VOL. 12, NO. 24, DECEMBER 2017 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

# FPGA IMPLEMENTATION OF SUM OF ABSOLUTE DIFFERENCE (SAD) FOR VIDEO APPLICATIONS

D. V. Manjunatha[1], Pradeep Kumar[1] and R. Karthik[2]
[1]Department of Electrical and Computer Engineering, Alvas Institute of Engineering and Technology, Moodbidri, India
[2]Department of Electrical and Computer Engineering, MLR Institute of Technology, Hyderabad, India
E-Mail: dvmanjunatha@gmail.com

**ABSTRACT**

Advances in multimedia have expanded the boundaries of communication systems and changed the communication industry over the past a few decades in the applications such as digital TV, DVD video, HDTV, internet video streaming, video conferencing, mobile technology, patrolling, object tracking, and medical applications. Video compression (VC) placed a significant part in the realization of these technologies by bridging the gap between the demand for quality, performance and limitations of current storage and transmission capabilities. Motion estimation (ME) is the power hungry block in the video compression system (VCS). The sum of absolute difference (SAD) is the most repeated operation in the motion estimation subsystem. This paper proposed the field programmable gate array (FPGA) Implementation of 4X4 SAD architecture. The design is simulated using Xilinx integrated software environment (ISE) and synthesized using Xilinx synthesis tool (XST) on Spartan-3 FPGA board. The proposed SAD estimates area acquired and latency.

**Keywords:** VC, ME, VCS, SAD, FPGA, ISE, XST.

## 1. INTRODUCTION

Video compression systems are very helpful in commercial products, from consumer electronic devices as digital camcorders, cellular phones to video teleconferencing systems. Block Based motion estimation searches for the best matching block between the current and reference macro blocks (MBs), for the operation, the sum of absolute difference is one of the most frequent employed criteria as a result, motion estimation produces a motion vector , which represents the motion of the macro block. Sum of absolute difference is an algorithm for measuring the similarity between the image blocks. It works by taking the absolute difference between each pixel in the original block and the corresponding pixel in the block being used for comparison. These differences are summed to create a simple metric of block similarity. For example the sum of absolute differences to identify which part of a search image is most similar to a template image. In the example, the template image is 3 by 3 pixels in size, while the search image is 3 by 5 pixels in size. Each pixel is represented by a single integer from 0 to 9. The template and the search image pixel values are as shown in Table-1. There are exactly three unique locations within the search image where the template may fit: the left side of the image, the centre of the image, and the right side of the image. To calculate the SAD values, the absolute value of the difference between each corresponding pair of pixels is used: the difference between 2 and 2 is 0, 4 and 1 is 3, 7 and 8 is 1, and so forth.

**Table-1.** Pixel values of the image / frame.

| Template image | Search image |
|---|---|
| 2 5 5 | 2 7 5 8 6 |
| 4 0 7 | 1 7 4 2 7 |
| 7 5 9 | 8 4 6 8 5 |

Calculation of SAD values for each of these locations is given below:

**Table-2.** Absolute difference values of three images.

| Left | Centre | Right |
|---|---|---|
| 0 2 0 | 5 0 3 | 3 3 1 |
| 3 7 3 | 3 4 5 | 0 2 0 |
| 1 1 3 | 3 1 1 | 1 3 4 |

For each of these three image patches, the absolute differences are added together, giving a SAD value of 20, 25, and 17, respectively, from these SAD values, it is apparent that the right side of the search image is the most similar to the template image, because it has the least difference as compared to the other locations. The rest of the paper is organized as follows: Section 2 describes the related work in the sum of absolute differences as a metric of motion estimation to find the motion vectors. Proposed adder architecture is described in section3. Section 4 gives the details sum absolute difference architecture. Section 5 describes the results and discussion, finally we conclude in section 6.

