IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)
Volume 5, Issue 1, Ver. II (Jan - Feb. 2015), PP 41-49
e-ISSN: 2319 – 4200, p-ISSN No. : 2319 – 4197
www.iosrjournals.org

High Performance Architecture for Full Search Block matching Algorithm

P.Muralidhar[1], C. B. Ramarao[2]
National Institute of Technology, Warangal, India

Abstract: *Video compression has two major issues to be handled, one is video compression rate and other one is quality. There is always a trade-off between speed and quality. Full search block matching algorithm (FSBMA) is most popular motion estimation algorithm. But high computational complexity is the major challenge of FSBM. This makes FSBM to be very difficult to use for real time video processing with the low power batteries. Other algorithm gives better speed on the expense of quality of video. The proposed algorithm i.e. modified full search block matching algorithm (MFSBMA) reduces the computational complexity by keeping the PSNR same as of FSBMA. MFSBMA skips the SAD calculations for a current background macroblock and it does SAD calculations for foreground current microblock. This method reduces SAD calculations drastically. This work presents apipelined architecture forMFSBMA which can work on real time HDTV video processing. The proposed algorithm reduces computational complexity by 50% keeping PSNR same with the full search algorithm.*

Keywords: *Full Search Block Matching algorithm, Block matching algorithm, Motion Estimation, H.264/AVC, motion vector.*

I. Introduction

There are many video compression standards like MPEG1, MPEG2, MPEG-3, H.261, H. 262 and H.264, etc. H.264/AVC (Advanced Video Coding)[1,2] is also known as MPEG-4 Part 10. H.264/AVC aims at providing good video quality with lower bit rate than previous coding standards. Motion estimation block is computationally intensive block. In an encoder 60% of computations are done by Motion Estimation (ME) block and rest of blocks does 40% of computational work.

Block Matching algorithms (BMA) are used for ME. BMA works on temporal and spatial redundancies present in a video. In block matching algorithm each frame of video is divided into NxN blocks. The motion of each block from current frame to reference frame is represented by Motion Vectors (MV). To calculate these motion vectors most of the algorithms use temporal redundancy present in a video. To work with this temporal redundancies block matching algorithms need block matching criteria. MSE is the Euclidean distance between current and reference MBs. It is considered to be a better BMD because it is closer to our visual perception. The drawback of MSE is that it is more complex than other BMDs as it needs the square operation. Therefore it is less commonly used in hardware implementations due to its complexity.

$$MSE(w_x, w_y) = \frac{1}{N \times N} \sum_{i=x}^{x+N-1} \sum_{j=x}^{y+N-1} \left[F_t(i,j) - F_{t-1}(i+w_x, i+w_y) \right]^2$$

$$(u,v) = \min_{-w \le w_x, w_y \le +w} MSE(w_x, w_y)$$

SAD is almost the same as MAE, except that it omits the constant denominator N x N. It is a popular BDM in hardware implements. To calculate the absolute difference for one pixel, it requires one subtraction and one absolute operation, plus one addition for the accumulation of the difference from the previous pixel. SAD requires the minimum number of computation among all BDMs and it requires the least amount of hardware resources in addition to that it performs the simplest operation that reduces the time in calculating SAD w.r.t to other BDMs.

$$SAD(w_x, w_y) = \sum_{i=x}^{x+N-1} \sum_{j=x}^{y+N-1} \left| F_t(i,j) - F_{t-1}(i+w_x, i+w_y) \right|$$

$$(u,v) = \min_{-w \le w_x, w_y \le +w} SAD(w_x, w_y)$$

In Video compression standards Motion Estimation (ME) block uses block matching algorithms such as full search block matching (FSBM)[3], Three step search (TSS)[4], four step search (FSS)[5],cross diamond

search[7],New diamond search [8]etc. FSBM gives best PSNR among all mentioned algorithms. FSBM is an exhaustive search of NxN current frame block in previous frame search region. Apart from advantage of good quality video compression in FSBM it requires large number of computations. Another block matching algorithm is Three step search (TSS)[4] algorithm . It selects centre of search region and calculates sum of absolute difference (SAD) for locations which are 4 positions apart from centre.

