ISSN: 2278 - 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering
Vol. 1, Issue 5, November 2012

Efficient Motion Estimation by Fast Three Step Search Algorithms

Namrata Verma[1], Tejeshwari Sahu [2], Pallavi Sahu[3]

Assistant professor, Dept. of Electronics & Telecommunication Engineering, BIT Raipur, Chhattisgarh, India [1,2]

Lecturer, Dept. of Electronics & Telecommunication Engineering, BIT Raipur, Chhattisgarh, India[3]

Abstract*: Video Compression algorithms have a large number of applications ranging from Video Conferencing to Video on Demand to Video phones. It reduces the high bandwidth requirement to a large extent. Block matching is a widely used method for stereo vision, visual tracking and video compression. Efficient motion estimation is an important problem because it determines the compression efficiency and complexity of a video encoder. Many fast algorithms for block matching have been proposed in the past. This paper present a parallel architecture for a fast three step search (FTSS) algorithm for motion estimation, which involves reduced number of checking points compared to the standard three step search (TSS) algorithm. Block matching motion estimation is the most important part of video coding system. Degradation of performance while applying the improved algorithm to several standard images has been shown to be insignificant compared to the standard algorithm. The proposed architecture uses only three processing elements accompanied with use of intelligent data arrangement and memory configuration. Because of their low area requirement and low memory bandwidth requirements, the proposed architecture provides an efficient solution for real-time motion estimations as required by video applications of various data rates, from low bit-rate video to HDTV systems. FTSS is much more robust, produces smaller motion compensation error, and has a very compatible computational complexity.

Keywords: Video Coding, Motion Estimation, Full Search Block matching, Three Step Search, VLSI Architecture.

I. INTRODUCTION

Due to the stringent requirements of the real time video playback system, video coding is the most essential part of any visual application. Furthermore, due to the limited channel capacity, these applications require a very high compression ratio as well. It is well known that Motion Estimation algorithms play an important role in video sequence compression. The motion estimation is a technique, which exploits and tries to minimize the temporal redundancy present between the successive frames. ME, which is computational very intensive, involves about 80% of the computational power of the encoder [1]. Block Matching motion estimation is one of the most efficient and popular techniques to removes temporal redundancy present between the frames [2]. In this technique, the current frame is divided in to blocks (candidate blocks), and for each blocks one searches for the best matched block (within the search range) in an available previous frame. Motion Vector (MV) is defined as the displacement between the candidate block (CB) position and the best matched one in the previous (Stored) frame. The predicted Macro Blocks (MB) for the current frame is produced by Motion Compensation (MC). The residual MB is the difference between the original MB and the predicted MB. The goal of video compression is to minimize the energy contain in the residual MB. It is clear that a better prediction will reduce the energy contain in the residual MB. It is necessary to discuss trade-off for algorithms and architectures of motion estimation, applications, and take into account design issue, chip area, speed, I/O bandwidth, memory band width, power consumption, image quality, and some requirements for application. 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. One of the main research goals has been the reduction of computational complexity and the power consumption of the motion estimation while keeping quality of image.

The block-matching algorithms (BMA's) [3]-[7] for motion estimation can be realized by two different approaches, namely programmable processors and dedicated hardware. And they are programmable processors and dedicated hardware Programmable general purpose processors are very flexible, but this advantage is not essential for BMA's because of their fixed computation types. Moreover, the very high computational complexity requires multiple high-performances (thus high-cost) video signal processors. In contrast, mapping the BMA to a dedicated architecture provides a less flexible yet much more efficient solution, because it can be optimized for this special purpose. For this reason, dedicated motion estimation architectures have been applied to several video coding chip sets for low bit-rate applications. Furthermore, dedicated BMA chips or modules can be combined with programmable processors to provide flexible video encoders with adequate throughput rates.

