ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 11, November 2014

Motion Estimation Testing Applications using Error Detection and Data Recovery Architecture

BANDI DHILLESWARA RAO [1], BONTALAKOTI PRASAD KUMAR [2]

M.Tech [Scholar], Dept. Of ECE, Swamy Vivekananda Engineering College, Bobbili, AP, India [1]

Assistant Professor, Dept. Of ECE, Swamy Vivekananda Engineering College, Bobbili, AP, India [2]

Abstract: This paper presents an error detection and data recovery (EDDR) design, based on the residue-and-quotient (RQ) code, to embed into motion estimation (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 EDDR design. The proposed EDDR design for ME testing can detect errors and recover data with an acceptable area overhead and timing penalty. The functional verification and synthesis can be done by Xilinx ISE. That is when compare to the existing design the implemented design area and timing will be reduced..

Keywords: EDDR, Motion estimation, processing elements, testing

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 Project 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

II. MOTION ESTIMATION

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.

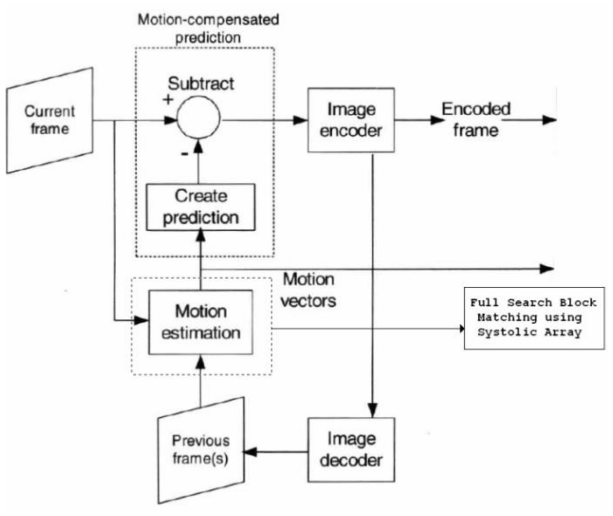

Fig.1: Video Encoding System

Fig.1 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).

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. For motion estimation the idea is that one block b of a current frame C is sought for in a previous (or future) frame R. If a block of pixels which is similar enough to block b is found in R, then instead of transmitting the whole block just a "motion vector" is transmitted.

Ideally, a given macro blocks would be sought for in the whole reference frame; however, due to the computational complexity of the motion estimation stage the search is usually limited to a pre-defined region around the macro blocks. Most often such region includes 15 or 7 pixels to all four directions in a given reference frame. The search region is often denoted by the interval [-p, p] meaning that it includes p pixels in all directions. The growing need of real-time video applications, video compression plays a vital role in achieving bandwidth efficiency for both transmission and storage and efficient motion estimation is a key factor for achieving enhanced compression ratio. However, motion estimation involves high computational complexity, causing bottleneck in the real-time applications. 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.

One of the main design goals is to reduce the computational complexity and power consumptions, without sacrificing image quality. Some algorithms and architectures succeeded in reducing power consumption and satisfied the required performance. 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).

A full search block matching process with a search range p has a search window of size (2p+N) x (2p+N) pixels and a total of (2p+1)2 candidate blocks in the reference frame for each block of the current frame. The distortion values are computed for each of the candidate blocks and its minimum value is found from the set of (2p+1)2 candidate blocks. The distortion measure is Sum of Absolute Difference for its simplicity, in which the candidate block with minimum amount of distortion is considered as the best-match. To achieve a best trade-off between the computational complexity of FSBM and degraded PSNR of motion compensated frame using faster algorithms, recently some researchers have investigated reduction of computational complexities of FSBM.

All these algorithms are not optimal in the sense that instead of exhaustive search, only some fixed positions are searched, based on the predictions of motion. Any error in motion prediction may lead to wrong motion vectors, resulting in poor peak signal-to-noise ratio (PSNR) of the motion-compensated frame.