Thus, it calculates SAD at 8 locations in first step. In next step, it repeats the procedure by reducing the step size by 2. In third step, it reduces step size by 1. This reduces number of computations but it is prone to missing small motions. Four step search (FSS) is same as TSS. Instead of 3 steps 4 steps are required in FSS. First stage of FSS requires 5x5 window instead of 9x9 window in TSS[4]. Diamond search method[8] selects centre and calculates SAD at 9 locations in first stage. The 8 points around centre makes shape of diamond. In next stage 5 checkpoints around centre makes diamond shape. At every stage it converges to minimum SAD checkpoint and next step proceeds from SADmin checkpoint. These algorithms suffer from irregular data flow and does not guarantee about the optimum solution. The proposed algorithm separates the background pixels from foreground pixels. It calculates SAD for foreground blocks only. As in video frame minimum50% of area is background area, so number of SAD calculations required reduces drastically. SAD is block matching criteria to find best match for current microblock in search region.

II. Background Elimination Algorithms

Background subtraction is used in many emerging video applications, such as video surveillance, traffic monitoring, and gesture recognition for human-machine interfaces. There are many background elimination methods present. First method is frame differencing[9] which has less complexity. Second is approximate median method which has medium complexity and third has high complexity mixture of Gaussian method[9]. In Frame differencing the current frame is subtracted from the previous frame, and if the difference is greater than a threshold Ts, the pixel is considered part of the foreground.

$$(Frame_i\text{-}Frame_{i-1})j>T_s \qquad (3)$$

This method does have two major advantages. One obvious advantage is the modest computational load another is that the background model is highly adaptive. Since the back-ground is based solely on the previous frame, it can adapt to changes in the background faster than any other method (at 1/fps to be precise). As we'll see later on, the frame difference method subtracts out extraneous background noise (such as waving trees), much better than the more complex approximate median and mixture of Gaussians methods. A challenge with this method is determining the threshold value. This is also a problem for the other methods. The threshold is typically found empirically, which can be tricky.

III. Proposed Algorithm: Modified Fsbm

We designed the proposed motion estimational gorithm to decrease the computation complexity of the full searchalgorithm and also to tracking one object. First we read a frame after a frame from the video. The algorithms works as follow:

1. Read first frame name it as reference frame (RF). 2. Read second frame name it as current frame (CF). 3. Take the difference of CF and RF,it gives background frame (BF). 4. Threshold (T) each background pixel value i.e. if BF pixel value is less than threshold then replace CF pixel to zero. It means that pixel is background pixel. 5. If BF pixel is greater than T, it means it is foreground pixel. 6. Calculate motion vectors for foreground pixels. 7. Repeat same procedure for all frames.

In a video, frame to frame pixel movement is very less. Background area is more than foreground area. This reduces the number of computations required for full search block matching algorithms significantly.

IV. Proposed Architecture

The architecture design supports functionality of Modified full search block matching algorithm. Background elimination logic improves the speed of full search block matching algorithm. This also supports variable block size motion estimation(VBSME) algorithm[10]. The block diagram of top module of MFSBM is shown in figure 1. The data path contains RAMs for current frame data and for reference frame data. Background elimination logic and motion vector calculation block. The required signals for data path come from controller like addresses of current microblock data (CMD) memory and search range data (SRD) memory, selection signal for SRD and CMD data, clear signals for SAD adders. Data path revert back status signal to controller. Status signal tells about the current microblock, whether it is background block or foreground block. This status signal decides whether to calculate SADs for that current microblock over search range or not. Finally, data path gives 41 motion vectors for a current microblock.

A. Data path

Data path shown in figure 2 consists of RAMs, Registers, Background elimination logic, SAD calculation logic, Adder tree and Motionvector calculator. There are two RAMs SRD strips and CMD strips(Fig.3&4). CMD strips stores current microblock data of size 16x16.

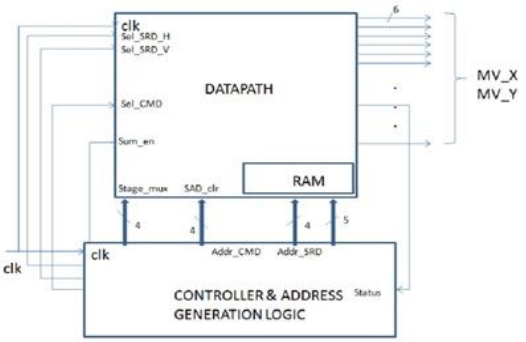

Fig. 1. Architecture of Modified Full search method.

