Design of Modules to Implement a Structure by Discrete Reckoning Codes for Embedding Into Video Coding Testing Applications

Susrutha Babu Sukhavasi[1], Suparshya Babu Sukhavasi[1] Sr Sastry K[2], Ranga Rao Orugu [3] P.Bosebabu[3], M.Aravind Kumar[4]

[1]Faculty, Department of ECE, K L University, Guntur, AP, India.
[2]M.Tech -VLSI Student, Department of ECE, K L University, Guntur, AP, India.
[3] Faculty, Department of ECE, Andhra Loyola College Of Eng &Tech ,Vijayawada, AP, India.
[4]M.Tech -VLSI Student, Padmasri Dr.B.V.Raju Institute of Technology, Medak, A.P, India.

Abstract : *In this paper, necessary modules have been implemented to make architecture to insert into motion estimation. The significant function 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 modules required for 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 – *Motion Estimation,SAD Tree,Error detection and recover circuits.*

I. Introduction

Methodologies

Coding approaches such as parity code, Berger code, and residue code have been considered for design applications to detect circuit errors. 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, N1 and N2 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 m is the residue N of m modulo . 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. 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. 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.

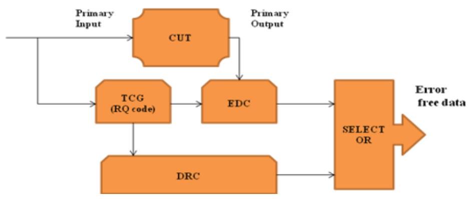


Fig 1: Circuit Diagram

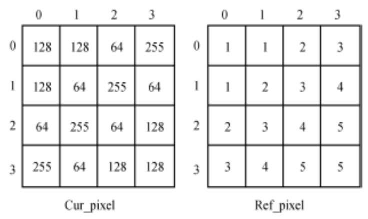
Sum Of Absolute Difference Calculation

By utilizing PEs, SAD shown in as follows, in a macro block with size N X N of can be evaluated:

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

Where rxi,j,qxi,j and ryi,j ,qyi,j denote the corresponding RQ code of Xi,j , Yi,jand modulo M. Importantly, and represent the luminance pixel value of Cur_pixel and Ref_pixel, respectively.

PIXEL VALUES



Cur_pixel

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

Ref_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

Fig 2: Circuit Diagram

NUMERIC CALCULATION

A numerical example of the 16 pixels for a 4X 4 macroblock in a specific PEi of a ME is described . We presents an example of pixel values of the Cur_pixel and Ref_pixel. Based on, the SAD value of the 4X 4 macroblock is

$$\begin{aligned}\mathrm{SAD}&=\sum_{i=0}^{3}\sum_{j=0}^{3}|X_{ij}-Y_{ij}\\&=X_{00}-Y_{00}+|X_{01}-Y_{01}|+\ldots+X_{33}-Y_{33}\\&=(128-1)+(128-1)+\ldots+(128-5)\\&=2124.\end{aligned}$$

According to describe about RQ Code for modulo operation is assumed M=2^6-1=63 RQ code for SAD value RT=R**PEi**=|2124|63=45 and QT=Q**PEi**=|2124/63|=33.

II. MAIN MODULE'S:

1. SAD TREE
2. RQ CODE GENERATION
3. ERROR DETECTION CIRCUIT
4. DATA RECOVERY CIRCUIT

MODULE DESCRIPTION:

2.1 SAD TREE

We propose a 2-D intra-level architecture called the Propagate Partial SAD The architecture is composed of PE arrays with a 1-D adder tree in the vertical direction. Current pixels are stored in each PE, and two sets of continuous reference pixels in a row are broadcasted to PE arrays at the same time. In each PE array with a 1-D adder tree, distortions are computed and summed by a 1-D adder tree to generate one-row SAD. The row SADs are accumulated and propagated with propagation registers in the vertical direction The reference data of searching candidates in the even and odd columns are inputted by Ref. Pixels 0 and Ref Pixels 1, respectively. After initial cycles, the SAD of the first searching candidate in the zero th column is generated, and the SADs of the other searching candidates are sequentially generated in the following cycles. When computing the last searching candidates in each column, the reference data of searching candidates in the next columns begin to be inputted through another reference input. In Propagate Partial SAD, by broadcasting reference pixel rows and propagating partial-row SADs in the vertical direction, it provides the advantages of fewer reference pixel registers and a shorter critical path. Since Rt(Qt) is equal to RPEi (QPEi) EDC is enabled and a signal "0" is generated to describe a situation in which the specific PEi is error-free. Conversely, if SA1 and SA0 errors occur in bits 1 and 12 of a specific , PEi i.e. the pixel values of PEi= 2124. for video coding systems, motion estimation (ME) can remove most of temporal redundancy, so a high compression ratio can be achieved. Among various ME algorithms, a full-search block matching algorithm (FSBMA) is usually adopted because of its good quality and regular computation. In FSBMA, the current frame is partitioned into many small macroblocks (MBs) of size For each MB in the current frame (current MB), one reference block that is the most similar to current MB is sought in the searching range of size in the reference frame. The most common used criterion of the similarity is the sum of absolute differences (SAD).

