International Journal of Electronics Signals and Systems

Volume 2 | Issue 4 Article 12

April 2013

BUILT IN SELF TEST FOR SAD MODULE IN MOTION ARRAY DETECTION

K. K. GHOUSE
Department of ECE, JNTU ANANTAPUR, kk.ghouse@gmail.com

BUILT IN SELF TEST FOR SAD MODULE IN MOTION ARRAY DETECTION

KK GHOUSE[1] & S ARUNA MASTANI[2]

[1,2]Department of ECE, JNTU ANANTAPUR

Abstract- A novel method 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 i.e single error bit detection and correction . An advanced model has been proposed for multi bit detection using efficient adder implementation .a comparision is performed between efficient adder and processing element resultant .

1. 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.

The rest of the paper as follows .section 2 describes about the single detection error and correction section 3 describes about the proposed method multi detection error and correction followed by results and conclusion.

2. SYSTEM DESCRIPTION

Single bit error detection and correction:

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.

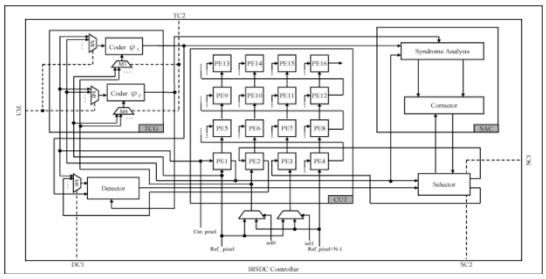

Fig1: Block Diagram of Proposed MECA BISDC.

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.

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.
Elements involved in bist:

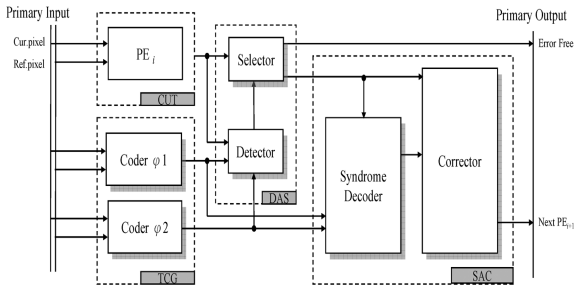

Fig 1 Example of the self-detection/correction operations

The self-detection and self correction operations (Fig.4.1) 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.
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.
The SAD is given by

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

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

In the Project 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

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 the bit position of SAD. Syndromes can be expressed as

$(S\varphi1, S\varphi2)= (|N'j-X|\varphi1, |N'j-Y|\varphi2) = (|e|\varphi1, |e|\varphi2)$

CORRECTOR
Input to the corrector module is the output of the selector module which is SAD that needs to be corrected; the bit position which needs correction is specified by the syndrome decoder. The corrector architecture consists of LUT and 12 multiplexers.

3. PROPOSED METHOD

Our design is intended to speed up the computation of the minimum SAD by its implemen- tation on a FPGA (SAD processor in ¯gure 1), while a core processor supplies the reference and candidate blocks to the FPGA device (dispatcher in figure 1). In this paper we propose a FPGA architecture to compute the minimum SAD. This design can be integrated with any BMA (full search or another efficient search strategy).

On-line arithmetic techniques have been considered to solve many signal processing problems, such us digital filtering, Fourier transform and others [6] [9]. Recent works have presented the attractiveness of OLA for FPGAs designs.
The MSD first mode of computation requires a flexibility in computing digits on the basis of partial information about inputs. This is achieved by the use of redundant representation system. In a redundant representation with radix r, each digit has more than r possible values. This allows several representation of a given value. Therefore, there is a flexibility in choosing an output digit at a given step so that, a compensation can be introduced if needed. SD representation system is used in this paper. In radix-2 SD representation, the digit set is {-1; 0; 1}. Two bits are required to represent each digit, as shown in Table 1. The first bit is negative weighted and the second one is positive weighted. This number system eliminates the long carry propagation chains in addition operation, although requires the carry of the two previous digits. In short, the advantages of using on-line arithmetic are: due to digit serial nature it reduces the number of signal lines connecting modules, the MSD first computation allows subsequent calculations to occur at much earlier stage, and it eliminates carry propagation chains since it uses a redundant number representation system.

