A High Throughput Diamond Search Architecture with Shift Mechanism for 720p Motion Estimation

Gaurav Srivastava
Dept. of ECE, NIT Warangal
Warangal, India
srivastavag89@gmail.com

Muralidhar P.
Dept. of ECE, NIT Warangal
Warangal, India
pmurali@nitw.ac.in

Dr.C.B.Rama Rao
Dept. of ECE, NIT Warangal
Warangal, India
cbrr@nitw.ac.in

***Abstract*— The paper implements a shift based diamond search architecture, for HDTV videos. High resolution frames like 720p need large memory to store the pixel data. This large pixel data is stored on an external memory and an internal local memory is updated with the pixel data on the reference frame and current frame required for particular diamond search iterations. This is done as internal memory is expensive. However, accessing pixel data from external memory incurs high timing latency. This paper implements a shift based DS architecture to shift data in internal local memory and update from external memory only the pixels which are new to search area. This reduces the redundant operation of accessing same pixel data for successive searches.**

Even when considering equal latency for internal and external memory access, when simulation was performed on the same sample 720p frame, architectures with no shift block took 173.170ms vs with shift block took 90.564ms. When considering the latency of external memory access this difference in timing becomes largely significant.

***Keywords*— Motion estimation, macro block, video compression, fast algorithm, 720p, Sum of Absolute Differences (SAD)**

I. INTRODUCTION

Block matching algorithms are used for motion estimation (ME) in video processing. Implementing these algorithms in hardware requires hardware resources to make these operations parallel. Minimum use of these resources and yet keeping the motion estimation fast is a challenge and trade off. Diamond Search (DS) algorithm [3] is a motion estimation algorithm close in accuracy to Full Search (FS) algorithm. DS reduces the number of Sum of Absolute Differences (SAD) operations performed per macro block, hence requiring less hardware resources and making computation fast, while keeping the accuracy close to Full Search.

In 2008, Porto M and et. el. [5] implemented a high throughput and low cost DS architecture. The architecture has three types of internal memories on the chip. The Local Memory (LM) which stores the search area on reference frame needed for nine candidate block for the Large Diamond Search Pattern (LDSP), Small Diamond Search Pattern (SDSP) and all the possible blocks for the next step on DS. There are thirteen Candidates Block Memory (CBM) for storing reference frame. There is one Current Block (CB) memory to store the macro block from the current frame. The blocks stored are all 2:1 sub sampled blocks 8*16. For the calculation of SAD, Processing Units (PUs) were used which can process 8 samples at a time, hence one line of the sub sampled block will be processed in parallel. Comparator is used to compare the candidates and select the candidate with minimum SAD and pass on the information regarding vector with minimum SAD and minimum SAD value to Control Unit (CU).

This paper has introduced a shift operation block which performs shift operation within LM. This shift operation is controlled by CU. The synthesis results in this paper are for 720p HDTV frame. Also CBM has a provision to read and write at the same time and hence the SAD calculations start even when CBM is not fully filled and the pixels needed for SAD computation for the particular step on DS have been added. This helps reduce the number of read access needed from external memory.

II. PROPOSED ARCHITECTURE

External to the chip a ROM is assumed to have stored current frame data (for which the vectors are to be computed) and reference frame data (to which the current frame will be compared). The interface of these memory blocks to the internal memory is through demultiplexer DMUX. DMUX gets 8 bit pixel data as input form external ROM. The pixel data was assumed to be 8 bit monochrome for the purpose of analysis and synthesis. However the architecture can be extended for pixels with color data. DMUX passes the pixel data to CB or LM depending on whether the pixel is from current frame and reference frame respectively. The architecture deals with single macro block in a frame at a time. The architecture considers a macro block of 16*16 pixels. Architecture computes Minimum SAD and the Vector with minimum SAD for each macro block. Search area is considered to be 100*100 thereby covering large movements within the frame. The block from current frame for which the vector is computed is stored in Current Block (CB) memory. The architecture performs 2:1 sub sampled SAD computation hence only 8*16 pixels are stored in CB, storing only alternate pixels in a row. CB is a RAM of 128 words with word width of 8 bits, each storing a pixel data of 8 bits. Search area with all the pixels needed for the current 9 candidates of Large Diamond Search Pattern (LDSP) and 4 of Small Diamond

978-1-4799-6052-1/14/$31.00 ©2014 IEEE

Search Pattern (SDSP) are stored in LM. Search area consists of 20*20 pixels. LM is RAM of 400 words with a word width of 8 bits, each storing a pixel. LM is connected with 13 Pattern Generators (PG0-PG12), each for the 9 LDSP candidates and 4 SDSP candidates. Refer Fig.1 for the numbering of the candidate blocks on the DS pattern. PG selects the pixels in a row on LM which are needed for the respective candidate on diamond search. Input to PG is a row of 20 pixels form LM i.e. 160 bits and output is 8 pixel pattern needed for particular candidate block.

