IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)
Volume 4, Issue 6, Ver. III (Nov - Dec. 2014), PP 50-55
e-ISSN: 2319 – 4200, p-ISSN No. : 2319 – 4197
www.iosrjournals.org

Area Efficient Realization of Error Detection and Data Recovery Architecture in Motion Estimation

T.Swetha Priya[1], K.Prabhakara Rao[2]
[1,2] *Department of ECE, B.V.Raju Institute of Technology, Narsapur, Telangana State, INDIA*

Abstract:. *This paper proposes a built-in self-detection and correction (BISDC) architecture for motion estimation computing arrays(MECAs).Based on the error detection & correction concepts of bi-residue codes, any single error in each processing element in an MECA can be effectively detected and corrected online using the proposed BISD and built-in self-correction circuits. Performance analysis and evaluation demonstrate that the proposed BISDC architecture performs well in error detection and correction with minor area.*

Keywords: *Built-in self-detection and correction (BISDC), Block Matching Motion Estimation (BME) algorithm, Digital Video Compression, Motion estimation computing arrays (MECA), video coding standard*

I. Introduction

The Motion Estimation Computing Array is used in Video Encoding applications to calculate the best motion between the current frame and reference frames. The MECA is in decoding application occupies large amount of area and timing penalty. By introducing the concept of Built-in Self-test technique the area overhead is increased in less amount of area. Video data needs to be compressed before storage and transmission, complex algorithms are required to eliminate the redundancy, extracting the redundant information. Motion Estimation (ME) is the process of creating motion vectors to track the motion of objects within video footage. It is an essential part of many compression standards and is a crucial component of the H.264 video compression standard. Motion estimation is the technique of finding a suitable Motion Vector (MV) that best describes the movement of a set of pixels from its original position within one frame to its new positions in the subsequent frame. Encoding just the motion vector for the set of pixels requires significantly less bits than what is required to encode the entire set of pixels, while still retaining enough information to reproduce the original video sequence.

One of the main design goals is to reduce the computational complexity and power consumptions, without sacrificing image quality. The simplest and most effective method of motion estimation is to exhaustively compare each NxN macro block of the current frame with all the candidate blocks in the search window defined with in the previous processed frame and find the best matching position with the lowest distortion. This is called Full Search Block Matching algorithm (FSBM). Fig.1 gives details on motion estimation we need to describe briefly how a video sequence is organized.

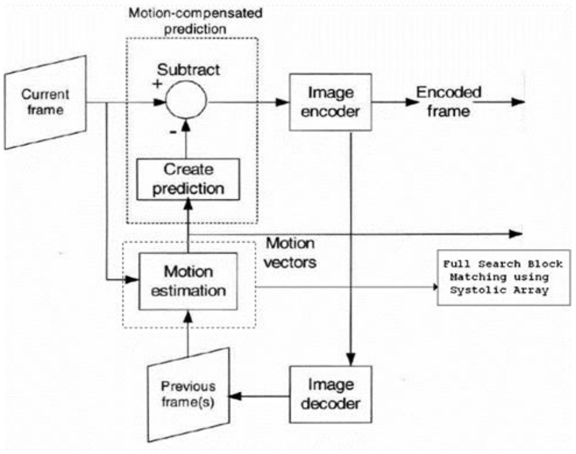


Fig. 1 Video Encoding System

A scene usually has at least three seconds. A movie in the cinema is shown as a sequence of still pictures, at a rate of 24 frames per second. The name motion picture comes from the fact that a video, once encoded, is nothing but a sequence of still pictures that are shown at a reasonably high frequency. That gives the viewer the illusion that it is in fact a continuous animation. Each picture is composed of a number of pixels or peals (picture elements). A video frame has its pixels grouped in 8×8 blocks. The blocks are then grouped in macro blocks (MB), which are composed of 4 luminance blocks each (plus equivalent chrominance blocks). Macro blocks are then organized in "groups of blocks" (GOBs) which are grouped in pictures (or in layers and then pictures). Pictures are further grouped in scenes, as described above, and we can consider scenes grouped as movies. Motion estimation is often performed in the macro block domain. For simplicity' sake we'll refer to the macro blocks as blocks, but we shall remember that most often the macro block domain is the one in use for motion estimation. To meet real-time processing needs, several motion vector search strategies and hardware designs have been proposed. These primarily focus on reducing the number of Sum-of-Absolute-Difference (SAD) operations at the cost of controller complexity.