The computationally intensive nature of FSBM and the demand of real-time processing render the VLSI implementation of FSBM is a necessity. In this paper we propose efficient and low power VLSI architecture, which has been developed to meet the speed requirement of the current video coding system.

III. BIST

With recent advances in semiconductor manufacturing technology, the production and usage of very-large-scale integration (VLSI) circuits has run into a variety of testing challenges during wafer probe, wafer sort, pre-ship screening, incoming test of chips and boards, test of assembled boards, system test, periodic maintenance, repair test, etc. Traditional test techniques that use automatic test pattern generation (ATPG) software to target single faults for digital circuit testing have become quite expensive and can no longer provide sufficiently high fault coverage for deep submicron or nanometer designs from the chip level to the board and system levels.

One approach to alleviate these testing problems is to incorporate Built-in-self test (BIST) features into a digital circuit at the design stage with logic BIST, circuits that generate test patterns and analyze the output responses of the functional circuitry are embedded in the chip or elsewhere on the same board where the chip resides. There are two general categories of BIST techniques for testing random logic:
(1) Offline BIST and
(2) Online BIST.
A general form of logic BIST techniques is

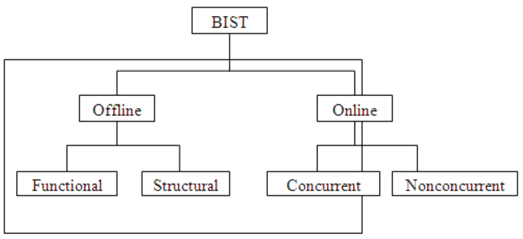

Fig.2: Types of BIST

OFFLINE BIST: It is performed when the functional circuitry is not in normal mode. This technique does not detect any real-time errors but is widely used in the industry. Logic BIST techniques for testing the functional circuitry at the system, board, or chip level to ensure product quality.
FUNCTIONAL OFFILINE BIST: It performs a test based on the functional specification of the functional circuitry and often employs a functional or high-level fault model. Normally such a test is implemented as diagnostic software or firmware.
STRUCTURAL OFFLINE BIST: It performs a test based on the structure of the functional circuitry. There are two general classes of structural offline BIST techniques:
EXTERNAL BIST: In which test pattern generation and output response analysis is done by circuitry that is separate from the functional circuitry being tested.
INTERNAL BIST: In which the functional storage elements are converted into test pattern generators and output response analyzers. Some external BIST schemes test sequential logic directly by applying test patterns at the inputs and analyzing the responses at its outputs. Such techniques are often used for board-level and system level self-test. The BIST schemes discussed here all assume that the functional storage elements of the circuit are converted into a scan chain or multiple scan chains for combinational circuit testing.

ONLINE BIST: It is performed when the functional circuitry is in normal operational mode. It can be done either concurrently or nonconcurrently.
concurrent online bist: Testing is conducted simultaneously during normal functional operation. The functional circuitry is usually implemented with coding techniques or with duplication and comparison. When an intermittent or transient error is detected, the system will correct the error on the spot, rollback to its previously stored system states, and repeat the operation, or generate an interrupt signal for repeated failures.
nonconcurrent online bist: testing is performed when the functional circuitry is in idle mode. This is often accomplished by executing diagnosis software routines (macrocode) or diagnosis firmware routines (microcode). The test process can be interrupted at any time so that normal operation can resume.

The generalized implementation flow diagram of the project is represented as follows. Initially the market research should be carried out which covers the previous version of the design and the current requirements on the design. Based on this survey, the specification and the architecture must be identified. Then the RTL modeling should be carried out in VHDL with respect to the identified architecture. Once the RTL modelling is done, it should be simulated and verified for all the cases. The functional verification should meet the intended architecture and should pass all the test cases.