ISSN: 2278 - 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering
Vol. 1, Issue 5, November 2012

Many fast BMA's have been developed to reduce the very high computational complexity of the optimal full search (FS) procedure. One of the commonly used fast motion estimation algorithm is the 3-step search (TSS) is considered as one of the best algorithms and is recommended by CCITT RM8 [8] and MPEG SM3 [9]. But 3-step hierarchical search (TSS) will have more complexity and more number of checking points. This paper presents a fast three step search (FTSS). Compared to other fast algorithm, FTSS has better performance which is close to the performance of full search block matching algorithm for video phone and video conference applications. The present paper has developed an architecture, which involves three PEs, to implement improve the algorithm. The architecture makes use of intelligent data reuse and memory configuration as well as a technique to reduce external memory accesses to pave way for low memory bandwidth requirement. This, coupled with low area requirement makes the proposed architecture efficient and suitable for low power low bit rate video processing applications.

This paper is organized as follows. The next section describes the detail of the FTSS algorithm. Section III covers the efficient 3-PE's FTSS architecture. Results and Comparisons are presented in section IV. Finally, conclusions are given in section V.

II. FAST MOTION ESTIMATION ALGORITHM

Generally, there is a relationship between the displacement of current block and the displacement of previous block which is known as motion vector. This is particularly true in low bit-rate video applications, including videophone and video conferencing, where fast and complex movements are involved rarely. The FTSS algorithm [3] makes use of the directional information from adjacent previous block motion vector.

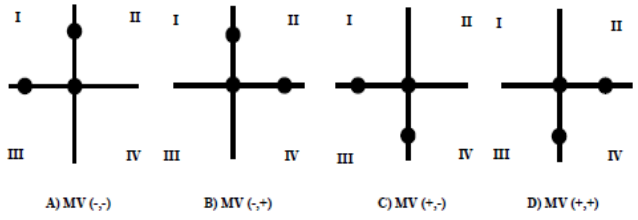

Fig.1 Direction of the current MV from the previous MV

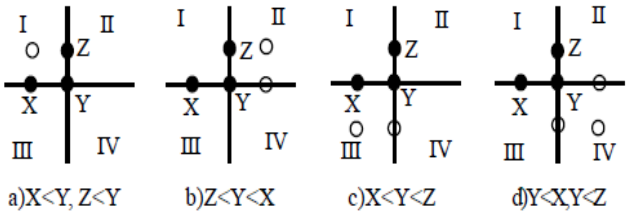

Fig.2 Next checking points after checking 3 points

The current motion vector is estimated by the directional information of the previous motion vector. The adjacent motion vector for the current frame helps in the prediction of the direction of the current motion vector. The motion vector of the left neighbour i.e. the sign of previous motion vector can be determine from the current motion vector shown in Fig. 1. The set of checking points (Search points) in the following phase, when the previous motion vector is directed towards top left (as shown in Fig.1(a) is found out based on four exhaustive conditions as depicted in Fig.2. The second phase of checking points for the remaining three possible previous motion vectors (as shown in Fig.1 (a), (b), (c) will be determined in a similar manner. By selecting the direction from the contiguous previous motion vector instead of random search direction as shown in Fig.1, in the first step the probability of occurrence of 4 checking points will be increased. We can check only four searching points in the first step, not five or six checking points as [7] by accurate estimation of direction of current motion vector. By using the directional information, we can decrease one or two checking points in the first step compared with the Lu and Liou's algorithm [7]. According to the original TSS algorithm the number of checking points for the first step is nine. There is a reduction by 4 or 5 points in each step by the FTSS algorithm, while ensuring a performance close to TSS algorithm.

A. Calculation of Mean Absolute Difference (MAD)

Taking a block of size N x N, the block motion estimation searches for a motion vector that yields the minimum Mean Absolute Difference (MAD) within a neighbourhood. Suppose that the maximum motion in vertical and horizontal direction is $\pm$W. If the Full Search (FS) method is used then, there are totally $(2W+1)^2$ motion vectors to be checked, each one subsequent to a point in the search window. The MAD values on all points in the search window form an error surface. As a criterion of distortion, the MAD is calculated for each candidate location (u, v) as

$$MAD(u,v)=\sum_{l=0}^{N-1}\sum_{m=0}^{N-1}|F_t(x+l, y+m) - F_{t-1}(x+l+u, y+m+v)|$$

