IOSR Journal of Electronics and Communication Engineering (IOSR-JECE)
e-ISSN: 2278-2834,p- ISSN: 2278-8735.Volume 8, Issue 3 (Nov. - Dec. 2013), PP 01-08
www.iosrjournals.org

Design of EDDR Architecture for Motion Estimation Testing Applications

[1]Meena Nagaraju, [2]Dr. Giri Babu Kande
[1]*PG Student (M.Tech VLSI), Dept. Of ECE, Vasireddy Venkatadri Ins. Tech., Nambur, Guntur, AP, India*
[2] *Professor & Head, Dept. Of ECE, Vasireddy Venkatadri Ins. Tech., Nambur, Guntur, AP, India*

Abstract: *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. In this Paper the Built-in Self test Technique (BIST) is included in the MECA and in each of Processing Element in MECA is tested using residue codes .the quotient and remainder was cross checked across the processing element and test code generator. further the residue code is replaced with the boolen logic adder in trst code generator in order to reduce the area of the circuit.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.*

Keywords: *Data recovery, error detection, motion estimation, reliability, 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 project 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.

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. 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.

Fig.1 shows the corresponding BISDC implementation. Signals TC1and TC2 are utilized to select data paths from Cur. Pixel and Ref.pixel, respectively. The output of a specific PEi 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 PEi+1, and then export these data to the next specific PEi or syndrome analysis and corrector (SAC) for error correction.

Based on the concepts of BIST and biresidue 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 biresidue codes.

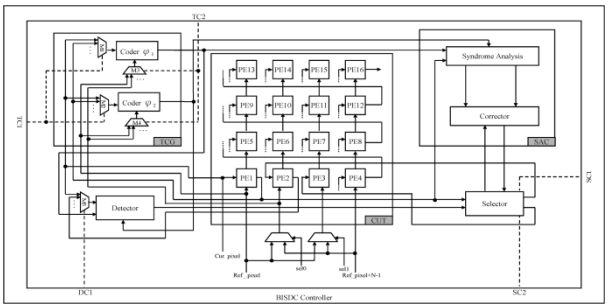


Figure 1 Block Diagram of Proposed MECA BISDC

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 an 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.

I. Motion Estimation Array

Generally, motion estimation computing array (MECA) performs up to 50% of computations in the entire video coding system (VCS).In VCS, 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 .In particular ME can consist of over 40% of the total computation.

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.

A standard movie, which is also known as motion picture, can be defined as a sequence of several scenes. A scene is then defined as a sequence of several seconds of motion recorded without interruption. 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. Similarly, a TV broadcast consists of a transmission of 30 frames per second (NTSC, and some flavors of PAL, such as PAL-M), 25 frames per second (PAL, SECAM) or anything from 5 to 30 frames per second for typical videos in the Internet.

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 frame is shown for one small fraction of a second, more precisely 1/ k seconds, where k is the number of frames per second.

Coming back to the definition of a scene, where the frames are captured without interruption, one can expect consecutive frames to be quite similar to one another, as very little time is allowed until the next frame is to be captured. With all this in mind we can finally conclude that each scene is composed of at least 3 × k frames (since a scene is at least 3 seconds long). In the NTSC case, for example, that means that a movie is composed of a sequence of various segments (scenes) each of which has at least 90 frames similar to one another.

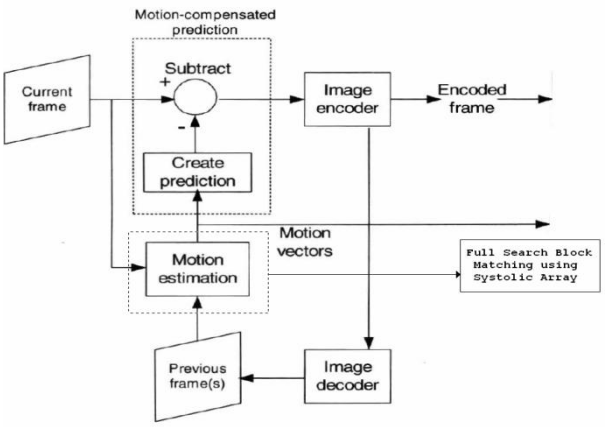


Figure 2 Video Encoding System

Fig 2 gives details on motion estimation we need to describe briefly how a video sequence is organized. As mentioned earlier a video is composed of a number of pictures. 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).

