Efficient CA-Based Byte Error Correcting Codes Using Parallel Processing

Francily C Francis[1] Suvitha P S[2]
[1,2]Department of Electronics and Communication Engineering
[1,2]IES College of Engineering, Thrissur, Kerala, India

Abstract— Cellular Automata (CA) is a tool used to generate byte error correcting code. CA-based byte Error Correcting Code (ECC) which overcomes the weaknesses and limitation of existing scheme. Proposed byte ECC scheme can correct errors when errors are distributed within information and check bytes or concentrated in any one of them. Cellular automata are already employed by several researchers for designing bit and byte error detecting and correcting codes. CA based VLSI design is more attractive because of its modular, regular and cascadable construction. RS codes are popularly used to detect and correct burst and as well as random errors in different communication systems and storage mediums. Simple and modular architecture of cellular automata based encoder and syndrome generator are proposed by employing the regular structure of cellular automata. Proposed design has been optimized using a parallel processing technique. In this work, a new decoding logic circuit has been introduced for error correction. Proposed encoder and syndrome generator circuits have less computation time compared to existing cellular automata based designs. All functional blocks are simulated and synthesized using FPGA based Xilinx ISE simulator.

Keywords: Soft Errors, Memory, Error Correction Codes (ECC), Single Bit Error Correction-Double Bits Error Detection Codes, Reed Solomon (RS) Codes, Single Byte Error Correcting Reed Solomon (SEC-RS) Code, Cellular Automata (CA), FPGA, ASIC

I. INTRODUCTION

Error correcting codes have a wide range of applications in digital data communications, memory system design, fault tolerant computer design etc. Reed Solomon(RS)code is a well-known non-binary, block code and popularly used for error correction in many applications like wireless communication, high speed modems and storage devices (CD,DVD).Reliability of memory system is reduced due to soft errors which are occurred by radiation particles and external noises. Logical values of memory elements: latch, flip-flop, register are changed when alpha and cosmic particles hit them. This changes can also be occurred due to atmospheric variations. This occurrence is known as Single Event Upset (SEU) or Multiple Bit Upset (MBU) or Multiple Cell Upset (MCU).

Single bit error correction-double bits errors detection codes are popularly used in memory system. These codes can be constructed using extended Hamming codes and Hsiao code. Single bit error correction-double bits errors detection codes are combined with interleaving to ensure that the errors affect only a single byte per logical word. But use of interleaving code for memory design increases area and power consumption. A very simple and fast single bit error correction-double bits errors detection-double bits adjacent errors correction codes are introduced based on orthogonal Latin square codes in [12].

Difference Set (DS) codes are also employed for memory protection where the codes are decoded by an iterative algorithm to detect and correct errors. In [27], a different approach is used where a simple Hamming codes are used for error detection and a more powerful BCH codes are used for error correction. There are other error correcting codes like Low Density Parity Check (LDPC) codes, Reed Solomon (RS) codes etc which are used to protect the memory systems.

RS codes are non-binary error correcting codes which can detect and correct multiple bytes of errors in memory systems. Recently, error detection and correction delays of RS codes for memory are reduced over traditional RS codes by increasing the overhead area complexity in Pontarelli et al. [26]. But authors did not mentioned about the power consumption and Area Delay Product (ADP) of codec. Cellular Automata (CA) based error correcting codes are useful for saving the hardware. A compact and power efficient improved CA-based double byte error correcting codes is designed and implemented in [33]. Also recently, a compact CA-based three byte error detecting codes are proposed using local and global optimization algorithm in [10]. In [33] [10], authors have given their full concentration for area reduction using CA without bothering about its delay. However, the encoding and decoding delays are reduced in this proposed codec by using parallel processing technique.

A SEC codec using CA is designed and implemented. Major contributions of this project are (a) proposal for a CASEC codec and (b) design and implementation both in FPGA platform. Proposed design is compact as well as efficient and computation time is minimum compared to existing related works. Synthesis results show that a significant time reduction is achieved using parallel processing technique.

II. EXISTING SINGLE ERROR CORRECTING REED SOLOMON (SEC-RS) CODES

RS codes are non-binary codes that are used in many digital transmission and memory systems. RS encoding and decoding are defined over Galois Feld (GF). RS (n,k,m) codes have codeword length of n symbols and message length of k symbols, where each symbol of size m bits. These codes consist of (n − k) or 2t number of parity check symbols. Minimum distance of separation between two distinct codewords is $d_{min} = 2t + 1$. For a SEC-RS codes, t is equal to one and minimum distance (d_{min}) is 3. In this section, basics of existing SEC-RS codes are discussed. SECRS codes can detect and correct maximum one byte error. For the sake of completeness, two related coding schemes are discussed in the following subsection.

A. Conventional SEC-RS Codes (SEC-RS)

For a traditional t-byte error correcting RS codes defined over GF(2^8), parity matrix (P) [34] can be written in the following form.

$$P = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \alpha & \alpha^2 & \cdots & \alpha^{k-1} \\ 1 & (\alpha^2) & (\alpha^2)^2 & \cdots & (\alpha^2)^{k-1} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & (\alpha^{2t}) & (\alpha^{2t})^2 & \cdots & (\alpha^{2t})^{k-1} \end{bmatrix}$$

where α is a primitive element over GF(2^8). The parity matrix (P) [34] of a conventional SEC-RS codes are as follows.

$$P = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \alpha & \alpha^2 & \cdots & \alpha^{k-1} \end{bmatrix}$$

Two parity check symbols (P0, P1) can be computed from the parity matrix (P). For a conventional SEC-RS codes [19] the parity check matrix (H) can be written in the following form.

$$H = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 & 1 & 0 \\ 1 & \alpha & \alpha^2 & \cdots & \alpha^{k-1} & 0 & 1 \end{bmatrix}$$

Syndromes can be calculated from this H matrix directly. The syndromes S=[S0,S1]t of a SEC-RS codes are computed from received codeword (r) using the following relationship. S=H.r.