CMD strips have 16 strips of each 16 pixel wide. SRDstrips stores search range data of size 32x32. Search range data is stored in 32 RAM strips, each of size 32bits wide. The addresses required to access the data from both the RAM strips are given by controller.Addr SRD is of size 5 bits wide and Addr CMD is of size 4 bits wide. Data path reads 64 pixels at a time from CMD and 19 pixels from SRD RAM. Register clusters[Fig.8] are used to hold current microblock 16 pixels and 16+12 pixels from search region such 16 clusters are present in data path.Addr SRD gives address for every SRD strip. We need data from selective strips depending upon the current stage. For given address every stripgives one pixel from SRD. Multiplexers are used to select which strips data should be used for SAD calculation at present stage. Stage mux selection lines of size 4 bits are given by controller. Background elimination logic blocks need to calculate block sum of both microblocks CMD and SRD. Then difference of these blocksumsis calculated. Background elimination logic thresholds this difference value and gives status signal to controller. SAD calculation logic works in pipelined manner. The basic building block to calculate SAD is processing element(PE) [Fig.6].

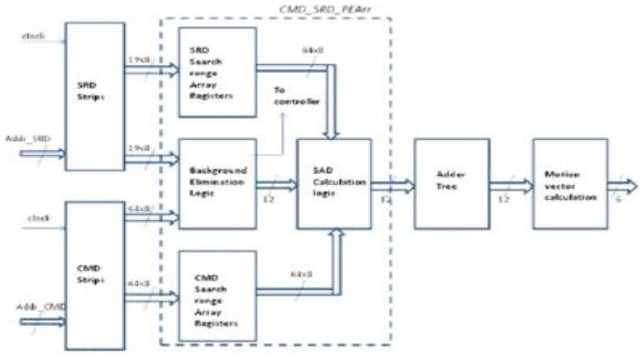

Fig. 2. Data path

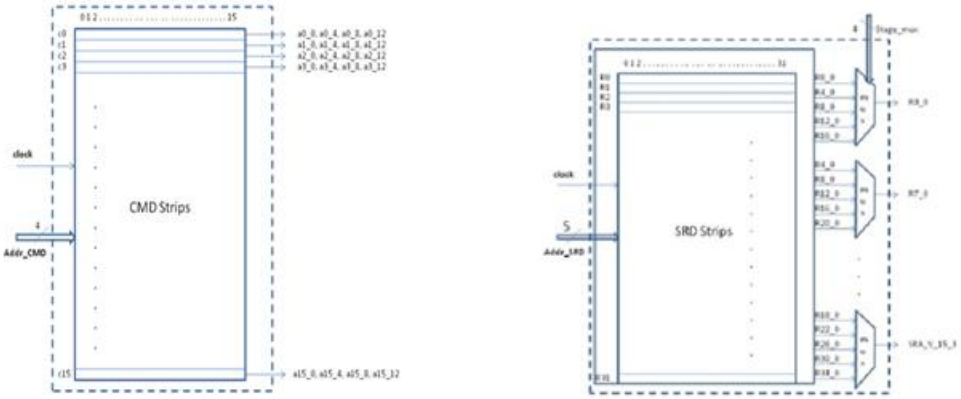

Fig. 3. Current Memory Strips **Fig. 4.** Reference Memory Strips

PE registers one pixel from SRD (R) and CMD (C) and SAD in from previous PE. Absolute difference of R and C is calculated and added to SAD in. This gives 12 bit SAD out. A PE plays a vital role in processing three different tasks in this design. PE computes the absolute difference between the CMD and the SRD. It Propagates the CMD and SRD values to the next PE (PCMD and PSRD). Finally the PE sums up the difference values (DV) of the current and the previous data PEA is array of 16 PEs(Fig.7), which are processingparallel.

