RESEARCH ARTICLE OPEN ACCESS

Improved Error Detection and Data Recovery Architecture for Motion Estimation Testing Applications

K.Sarada[1], Modukuri Kishore Babu[2]
[1]Department of ECE, Malineni Perumalu Educational Society, Puladigunta, Guntur(dt)., A. P, India.
[2]Assistant.proffesor Department of ECE, Malineni Perumalu Educational Society, Puladigunta, Guntur (dt)., A.P, India.

Abstract
Given the critical role of motion estimation (ME) in a video coder, testing such a module is of priority concern. While focusing on the testing of ME in a video coding system, this work presents an error detection and data recovery (EDDR) design, based on the proposed residue-and-quotient (RQ) code, to embed into ME for video coding testing applications. An error in processing elements (PEs), i.e. key components of a ME, can be detected and recovered effectively by using the proposed EDDR design. Experimental results indicate that the proposed EDDR design for ME testing can detect errors and recover data with an acceptable area overhead and timing penalty. Importantly, the proposed EDDR design performs satisfactorily in terms of throughput and reliability for ME testing applications.

Keywords: Area overhead, data recovery, error detection, motion estimation, reliability, proposed residue-and-quotient (RQ) code.

I. INTRODUCTION

The new Joint Video Team (JVT) video coding standard has garnered increased attention recently. Generally, motion estimation computing array (MECA) performs up to 50% of computations in the entire video coding system, and is typically considered the computationally most important part of video coding systems. Thus, integrating the MECA into a system-on-chip (SOC) design has become increasingly important for video coding applications.

Although advances in VLSI technology allow integration of a large number of processing elements (PEs) in an MECA into an SOC, this increases the logic-per-pin ratio, thereby significantly decreasing the efficiency of chip logic testing. For a commercial chip, a video coding system must introduce design for testability (DFT), especially in an MECA.

The objective of DFT is to increase the ease with which a device can be tested to guarantee high system reliability. Many DFT approaches have been developed.

These approaches can be divided into three categories: ad hoc (problem oriented), structured, and built-in self-test (BIST). Among these techniques, BIST has an obvious advantage in that expensive test equipment is not needed and tests are low cost.

This paper develops a built-in self-detection and correction (BISDC) architecture for motion estimation computing arrays(MECAs).Based on the error detection & correction concepts of biresidue 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.

Section II describes the mathematical model of proposed RQ code and the corresponding circuit design of the RQ code generator (RQCG). Section III then introduces the proposed EDDR architecture, fault model definition, and test method. Next, Section IV evaluates the performance in area overhead, timing penalty, throughput and reliability analysis to demonstrate the feasibility of the proposed EDDR architecture for ME testing applications. Conclusions are finally drawn in Section V.

II. BACKGROUND

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.

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. Since compression through removing spatial redundancies does not involve the use of motion estimation, this topic is not examined further.

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.

The spatial and temporal compression techniques discussed above have been widely implemented in former compression standards such as MPEG 2 (developed by the Motion Picture Experts Group committee). At present, the same two fundamental techniques have been enhanced and optimized to form the new standard used in H.264 video compression. The H.264 standard was jointly formed by the International Telecommunications Union – Telecommunications Standardization Sector (ITU – T) Video Coding Experts Group (VCEG) and the International Organization for Standardization(ISO) MPEG committee. MPEG 2 had compression ratios of between 20:1 and 30:1, the new H.264 standard can achieve compression ratios as high as 50:1 and 60:1 and achieve better video quality.

Among many other new features and enhancements, the most notable features of the H.264 standard for this work are its ability to achieve a finer granularity of motion estimation and its ability to capture periodic motion.