Where Ft and Ft-1 refer to the blocks in a current and the previous frames which are to be compared, (x, y) is the coordinate of the left-top pixel of the current block in Ft, and the values of u and v are limited between d1 and d2. In low bit rate video applications, the search is usually performed for an area of size 16 × 16, thus d1=d2=7. The FTSS algorithm [1] differs from the TSS [2] by

- Considering previous motion vector

ISSN: 2278 - 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering
Vol. 1, Issue 5, November 2012

- For each step, in first phase it considers the three search points as specified in Figure 1, in second phase it considers one or two search point(s) as specified in Figure 2.

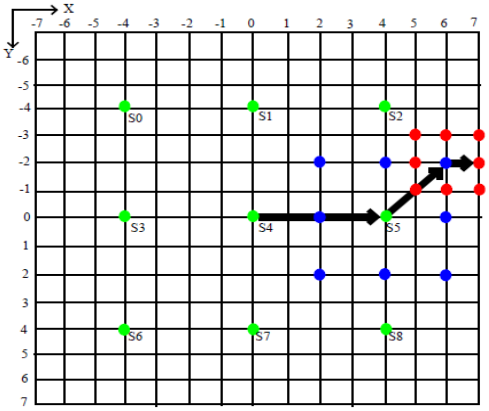

Fig.3 The FTSS Algorithm (the 9 search points in a step labelled by (*S*0- *S*8)

So for each step it checks four or five search points. So the minimum number of search points is 12 and the maximum number of search points is 15. But in TSS algorithm number of search points is fixed to 25. The arrangement of search points is shown in Figure 3.

III. ARCHITECTURE FOR FTSS

The architecture for FTSS consists of memory sub system, control unit, and process control unit. As shown in figure 4. The memory subsystem is having three search area buffers, which are used to store the search block from external memory. This blocks outputs the index numbers of the blocks of the previous frame that have to be compared with each of the current frame block with candidate blocks based within the search range. The state machine block in control unit enables this block immediately after the idle state. This block gets its inputs from search and gives outputs to process control unit. Two half search area buffers are used for each task. And these buffers are again stored in to nine memory modules to enhance memory bandwidth, and to apply memory interleaving for simultaneous accesses.

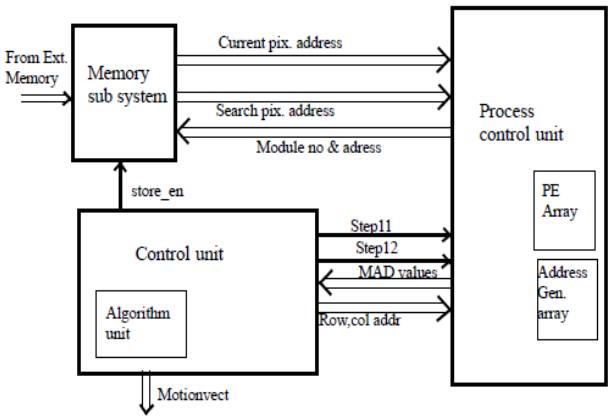

Fig.4 Block diagram of proposed architecture for FTSS Algorithm

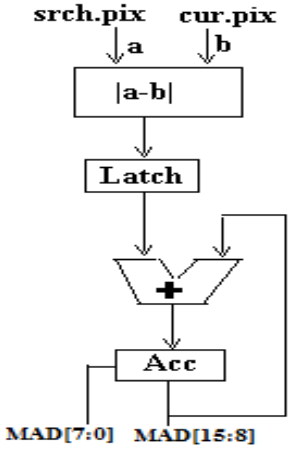

Fig. 5 Block diagram for processing element (PE)

Functions of control unit are listed as follows:

- Control unit is made by a finite state machine which generates the timing signals and control signals for all the blocks to control their operation by enabling and disabling signal.
- Activates the required half search area buffer to access the external memory.
- Send search points base addresses to process control unit for each step to calculate MAD.
- Take the MAD values, from the process control block, according to min MAD value.
- This minimum MAD is fed to address generator to produce the new address for the next search

ISSN: 2278 - 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering
Vol. 1, Issue 5, November 2012

- After completion of three steps find the motion vector.

