N-BYTE ERROR DETECTING AND CORRECTING CODE USING REED-SOLOMON AND CELLULAR AUTOMATA APPROACH

Parul Gaur[1], Deepak Gaur[2], Aruna Tomar[3]

Department of Electrical and Electronics Engineering, UIET, PanjabUniversity, Chandigarh, India., ***parul34@gmail.com***
Department of Computer Science and Engineering, ASET, Amity University, Noida(U.P.), India, ***dgaur@amity.edu***
Department of ECE, Marathwada Institute of Technology, Delhi, India, ***aruna_tomar07@yahoo.com***

Abstract

Single and double byte errors are most common errors in memory systems. With the advancement of technologies, more information bytes can be send over a transmission channel, but this increases the probability of more errors. Here we propose a methodology for detecting and correcting N-byte errors in the information bytes based on cellular automata (CA) concept. Cellular automata already accepted as an attractive structure for error detecting and correcting codes. In this paper, highly efficient, reliable and less complex cellular automata based N-byte error correcting encoder and decoder has been proposed. The design is capable of adding 2N check bytes corresponding to N information bytes at the encoder, which are used at the decoder section to detect and correct the byte errors.

Keywords— *Cellular Automata (CA), Reed-Solomon (RS) code, Information Bytes (IB), Check Bytes (CB)*

----------***----------

1. INTRODUCTION

During transmission of information bytes, it may be possible that information signal is corrupted by noise, which leads to change in information signal. It is very important to detect the errors in the information bytes and correct them. Various coding techniques have been used for error detecting and correcting. Error correcting codes have been used to enhance system reliability and data integrity. The basic concept is to add check bytes or redundancy to the information bytes at the encoder so that decoder can recover the information bytes from the received block possibly corrupted by the channel noise. Reed -Solomon (RS) codes are most commonly used codes for error correction because these codes can detect both random as well as burst errors. Reed-Solomon codes were invented in 1960 by Irving S. Reed and Gustave Solomon, at the MIT Lincoln Laboratory. Reed-Solomon codes are used in many applications like wireless communication, satellite communications, high speed modems and storage devices like CD, DVD. For example RS (28, 24) and RS (32, 28) is popularily used for multistorage CD. In this four check bytes are added to the information bytes, which detects the two byte errors. Complexity of RS encoder and decoder increases with the error correcting capability of the codes. A general decoding scheme for RS decoder based upon pipelined degree computationless modified Euclidean algorithm is found in [1]. In [1], a high speed pipelined architecture was proposed with the aim of reducing hardware complexity because of absence of degree computation circuit and improving the clock frequency in RS decoders. However conventional RS decoders [2]-[4] induce relatively huge hardware complexity. A system designer always prefer to have simple and regular structure for reliable high speed operations of the circuit. It has been found that these parameters are supported by local neighbourhood cellular automata (CA). CA based byte error correcting code has been proposed in [5]. Till now single, double and atmost three bytes errors are corrected [5]. In our research paper, a methodology for detecting and correcting N-byte errors based upon cellular automata has been proposed. Whatever will be the value of N, system will detect and correct the errors in the information bytes. A logic has been generated, which detects and corrects the errors.

2. PRELIMINARIES

A. CA Preliminaries

CA is an array of cells, where each cell is in any one of the permissible states. At each discrete time step, the evolution of the cell depends on the combination logic which is function of the present state of the cell itself and states of the neighbouring cells. For two state three neighbourhood, the evolution of the ith cell can be represented as a function of the present states of $(i-1)th$, ith and $(i+1)th$ cells as: $x_i(t+1) = f\{x_{i-1}(t), x_i(t), x_{i+1}(t)\}$ where f represents the combinational logic. For a two state three cellular automata there are 2^3 that is, 8 distinct neighbourhood configurations and 2^8 that is, 256 distinct next state functions. The CA characterized by a rule known as rule 90, specifies an evolution from neighbourhood configuration to next state as:

111	110	101	100	011	010	001	000
0	1	0	1	1	0	1	0

he corresponding combinational logic for rule 90 is:

$$x_i(t+1) = x_{i+1}(t)\bar{x}_{i-1}(t) + \bar{x}_{i+1}(t)x_{i-1}(t)$$
$$x_i(t+1) = x_{i+1}(t) \oplus x_{i-1}(t)$$

that is, the next state of ith cell depends on the present states of its left and right neighbours (Figure 1). Similarly, the combinational logic for rule 150 is given by:

$$x_i(t+1) = x_{i+1}(t) \oplus x_i(t) \oplus x_{i-1}(t)$$

that is, the next state of ith cell depends on the present states of its left and right neighbours as well as its own present state (Figure 1).

