VLSI-BASED PARALLEL ARCHITECTURE FOR BLOCK-MATCHING MOTION ESTIMATION IN LOW BIT-RATE VIDEO CODING

Donglai Xu and John Bentley

School of Science and Technology, University of Teesside
Middlesbrough, TS1 3BA, UK
D.Xu {J.P.Bentley} @Tees.ac.uk

Abstract: In this paper, we proposed a flexible VLSI-based parallel processing architecture for an improved three-step search (ITSS) motion estimation algorithm that is superior to the existing three-step search (TSS) algorithm in all cases and also to the recently proposed new three-step search (NTSS) algorithm if used for low bit-rate video coding, as with the H.261 standard. Based on a VLSI tree processor and an FPGA addressing circuit, the architecture can successfully implement the ITSS algorithm on silicon with the minimum number of gates. Because of the flexibility of the architecture, it can also be extended to implement other three-step search algorithms.

1. IUTRODUCTION

It is well known that motion estimation algorithms play an important role in video sequence compression. Many fast block-matching algorithms for motion estimation have been proposed because of their lower computation overhead than that of full-search block-matching algorithm, such as the existing three-step search (TSS) algorithm [1] and the recently proposed new three-step search (NTSS) algorithm [2]. Recent studies show that the motion vector distribution of a real world image sequence within the search window is highly centre-biased. Based on this fact, we propose an improved version of the well-known TSS method, the ITSS algorithm [3], specifically aiming towards low bit-rate video coding applications. The ITSS has much better performance and faster speed than the original. Compared to the NTSS, its performance is better when applied to our target applications, such as videophone, and its speed is faster as well, without any direct or hidden costs.

Compared with TSS, the ITSS uses the same number of checking points in each step, but a different search pattern. This leads to better performance of ITSS while maintaining the same data flow of TSS. That means the architecture for TSS can also be used for ITSS. Many VLSI architectures [4], [5], [6] have been proposed for TSS. However, these architectures either have low throughput, or high hardware cost, or too little flexibility. To avoid these drawbacks and to implement the proposed ITSS algorithm on silicon with the minimum number of gates, a low-latency and high-throughput parallel pipeline computing architecture, based on a VLSI tree processor and an FPGA addressing circuit, is presented. Owing to its simple and modular properties, the tree processor is suitable for VLSI implementation, and because of the use of FPGA to implement addressing and control circuits, the architecture is flexible enough to implement different three-step search algorithms. Furthermore, the tree processor can be decomposed into sub-trees to reduce hardware cost and pin count. Memory interleaving and pipeline interleaving are also employed to enhance memory bandwidth and to raise pipeline utilisation to 100%.

In the next section, we will describe the ITSS algorithm. In section 3, we will present the computing architecture for ITSS. A sample design for videophone application will be discussed in section 4. Finally, the conclusion will be given in section 5.

2. ITSS ALGORITHM

The experimental results in [2] have shown that the block motion field of real world image sequences is usually gentle, smooth, and varies slowly. It results in a centre-biased global minimum motion vector distribution instead of a uniform distribution. For such a distribution, we have developed an improved three-step search algorithm (ITSS) which uses a centre-biased checking point search pattern adapted to the centre-biased motion vector distribution, hence its performance is expected to be far better than that of TSS which uses a uniformly distributed checking point search pattern. Additionally, the ITSS employs a smaller number of search points than TSS to speed up block matching. The details of the algorithm are given in the example described below.

Following earlier block-matching techniques, our example takes a block size of 16×16 pixels and a maximum search range of ±7 pixels in both horizontal and vertical directions. The mean absolute error (MAE) is used as an appropriate estimate of the block distortion measure (BDM). For a given *(x,y)*, the MAE between *block(m,n)* of the current frame and *block(m+x,n+y)* of the previous (reference) frame is defined as:

$$MAE_{(m,n)}(x,y)=\frac{1}{256}\sum_{i=0}^{15}\sum_{j=0}^{15}\left|f_k(m+i,n+j)-f_{k-1}(m+x+i,n+y+j)\right|$$

where $f_k(i,j)$ and $f_{k-1}(i,j)$ are the pixel intensities at position *(i,j)* of the current frame *k* and the previous frame *k-1* respectively, and the *block(m,n)* is the block with its upper left corner at position *(m,n)* of a frame. The first step of the algorithm employs a centre-biased search pattern with nine checking points on a 5×5 window (Figure 1) instead of the 9×9 window with TSS. The centre of the search window is then shifted to the point with minimum BDM. The search window size of the next two steps depends on the location of the minimum. If the minimum lies at the centre of the search window, the search will go to the final step (step 3) with a 3×3 search window. Otherwise, the search window size is maintained at 5×5 for step 2. In the final step, the search window is reduced to 3×3 and the search stops at this small search window.

