ISSN 2319 – 2518 www.ijeetc.com
Vol. 2, No. 4, October 2013
© 2013 IJEETC. All Rights Reserved

Research Paper

HIGH SPEED ERROR DETECTION AND DATA RECOVERY ARCHITECTURE FOR VIDEO APPLICATIONS

N Nagendra Prasad[1]* and J Rajesh Kumar[1]

*Corresponding Author: N Nagendra Prasad, ✉ nageendra@gmail.com

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 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, Residue-and-Quotient (RQ) code

INTRODUCTION

Advances in semiconductors, digital signal processing, and communication technologies have made multimedia applications more flexible and reliable. A good example is the H.264 video standard, also known as MPEG-4 Part 10.

Advanced Video Coding, which is widely regarded as the next generation video compression standard (Information Technology-Coding of Audio-Visual Objects-Part 2: Visual, 1999; and Advanced Video Coding for Generic Audiovisual Services, 2005). Video compression is necessary in a wide range of applications to reduce the total data amount required for transmitting or storing video data. Among the coding systems, a ME is of priority concern in exploiting the temporal redundancy between suc-cessive frames, yet also the most time consuming aspect of coding. Additionally, while performing up to 60%-90% of the computations encountered in the entire coding system, a ME is widely regarded as

[1] Department of ECE, SRTS, Kadapa, Andhra Pradesh, India.

the most computationally intensive of a video coding system (Huang *et al.*, 2006).

A ME generally consists of PEs with a size of 4 x 4. However, accelerating the computation speed depends on a large PE array, especially in high-resolution devices with a large search range such as HDTV (Chen *et al.*, 2006). Additionally, the visual quality and Peak Signal-to-Noise Ratio (PSNR) at a given bit rate are influenced if an error occurred in ME process. A testable design is thus increasingly important to ensure the reliability of numerous PEs in a ME. Moreover, although the advance of VLSI technologies facilitate the integration of a large number of PEs of a ME into a chip, the logic-perpin ratio is subsequently increased, thus decreasing significantly the efficiency of logic testing on the chip. As a commercial chip, it is absolutely necessary for the ME to introduce Design For Testability (DFT) (Wu *et al.*, 2007; Dong *et al.*, 2008; and Huang *et al.*, 2009). DFT focuses on increasing the ease of device testing, thus guaranteeing high reliability of a system. DFT methods rely on reconfiguration of a Circuit Under Test (CUT) to improve testability. While DFT approaches enhance the testability of circuits, advances in sub-micron technology and resulting increases in the complexity of electronic circuits and systems have meant that Builtin Self-Test (BIST) schemes have rapidly become necessary in the digital world. BIST for the ME does not expensive test equipment, ultimately lowering test costs (Li *et al.*, 2004; Huang *et al.*, 2008; and Liu *et al.*, 2009). Moreover, BIST can generate test simulations and analyze test responses without outside support, subsequently streamlining the testing and diagnosis of digital systems. However, increasingly complex density of circuitry requires that the built-in testing approach not only detect faults but also specify their locations for error correcting. Thus, extended schemes of BIST referred to as built-in self-diagnosis (Portal *et al.*, 2003) and built-in self-correction (Lin *et al.*, 2005; Cheng *et al.*, 2009; and Hsu *et al.*, 2010) have been developed recently.

While the extended BIST schemes generally focus on memory circuit, testing-related issues of video coding have been seldom addressed. Thus, exploring the feasibility of an embedded testing approach to detect errors and recover data of a ME is of worthwhile interest. Additionally, the reliability issue of numerous PEs in a ME can be improved by enhancing the capabilities of Concurrent Error Detection (CED) (Bayat-Sarmadi and Hasan, 2007; and Chiou *et al.*, 2009). The CED approach can detect errors through conflicting and undesired results generated from operations on the same operands. CED can also test the circuit at full operating speed without interrupting a system. Thus, based on the CED concept, this work develops a novel EDDR architecture based on the RQ code to detect errors and recovery data in PEs of a ME and, in doing so, further guarantee the excellent reliability for video coding testing applications.

RQ CODE GENERATION

Coding approaches such as parity code, Berger code, and residue code have been considered for design applications to detect circuit errors (Piestrak *et al.*, 2001; and Breveglieri *et al.*, 2006). 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. For instance, assume that N denotes an integer, N_1 and N_2 represent data words, and m refers to the modulus. A separate residue code of interest is one in which N is coded as a pair $(N|N|_m)$. Notably, $|N|_m$is the residue of N modulo m. 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 not be recovered effectively by using the residue codes. Therefore, this work presents a quotient code, which is derived from the residue code, to assist the residue code in detecting multiple errors and recovering errors. The mathematical model of RQ code is simply described as follows. Assume that binary data X is expressed as

$$X = \{b_{n-1}b_{n-2}\dots b_2b_1b_0\} = \sum_{j=0}^{n-1} b_j 2^j \qquad \text{...(1)}$$

The RQ code of X modulo m expressed as $R = |X|_m$ $Q = \lfloor X/m \rfloor$, respectively. Notably, $\lfloor i \rfloor$ denotes the largest integer not exceeding i.

According to the above RQ code expression, the corresponding circuit design of the RQCG can be realized. In order to simplify the complexity of circuit design, the implementation of the module is generally dependent on the addition operation. Additionally, based on the concept of residue code, the following definitions shown can be applied to generate the RQ code for circuit design.

Definition 1

$$|N_1 + N_2|_m = \big||N_1|_m + |N_2|_m\big|_m \qquad \text{...(2)}$$

Definition 2

Let $N_j = n_1 + n_2 + ... + n_j$, then

$$|N_j|_m = \big||n_1|_m + |n_2|_m + \dots + |n_j|_m\big|_m \qquad \text{...(3)}$$