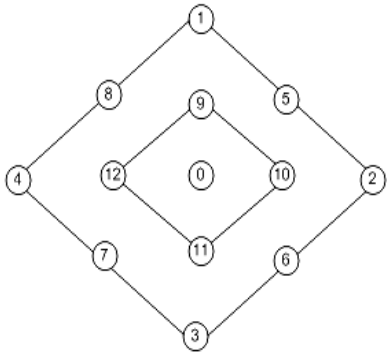

Fig. 1. LDSP and SDSP candidate points on the diamond

The output of PG is stored in corresponding Candidate Block Memories (CBM0-CBM12), each for the 13 candidate blocks. CBMs are RAM of 128 words, each word being 8 bits. They store sub sampled 8*16 pixels for a particular candidate block. The CBMs0-8 store the LDSP candidates and CBMs9-12 store SDSP candidates. The row of 8 pixels from CBMs i.e. 64 bits is output of CBMs. 9 Processing Units (PU) compute SAD for respective candidate blocks. Each PU accumulates SAD for a row on a single clock pulse. MUX X1-4 are used to multiplex the input to PUs 1-4 with one input coming from candidate 1-4 from LDSP and the other coming from candidates 9-12 from SDSP (for numbering of candidate blocks refer Fig 1). 16 bit SAD output from 9 PUs is input to comparator through flops. Comparator compares all the 9 SADs and selects the minimum SAD and vector candidate with minimum SAD and passes this information to the controller. The shift block performs the shifting within LM based on the best candidate selected by the comparator. The shift block is embedded within CU. (Refer Fig. 2 for the architecture diagram)

A. Computation flow within the architecture

Step 1: The macro block from the current frame is copied from external memory to CB from external ROM. 128 clocks are needed to full copy 8*16 pixels to CB.

Step 2: 20*20 pixels covering all the search area pixels for the candidate blocks are copied to the LM from external ROM in 400 clocks.

Step 3: 20 pixels in a row or 400 bits output of the LM is fed to PGs. The PG selects the 8 pixels present in the respective candidate block on the diamond and passes them to CBMs.

.

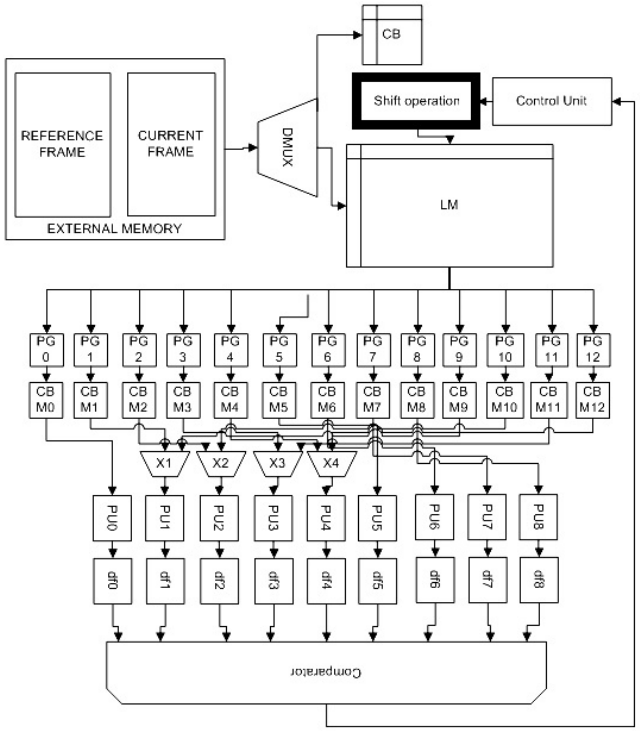

Fig.2. Sub-sampled diamond search architecture with shift operation block

Step 4: CBMs store the output of PGs on the next clock as soon as PG receives the data.

