IMPLEMENTATION OF SAD ALGORITHM FOR MOTION ESTIMATION

Vasanthapriya.K[1], Vaishnavi.P[2], Nithya Lakshmi.A[3]

[1]*Student, Electronic and Communication Engineering, Prince Shri Venkateshwara Engineering College, Tamil Nadu, India*
[2]*Student, Electronic and Communication Engineering, Prince Shri Venkateshwara Engineering College, Tamil Nadu, India*
[3]*Assistant Professor, Electronic and Communication Engineering, Prince Shri Venkateshwara Engineering College, Tamil Nadu, India*

ABSTRACT

Motion estimation (ME) is theprocess of determining motion vectors that describe the transformation from one 2D image to another, usually from adjacent frames in a video sequence and it is the most complex part of most popular video compression standards. In the proposed project, an efficient implementation of SAD algorithm to find the motion vectors for the motion estimation is proposed. The proposed algorithm makes motion estimation more concise. The texture-based search strategies are based on temporal stationarity and spatial homogeneity with Sobel edge operator. The four-pixel SAD unit which is the basic processing element (PE) in our proposed architecture is used for SAD calculation and Sobel edge operator computation. The architecture achieves very high data utilization and data throughput. The proposed method for calculation of motion vector is by using SAD algorithm. The processing element is used in a pipelined manner for computing SAD value thus the proposed architecture can significantly reduce the hardware cost since it reduces the number of processing elements used for calculation of motion vectors, decreases the area utilization and the power consumption. The design can be implemented with SMIC 0.18µm CMOS technology.

Keyword: -*Adaptive multi-resolution motion estimation1, SAD algorithm2, VLSI architecture3, and FPGAs4 etc.*

1. INTRODUCTION

Motion estimation (ME) is the most complex part of most popular video compression standards such as MPEG-1/2/4 and H.264/AVC. The goal of integer motion estimation is to reduce temporal redundancies between the current frame and the reference frame. These video coding standards also use new techniques such as variable block size motion estimation (VBSME) and multiple reference frames. Therefore, real-time motion estimation implementation for high definition (HD) video encoder brings great challenges for hardware resources and power consumption.

1.1 Motion Estimation

In the latest CTI machines and MRI scanners, medical data base require large amount of memory storage and large bandwidth for transmission of Data. Thus it needs video compression. As the Database of medical images contain number of Frames, while coding of these images, there is need of motion estimation. This Disparity means pixel displacement between corresponding points in multi view images or stereo images. Researchers have been giving special attention to stereo vision systems. Stereo vision systems aim at reconstructing 3D scenes by matching two or more images taken from slightly different viewpoints.

1.2 Stereo Vision

Stereo vision systems generate accurate depth information of an observed scene. Most stereo vision implementations are based on two forward-facing cameras, where each camera delivers a 2D projection of a scene. Then all disparities are determined simultaneously by applying energy minimization techniques. However, most of these methods are computationally expensive. However, the proposed AMMEA doesn't use a fixed search range and the same down-sampling. For different sequences, adaptive search strategies are adopted to have a better balance between computational expense and performance. The spatial homogeneity and temporal stationary characteristics of video sequences are detected to adaptively determine search strategies. SAD algorithm is used to find the motion vector.

2. SAD ALGORITHM

In digital image processing, the sum of absolute differences (SAD) is a measure of the similarity between image blocks. It is calculated 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, the L1 norm of the difference image or Manhattan distance between two image blocks.

The sum of absolute differences may be used for a variety of purposes, such as object recognition, the generation of disparity maps for stereo images, and motion estimation for video compression. Example: This example uses the sum of absolute differences to identify which part of a search image is most similar to a template image. In this 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.

There are exactly three unique locations within the search image where the template may fit: the left side of the image, the center 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.

2.1 Comparison to other metrics

- **Object recognition**

The sum of absolute differences provides a simple way to automate the searching for objects inside an image, but may be unreliable due to the effects of contextual factors such as changes in lighting, color, viewing direction, size, or shape. The SAD may be used in conjunction with other object recognition methods, such as edge detection, to improve the reliability of results.