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

where cur and ref are pixel values in the current MB (current pixel) and reference block (reference pixel), respectively, is one searching candidate in the search range, Distortion is the difference between the current pixel and the reference pixel, and SAD is the total distortion of this searching candidate. The row column) SAD is the summation of distortions in a row (column). After all searching candidates are examined, the searching candidate that has the smallest SAD is selected as the motion vector of the current MB. Although FSBMA provides the best quality among various ME algorithms, it consumes the largest computation power. In general, the computation complexity of ME varies from maximum of a typical video coding system. Hence, a hardware accelerator of ME is required. Variable block-size motion estimation (VBSME) is a new coding technique and provides more accurate predictions compared to traditional fixed block-size motion estimation (FBSME).With FBSME, if an MB consists of two objects with different motion directions, the coding performance of this MB is worse. On the other hand, for the same condition, the MB can be divided into smaller blocks in order to fit the different motion directions with VBSME. Hence, the coding performance is improved. VBSME has been adopted in the latest video coding standards, including H.263 , MPEG-4 , WMV9.0 , and H.264/AVC . For instance, in H.264/AVC, an MB with a variable block size can be divided into seven kinds of blocks including 4x4, 4x8, 8x4, 8x8, 8x16, 16x8, and 16x16. Although VBSME can achieve a higher compression ratio, it not only requires huge computation complexity but also increases the difficulty of hardware implementation for ME. Traditional ME hardware architectures are designed for FBSME, and they can be classified into two categories.

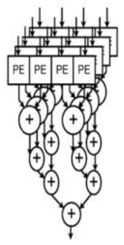


Fig3: Partial SAD tree

One is an inter-level architecture, where each processing element (PE) is responsible for one SAD of a specific searching candidate and the other is an intra-level architecture, where each PE is responsible for the distortion of a specific current pixel .

The concept of the proposed SADTree architecture. The proposed SAD Tree is a 2-D intra-level architecture and consists of a 2-D PE array and one 2-D adder tree with propagation registers Current pixels are stored in each PE, and reference pixels are stored in propagation registers for data reuse. In each cycle, current and reference pixels are inputted to PEs. Simultaneously, continuous reference pixels in a row are inputted into propagation registers to update reference pixels. In propagation registers, reference pixels are propagated in the vertical direction row by row. In SAD Tree architecture, all distortions of a searching candidate are generated in the same cycle, and by an adder tree, distortions are accumulated to derive the SAD in one cycle. In order to provide a high utilization and data reuse, the snake scan is adopted and reconfigurable data path propagation registers are developed in the proposed SAD Tree, which consists of five basic steps from A to E. The first step, A, fetches pixels in a row and the shift direction of propagation registers is downward. When calculating the last candidates in a column, one extra reference pixel is required to be inputted, that is, step B. When finishing the computation of one column, the reference pixels in the propagation registers are shifted left in step C. Because the reference data have already been stored in the propagation registers, the SAD can be directly calculated. The next two steps, D and E, are the same as steps A and B except that the shift direction is upward. After finishing the computation of one column in the search range, we execute step C and then go back to step A. This procedure will iterate until all searching candidates in the search range have been calculated. the data reuse between two successive searching candidates can be maximized

SUB MODULES'S

- PROCESSING ELEMENT
- ADDER TREE
- HALF ADDER
- FULL ADDER
- RIPPLE CARRY ADDER

2.1.1 PROCESSING ELEMENT

A ME (Motion Estimation) 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 Notably, some registers and latches may exist in ME to complete the data shift and storage. encoding applications. Notably, some registers and latches may exist in ME to complete the data shift and storage. 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). 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. 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. 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-per-pin 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). Motion estimation is the process of determining motion vectors that describe the transformation from one 2D image to another; usually from adjacent frames in a video sequence. It is an ill-posed problem as the motion is in three dimensions but the images are a projection of the 3D scene onto a 2D plane. The motion vectors may relate to the whole image (global motion estimation) or specific parts, such as rectangular blocks, arbitrary shaped patches or even per pixel. The motion vectors may be represented by a translational model or many other models that can approximate the motion of a real video camera, such as rotation and translation in all three dimensions and zoom. Closely related to motion estimation is optical flow, where the vectors correspond to the perceived movement of pixels. In motion estimation an exact 1:1 correspondence of pixel positions is not a requirement. Applying the motion vectors to an image to synthesize the transformation to the next image is called motion compensation. The combination of motion estimation and motion compensation is a key part of video compression as used by MPEG 1, 2 and 4 as well as many other video codec's.

2.1.2 ADDER TREE

In this module Half Adder is a digital combinational circuit that is used for the addition of two bits and provides an output in the form of a sum bit and a carry bit. The logical functional equations that relate the outputs S and C of a half adder circuit to the input bits are given below

Sum(S) = A ex-OR B

Carry(C) = A.B

Thus a half adder circuit can easily be synthesized by using 1 ex-OR gate and 1 AND gate. Since a half adder circuit can only be used to add two bits, it becomes obsolete in case of multi-bit addition in practical applications.