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.

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 3.

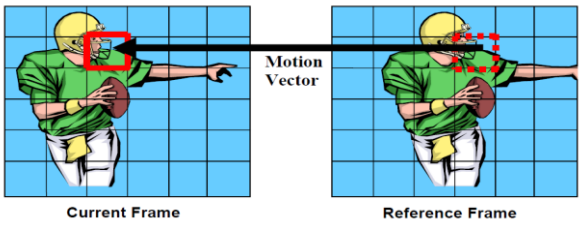


Figure 3 Block Matching between Current & reference frames

II. Implementation of The Proposed System

The self-detection and self correction operations (Fig.4) 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 detector and selector (DAS) circuits to determine whether the specific PEi has an error. In other words, the self-detection capability uses the detector circuit in DAS. Third, the selector circuit

in DAS delivers the error signal to SAC for error correction. Finally, the error correction data from SAC, or error-free data from the selector circuit in DAS, are passed to the next specific PEi+1 for subsequent testing.

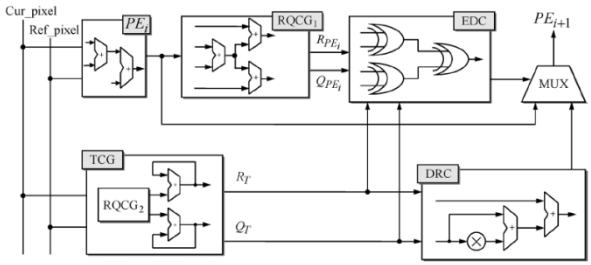


Figure 4 block diagram of the self-detection/correction operations

Processing Element (PE)

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.

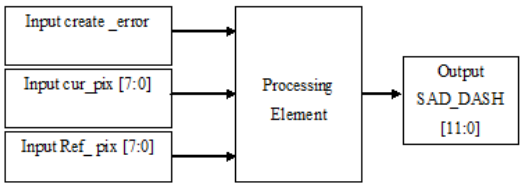


Figure 5 Processing Element Module Schematic Diagram

The SAD is given by

$$SAD = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} | C(r, j) - r(r, j) | \quad \ldots.(1)$$

Where c (i, j) and r (i, j) are the current pixel and reference pixel. The 16 pixels absolute difference is given to adder unit to perform SAD.

CODER

The following definitions, based on the bi-residues codes, are applied to verify the feasibility of the two coders in the TCG

Definition 1:

$| N1+N2|\varphi = || N1| \varphi + |N2|\varphi| \varphi.$

Definition 2:

Let Nj= n1+n2+……. +nj.

Then

$| NJ | \varphi = || n1| \varphi + |n2| \varphi + \ldots\ldots. + | nj|\varphi|\varphi.$

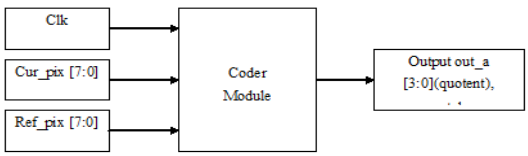


Figure 6 Coder Schematic Diagram

In the Paper we are using two Coder modules because bi residue Codes are used for calculation of SAD.And the Phi1 (φ1) and phi2 (φ2) values are selected by satisfying the conditions. i.e.

A = 2a -1 and B = 2b -1 such that GCD (a, b) = 1.

