

# Null Boundary 90/150 Cellular Automata for Multi-byte Error Correcting Code

No Author Given

No Institute Given

**Abstract.** Cellular Automata(CA) is a well known tool to generate byte error correcting code. In this paper, we propose a CA-based multi-byte Error Correcting Code (ECC) which overcomes the weaknesses and limitation of existing scheme. As a case study three and four bytes ECC is discussed in detailed. A complete decoding algorithm of CA-based 3-byte error correcting code is presented in this work. Proposed 3-byte ECC scheme can correct errors when errors are distributed within information and check bytes or concentrated in any one of them. In case of CA-based 4-byte ECC, at most 4-byte errors can be corrected if all the errors are concentrated in information or check bytes.

## 1 Introduction

Error correcting codes have wide range of applications in digital data communications, memory system design [1], 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 communications, high speed modems and storage devices (CD, DVD). A number of general encoding and decoding schemes of the RS codes is available in the literature [10]. Many researchers have put their effort to minimize the complexity of RS decoder for communication applications.

Cellular automata(CA) has already established its novelty for bits and bytes error correcting codes[2][3]. A scheme for pipeline implementation of CA based  $t$ -byte error correcting and  $t$ -byte error detecting codes has been proposed in [4]. Another scheme for parallel implementation of CA based single byte error correcting-double byte error detecting, double byte error correcting-double byte error detecting code has been reported in [5]. Application of  $GF(2^p)$  CA in burst error correcting codes has been reported in [6]. The CA-based byte error correcting code in [3] is simpler to design compared to other schemes. But few mistakes have been identified in the error location and error magnitude computation algorithm of scheme [3]. Also the scheme [3] has one limitation that decoder can correct  $t$ -byte errors ( $t \geq 2$ ) provided errors are totally confined to information or check bytes only. An improved scheme has been proposed in [7], which eliminates the weaknesses and limitation of the previous scheme [3], for 2-byte error detection and correction only. VLSI implementation of CA-based improved double byte error correcting encoder and decoder [7] is presented in [8].

It is mentioned that the scheme for CA-based 2-byte ECC can be extended to 3-byte ECC, however the proposal[8] is for a restricted situation. The restriction is that the determination of error locations and the computation of error values are possible only if all the errors are concentrated in information bytes only. It is also mentioned that errors can be corrected if errors are distributed between information and check bytes. But the full decoding algorithm for 3-byte ECC has not been provided in [8].

In this paper, we propose a multi-byte ECC using CA and as a case study, three and four bytes ECC are discussed in detailed. A full decoding algorithm for a 3-byte ECC is proposed, which can detect and correct at most three errors and the scheme is independent of error position i.e. whether errors are distributed between information and check bytes or concentrated in information bytes only. Also the scheme has been extended for 4-byte ECC. Maximum length CA is essential for CA-based byte error correcting code. For an  $n$ -bit maximum length CA the characteristic polynomial is a primitive polynomial of degree  $n$ . There exist  $\frac{\phi(2^n-1)}{n}$  primitive polynomials in  $GF(2^n)$ , with the coefficients of the polynomials are in  $GF(2)$ . Different codes can be generated using different primitive polynomials. These set of codes are required in several cryptographic applications for better security. One such example is presented in [9], where an integrated code is used for both error correction and message authentication. In this paper, we have computed one such maximum length CA rule vector for each primitive polynomial in  $GF(2^8)$  based on CA-rules 90 and 150. These 8-bit CA rule vectors can be considered as a toy example for the scheme [9].

The rest of this paper is organized as follows. In the next section, proposed CA based multi-byte error correcting code is described. The paper is concluded in section 3.

## 2 CA-Based Multi-byte Error Correcting Code

In case of  $t$ -byte ECC in  $GF(2^8)$ , it is essential to send  $2t$  number of check bytes with the block of information bytes. Each check byte  $C_b$  is computed by running an 8-bit CA with characteristic matrix  $T^b$  for  $N$  cycles while sequentially feeding  $N$  information bytes using the expression.

$$C_b = T^b D_{N-1} \oplus T^{b(2)}[D_{N-2}] \oplus \dots \oplus T^{b(N)}[D_0] \quad (1)$$

where  $0 \leq b \leq (2t - 1)$  and  $T$  is the characteristic matrix of an 8 cell maximum length CA. The following algorithm explains method for computing  $C_b$