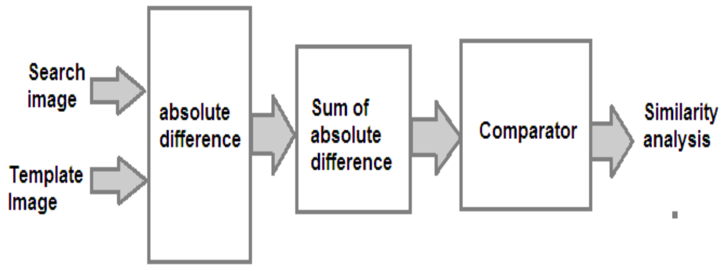
- **Video compression**

SAD is an extremely fast metric due to its simplicity; it is effectively the simplest possible metric that takes into account every pixel in a block. Therefore it is very effective for a wide motion search of many different blocks. SAD is also easily parallelizable since it analyzes each pixel separately, making it easily implementable with such instructions as ARM NEON or x86 SSE2. For example, SSE has packed sum of absolute differences instruction (PSADBW) specifically for this purpose. Once candidate blocks are found, the final refinement of the motion estimation process is often done with other slower but more accurate metrics, which better take into account human perception. These include the sum of absolute transformed differences (SATD), the sum of squared differences (SSD), and rate-distortion optimization.

2.2 Blocks of SAD processor

2.2.1 Absolute difference block

In SAD processor, the Absolute difference block is used to calculate the absolute differences between the reference pixels of 4X4 block size and the corresponding current input pixel data of 4X4 block size; in the larger search area. 16 Absolute difference units are required for 4X4 block size. Figure 1 represents the block diagram of SAD processor.



Simple Block diagram of SAD Algorithm for motion estimation

Fig 1: Block diagram of SAD processor

2.2.2 Sum of absolute difference block:

The outputs of each absolute difference units of 4X4 block are summed to form the single SAD value, and the process is repeated for next input 4X4 block size with reference pixel of 4X4 block size.

2.2.3 Comparator

The single SAD values of the each 4X4 block size in the search area are compared using comparator for the minimum SAD value. The corresponding 4X4 block sized current input pixel data of the minimum SAD value is considered as the block similar to the reference block against the other SAD blocks. The block size depends on the size of the reference block.

