International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE)
Volume 4, Issue 4, April 2015

Full Search Block Matching Algorithm Motion Estimation on FPGA

Harsha Prakash Redkar, Sonia Kuwelkar

***Abstract*— Motion Estimation is used to reduce inter frame temporal redundancy and achieve video compression in recent video compression standards. Various algorithms and hardware architectures are introduced by many authors to achieve motion estimation. But Full search block matching motion estimation method gives accurate results. For block matching sum of absolute difference algorithm is suitable as it requires less number of computations. In this paper we have analyzed two methods to implement sum of absolute difference method viz.bit series method and bit parallel method. Based on analysis we have selected best method to achieve high throughput. Implementation shows that bit parallel method gives faster results of sum of absolute difference. Thus further motion estimation algorithm implementation is carried out with bit parallel method. Motion estimation design is implemented on FPGA. We performed simulations to validate the functionality of design using Modelsim simulator and synthesized using Xilinx ISE10.1 software targeting Virtex II XC2V6000 FPGA family.**

Index Terms— FPGA,Motion estimation, Sum of absolute differences .

I. INTRODUCTION

Motion estimation has been introduced in an attempt to capture the motion of objects within a video scene. i.e. find the best match between the pel(s) in the current frame and the pel(s) in the reference frame. To this end, a search area within the reference frame must be traversed in order to find the best match. After finding the best match, the difference(s) between the pels must be coded together with the difference between the locations (motion vector). Motion estimation can be performed for single pels in the current frame, but it is rarely used, because the coding of motion vectors for single pels reverses the gains of predictive coding. Therefore, block-based motion estimation is the most commonly used form in which a search is performed in the reference frame for a block of pels in the current frame.

The full-search block matching algorithm (FSBMA) is usually used in the hardware implementation of motion estimation (ME), because of its simplicity, regularity, and optimum result. There are three different metric to determine best matching block namely Mean of Squared Error (MSE),Sum of Absolute Differences (SAD) and Matching Pel Count (MPC). For each of these criteria, square block of size N × N pixels is considered. The intensity value of the pixel at coordinate (n1, n2) in the frame k is given by, S (n1, n2, k) where (0 ≤ n1, n2≤ N−1). The frame k is referred to as the candidate frame, k-l is reference frame and the block of pixels defined above is the candidates block.

The most commonly used metric to determine the best match for FSBMA in hardware is the (SAD) computing the minimum SAD from among all the candidate blocks.SAD requires less number of multiplication and comparison operation. Search iteration is performed for each candidate block. The SAD adds up the absolute differences between corresponding elements in the candidate and reference block and can be calculated by (1) and motion vector is given by (2).

$$SAD(i,j)=\sum_{n1=0}^{N-1}\sum_{n2=0}^{N-1}[s(n1,n2,k)-s(n1+i,n2+j,k-l)] \quad (1)$$

$$[d_1,d_2]=\arg\min[SAD(i,j)] \quad (2)$$

Field programmable logic devices support a high number of processor elements (PE) in parallel mode. This property can be used to process, at the same time, all SAD operations from a macroblock, in a search area. With this, real time (RT) video encoder for ME can be reached. The high performance of current FPGA technology permits to implement new designs to solve the ME problem.

Bascally there are two ways to give input to hardware architecture viz. bit parallel and bit series method.

Following part discusses few bit parallel and bit series architectures .In bit parallel method all bits of pixels are taken parallel. Hence it is called bit parallel method.

S. Wong et al. [1] implements a 16X1 SAD unit, called SAD16, which is equivalent to a macroblock row for MPEG. This design is inspired on the adder tree model presented in [7]. The authors state briefly how to extend the design to compute a 16X16 SAD reusing the original SAD16 unit to compute the remaining rows. The SAD16 unit is reused to implement the complete 16 X 16 SAD operation. Ingu Hwang et al. [2] proposed architecture, which consists of 4 basic blocks. The processing element array computes sixteen 4x4 SADs of a 16x16 macroblock. The adder & comparator block sums up the 4x4 SADs to form the SADs for 7 different block sizes and finds the minimum distortions and corresponding motion vectors. Sixteen 4x4 SADs, which are the outputs of a 16x16 PE array, are inputted to an adder & comparator block. Before adding them up, the 16 4x4 SADs are stored in the temporal registers and an adder tree sums them up to produce 8x4, 4x8, 8x8, 16x8, 8x16 and 16x16 SADs. Comparators compare total 41 SADs and save the 41 minimums with their corresponding motion vectors.

Ref.[3] optimize scan direction of the macroblock in the current frame, and the scan direction of the matching in the search area in order to reduce the access to the off-chip memory banks which stores the reference frame, and the

Manuscript received April 2015.