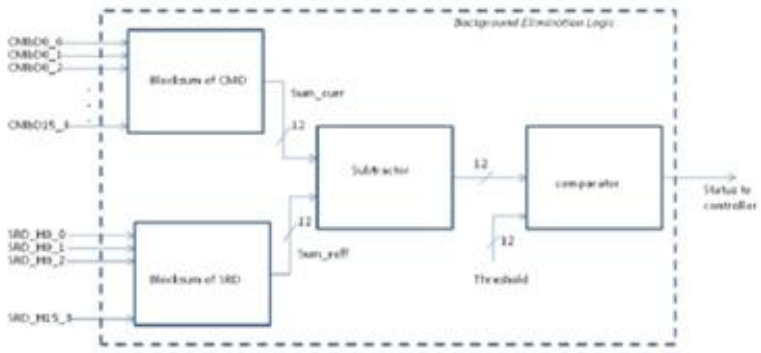

Fig. 5. Background Elimination Block

Four PEs (PE0, PE4, PE8, PE12) give one 4x4 SAD calculation. All 16 PEs give four 4x4 SADs. Current RAMgives four pixels at a time.

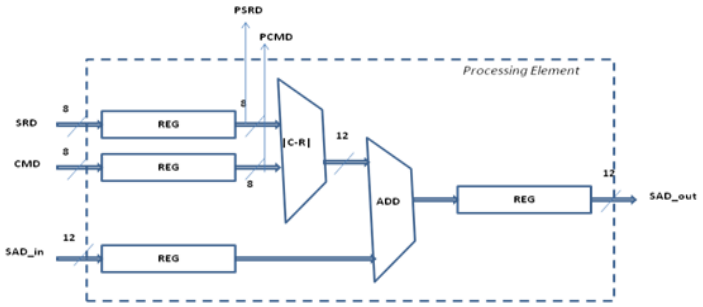

Fig 6 Processing Element

Input current RAM data isconstant for four clock cycles. Input Reference data is constant for 3clock cycles. Current data flowshorizontally through PEA,PE0, PE4, PE8, PE12 forward current data to PE1, PE5, PE9,PE13 respectively and so on. Reference dataflows vertically i.e., from PE1 to PE4, PE2 to PE5, PE3 to PE6 and so on. PE7, PE11, PE15 require extra 3 reference pixels. It achieves 100% PE utilization by employing a preload register and a search data buffer. Such 16 PEAS should be connected in pipelined manner (Fig.9). The vertical data of one PEA comes from horizontal data of above PEA. Top most PEAs get vertical data from reference RAM. 16 PEAs work on 4 microblock SAD calculation in pipelined manner.