Where r is a column matrix of size n. When all the syndromes are zero, then there is no error in received codeword. Existence of at least one non-zero syndrome indicates error in received codeword and therefore error correction has to be performed. From matrix H, it is noted that in case of conventional SEC RS codes, 1st and 2nd rows have unbalanced computation time. The computation of syndrome S0 by employing the 1st row of H matrix does not require any multiplication over finite field. Whereas (k−1) numbers of multiplications are needed to compute the other syndrome S1. In this context, Pontarelli et al. [8] modified the H matrix and reduced the overall delay of the SEC-RS code, which is known as modified SEC-RS code. These modified codes are described in the following subsection.

Modified SEC-RS Codes (SEC-RS_{mod1})

Pontarelli et al. [8], proposed a modified RS code to reduce delay. For a modified SEC-RS code(SEC-RS_{mod1})[8] parity check matrix (H) is as follows.

$$H = \begin{bmatrix} 1 & 1 & \alpha^{-2} & 1 & \cdots & \alpha^{-(k-1)} & 1 & 0 \\ 1 & \alpha & 1 & \alpha^3 & \cdots & 1 & 0 & 1 \end{bmatrix}.$$

In this case equal number of multiplications are needed to compute each of the parity check (P) and syndrome (S) symbols during the encoding and decoding process respectively. This modification reduces critical path and therefore lowers the delay. Decoding of the modified SEC codes [8] is similar to conventional RS codes. Pontarelli et al. [8], also proposed another modification in SEC-RS codes and denoted as 'SECRS$_{mod2}$' code.

$$H_m = \begin{bmatrix} \beta & 1 & 1 & \beta^2 & 1 & 1 & \cdots & 1 & 0 & 0 \\ 1 & \beta & 1 & 1 & \beta^2 & 1 & \cdots & 0 & 1 & 0 \\ 1 & 1 & \beta & 1 & 1 & \beta^2 & \cdots & 0 & 0 & 1 \end{bmatrix}$$

where β is a primitive element of GF(2^4). Using this parity check matrix, SEC-RS(19,16,4) codec is designed over GF(2^4) field. In this paper, the area complexity of the SEC is further reduced using Cellular Automata (CA).

III. PROPOSED CA BASED SEC(CASEC)

A brief overview of CA and its application in encoding and decoding techniques of CASEC are discussed here.

A. Error Correcting Codes

Because of some interferences, whenever the Information Bits (IB) flow from one point to another, they are subjected to unpredictable changes. The use of ECC to eliminate the necessity of re transmitting the data is called Forward Error Correction (FEC). The basic concept includes systematically adding some redundancies to messages at the encoder, such that the decoder can successfully recover the messages from the received blocks, possibly corrupted by the channel noise. A code denoted by C(n,k,d) is a linear code in which, n is the length of code word, k is the number of data symbols in a code word and d is the minimum distance of C. The code C can detect up to t errors if $d(C) \geq t + 1$, and can correct up to t errors if $d(C) \geq 2t + 1$. The minimum distance (d_{min}) is the minimum hamming distance between two code words. A good code has a small n, a large k, and also a large d. various coding techniques have been used for error detecting and correcting (Hamming, 1950; Fujiwara et al., 1990; Rao et al., 1989). Out of many error correction codes, the Reed–Solomon (RS) codes have been widely used in a variety of communication systems, such as space communication link and digital subscriber loops. The RS decoders can be used to protect digital data, against the errors which reduce the signal to noise ratio in the transmission process.

B. Overview of CA

CA is a mathematical idealization of physical systems, in which the space and time are discrete, and each cell can assume the value 0 or 1. The cells evolve in discrete time steps, according to some deterministic rules that depend only on logical neighbourhood. In the simplest case, a three-neighbourhood CA with finite cells, linearly arranged in one dimension (1-D CA), can be imagined. The next state of each cell is determined by a transition function. Such this function for a cell of a three- neighbourhood CA can be expressed as follows:

$$q_i^{t+1} = f_i(q_{i-1}^t, q_i^t, q_{i+1}^t)$$

Where q_i^t denotes the state of the i^{th} cell, at time t, and f_i is the next state function, so called the rule of the cell (Wolfram,1983). For a three- neighbourhood dependence, the next state of a cell is assumed to be dependent on itself and its two neighbouring cells. So, for such this dependence, the rule is a function of three variables. Therefore there can be $2^4 = 256$ possible transition functions.

There are several definitions used extensively to characterize a CA. If all the CA cells obey the same rule, then the CA is said to be a uniform CA; otherwise it is a hybrid one. The rules with AND-OR logic are non-additive rules On the other hand, a rule involving only XOR logic is called a linear rule and the rules involving XNOR logic are referred to as complemented rules. A CA with all the cells having linear rules is called a linear CA and a CA having a combination of XOR and XNOR rules is called an Additive CA (ACA). For example, among all 256 possible rules for a

three-neighbourhood CA, 14 rules which can be realized by xor/xnor logic are known as additive rules (Chaudhuri, 1997). The ACA has been of special interest by researchers, since it can be characterized by matrix algebraic tools. These tools are used to represent an ACA with different rules in its different cells. A 1-dimensional ACA having n cells is characterized by a linear operator, modelled by a matrix [T]n×n, in which T is the characteristic matrix of CA. The most common liner rules used in a byte ECC system are given as follows:

Rule 60: $q_i(t + 1) = q_{i-1}(t)$ xor $q_i(t)$;
Rule 90: $q_i(t + 1) = q_{i-1}(t)$ xor $q_{i+1}(t)$;
Rule 102:$q_i(t + 1) = q_i(t)$ xor $q_{i+1}(t)$;
Rule 150:$q_i(t + 1) = q_{i-1}(t)$ xor $q_i(t)$ xor $q_{i+1}(t)$;

Depend on the properties of state transition function, the additive CA can be broadly classified into two classes - group CA and non-group CA. In the former, every state has a unique predecessor. Whereas in the later, all states belong to some disjoint set of cycles. Really, the non-group CA is characterized by the presence of some non-reachable states in the state transition graph. Furthermore, non-group CA is formed by the tree-structured transition behaviour of acyclic state in the state transition graph. In general, a CA uses the following three boundary conditions: If the two end cells of the CA are connected to a constant logic value, either 0 or 1, then it is said to be a Null Boundary CA (NBCA); if the extreme cells are assumed to be adjacent with each other, then it is called a Periodic Boundary CA (PBCA).