## 2. RELATED WORK

There are varieties of methods available in the literature to find the motion vectors where there is lot of tradeoffs between the speed, power and area during the hardware implementation. The work presented by [1-6] shows that motion estimation aims at reducing the temporal redundancy between successive frames in a video sequence using high speed SAD. The work in [7] proposed that SAD architecture and compared much architecture in terms of area and delay. The work in [8] presented H.264/AVC encoder which employs 1024 SAD processing units (PE) which use 305k gates. The work in [9] proposed that SAD architecture and compared much architecture in terms of area and delay. The authors in [10] proposed that the SAD architecture with 1 and 2stage pipeline, the work done in [8]-[12] presented the SAD architectures in terms of gate count and delay optimization, but the aspects of power consumption focusing on low power devices were not presented. In this work the Area in terms of LUTs from FPGA implementation and the power consumption, area and delay are presented from the ASIC implementation using RTL Compiler with 180nm technology.

## 3. PROPOSED ADDER ARCHITECTURE

The addition of two binary numbers is the fundamental and most commonly used arithmetic operation in video compression for determining the motion vectors in the video codec. Hence, binary adders are critical building block of most of the video systems. The key challenge addressed in the adder design is the carry propagation operation involving all operand bits. Various different architectures for binary addition have been proposed, based upon the delay, there are two fundamental adder architectures:

### 3.1 Ripple carry adder

It is the simplest architecture. For an n-bit adder, intermediate carries are generated sequentially. It has smallest area, longest delay & consumes lowest power. The 4 bit ripple carry adder is as shown in Figure-1 as below:

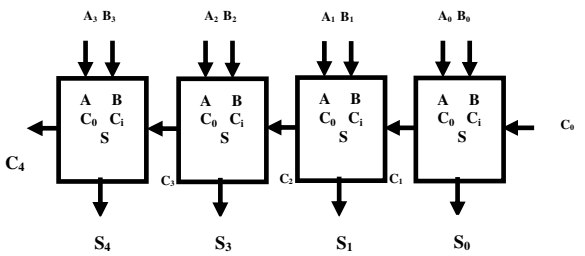


**Figure-1.** 4 bit ripple carry adder architecture.

### 3.2 Carry look ahead adder

Here the carry is generated in parallel to reduce the processing delay; hence it is faster than ripple carry architecture. It consumes more area & power. The 4 bit carry look ahead adder is as shown in Figure-2 as below:

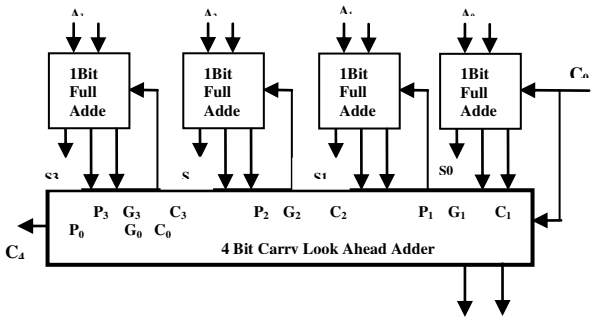


**Figure-2.** 4 bit carry look ahead adder architecture.

In this work we have to develop the low power 4X4 sum of absolute difference algorithm so we use the ripple carry adder architecture.

## 4. SUM OF ABSOLUTE DIFFERENCE (SAD)

SAD algorithm is used for measuring the similarities between the images by calculating the absolute differences between the pixels of the image (Template image) and their corresponding ones (Search Image) in the macro block, then these differences are added up to result in the similarity block. It requires only two basic mathematical operations addition & shifting. The SAD algorithm is the simplest metric which considers all the pixels in the block for computation and also separately, which makes its implementation easier and parallel. Due to its simplicity this algorithm is one of the fastest and can be used widely in block motion estimation and object recognition. In this work we implemented the 4X4 sum of absolute difference algorithm on both FPGA (Spartan -3) for prototyping using Xilinx ISE simulation tool and the Xilinx XST synthesis tool for the area in terms of LUTs, to find the latency and on ASIC using cadence RTL Compiler to find the power consumed, area occupied and the delay taken. The block diagram of sum of absolute difference is as shown in Figure-4 and the SAD hierarchy is shown in Figure-3 below:

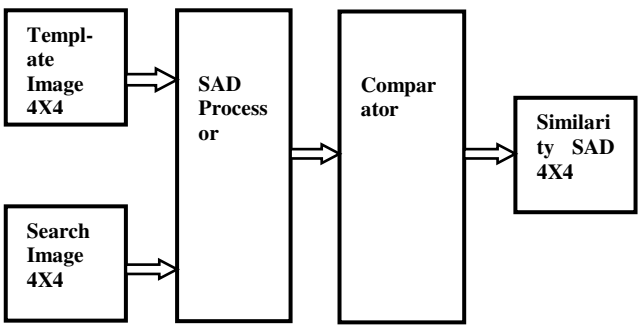


**Figure-3.** Block diagram of 4x4 SAD architecture.

The application of sum of absolute difference architecture includes the following:

- Object recognition
- Video Compression
- The generation of disparity maps for stereo images.
- Video Conferencing.