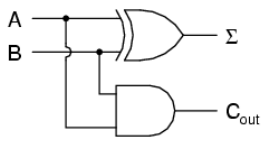
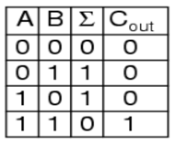


Fig 4: Logic Diagram



A	B	Σ	C_{out}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Fig5: Truth Table

APPLICATIONS:

- In electronics, an adder or summer is a digital circuit that performs addition of numbers. In many computers and other kinds of processors, adders are used not only in the arithmetic logic unit(s), but also in other parts of the processor, where they are used to calculate addresses, table indices, and similar.
- Although adders can be constructed for many numerical representations, such as binary-coded decimal or excess-3, the most common adders operate on binary numbers.
- In cases where two's complement or ones' complement is being used to represent negative numbers, it is trivial to modify an adder into an adder–subtractor. Other signed number representations require a more complex adder.

2.1.3 FULL ADDER:

In this module describe about full adder operation. The limitation of a half-adder is that it cannot accept a carry-in bit. the carry-in bit represents the carry-out of the previous low-order bit position. Thus a half-adder can be used only for the two least significant digits when adding two multi bit binary numbers, since there can be no possibility of a propagated carry to this stage. In multi bit addition, a carry bit from a previous stage must be taken into account, which gives rise to the necessity for designing a full adder. A full adder can accept two operands bits, ai and bi, and a carry-in bit ci from previous stage; it produces a sum bit si and a carry-out bit c0. sum bit si is 1 if there is an odd number of 1's at the inputs of the full adder, whereas the carry-out c0 is 1 if there are two or more 1's at the inputs. The sum and carry out bits will be 0 otherwise. In ripple carry adder, the carry signals must ripple through all the full adders before the outputs stabilize to the correct values; hence such an adder is often called a ripple adder. addition is to be performed the carry-out generated from the least significant stage of the adder propagates through the successive stages and produces a carry-in into the most significant stage of the adder. The time required to perform addition in a ripple adder depends on the time needed for the propagation of carry signals through the individual stages of the adder. Thus ripple carry addition is not instantaneous. The greater the number of stages in a ripple carry adder the longer is the carry propagation time, and consequently the slower the adder.

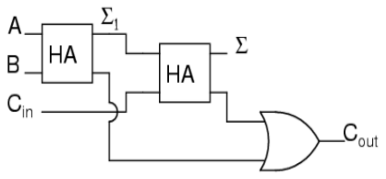


Fig 6: Full Adder Using Half Adder

C_1	X_1	Y_1	Z_1	C_2
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

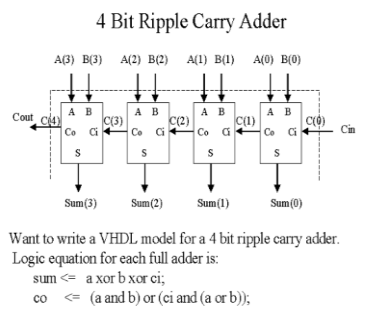
Fig7: Truth table

APPLICATIONS:

1. Addition is an essential function in fundamental arithmetic operations.
2. It is also the most copiously used operation in application-specific processors and digital signal processing application (DSP).
3. Full-adder has been introduced by integrating the full-adder into a multiplier-less finite impulse response (FIR) filter that is commonly used in the multi rate filter bank for biomedical applications.

2.1.4 RIPPLE CARRY ADDER

A standard 8-bit ripple-carry adder built as a cascade from eight 1-bit full-adders. Click the input switches or use the following bind keys: ('c') for carry-in, for A0..A7 and ('1','2', ..., '8') for B0..B7. To demonstrate the typical behavior of the ripple-carry adder, very large gate-delays are used for the gates inside the 1-bit adders - resulting in an addition time of about 0.6 seconds per adder. Note that each stage of the adder has to wait until the previous stage has calculated and propagates its carry output signal. Obviously, the longest delay results for operands like A = 0b0000000, B=0b11111111 or A=0b01010101 and B=0b10101010 (select these, and then switch carry-in to both 0 and 1, and watch the circuit to settle). Therefore, the total delay of a ripple-carry adder is proportional to the number of bits. Faster adders are often required for bit widths of 16 or greater. A carry-look ahead adder (CLA) is a type of adder used in digital logic. A carry-look ahead adder improves speed by reducing the amount of time required to determine carry bits. It can be contrasted with the simpler, but usually slower, ripple carry adder for which the carry bit is calculated alongside the sum bit, and each bit must wait until the previous carry has been calculated to begin calculating its own result and carry bits (see adder for detail on ripple carry adders). The carry-look ahead adder calculates one or more carry bits before the sum, which reduces the wait time to calculate the result of the larger value bits. The Kogge-Stone adder and Brent-Kung adder are examples of this type of adder.Arithmetic operations like addition, subtraction, multiplication, division are basic operations to be implemented in digital computers using basic gates likr AND, OR, NOR, NAND etc. Among all the arithmetic operations if we can implement addition then it is easy to perform multiplication (by repeated addition), subtraction (by negating one operand) or division (repeated subtraction).Half Adders can be used to add two one bit binary numbers. It is also possible to create a logical circuit using multiple full adders to add N-bit binary numbers. Each full adder inputs a **Cin,** which is the **Cout** of the previous adder. This kind of adder is a **Ripple Carry Adder**, since each carry bit "ripples" to the next full adder. The first (and only the first) full adder may be replaced by a half adder.The block diagram of 4-bit Ripple Carry Adder is shown here below The layout of ripple carry adder is simple, which allows for fast design time; however, the ripple carry adder is relatively slow, since each full adder must wait for the carry bit to be calculated from the previous full adder. The gate delay can easily be calculated by inspection of the full adder circuit. Each full adder requires three levels of logic. In a 32-bit [ripple carry] adder, there are 32 full adders, so the critical path (worst case) delay is 31 * 2(for carry propagation) + 3(for sum) = 65 gate delays. Arithmetic Module Generator (AMG) supports various hardware algorithms for two-operand adders and multi-operand adders. These hardware algorithms are also used to generate multipliers, constant-coefficient multipliers and multiply accumulators. In the following, we briefly describe the hardware algorithms that can be handled by AMG.