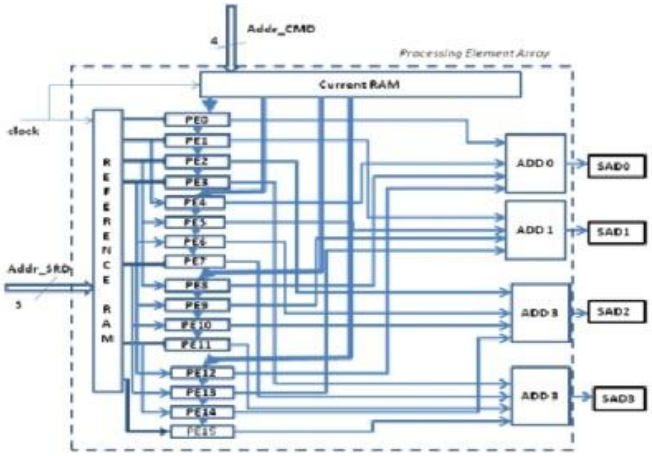

Fig.7a PEA

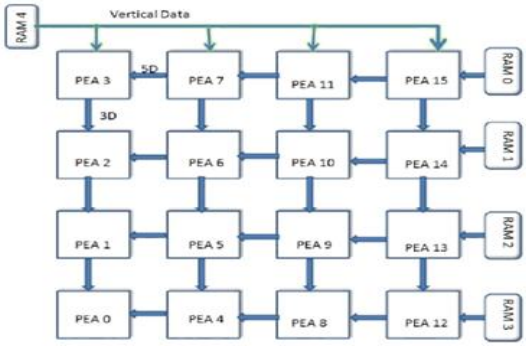

Fig. 7. Processing Element Array

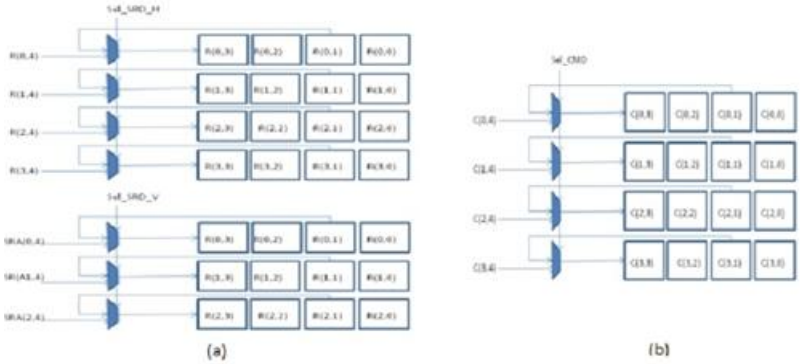

Fig. 8. SRD and CMD shift registers

B. Controller

Controller[fig.10&11] gives control signals as well as generates addresses for block RAM memory. CMD Addr of size 4 bits is address to current microblock RAM strips. SRD Addr of size 5 bits is address to reference microblock RAM strips. Stage mux signals are used to select which 16x16 microblockis selected as reference microblock for SAD calculation.Background elimination block(Fig. 5) calculates block sum of current microblock as well as reference microblock. Sum-en signal is used as enable signal for block sum calculation. In the architecture design, each PEA needs reference data that flows in two ways, one is horizontal and other is vertical. Reuse of data is major advantage of this architecture so, it needs registers. One Processing Element Array (PEA) needs 16 horizontal pixels and 9 vertical pixels. It reads 4 pixels from horizontal data and 3 pixels from vertical data at a time. So, it uses 16 + 9 registers to store total reference data for a PEA. Sel-SRD H is given to multiplexer which decides whether to reuse the horizontal reference data or to read new data into registers. Sel-SRD V is given to multiplexers which decide whether to reuse vertical reference data or to read new data into registers.

A PEA reads 16 pixels form current block RAM 4 pixels at a time. PEA reuses current microblock data so it also needs 16 registers. Sel-CMD signal is given to multiplexer which decides whether to reuse current microblock data or to read new data into registers. A PEA uses four adders/accumulators for SAD calculation. SAD-clr signal are given to these accumulators. After block sum calculation data path gives status signal to controller. This status signal indicates whether this block is foreground current microblock or background microblock. Initially all SRD (search range data) and CMD (current microblock data) registers are empty.