## 5. RESULTS AND DISCUSSIONS

Here first the template image and the search images of the video sequence are given to absolute difference block to find the absolute of the difference and then all the absolute difference values are added up for the entire macro block. Further, the same procedure is repeated for all the macro blocks of the video sequence and saved the added value finally the out of many saved added values of the whichever is having the smallest value that will correspond to the motion vector and the motion estimation can then be performed. The work in this paper is implemented in both ASIC design using RTL compiler and the FPGA implementation done on Spartan-3 device. The results are presented in the following two sections

### 5.1. FPGA results

In this work we developed the verilog code for low power 4X4 Sum of Absolute Difference algorithm and synthesized using Xilinx synthesis Tool. Xilinx ISE (Integrated Software Environment) is a software tool produced by Xilinx for synthesis and analysis of HDL designs, enabling the developer to synthesize ("compile") their designs, perform timing analysis, examine RTL diagrams, simulate a design's reaction to different stimuli, and configure the target device with the programmer. The synthesis results are mentioned below

**Table-3.** FPGA synthesis results of adders.

| | **Full adder** | **8-bit RCA** | **9-bit RCA** | **10-bit RCA** |
|---|---|---|---|---|
| No. Of slices | 1 out of 3584 | 9 out of 3584 | 11 out of 3584 | 11 out of 3584 |
| 4 input LUTs | 2 out of 7168 | 15 out of 7168 | 19 out of 7168 | 19 out of 7168 |
| Delay | 7.824 ns | 16.191 ns | 18.497 ns | 18.980 ns |
| Total memory | 122160 Kbytes | 112140 Kbytes | 131168 Kbytes | 144556 Kbytes |

**Table-4.** FPGA synthesis results of 4X4 SAD.

| **particulars** | **8-bit sum of absolute difference (Abs)** | **4x4 SAD** |
|---|---|---|
| No. of slices | 14 out of 3584 | 218 out of 3584 |
| No. Of 4 input LUTs | 25 out of 7168 | 400 out of 7168 |
| IO Buffers | 25 | 400 |
| Memory usage | 111528 Kbyte | 143260 Kbyte |

Observing the above Tables 3 and 4 the above table clearly gives the area consumed in terms of Look up Tables the memory usage the design implemented and the delay. In FPGA implementation we can only find out the area in terms of LUTs and the delay but we need to find the power so we go for ASIC design.

### 5.2. ASIC DESIGN RESULTS

Here the synthesis is done using the RTL Compiler of Cadence EDA tools. Cadence RTL Compiler (Cadence RC) is a logic synthesizer. It takes a HDL model, a library of standard logic cells, a set of synthesis commands and it generates a network of interconnected logic cells (a net list).

The designs are targeted using 180 nm technology for Cadence RTL compiler. In ASIC Design the power consumed area acquired, Delay of the 1 bit adder and 4X4 sum of absolute dif ference are implemented which are tabulated in Tables 5 and 6 as below:

**Table-5.** Synthesis results 1 bit adder.

| **S. No** | **Particulars** | **Full adder which uses EXOR & Nand gates** |
|---|---|---|
| 01 | Leakage Power in nW | 96.144 |
| 02 | Dynamic Power in nW | 492.096 |
| 03 | Total Power in nW | 588.590 |
| 04 | Delay T in Ps | 157 |
| 05 | Area in SQ Microns | 24 |

**Table-6.** Synthesis results low power 4X4 SAD.

| S. No | Particulars | Low power 4X4 SAD |
|---|---|---|
| 01 | Leakage Power in nW | 25149.932 |
| 02 | Dynamic Power in nW | 500144.707 |
| 03 | Total Power in nW | 525294.390 |
| 04 | Delay T in Ps | 04490 |
| 05 | Area in SQ Microns | 6397 |