Functions of process control unit are given below:

- Process control unit itself is having processing element array unit (PE Array) and address generator array unit (Address Gen. Array).
- Takes step activation signal and base addresses of search points from the control unit. From base address and offset address, the address generator calculates memory module number and module address.
- Process control unit sends module numbers and module addresses to memory sub system, then take search pixels from memory sub system, and current pixel from ext memory.
- Calculate MAD values for each search point and send to control unit.

The address generator array is having three address generators. Each address generator will generates module number and module address from base address received by the process control unit and offset address. According to FTSS algorithm each step is having 2 phases. Algorithm unit will selects, 3 search points in 1st phase out of nine, according to sign of adjacent previous motion vector. Base addresses of the selected search points will send to the process control unit. After 256 clocks process control unit will send the MAD PE array consists of three processing elements (PEs). Each PE will take search pixel and current pixel as input from memory sub system and find the mean absolute difference (MAD). The difference will cumulatively add and gives the MAD value for each search point (16 × 16 blocks) for every 256 clocks. The structure of PE is shown in Fig.5.

The proposed parallel architecture with 3 PE's is a good choice for the FTSS [3] algorithm. Because there are two phases in each step, according to FTSS algorithm first stage in each step number of checking points are 3.In second stage in each step number of checking points are 1 or 2. So the number of parallel checking points never exceeds three, which entails only three PE's. However, the difficulties on data addressing and interconnection complexity make it hard to implement. The present paper suggests a hardware structure that can effectively solve all these problems; this architecture is based on two data management techniques, namely (1) an on chip buffer configuration for reducing the number of external memory accesses, (2) the residual memory interleaving for parallel data accesses.

IV. RESULTS AND COMPARISONS

The proposed architecture has been simulated in Xilinx- ISE 8.1i platform, with vertex4 device family and MATLAB. In this simulation, images which we have taken was 'Lena' and 'Palace' of size 128X128 and block size 16X16, Search range was . Fig.6 shows the number of checking points per block for TSS and FTSS. Next simulation experiments, the block size is fixed at 16 x 16. Block matching is conducted within a 15 x 15 search window on full-pixel basis, i.e., the maximum block displacement in horizontal or vertical direction is +/-7 pels.

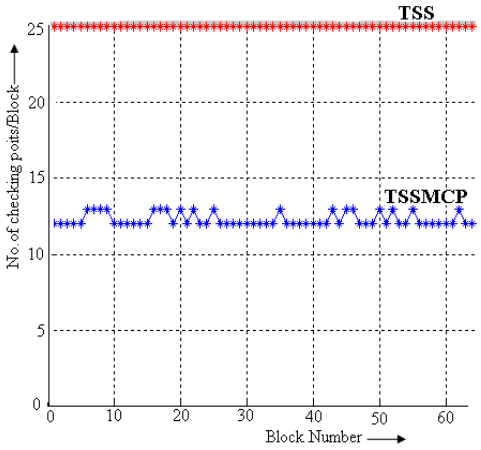

Fig. 6 Simulation result

Mean absolute distance (MAD) is used as the matching criterion to reduce the block-matching computation in practice for reduced computation. Four test image sequences with large, moderate or small motion are exploited in the experiments: the **Akiyo** sequence (QCIF-176x144,Low motion), the **Foreman** sequence (CIF-352x288, Moderate motion),the **BUS** (QCIF-176x144, High motion) sequence and the **Container** (CIF- motion).The performance of an algorithm is a compromise between the PSNR computed and the Average Search Point needed by each algorithm. The following table summarizes the results of simulation of the four sequences for PSNR expressed in decibel (dB) and the Average Search Point. Figures 7(a), 7(b), 7(c), 7(d), shows PSNR and average search points for the sequences Akiyo and Bus respectively. The results obtained with the first

ISSN: 2278 - 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering
Vol. 1, Issue 5, November 2012

100 frames for four video sequences are presented in Table 1. Results show that the mean PSNR values for first 100 frames of the low motion videos such as 'Akiyo', FTSS gives best performance similar to Full Search.