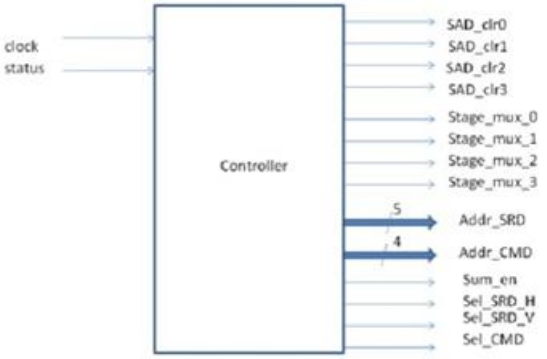

Fig. 10. Controller

All SAD-clr signals are enabled. Address of search range data of size 5 bit (Addr SRD) as well as Address of current microblock data of size 4 bit (Addr CMD) are initialized to ”00000” and ”0000” respectively. When process starts data from SRD RAM strips is taken into SRD registers. At a time 16+3 pixels are read from SRD RAM strips and stored into four 4 x 4 horizontal SRD registers and one 3 x 3 vertical SRD registers. These registered data is given to processing element array (PEA) 3, 7, 11, 15 and these blocks forward the horizontal and vertical data to other PEA blocks. To read one 4 x 4 search range data four clock cycles are required. And 3 clock cycles to propagate the SRD data to other PEAs. So, initially total 28 clock cycles are required to provide total 16 x 16 microblock data to architecture. Later after every four clock cycles SRD registers are updated by new SRD data (16+3). Otherwise SRD registers use previous data stored in registers. After 23 clock cycles CMD registers start reading data from CMD RAM strips. It can read 64 pixels at a time from current microblock data (CMD) RAM strips. So, only 4 clock cycles are required to read whole 16 x 16 current microblock. Simultaneously background elimination block calculates block sums for both SRD and CMD and on 28 cycle it tells that whether current microblock is foreground microblock or background microblock. While calculating block sum it continues to calculate SAD for every 4 x 4 block. After 28 clock cycle controller decides whether to proceed with SAD calculation or to stop depending on foreground or background current microblock. If current microblock is background microblock then every signals are initialized to origin. And if current microblock is foreground block then architecture continues to calculate SADs. Every PEA requires 8 clock cycles to provide first SAD later for every clock it gives SAD for 4 x 4 micro blocks. 72 clock cycles are required to produce four (16 X 16) SAD calculation. Then all 41 variable block size SADs are calculated and comparison is done for all 41 SADmin. Motion vectors are calculated for all possible 41 micro blocks. Motion vectors are of size 6 bits. This process repeats for whole search range 32 X 32.The pipelined architecture made possible to achieve full HD requirements.

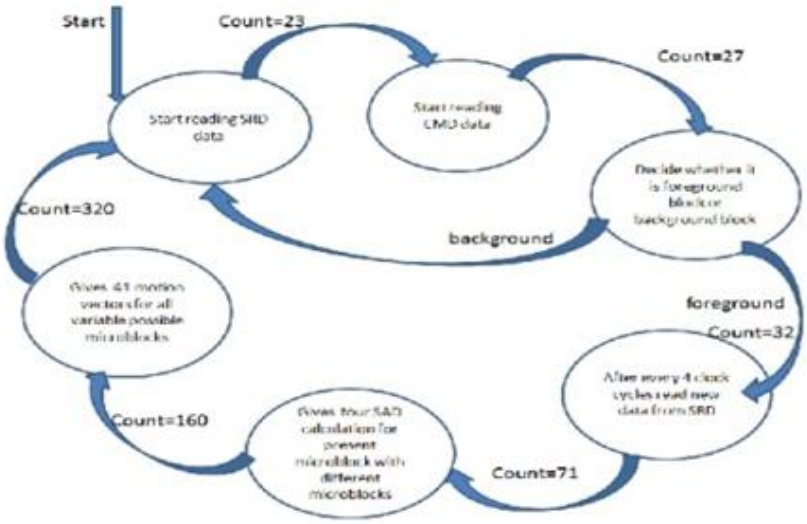

Fig. 11. State diagram of controller

V. Results & Discussion

The proposed algorithm is simulated in MATLAB and Xilinx ISE 13.5. The experiments were conducted on five different video clips. The video clips used are Coastguard (qcif) 176x 144 300 frames, akiyo (qcif), foreman (qcif), mobile (qcif), Highway (cif) 352x288. The quality of recovered video clips was verified by calculating PSNR. Computational complexity (CC) was also calculated for every video.

$PSNR(I_t,I_{t+1}) = 10 \log(255^2 / MSE)$

$$MSE((I_t,I_{t+1}))=(1/MN)\sum_{m=1}^{M} \sum_{n=1}^{N} I_t(m,n) - I_{t-1}(m,n)$$

The Fast full search block matching algorithm (FFSBMA) results are compared with Full search block matching algorithm (FSBMA) results. Comparison shows that computational complexity of FFSBMA is 50% less than FSBM, while PSNR is almost same.