Want to write a VHDL model for a 4 bit ripple carry adder.
Logic equation for each full adder is:

```
sum <=  a xor b xor ci;
co  <=  (a and b) or (ci and (a or b));
```

Fig8: Ripple Carry Adder

The most straightforward implementation of a final stage adder for two n-bit operands is a ripple carry adder, which requires n full adders (FAs). The carry-out of the ith FA is connected to the carry-in of the (i+1)th FA. Figure shows a ripple carry adder for n-bit operands, producing n-bit sum outputs and a carry out.

III. RQ CODE GENERATION

Coding approaches such as parity code, Berger code, and residue code have been considered for design applications to detect circuit errors . 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. The basic problem we have to resolve is that memory and communications technology isn't totally reliable; we have to expect and be ready to deal with errors in the hardware. This document will describe two very different technologies for detecting, and maybe correcting, errors that may occur in data storage and transmission. The first approach to be described is more appropriate for environments like memory: a relatively small amount of data is fetched in parallel. This approach, called "error detecting and correcting codes," is based on defining a distance between two bit strings in terms of the number of bits that have to change to get from the first string to the second. Extra bits are added to each string, which are set so that some minimum number of bits must change to get from one valid string to another. If the received string isn't valid, it is assumed that the correct string is the one "closest" to the received string. The second approach is more appropriate to environments in which relatively large amounts of data are to be transferred, but they are transferred serially. In this approach a "signature" is appended to the data string; the number of bits in the signature is much less than the number of bits that would be required to do an error correcting code. This approach will lead to adding checksums or cyclic redundancy checks to the string. R. W. Hamming wrote the paper that both opened and closed this field in 1950. His interest was in providing a means of self-checking in computers, which were just being developed at the time he wrote this. the paper appeared in the *Bell System Technical Journal*, April, 1950. Definitely worth tracking down in the library and reading The Hamming distance between two bit strings is the number of bits you have to change to convert one to the other: this is the same as the number of edges you have to traverse in a binary hypercube to get from one of the vertices to the other. The basic idea of an error correcting code is to use extra bits to increase the dimensionality of the hypercube, and make sure the Hamming distance between any two valid points is greater than one.

- If the Hamming distance between valid strings is only one, a single-bit error results in another valid string. This means we can't detect an error.
- If it's two, then changing one bit results in an invalid string, and can be detected as an error.

Unfortunately, changing just one more bit can result in another valid string, which means we can't know which bit was wrong: so we can detect an error but not correct it.

3.1 CONCURRENT ERROR DETECTION (CED)

While the extended BIST schemes generally focus on memory circuit, testing-related issues of video coding have seldom been 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).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. 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’t 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 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. Concurrent test methods enable integrated circuits to verify the correctness of their results during normal operation. While this ability is highly desirable, especially in high safety applications, designing a cost-effective concurrently testable circuit is a challenging task. Quality assessment of concurrent test methods relies on several parameters, including the model of detectable faults or errors, the worst-case detection latency, and the incurred area overhead. Additionally, an important consideration is whether a concurrent test method is intrusive or non-intrusive, i.e. whether the original circuit is modified or left intact, respectively. The importance of concurrent test in only accentuated by the plethora and variety of previous research efforts in this area. Several low-cost, non-intrusive, concurrent fault detection (CFD) methods have been proposed for stuck-at faults in combinational circuits. C-BIST employs input monitoring to perform concurrent self-test. While hardware overhead is very low, the method relies on an ordered appearance of all possible input vectors before a signature indicating circuit correctness can be calculated, resulting in very long detection latency. This problem is alleviated in R-CBIST, where the requirement for a uniquely ordered appearance of all input combinations is relaxed at the cost of a small RAM. Alternatively, latency is reduced through the comparison-based method which uses additional logic to predict the circuit responses for a complete test set. Towards the high-cost end, several concurrent error detection (CED) zero-latency methods have been proposed for both combinational and sequential circuits. Reducing the area overhead below the cost of duplication typically requires redesign of the original circuit, thus leading to intrusive methodologies. In more and more applications, cryptographic operations are performed on embedded processors. Some of the most important applications in this context are payment, identification, access control, digital rights management and

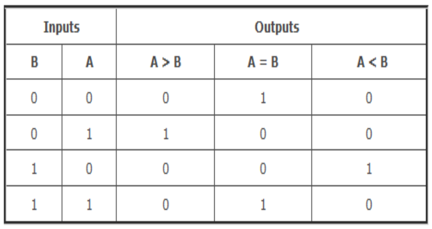
IP protection. In order to guarantee the security of these applications, it is necessary to implement countermeasures against physical attacks on the embedded processors. During the last 15 years numerous physical attacks have been published that allow the extraction of secret information based on the observation or manipulation of an embedded device and its environment. Typical examples of physical attacks are timing attacks, power analysis attacks and fault attacks . While timing and power analysis attacks have received much attention already immediately after their publication, not so much attention has been paid on fault attacks so far. However, fault attacks become increasingly important. Meanwhile there exist several publication that discuss methods to induce faults in order to reveal secret information. In general there are two types of fault attacks.