***Harsha Prakash Redkar**,Electronics and Telecommunicaton Engineering,GoaCollegeofEngineering, Goa, India.*

***Sonia Kuwelkar**,Electronics and Telecommunication Engineering,Goa college of engineering,Goa,India.*

ISSN: 2278 – 909X

on-chip memory banks which cache the search area. It uses zigzag scan of the macroblock in the current frame

In bit series method, input is taken one bit per cycle. It is called as online arithmetic (OLA). In OLA, all operation starts from most significant bit. In [4], MSB first method is used. A most significant bit (MSB)-first bit-serial design with early termination was proposed [4] which employed a FS within the range -15 to +16. Their experiments showed that on average, 50% of the computation, can be saved when an early termination scheme is employed. In [5] A SAD engine employing on-line arithmetic was also reported. This design has improved area-time product over previous bit-serial architectures.

Thus basic building block in ME architecture is SAD calculation and it can be achieved by bit series or bit parallel method. Hence we have implemented SAD by both bit series and bit parallel method.

II. SAD IMPLEMENTATION

A. Bit series method

In this method each pixel is processed in MSB first manner. We have designed the SAD unit which can process 16 pixels simultaneously. Processing 16 pixels simultaneously is advantageous as it can speed up the macroblock SAD calculation or the same unit can be reused for variable block size motion estimation. Standard Macroblock size is 16x16 hence processing 16 pixels simultaneously is equivalent to processing one row of macroblock and hence this unit speeds up the macroblock SAD calculation. Also smallest block size possible is 4x4 which includes 16 pixels. Hence same unit can be used as 4X4 SAD calculator and reused to calculate other larger block SAD. In this way same unit can be modified to achieve variable block size motion estimation.

The basic idea of OLA is to perform computations which overlap with the digit-by-digit communications of operands/results [5]. To generate the first digit of the result, $\delta + 1$ digits of the input operands are needed. Thus, after δ digits of the operands are received, for each new digit of the operands, a new digit of the result is obtained. For this reason, δ is known as online delay. Due to the online delay, after the last digits of the inputs are introduced into the system, a number of zero digits equal to the online delay have to be introduced to ensure a correct result.

Online arithmetic use redundant number system. One of the redundant number system is radix-2 signed digit representation. In radix-2 SD representation, the digit set is {-1, 0, +1}. Two bits are required to represent each digit, as shown in Table I. The 1st bit is positively weighted the second one is negatively weighted. From table, it is clear that SD representation can be interpreted as difference between two unsigned numbers. Example of SD number

e.g. 245=111111'11' ,SD=1010101010011001

TABLE I. DIGIT CODIFICATION IN RADIX-2 SD REPRESENTATION

Digit value	Digit Representation
+1	10
0	00
0	11
-1	01

Basic building blocks in bit series method are online carry save full adder (ol-CSFA) and online signed digit full adder(ol-SDFA). These components are designed as shown in fig.1 and fig.2.

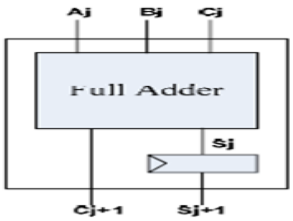

Fig.1. ol-CSFA

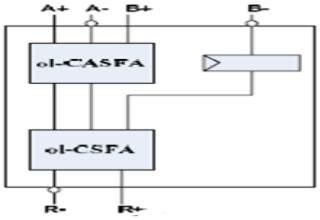

Fig.2. ol-SDFA

Thus in the motion estimation method SD representation is advantageous. Because SD calculates difference between current pixel (Ci,j) and reference block(Ri,j) with no computational cost. To calculate Di,j=Ci,j - Ri,j in SD form, we can use Ci,j as positively weighted and Ri,j as negatively weighed value.

These basic components are used to design adder tree. Adder tree and ol-SDFA are used to calculate SAD value of 16 pixels in signed digit form shown in fig.4. Input required for the adder tree should be in SD format. We used two 16 operand carry save adder trees to add positively weighted bits and negatively weighted bits separately. We required one more stage to convert conventional number to SD number before adder tree. This stage converts pixel values in unsigned form to SD form. This design follows the algorithm explained in [6] and Adder tree design is shown in fig.3.

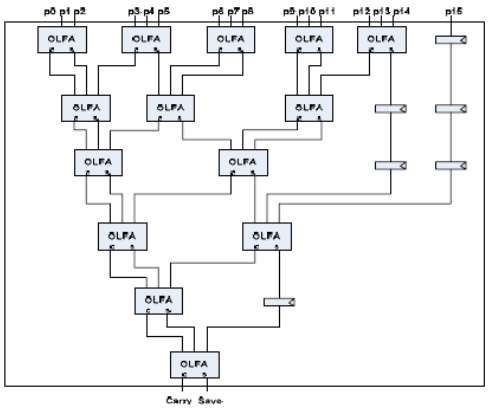

Fig.3. 16 operands carry save full adder

In this way bit series method is used to calculate SAD value.