Block Matching Motion Estimation

Several different algorithms derived from various theories, including object-oriented tracking, exist to perform motion estimation. Among them, one of the most popular algorithms is the Block Matching Motion Estimation (BME) algorithm. BME 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 Figure 1

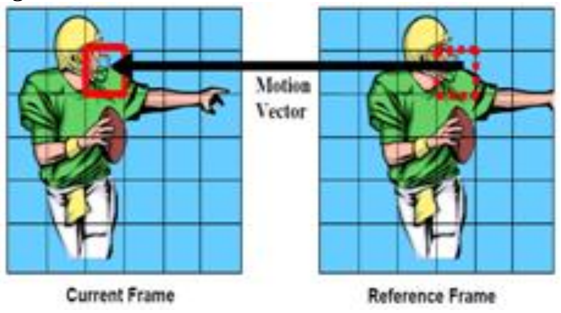

Fig 1: Block Matching between Current & reference frames

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 Figure 2

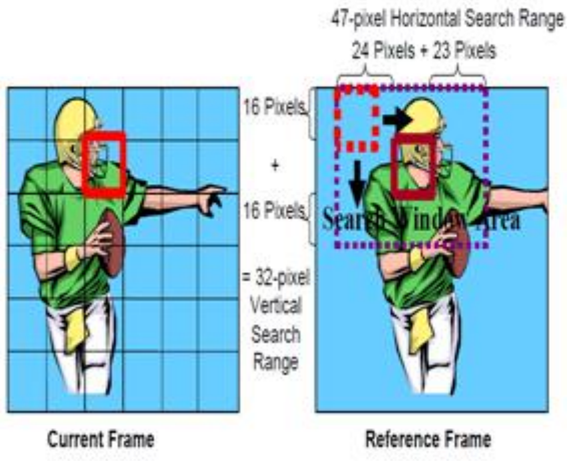

Fig 2: Search Window size Definition

In the Fig 2, 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

In this paper 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. The area results are compared with the MECA without BIST technique.

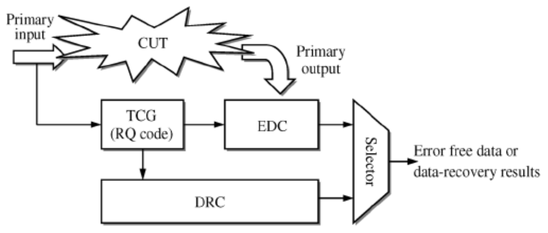

Fig3: Block Diagram of Proposed MECA BIST

Signals TC1and TC2 are utilized to select data paths from Cur.Pixel and Ref.pixel, respectively. The output of a specific PE_i can be delivered to a detector for detecting errors using the DC1 signal. Moreover, the selector circuit is controlled by signals SC1 and SC2 that receive data from a specific PE_{i+1}, and then export these data to the next specific PE_i or syndrome analysis and corrector (SAC) for error correction.

Based on the concepts of BIST and bi-residue codes, this paper presents a built-in self-detection/correction (BISDC) architecture that effectively self-detects and self-corrects PE errors in an MECA. Notably, any array-based computing structure, such as the discrete cosine transform (DCT), iterative logic array (ILA), and finite-impulse filter (FIR), is suitable for the proposed method to detect and correct errors based on bi-residue codes.

Fault Model

The PEs are important building blocks and are connected in a regular manner to construct an MECA. Generally, PEs are surrounded by sets of adders and accumulators that determine how data flows through them.

Thus, PEs can be considered the class of circuits called ILAs, whose testing assignment can be easily achieved using the fault model called as cell fault model (CFM). The use of the CFM is currently of considerable interest due to the rapid growth in the use of high-level synthesis and the parallel increase in complexity and density of ICs. Using the CFM allows tests to be independent of the adopted synthesis tool and vendor library. Arithmetic modules, like adders (the primary element in a PE), due to their regularity, are designed in a very dense configuration.

Moreover, the use of a relatively more comprehensive fault model, the single stuck-at (SSA) model, is required to cover actual failures in the interconnect data bus between PEs. The SSA fault is a well-known structural fault model that assumes faults cause a line in the circuit to behave as if it were permanently at logic “0” [stuck-at 0 (SA0)] or logic “1” [stuck-at 1 (SA1)]. The SSA fault in MECA architecture can result in errors in computed SAD values. This paper refers to this as a distorted computational error; its magnitude is e = SAD’-SAD. Where SAD’ is the computed SAD value with an SSA fault.