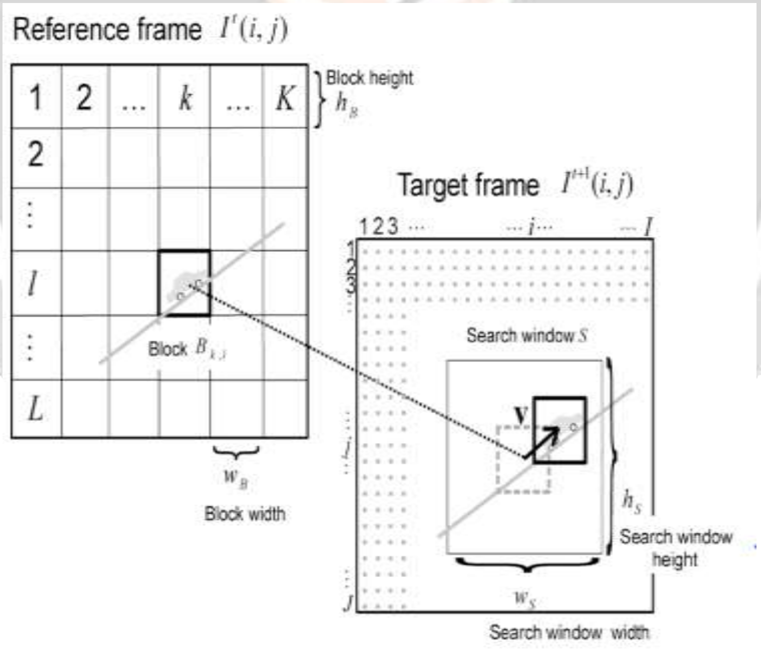


Fig 2: Block matching with search window range

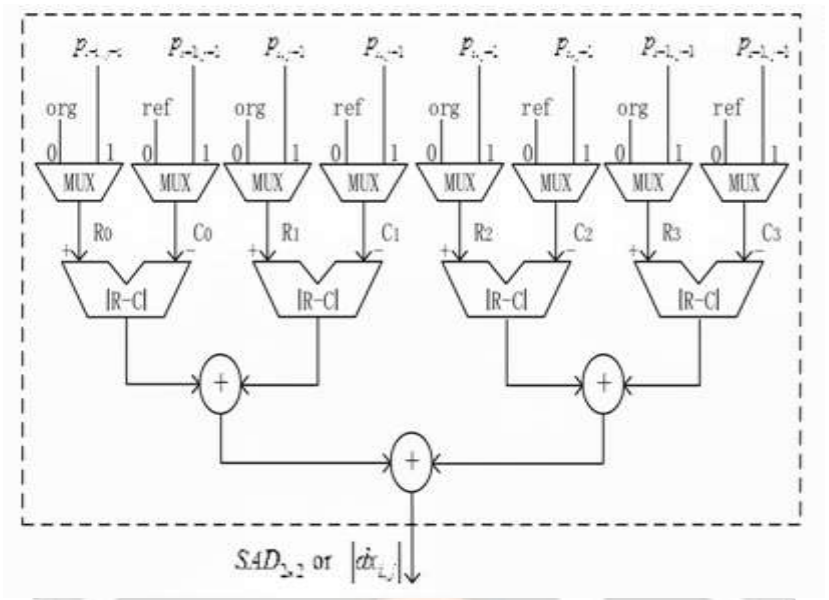


Fig 3: Four pixel PE architecture

SAD (Sum of Absolute Difference) algorithm is based on pixel based approach is use for finding the disparity. Sum of absolute differences (SAD) is an algorithm for measuring the similarity between 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 if the left and right images exactly match, the resultant will be zero. Fig 3 represents the four pixel SAD unit which is the PE of the proposed architecture.

Motion estimation is defined as searching the best motion vector which is the displacement of the coordinate of the best similar block in previous frame for the block in current frame. The most commonly used metric to calculate the distortion is the Sum of Absolute Differences (SAD) which adds up the absolute differences between corresponding elements in the candidate and reference block. The heavy computational cost of the block matching algorithms (BMA) can be a 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. However, the necessity of specialized hardware goes against the demand of flexibility required by the current coding video systems. A viable solution to this problem is to use a programmable processor core along with programmable gate arrays devices (FPGAs) which is in charge of performing critical tasks.

The reasons for and the benefits of using FPGAs are: Increased flexibility and quick adaption of new developments, sufficient performance and faster design times are achieved by re-using IP cores and high-level design languages (such as VHDL) speed up designs significantly.

In this context, the design is intended to speed up the computation of the minimum SAD by its implementation on a FPGA while a core processor supplies the reference and candidate blocks to the FPGA device. It proposes a FPGA architecture to compute the minimum SAD. This design can be integrated with any BMA (full search or another efficient search strategy). Figure 4 represents the motion estimation system.

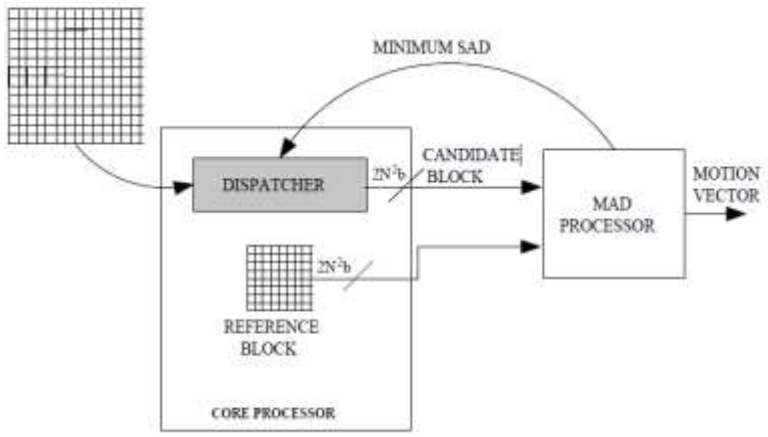


Figure 4 Motion estimation system

In spite of inherent parallelism in SAD, the full parallel implementation re-quires a large amount of operands for the typical block size (16 x 16 pixel macro block needs 512-8(bit operands). Due to the large amount of hardware, the computation of the SAD in only one row of a macro block (16 x 16) is implemented on a FPGA device and they propose between replicate or pipeline the design to obtain the 16x16 computation.