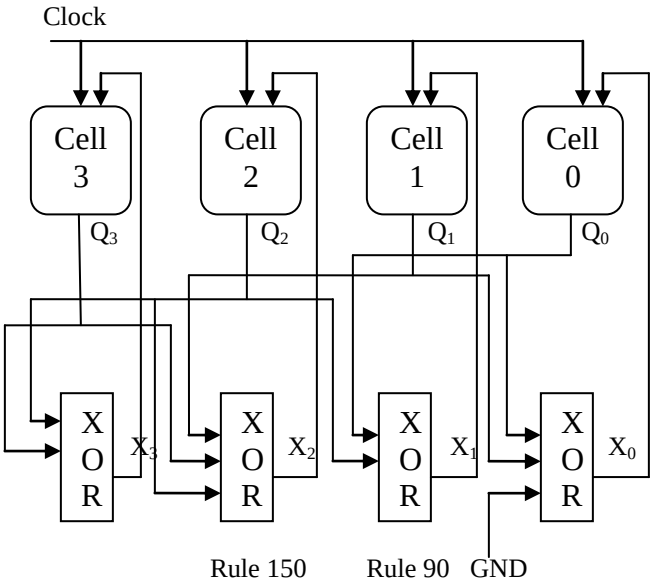


Figure 1 A Hybrid Cellular Automata

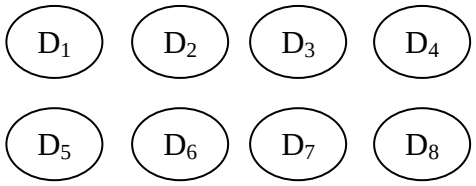
A CA characterized by EXOR or EXNOR dependence is called an additive CA. If in a CA, the neighbourhood dependence is EXOR, then it is called non-complemented CA and the corresponding rule is non-complemented rule. For EXNOR dependence, CA is called complemented CA and the corresponding rule is called complemented rule. There exists 16 additive rules which are:

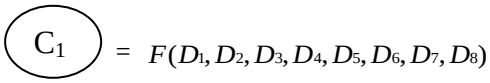
Rule 0, 15, 51, 60, 85, 90, 102, 105, 150, 153, 165, 170, 195, 204, 240 & 255

A uniform CA is one in which same rule applies to all cells while in hybrid CA (Figure 1) different rules are applied to different cells. In our methodology, rule 90 and rule 150 are used for cells in CA. Based upon this, an efficient error detecting and correcting code is designed.

B. Reed-Solomon Code

In RS code, we assume that n storage devices hold data $D_1, D_2, D_3, ..., D_{n-1}, Dn$ and m storage devices hold checksums $C_1, C_2, C_3, ..., C_{m-1}, C_m$. These checksums are computed from the data. In this code, if any m devices fail then they can be reconstructed from the surviving devices. The value of checksum device is computed using a function F:





If any data Dj changes that is becomes $D'j$ then from check bytes we can recover the data.

3. MECHANISM FOR CA BASED N-BYTE ERROR DETECTING AND CORRECTING CODE

A. Encoder Section

In encoder, check bytes are added. For N-byte error detecting and correcting code, encoder generates 2N check bytes and these check bytes are concatenated with the information bytes, which forms the codeword, given as $C = DG$ (Figure 2) where $D = (D_0, D_1, D_2, ..., D_{N-1})$ is N-byte information data and G is generator matrix given as:

$$G = \begin{matrix} I\ 0\ 0 - - - - - 0\ I\ T\ T^2\ T^3 \\ 0\ I\ 0 - - - - - 0\ I\ T^2\ T^4\ T^6 \\ 0\ 0\ I - - - - - 0\ I\ T^3\ T^6\ T^9 \\ - - - - - - - - - - - - - - - - - - \\ - - - - - - - - - - - - - - - - - - \\ - - - - - - - - - - - - - - - - - - \\ 0\ 0\ 0 - - - - - I\ I\ T^N\ T^{2N}\ T^{3N} \end{matrix}$$