IV. PROPOSED RQ CODE GENERATION

Generally RQ code is useful for to identify the errors and to rectify those errors. In previous technique, finding the quotient and reminder becomes difficult and error may be generated by the TCG, to overcome that problem we are proposing a new technique in this paper. In the proposed technique we are finding the residue & quotient by fixing modulus value as ‘2’.In this technique the quotient and residue values for a pixel value can be obtained by simple operation and is explained below.

Step-1 :Represent the input pixel value in the binary format.

Step-2: Perform the single bit shift right operation and place bit ‘0’ in the MSB position of the above result and assign the integer equivalent of result of the above operation to variable Quotient.

Step-3: Verify the LSB bit in the binary format of the input pixel either it is Zero or One and assign the integer equivalent of the above result to the variable Reminder.

Step-4 : Finally Quotient and Reminder values are obtained and those values can be used for the comparison between TCG output and PE output

Step-5 : Find the quotient and reminder in TCG by performing the subtraction between current and reference pixels before the RQ code generation and add those values to previously calculated values.

Numerical Example:

Let assume the pixel value as 36.We need 8-bits to represent the above pixel value.
Binary representation of it is: **00100100**
Output of the Shift right operation is : **00010010**
Finally output of step-2 is: **00010010**
Quotient value is :18
Output of step-3 is: **00000000**
Reminder value is :**0**

V. RESULTS AND DISCUSSIONS

Absolute Difference

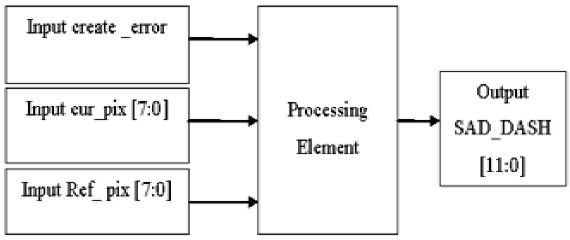

Fig 4: Absolute Result Schematic Diagram

Description: Absolute Difference as a 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 Absolute Difference is shown in Fig 4. In the Fig 4 shows the two inputs with 8-bit length are ‘a’ (current Pixels) and ‘b’ (reference pixels) and 8-bit output result.

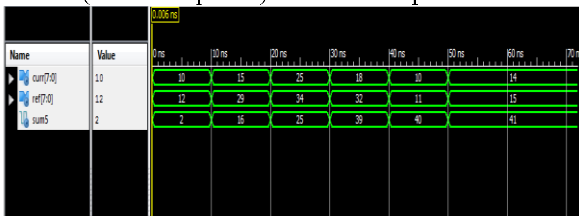

Fig 5: Simulation Waveform for Absolute difference

Processing Element:

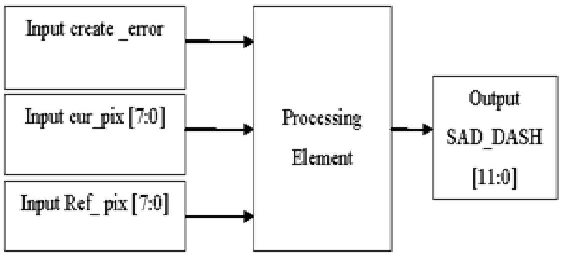

Fig6 Processing Element Module Schematic Diagram

Description: Processing Element as a three inputs create_error, current pixel, reference pixel each of 8-bit and output is a sad_dash as a 12-bit data. The input of PE is a current pixel and reference pixels are shown in Fig 6. The behavioral simulation waveform for the Processing Element is shown in Fig 7. In the Fig 7 the two inputs are 8-bit length are ‘a’ (current Pixels) and ‘b’ (reference pixels) input and 12-bit output.