### 5.3. SNAPSHOT RESULTS

#### 5.3.1 The simulation results of 4X4 SAD

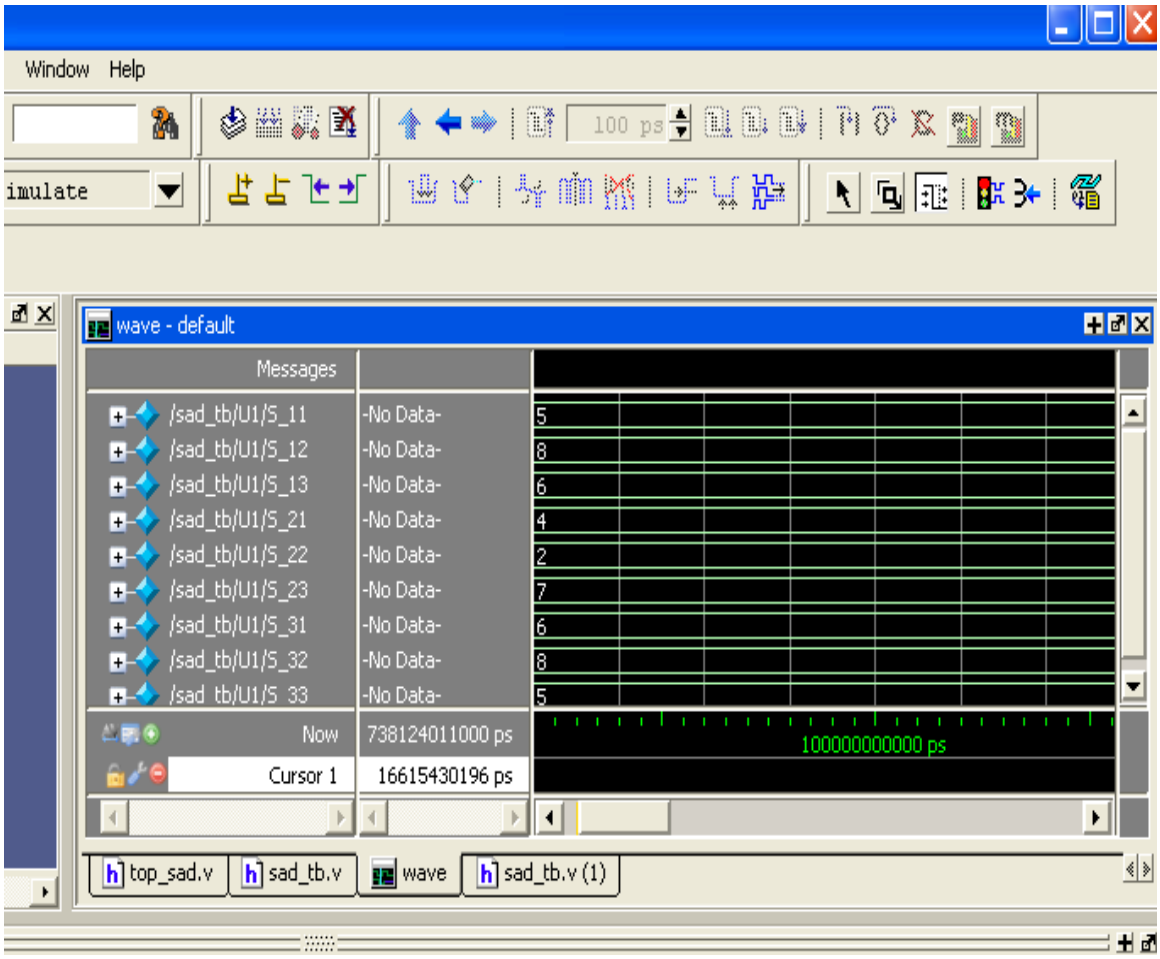


**Figure-4.** Simulation results of 4X4 sad.

The 4X4 SAD consists of 1) Absolute difference block 2) Full Adder 3) 8 bit Ripple Carry adder 4) 9 bit ripple carry adder 5) 9 bit ripple carry adder 6) 10 bit ripple carry adder 7) 11bit ripple carry adder 8) 12 bit ripple carry adder circuits