DETECTOR

Detector module will detect whether there is an error in output of PE i.e. SAD. Here the Error is calculated. The output of PE and theoretically calculated SAD will be subtracted which is given as e = SAD'-SAD

SELECTOR

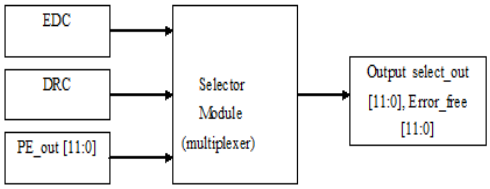


Figure 7 Selector Module Schematic diagram

Selector takes the output of the processing element as an input. Another input to the selector is the output of the detector. If the Detector block detects any error in the PE output then the Selector block will give the PE output to Syndrome Decoder to detect in which bit position there is an error and also to the Corrector block to correct the single bit error.

SYNDROME DECODER

This module decodes the syndrome values which specify the error in SAD. Syndromes can be expressed as

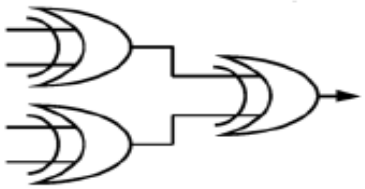


Figure 8 Error detection circuit.

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. 9 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, right or left one.

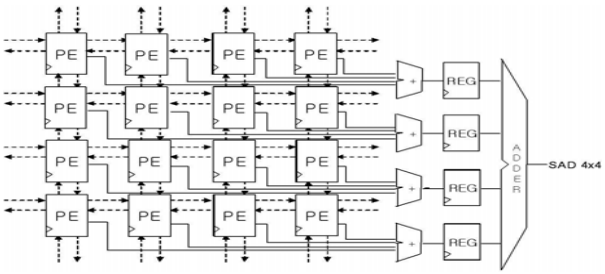


Figure 9 PE Array 4x4 architecture

This structure generates SAD 4x1 by accumulating the absolute difference of each PE. Furthermore, SAD 4x4 is generated by accumulating generated SAD 4x1.

However, because long delay will be induced by the multiple arithmetic accumulating modules which generates SAD 4x1 and SAD 4x4, registers are inserted to improve maximum clock frequency traditional PE can shift pixel data in one direction by connecting PEs of the same direction. However, the proposed PE can shift reference pixel data to top, bottom, right or left. The current pixel data is always shifting into the PE Array from top direction. The PE module is designed to be able to shift to top, bottom, right or left. The traditional PE and the proposed one are shown in Fig. 10 and Fig. 11, respectively.