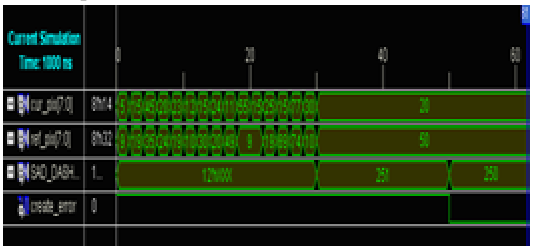

Fig 7: Simulation Waveform of Processing Element

Modulus code

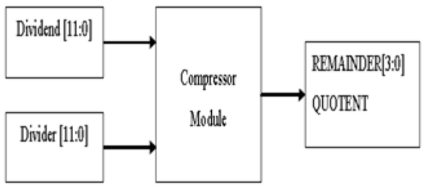

Fig 8 Modulus code Schematic Diagram

Description: Modulus code as a two inputs i.e. dividend, divider each of 12-bit length and it has one output it as a modulus 4 –bit of length is shown in Fig 8. The behavioral simulation waveform for the Modulus Division code as a two inputs i.e. dividend, divider each of 12-bit length and it has one output it as a modulus 4 –bit of length is shown in Fig 9.

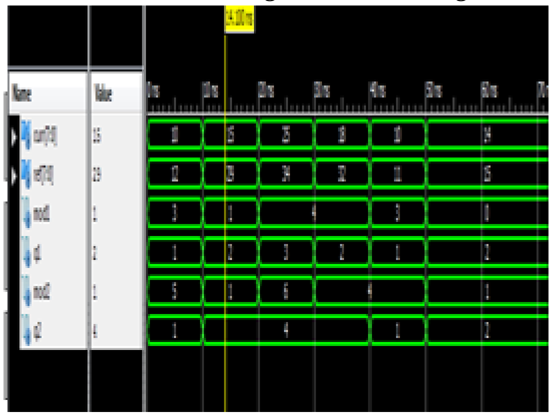

Fig 9 Simulation Waveform of Modulus

Coder module

Description: 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.

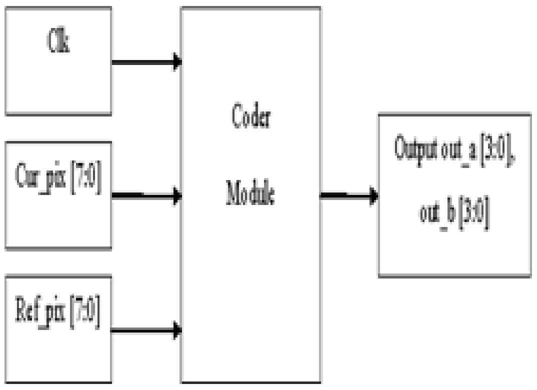

Fig 10 Coder Schematic Diagram

The input of a coder is clk, current and reference pixels are shown in Fig 10. 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 is shown in Fig 11.