In four FPGA chips with 1234 I/O pins each are used for a completely parallel design. On the other hand, the use of on line arithmetic (OLA) for motion estimation is proposed to speed up the computation by early truncation of the SAD calculation. In, a serial architecture (pixel by pixel) for 8X8 blocks is proposed based on ASIC implementation. The following figures (5, 6 & 7) depict the steps followed in the estimation of the match block.

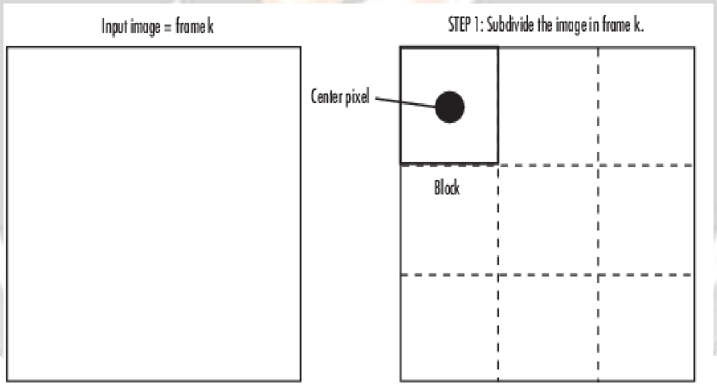


Fig5: Input image is divided into blocks

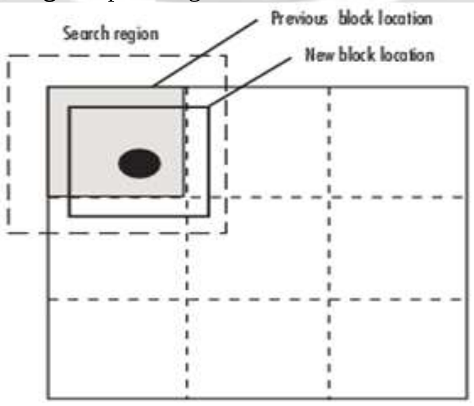


Fig 6: Search window is set in frame k+1

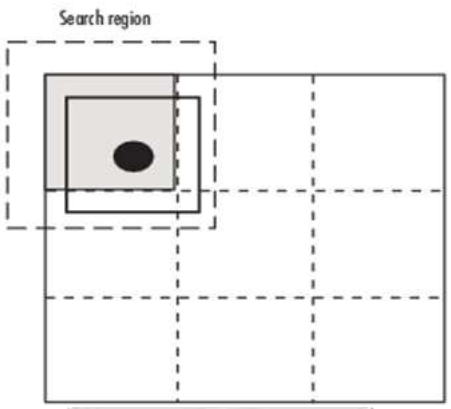


Fig 7: Search is done within the window

2.3 Search Strategy Decision

In Search Strategy Decision module, PE arrays calculate a current MB Diff and H(r, c) first, then, it will determine which search strategies can be used will be transferred to each level's FSM. Finally, motion estimation module will work with the configuration parameters. The detailed search strategy decision procedure will be analyzed below:

Step 1: Stationary MB Decision.The method for detecting stationary region is to calculate the difference between current and reference MB .When computing the difference, the control signal of MUXs will be set to "0" to calculate SAD. The calculation result will be directed to search strategy decision module and compared with threshold T to determine the adopted search strategy.

Step 2: Homogeneous MB Decision. If the difference between current MB and reference MB is greater than threshold T, homogeneous decision module will be in work.