CA is basically array of cells where every cell consists of a D-flip-flop and a combinational logic to implement the next-state function. There are number of applications of CA in the field of digital VLSI design and testing, error control coding, cryptography, image processing, DNA computing, etc. CA has already shown its novelty to design the different error correcting codes. CA-based double-byte error correcting codes are extended from binary field to prime fields.

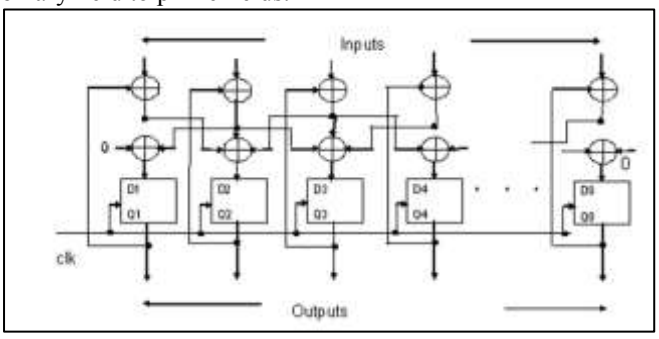

Fig. 1: Internal architecture of CA-T having eight cells

A maximum length CA has a characteristic polynomial which is primitive and is used to design CA based codes. In this paper, an 8-cell maximum length CA with rule vector < 90,150,150,90,90,90,90,90 > is employed to design codec. CA-based rule-90 corresponds to connection with left and right neighbours. On the other hand rule-150, corresponds to connection with left, self and right neighbours. For implementation of maximum length CA, rule-90 and rule 150 are used. The corresponding characteristic matrix (T) is as follows.

$$T = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Where '0' means there is no connection and '1' means there is connection. For the left most cell there is no left neighbour. Similarly for right most cell there is no right neighbour, so it is considered as null boundary. Characteristic polynomial associated with (T) matrix is $p(x) = x^8 + x^4 + x^3 + x^2 + 1$, which is a primitive polynomial in GF(28). Note that CA-Ti is an 8-cell cellular automata having the characteristic matrix T^i. Fig.1 shows the internal architecture of CA-T which contains 16 two input XOR (XOR2) gates and 8 numbers of D flip-flops. Except 2nd and 3rd cell, all other cells are constructed using CA rule-90. First level XOR gates are used to XOR the input with present state of CA and second level XOR gates are needed to implement the rule vector. For a general CA based t-byte error correcting codes the parity matrix is presented in the following form.

$$P = \begin{bmatrix} I & I & \cdots & I \\ T & T^2 & \cdots & T^k \\ T^2 & (T^2)^2 & \cdots & (T^2)^k \\ \cdots & \cdots & \cdots & \cdots \\ T^{2t-1} & (T^{2t-1})^2 & \cdots & (T^{2t-1})^k \end{bmatrix}$$

In a CA-based t-byte error correcting code, the b^{th} check byte P_b is generated from the information symbols D $(D_1,D_2,\cdots,D_k)$ by running the CA for k cycles. where $D_i \in GF(2^8)$ and i=1,2,⋯,k. The expression for the b^{th} check byte is

$$P_b = T_b[D_k] \oplus T^{b(2)}[D_{k-1}] \oplus \cdots \oplus T^{b(k)}[D_1]$$

Where $0 \le b \le (2t-1)$ and T is the characteristic matrix of an 8-cell maximum length CA.

C. Encoding of CASEC

Encoding of two different CASEC based on modified parity matrices are described here.

1) Encoding of CASEC (10,8,8)

The parity matrix (P) for CASEC(10,8,8) codes in $GF(2^8)$ is as follows.

$$P = \begin{bmatrix} I & T^{-2} & I & T^{-4} & I & T^{-6} & I & T^{-8} \\ T & I & T^3 & I & T^5 & I & T^7 & I \end{bmatrix}$$

Employing parity matrix P, two parity check symbols (P_0 and P_1) for CASEC are computed using the following equations.

$P_0 = D_8+T^{-2}D_7+D_6+T^{-4}D_5+D_4+T^{-6}D_3+D_2+T^{-8}D_1.$
$P_1 = TD_8 +D_7 +T^3D_6 +D_5 +T^5D_4 +D_3 +T^7D_2 +D_1.$

From the properties of 8-cell maximum length CA, it can be shown that $T^{-i} = T^{255-i}$, where $1 \le i \le 255$. Therefore, $T^{-2} = T^{253}$. Finally the parity matrix (P) can be rewritten in the following form.

$$P = \begin{bmatrix} I & T^{253} & I & (T^{253})^2 & I & (T^{253})^3 & I & (T^{253})^4 \\ T & I & T^3 & I & T^5 & I & T^7 & I \end{bmatrix}$$

This form of P matrix is used here for hardware implementation of CASEC (10,8,8) encoder.

2) Encoding of CASEC (19,16,4)

A 4-cell maximum length CA is necessary to design an encoder and decoder with symbol size 4 bits. One of the rule vector for a 4-cell maximum length CA is < 150,90,150,90 >. The characteristic matrix (T4) corresponding rule vector is as follows.

$$T_4 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

The characteristic polynomial associated with the matrix (T^4) is $p(x) = x^4 + x + 1$. The parity matrix for a modified CASEC(19,16,4) codes can be written in the following form.

$$P_m = \begin{bmatrix} T_4 & I & I & T_4^2 & I & I & \cdots & T_4^6 & I & I & T_4^6 \\ I & T_4 & I & I & T_4^2 & I & \cdots & I & T_4^6 & I & I \\ I & I & T_4 & I & I & T_4^2 & \cdots & I & I & T_4^6 & I \end{bmatrix}$$

CASEC encoder generates the codeword by adding the parity check symbols with message symbols. Decoding procedure of CASEC codec is presented in the following subsection.