TABLE I
PERFORMANCE COMPARISON BETWEEN FTSS, TSS & FS IN TERMS OF PSNR

Video file	FS	TSS	FTSS
	PSNR	PSNR	PSNR
Akiyo	42.744	40.543	41.644
Bus	23.305	20.603	21.192
Forman	32.873	30.684	31.2720
Container	39.294	36.192	37.193

TABLE II
COMPARISON OF AVERAGE SEARCH POINT FOR FOUR VIDEO SEQUENCES

Video file	FS	TSS	FTSS
	Avg. search Pt.	Avg. search Pt.	Avg. search Pt.
Akiyo	184.555	22.48	13.227
Bus	204.2828	23.589	13.607
Forman	204.2828	24.290	14.014
Container	204.2828	24.224	13.148

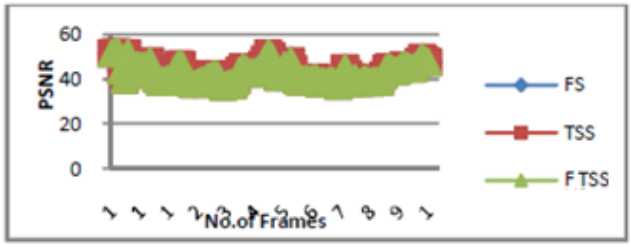

Fig. 7(a) PSNR for Akiyo

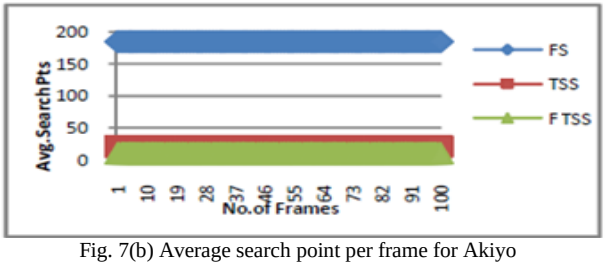

Fig. 7(b) Average search point per frame for Akiyo

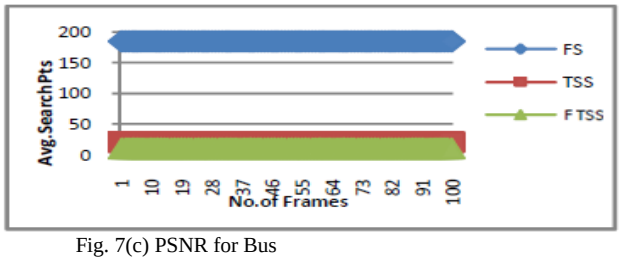

Fig. 7(c) PSNR for Bus

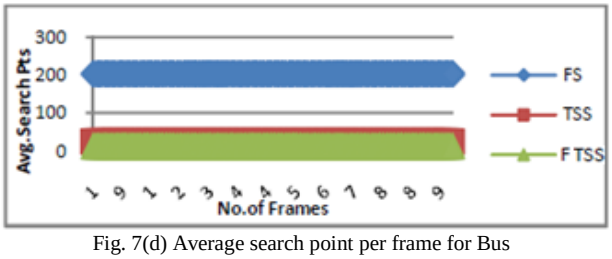

Fig. 7(d) Average search point per frame for Bus

V. CONCLUSION

The performance of fast motion estimation algorithms partly depends on the pattern used in the algorithm. In this paper, we have presented an efficient VLSI architecture for the FTSS algorithm. Configuration of random access on-chip buffer solves the problem of chip I/O and memory bandwidth requirements. The buffer and the input data are arranged by a principle called residual memory interleaving for parallel accessing of data. The proposed architecture was designed with 3 PEs the chip area

has been reduced. A high reduction in computational complexity is achieved by using this proposed technique. Performance of the implemented FTSS is compared against FS and TSS Block Matching Algorithms. Results for Peak Signal to Noise Ratio (PSNR) and average number of search points per frame are obtained through experimental evaluation. This paves the way for reduced power dissipation. It is observed that average number of search points required for proposed algorithm is comparatively much less than other two algorithms. This architecture is considered to be useful for low bit rate, low power video applications like video telephony, video conference and HDTV.