B. Bit Parallel Method

To calculate SAD value by bit parallel method following basic blocks are required:

1) Absolute value calculator (AVC)

2) Row_ SAD unit

In bit parallel method it is not necessary to convert conventional pixel value to SD format. Only we need to determine which pixel value is greater to calculate absolute

ISSN: 2278 – 909X

difference. Absolute value calculator performs following operations:

- Compare Ci with Ri and determine which one is greater.
- (Ci-Ri) if (Ci>Ri) or (Ri-Ci) if (Ri>Ci)

In this way 16 AVC units are used in parallel and give 16 absolute difference values. These values are then passed to row_SAD unit. It adds outputs from 16 absolute value calculator as shown in fig.5.

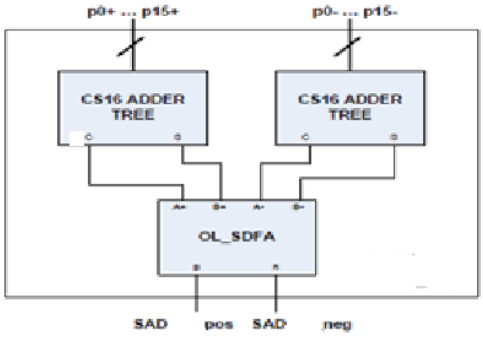

Fig.4.16 operands signed digit adder tree for SAD

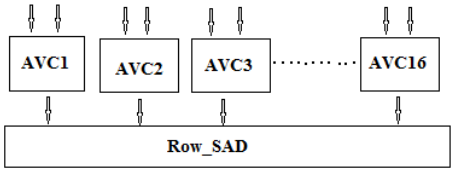

Fig.5. Row_SAD unit

These two designs are synthesized on FPGA and simulated by Modelsim 5.7f simulator. Simulation results are as shown in table II.

	Bit series	Bit parallel
Online delay (No. of clock cycles of zeros)	4	none
Output after clock cycles	12	Immediate

Bit parallel has online delay of 4 cycles and 8 cycles are required to input each bit of pixel. Hence after 12 cycles we could get the result of SAD value whereas in bit parallel method we could obtain SAD result immediately without online delay. Hence using bit parallel method we can achieve better throughput value than bit series. With this conclusion, we have performed motion estimation process using bit parallel method.

III. SYSTEM ARCHITECTURE

The Proposed ME system architecture consists of memory elements, absolute value calculator, SAD unit and motion vector calculator as shown in fig.6.

For implementation of motion estimation we used macroblock size of 16x16 and search area of 32x32 from reference frame. Hence current frame memory stores the current macroblock for which motion vectors are to be calculated and reference frame memory stores 32x32 search area pixel values.

As the macroblock size is 16x16,we used Row_SAD unit with accumulator to calculate SAD value of one Macroblock. For first SAD value of current macroblock and reference block, SAD value is compared with stored SAD minimum threshold value and minimum one is stored as current SAD minimum value. For subsequent blocks, a SAD value is compared with current SAD minimum value and updated at every macroblock completion. We used threshold value of SAD minimum is 65536.

Along with SAD minimum value, SAD unit outputs position of the reference block at which minimum has occurred. Hence this position in terms of row and column gives motion vector. Motion vector value is updated when SAD minimum value is updated. Thus SAD unit gives both SAD minimum and motion vector values.

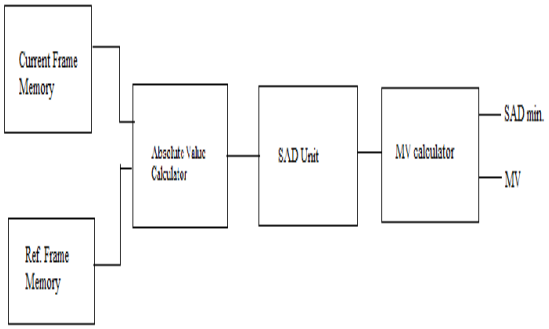

Figure 6. System Architecture

IV. SIMULATION AND IMPLEMENTATION OF HARDWARE MODEL

Simulations of hardware models are important to verify functionality of intended system. There are various sub units in our system. Hence test benches were developed to verify functionality of each sub unit independently. This simulation is carried out using Modelsim SE Plus 5.7f simulator and Hardware model is designed in VHDL language. Then all sub units were integrated to form complete motion estimation unit. A suitable test bench was developed to test functionality of integrated Motion estimation unit.