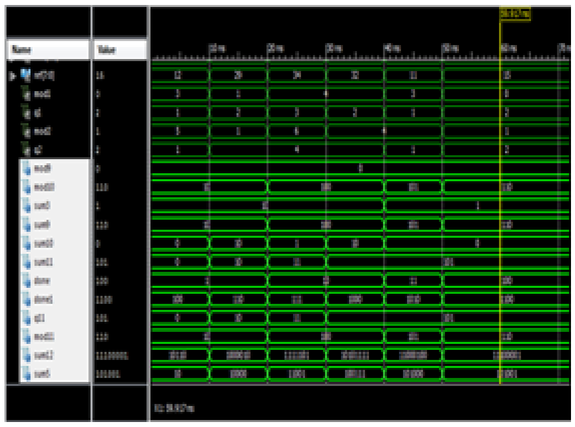

Fig 11 Simulation Waveform of Coder

Selector Module

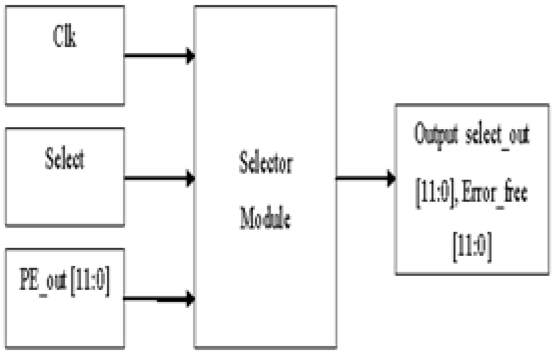

Fig 12. Selector Module Schematic diagram

Description: Selector takes the output of the PE as an input. Another input to the selector is the output of the detector. It has three inputs clk, select, PE_out. And the output is select_out, error_free each of 12-bit length is shown in Fig 12. The behavioral simulation waveform for the Selector takes the output of the PE as an input. Another input to the selector is the output of the **detector**. It has three inputs clk, select, PE_out. And the output is select_out, error_free each of 12-bit length is shown in Fig 13

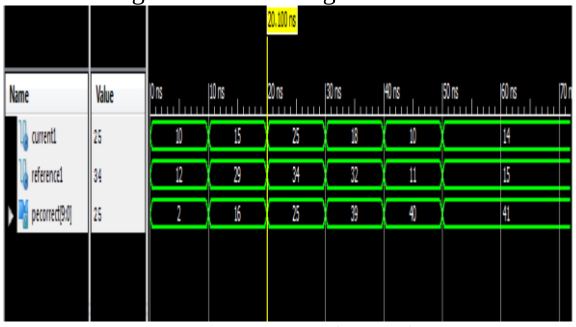

Fig 13 Simulation Waveform of Selector

Corrector Module

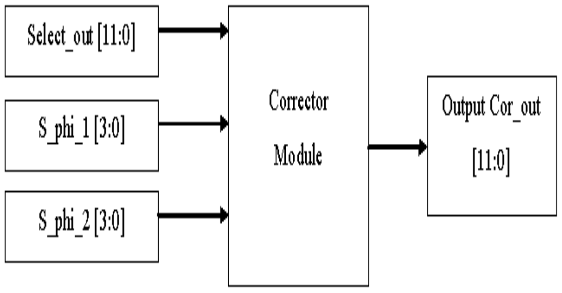

Fig 14 Corrector Module Schematic Diagram

Description: Input to the Corrector module is the output of the selector module which is SAD that needs to be corrected. It as three inputs select_out, sphi_1, sphi_2 and output as a corr_out as a12-bit length is shown in Fig 14. The behavioral simulation waveform for the Input to the Corrector module is the output of the selector module which is SAD that needs to be corrected. It as three inputs select_out, sphi_1, sphi_2 and output as a corr_out as a12-bit length is shown in Fig 15.

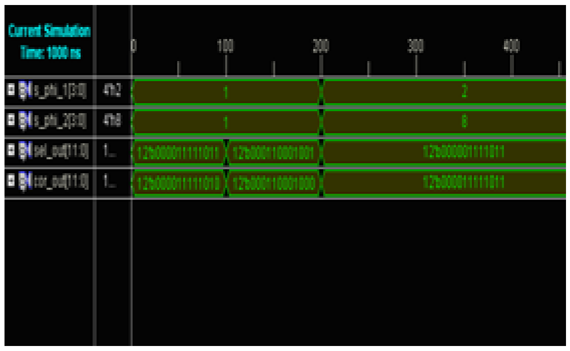

Fig 15 simulation waveform of Corrector

Top Module: (Simulation Waveform)