To accelerate the circuit design of RQCG, the binary data shown in (1) can generally be divided into two parts:

$$X = \sum_{j=0}^{n-1} b_j 2^j$$

$$= \left(\sum_{j=0}^{k-1} b_j 2^j\right) + \left(\sum_{j=k}^{n-1} b_j 2^{j-k}\right) 2^k$$

$$= Y_0 + Y_1 2^k \qquad \text{...(4)}$$

Significantly, the value of k is equal to $\lfloor n/2 \rfloor$and the data formation of Y_0 and Y_1 are a decimal system. If the modulus $m = 2^k 1$, then the residue code of X modulo m is given by

$$R = |X|_m$$

$$= |Y_0 + Y_1|_m = |Z_0 + Z_1|_m = (Z_0 + Z_1) \qquad \text{...(5)}$$

$$Q = \left\lfloor \frac{X}{m} \right\rfloor$$

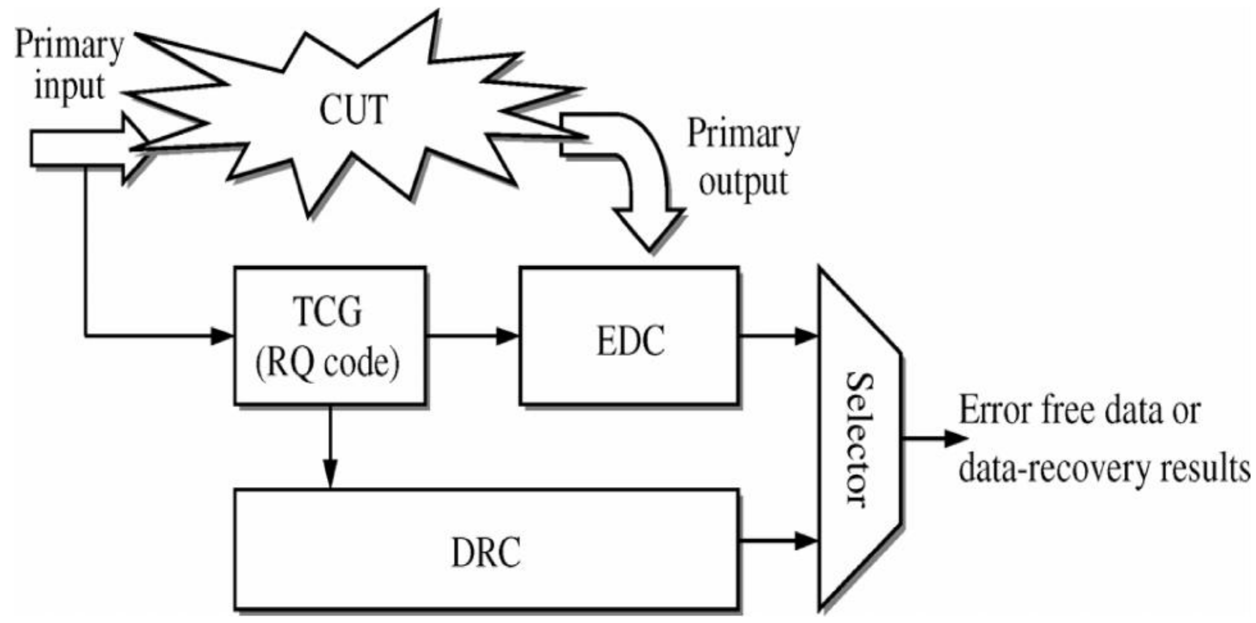
$$= \left\lfloor \frac{Y_0 + Y_1}{m} \right\rfloor + Y_1 = \left\lfloor \frac{Z_0 + Z_1}{m} \right\rfloor + Z_1 + Y_1$$

$$= Z_1 + Y_1 + \qquad \text{...(6)}$$

where

$$\sim(\ ) = \begin{cases} 0(1), & \text{if } Z_0 + Z_1 = m \\ 1(0), & \text{if } Z_0 + Z_1 < m \end{cases}$$

Figure 1: Conceptual View of the Proposed EDDR Architecture



Notably, since the value of $Y_0 + Y_1$ is generally greater than that of modulus *m*, the Equations in (5) and (6) must be simplified further to replace the complex module operation with a simple addition operation by using the parameters Z_0, Z_1, and .

Based on (5) and (6), 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.

PROPOSED EDDR ARCHITECTURE DESIGN

Figure 1 shows the conceptual view of the proposed EDDR 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 Figure 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 EDDR scheme to detect errors and recover the corresponding data.

This work adopts the systolic ME (Surin and Hu, 2001) as a CUT to demonstrate the feasibility of the proposed EDDR 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 (Park *et al.*, 2007). Notably, some registers and latches may exist in ME to complete the data shift and storage. Figure 2 shows an example of the proposed EDDR circuit design for a specific PE_i of a ME. The fault model definition, RQCG-based TCG design, operations of error detection and data recovery, and the overall test strategy are described carefully as follows.

Fault Model

The PEs are essential building blocks and are connected regularly to construct a ME. Generally, PEs are surrounded by sets of ADDs and accumulators that determine how data flows through them. PEs can thus be considered the class of circuits called ILAs, whose testing assignment can be easily achieved by using the fault model, Cell Fault Model (CFM) (Li and Hsu, 2004). Using CFM has received considerable interest due to accelerated growth in the use of high-level synthesis, as well as the parallel increase in complexity and density of Integration Circuits (ICs). Using CFM makes the tests independent of the adopted synthesis tool and vendor library. Arithmetic modules, like ADDs (the primary element in a PE), due to their regularity, are designed in an extremely dense configuration.

Moreover, a more comprehensive fault model, i.e., the Stuck-At (SA) model, must be adopted to cover actual failures in the interconnect data bus between PEs (Vasudevan *et al.*, 2007). The SA fault is a well-known structural fault model, which assumes that 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 SA fault in a ME architecture can incur errors in computing SAD values. A distorted computational error (e) and the magnitude of e are assumed here to be equal to SAD'-SAD, where SAD' denotes the computed SAD value with SA faults.

TCG Design

According to Figure 2, 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 PE_i in Figure 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 x N can be evaluated:

$$SAD = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left|X_{ij} - Y_{ij}\right|$$

$$= \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left|\left(q_{xij}\cdot m + r_{xij}\right) - \left(q_{yij}\cdot m + r_{yij}\right)\right| \qquad ...(7)$$

where r_{xij}, q_{xij} and r_{yij}, q_{yij} denote the corresponding RQ code of X_{ij} and Y_{ij} modulo m. Importantly, X_{ij} and Y_{ij} 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 (R_T and Q_T) form TCG. Namely, the circuit