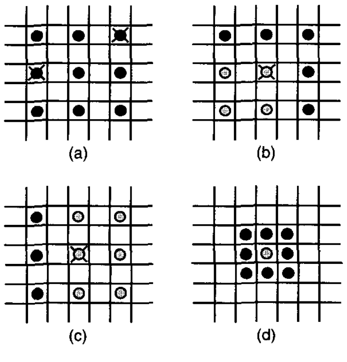

Figure 1 Search pattern of the ITSS. (a) First step centred on centre pixel; (b) Second step centred on a corner pixel; (c) Second step centred on a middle pixel; (d) Third step

3. PROPOSED ARCHITECTURE

The system architecture of ITSS, which consists of memory subsystem, tree processor, and address generation circuit, is described in Figure 2.

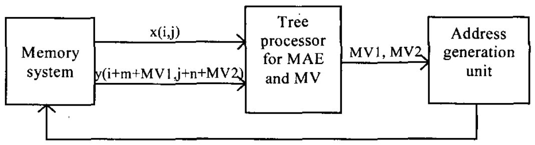

Figure 2 Block diagram of system architecture for ITSS algorithm

The memory subsystem stores the current and previous video frames by which the tree processor is provided with input data. To enhance memory bandwidth, the memory system adopts multiple memory modules, such as an N×N module array, to apply memory interleaving for simultaneous accesses. In addition, the memory system is divided into multi-banks and multi-ports in order that the memory cycle time can afford the huge partitions of pixels in a matching block to be interleaved.

The tree processor, with the major computation overhead in this architecture, computes MAE and determines motion vector (MV). It can be a modular-processing engine because it is not directly related to the search pattern of ITSS algorithm and the position of search points. Therefore, we implement it in a single-chip VLSI. As shown in Figure 3, pixels of the two 16x16 blocks respectively in the present frame X and in the previous frame Y are fed to the tree processor. The processing element D computes differences between pixels of the present frame X and those of the previous frame Y in parallel. The differences are concurrently accumulated by the adders that comprise the binary tree architecture. Each tree level can be viewed as a parallel pipeline stage; not only does this reduce the computational data path length (tree height) but also prevents the data from skewing in the course of doing the parallel computations. However, extensive pipelining always leads to lost resources if computed results are not ready when are needed, thus resulting in hazards. In the pipelined computation of ITSS, during its three-step operations, the current step must complete its computation before the next step begins; therefore, it will lead to lost resources because the computed results of the current step

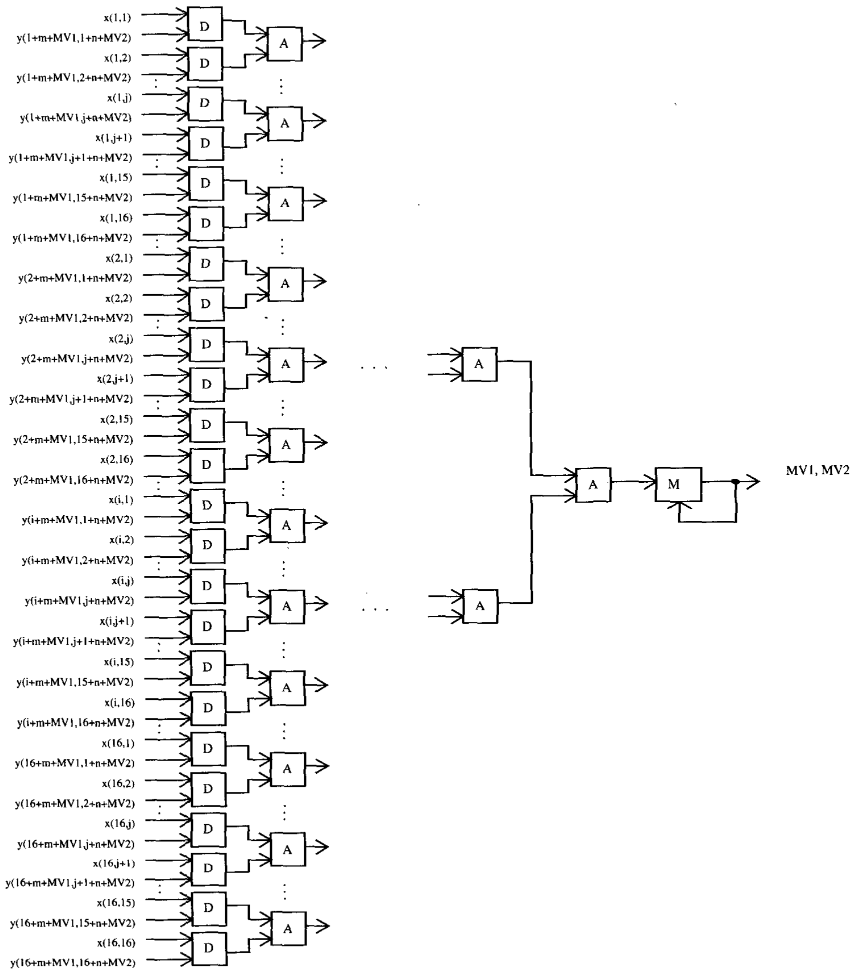

Figure 3 Tree architecture processor for MAE and MV computation

are not ready when they are needed by the next step. To avoid performance degradation caused by these hazards, pipeline interleaving is employed in the processor.