Here T_N is $N \times N$ characteristics matrix corresponding to N information bytes and I is $N \times N$ identity matrix.

$$T_{N \times N} = \begin{matrix} 1\ 1\ 0\ 0 - - - - - - - 0\ 0 \\ 1\ 1\ 1\ 0 - - - - - - - 0\ 0 \\ 0\ 1\ 0\ 1 - - - \ - - - - 0\ 0 \\ - - - - - - - - - - - - - - - - \\ - - - - - - - - - - - - - - - - \\ - - - - - - - - - - - - - - - - \\ - - - - - - - - - - - - - - - - \\ 0\ 0\ 0\ 0 - - - - - - - - 1\ 1 \end{matrix}$$

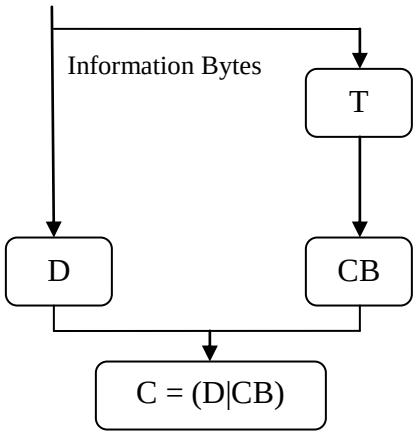


Figure 2. Codeword Generation

One such rule vector of N cell length CA is < 150,150,90,150,.....,150,90,150>. Now check bytes (CB) computed from characteristics matrix can be given as:

$$CB_\alpha = T^\alpha D_{N-1} \oplus T^{2\alpha} D_{N-2} \oplus T^{3\alpha} D_{N-3} \oplus \text{-------} \oplus T^{(N-1)\alpha} D_1 \oplus T^{N\alpha} D_0 \quad (1)$$

Where $0 \le \alpha \le (2N-1)$, corresponding to N-byte error correcting code. Based upon this, figure 3 shows the encoder section.

B. Decoder Section

We consider the problem of decoding to detect and correct the erroneous byte positions from the received codeword C. The first step is to compute the syndrome. If the received codeword is $C = D' \mid CB'$, where $D' = D \oplus D_e$ (D is original information and D_e is error in information) and $CB' = CB \oplus CB_e$,then the syndrome of codeword can be expressed as $S(C) = HC^T = H(D' \mid CB')$. Here H is the parity check matrix formed by concatenating the compressed matrix T' with the identity matrix $I_{N \times N}$.

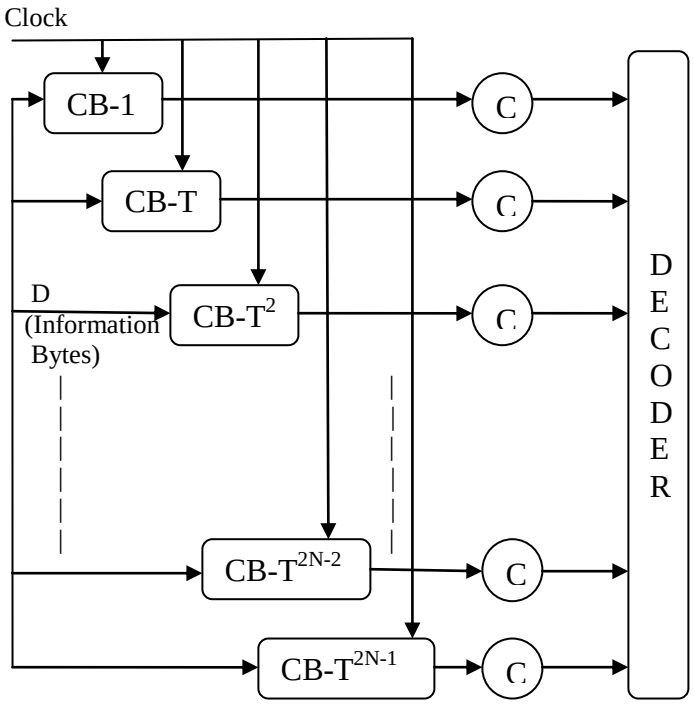


Figure 3. Encoder Section

Figure 4 shows the syndrome computation block for decoder section. The syndrome corresponding to jth check byte Sj is given by equation (2):

$$S_j(C) = CB_j \oplus CB'_j \quad (2)$$

where $0 \le j \le (2N-1)$.