Then all sub units and integrated motion estimation unit are synthesized and implemented on FPGA using Xilinx ISE 10.1 .For synthesis targeted FPGA is Virtex II XC2V6000 and synthesized ME unit has following external port as shown in fig.7.

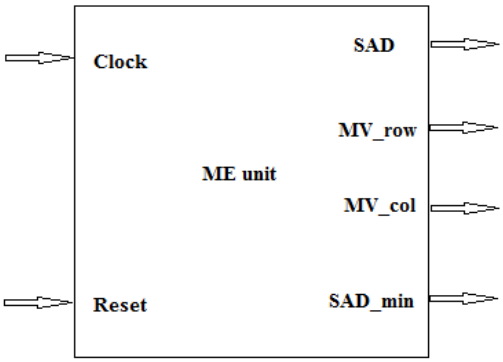

Fig.7 Motion Estimation Unit External ports

ISSN: 2278 – 909X

The proposed architecture runs at clock frequency of 125MHz.SAD unit which calculates SAD value of one macroblock has FPGA hardware utilization of 598 LUTs and runs at same frequency. To find best matching block for one current macroblock within 32x32 search area 39336ns are required i.e. one motion vector calculation time is 39.336 microseconds.Hence 25422 macroblock can be processed per second.

Table III shows Xilinx Virtex II device utilization after synthesis complete motion estimation design.

Comparative results for the proposed architecture and some related works are presented in Table IV.

TABLE III. DEVICE UTILIZATION

Logic Utilization	Used	Available	Utilization
Number of Slices	830	33792	2%
Number of Slice Flip-Flops	727	67584	1%
Number of 4 input LUTs	1561	67584	2%
Number of IOBs	98	1104	8%

TABLE IV. COMPARISON OF THE PERFORMANCE AND CIRCUIT SIZE

Design	MB per second	CLB slice/ LUTs	Frequency (MHz)	Device
[2]	16236	955	197	Altera Flex20KE
[5]AB1	1980	184	25	Xilinx XC40250
AS1	1900	1214	24	
AB2	19800	948	30	
AS2	28908	3732	22	
[6]	18532	1654	103.8	Altera Stratix
Our design	25422	1561	125	Virtex II XC2V6000

V. CONCLUSION

In this paper, we presented high performance hardware architecture for implementation of SAD based full search block matching motion estimation. The proposed architecture is synthesized on Virtex II XC2v6000 FPGA using Xilinx ISE 10.1 and tested using Modelsim SE 5.7f simulator. The architecture is verified at 125 MHz clock frequency with hardware utilization of 1561 LUTs. The design can process 25422 macro blocks per sec which is sufficient for video processing.

REFERENCES

[1] J. Olivares, J. Hormigo, J. Villalba, I. Benavides, E. L. Zapata, "SAD computation based on online arithmetic for motion estimation, "Microprocessors and Microsystems, Accepted with ref IJB/2005/4, 2005.

[2] S. Wong, S. Vassiliadis, S. Cotofana, "A Sum of Absolute Differences Implementation in FPGA Hardware," 28th Euromicro Conference (EUROMICRO'02), pp. 183–188, Dortmund, Germany, 2002.

[3] Shuichi Asano, ZhengZhi Shun and Tsutomu Maruyama, "An FPGA implementation of full-search variable block size motion estimation," Proc. IEEE International Conference on Field Programmable Technology, Beijing, Dec 2010.

[4] P.Muralidhar,C.B.RamaRao,I.Ranjith Kumar,"Efficient Architecture for Variable block size Motion Estimation of H.264 Encoder", International Conference on Solid-State and Integrated Circuit (ICSIC 2012)IPCSIT vol. 32 (2012), Singapore,2012.

[5] A. Ryszko, K. Wiatr, "An assesment of FPGA suitability for implementation of real-time motion estimation, "EUROMICRO Symposium On Digital Systems Design, Proceedings, pp.364-367, 2001.

[6] H. Loukil, F. Ghozzi, A. Samet, et al. "Hardware implementation of block matching algorithm with FPGA technology," 6th International Conference On Microelectronics, Proceedings, pp. 542–546, 2004.

[7] C.L. Su, C.W. Jen, "Motion estimation using MSD-first processing", in Proc. IEEE Circuits, device and systems, vol. 150, Issue 2, pp. 124-133, Apr.2003.

[8] S. Chen, B. Mulgrew, and P. M. Grant, "A clustering technique for digital communications channel equalization using radial basis function networks," *IEEE Trans. on Neural Networks*, vol. 4, pp. 570-578, July 1993.

.

ISSN: 2278 – 909X