### **Comp-check-byte: check byte $C_b$ computation algorithm**

$s$  denotes the state of the 8 bits CA

**begin**

$s := 0$ ; **for**  $k = 0$  **to**  $N - 1$  **do**

**begin**

$s := s \oplus D_k$ ;

Run the CA for one cycle; (CA with characteristic matrix  $T^b$ )

**end**;

$C_b := s$ ;

**end;**

Decoder computes the  $b$ -th syndrome byte  $S_b$  using the following equation.

$$S_b = C_b \oplus C'_b \quad ; \quad 0 \leq b \leq (2t - 1). \quad (2)$$

where  $C_b$  is the  $b$ -th received check byte and  $C'_b$  is the  $b$ -th check byte recomputed from the received information bytes. Assume  $D'_m$ ,  $E_m$  are the received  $m$ -th information byte and calculated  $m$ -th error byte respectively then the corrected information byte is obtained by using the equation as follows.

$$D_m = D'_m \oplus E_m; \quad \text{where } 0 \leq m \leq (N - 1) \quad (3)$$

## 2.1 8-bit CA Rule Vectors for all Primitive Polynomials

Proposed CA-based byte error correcting code is based on null boundary maximum length CA. Therefore, rule vectors for the 8-bit maximum length null boundary CA is discussed in this section. We have considered the CA-rules 90 and 150 only. By simulation, we have found the rule vectors for each primitive polynomial in  $GF(2^8)$  and they have been listed in Table 1. In Table 1, '0' and '1' correspond to rule 90 and 150 respectively. It has been observed that mirror image of each rule vector corresponds to same primitive polynomial. For example, 00000110 and 01100000 are two rule vectors for primitive polynomial  $x^8 + x^4 + x^3 + x^2 + 1$ , where rule vectors are mirror image of each other. Different rule vectors generate different codewords for same data block. Designers may select any one of the 16 rule vectors given in Table 1. As a case study, CA-based

**Table 1.** 8-bit CA-rule vectors for all primitive polynomials in  $GF(2^8)$

No.	Primitive Polynomial	Corresponding CA-rule vector
1	$x^8 + x^4 + x^3 + x^2 + 1$	00000110
2	$x^8 + x^5 + x^3 + x + 1$	01011111
3	$x^8 + x^5 + x^3 + x^2 + 1$	01110111
4	$x^8 + x^6 + x^3 + x^2 + 1$	01101100
5	$x^8 + x^6 + x^4 + x^3 + x^2 + x + 1$	10010011
6	$x^8 + x^6 + x^5 + x + 1$	11010010
7	$x^8 + x^6 + x^5 + x^2 + 1$	00101101
8	$x^8 + x^6 + x^5 + x^3 + 1$	00001111
9	$x^8 + x^6 + x^5 + x^4 + 1$	00111001
10	$x^8 + x^7 + x^2 + x + 1$	11101111
11	$x^8 + x^7 + x^3 + x^2 + 1$	00101010
12	$x^8 + x^7 + x^5 + x^3 + 1$	01000101
13	$x^8 + x^7 + x^6 + x + 1$	01011101
14	$x^8 + x^7 + x^6 + x^3 + x^2 + x + 1$	01011011
15	$x^8 + x^7 + x^6 + x^5 + x^2 + x + 1$	11001011
16	$x^8 + x^7 + x^6 + x^5 + x^4 + x^2 + 1$	11010101

3-byte and 4-byte error correcting codes are discussed in following subsection.

## 2.2 3-byte Error Correcting Code

This section explains the complete decoding algorithm for the CA-based three bytes error correcting code. Proposed scheme can detect and correct at most three errors and it is independent of error positions. In three byte error correcting code six check bytes are generated by using (1) and syndrome values are computed using (2). Depending on number of syndromes having value non-zero and zero some conclusion may be drawn. The results are shown in Table 2. In