D. Decoding of CASEC

The first block of decoding circuit is syndrome block. Syndromes are calculated by XORing recomputed check byte (P_b') obtained from received codewords and respective received check byte (P_b).The b^{th} syndrome byte S_b is defined as

$$S_b = P_b \oplus P_b'$$

Depending upon syndromes values, there are three possibilities

1) Two syndromes are zero then there is no error in received codeword.
2) Any one of these two syndromes is non-zero then error occurs in the corresponding parity check symbol position.
3) Two non-zero syndromes correspond to one or more errors in received codeword. Error position and magnitude are computed employing following relationships.

1) Error in Even Position

If there is an error in even byte position, i.e. for even i, then syndromes are given below where j = (9 − i). The relationship between S0 and S1 is derived from the equation

$$S_0 = e_i;\ S_1 = T^{(9-i)}e_i = T^j e_i\ ;\ S_1 = T^j S_0$$

First, the value of j is determined for which equation $S_1 = T^j S_0$ is satisfied. Then, (9−j) is the error position and S_0 is the error magnitude.

2) Error in odd position

If there is an error in odd byte position, i.e. for odd i, then syndromes are given below. The relationship between S_0 and S_1 is obtained from this equation.

$$S_1 = e_i;\ S_0 = T^{-(9-i)}e_i;\ T^{(9-i)}S_0 = e_i;\ S_1 = T^j S_0$$

The value of j for which above equation is satisfied that is used to determine error position i = (9−j) and error magnitude is S_1. The two equations $S_1 = T^j S_0$ and $S_1 = T^j S_0$ are same and used to calculate error position. For the sake of understanding following examples are given. If there is an error in D8 position.

$$S_0 = e_8; S_1 = T^1 e_8; S_1 = T^1 S_0$$

So the value of j is one and error position is eighth and magnitude is S_0..If error in D_7 position then,

$$S_0 = T^{-2}e_7; S_1 = e_7; S_1 = T^2 S_0$$

From above equation, it is found that value of j is 2, hence error position is 7 and magnitude is S_1. It is noted that proposed decoding technique is very simple and easy to implement.

IV. Design of Proposed CASEC Codec

A new architecture of proposed CASEC(10,8,8) encoder and decoder are presented in this section. In practice, most of the computing systems need 64 or 128 bits data widths to build the different modules using 8 bits memory devices. For 64 bits, the codes are SEC-RS(10,8,8), 2xSEC-RS(10,8,4) and for 128 bits codes are SEC-RS(18,16,8), 2xSEC-RS(19,16,4).

A. CASEC (10,8,8) Encoder

The architecture of proposed CASEC(10,8,8) encoder is depicted in the Fig 2 The input data divided into odd and even symbols by employing 'Even-odd spitter circuit' block. Odd and even part of P_0 and P_1 are written from equation

$P_0 = D_8 + T^{-2}D_7 + D_6 + T^{-4}D_5 + D_4 + T^{-6}D_3 + D_2 + T^{-8}D_1.$

$P_1 = TD_8 + D_7 + T^3D_6 + D_5 + T^5D_4 + D_3 + T^7D_2 + D_1.$

$P_{0o} = T^{253}D_7 + (T^{253})^2D_5 + (T^{253})^3D_3 + (T^{253})^4D_1.$

$P_{0e} = D_8 + D_6 + D_4 + D_2.$

$P_{1o} = D_7 + D_5 + D_3 + D_1.$

$P_{1e} = TD_8 + T^3D_6 + T^5D_4 + T^7D_2.$

The parity symbol (P0) is computed by XORing P_{0o} (odd part of P_0, calculated using CA-T^{253} and odd data inputs) with P_{0e} (even part of P_0, obtained by XORing the even data inputs). On the other hand, the parity symbol (P_1) is computed by XORing P_{1e} (even part of P_1 which is calculated employing CA-T^2, CA-T and even data inputs) with P_{1o} (odd part of P_1 which is formed by XORing the odd data inputs). The architecture of 2xCASEC(10,8,4) encoder is displayed in Fig.2

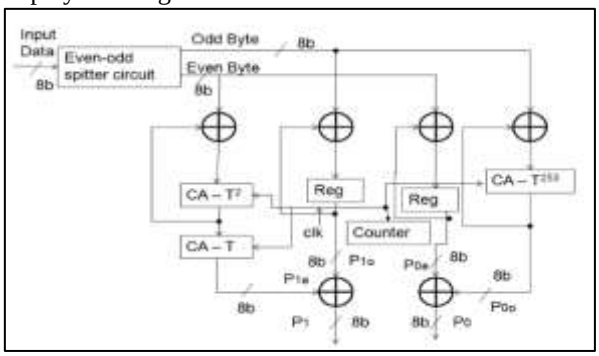

Fig. 2: Architecture of CASEC(10,8,8) encoder

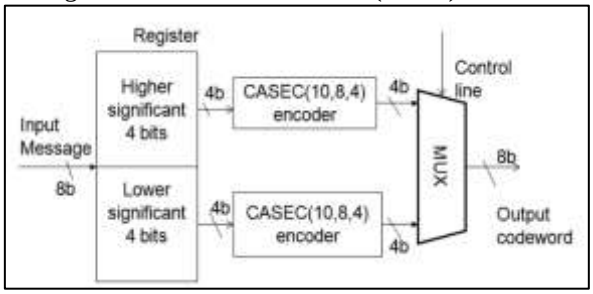

Fig. 3: Architecture of 2xCASEC (10,8,4) encoder

This is constructed by interleaving two CASEC(10,8,4) encoders. First, the 8 bits input data are stored in the 8 bits register and it is divided into two 4 bits data. Higher significant 4 bits form a group and lower significant 4 bits form another group. Four bits parity symbols are generated individually from upper and lower 4 bits data. Then, these parity symbols are multiplexed and codeword is formed. Similarly, 2xCASEC(19,16,4) encoder circuit is designed.