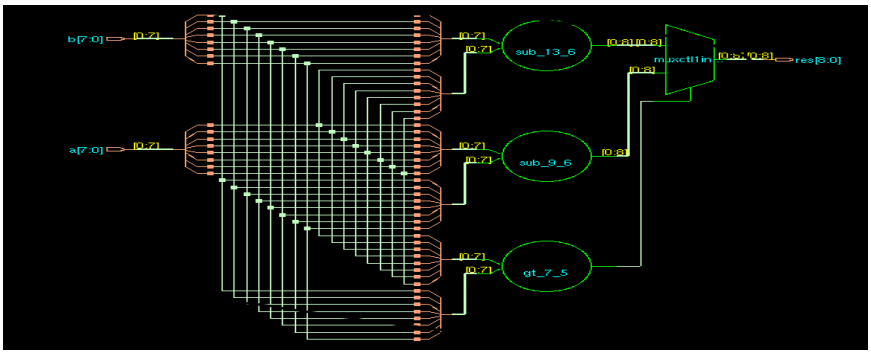


**Figure-5.** Synthesis results of absolute difference block.

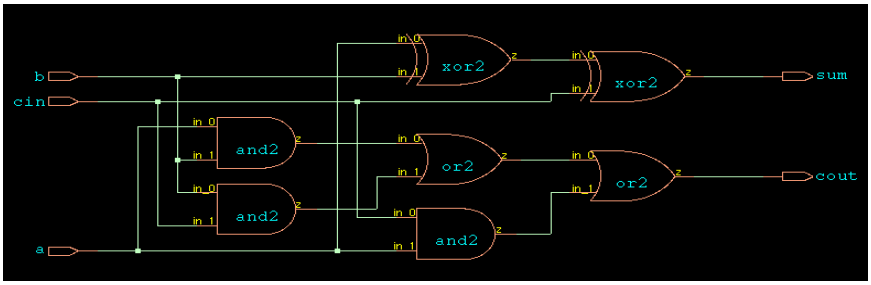


**Figure-6.** Synthesis results of full adder.

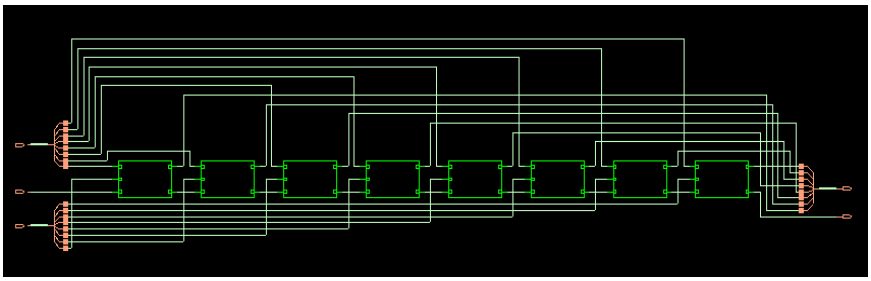

**Figure-7.** Synthesis results of 8 bit ripple carry adder.

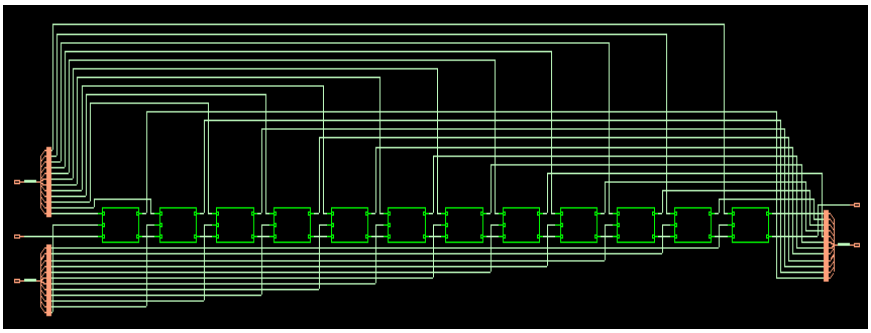

**Figure-8.** Synthesis results of 12 bit ripple carry adder.

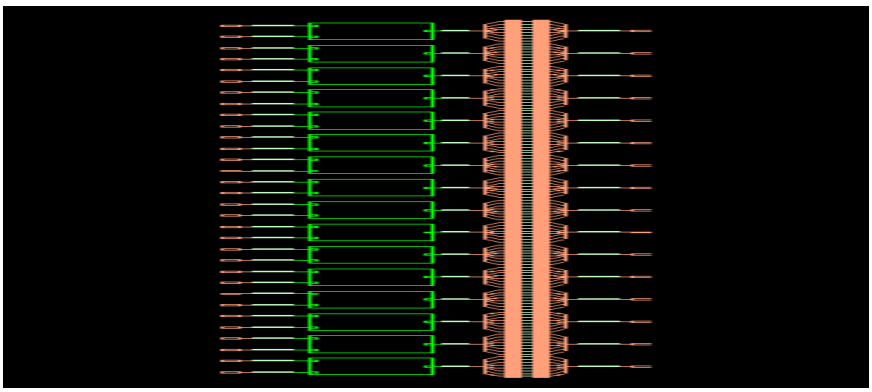

**Figure-9.** Synthesis results of 4X4 SAD.

## 6. CONCLUSIONS

The heavy computational cost of the block matching algorithms (BMA) can be significant problem in real -time coding applications. To reduce the computational complexity, different VLSI architectures are designed to speed up the associated massive arithmetic calculation. Usually VLSI architecture means the FPGA design and the ASIC Design which works at very high speed as they are semicustom designs. In this work the low power 4X4 SAD both in FPGA and ASIC are implemented, the results are as shown in the results and discussion section. Finally the conclusion is that the prototypes can be developed using FPGA and their implementation to find the actual area power and the delay parameters, can be found out by many methods but in this work, the considerable reduction in power and area with the corresponding speed is achieved. Further, improvements can be achieved using the proper power, area and speed optimization techniques.

## REFERENCES

[1] Y. Wang *et al.* 1999. Hilbert scanning search algorithm for motion estimation. IEEE transactions on circuits and systems for video technology. 9(5): 683-691.

[2] S. Lee *et al.* 1998. New motion estimation algorithm using adaptively quantized low bit-resolution image and its VLSI architecture for MPEG2 video encoding. IEEE transactions on circuits and systems for video technology. 8(6): 734-744.