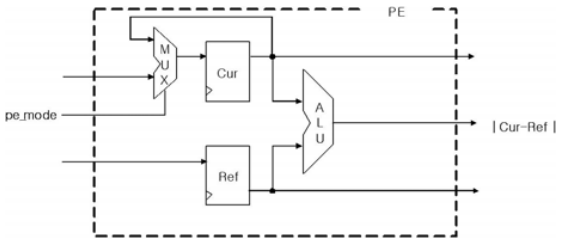


Figure 10 Traditional PE

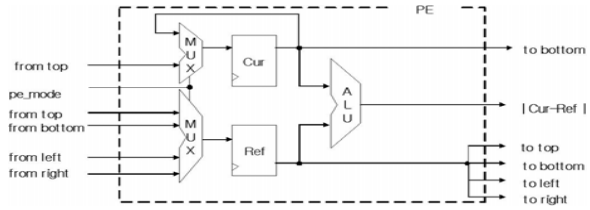


Figure 11 Proposed PE

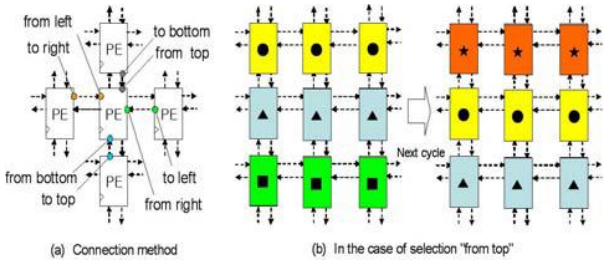


Figure 12 PE Configuration

In Fig 11 Reference pixel data need input from four directions as well as output to four directions. Therefore, four input ports and four output ports are prepared in each PE. The connection of PEs is shown in Fig.12 As shown in Fig.12(a), each PE is connected with surrounding PEs: "from top" connects to "to bottom", "from bottom" connects to "to top", "from left" connects to "to right", and "from right" connects to "to left". An example when "from top" is selected by the multiplexer is shown in Fig. 12 (b). The reference data from the output port "to bottom" of upper PEs is shifted to bottom PEs and the search position is shifted. In this way, each I/O port of PE enables the shift of the reference pixel data by selecting the input of reference pixel data using multiplexer.

TCG is an important component of the proposed EDDR architecture. Notably, TCG design is based on the ability of the RQCG circuit to generate corresponding test codes in order to detect errors and recover data. The specific in Fig. 2 estimates the absolute difference between the Cur_pixel of the search area and the Ref_pixel of the current macroblock. Thus, by utilizing PEs, SAD shown in as follows, in a macroblock with size of N × N can be evaluated:

$$\begin{aligned}\mathrm{SAD} &= \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}|X_{ij}-Y_{ij}| \\ &= \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}|(q_{xij}\cdot m+r_{xij})-(q_{yij}\cdot m+r_{yij})|\end{aligned}$$

Importantly, Xij and Yij represent the luminance pixel value of Cur_pixel and Ref_pixel, respectively. Based on the residue code, the definitions shown in (2) and (3) can be applied to facilitate generation of the RQ code (and) form TCG. Namely, the circuit design of TCG can be easily achieved (see Fig. 3) by using remainder is computed as follows in hardware design.

$$\begin{aligned}R_T &= \left|\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}(X_{ij}-Y_{ij})\right|_m \\ &= \left|\left|(X_{00}-Y_{00})\right|_m+\left|(X_{01}-Y_{01})\right|_m+\ldots\right. \\ &\quad \left.+\left|\left(X_{(N-1)(N-1)}-Y_{(N-1)(N-1)}\right)\right|_m\right|_m \\ &= \left|\left|(q_{x00}\cdot m+r_{x00})-(q_{y00}\cdot m+r_{y00})\right|_m\right. \\ &\quad +\ldots\left|\left(q_{x(N-1)(N-1)}\cdot m+r_{x(N-1)(N-1)}\right)\right. \\ &\quad \left.\left.-\left(q_{y(N-1)(N-1)}\cdot m+r_{y(N-1)(N-1)}\right)\right|_m\right|_m \\ &= \left|\left|(r_{x00}-r_{y00})\right|_m+\left|(r_{x01}-r_{y01})\right|_m+\ldots\right. \\ &\quad \left.+\left|\left(r_{x(N-1)(N-1)}-r_{y(N-1)(N-1)}\right)\right|_m\right|_m \\ &= \left|\left|r_{00}\right|_m+\left|r_{01}\right|_m+\ldots+\left|r_{(N-1)(N-1)}\right|_m\right|_m\end{aligned}$$

EDC, which is utilized to compare the outputs between TCG and in order to determine whether errors have occurred. If the values of and/or , then the errors in a specific can be detected. The EDC output is then used to generate a 0/1 signal to indicate that the tested is error-free/errancy. This work presents a mathematical statement to verify the operations of error detection. Based on the definition of the fault model, the SAD value is

influenced if either SA1 and/or SA0 errors have occurred in a specific. In other words, the SAD value is transformed to if an error occurred. Notably, the error signal is expressed as

$$e = q_e \cdot m + r_e$$

Which comply with the definition of RQ code, under the faulty case, the RQ code from of the TCG is still equal

In order to improve the shortcoming of carry ripple adder to remove the linear dependency between computation delay time and input word length, carry select adder is presented [2]. The carry select adder divides the carry ripple adder into M parts, while each part consists of a duplicated (N/M)-bit carry ripple adder pair, as illustrated in Fig. 13 as M=16 and N=4. This duplicated carry ripple adder pair is to anticipate both possible carry input values, where one carry ripple adder is calculated as carry input value is logic "0" and another carry ripple adder is calculated as carry input value is logic "1". When the actual carry input is ready, either the result of carry "0" path or the result of carry "1" path is selected by the multiplexer according to its carry input value. To anticipate both possible carry input values in advance, the start of each M part carry ripple adder pair no longer need to wait for the coming of previous carry input. As a result, each M part carry ripple adder pair in the carry select adder can compute in parallel.