B. CASEC(10,8,8) Decoder

Operation of CASEC(10,8,8) decoder is explained here. For decoding two important blocks are needed a) syndrome generator which is similar to the CASEC(10,8,8) encoder circuit with an extra XOR operation and b) Error Position and Magnitude Computation (EPMC) block. The architecture of proposed EPMC circuit is shown in the Fig. 4. In EPMC block, CA-T runs for j cycles until it satisfies the relationship $S_1 = T^jS_0$. After obtaining the value of j from counter, error location is computed using combinational circuit which will compute (9 − j) and is shown in Fig.5

In Fig. 4, a gated clock is used to control the counter i.e. it will stop when the output of OR gate is zero (p=0). Least Significant Bit (LSB) of counter output acts as the select line of the multiplexer (MUX). Depending on this select input, MUX will generate the error magnitude. Proposed EPMC needs lesser area than existing SEC decoders. The number of clock cycles required for SEC-RS(10,8,8) and SEC-RSmod1(10,8,8) decoder is 10 [22]. But only 8 clock cycles are needed in CASEC(10,8,8) EPMC block.

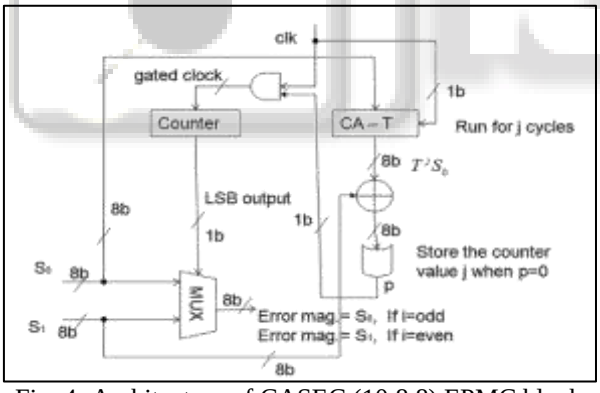

Fig. 4: Architecture of CASEC (10,8,8) EPMC block

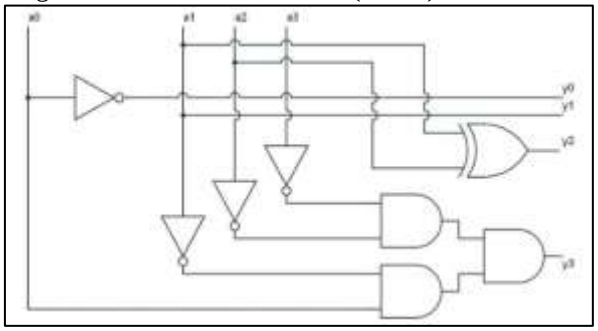

Fig. 5: Combinational circuit to compute (9−j)

C. Parallel Processing

Parallel computing may be a sort of computation during which many calculations or the execution of processes are administered simultaneously. Large problems can often be divided into smaller ones, which may then be solved at an equivalent time. There are several different sorts of parallel computing: bit-level, instruction-level, data, and task parallelism. Parallel computing may be a sort of computation during which many calculations or the execution of processes are administered simultaneously. Large problems can often be divided into smaller ones, which may then be solved at an equivalent time. There are several different sorts of parallel computing: bit-level, instruction-level, data, and task parallelism.

It's possible to possess parallelism without concurrency (such as bit-level parallelism), and concurrency without parallelism (such as multitasking by time-sharing on a single-core CPU).In parallel computing, a computational task is usually weakened into several, often many, very similar sub-tasks which will be processed independently and whose results are combined afterwards, upon completion.

In concurrent computing, the numerous processes often don't address related tasks; once they are doing, as is typical in distributed computing, the separate tasks may have a varied nature and sometimes require some inter-process communication during execution. Communication and synchronization between the various subtasks are typically a number of the best obstacles to getting optimal parallel program performance. A theoretical boundary on the speed-up of one program as a results of parallelization is given by Amdahl's law.

V. SOFTWARE IMPLEMENTATION AND DISCUSSION

In this section, FPGA implementation results of different CASEC codecs are discussed. Performance of CASEC(10,8,8) decoder is also discussed here. All the circuits are expressed in Verilog HD

A. Proposed CASEC(10,8,8) decoder

Parallel processing technique used for getting more efficient decoder. In this block diagram S0 and S1 are the inputs of the decoder circuitry. S0 and S1 inputs are getting from the Syndrome block. After satisfied the condition $S_0 = T^jS_1$ get the integer count value as the match block output. Integer to Binary Converter block used to convert the resulted count value as binary form. Error position getting by using the Formula 9-Count. When the LSB of the Count value as 0 then S0 is the Error Magnitude. In other case the S1 will be the Error Magnitude. Fig.6 shows the Block diagram of the modified Decoder

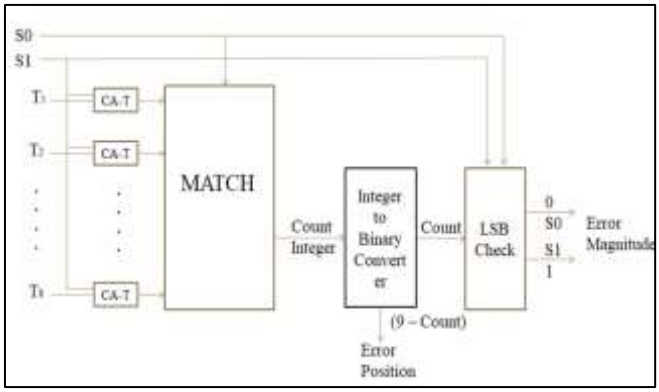

Fig. 6: Modified Decoder Block Diagram

B. Simulation Results

1) Encoder:

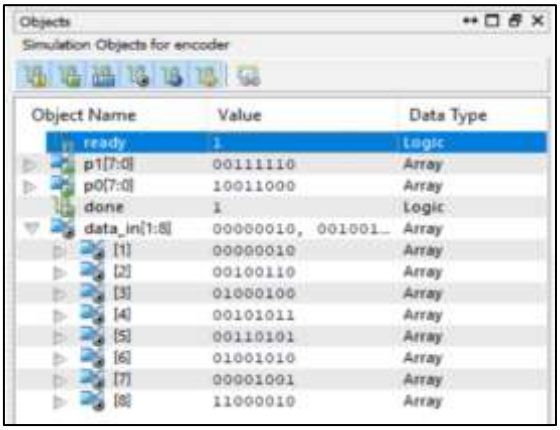

Fig. 7: Encoding of CASEC (10,8,8)

This the output of the Encoder CASEC (10,8,8). Here ready is the signal used for start the process of the Encoder. For start the process set the ready signal as 1. In the encoding side entered 8 data's with 8 bit's as the inputs. Based on the 8 data inputs get the two parity check bytes as the output. So the Data bytes send along with Parity check bytes in the encoding process.