VI. MATLAB SIMULATION RESULTS

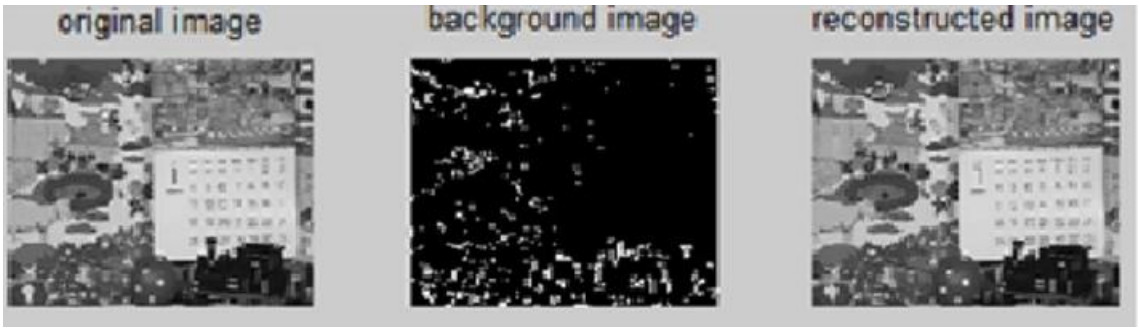

Fig. 12. Mobile MFSBMA Matlab simulation

Fig. 13. Highway MFSBMA Matlab simulation.

Fig. 14. Foreman MFSBMA matlab simulation

For mobile fig(12) video input qcif (172 x 144) threshold was 10. For Foreman fig(14) video input qcif of size (172 x 144) threshold was 25. For third input video highway cif of fig(13) of size (352 x 288) threshold was 25 computations. Still the quality of MFSBMA video is almost same as FSMBMA video quality.Table2& Table3 gives the compassion of full search and Modified full search block matching algorithm(MFSBA) PSNR and number of computations.

Three input video are simulated using MATLAB. Brightness of one of the frame of each video is changed by adding DC value to every pixel of that frame. First is foreman fig (14) video input qcif (172 x 144) second is mobile qcif fig (12) of size (172 x 144) and last one is highway CIF fig(13) of size (352 x 288).

VI. Synthesis Report

The proposed architecture is implemented on Xilinx Vertex 6FPGA XC6vlx240T FF1156. The resource utilization is shown below in table 1. The output of hardware is seen on chipscope pro of Xilinx. The functional output of controller and final motion vectors timing diagrams are shown in Fig.15& Fig.16.

Table1 Resource Utilization

Slice Logic Utilization	Used	Available	Utilization
Number of Slice Registers	12,598	301,440	4%
Number used as Flip Flops	12,521		
Number used as Latches	1		
Number used as Latch-thrus	0		

Table 2psnr Comparison Of Fs And Mfsba Method

Method	PSNR(dB) Video Input				
	Coastguard	Akiyo	Foreman	Mobile	Highway
FS	30.24	38.04	30.5	24.82	33.82
MFSBA	29.22	40.63	29.8	26.81	30.7

Table 3: Computational Complexity Comparison Of Fs And Mfsba Method.

Method	Computational Complexity Video Input				
	Coastguard	Akiyo	Foreman	Mobile	Highway
FS	184.5	184.5	184.5	184.5	184.55
MFSBA	66.58	19.36	73.07	115.38	14.54

Timing Report:
Constraints cover 2477 paths, 0 nets, and 755 connections Design statistics:
Minimum period: 4.124ns1 (Maximum frequency: 211.483MHz)
Maximum path delay from/to any node: 2.054ns