**Table 2.** Number of non-zero syndromes vs. decision

Number of syndromes		Decision
Value zero	Value non-zero	
6	0	no error
5	1	1 check byte error
4	2	2 check bytes error
3	3	3 check bytes error
2	4	4 check bytes error / 2 check bytes and 1 information byte
1	5	two/three bytes error
0	6	one/two/three bytes error

case of 3-byte ECC, four possibilities may occur: no error, one error, two errors, three errors. The obvious distribution of errors are summarized in Table 3 to make the analysis more systematic. When total number of errors is same as the number of errors in check bytes then errors can be identified by finding the number of non-zero syndromes and it is shown in Table 2. Also, all syndromes having value zero indicates there is no error. In this section, we first describe

**Table 3.** Error Distribution

Total number of errors in bytes	Number of errors in	
	information bytes	check bytes
3	3	0
	0	3
	2	1
	1	2
2	2	0
	0	2
	1	1
1	1	0
	0	1

the decoding method where all three errors are concentrated within information bytes, from [8] for the sake of completeness.

**Three information bytes error** Suppose  $E_m$ ,  $E_n$  and  $E_o$  are the error magnitudes in the  $m$ -th,  $n$ -th, and  $o$ -th information bytes respectively with  $m \neq n \neq o$ , then corresponding syndrome equations are as follows

$$\begin{aligned} S_0 &= E_m \oplus E_n \oplus E_o ; S_1 = T^i E_m \oplus T^j E_n \oplus T^k E_o \\ S_2 &= T^{2i} E_m \oplus T^{2j} E_n \oplus T^{2k} E_o ; S_3 = T^{3i} E_m \oplus T^{3j} E_n \oplus T^{3k} E_o \\ S_4 &= T^{4i} E_m \oplus T^{4j} E_n \oplus T^{4k} E_o ; S_5 = T^{5i} E_m \oplus T^{5j} E_n \oplus T^{5k} E_o \end{aligned} \quad (4)$$

where  $S_0, S_1, S_2, S_3, S_4$  and  $S_5$  are six syndrome bytes and  $i + m = N$ ,  $j + n = N$ ,  $k + o = N$  and  $N$  is the number of information bytes in the codeword. The required equations to compute the error locations are as follows.

$$\begin{aligned} T^{2j}(T^i S_2 \oplus S_3) \oplus T^i S_4 \oplus S_5 &= T^{2k}(T^{2j}(T^i S_0 \oplus S_1) \oplus T^i S_2 \oplus S_3) \\ T^{2j}(T^{2i} S_1 \oplus S_3) \oplus T^{2i} S_3 \oplus S_5 &= T^k(T^{2j}(T^{2i} S_0 \oplus S_2) \oplus T^{2i} S_2 \oplus S_4) \\ T^j(T^{3i} S_1 \oplus S_4) \oplus T^{3i} S_2 \oplus S_5 &= T^k(T^j(T^{3i} S_0 \oplus S_3) \oplus T^{3i} S_1 \oplus S_4) \end{aligned} \quad (5)$$

Simultaneous solution of three equations estimates the three error locations. Error magnitudes in three bytes  $m$ ,  $n$  and  $o$  are calculated using following equations.

$$\begin{aligned} E_n &= T^{L-a}(T^{2k}(T^i S_0 \oplus S_1) \oplus T^i S_2 \oplus S_3) \\ E_o &= T^{L-b}(T^{2j}(T^i S_0 \oplus S_1) \oplus T^i S_2 \oplus S_3); E_m = S_0 \oplus E_o \oplus E_n \end{aligned} \quad (6)$$

where  $T^a = (T^i \oplus T^j)(T^{2j} \oplus T^{2k})$ ,  $T^b = (T^i \oplus T^k)(T^{2j} \oplus T^{2k})$  and  $L = 2^r - 1$  is the cycle length of a  $r$ -cell maximum length group CA.

**Two information bytes and one check byte error** There are six possibilities when any one of the six check bytes and any two information bytes are erroneous. Assume  $e_0$ ,  $E_m$  and  $E_n$  are the errors in check byte  $C_0$ ,  $m$ -th and  $n$ -th information bytes respectively, then we can write the syndrome equations as follows.

$$\begin{aligned} S_0 &= E_m \oplus E_n \oplus e_0 ; S_1 = T^i E_m \oplus T^j E_n ; S_2 = T^{2i} E_m \oplus T^{2j} E_n \\ S_3 &= T^{3i} E_m \oplus T^{3j} E_n ; S_4 = T^{4i} E_m \oplus T^{4j} E_n ; S_5 = T^{5i} E_m \oplus T^{5j} E_n \end{aligned} \quad (7)$$