If the syndrome S(C) of a received codeword C is 0, then the word is assumed to be error free. If S(C) $\neq$ 0, then errors are detected (Table I).

4. ALGORITHM FOR ERROR CORRECTION

For N-byte error correcting code, following (N+1) cases may occur:

Case I. Single information byte error
Case II. Double information byte error
Case III. Triple information byte error
Case IV. Quadruple information byte error

Case N. N information byte error
Case (N+1). Infromation byte error and check byte error

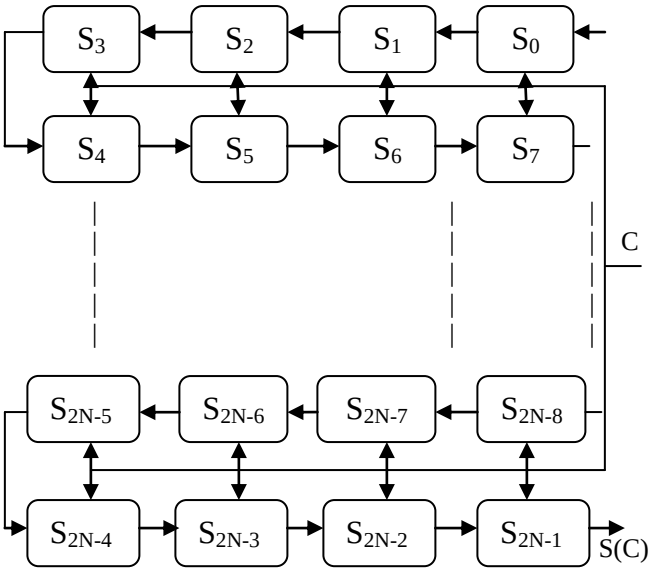


Figure4. Syndrome Computation Block

Table1. Error Detection

S_0	S_1	S_2	S_3	-	-	S_{2N-2}	S_{2N-1}	Errors In
Z	Z	Z	Z	-	-	Z	Z	Nil
NZ	Z	Z	Z	-	-	Z	Z	CB 1
Z	NZ	Z	Z	-	-	Z	Z	CB 2
|	|	|	|	|	|	|	|	|
|	|	|	|	|	|	|	|	|
Z	NZ	NZ	NZ	-	-	NZ	NZ	>3 CB
NZ	NZ	NZ	NZ	-	-	NZ	NZ	N CB

For single information byte (IB) error, suppose jth IB is in error, E_k is error magnitude, then syndrome equations from (2) are:

$$S_0 = E_k, S_1 = T^i E_k, S_2 = T^{2i} E_k, --$$
$$----, S_{2N} = T^{2Ni} E_k$$

Where $i + k = N$.
From above equations, we get:

$$T^i S_0 = S_1, T^i S_1 = S_2, T^i S_2 = S_3, -----$$
$$-------, T^i S_{2N-1} = S_{2N} \quad (3)$$

$$E_k = S_0 \quad (4)$$

Equation (3) gives the error location and (4) gives the error magnitude.

For double IB error, suppose jth and kth IB is in error, so corresponding syndrome equations are:

$$S_\alpha = T^{i\alpha} E_j \oplus T^{l\alpha} E_k$$

Where $0 \le \alpha \le 2N, i + j = N, l + k = N$.

From above equations, we get:

$$E_k = T^i [S_0] \oplus S_1 \quad (5)$$

$$Ej = S_0 \oplus E_l \quad (6)$$

Equations (5) and (6) gives the errors.

Similarly for triple, quadrule,.....,N IB, error equations can be found.