SUBMODULES

- COMPARATOR
- PRIORITY ENCODER
- MULTIPLEXER
- SUBTRACTOR
- QUASI BLOCK

3.1.1 COMPARATOR

A digital comparator or magnitude comparator is a hardware electronic device that takes two numbers as input in binary form and determines whether one number is greater than, less than or equal to the other number. Comparators are used in a central processing units (CPU) and microcontrollers. Examples of digital comparator include the CMOS 4063 and 4585 and the TTL 7485 and 74682-'89. The analog equivalent of digital comparator is the voltage comparator. Many microcontrollers have analog comparators on some of their inputs that can be read or trigger an interrupt. A digital comparator is an electronic circuit or device capable of accepting two binary signals and performing tests on those signals to determine their equivalence to each other. The simplest form of a digital comparator compares two binary signals, known in computer processing as bits, and uses a series of logical gates to determine if the two bits are equal or if one is greater than the other based on binary logic in which each signal is given the value of either zero or one. Most digital comparator circuits are designed to accept multiple bits for comparison, where in many applications the bits are combined by external software or hardware into actual numbers. At the heart of most central processing units (CPUs) in computers and other digital devices, a comparator performs a large portion of the logical operations that allow a computer function. Outside of computers, digital comparators also are used in some devices in which analog input is converted into digital information that is measured or monitored, such as in some testing meters. The way a digital comparator functions starts with the input of information. The comparator can only handle binary data, meaning that whatever the input mechanism is, the signal coming into the circuit can only have two states, which commonly are referred to as zero and one. When a bit is compared to another bit, it can be tested in one of three ways by the digital comparator. The first is equivalency, meaning the result of comparing one bit to another will result in a positive, or true, result if both of the bits equal one or if both of the bits equal zero. Individual bits also can be checked to see of one is greater than or less than another. For a sequence of bits, however, comparisons to determine which set has a higher or lower value need to check each bit to see which set has a more highly placed most significant bit, because this determines the actual numerical value of the bit set. Beyond computer processors, a digital comparator can be used in some devices that contrast visual images with digital images, as can be the case in engineering that relies on computer-aided drafting (CAD) programs to check if the physically manufactured products match specifications. They also can be employed to convert analog signals into digital patterns. A digital comparator also can be used in conjunction with a number of other devices to act as a monitor in an industrial setting to see accurate digital information about the state of a machine.



Inputs		Outputs		
B	A	A > B	A = B	A < B
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

Table 1:1-bit Comparator

This is useful if we want to compare two variables and want to produce an output when any of the above three conditions are achieved. For example, produce an output from a counter when a certain count number is reached. Consider the simple 1-bit comparator above.

3.1.2 PRIORITY ENCODER

Here we applied input width is 5 bit and output width is 3 bit by using priority encoder. A priority encoder is a circuit or algorithm that compresses multiple binary inputs into a smaller number of outputs. The output of a priority encoder is the binary representation of the ordinal number starting from zero of the most significant input bit. They are often used to control interrupt requests by acting on the highest priority request. If two or more inputs are given at the same time, the input having the highest priority will take precedence. An example of a single bit 4 to 2 encoder is shown, where highest-priority inputs are to the left and "x" indicates an irrelevant value - i.e. any input value there yields the same output since it is superseded by higher-priority input. The output V indicates if the input is valid. Priority encoders can be easily connected in arrays to make larger encoders, such as one 16-to-4 encoder made from six 4-to-2 priority encoders - four 4-to-2 encoders having the signal source connected to their inputs, and the two remaining encoders take the output of the first four as input. The priority encoder is an improvement on simple encoder circuit in terms of all handling possible input configurations.

3.1.3 MULTIPLEXER

In this module will generate reminder code according to selection line. We applied three input signal as mux_in and m. we obtain the output (specific signal) from given input signal.

If A greater than B is equal to 1 will get mux_in signal. If A lesser than B we get m signal. Otherwise we get 0(8 bit signal). A data selector, more commonly called a **Multiplexer,** shortened to "Mux" or "MPX", are combinational logic switching devices that operate like a very fast acting multiple position rotary switch. They connect or control, multiple input lines called "channels" consisting of either 2, 4, 8 or 16 individual inputs, one at a time to an output. Then the job of a multiplexer is to allow multiple signals to *share* a single common output. For example, a single 8-channel multiplexer would connect one of its eight inputs to the single data output. Multiplexers are used as one method of reducing the number of logic gates required in a circuit or when a single data line is required to carry two or more different digital signals.