Figure 2: A Specific PE_i Testing Processes of the Proposed EDDR Architecture

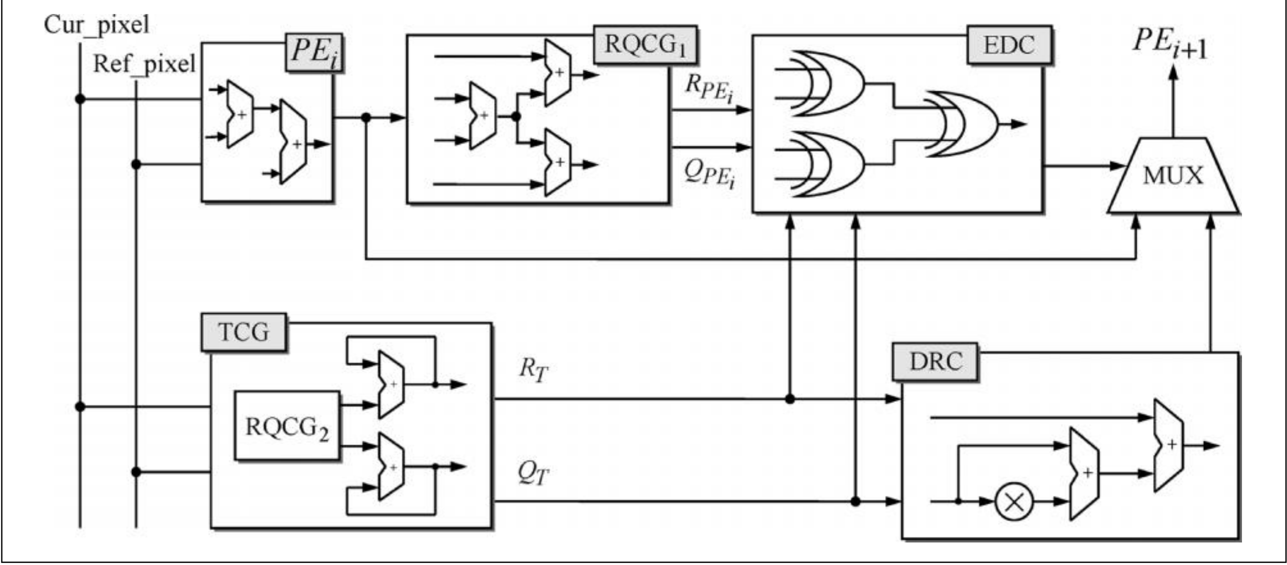
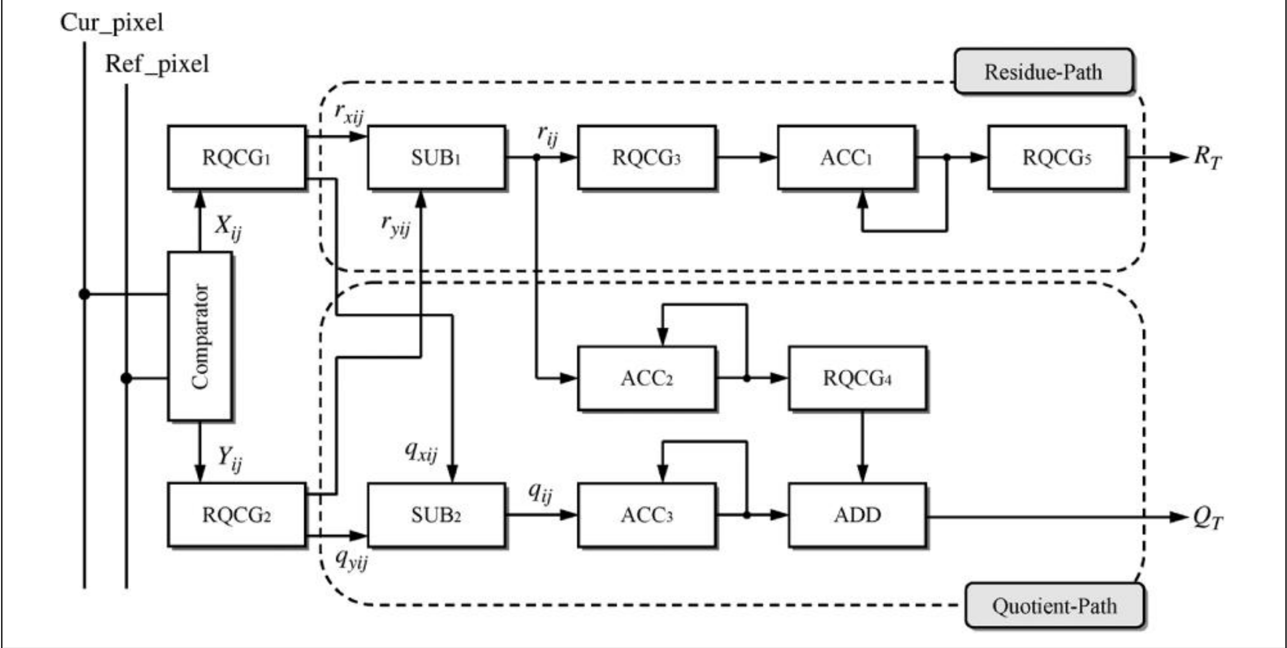


Figure 3: Circuit Design of the TCG



design of TCG can be easily achieved (see Figure 3) by using

$$R_T = \left|\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left(X_{ij} - Y_{ij}\right)\right|$$

$$= \left\|\left(X_{00} - Y_{00}\right)\right|_m + \left|\left(X_{01} - Y_{01}\right)\right|_m + \ldots$$

$$+ \left|\left(X_{(N-1)(N-1)} - Y_{(N-1)(N-1)}\right)\right|_m\Big|_m$$

$$= \left\|\left(q_{x00}\cdot m + r_{x00}\right) - \left(q_{y00}\cdot m + r_{y00}\right)\right|_m$$

$$+ \ldots\left|\left(q_{x(N-1)(N-1)}\cdot m + r_{x(N-1)(N-1)}\right)\right.$$

$$\left. - \left(q_{y(N-1)(N-1)}\cdot m + r_{y(N-1)(N-1)}\right)\right|_m\Big|_m$$