The address generation unit will address memory for computing MAE and motion vector. It controls the search pattern of the ITSS algorithm and assigns search points by selecting the proper block address and pixel address within the block. By changing the design of this component, different hierarchical search algorithms could be implemented in this architecture. Therefore, FPGA implementation is used, to provide future flexibility.

4. APPLICATION DESIGN

In VLSI implementation of the tree processor for specific applications, it is not always necessary to implement the whole tree architecture on a single chip because the area-speed product is always taken as the most important consideration. To reduce hardware cost, the tree architecture can be cut into sub-trees.

In the following, we consider the hardware implementation of the tree architecture (N=16) for a videophone application. Suppose dynamic random access memories (DRAM's) with 200ns access time are used to implement the memory system and the time to latch a pixel from a memory module is simulated to be about 50ns. If a 1/C-cut sub-tree architecture and a B-bank memory system are adopted, then we have the following constraint:

$$\frac{200\text{ns} \times \text{B}}{(256 \text{ memory modules})/\text{C}} \geq 50\text{ns}; \text{ and thus } \text{C} \times \text{B} \geq 64.$$

Thus the 1/C-cut sub-tree architecture must connect with a memory system of at least 64/C memory banks to allow simultaneous accesses. On the other hand, the period time for each pipeline stage is also constrained by memory cycle time and is 200ns. For videophone application (288×352 frame size, 16×16 block size, and 10Hz frame rate), the throughput required by the processing system is

$$\frac{352 \times 288}{16 \times 16} \times 10 = 3.96\text{k blocks/sec ond,}$$

so that 252.5us is allowed for each vector. With a 1/C-cut sub-tree, the number of time cycles required to compute a motion vector is 27×C. Thus, the sub-tree architecture for the application is constrained by:

$$200\text{ns} \times (27 \times \text{C time cycles for 1/C - cut sub - tree}) \leq 252.5\text{us};$$
$$\text{and thus C} \leq 46.76.$$

Consequently, the 1/32-cut (C=32) sub-tree architecture with two memory bands (B=2) for this videophone application is optimal in speed or hardware cost for the realisation of ITSS algorithm.

The real throughput rate depends on the clock rate applied to it. For the above application, the clock cycles required to estimate a motion vector is 27×C=864. Suppose a clock rate of E MHz is applied to the sub-tree processor. Then, in order to estimate motion vector in real time for the application, there should be the following constraints:

$$\frac{\text{E} \times 10^6}{864} \geq 3.96 \times 10^3; \text{then E} \geq 3.42 \text{ MHz.}$$

This means that the minimum clock rate to do real-time block matching is 3.42MHz. The single chip 1/32-cut sub-tree is currently synthesised and simulated in a 0.7um CMOS standard cell technology, requiring about 5000 equivalent gates.

5. CONCLUSIONS

In this paper, we describe a flexible and efficient architecture for implementation of the improved ITSS algorithm. In this architecture, simple and modular tree structures facilitate VLSI implementation, the FPGA designs for addressing and control circuits improve the flexibility of the system, and memory interleaving and pipeline interleaving enhance the overall system computing performance. Furthermore, the tree-cut technique is introduced to reduce the hardware cost of the tree processor; but still allowing real-time processing requirements to be met in the sub-tree architecture for specific applications. Finally, a videophone application is considered and its speed and hardware cost trade-offs are evaluated to find out an optimal VLSI implementation.

REFERENCES

1. T. Koga, K. Iinuma, A. Hirano, Y. Iijima and T. Ishiguro, "Motion-compensated inter-frame coding for video conferencing", *National Telecommunication Conference*, New Orleans, USA, November 1981, pp. G5.3.1-G5.3.5.
2. R. Li., B. Zeng and M. L. Liou, "A new three-step search algorithm for fast block motion estimation", *IEEE Trans. on Circuits and System for Video Technology*, Vol. 4, 1994, pp. 438-442.
3. W. Booth, J. M. Noras and D. Xu, "A novel fast three-step search algorithm for block-matching motion estimation". *In: Chin R., Pong T.C. (eds.): Computer Vision. Lecture Notes in Computer Science*, Springer-Verlag, Vol. 1352, 1998, pp. 623-630.
4. Z. He, M. L. Liou, P. C. H. Chan and R. Li, "An efficient VLSI architecture for new three-step search algorithm", *IEEE 38th Midwest Symposium on Circuits and Systems*, USA, August 1995, pp. 1228-1231.
5. G. Gupta and C. Chakrabarti, "Architectures for hierarchical and other block matching algorithms", *IEEE Trans on Circuits and Systems for Video Technology*, Vol. 5, 1995, pp. 477-489.
6. P. M. Kuhn and W. Stechele, "Complexity analysis of emerging MPEG-4 standard as a basis for VLSI implementation", *Proceedings of The Society of Photo-Optical Instrumentation Engineers (SPIE)*, Vol. 3309, 1998, pp. 498-509.