Step 5: 8 pixel output of CBM is fed to PU, for a particular row. The output operation is performed even when CBM is not fully filled, as while CBM outputs the row at one end it receives another row data as input from PG. Hence a 2 port RAM is to be used for CBM to read and write at same time (as read and write locations are different) and hence saving clocks for operation.

Step 6: PU takes 16 clocks for all the rows of a candidate to be fully read from CBM. The SAD is calculated by PU and accumulated to give the final 16 bit SAD, which is fed to comparator through D Flip flops.

Step 7: Comparator computes the minimum SAD among all the 9 candidate blocks and passes the index of the best candidate to the controller and its SAD value is bought to top level

Step 8: Depending if best candidate is from 0-12 on the diamond pattern, the shift block shifts the contents in LM and then new pixels are added for next step of search

Candidate 0: No shifting, muxes X1-4 select input from CBMs 9-12 instead of CBMs 1-4 now and inputs to comparator from df1, df5-8 are invalidated. Now next diamond pattern is SDSP.
Candidate 1: shift down, 2 rows and add top 2 LM rows with new pixels from external memory.
Candidate 2: Shift left, 2 columns and add 2 columns at right end of LM from external memory.
Candidate 3: Shift up, 2 rows and add last 2 LM rows with new pixels from external memory.
Candidate 4: Shift right, 2 columns and add 2 columns at left end of LM from external memory.
Candidate 5: Down shift, 1 row and shift left, 1 column and add 1 top row and 1 column at right of LM from external memory.
Candidate 6: Shift up, 1 row and shifts left, 1 column and add 1 row at row at bottom and 1 column at right of LM.
Candidate 7: Shift up, 1 row and shifts left, 1 column and add 1 row at bottom and 1 column at left of LM.
Candidate 8: Shift down, 1 row and shift right, 1 column and add 1 row at top and 1 column at left of LM.
Candidate 9-12: These are selected at the last stage of DS and hence they are the final vectors.

Step 9: If the number of LDSP searches reach 20, next iteration is set as the last one and the pattern is set to SDSP even when best candidate is not at 0^{th} position. This is done to limit the number of search steps to 21. As after 20 steps vector will move to the edge of search area. Also this will prevent the vector to limit in circular periphery preventing to move towards the vertex of the search area where the probability of finding the vector is less.

III. Statistical Analysis

Following is a statistical comparison of DS architecture with shift mechanism vs DS architecture without shift mechanism (DS without shift mechanism is similar to [5]). Consider n iterations before best candidate is found at 0^{th} position.

Diamond search architecture without shift mechanism:

No. of clocks needed = 558 + (n-1)432 =126+432n (1)

Here,
558 clocks are needed for first LDSP which includes clocks to read the current macro block, read the first search area and for computation on first step of DS. Then 432 clocks are needed on each step of diamond search for reading full search area and computation of SAD.

Diamond search architecture with shift mechanism:

No of clocks needed = 558 + (n-1)(40+30) =488 + 70n (2)

Here,
558 clocks are needed similar to above case. Then on each DS step the number of clock for external memory access will be 40 as for any candidate 2 rows and column combined will be new in the search area. Other 30 clocks are needed for SAD computation.

Hence for n steps,
No of clocks saved with shift mechanism per macro block =362*(n-1) (3)

Of these 360n are saved on access to external memory.
If n=21 for max number of steps on diamond search,Number of clocks saved per macro block = 362*(21-1) =7240 clocks

Considering 720p frame if each macro block has 21 steps on DS, number of clocks saved = 7240*80*45 =26,064,000 clocks

IV. Synthesis & Simulation Results

The proposed architecture was described in VHDL. Industry standard synthesis tool Synopsis Synplify tool was used to synthesize the architecture. Industry standard simulation tool ModelSim 10.1 tool was used to simulate and validate the architecture design. Comparison is made between architecture with shift block and without shift block (DS without shift mechanism is similar to [5]). Due to addition of shift mechanism the utilization of logic was increased. Table I gives a comparison of the resources utilized in both the architectures. There is 116% increase in combinational logic on FPGA needed for shift operation block. MACC (dedicated Math block in SmartFusion2 for arithmetical operation) utilization increased from 2 to 7 due to increase in calculations needed for shifting.