II. Background Study

2.1 Digital Video Compression

Video compression is achieved on two separate fronts by eliminating spatial redundancies and temporal redundancies from video signals. Removing spatial redundancies involves the task of removing video information that is consistently repeated within certain areas of a single frame. For example a frame shot of a blue sky will have a consistent shade of blue across the entire frame. This information can be compressed through the use of various discreet cosine transformations that map a given image in terms of its light or color intensities. This paves the way for spatial compression by only capturing the distinct intensities, instead of the spread of intensities over the entire frame.

Compression through the removal of temporal redundancies involves compressing information that is repeated over a given sequence of frames. For example the objects in the background of a news anchor being filmed are not likely to change over the course of the footage. This redundancy can be taken advantage of to reduce the storage space required for the footage. When the background does happen to move, recording only the motion of objects over consecutive frames in the form of motion vectors can still achieve significant amounts of compression. Consequently, the motion estimation process is the process of deriving a suitable Motion Vector (MV) that best describes the spatial movement of objects from one frame to the next.

2.2 Block Matching Motion Estimation

Block Matching Motion Estimation (BME) algorithm treats a frame as being composed of many individual sub-frame blocks, known as macro Blocks. Motion vectors are then used to encode the motion of the macro Blocks through frames of video via a frame by frame matching process. When a frame is brought into the encoder for compression, it is referred to as the current frame. It is the goal of the BME unit to describe the motion of the macro Blocks within the current frame relative to a set of reference frames. The reference frames may be previous or future frames relative to the current frame. Each reference frame is also divided into a set of sub frame blocks, which are equal to the size of the macro Blocks. These blocks are referred to as reference Blocks. The BME algorithm will scan several candidate reference Blocks within a reference frame to find the best match to a macro Block. Once the best reference Block is found a motion vector is then calculated to record the spatial displacement of the macro Block relative to the matching reference Block, as shown in Fig.2.

2.3 Search Windows

When searching a reference frame for possible macro Block matches, the entire reference frame is not searched. Instead the search is restricted within a search window. Search windows in most H.264 implementations have a size of 48-pixel (rows) x 63-pixel (columns). In this thesis, we use the same 48x63 search window size. This window consists of a vertical search range of [-16, +16] and a horizontal search range of [-24, +23] pixels as illustrated in Fig. 3

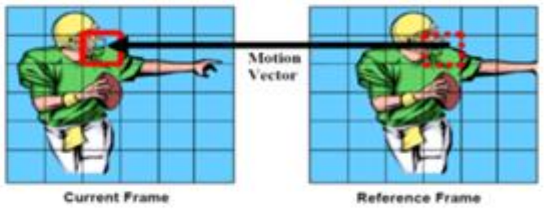


Fig. 2. Block Matching between Current & reference frames

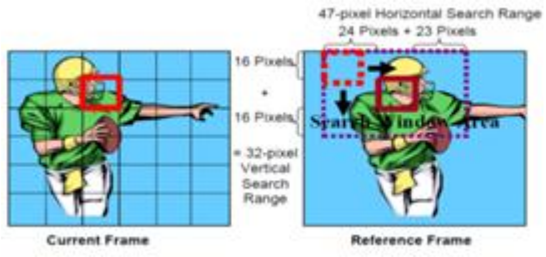


Fig. 3. Search Window size Definition

In the Fig.3, the dashed large rectangle in the reference frame represents the 48x63 search window area. The dashed square in the top left corner of the search window represents the first of the 1584 possible candidate 16x16 reference Blocks. Each subsequent reference Block is offset by either one pixel row or one pixel column from its predecessor while the entire search window area is covered by the overlapping candidate reference Blocks. Note that the original 16x16 macro Block is positioned at the centre of the search window. In order to compare it to every candidate reference Block within the search window, the macro Block has a maximum displacement of 24 pixels to the left, 23 pixels to the right, 16 pixels up, and 16 pixels down from its original position – resulting in a horizontal search range of [-24, +23] and a vertical search range of [-16, +16].

III. Proposed System & Design Approach

3.1 BISDC Architecture

In this proposal the Built-in Self-test Technique (BIST) is included in the MECA and in each of Processing Element in MECA. Thus by introducing the BIST Concept the testing is done internally without Connecting outside testing Requirements. So the area required is also reduces. And in this Project the Errors in MECA are Calculated and the Concept of Diagnoses i.e. Self-Detect and Self Repair Concepts are introduced.