From the Equations in (7), we get

$$\begin{aligned} T^i S_0 \oplus S_1 &= (T^i \oplus T^j) E_n \oplus T^i e_0 ; T^i S_2 \oplus S_3 = T^{2j}(T^i \oplus T^j) E_n \\ T^i S_4 \oplus S_5 &= T^{4j}(T^i \oplus T^j) E_n ; T^{2i} S_0 \oplus S_2 = (T^{2i} \oplus T^{2j}) E_n \oplus T^{2i} e_0 \\ T^{2i} S_1 \oplus S_3 &= T^j(T^{2i} \oplus T^{2j}) E_n ; T^{2i} S_2 \oplus S_4 = T^{2j}(T^{2i} \oplus T^{2j}) E_n \\ T^{2i} S_3 \oplus S_5 &= T^{3j}(T^{2i} \oplus T^{2j}) E_n ; T^{3i} S_0 \oplus S_3 = (T^{3i} \oplus T^{3j}) E_n \oplus T^{3i} e_0 \\ T^{3i} S_1 \oplus S_4 &= T^j(T^{3i} \oplus T^{3j}) E_n ; T^{3i} S_2 \oplus S_5 = T^{2j}(T^{3i} \oplus T^{3j}) E_n \end{aligned} \quad (8)$$

Combining the equations in (8), we get

$$\begin{aligned} T^{2j}(T^i S_2 \oplus S_3) \oplus T^i S_4 \oplus S_5 &= 0; T^{2j}(T^{2i} S_0 \oplus S_2) \oplus T^{2i} S_2 \oplus S_4 = T^{2j}(T^{2i} e_0) \\ T^j(T^{3i} S_0 \oplus S_3) \oplus T^{3i} S_1 \oplus S_4 &= T^j(T^{3i} e_0); T^{2j}(T^{2i} S_1 \oplus S_3) \oplus T^{2i} S_3 \oplus S_5 = 0 \\ T^{2j}(T^i S_0 \oplus S_1) \oplus T^i S_2 \oplus S_3 &= T^{2j}(T^i e_0); T^j(T^{3i} S_1 \oplus S_4) \oplus T^{3i} S_2 \oplus S_5 = 0 \end{aligned} \quad (9)$$

**Table 4.** Equations for computing at most 2-error in information bytes

U	V	W	X	Y	Z	Errors in	Error value
0	1	1	0	1	0	$C_0, D_m, D_n$	$E_n = T^{(L-\alpha)}(T^i S_1 \oplus S_2), E_m = T^{(L-\beta)}(T^j S_1 \oplus S_2)$
0	0	1	1	1	1	$C_1, D_m, D_n$	$E_n = T^{(L-\gamma)}(T^{2i} S_0 \oplus S_2), E_m = S_0 \oplus E_n$
1	1	0	0	1	1	$C_2, D_m, D_n$	$E_n = T^{(L-\delta)}(T^i S_0 \oplus S_1), E_m = S_0 \oplus E_n$
1	0	1	1	1	0	$C_3, D_m, D_n$	$E_n = T^{(L-\delta)}(T^i S_0 \oplus S_1), E_m = S_0 \oplus E_n$
1	1	1	0	0	1	$C_4, D_m, D_n$	$E_n = T^{(L-\delta)}(T^i S_0 \oplus S_1), E_m = S_0 \oplus E_n$
1	0	0	1	0	1	$C_5, D_m, D_n$	$E_n = T^{(L-\delta)}(T^i S_0 \oplus S_1), E_m = S_0 \oplus E_n$
0	0	0	0	0	0	$D_m, D_n$	$E_n = T^{(L-\delta)}(T^i S_0 \oplus S_1), E_m = S_0 \oplus E_n$

$$\begin{aligned}
U &= T^{2j}(T^i S_2 \oplus S_3) \oplus T^i S_4 \oplus S_5, V = T^{2j}(T^{2i} S_0 \oplus S_2) \oplus T^{2i} S_2 \oplus S_4 \\
W &= T^j(T^{3i} S_0 \oplus S_3) \oplus T^{3i} S_1 \oplus S_4, X = T^{2j}(T^{2i} S_1 \oplus S_3) \oplus T^{2i} S_3 \oplus S_5 \\
Y &= T^{2j}(T^i S_0 \oplus S_1) \oplus T^i S_2 \oplus S_3, Z = T^j(T^{3i} S_1 \oplus S_4) \oplus T^{3i} S_2 \oplus S_5 \\
T^\alpha &= T^j(T^i \oplus T^j), T^\beta = T^i(T^i \oplus T^j), T^\gamma = (T^{2i} \oplus T^{2j}), T^\delta = (T^i \oplus T^j)
\end{aligned}$$