Since there are totally 64 parallel four-pixel PEs in one search path, all PE arrays can be connected to the 18×4 systolic array for calculating 32 amplitudes of the edge vector defined. Thus, just 8 cycles are needed to calculate all the sum edge amplitude of one MB. One original MB pixels are buffered into the 16×16 systolic arrays as the blue circles. Since the proposed architecture of AMMEA is block-level pipelined and the scan mode is zigzag coding pattern, it is difficult to fetch adjacent block pixels. So, in order to calculate the Sobel edge operator of the edge pixels in the MB, it is necessary to pad one pixel in the upper and lower boundary as the yellow circle shown. Meanwhile, the padding pixels of the left and right boundaries are fed back to the edge of original MB pixels as the white circle shown. Therefore, 32 amplitudes of the edge vector can be calculated per cycle and 8 times of upward move are needed to calculate the sum edge amplitude of one MB. The calculation result will be directed to search strategy decision module for search strategy decision.

Step 3: Search Strategy Decision. The calculation results from step1 and step2 will be sent into the threshold compare module at first. The best search strategy will be selected by the criterion as discussed in Section 2.2.

Configuration parameters will be transferred to each level's FSM and set the control signal of MUXs to "0".

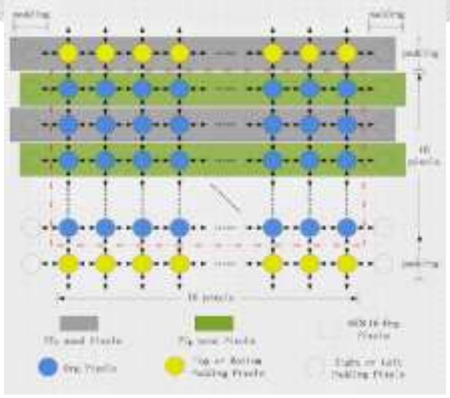


Fig 8: 16×16 Systolic array of current MB

3. COMPUTATION OF SAD IN PIPELINE MANNER

Initially for the given 8X8 block size, each block holds 64 pixels. With the 4-pixel SAD unit (2X2), 8 SAD unit is allotted for a block in the existing system. A single SAD unit goes to an idle state after calculating the SAD value for the given range. The proposed project implements this process in a pipelined manner.

The timing of the computation for the 4 x4 SAD processor is shown in Figure 9. In each cycle, the outputs corresponding to the absolute value block (*jci;j ¡ ri;j j*, each of the four steps in the adder-tree (P (*i*)), and the comparator (COMP) are represented. In fact, regarding the comparator, this does not really constitute the output, but rather the last digit used for the comparison. The zero digits represent the zero values which indicates the exact match of the block considered.

The worst case occurs when a new minimum SAD is found, and then 21 cycles are required for the full process, as shown in the figure 9 a new SAD computation can start after 16 cycles in which case this period of time is the maximum between two consecutive SAD computations. This period is reduced if the candidate SAD is rejected before. In the best case, this happens after analyzing the MSD of the candidate SAD, i.e., after 9 cycles. Therefore, the number of cycles for a SAD computation and comparison is between 9 and 16 for a 4 x4 SAD processor. This period ranges from 13 to 20 cycles for an 8 x8 block, and from 17 to 24 cycles for a 16 x16 block.

In the pipelining process, the SAD unit is not been sent to the idle state whereas it starts calculating the SAD value for the regions where the calculation is not yet been initiated. By this way, this project reduces the Processing elements to a large extent and the limited PE are efficiently utilized.