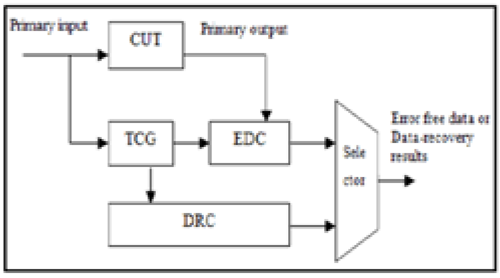


Fig. 4. Conceptual view of the proposed BISDC architecture

Fig.4 shows the conceptual view of the proposed BISDC scheme, which comprises two major circuit designs, i.e. error detection circuit (EDC) and data recovery circuit (DRC), to detect errors and recover the corresponding data in a specific CUT. The test code generator (TCG) in Fig. 1 utilizes the concepts of RQ code to generate the corresponding test codes for error detection and data recovery. In other words, the test codes from TCG and the primary output from CUT are delivered to EDC to determine whether the CUT has errors. DRC is in charge of recovering data from TCG. Additionally, a selector is enabled to export error-free data or data-recovery results. Importantly, an array based computing structure, such as ME, discrete cosine transform (DCT), iterative logic array (ILA), and finite impulse filter (FIR), is feasible for the proposed BISDC scheme to detect errors and recover the corresponding data.

This work adopts the systolic ME [19] as a CUT to demonstrate the feasibility of the proposed BISDC architecture. A ME consists of many PEs incorporated in a 1-D or 2-D array for video encoding applications. A PE generally consists of two ADDs (i.e. an 8-b ADD and a 12-b ADD) and an accumulator (ACC). Next, the 8-b ADD (a pixel has 8-b data) is used to estimate the addition of the current pixel (Cur_pixel) and reference pixel (Ref_pixel). Additionally, a 12-b ADD and an ACC are required to accumulate the results from the 8-b ADD in order to determine the sum of absolute difference (SAD) value for video encoding applications. Notably, some registers and latches may exist in ME to complete the data shift and storage. Fig.5 shows the proposed BISDC circuit design for a specific PEi of a ME.

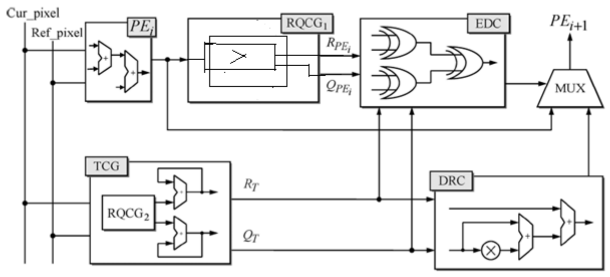


Fig. 5. A specific PEi testing processes of the Proposed BISDC architecture

The self-detection and self-correction operations (Fig.5) are simply described as follows. First, the input data of Cur. Pixel and Ref.pixel for a specific PEi in the MECA are sent to the test code generator (TCG) to generate the corresponding test codes. Second, the test codes from the TCG and output data from the specific PEi are detected and verified in Error Detection Circuit (EDC) to determine whether the specific PEi has an error. In other words, the self-detection capability uses detection of the error. Third, the Data recovery circuit (DRC) comes to play for error correction. Finally, the error correction data from DRC, or error-free data from the EDC, are passed to the next specific PEi+1 for subsequent testing. Processing element calculates the sum of absolute differences (SAD) between current pixels and reference pixels. Generally, a PE is made up of two adders (an 8-bit adder and a 12-bit adder and accumulator and formulated as:

$$\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left|C(i,j)-r(i-j)\right|$$

3.2 PE Array Architecture

PE is a module that calculates the absolute difference between the pixel of the reference block and the pixel of the current block. Fig.6 shows the architecture of PE Array 4x4. To enable the reference data shifting to top, bottom, right or left in PE Array 4x4, each PE is connected to the PE of top, bottom, and right or left one. This structure generates SAD 4x1 by accumulating the absolute difference of each PE. Furthermore, SAD 4x4 is generated by accumulating generated SAD 4x1.