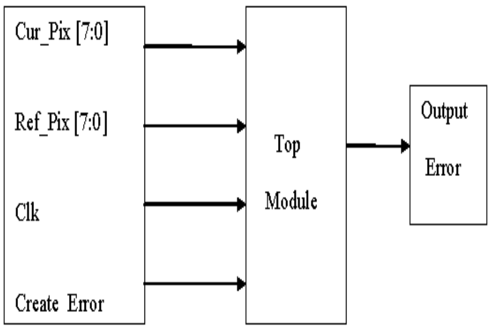

Fig 16 Top Module schematic Diagram

Description: The proposed design is developed in a top down design methodology that the code is a mixed version of both behavioral and structural. The proposed Architecture consists of basic modules like Absolute Difference, Compressor, Processing Element, Modulus Division, Coder, Selector and Corrector modules. The schematic of Top Module for BISDC Architecture for MECA is shown in Fig 16.

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], Create_error and outputs with error, with_out_error are given in Fig 16. This waveform contains signals like N (sum of total number of current pixels and reference pixels without error), N_dash_error (sum of total number of current pixels and reference pixels with error), syndrome_7 [3:0], syndrome_15 [3:0].

AREA RESULTS OF TCG
Based on Existing Division Method:

Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	126	768	16%
Number of 4 input LUTs	207	1536	13%
Number of bonded IOBs	32	124	25%

Based on Proposed Division Method;
Extended Method-I:

Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	39	768	5%
Number of 4 input LUTs	71	1536	4%
Number of bonded IOBs	30	124	24%

Extended Method-II:

Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	25	768	3%
Number of 4 input LUTs	47	1536	3%
Number of bonded IOBs	32	124	25%

COMPARISION OF TCG PERFORMANCE IN TERMS OF TIME
Based on Existing Division : 63.970ns
Based on Proposed Division
Extended Method-I : **37.678 ns**
Extended Method-II : **33.411 ns**

VI. CONCLUSION

This project proposes BISDC architecture for self-detection and self-correction of errors of PEs in an MECA. Based on the error detection correction concepts of bi residue codes, this paper presents the corresponding definitions used in designing the BISD and BISC circuits to achieve self-detection and self-correction operations. Performance evaluation reveals that the proposed BISDC architecture effectively achieves self-detection and self-correction capabilities with minimal area.

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/XST and Cadence RTL Compiler of BISDC architecture for MECA. In this project the Area obtained is 87% using Cadence RTL Compiler.

REFERENCES

[1] Chun-lung Hsu, chang-Hsin Cheng, and Yu Liu, "*Built- in self-detection/correction Architecture for Motion Estimation Computing Arrays*", IEEE Transcations on Very Large Scale Integration (VLSI) systems, VOL.18, NO.2, February 2010, pp.319-324.

[2] Thammavarapu R.N Rao, Member, IEEE, "*Biresidue Error-Correcting Codes for Computer Arithmetic*", IEEE Transactions on computers, VOL. C-19, NO. 5, May 1970, pp.398-402.

[3] Meihua GU, Ningmei YU, Lei ZHU, Wenhua JIA, "*High Throughput and Cost Efficient VLSI Architecture of Integer Motion Estimation for H.264/AVC*", Journal of Computational Information Systems 7:4 (2011), pp.1310-1318.

[4] 4. Zhong-Li He, Chi-Ying Tsui, Member, IEEE, Kai-Keung Chan, and Ming L. Liou, Fellow, IEEE, "*Low-Power VLSI Design for Motion Estimation Using Adaptive Pixel Truncation*", IEEE Transactions on circuits and systems for video technology, VOL.10, NO.5, August 2000, pp.669- 677.

[5] R. J. Higgs and J. F. Humphreys, "*Two-error-location for quadratic residue codes*," Proc. Inst. Electr. Eng. Commun, vol. 149, no. 3, Jun.2002, pp.129–131.

[6] Charles E.Stroud, "*A Designer's Guide to Built in Self Test*", Kluwer Academic Publishes, 2002.

[7] Laung-Terng Wang, Cheng-Wen Wu, Xiaoqing Wen,*" VLSI TEST PRINCIPLES AND ARCHITECTURES"*, Elsevier, 2006.