In general,

$$E_1 = S_0 \oplus E_l$$
$$E_2 = T^i [S_0] \oplus S_1$$
$$|$$
$$|$$
$$|$$
$$|$$
$$E_N = T^i [S_{N-2}] \oplus S_{N-1}$$

For IB and CB errors, out of 2N CB error can be in any CB, so we have found that there will be 2N cases for CB. Suppose E_i is the error in ith information byte and e_0 is in first CB, then the syndrome equations are:

$$S_0 = E_i \oplus e_0, S_1 = T^k E_i, S_2 = T^{2k} E_i, ----$$
$$-----, S_{N-1} = T^{(N-1)k} E_i$$

From above equations, we can find E_i. If E_i is error in ith IB and $e_1, e_2, e_3, ------, e_{2N-2}, e_{2N-1}$ are errors in CB, we can find syndrome equations. In general, we have:

$$F_{2N} = S_{2N-2} \oplus - - - - - - -$$
$$- - - \oplus S_4 \oplus S_2 \oplus T^{2i} S_0$$
$$F_{2N-1} = S_{2N-1} \oplus - - - - - -$$
$$- - - \oplus S_2 \oplus S_1 \oplus T^{i} S_0$$
$$|$$
$$|$$
$$|$$
$$|$$
$$F_2 = S_{2N-1} \oplus - - - - - - - - -$$
$$- - - \oplus S_5 \oplus S_3 \oplus T^{2i} S_1$$
$$F_1 = S_{2N-1} \oplus - - - - - - - - -$$
$$- - - \oplus S_5 \oplus S_3 \oplus T^{i} S_1$$

Knowing the error location and error magnitudes, errors are corrected using XOR operation. Suppose D'_i and E_i are the received ith IB and the calculated ith error byte respectively, then the correct IB can be calculated as:

$$D_i = D'_i, \oplus E_i \qquad (7)$$

Where $0 \le i \le N$.

5. RESULTS AND COMPARISON FROM PREVIOUS EXISTING TWO BYTES CODES

Table II. Error Detection and Correction

F_{2N}	F_{2N-1}	-	-	F_3	F_3	F_3	Errors Detected In
Z	Z	-	-	Z	Z	Z	IB
NZ	NZ	-	-	Z	Z	Z	IB,CB
NZ	NZ	-	-	NZ	Z	Z	IB,>1 CB
|	|	|				|	|
|	|	|				|	|
NZ	NZ	-	-	NZ	NZ	Z	IB,>1 CB
NZ	NZ	-	-	NZ	NZ	NZ	N Positions

Table II shows the final errors detection and from equation (7), errors can be corrected resulting in correct information bytes. In [5], byte error correcting code using CA has been given. But this code detects and corrects the atmost three bytes code. In our research paper, general algorithm for detecting and correcting N-byte code has been proposed. Whatever will be value of N, depending upon information bytes proposed system will detect and correct the errors. In contrast, the proposed encoder section has less hardware complexity than the other previously reported architectures [1]-[4].

CONCLUSION AND FUTURE SCOPE

This paper presents a novel design and algorithm for N-byte error detecting and correcting code using cellular automata and Reed-Solomon code approach. The circuit design requires much less hardware. So the implementation of the proposed scheme provides a simple high speed and cost effective solution. The proposed algorithm can be implemented in VHDL by using Xilinx ISE 9.1i tool for N-byte information signal.

REFERENCES

[1]. Seungbeom Lee, Hanho Lee, Jongyoon Shin and Je-Soo-Ko, "A High-Speed Pipelined Degree Computationless Modified Euclidean Algorithm Architecture for Reed-Solomon Decoders", ISCAS 2007.

[2].H.M.Shao,T.K.Truong,L.J.Deutsch, J.H.Yuen and I.S.Reed, "A VLSI design of a pipeline Reed Solomon decoder," IEEE Transctions on Computers, vol. C-34, no.5, pp.393-403, May 1985.

[3].W.Wilhelm, "A New Scalable VLSI Architecture for Reed-Solomon Decoders," IEEE Journal of Solid-State Circuits, vol.34, no. 3, March 1999.

[4]. H.Lee, "High speed VLSI Architecture for Parallel Reed-Solomon Decoder," IEEE Transctions on VLSI Systems, vol. 11, no. 2, pp. 288-294, April 2003.

[5]. R.Nithiya, S.Sridevi, "Byte Error Correcting Code Using Cellular Automata," International Journal of Communications and Engineering, vol. 5, no. 5, March 2012.

[6]. Amrita Sajja, Lakshmi Sarojini Pilla, N.V.S. Prabhavati, "FPGA Implementation of CA-Based Byte Error Detecting and Correcting Codec," IACQER.

[7]. J.Bhaumik, D.Roy Chowdhury, I.Chakrabarti, "An Improved Double Byte Error Correcting Code using Cellular Automata," ACRI 2008.