fig. 4, some data need to be distributed to more than one leaf processor in order to calculate the motion by parallel computation within the SA. To make a efficient data distribution network, we propose a data shuffling network for the MBTA. This is composed of 3 stages. The first stage is fed by memory modules and performs the column position movement of the data. The next stage changes the row position of the data and the last stage feeds the correct data to the leaf processors using MUX's. The A, B, and C in Fig. 5 are switch units, which were used in [4]. Even though channels among switch units in Fig. 5 are tailored to [7] and [8], we can change these structure depending on HBME without problem. The number of switch levels are $2 \times N + 2$ in a N^2 block size. By using the data shuffling network and parallel processing, we can reduce the number of cycles for HBME.

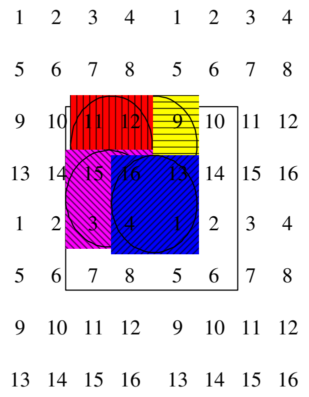

Fig. 4. Image pixel data distribution among 16 memory modules for 16 leaf processors and a 2×2 block size. (Four parallel processing of block matching operation is possible for a 2×2 block size)

5. PERFORMANCE COMPARISON AND DISCUSSION

In this section, we compare the efficiency of the MBTA and BTA for wavelet-based MRME algorithms, which are HBME algorithms with variable block sizes [7], [8]. In conventional wavelet-based MRME schemes, motion vectors (MV's) are first estimated at the lowest resolution. And then MV's are refined at other subbands depending on the corresponding initial MV's at the lowest resolution or those of one-level lower resolution subband with same direction, such as horizontal, vertical, and diagonal (Fig. 1). The search range is the same for all subbands. If we assume the block size for the full resolution image to be 16×16 then the smallest block size in the algorithm is 2×2 and largest block size is 8×8. In equation (1), $K_3 = 1$, $K_2 = K_1 = 3$, $K_0 = 0$, and which are the number of subbands at different levels.

The parameter values from [4] are used to perform the comparison between different architectures. Therefore, we assume that on-chip memory access time is 70 ns and off-chip memory access time is 200 ns. Table 1 shows the area estimation based on the minimum number of adders for real-time video encoding because most data-path units in our architecture can be implemented using adders and the areas of MUX's and DEMUX's are much smaller than other units in the MBTA. The number of adders for the BTA and MBTA are $2 \times N^2$ and $2 \times N^2 + 2(N/2 - 1)$, where N^2 is a block size. According to Table 1, the number of adders for the MBTA ranges from 100 to about 60% of that needed for the BTA.

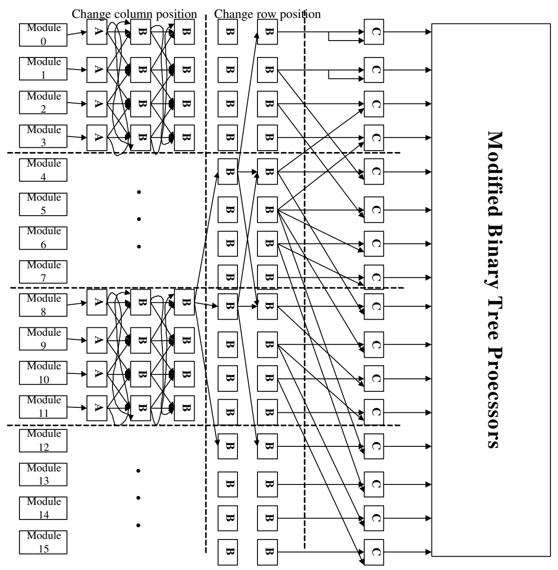

Fig. 5. Data shuffling network for modified binary tree architecture with 16 leaf processors for [7] and [8].