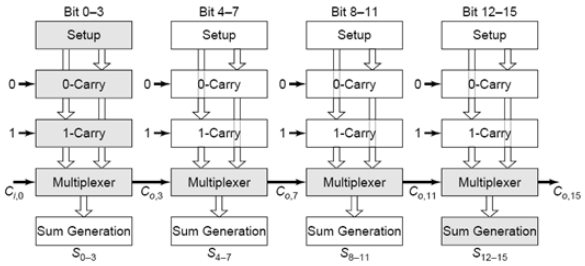
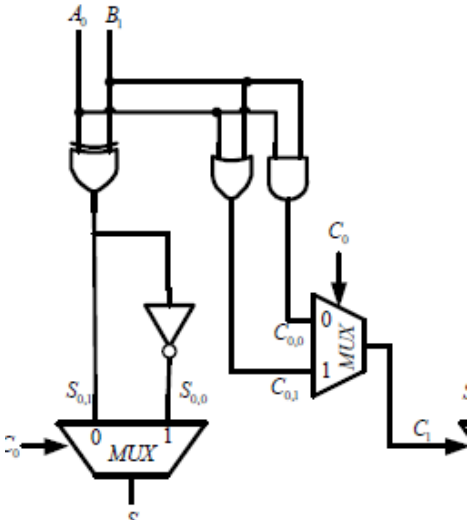


Figure 13 CSA Division Set Up

Test code generator using boolean logic:

We proposed an area-efficient carry select adder by sharing the common Boolean logic term. After Boolean simplification, we can remove the duplicated adder cells in the conventional carry select adder. Alternatively, we generate duplicate carry-out and sum signal in each single bit adder cell. By utilizing the multiplexer to select the correct output according to its previous carry-out signal, we can still preserve the original characteristics of the parallel architecture in the conventional carry select adder. In this way, the circuit area and transistor count can be greatly reduced and power delay product of the adder circuit can be also greatly lowered test code generator using boolean logic.



III. Results and Discussion

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 7.14.

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 7.16. 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]. We have undergone the simulation of several sub modules in using Xilinx 12.2i.

corrected processing element output

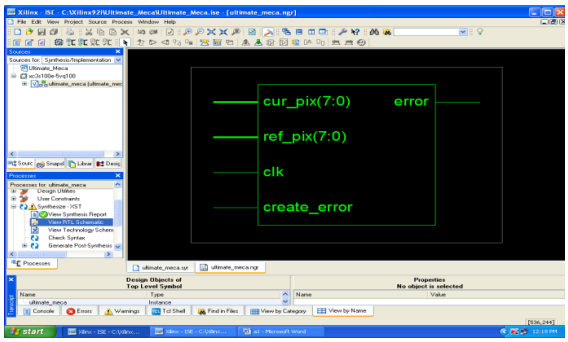


Figure 14 RTL Schematic Result of Top Module

IV. Conclusions

Using this work we 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 paper the Area obtained is 87%. 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.

Acknowledgements

The authors would like to thank the anonymous reviewers for their comments which were very helpful in improving the quality and presentation of this paper.

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] 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] 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.

[7] 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.

[8] 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.

[9] D. Li, M. Hu, and O. A. Mohamed, "Built-in self-test design of motion estimation computing array," in Proc. IEEE Northeast Workshop Circuits Syst., Jun. 2004, pp. 349–352.