On-line computation of the minimum SAD

The candidate blocks (supplied by the dispatcher) best match the reference block. The most commonly used metric to determine the best match is the sum of absolute differences (SAD). Thus, our design compute the minimum SAD among all the candidate blocks. To do this, a search iteration is performed for each candidate block. On each search iteration, the SAD corresponding of a candidate block is computed using all its pixels simultaneously. The value obtained is compared with the reference SAD (SADr) which is the minimum SAD computed before this iteration. If the current SAD (SADc) is less than SADr, it is stored as SADr for the remaining search iterations. Both the SAD computation and comparison operation is performed using on-line arithmetic. This allows us to begin the comparison when the first digit of the SAD is obtained and early stop the computation if the digits computed are enough to ensure that SADc is greater than SADr. The online comparator that we design for SAD comparison is described in [11].

The SAD adds up the absolute differences between corresponding elements in the candidate and reference block.

$$SAD = \sum_{i=1}^{N} \sum_{j=1}^{N} |c_{i,j} - r_{i,j}|$$

where r(i,j) are the elements of the reference block and c(i,j) the elements of the candidate block. Thus, the computation of the SAD is divided in three steps:
1 Computation of differences between corresponding elements d(i,j) = c(i,j) j(I,j)
2 Determine the absolute value of each di®erences d(I,j)
3 Add all absolute values

Conversion to SD representation and difference computation: In radix-2 SD representation, each digit is composed by two bits, the first one negative weighted and the second one positive weighted. Thus, a SD number can be interpreted as the difference of two unsigned numbers, the one composed by bits positive weighted of each digit, minus the one composed by bits negative weighted. In fact, to convert SD number to non redundant representation, this difference is performed.

Table 1. Digit codification in radix-2 signed-digit representation

Digit value	Digit representation
+1	01
0	00
0	11
-1	10

This property is used to perform simultaneously the conversion of each pixel value to SD and the difference between pixels of reference block and current block with no computational cost. In that way, each digit of the value d(i,j) = c(i,j) r(i,j) is obtained in SD representation by only taking the corresponding bit of ci;j as the positive weighted and the corresponding bit of r(i,j) as the negative weighted, since c(i,j) and r(i,j) are unsigned numbers.
Absolute value: To compute the absolute value of d(i,j) , the sign of this value have to be changed if di;j is negative. In SD the negation operation is performed by interchanging both bits of each digit.

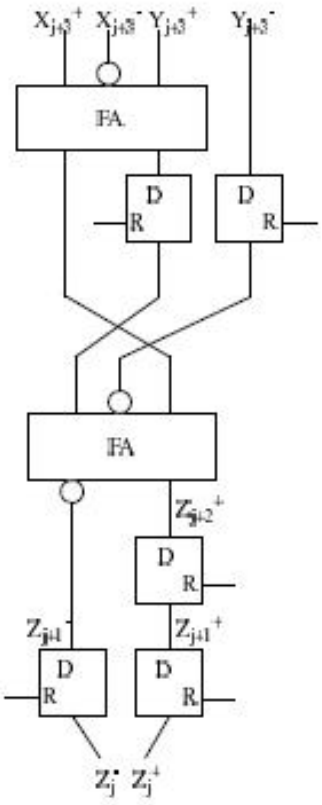

Fig. 2. On-line adder design.