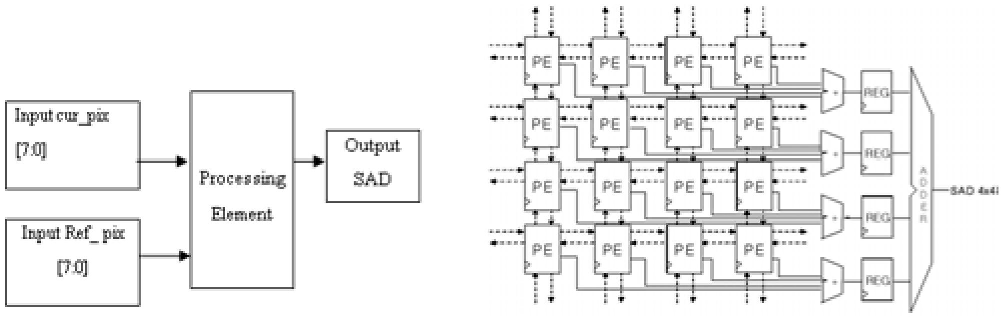
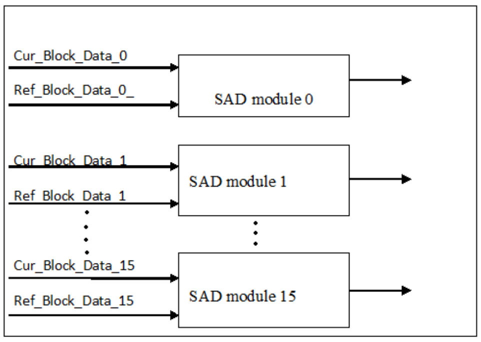


Fig. 6. Processing Element Module Schematic Diagram & Array 4x4 architecture

3.3 SAD Module & RQ Code Generation

There are 16 SAD modules in the architecture, where each one is in charge of the SAD computation of one primitive 4x4 sub-block in parallel, as shown in fig.7. In the SAD module, there are 16 absolute difference computing unit processing the 16 pair of pixels in parallel, and then the 16 absolute residues are fed into the adder unit to get one 4x4SAD.



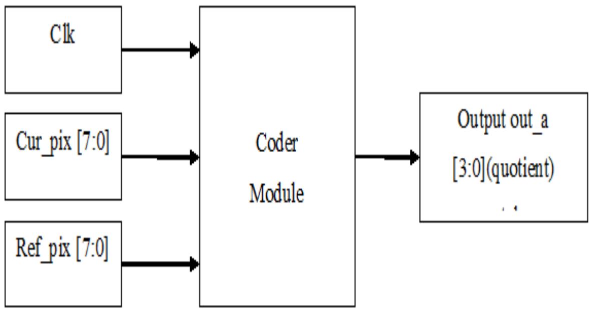


Fig. 7 Basic Structure of SAD Modules **Fig. 8** RQCG Schematic Diagram

Coding approaches such as parity code, Berger code, and residue code have been considered for design applications to detect circuit errors. Residue code is generally separable arithmetic codes by estimating a residue for data and appending it to data. Error detection logic for operations is typically derived by a separate residue code, making the detection logic is simple and easily implemented. Error detection logic for operations is typically derived using a separate residue code such that detection logic is simply and easily implemented. However, only a bit error can be detected based on the residue code. Additionally, an error can't be recovered effectively by using the residue codes.

Therefore, this work presents a quotient code as shown in Fig.8, which is derived from the residue code, to assist the residue code in detecting multiple errors and recovering errors. The corresponding circuit design of the RQCG is easily realized by using the simple adders (ADDs). Namely, the RQ code can be generated with a low complexity and little hardware cost. The mathematical model of RQ code is simply described as follows. Assume that binary data X is expressed as

$$X=b_{n-1}b_{n-2}.....b_2b_1b_0=\sum_{j=0}^{n-1} bj2j$$

IV. Results

Two inputs a, b i.e. current and reference pixels each of 8-bit length and one output result also 8-bit length. The behavioral simulation waveform for the Processing Element is shown in Fig. 9. The behavioral simulation waveform for the Coder as a three inputs clk, cur_pix, ref_pix and each of 8-bit length and output consists two coders i.e. out_a, out_b it consists of 4-bit length. The input of a coder is clk, current and reference pixels are shown in Fig 10. The behavioral simulation waveform for the Modulus Division code as a two inputs i.e. dividend, divider each of 8-bit length and it has one output it as a modulus 4 –bit of length is shown in Fig.11. The behavioral simulation results for Top Module i.e., BISDC Architecture for MECA with inputs of clk, cur_pixel[7:0], ref_pixel[7:0], PE Output, RQCG Output, TCG Output are given in Fig. 12.