TABLE I. Synthesis Results

FPGA resources	*Architecture without shift mechanism**	*Architecture with shift mechanism*	*% increase with shift block*
Combinational logic	21848	47215	116
Sequential logic	10842	11055	1.96
IO	126	126	0
Global	4	5	25
RGB	4	5	25
RAM 64K*18	60	60	0
RAM 1K*18	0	0	0
MACC	2	7	250

*this architecture is similar to the DS architecture by Porto M and et. el. [5]

The addition of shift block also limits the timing constraints for the architecture. Post synthesis timing results on the architectures report maximum frequency for DS architecture with shift block is 33.27MHz, whereas the maximum frequency for DS without shift block is 52.34MHz.

Simulations were performed on HDTV 720p frames for the calculation of vectors and total number of clocks needed

for vector computation was recorded. The calculations were performed when reference frames were at a distance of -1,-2,-3,-4 and -5 from the candidate frame. Table II details the comparison of total number of clocks needed for vector computation of all the macro blocks on 720p frames on architectures with and without shift logic. The comparison were made keeping reference frame same and changing the candidate frame as the next one in the video.

TABLE II. COMPARISON OF NO. OF CLOCKS NEEDED PER FRAME

Distance of reference frame with respect to candidate frame	*Architecture without shift**	*Architecture with shift*	*% decrease with shift*
-1	8,580,000	3,013,406	184.72
-2	34,386,077	11,048,544	211.22
-3	12,828,384	3,744,591	242.58
-4	14,062,176	3,957,166	255.35
-5	14,880,384	4,097,872	263.12

*this architecture is similar to the DS architecture by Porto M and et. el. [5]

TABLE III. SIMULATION RESULTS ON TIME TAKEN

Distance of reference frame from candidate frame	*Architecture without shift (ns)*#*	*Architecture with shift (ns)***	*% decrease with shift*
-1	173,170,140	90,564,903	91.21
-2	673,911,033	332,052,941	102.95
-3	258,915,274	112,539,937	130.06
-4	283,816,898	118,928,666	138.64
-5	300,330,700	123,157,445	143.85

this architecture is similar to the DS architecture by Porto M and et. el. [5]

*Max frequency =52.34MHz

**Max frequency = 33.27MHz.

When the clocks used were taken as the maximum allowed frequency by the synthesized netlist as 33.27MHz (30.054 ns) with shift and 52.34MHz (20.183 ns) without shift, the comparison of the total time taken for computation of vectors for the complete frame is given in Table III. There is 90% - 150% increase in time needed for computation of frame vectors. This data has considered read access from external memory to have same latency as the operations internal to the chip.

V. CONCLUSIONS

This paper presents hardware architecture for sub-sampled Diamond Search algorithm with shift mechanism for HDTV (720p frames). The synthesis results show that including a shift mechanism increases hardware utilization and limits maximum clock frequency. However the number of clock pulses saved by using shift operation on LM ranged from 180%-260%. Summing up the effect of limiting in max clock frequency and the number of clocks saved the total time saved per frame for vector calculations was 90%-150%. Hence it is evident that using a shift block definitely makes the architecture faster.

If the latency of access of external memory is considered this will increase many times as in section III it is proved that out of total 360(n-1) clocks saved, 360n are for external memory access.

The above architecture acts on a single macro block at a time and hence based on requirement to speed up the computation; the whole architecture can be duplicated multiple times. Hence, make the vector computation of different macro blocks parallel.

REFERENCES

[1] E. Iain , G. Richardson, Video Codec Design, West Sussex: John Wiley & Sons Ltd., 2002, Ch. 1-6.

[2] Coding of moving pictures and audio, ISO/IEC JTC1/SC29/WG11 N2932, Oct. 1999.

[3] Shan Zhu, and Kai-Kuang Ma, "A New Diamond Search Algorithm for Fast Block-Matching Motion Estimation," IEEE Trans. Image Processing, February2000, vol 9, no. 2, pp. 287-290.

[4] Y. Cheng, L. Yang, Z. Fang, H. Hou and G. Chen, "A Fast Motion Estimation Algorithm Based on Diamond and Hexagon Search Patterns," IEEE Joint Conference on Pervasive Computation, Tamsui, Taipei, 2009, pp. 595-598.

[5] M. Porto, L. Agostini, S. Bampi and A. Susin, "A high throughput and low cost diamond search architecture for HDTV motion estimation," IEEE Int. Conference on Multimedia and Expo, Hannover, 2008, pp.1033-1036.