2) Decoder

Based on the parity check bytes and the data bytes received at the decoder find out whether any errors occurred in the decoding process. Here 5 error possibilities occurred when transmitting one data to receiver side.

First case will be errors occurred in the odd position. Here 7th data byte changed from 01101101 to 00001001. Bit changes make it as 1 in the magnitude. When the error in odd position the syndrome byte S1 will be the magnitude. So the magnitude get as 01100100. Error position get as 00000111.Flags are used to indicates the errors. Here error flag is set as 1 and other flag set as 0. After completing the decoding process, done signal make it as 1.

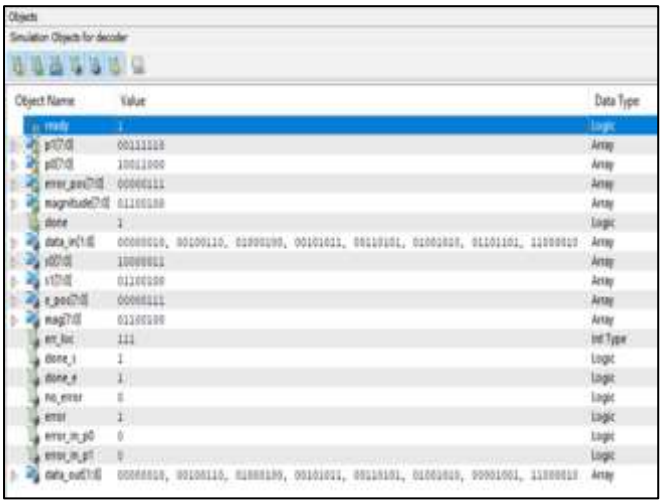

Fig. 8: Decoding of CASEC (10,8,8) – Error in Odd Position

Similarly in the second case will be errors occurred in the even position. Here 8th data byte changed from 00100100 to 11000010. When the error in even position the syndrome byte S0 will be the magnitude. So the magnitude get as 11100110. Error position get as 00001000.

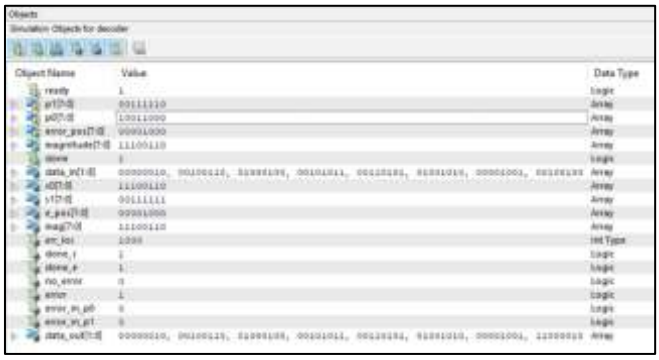

Fig. 9: Decoding of CASEC (10,8,8) – Error in Even Position

Sometimes there is no errors occurred at the decoding side. So the 3rd possibility will be no errors happened in the decoding process. For indicating this condition, no_error flag set as 1 and other flags are 0. Here get the transmitted input data as the output without any changes happened in the bytes. Figure 6.5 shows the output of decoding of CASEC (10,8,8) without any error.

Sometimes errors occurred in the parity check bytes P0 or P1. That time the corresponding flags (error_in P0, error_in P1) set as 1. When syndrome S0 is non-zero then error occurs in the corresponding parity check symbol position P0. Other case when syndrome S1 is non-zero then error occurs in the corresponding parity check symbol position P1.