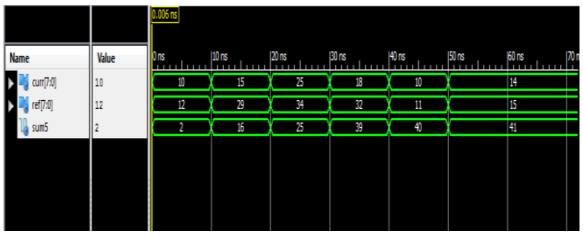


Fig. 9. Simulation Waveform for Processing Element

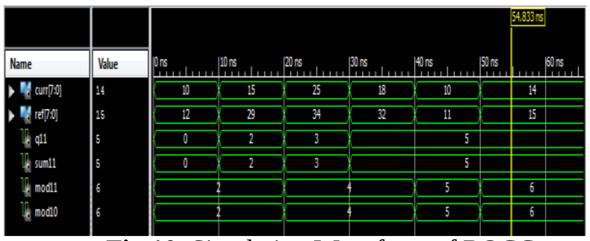


Fig.10. Simulation Waveform of RQCG

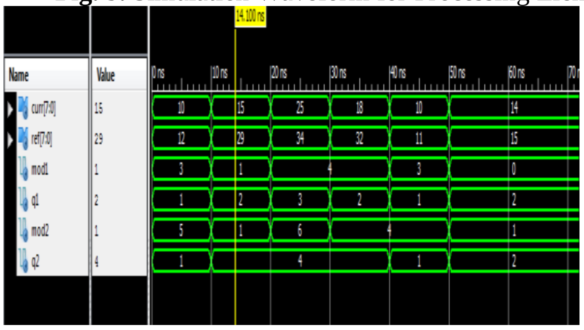


Fig. 11. Simulation Waveform of Modulus

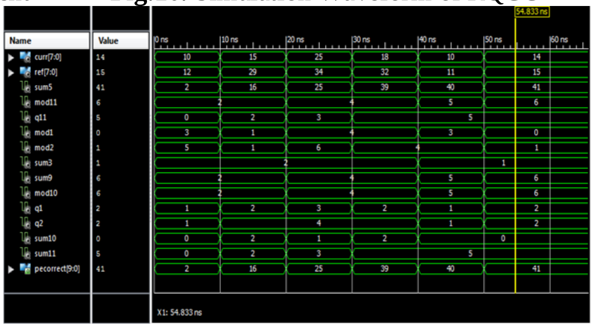


Fig.12 Top Module Simulation Result of BISDC Architecture

V. Conclusion

This project proposes BISDC architecture for self-detection and self-correction of errors of PEs in an ME. Based on the RQ code, a RQCG-based TCG design is developed to generate the corresponding test codes to detect errors and recover data. Performance evaluation reveals that the proposed BISDC architecture effectively achieves self-detection and self-correction capabilities with minimal area (LUT). The Functional-simulation has been successfully carried out with the results matching with expected ones. The design functional verification and Synthesis is done by using Xilinx-ISE 12.3 Version The input to the MECA is taken in binary format. By Adding the Image to Bit Converter input to MECA is directly in the form of frames, timing required for Motion Estimation will be reduced. The input to the MECA is 8-bit data. It also can be extended to higher volume of data. But the Calculation time required is also high.

References

[1]. Advanced Video Coding for Generic Audiovisual Services, ISO/IEC 14496-10:2005 (E), Mar. 2005, ITU-T Rec. H.264 (E).

[2]. Information Technology-Coding of Audio-Visual Objects—Part 2: Visual, ISO/IEC 14 496-2, 1999.

[3]. Y. W. Huang, B. Y. Hsieh, S. Y. Chien, S. Y. Ma, and L. G. Chen, "Analysis and complexity reduction of multiple reference frames motion estimation in H.264/AVC," IEEE Trans. Circuits Syst. Video Technol., vol. 16, no. 4, pp. 507–522, Apr. 2006.

[4]. C. Y. Chen, S. Y. Chien, Y. W. Huang, T. C. Chen, T. C. Wang, and L. G. Chen, "Analysis and architecture design of variable block-size motion estimation for H.264/AVC," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 53, no. 3, pp. 578–593, Mar. 2006.

[5]. T. H. Wu, Y. L. Tsai, and S. J. Chang, "An efficient design-for-testability scheme for motion estimation in H.264/AVC," in Proc. Int. Symp VLSI Design, Autom. Test, Apr. 2007, pp. 1–4.

[6]. M. Y. Dong, S. H. Yang, and S. K. Lu, "Design-for-testability techniques for motion estimation computing arrays," in Proc. Int. Conf. Commun., Circuits Syst., May 2008, pp. 1188–1191.

[7]. Y. S. Huang, C. J. Yang, and C. L. Hsu, "C-testable motion estimation design for video coding systems," J. Electron. Sci. Technol., vol. 7, no.4, pp. 370–374, Dec. 2009.