REFERENCES

[1] R. Srinivasan and K. R. Rao "Motion-compensated coder for videoconferencing," *IEEE Trans. Comm.*, Vol. COM-35, pp.297-304, March 1987.
[2] C. H. Hsieh and T. P. Lin," VLSI architecture for block-matching motion estimation algorithm," IEEE Trans. on Circuit and System for Video Tech., Vol. 2, no. 2, pp. 169-175,June 1992.
[3] Jong-Nam Kim and Tae-Sun Choi, *"A Fast Three Step Search Algorithm with Minimum Checking Points,"* Proc. of IEEE conference on Consumer Electronics, pp.132-133, 2-4 June 1998.
[4] Her-Ming Jong, Liang-Gee Chen, and Tzi-Dar Chiueh, "*Parallel Architectures for 3-Step Hierarchical Search Block-Matching Algorithm,"* IEEE Transactions on circuit and systems for video technology, Vol.4.No.4, pp.407-416, August 1994.
[5] Seth.K, Rangarajan.P, Srinivasan.S, Kamakoti.V, Bala Kuteshwar.V, *"A Parallel Architectural Implementation of The New Three-Step Search Algorithm For Block Motion Estimation,"* Proc. of IEEE conference on VLSI Design, 2004, pp.1071-1076.
[6] Renxiang Li, Bing Zeng, and Ming L. Liou, "*A New Three-Step Search Algorithm for Block Motion Estimation"* IEEE Transactions on Circuits and Systems for Video Technology, Vol.4,no.4, pp.438-442,August 1994.
[7] Jianjua Lu, and Ming L. Liou,*"A Simple and Efficient Search Algorithm for Block-Matching Motion Estimation,"* IEEE Transactions on Circuits and Systems for Video Technology, Vol.7, no.2, pp.429-433, April. 1997.
[8] CCITT SGXV, *"Description of reference model 8 (RM8),"* Document 525, Working Party XV/4, specialists Group on Coding for Visual Telephony, Jun. 1989.
[9] MPEG, *"IS0 CD11172-2; Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbits/s,"* Nov. 1991
[10] B. Zeng, et al., "A new three-step search algorithm for block motion estimation", *IEEE Trans. Circuits Syst. Video Technol*. 4 (1994) 438–442.
[11] P. Kuhn, Algorithms, "Complexity Analysis and VLSI Architectures for MPEG-4 Motion Estimation", Kluwer Academic Publisher, Dordrecht, 1999.
[12] H. G. Musmann, P. Pirsch, and H. J. Grallert, "Advances in picture coding," Proc. IEEE, vol. 13, pp. 523-548, A r 1985.
[13] Iain E. G. Richardson, "*Video Codec Design*", John wiley & Sons, Ltd.

Biography

I Namrata Verma currently am working as an Assistant professor in a Department of Electronics & Telecommunication Engineering in BIT Raipur, Chhattisgarh, India. I have 2 years teaching experience. I have completed my M Tech. from IIT Kharagpur (W.B.). My research area is Image and Video Processing, VLSI Architecture Design.

I Tejeshwari Sahu currently am working as an Assistant professor in a Department of Electronics & Telecommunication Engineering in BIT Raipur, Chhattisgarh, India. I have more than 4 years teaching experience. I have completed my M Tech. from BIT Durg(C.G.). My research area is Image and Video Processing, Water Marking.

I Pallavi Sahu currently am working as a Lecturer in a Department of Electronics & Telecommunication Engineering in BIT Raipur, Chhattisgarh, India. I am pursuing my M Tech. from Rungta Bhilai (C.G). My research area is VLSI Design & Wireless sensor and networking Design.