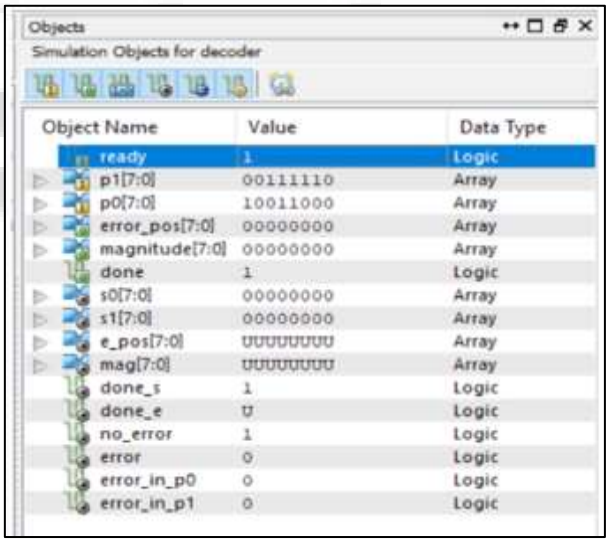

Fig. 10: Decoding of CASEC (10,8,8) – No Error

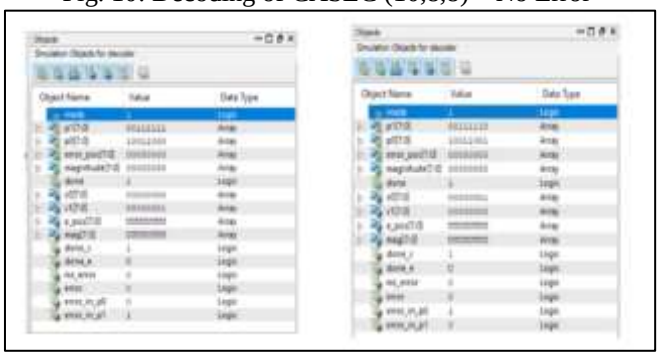

Fig. 11: Decoding of CASEC (10,8,8) – Error in Parity check byte P1 and P0

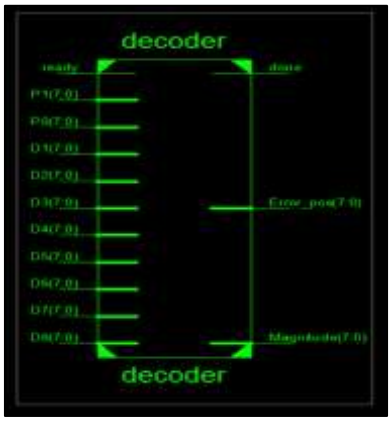

Fig. 12: RTL Schematic of Decoder

In this system contains adder register and multiplication block. In this schematic contains two parity bytes and 8 data bytes as input also the error position and magnitude as the output of the decoding process.

C. Performance of CASEC(10,8,8) decoder

1) Timing and Hardware Utilization

In this system computation time of the decoding process reduced from 2.050ns to 1.663ns because of the Parallel processing. But after using the parallel processing technique have small increase in the hardware utilization. Below tables shows the Design summary report and Timing summary reports of the CASE(10,8,8) Decoder before and after using the parallel processing technique.

```
Timing Summary:
---------------
Speed Grade: -2

   Minimum period: 2.050ns (Maximum Frequency: 487.864MHz)
   Minimum input arrival time before clock: 7.884ns
   Maximum output required time after clock: 0.668ns
   Maximum combinational path delay: No path found
```

Table 1: Timing summary report of CASEC (10,8,8) Decoder

```
---------------
Speed Grade: -2

   Minimum period: 1.663ns (Maximum Frequency: 601.504MHz)
   Minimum input arrival time before clock: 2.284ns
   Maximum output required time after clock: 0.682ns
   Maximum combinational path delay: No path found
```

Table 2: Timing summary report of CASEC (10,8,8) Decoder using Parallel Processing

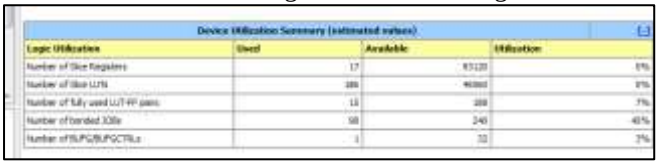

Table 3: Design summary report of CASEC (10,8,8) Decoder

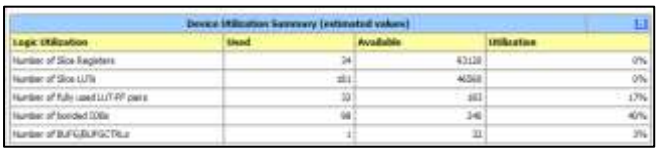

Table 4: Design summary report of CASEC (10,8,8) Decoder using Parallel Processing

SYSTEM	TIMING
EXISTING	2.050ns
PROPOSED	1.663ns

Table 5: Analysis Table

All modules are simulated and synthesized using Xilinx 14.3 simulation tool and vertex-6 FPGA device. Area is measured in terms 'sliceregister (slicereg.)'and 'Look-Up-Table (LUT)'.Here delay indicates the encoding and decoding time delay of the encoder and decoder respectively. Proposed CASEC encoder, syndrome and EPMC blocks required more number of logic gates than conventional (Convent.) but it less computation time.

VI. CONCLUSION

In the recent years, because of parallel and regular structure, the cellular automata have become an important tool for error correcting codes. These techniques are suitable from VLSI design viewpoint and attractive via their simplicity, regularity and higher throughput. The decoding scheme is very simple and suitable for high speed memory systems. Also requires significantly less hardware, compared to the existing techniques. CASEC codecs are designed and implemented both FPGA platform. Our design is also efficient than existing designs. Speed of proposed design is also comparable to the existing related designs. Proposed CASEC encoder, syndrome and EPMC blocks required more number of logic gates than conventional (Convent.) but it less computation time. Our CASEC decoder can determine the error location and error magnitude correctly, if the number of error is one. This compact and efficient codec is suitable for digital memory systems.

REFERENCES

[1] Samanta, J., Bhaumik, J., & Barman, S. (2018). "Compact CA-Based Single Byte Error Correcting Codec,". IEEE Transactions on Computers, 67(2), 291–298.

[2] Chowdhury, D. R., Gupta, I. S., & Chaudhuri, P. P. (1995). CA-based byte error-correcting code. Computers, IEEE Transactions on, 44(3), 371-382.

[3] Bhaumik, J., Chowdhury, D. R., & Chakrabarti, I. (2008). An improved double byte error correcting code using cellular automata. In Cellular Automata (pp. 463-470). Springer Berlin Heidelberg.

[4] Jaydeb Bhaumik, Dipanwita Roy Chowdhury, and Indrajit Chakrabarti (2008). "Null Boundary 90/150 Cellular Automata for Multi-byte Error Correcting Code". Indian Institute of Technology, India.

[5] Parul, G., Deepak, G., Aruna T. (2012). N-byte error detecting and correcting code using Reed-Solomon and cellular automata approach.(Vol.1),511-515.

[6] Nithiya, R., Seridevi, S(2012). "Byte error correcting code using cellular automata." International Journal of Communications and Engineering. (Vol.5)

[7] M. Y. Hsiao, "A class of optimal minimum odd-weight column SEC-DED codes," IBM J. Res. Develop., vol. 14, pp. 395-301, Jul. 1970.

[8] S. Pontarelli, P. Reviriego, M. Ottavi and J. A. Maestro, "Low Delay Single Symbol Error Correction Codes Based on Reed Solomon Codes," IEEE trans. on comput., vol. 64, no.5, May. 2015.

[9] S. Liu, P. Reviriego, and J. A. Maestro, "Efficient majority logic fault detection with difference-set codes for memory applications,"IEEE Trans. Very Large

Scale Integr. (VLSI) Syst., vol. 20, no. 1, pp. 148-156, Jan. 2012.