The area for both is the same up to 8 adders, which is same as a 2×2 block, but as the image size, frame rates, and/or search range increases the MBTA needs less area than the BTA. For up to 8 adders, there is very little improvement since parallel computation is impossible. As the number of adders increase, the MBTA has the chance to perform parallel computations leading to improvements. Beyond a certain number of adders, the reduction in number of cycles can also lead to reduction in the required number of leaf processors for real time execution. Table 2 shows the number of execution cycles for the BTA and MBTA with the same number of leaf processors. As expected, with the increase in number of leaf processors, MBTA provides gains in terms of execution cycles because more parallel computations are possible. The speed-up is from around 1 to 1.62. Therefore, MBTA shows better process utilization than BTA.

Hardware resource requirement comparison for MRME MBTA versus BTA (for block size of 16 x 16)				
Picture format	No. of adders (search range = 4 x 4)		No. of adders (search range = 8 x 8)	
	MBTA	BTA	MBTA	BTA
288x352, 10 Hz	4	4	16	16
288x352, 30 Hz	8	8	34	64
576x720, 25 Hz	34	64	142	256

Table 1. Area estimation by the minimum number of adders for real-time video encoding.

6. CONCLUSIONS

In this paper, we propose a modified binary tree architecture and a data shuffling network for hierarchical block motion estimation algorithms using variable block sizes. Using parallel computations, the MBTA improves the processor element utilization and area efficiency compared to the binary tree architecture, without adding significant overhead. Parallel computations are easily performed with the support of the proposed data shuffling network.

We compare the performance of BTA and MBTA for wavelet-based multi-resolution motion estimation algorithms. For real-time execution, the estimated area for MBTA based on the minimum number of adders is reduced (relative to BTA) as the image size, frame rates, and/or search range increases. This is because the reduction in the number of cycles decreases the required number of leaf processors for real time execution. For the same number of leaf processors, the relative decrease in execution cycles of MBTA is higher as the number of leaf processor increases because more parallel computations are possible. Based on these results, we conclude that MBTA is an effective way to improve the process utilization with low overhead.

Execution cycles for MBTA and BTA Block size = 16 x 16						
Picture format	No. of execution cycles (in mega cycles) search range = 4 x 4					
	8 leaf processors		16 leaf processors		32 leaf processors	
	MBTA	BTA	MBTA	BTA	MBTA	BTA
288x352, 10 Hz	2.032	2.154	1.018	1.204	0.507	0.824
288x352, 30 Hz	6.095	6.463	3.053	3.612	1.521	2.471
576x720, 25 Hz	20.78	22.07	10.41	12.35	5.225	8.464

Table 2. The execution cycles for the same number of leaf processors in BTA and MBTA.

7. REFERENCES

[1] 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.

[2] Y.-S. Jehng, L.-G. Chen, and T.-D. Chiueh, "An efficient and simple vlsi tree architecture for motion estimation algorithms," *IEEE Trans. Signal Processing*, vol. 41, pp. 889–900, Feb. 1993.

[3] J.H. Lee, K.W. Lim, B.C. Song, and J.B. Ra, "A fast multi-resolution block matching algorithm and its lsi architecture for low bit-rate video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, pp. 1289–1301, Dec. 2001.

[4] G. Gupta and C. Chakrabarti, "Architectures for hierarchical and other block matching algorithms," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 5, pp. 477–489, Dec. 1995.

[5] H.M. Jong, L.-G. Chen, and T.-D. Chiueh, "Parallel architectures for 3-step hierarchical search block-matching algorithm," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 4, pp. 407–416, Aug. 1994.

[6] B.-M. Wang, J.-C. Yen, and S. Chang, "Zero waiting-cycle hierarchical block matching algorithm and its array architecture," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 4, pp. 18–28, Feb. 1994.

[7] Y.-Q. Zhang and S. Zafar, "Motion-compensated wavelet transform coding for color video compression," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 2, pp. 285–296, Sept. 1992.

[8] J. Zan, M.O. Ahmad, and M.N.S. Swamy, "New techniques for multiresolution motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, pp. 793–802, Sept. 2002.