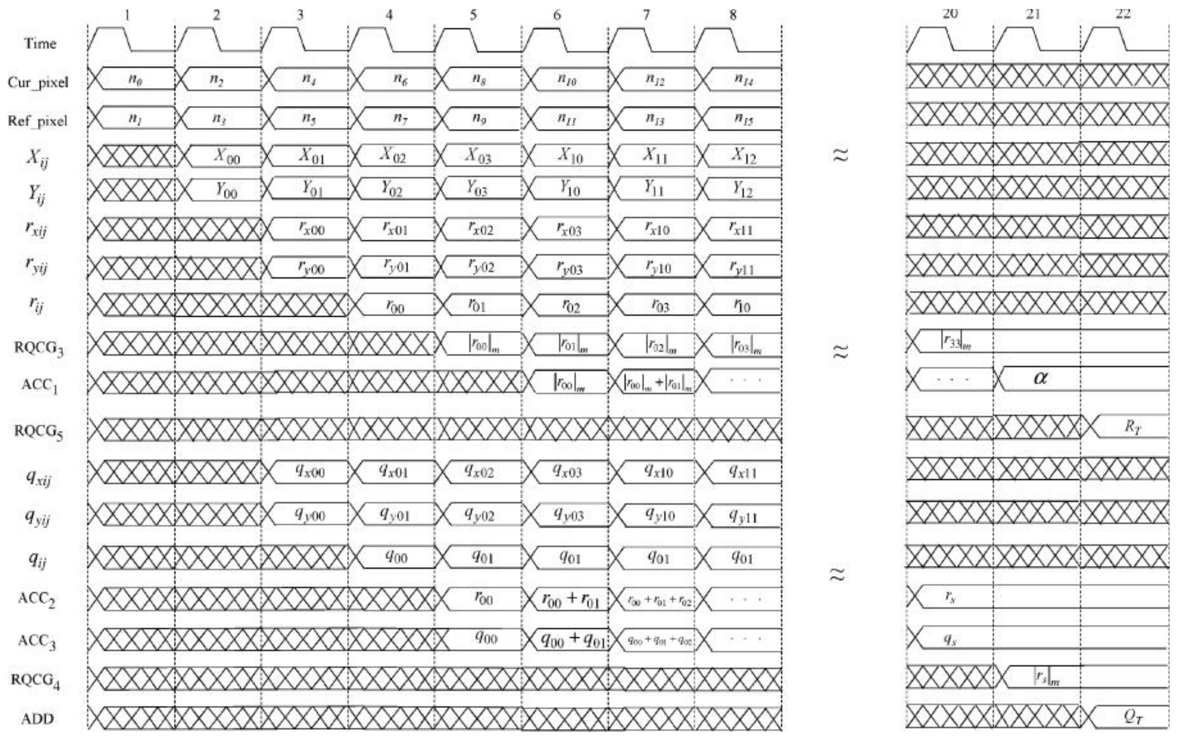
$$= \left| \left| (r_{x00} - r_{y00}) \right|_m + \left| (r_{x01} - r_{y01}) \right|_m + \dots + \left| (r_{x(N-1)(N-1)} - r_{y(N-1)(N-1)}) \right|_m \right|_m$$

$$= \left| |r_{00}|_m + |r_{01}|_m + \dots + |r_{(N-1)(N-1)}|_m \right|_m \quad \text{...(8)}$$

and (9), shown at the bottom of the following page, to derive the corresponding RQ code.

Figure 4 shows the timing chart for a macroblock with a size of 4 x 4 in a specific PE_i to demonstrate the operations of the TCG circuit. The data n_0 and n_1 from Cur_pixel and Ref_pixel must be sent to a comparator in order to determine the luminance pixel value X_{ij} and Y_{ij} at the 1st clock. Notably, if $X_{ij} \geq Y_{ij}$, then X_{ij} and Y_{ij} are the luminance pixel value of Cur_pixel and Ref_pixel, respectively. Conversely, X_{ij} represents the luminance pixel value of Ref_pixel, and Y_{ij} denotes the luminance pixel value of Cur_pixel when $X_{ij} < Y_{ij}$. At the 2nd clock, the values of X_{00} and Y_{00} are generated and the corresponding RQ code r_{x00}, q_{x00}, r_{y00}, q_{x00} can be captured by the $RQCG_1$ and $RQCG_2$ circuits if the 3rd clock is triggered. Equations (8) and (9) clearly indicate that the codes of r_{00} and q_{00} can be obtained by using the circuit of a subtracter (SUB). The 4th clock displays the operating results. The modulus value of r_{00} is then obtained at the 5th clock. Next, the summation of quotient values and residue values of modulo m are proceeded with from clocks 5-21 through the circuits of ACCs. Since a 4 x 4 macroblock in a specific PE_i of a ME contains 16 pixels, the corresponding RQ code (R_T and Q_T) is exported to the EDC and DRC circuits in order to detect errors and recover data after 22 clocks. Based on the TCG circuit design shown in Figure 4, the error detection and data recovery operations of a specific PE_i in a ME can be achieved.

Figure 4: Timing Chart of the TCG



1
2
3
4
5
6
7
8
20
21
22
Time
Cur_pixel
Ref_pixel
X_{ij}
Y_{ij}
r_{xij}
r_{yij}
r_{ij}
$RQCG_3$
ACC_1
$RQCG_5$
q_{xij}
q_{yij}
q_{ij}
ACC_2
ACC_3
$RQCG_4$
ADD
n_0 n_2 n_4 n_6 n_8 n_{10} n_{12} n_{14}
n_1 n_3 n_5 n_7 n_9 n_{11} n_{13} n_{15}
X_{00} X_{01} X_{02} X_{03} X_{10} X_{11} X_{12}
Y_{00} Y_{01} Y_{02} Y_{03} Y_{10} Y_{11} Y_{12}
r_{x00} r_{x01} r_{x02} r_{x03} r_{x10} r_{x11}
r_{y00} r_{y01} r_{y02} r_{y03} r_{y10} r_{y11}
r_{00} r_{01} r_{02} r_{03} r_{10}
$|r_{00}|_m$ $|r_{01}|_m$ $|r_{02}|_m$ $|r_{03}|_m$ $|r_{33}|_m$
$|r_{00}|_m$ $|r_{00}|_m + |r_{01}|_m$ α
R_T
q_{x00} q_{x01} q_{x02} q_{x03} q_{x10} q_{x11}
q_{y00} q_{y01} q_{y02} q_{y03} q_{y10} q_{y11}
q_{00} q_{01} q_{01} q_{01} q_{01}
r_{00} $r_{00} + r_{01}$ $r_{00} + r_{01} + r_{02}$ r_s
q_{00} $q_{00} + q_{01}$ $q_{00} + q_{01} + q_{02}$ q_s
$|r_s|_m$
Q_T

EDDR Processes

Figure 2 clearly indicates that the operations of error detection in a specific PE_i is achieved by using EDC, which is utilized to compare the outputs between TCG and $RQCG_1$ in order to determine whether errors have occurred. If the values of $R_{PE_i} \neq R_T$ and/or $Q_{PE_i} \neq Q_T$, then the errors in a specific PE_i can be detected. The EDC output is then used to generate a 0/1 signal to indicate that the tested PE_i is error-free/errancy.