[10] J. Samanta, J. Bhaumik and S. Barman,"CA-based Area Optimized Three Bytes Error Detecting Codes," J. of Cellular Automata, vol.10, no.5-6, pp:409-423, Nov. 2015.

[11] Bhaumik, J., Janakiram, B., & Chowdhury, D. R. (2008). "Architectural Design of CA-Based Double Byte Error Correcting Codec". 2008 IEEE Region 10 and the Third International Conference on Industrial and Information Systems.

[12] P. Reviriego, S. Liu, L. Xiao, and J. A. Maestro,"An Efficient Single and Double-Adjacent Error Correcting Parallel Decoder for the (24,12) Extended Golay Code," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., pp.1-4, 2015.

[13] A.S. Macian, P. Reviriego and J. A. Maestro, "Enhanced Detection of Double and Triple Adjacent Errors in Hamming Codes Through Selective Bit Placement," IEEE Trans. Dev. Mat. Rel., vol. 12, no. 2, pp.357-362, Jun. 2012

[14] R. C. Baumann,"Radiation-induced soft errors in advanced semiconductor technologies," IEEE Trans. Dev. Mat. Rel., vol. 5, no. 3, pp. 301-316, Sep. 2005.

[15] S. Lee, S. H. Jeon, S. Baeg, and D. Lee, "Memory reliability analysis for multiple block effect of soft errors," IEEE Trans. Nucl. Sci., vol.60, no.2, pp.13841389, Apr. 2013.

[16] E. Fujiwara,"Code design for dependable systems: theory and practical applications," John Wiley and Sons, 2006

[17] X. She, N. Li and D. W. Jensen, "SEU tolerant memory using error correction code," IEEE Trans. on Nuclear Sci., vol.59, no.1, pp.205-210, Feb. 2012.

[18] S. M. Abbas, S. Baeg, and S. Park, "Multiple cell upsets tolerant content-addressable memory," Proc. in 49th IEEE Int. Rel. Physics Symp., Monterey, CA, US, pp.863867, Apr. 2011.

[19] K. Zhang, J. Furuta, R. Yamamoto, K. Kobayashi, and H. Onodera, "A radiation-hard redundant flip-flop to suppress multiple cell upset by utilizing the parasitic bipolar effect," IEICE Trans. Electron., vol.E96-C, no.4, pp.511517, Apr. 2013.

[20] J. Hoyoon, and L. Yongsurk. "Protection of On-chip Memory Systems against Multiple Cell Upsets Using Double-adjacent Error Correction Codes," IEICE Trans. on Electronics vol. 98, no.3, pp.267-274, 2015.

[21] P. Reviriego, S. Pontarelli, J. A. Maestro, and M. Ottavi, "A method to construct low delay single error correction codes for protecting data bits only," IEEE Trans. Comput. Aided Des. Integr. Circuits Syst., vol. 32, no. 3, pp. 479-483, Mar. 2013.

[22] J. A. Maestro, P. Reviriego, C. Argyrides and D. K. Pradhan, "Fault tolerant single error correcting encoders," J. of Electron Test, vol. 27, pp. 215-218, 2011.

[23] R. Datta and N. A. Touba,"Exploiting Unused Spare Columns to Improve Memory ECC," in Proc. 27th IEEE VLSI Test Symp., 2009, pp.47-52

[24] P. Reviriego, S.Pontarelli, A. Evans and J. A. Maestro,"A Class of SECDED-DAEC Codec Derived From Orthogonal Latin Square Codes," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 23, no. 5, pp.968-972, May. 2015.

[25] .P. Reviriego, C. Argyrides, J. A. Maestro,"Efficient error detection in Double Error Correction BCH codes for memory applications", Microelectronics Rel., 52 (2012) 1528-1530.

[26] S. Pontarelli, P. Reviriego, M. Ottavi and J. A. Maestro, "Low Delay Single Symbol Error Correction Codes Based on Reed Solomon Codes," IEEE trans. on comput., vol. 64, no.5, May. 2015.

[27] P. Ankolekar,S. Rosner,R. Isaac and J. Bredow, "Multi-bit error correction methods for latency-constrained flash memory systems," IEEE Trans. Dev. Mat. Rel., vol.10, no.1,pp.33-39, 2010.

[28] S. Ghosh and P. D. Lincoln,"Dynamic low-density parity check codes for fault-tolerant nano-scale memory," in Proc. of Found. of Nanosci., Snowbird, Utah, USA, 2007.

[29] G. C. Cardarilli, M. Ottavi, S. Pontarelli, M. Re, and A. Salsano,"Data integrity evaluations of Reed solomon codes for storage systems" ,inProc. IEEE 19th Int. Defect Fault Tolerance VLSI Syst., 2004, pp. 158-164.

[30] G. C. Cardarilli, M. Ottavi, S. Pontarelli, M. Re, and A. Salsano, "Fault tolerant solid state mass memory for space applications," IEEE Tran. on Aerospace and Elec. Sys., vol.41, no. 4, pp. 1353-1372, 2005.

[31] G. C. Cardarilli, S. Pontarelli, M. Re, and A. Salsano,"Concurrent Error Detection in Reed Solomon Encoders and Decoders," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.15, no.7, pp.842-846,Jul. 2007.

[32] P. P. Chaudhuri, D. Roy Chowdhury, S. Nandi and S. Chattopadhyay, "Additive Cellular Automata: Theory and Applications,"IEEE Comput. Soc. Press,1997.

[33] J. Bhaumik and D. Roy Chowdhury,"New Architectural Design of CABased Codec," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 18, no. 7, pp:1139-1144, Jul. 2010.

[34] S. Lin and D.J. Costello, "Error Control Coding," 2nd ed. Englewood Cliffs, NJ,USA: Prentice-Hall,2004.

[35] M. E. Koroglu, I. Siap and H. Akin, "Cellular automata based byte error correcting codes over finite fields," 1st Int. Conf. on Analysis and Applied Mathematics (ICAAM 12), vol. 1470, no. 1, AIP Publishing, 2012