[3] M. Pickering *et al.* 1997. An adaptive search algorithm for block matching motion estimation. IEEE transactions on circuits and systems for video technology. 7(6): 906-912.

[4] J. Y. Tham *et al.* 1998. A novel unrestricted center biased diamond search algorithm for block motion estimation. IEEE transactions on circuits and systems for video technology. 8(4): 369-377.

[5] H. Wang *et al.* 1999. Fast algorithm for the estimation of motion vectors. IEEE transactions on image processing. 8(3): 435-438.

[6] J. W. Kim *et al.* 1994. Hierarchical variable block size motion estimation technique for motion sequence coding. Optical engineering. 33: 2553-2561.

[7] S. Wong, S. Vassiliadis, S. Cotofana. 2002. A Sum of Absolute Differences Implementation in FPGA Hardware. 28th Euromicro Conference. pp. 183-188.

[8] T. C. Chen, *et al.* 2006. Analysis and Architecture Design of an HDTV720p 30 Frames/s H.264/AVC Encoder. IEEE TCSVT. 16(6): 673-688.

[9] J. Vanne, E Aho, T D Hamalainen and K Kuusilinna. 2006. A High-Performance Sum of Absolute Difference Implementation for Motion Estimation. IEEE TCSVT. 16(7): 876-883.

[10] Yufei L, Feng Xiubo and Wang Qin. 2007. A High-Performance Low Cost SAD Architecture for Video Coding. Consumer Electronics IEEE Transactions. 53(2): 535-541.

[11] Z. Liu, *et al.* 2007. 32-Parallel SAD Tree Hardwired Engine for Variable Block Size Motion Estimation in HDTV 1080P Real-Time Encoding Application. IEEE SiPS. pp. 675-680.

[12] Stephan Wong, Stamatis Vassiliadis, and Sorin Cotofana. 2009. A Sum of Absolute Differences Implementation in FPGA Hardware. International Journal of Electrical and Computer Engineering. 4: 9.