Equations in (9) are used to determine the error locations  $m = N - i$  and  $n = N - j$ . From the syndrome equations in (7) we get

$$\begin{aligned}
T^i S_1 \oplus S_2 &= T^j(T^i \oplus T^j)E_n \text{ or } E_n = T^{L-\alpha}(T^i S_1 \oplus S_2) \\
T^j S_1 \oplus S_2 &= T^i(T^i \oplus T^j)E_m \text{ or } E_m = T^{L-\beta}(T^j S_1 \oplus S_2)
\end{aligned} \quad (10)$$

where  $T^\alpha = T^j(T^i \oplus T^j)$ ,  $T^\beta = T^i(T^i \oplus T^j)$  and  $L = 2^r - 1$ . Error magnitudes  $E_m$  and  $E_n$  are computed using equations in (10). Equations to determine the error locations and values for other five cases and the case discussed in this section are summarized in Table 4. Zero and non-zero value are represented by '0' and '1' in Table 4. For any  $e_b \neq 0$ ,  $T^p e_b \neq 0$ , where  $1 \leq p \leq 2^r - 1$  and  $e_b$  is the error in  $b$ -th check byte. Hence  $V = 1$ ,  $W = 1$  and  $Y = 1$  in first row of Table 4.

**Two information bytes error** Assume  $E_m$  and  $E_n$  are the errors in the  $m$ -th and  $n$ -th information byte respectively. Then syndromes are as follows.

$$\begin{aligned}
S_0 &= E_m \oplus E_n ; S_1 = T^i E_m \oplus T^j E_n ; S_2 = T^{2i} E_m \oplus T^{2j} E_n \\
S_3 &= T^{3i} E_m \oplus T^{3j} E_n ; S_4 = T^{4i} E_m \oplus T^{4j} E_n ; S_5 = T^{5i} E_m \oplus T^{5j} E_n
\end{aligned} \quad (11)$$

Combining the equations in (11), we get

$$\begin{aligned}
T^{2j}(T^i S_2 \oplus S_3) \oplus T^i S_4 \oplus S_5 &= 0; T^{2j}(T^{2i} S_0 \oplus S_2) \oplus T^{2i} S_2 \oplus S_4 = 0 \\
T^j(T^{3i} S_0 \oplus S_3) \oplus T^{3i} S_1 \oplus S_4 &= 0; T^{2j}(T^{2i} S_1 \oplus S_3) \oplus T^{2i} S_3 \oplus S_5 = 0 \\
T^{2j}(T^i S_0 \oplus S_1) \oplus T^i S_2 \oplus S_3 &= 0; T^j(T^{3i} S_1 \oplus S_4) \oplus T^{3i} S_2 \oplus S_5 = 0
\end{aligned} \quad (12)$$

Error locations  $m = N - i$  and  $n = N - j$  are determined using equation (12). Using syndrome equations in (11), we can write

$$T^i S_0 \oplus S_1 = (T^i \oplus T^j)E_n \text{ or } E_n = T^{(L-\delta)}(T^i S_0 \oplus S_1); E_m = S_0 \oplus E_n \quad (13)$$

where  $T^\delta = T^i \oplus T^j$  and  $L = 2^r - 1$ . Error magnitudes may be found using equation in (13). Equations are also shown in the last row of Table 4

**One information byte and two check bytes error** There are fifteen situations when any two check bytes and any one information byte are erroneous. In this section, out of the fifteen cases the proof of one case is given as follows and the results for the other cases are given in Table 5 for the sake of brevity. Assume  $e_0$ ,  $e_1$  and  $E_m$  are the error values in check bytes  $C_0$ ,  $C_1$  and  $m$ -th information byte respectively. Hence, corresponding syndrome equations are as follows.