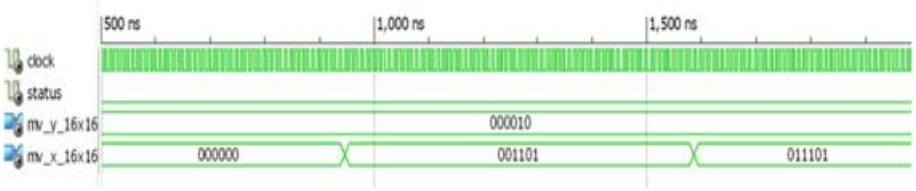

Fig.15 Motion vector timing diagram

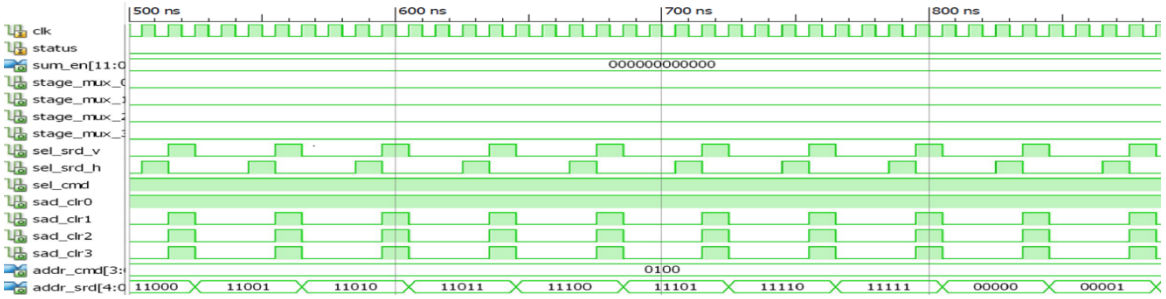

Fig.16 Controller timing diagram

Table 4: Comparasion of existing BMAs and proposed MFSBA

Design	1D Systolic	1D Tree	Compact systolic	SAD reuse architecture	MFSBA (proposed)
Board used	Virtex 2 pro	Virtex 2 pro	Virtex E	Virtex 2	Virtex 6
Search Range	[-16,+15]	[-16,+15]	[-16,+15]	[-4,+4]	[-32,+31]
Cycles/MV	16401	16393	1983	2086	158
No of slices	836	350	14728	7253	13,188
Max. Frequency	239 MHz	240 MHz	55 MHz	68.02MHz	211 MHz
Power	1.344 W	1.16 W	2 W	-	0.408 W
SD frame rate	12fps	12.2fps	23.11fps	27.17fps	164fps

VII. Conclusion

Modified full search block matching algorithm (FFSBMA) reduced the computation complexity significantly compare to full search block matching algorithmas shown in Table 4. Number of SAD calculations has reduced 50 % by applying background elimination method. FFSBMA had achieved the goal by keeping the quality of video same. The proposed architecture can work on HD video and Full HD video. The pipelined

architecture helped in achieving frequency of 211 MHz and SD frame rate of 164 fps.

References

[1]. International Telecommunication Union Telecommunication (ITU-T). Draft text of draft international standard for advance video coding. Recommendation H.264 (draft), 2003.

[2]. ITU-T Recommendation H.264 & ISO/IEC 14496-10 (MPEG-4) AVC. Advance video coding for generic audiovisual services. (Version 1: 2003, version 2: 2004, version 3: 2005).

[3]. Seung-Man Pyen, Kyeong-Yuk Min, Jong-Wha Chong," An Efficient Hardware Architecture for Full-Search Variable Block Size Motion Estimation in H.264/AVC," International Symposium on Visual Computing-2006 , pp. 554-563.

[4]. R. Li, B. Zeng, and M. L. Liou, "A new three-step search algorithm for block motion estimation," IEEE Trans. Circuits Syst. VideoTechnol., vol. 4, pp. 438442, Aug.1994.

[5]. L. M. Po and W. C. Ma, "A novel four-step search algorithm for fast block motion estimation,"IEEE Trans. Circuits Syst. Video Technol., vol. 6, pp. 313317, June 1996.

[6]. M. F. So and A. Wu, "Four-step genetic search for block motion estimation," in Proc. IEEE Int. Conf. Acoust. Speech Signal Process.(ICASSP '98), vol. 3. May 1998, pp. 1393–1396.

[7]. M. Ghanbari, "The cross-search algorithm for motion estimation," IEEE Trans.Commun., vol. 38, pp. 950-953, July 1990.

[8]. S. Zhu and K.-K.Ma, "A new diamond search algorithm for fast block-matching motion estimation," in Proc. Int. Conf. Inf. Commun. Signal Process. (ICICS '97), vol. 1, Sep. 9–12, 1997, pp.292–296.

[9]. Seth Benton, Background subtraction, part1: MATLA Bmodels, http://www.eetimes.com/design/militaryerospace design/4017685/Background subtraction-part-1- MATLAB-models.

[10]. S. Y. Yap, J. V. McCanny, "A VLSI Architecture for Variable Block Size Video Motion Estimation," IEEE Transactions on Circuits and Systems-II: Express Briefs. 2004, 51(7): 384~389.

[11]. Kroupis, N.; Dasygenis, M.; Markou, K.; Soudris, D.; Thanailakis, A., "A modified spiral search motion estimation algorithm and its embedded system implementation," Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on , vol., no., pp.3347,3350 Vol. 4, 23-26 May 2005.