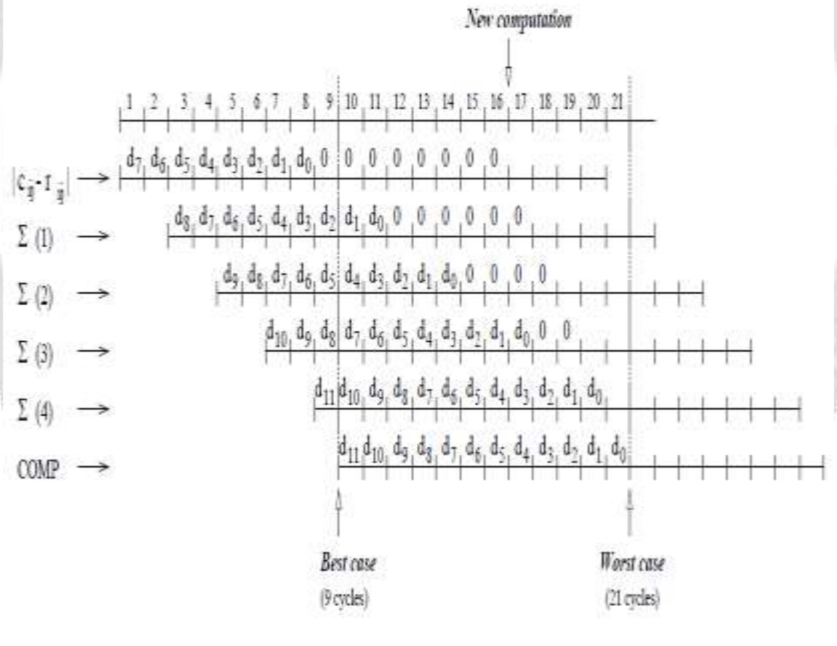


Fig 9: Timing of the 4 pixel SAD unit processor

3.1 Early termination of SAD calculation

Several video sequences have been processed to estimate the number of clock cycles saved. The parameters used are:
- 16x16 block size.
- 24x24 search window.
- Full-search block matching algorithm.
- 150 frames of each video have been evaluated.

The traditional model uses a final comparator for the SAD comparison. A new model is proposed, which introduces several comparison levels into the adder tree to evaluate partial SAD information. It is possible that partial SADs of 64 pixels or 128 pixels of a 16x16 block are greater than the reference SAD; if so, the SAD calculation can be stopped before running the entire number of cycles, which cannot be done with the traditional model. The new model for partial comparison. The added cost for the new model is the area occupied of six new comparators. Nevertheless, each comparator only requires 6 LUTs and involves less than 2% of the final area.

The impact of cell level computational logic at the block level is addressed by incorporating the Full adder in the comparator is addressed in this project. The transistor stacking concept in the proposed architecture has reduced the leakage power by a significant amount (7- 43 %).

The designs were modeled with the Verilog HDL and used 65nm technological library node for the synthesis in Design compiler tool. It can be observed that the proposed architectures are more power efficient than the counterpart regular architecture and enables the architectures to be analyzed with different corners of design constraints.

4. CONCLUSION

The proposed project is an efficient implementation of SAD algorithm to find the motion vectors for the motion estimation. The proposed algorithm exhibits its advantages to existing method by providing the motion estimation more concise and reducing the number of processing elements used for the motion estimation. And thus the area utilization, power consumption and hardware cost is reduced. It also provides high data reuse that is appropriate for VLSI implementation. In this architecture the four- pixel PE array reuse scheme is used. The design can be implemented with SMIC 0.18μm CMOS technology and costs 950K gates count, and it supports the real-time encoding of 1080P@30fps with two reference frames under a clock frequency of 150MHz.

5. REFERENCES

[1]. T.C.Chen et al, (2006) ‘Analysis and Architecture Design of an HD720p 30 Frames/s H.264/AVC Encoder,’ IEEE Trans. Cir. Syst. Video Tech., vol. 16, no. 6, pp. 673-688.

[2]. L.Deng, W. Gao, (2005) ‘An efficient hardware implementation for motion estimation of AVC standard,’ IEEE Trans.Consumer Electron., vol. 51, no. 4, pp. 1360-1366.

[3]. X.Q,C.J.Duanmu, C.R. Zou. (2005) ‘A multilevel successive elimination algorithm for block matching motion estimation’, IEEE Trans. Image Process, vol. 9, no. 3, pp. 501 –504.

[4]. Jie Liu, Yuan Li, XiaodongXie (2016) ‘Hardware-Oriented Adaptive Multi-resolution Motion Estimation Algorithm and Its VLSI Architecture’, vol.78, no.1, pp. 4799-5341.