$$\begin{aligned} S_0 &= E_m \oplus e_0 ; S_1 = T^i E_m \oplus e_1 ; S_2 = T^{2i} E_m \\ S_3 &= T^{3i} E_m ; S_4 = T^{4i} E_m ; S_5 = T^{5i} E_m \end{aligned} \quad (14)$$

From syndrome equations in (14), we get

$$\begin{aligned} T^i S_0 \oplus S_1 &= T^i e_0 \oplus e_1 ; T^i S_2 \oplus S_3 = 0 ; T^i S_4 \oplus S_5 = 0 ; T^{2i} S_2 \oplus S_4 = 0 \\ T^{2i} S_3 \oplus S_5 &= 0 ; T^{2i} S_0 \oplus S_2 = T^{2i} e_0 ; T^{2i} S_1 \oplus S_3 = T^{2i} e_1 \\ T^{3i} S_1 \oplus S_4 &= T^{3i} e_1 ; T^{3i} S_2 \oplus S_5 = 0 ; T^{3i} S_0 \oplus S_3 = T^{3i} e_0 \end{aligned} \quad (15)$$

For any non-zero value of  $e_0$  and  $e_1$ ,  $T^g e_0 \neq 0$  and  $T^h e_1 \neq 0$ , where  $1 \leq g, h \leq (2^r - 1)$ . Let  $A = T^i e_0 \oplus e_1$ , then the value of  $A$  may or may not be zero, because it depends on value of  $e_0$ ,  $e_1$  and  $i$ . In Table 5, x, 1 and 0 indicate the don't care, non-zero value and zero value respectively.

**One information byte error** Consider the case when only one information byte is erroneous. Syndrome equations in this case are as follows.

$$\begin{aligned} S_0 &= E_m ; S_1 = T^i E_m ; S_2 = T^{2i} E_m \\ S_3 &= T^{3i} E_m ; S_4 = T^{4i} E_m ; S_5 = T^{5i} E_m \end{aligned} \quad (16)$$

Combining the syndrome equations in (16), we get

$$\begin{aligned} T^i S_0 \oplus S_1 &= 0 ; T^i S_2 \oplus S_3 = 0 ; T^i S_4 \oplus S_5 = 0 ; T^{2i} S_2 \oplus S_4 = 0 \\ T^{2i} S_3 \oplus S_5 &= 0 ; T^{2i} S_0 \oplus S_2 = 0 ; T^{2i} S_1 \oplus S_3 = 0 ; T^{3i} S_1 \oplus S_4 = 0 \\ T^{3i} S_2 \oplus S_5 &= 0 ; T^{3i} S_0 \oplus S_3 = 0 \end{aligned} \quad (17)$$

If all the equations in (17) are satisfied then one error in  $(N - i)$ -th information byte is identified and the corresponding error magnitude is  $E_m = S_0$ .

**One information byte and one check byte error** There are six situations in which any one of the information bytes and any one of the check bytes are erroneous. Consider the case when  $E_m$  and  $e_0$  are the errors in  $m$ -th information byte and check byte  $C_0$  respectively. Syndromes are computed using (1).

$$\begin{aligned} S_0 &= E_m \oplus e_0 ; S_1 = T^i E_m ; S_2 = T^{2i} E_m \\ S_3 &= T^{3i} E_m ; S_4 = T^{4i} E_m ; S_5 = T^{5i} E_m \end{aligned} \quad (18)$$