$$Q_T = \left\lfloor \frac{\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}(x_{ij} - y_{ij})}{m} \right\rfloor$$

$$= \left\lfloor \frac{(x_{00} - y_{00}) + (x_{01} - y_{01}) + \ldots + (x_{(N-1)(N-1)} - y_{(N-1)(N-1)})}{m} \right\rfloor$$

$$= \left\lfloor \frac{(q_{x00} \cdot m - q_{y00} \cdot m)}{m} + \frac{(r_{x00} - r_{y00})}{m} + \frac{(q_{x01} \cdot m - q_{y01} \cdot m)}{m} + \frac{(r_{x01} - r_{y01})}{m} + \ldots \right\rfloor$$

$$= \left\lfloor (q_{x00} - q_{y00}) + (q_{x01} - q_{y01}) + \ldots + \frac{(r_{x00} - r_{y00}) + (r_{x01} - r_{y01})}{m} + \ldots \right\rfloor$$

$$= q_{00} + q_{01} + \ldots + q_{(N-1)(N-1)} + \left\lfloor \frac{r_{00} + r_{01} + \ldots + r_{(N-1)(N-1)}}{m} \right\rfloor \quad \ldots(9)$$

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 PE_i. In other words, the SAD value is transformed to $SAD' = SAD + e$ if an error e occurred. Notably, the error signal e is expressed as

$$e = q_e \cdot m + r_e \quad \ldots(10)$$

to comply with the definition of RQ code. Under the faulty case, the RQ code from $RQCG_2$ of the TCG is still equal to (8) and (9). However, R_{PE_i} and Q_{PE_i} are changed to (13) and (14) because an error e has occurred. Thus, the error in a specific PE_i can be detected if and only if (8) ≠ (11) and/or (9) ≠ (12):

$$R_{PE_i} = |SAD'|_m$$

$$= \left| \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}(X_{ij} - Y_{ij}) + e \right|_m$$

$$= \left| |r_{00}|_m + |r_{01}|_m + \ldots + |r_{(N-1)(N-1)}|_m + |r_e|_m \right|_m \quad \ldots(11)$$

$$Q_{PE_i} = \left\lfloor \frac{SAD'}{m} \right\rfloor$$

$$= \left\lfloor \frac{\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}(X_{ij} - Y_{ij}) + e}{m} \right\rfloor$$

$$= q_{00} + q_{01} + \ldots + q_{(N-1)(N-1)} + q_e$$

$$+ \left\lfloor \frac{(r_{00} + r_{01} + \ldots + r_{(N-1)(N-1)} + q_e)}{m} \right\rfloor \quad \ldots(12)$$

During data recovery, the circuit DRC plays a significant role in recovering RQ code from TCG. The data can be recovered by implementing the mathematical model as

$$SAD = m \times Q_T + R_T$$

$$= (2^j - 1) \times Q_T + R_T$$

$$= 2^j \times Q_T - Q_T + R_T \quad \ldots(13)$$

To realize the operation of data recovery in (13), a Barrel shift (Yu *et al.*, 2006) and a corrector circuits are necessary to achieve the functions of (2^j x Q_T) and ($-Q_T$ + R_T), respectively. Notably, the proposed EDDR design executes the error detection and data recovery operations simultaneously. Additionally, error-free data from the tested PE_i or the data recovery that results from DRC is selected by a multiplexer (MUX) to pass to the next specific PE_{i+1} for subsequent testing.

Numerical Example

A numerical example of the 16 pixels for a 4 x 4 macroblock in a specific PE_i of a ME is described as follows. Figure 5 presents an example of pixel values of the Cur_pixel and Ref_pixel. Based on (7), the SAD value of the 4 x 4 macroblock is

$$SAD = \sum_{i=0}^{3}\sum_{j=0}^{3}\left|X_{ij} - Y_{ij}\right|$$

$$= \left|X_{00} - Y_{00}\right| + \left|X_{01} - Y_{01}\right| + \ldots + \left|X_{33} - Y_{33}\right|$$

$$= (128 - 1) + (128 - 1) + \ldots + (128 - 5)$$

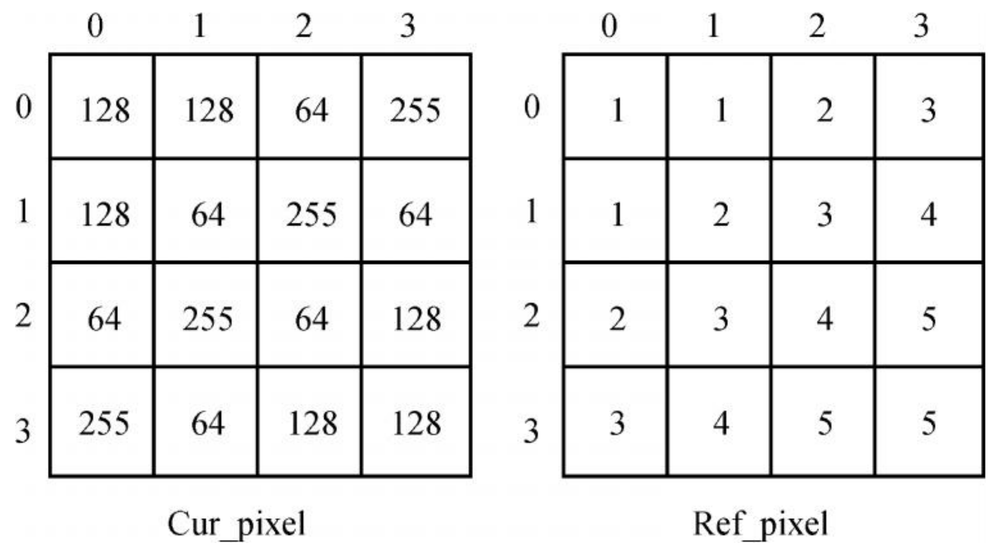
$$= 2124 \qquad \text{...(14)}$$

According to the description of RQ code generation, the modulo is assumed here to be $m = 2^6 1 = 63$. Thus, based on (8) and (9), the RQ code of the SAD value shown in (14) are $R_T = R_{PE_i} = \left|2124\right|_{63} = 45$ and $Q_T = Q_{PE_i} = \left|2124/63\right| = 33$. Since the value of $R_T(Q_T)$ is equal to $R_{PE_i}(Q_{PE_i})$, EDC is enabled and a signal "0" is generated to describe a situation in which the specific PE_i is error-free. Conversely, if SA1 and SA0 errors occur in bits 1 and 12 of a specific PE_i, i.e., the pixel values of PE_i, $2124 = 100001001100_2$ is turned into $77 = 000001001101_2$, resulting in a transformation of the RQ code of R_{PE_i} and Q_{PE_i} into $|77|_{63} = 14$ and $\lfloor 77/63 \rfloor = 1$. Thus, an error signal "1" is generated from EDC and sent to the MUX in order to select the recovery results from DRC.