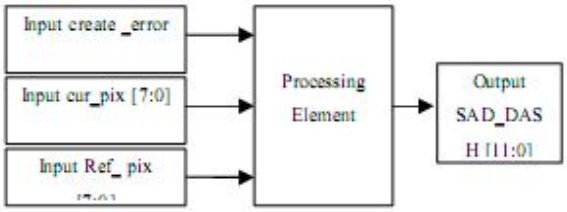

Fig 3 Processing Element Module Schematic Diagram

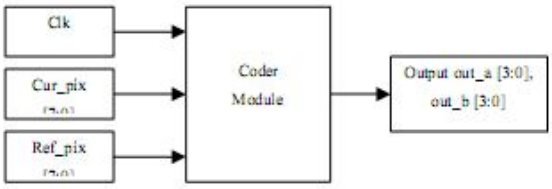

Fig 4 Coder Schematic Diagram

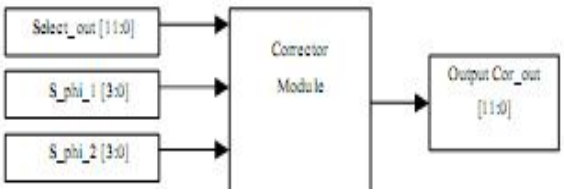

Fig 5 Selector Module Schematic diagram

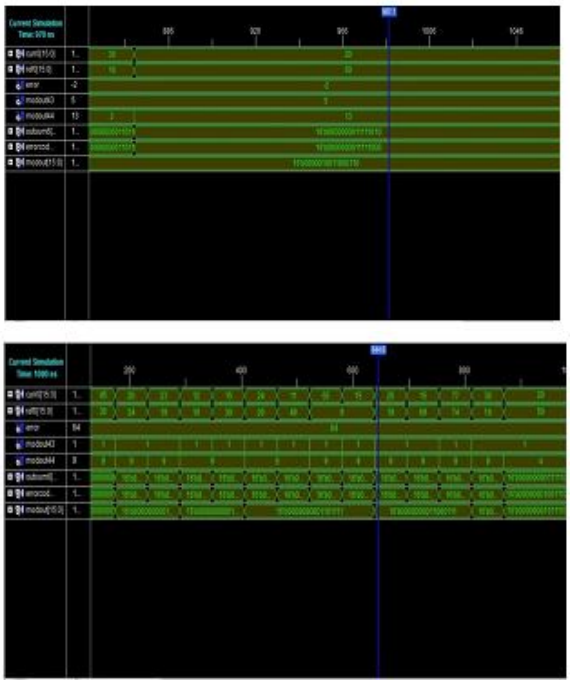

Fig 6 Corrector Module Schematic Diagram

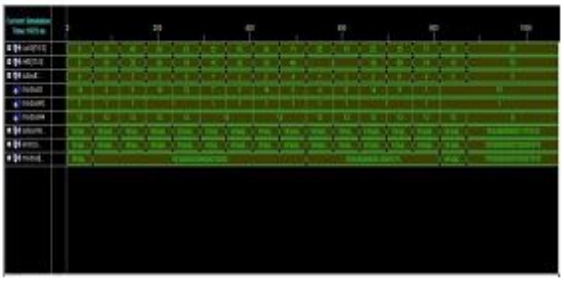

Figure 7 Simulation Results of the Output module

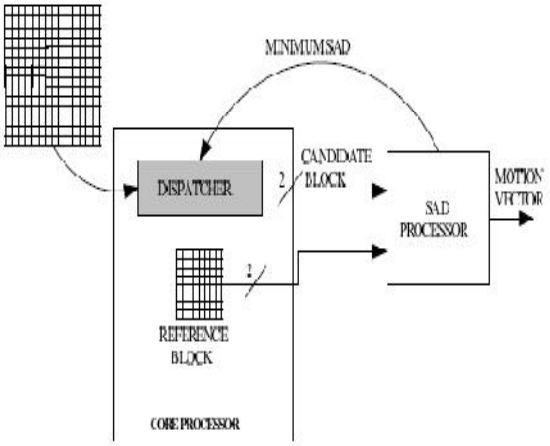

Figure 8 Core Processor Communication with SAD

Since MSD-first mode of computation is used, the sign detection of d(i,j) is performed on-the-°y by checking if the first non zero digit of d(i,j) is positive (01) or negative (10). The digits of d(i,j) are received in MSD-first mode and go directly to the output while they are zero (00 or 11). If the first non zero digit

received is positive (01), this and all the remaining digit correspond directly with the output. Nevertheless, if the first non zero digit received is negative (10), the bit of this and all the remaining digit are interchanged to obtain the output. The absolute value operation is performed with no online delay.

Sum of absolute di®erences: The absolute difference of all the pixels corresponding to the current and reference blocks are computed in parallel. Then N2 absolute difference blocks are required. An on-line adder tree is used to obtain the sum of all d(i,j) values as given in fig 2.

4. RESULTS AND CONCLUSIONS

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]. 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.