Once the functional verification is clear, the RTL model will be taken to the synthesis process. Three operations will be carried out in the synthesis process such as
a) Translate
b) Map
c) Place and Route

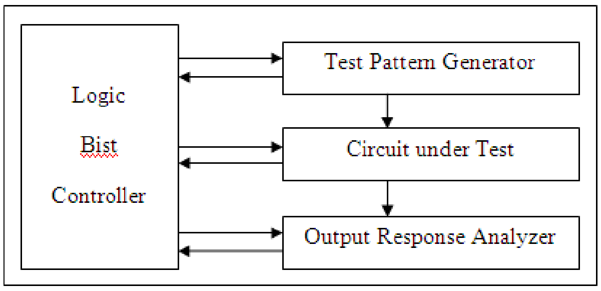

Fig.3: Basic Structure of BIST

IV. RESULTS

In the Fig 4 shows the two inputs with 8-bit length are ‘a’ (current Pixels) and ‘b’ (reference pixels) and 8-bit PE Output result. Here PE Output is nothing but sum of absolute difference (SAD).
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 5.

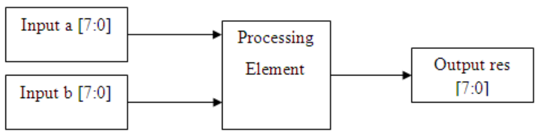

Fig.4: Processing Element Schematic Diagram

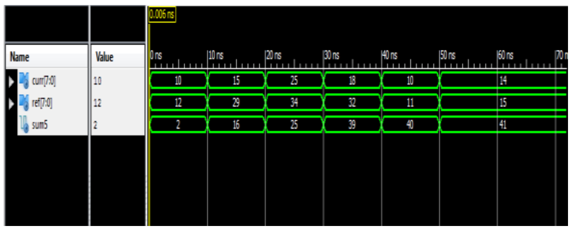

Fig.5: Simulation Waveform for Processing Element

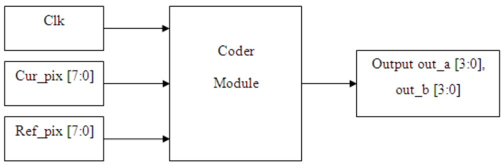

Fig.6: RQCG Schematic Diagram

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 6. 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 7.

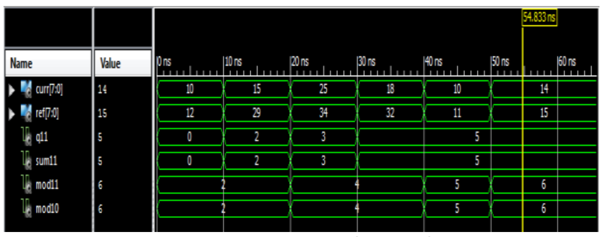

Fig 7: Simulation Waveform of RQCG

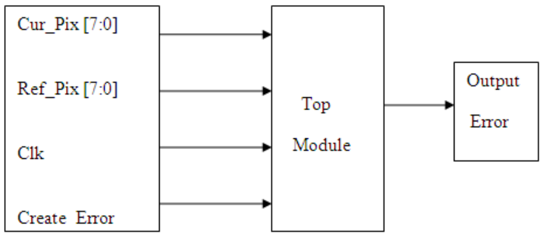

Fig.8: TOP Module

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 Sum of Absolute Difference, Processing Element, Modulus Division, RQCG, TCG, EDC and DRC modules. The schematic of Top Module for BISDC Architecture for MECA is shown in Fig 8.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 9.

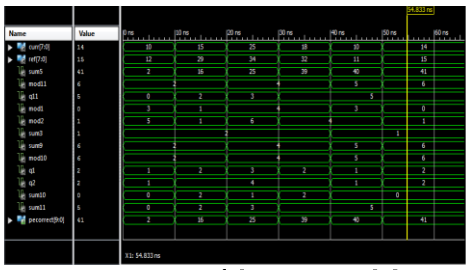

Fig.9: Output of the TOP Module

V. CONCLUSION

BISDC architecture for self-detection and self-correction of errors of PEs in an ME is proposed in this work. 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.

.

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.
[8] 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.