Overall Test Strategy

By extending the testing processes of a specific PE_i in Figure 2, Figure 6 illustrates the overall EDDR architecture design of a ME. First, the input data of Cur_pixel and Ref_pixel are sent simultaneously to PEs and TCGs in order to estimate the SAD values and generate the test RQ code R_T and Q_T. Second, the SAD value from the tested object PE_i,

Figure 5: Example of Pixel Values



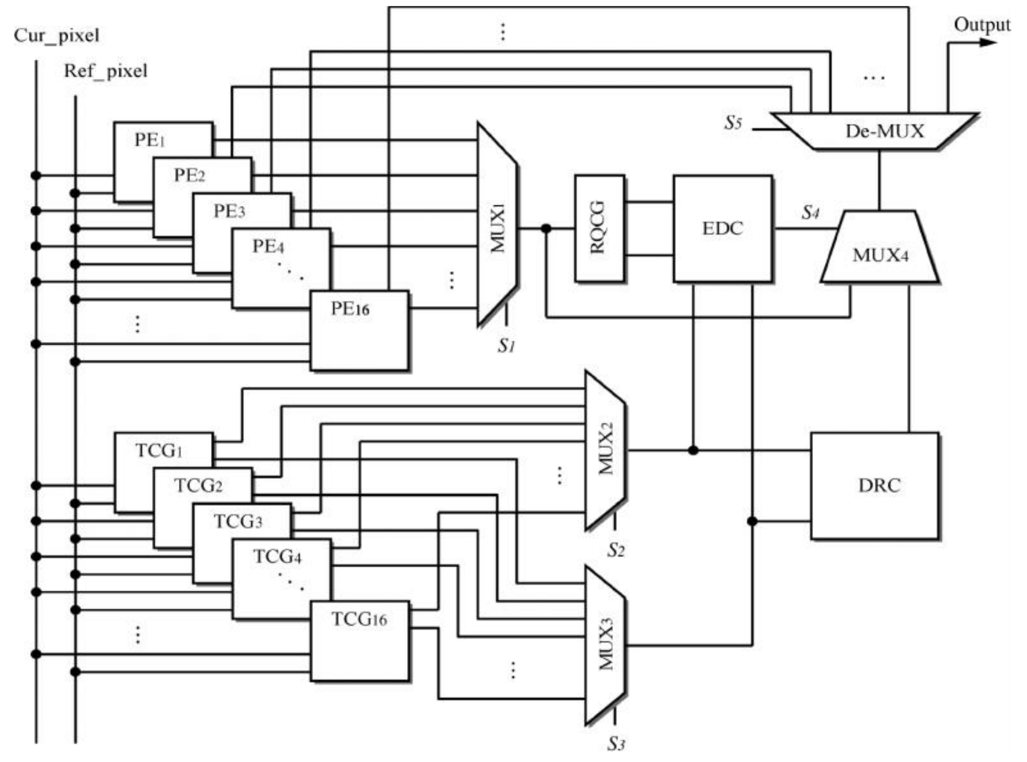
	0	1	2	3
0	128	128	64	255
1	128	64	255	64
2	64	255	64	128
3	255	64	128	128

Cur_pixel

	0	1	2	3
0	1	1	2	3
1	1	2	3	4
2	2	3	4	5
3	3	4	5	5

Ref_pixel

Figure 6: Proposed EDDR Architecture Design for a ME



which is selected by MUX_1, is then sent to the RQCG circuit in order to generate R_{PE_i} and Q_{PE_i} codes. Meanwhile, the corresponding test codes R_{T_i} and Q_{T_i} from a specific TCG_i are selected simultaneously by MUXs 2 and 3, respectively. Third, the RQ code from TCG_i and RQCG circuits are compared in EDC to determine whether the tested object PE_i have errors. The tested object PE_i is error-free if and only if $R_{PE_i} = R_{T_i}$ and $Q_{PE_i} = Q_{T_i}$. Additionally, DRC is used to recover data encoded by TCG_i, i.e., the appropriate R_{T_i} and Q_{T_i} codes from TCG_i are selected by MUXs 2 and 3, respectively, to recover data. Fourth, the error-free data or data recovery results are selected by MUX_4. Notably, control signal S_4 is generated from EDC, indicating that the comparison result is error-free ($S_4 = 0$) or errancy ($S_4 = 1$). Finally, the error-free data or the data-recovery result from the tested object PE_i is passed to a De-MUX, which is used to test the next specific PE_{i+1}; otherwise, the final result is exported.

RESULTS AND DISCUSSION

Extensive verification of the circuit design is performed using the VHDL and then synthesized by the Synopsys Design Compiler with TSMC 0.18-μm 1P6M CMOS technology to demonstrate the feasibility of the proposed EDDR architecture design for ME testing applications.

Experimental Results

Table 1 summarizes the synthesis results of area overhead and time penalty of the proposed EDDR architecture. The area is

Table 1: Estimation of Area Overhead and Time Penalty

Components	PE	RQCG	EDC	TCG	DRC
Area (Gate Counts)	69482	1779	667	3265	2376
Operation Time (ns)	973.76	10.17	6.02	1016.56	17.99
Area Overhead (%)			5.13		
Time Penalty (%)			6.24		

estimated based on the number of gate counts. By considering 16 PEs in a ME and 16 TCGs of the proposed EDDR architecture, the area overhead of error detection, data recovery, and overall EDDR architecture (AO_{ED}, AO_{DR}, and AO_{EDDR}) are

$$AO_{ED} = \frac{1779 + 3265 \times 16 + 667}{69482 \times 16} = 4.92\% \quad ...(15)$$

$$AO_{DR} = \frac{3265 \times 16 + 2376}{69482 \times 16} = 4.91\% \quad ...(16)$$

$$AO_{EDDR} = \frac{1779 + 667 + 3265 \times 16 + 2376}{69482 \times 16} = 5.13\% \quad ...(17)$$