3.1.4 SUBTRACTOR

Here we calculate the difference between input signal and specific constant value with help of subtraction. This model converts two rotations A and B into their difference A-B. This is useful for various applications. For instance, when you build a treaded vehicle such as a construction bulldozer, you'd like one motor to control total motion, and the other the turning. This construction does that: connect the treads to the (A+B) and (A-B) axles, and the motors to A and B. Now motor A makes the vehicle go forward or backward, and B turns it left or right. Of course you can also run them simultaneously. The great advantage is that you can now guarantee that the two sides of the treaded vehicle run at the same speed that does not happen with 2 motors, they always have a slight power difference. So it is more likely that it goes straight when you want it to.

3.1.5 QUASI BLOCK

This block will generate quotient value according to given input. Here we applied 3 bit input then generate 8 bit signal as output. Coding approaches such as parity code, Berger code, and residue code have been considered for design applications to detect circuit errors. 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.

3.1.6 ACCUMULATOR

In this module consists flip-flop act as a accumulator. We can store a bit of data. Flip-flop" is the common name given to two-state devices which offer basic memory for sequential logic operations. Flip-flops are heavily used for digital data storage and transfer and are commonly used in banks called "registers" for the storage of binary numerical data. There are some circuits that are not quite as straight forward as the gate circuits we have discussed in earlier lessons. However, you still need to learn about circuits that can store and remember information. They're the kind of circuits that are used in computers to store program information RAM memory.

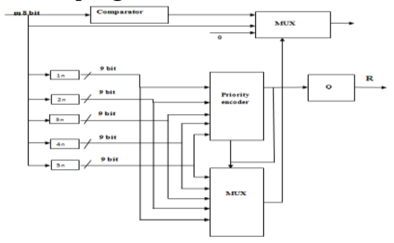


FIG 9: M MOD N OPERATION

IV. ERROR DETECTION CIRCIUT

In this module indicates that the operations of error detection in a specific PEi is achieved by using EDC, which is utilized to compare the outputs between TCG and in rder to determine whether errors have occurred. The EDC output is then used to generate a 0/1 signal to indicate that the tested PEi is error-free/errancy. Using XOR operation can be identify the error if any variation in terms of residue and quotient value. Because a fault only affects the logic in the fanout cone from the fault site, the good circuit and faulty circuits typically only differ in a small region. Concurrent fault simulation exploits this fact and simulates only the differential parts of the whole circuit . Concurrent fault simulation is essentially an event-driven simulation with the fault-free circuit and faulty circuits simulated altogether. In concurrent fault simulation, every gate has a concurrent fault list, which consists of a set of bad gates. A bad gate of gate x represents an imaginary copy of gate x in the presence of a fault. Every bad gate contains a fault index and the associated gate I/O values in the presence of the corresponding fault. Initially, the concurrent fault list of gate x contains local faults of gate x. The local faults of gate x are faults on the inputs or outputs of gate x. As the simulation proceeds, the concurrent fault list contains not only local faults but also faults propagated from previous stages. Local faults of gate x remain in the concurrent fault list of gate x until they are detected. As we move to the nanometer age, we have begun to see nanometer designs that contain hundreds of millions of transistors.

V. DATA RECOVERY CIRCUIT

In this module will be generate error free output by quotient multiply with constant value (64) and add with reminder code. During data recovery, the circuit DRC plays a significant role in recovering RQ code from TCG.

Notably, the proposed EDDR design executes the error detection and data recovery operations simultaneously. Additionally, error-free data from the tested PEi or data recovery that results from DRC is selected by a multiplexer (MUX) to pass to the next specific PEi+1for subsequent testing. Error concealment in video is intended to recover the loss due to channel noise, e.g., bit-errors in a noisy channel and cell- loss in an ATM network, by utilizing available picture information. The error concealment techniques can be categorized into two classes according to the roles that the encoder and the decoder play in the underlying approaches. Forward error concealment includes methods that add redundancy in the source to enhance error resilience of the coded bit streams. For example, I-picture motion vectors were introduced in MPEG-4 to improve the error concealment. However, a syntax change is required in this scheme. In contrast to this approach, error concealment by post-processing refers to operations at the decoder to recover the damaged images based on image and video characteristics. In this way, no syntax is needed to support the recovery of missing data. we have only discussed the case in which one frame has been damaged and we wish to recover damaged blocks using information that is already contained in the bit-stream.

The temporal domain techniques that we have considered rely on information in the previous frame to perform the reconstruction. However, if the previous frame is heavily damaged, the prediction of the next frame may also be affected. For this reason, we must consider making the prediction before the errors have occurred. Obviously, if one frame has been heavily damaged, but the frame before that has not been damaged, it makes senses to investigate how the motion vectors can be extrapolated to obtain a reasonable prediction from a past reference frame. Following this notion, we have essentially divided the problem of error concealment into two parts. The first part assumes that the previous frames are intact or are close to intact. This will always be the case for low BER and short error bursts.