The time penalty is another criterion to verify the feasibility of the proposed EDDR architecture. Table 1 also summarizes the operating time evaluation of a specific PE_i and each component in the proposed EDDR architecture. The following equations show the time penalty of error detection and data recovery (TP_{ED} and TP_{DB}) operations for a 4 x 4 macroblock (a PE_i with 16 pixels):

$$TP_{ED} = \frac{(1016.56 + 6.02)\, 973.76}{973.76} = 5.01\% \quad ...(18)$$

$$TP_{DB} = \frac{(1016.56 + 17.99)\, 973.76}{973.76} = 6.24\% \quad ...(19)$$

Notably, each PE of a ME is tested sequentially in the proposed EDDR architecture. Thus, if the proposed EDDR architecture is embedded into a ME for testing, in which the entire timing penalty is equivalent to that for testing a single PE., i.e., approximately about 5.01% and 6.24% time penalty of the operations of error detection and data recovery, respectively.

Notably, the operating time of the RQCG circuit can be neglected to evaluate TP_{ED} because TCG covers the operating time of RQCG. Additionally, the error-free/errancy signal from EDC is generated after 1022.58 ns (1016.56 + 6.02). Thus, the error-free data is selected directly from the tested object PE_i because the operating time of the tested object PE_i is faster than the results of data recovery from DRC.

Performance Discussion

The TCG component plays a major role in the proposed EDDR architecture to detect errors and recover data. Additionally, the number of TCGs significantly influences the circuit performance in terms of area overhead and throughput. Figures 7 and 8 illustrate the relations between the number of TCGs, area overhead and throughput. The area overhead is less than 2% if only one TCG is used to execute; however, at this time, the throughput is extremely small. Notably, the throughput of a ME without embedding the proposed EDDR architecture is about 25800 kMB/s. Figure 8 clearly indicates that the throughput is around 25000 kMB/s, if the proposed EDDR architecture with 16 TCGs is embedded into a ME for testing. Thus, to maintain the same throughput as much as possible, 16 TCGs must be adopted in the proposed EDDR

Figure 7: Relation Between TCG and Area Overhead

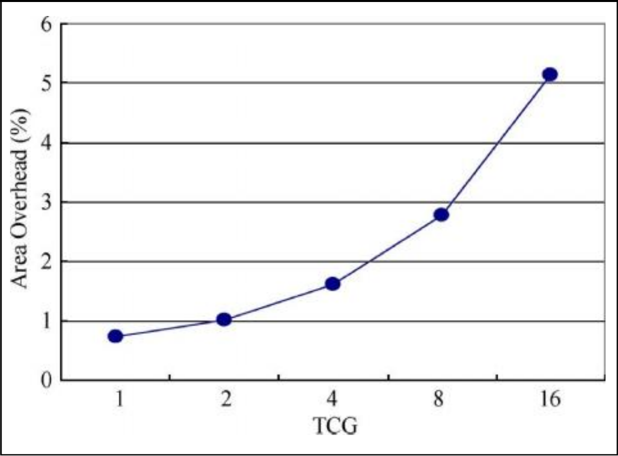
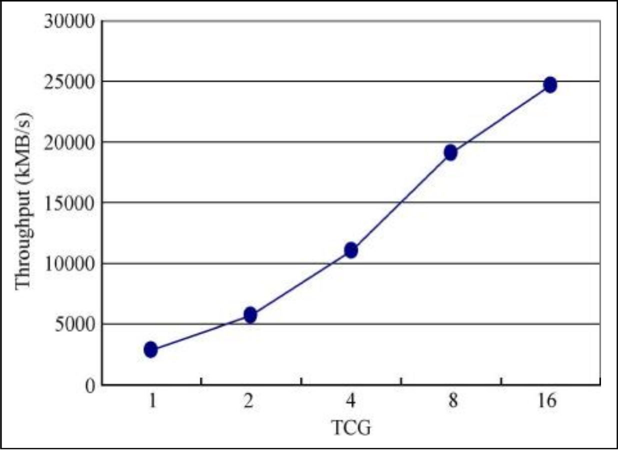


Figure 8: Relation Between TCG and Throughput



architecture for a ME testing applications. Although the area overhead is increased if 16 TCGs used (see Figure 7), the area overhead is only about 5.13%, i.e., an acceptable design for circuit testing.

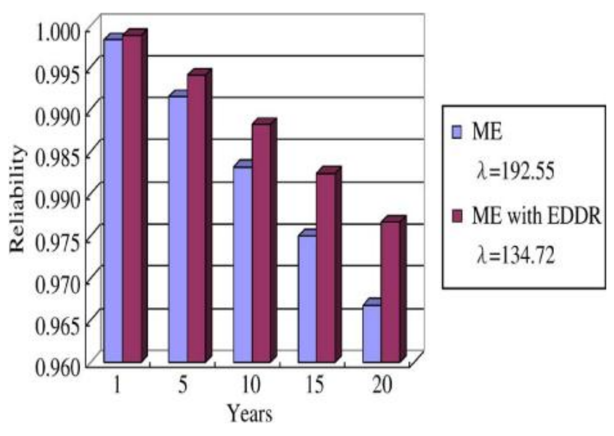
This work also addresses reliability-related issues to demon- strate the feasibility of the proposed EDDR architecture. Reliability is the probability that a component or a system performs its required function under different operating conditions encountered for a certain time period (Neubeck, 2004). The constant failurerate reliability model.

$$R(t) = e^{-\text{a}t} = e^{-\text{a}_b \Pi_T \Pi_Q \Pi_E t} = e^{-\left(\frac{1500}{T^{1.5}}\right) G^{T} \frac{5}{1.5} \Pi_T \Pi_Q \Pi_E t} \quad \text{...(20)}$$

is used to estimate the reliability of the proposed EDDR architecture for ME testing applications, where a denotes the failure-rate; a_b represents the base failure-rate of MOS digital logic; *G* refers to Gate count; $\Pi_T = 1.0$ (25 °C); $\Pi_Q = 1.0$ (hermetic package); and $\Pi_E = 1.0$ (ground benign environment).