Furthermore, a localized solution such as the techniques presented in the previous subsection will usually perform well. However, if the BER is high and/or the burst length is long, the impact of a damaged frame can propagate, hence the problem is more global and seems to require a more advanced solution, i.e., one which considers the impact over multiple frames. In the following, we propose an approach that considers making predictions from a past reference frame, which has not been damaged. The estimated motion information which differs from the actual one may be recovered from that of neighbor blocks. Because a moving object in an image sequence is larger than the block size of a minimal block in many occasions, motion information of neighbor blocks are usually the same as, or approximate to, current blocks. The concept of global motion is discussed in many researches on motion estimation or related interests. In method which reconstructs the frame with the aid of neighbor motion vector is successfully applied to motion estimation. Thus, an error signal "1" is generated from EDC and sent to mux in order to select the recovery results from DRC.

VI. SIMULATION RESULTS

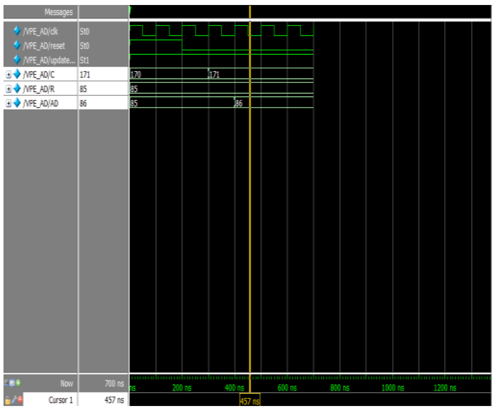


Fig10: Processing Element

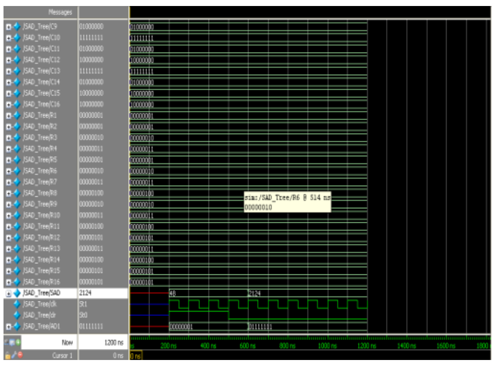


Fig11: SAD TREE

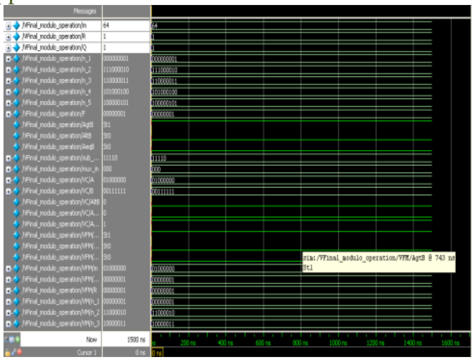


Fig12: RQ Code Generation

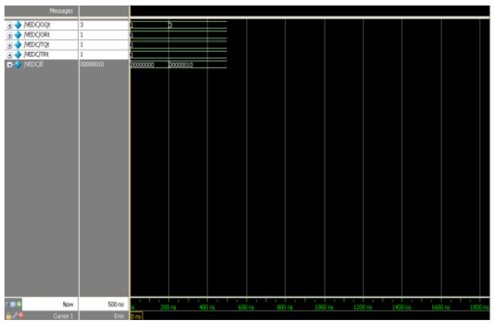


Fig13: Error Detection Circuit

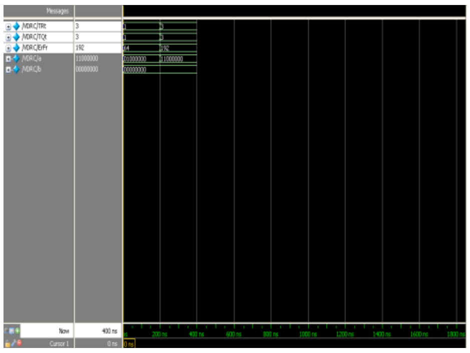


Fig14: Error Recover Circuit

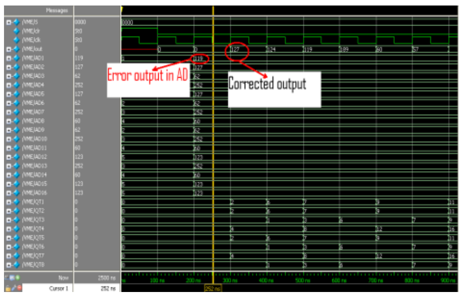


Fig15: Motion Estimation

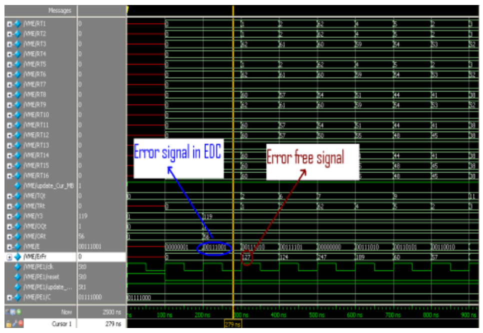


Fig16: Motion Estimation

VII. CONCLUSION

This paper presents an FPGA implementation of efficient architecture to make an application embedding into video code testing applications .The required modules have been taken and theoretical analysis have done and numerical calculations were calculated and verified. Combinational circuits are taken and simulation results have obtained which are required to built an architecture are explained and necessary types of codes which are generated to make errors to be detected and recovered.