The failure-rate a in (20) can be expressed as the ratio of the total number of failures to the total operating time, i.e., Failure-rate in Time (FIT), which represents the number of failures per device hours of accelerated stress tests (Yu *et al.*, 2006). Notably, the total operating time, *T* in a_b can be expressed as the year of manufacturing. Since the proposed EDDR architecture is synthesized by using TSMC 0.18 μm 1P6M CMOS technology, 1998 is given as the year of manufacturing for a wide variety of components. Thus, *T* is defined as 12 years, because the year of manufacturing is 1998. Figure 9 clearly indicates that the low failure-rate and high reliability levels can be obtained if the

Figure 9: Failure-Rate and Reliability Analysis



proposed EDDR architecture is embedded into a ME for testing applications.

CONCLUSION

This work presents an EDDR architecture for detecting the errors and recovering the data of PEs in a 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. The proposed EDDR architecture is also implemented by using VHDL and synthesized by the Synopsys Design Compiler with TSMC 0.18 μm 1P6M CMOS technology. Experimental results indicate that that the proposed EDDR architecture can effectively detect errors and recover data in PEs of a ME with reasonable area overhead and only a slight time penalty. Throughput and reliability issues are also discussed to demonstrate the satisfactory performance of the proposed EDDR architecture design for ME testing applications.

REFERENCES

1. Bayat-Sarmadi S and Hasan M A (2007), “On Concurrent Detection of Errors in Polynomial Basis Multiplication”, *IEEE Trans. Vary Large Scale Integr. (VLSI) Systs.*, Vol. 15, No. 4, pp. 413-426.
2. Breveglieri L, Maistri P and Koren I (2006), “A Note on Error Detection in an RSA Architecture by Means of Residue Codes”, in *Proc. IEEE Int. Symp. On-Line Testing*, July, pp. 176-177.
3. Chen C Y, Chien S Y, Huang Y W, Chen T C, Wang T C and Chen L G (2006), “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.
4. Cheng C H, Liu Y and Hsu C L (2009), “Low-Cost BISDC Design for Motion Estimation Computing Array”, in *Proc. IEEE Circuits Syst. Int. Conf.*, pp. 1-4.
5. Chiou C W, Chang C C, Lee C Y, Hou T W and Lin J M (2009), “Concurrent Error Detection and Correction in Gaussian Normal Basis Multiplier Over GF (2^m)”, *IEEE Trans.Comput.*, Vol. 58, No. 6, pp. 851-857.
6. Dong M Y, Yang S H and Lu S K (2008), “Design-for-Testability Techniques for Motion Estimation Computing Arrays”, in *Proc. Int. Conf. Commun., Circuits Syst.*, May, pp. 1188-1191.
7. Hsu C L, Cheng C H and Liu Y (2010), “Built-in Self-Detection/Correction Architecture for Motion Estimation Computing Arrays”, *IEEE Trans. Vary Large Scale Integr. (VLSI) Systs.*, Vol. 18, No. 2, pp. 319-324.
8. Huang Y S, Chen C K and Hsu C L (2008), “Efficient Built-in Self-Test for Video Coding Cores: A Case Study on Motion Estimation Computing Array”, in *Proc. IEEE Asia Pacific Conf. Circuit Syst.*, December, pp. 1751-1754.
9. Huang Y S, Yang C J and Hsu C L (2009), “C-Testable Motion Estimation Design for Video Coding Systems”, *J. Electron. Sci. Technol.*, Vol. 7, No. 4, pp. 370-374.
10. Huang Y W, Hsieh B Y, Chien S Y, Ma S Y and Chen L G (2006), “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.

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

12. Li D, Hu M and Mohamed O A (2004), "Built-in Self-Test Design of Motion Estimation Computing Array", *in Proc. IEEE Northeast Workshop Circuits Syst.*, June, pp. 349-352.

13. Li J F and Hsu C C (2004), "Efficient Testing Methodologies for Conditional Sum Adders", in *Proc. Asian Test Symp.*, pp. 319-324.

14. Lin J F, Yeh J C, Hung R F and Wu C W (2005), "A Built-in Self-Repair Design for RAMs with 2-D Redundancy", *IEEE Trans. Vary Large Scale Integr. (VLSI) Syst.*, Vol. 13, No. 6, pp. 742-745.

15. Li X, Qin J, Huang B, Zhang X and Bernstein J B (2006), "A New SPICE Reliability Simulation Method for Deep Submicrometer CMOS VLSI Circuits", *IEEE Trans. Device Mater. Reliabil.*, Vol. 6, No. 2, pp. 247-257.

16. Liu W Y, Huang J Y, Hong J H and Lu S K (2009), "Testable Design and BIST Techniques for Systolic Motion Estimators in the Transform Domain", in *Proc. IEEE Int. Conf. Circuits Syst.*, April, pp. 1-4.

17. Neubeck K (2004), *Practical Reliability Analysis*, Englewood Cliffs, Pearson Prentice-Hall, New Jersey.

18. Park D K, Cho H M, Cho S B and Lee J H (2007), "A Fast Motion Estimation Algorithm for SAD Optimization in Sub-Pixel", in *Proc. Int. Symp. Integr. Circuits*, September, pp. 528-531.

19. Piestrak S J, Bakalis D and Kavousianos X (2001), "On the Design of Self-Testing Checkers for Modified Berger Codes", in *Proc. IEEE Int. Work-Shop On-Line Testing*, July, pp. 153-157.

20. Portal J M, Aziza H and Nee D (2003), "EEPROM Memory: Threshold Voltage Built in Self Diagnosis", in *Proc. Int. Test Conf.*, September, pp. 23-28.

21. Surin S and Hu Y H (2001), "Frame-Level Pipeline Motion Estimation Array Processor", *IEEE Trans. Circuits Syst. Video Technol.*, Vol. 11, No. 2, pp. 248-251.

22. Vasudevan D P, Lala P K and Parkerson J P (2007), "Self-Checking Carry-Select Adder Design Based on Two-Rail Encoding", *IEEE Trans. Circuits Syst. I, Reg. Papers*, Vol. 54, No. 12, pp. 2696-2705.

23. Wu T H, Tsai Y L and Chang S J (2007), "An Efficient Design-for-Testability Scheme for Motion Estimation in H.264/AVC", in *Proc. Int. Symp. VLSI Design, Autom. Test*, April, pp. 1-4.

24. Yu X, Meng T, Dai Z and Yang X (2006), "Design and Implementation of Reconfigurable Shift Unit Using FPGAs", in *Proc. IEEE Int. Symp. Pervasive Comput. Applic.*, August, pp. 543-545.