Referrences

[1] J. M. Portal, H. Aziza, and D. Nee, "EEPROM memory: Threshold voltage built in self diagnosis," in Proc. Int. Test Conf., Sep. 2003, pp. 23–28.

[2] J. F. Lin, J. C. Yeh, R. F. Hung, and C. W. Wu, "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, Jun. 2005.

[3] C. L. Hsu, C. H. Cheng, and Y. Liu, "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, Feb. 2010.

[4] C. H. Cheng, Y. Liu, and C. L. Hsu, "Low-cost BISDC design for motion estimation computing array," in Proc. IEEE Circuits Syst. Int. Conf., 2009, pp. 1–4.

[5] S. Bayat-Sarmadi and M. A. Hasan, "On concurrent detection of errors in polynomial basis multiplication," IEEE Trans. Vary Large Scale Integr. (VLSI) Systs., vol. 15, no. 4, pp. 413–426, Apr. 2007.

[6] C. W. Chiou, C. C. Chang, C. Y. Lee, T. W. Hou, and J. M. Lin, "Concurrent error detection and correction in Gaussian normal basis multiplier over GF 2^M," IEEE Trans. Comput., vol. 58, no. 6, pp. 851–857, Jun. 2009.

[7] L. Breveglieri, P. Maistri, and I. Koren, "A note on error detection in an RSA architecture by means of residue codes," in Proc. IEEE Int. Symp. On-Line Testing, Jul. 2006, 176 177.

[8] S. J. Piestrak, D. Bakalis, and X. Kavousianos, "On the design of selftesting checkers for modified Berger codes," in Proc. IEEE Int. WorkshopOn- Line Testing, Jul. 2001, pp. 153–157.

[9] S. Surin and Y. H. Hu, "Frame-level pipeline motion estimation array processor," IEEE Trans. Circuits Syst. Video Technol., vol. 11, no. 2, pp. 248–251, Feb. 2001.

[10] D. K. Park, H. M. Cho, S. B. Cho, and J. H. Lee, "A fast motion estimation algorithm for SAD optimization in sub-pixel," in Proc. Int. Symp. Integr. Circuits, Sep. 2007, pp. 528–531.

[11] L. Breveglieri, P. Maistri, and I. Koren, "A note on error detection in an RSA architecture by means of residue codes," in Proc. IEEE Int. Symp. On-Line Testing, Jul. 2006, 176 177.

[12] S. J. Piestrak, D. Bakalis, and X. Kavousianos, "On the design of selftesting checkers for modified Berger codes," in Proc. IEEE Int. WorkshopOn- Line Testing, Jul. 2001, pp. 153–157.

Authors

Suparshya Babu Sukhavasi was born in India, A.P. He received the **B.Tech** degree from JNTU, A.P, and **M.Tech** degree from SRM University, Chennai, Tamil Nadu, and India in 2008 and 2010 respectively. He worked as **Assistant Professor** in Electronics & Communications Engineering in Bapatla Engineering College for academic year 2010-2011 and from 2011 to till date working in **K L University**. He is a member of Indian Society For Technical Education and International Association of Engineers. His research interests include Mixed and Analog VLSI Design, FPGA Implementation, Low Power Design and Wireless communications, VLSI in Robotics. He has published articles in various international journals and Conference in IEEE.

Susrutha Babu Sukhavasi was born in India, A.P. He received the **B.Tech** degree from JNTU, A.P, and **M.Tech** degree from SRM University, Chennai, Tamil Nadu, India in 2008 and 2010 respectively. He worked as **Assistant Professor** in Electronics & Communications Engineering in Bapatla Engineering College for academic year 2010-2011 and from 2011 to till date working in **K L University**. He is a member of Indian Society For Technical Education and International Association of Engineers. His research interests include Mixed and Analog VLSI Design, FPGA Implementation, Low Power Design and wireless Communications, Digital VLSI. He has published articles in various international journals and Conference in IEEE.

S R Sastry Kalavakolanu was born in A.P,India. He received the B.Tech degree in Electronics & communications Engineering from Jawaharlal Nehru Technological University in 2010. Presently he is pursuing M.Tech VLSI Design in KL University. His research interests include Low Power VLSI Design. He has undergone 3 International Journals and 1 publishment in IEEE.

M.Aravind Kumar was born in A.P,India He received his B.Tech degree in Electronics and Communication Engineering from S.V.H college of Engineering and Technology in the year 2005. He is presently pursuing masters in VLSI system design (4th semester) at Padmasri Dr.B.V.Raju Institute of Technology, Medak(dst), A.P.India.

P.Bose Babu was born in A.P,India, He Completed M.Tech in VLSI system Design from MVGR COLLEGE OF ENGG.&TECH, Vizianagaram in 2011 and B.Tech from QIS College of engineering, in 2009 in Electronics &Communication engineering. Presently he is Working as Asst.Professor in ANDHRA LOYOLA college of Engg.& Technology, Vijayawada,A.P,India.

Ranga Rao Orugu was born in A.P,India, Completed M.Tech in Communication Systems at C.R.Reddy engineering College, Eluru and B.Tech from Sri Saaradhi Institute of and technology in the year 2009 in Electronics &Communication engineering. Presently he is Working as Asst.Professor in ANDHRA LOYOLA college of Engg.& Technology, Vijayawada,A.P,India.