**Table 5.** Equations for computing at most 1 error in information byte

A	B	C	D	E	F	G	H	I	J	Errors in	Error value
x	0	0	0	0	1	1	1	0	1	$C_0, C_1, D_m$	$T^{L-2i}S_2$
1	1	0	1	0	x	0	0	1	1	$C_0, C_2, D_m$	$T^{L-i}S_1$
1	1	0	0	1	1	1	0	0	x	$C_0, C_3, D_m$	$T^{L-i}S_1$
1	0	1	1	0	1	0	1	0	1	$C_0, C_4, D_m$	$T^{L-i}S_1$
1	0	1	0	1	1	0	0	1	1	$C_0, C_5, D_m$	$T^{L-i}S_1$
1	1	0	1	0	1	1	1	1	0	$C_1, C_2, D_m$	$S_0$
1	1	0	0	1	0	x	1	0	1	$C_1, C_3, D_m$	$S_0$
1	0	1	1	0	0	1	x	0	0	$C_1, C_4, D_m$	$S_0$
1	0	1	0	1	0	1	1	1	0	$C_1, C_5, D_m$	$S_0$
0	x	0	1	1	1	1	0	1	1	$C_2, C_3, D_m$	$S_0$
0	1	1	x	0	1	0	1	1	0	$C_2, C_4, D_m$	$S_0$
0	0	1	1	1	1	1	0	0	x	$C_2, C_5, D_m$	$S_0$
0	1	1	1	1	0	1	1	0	1	$C_3, C_4, D_m$	$S_0$
0	1	1	0	x	0	1	0	1	1	$C_3, C_5, D_m$	$S_0$
0	0	x	1	1	0	0	1	1	0	$C_4, C_5, D_m$	$S_0$
0	0	0	0	0	0	0	0	0	0	$D_m$	$S_0$
1	0	0	0	0	1	0	0	0	1	$C_0, D_m$	$T^{L-2i}S_1$
1	0	0	0	0	0	1	1	0	0	$C_1, D_m$	$S_0$
0	1	0	1	0	1	0	0	1	0	$C_2, D_m$	$S_0$
0	1	0	0	1	0	1	0	0	1	$C_3, D_m$	$S_0$
0	0	1	1	0	0	0	1	0	0	$C_4, D_m$	$S_0$
0	0	1	0	1	0	0	0	1	0	$C_5, D_m$	$S_0$

$$\begin{aligned}
A &= T^i S_0 \oplus S_1, B = T^i S_2 \oplus S_3, C = T^i S_4 \oplus S_5, D = T^{2i} S_2 \oplus S_4 \\
E &= T^{2i} S_3 \oplus S_5, F = T^{2i} S_0 \oplus S_2, G = T^{2i} S_1 \oplus S_3, H = T^{3i} S_1 \oplus S_4 \\
I &= T^{3i} S_2 \oplus S_5, \text{ and } J = T^{3i} S_0 \oplus S_3
\end{aligned}$$

Combining the syndrome equations in (18), we get

$$\begin{aligned}
T^i S_0 \oplus S_1 &= T^i e_0 \neq 0 ; T^i S_2 \oplus S_3 = 0 ; T^i S_4 \oplus S_5 = 0 ; T^{2i} S_2 \oplus S_4 = 0 \\
T^{2i} S_3 \oplus S_5 &= 0 ; T^{2i} S_0 \oplus S_2 = T^{2i} e_0 \neq 0 ; T^{2i} S_1 \oplus S_3 = 0 \\
T^{3i} S_1 \oplus S_4 &= 0 ; T^{3i} S_2 \oplus S_5 = 0 ; T^{3i} S_0 \oplus S_3 = T^{3i} e_0 \neq 0
\end{aligned} \tag{19}$$

If all the equations in (19) are satisfied then one error in  $(N-i)$ -th information byte and another error in check byte  $C_0$  are identified. The error magnitude is  $E_m = T^{L-i}S_1$ . The required equations for other five cases are summarized in Table 5 for the sake of brevity. Therefore, it is possible to locate and correct all three-byte errors. It is noted that all expressions  $A, B, \dots, J$  in table 5 and all expressions  $U, V, \dots, Z$  in table 4 are part of the equations in (5). The following section shows how the CA-based 3-byte ECC can be extended for 4-byte ECC.



### 2.3 4-byte Error Correcting Code

In case of CA-based 4-byte error correcting code, eight check bytes are computed using (1) and the decoder computes the eight syndrome bytes employing (2). In this section, we derive the equations to determine the error locations and the error magnitudes, provided the errors are concentrated in the information bytes only. Suppose  $S_0, S_1, S_2, S_3, S_4, S_5, S_6, S_7$  are the eight syndromes and  $E_m, E_n, E_o, E_p$  are the errors in the  $m$ -th,  $n$ -th,  $o$ -th,  $p$ -th information bytes respectively, where  $m \neq n \neq o \neq p$ .

$$\begin{aligned}
S_0 &= E_m \oplus E_n \oplus E_o \oplus E_p \\
S_1 &= T^i E_m \oplus T^j E_n \oplus T^k E_o \oplus T^l E_p \\
S_2 &= T^{2i} E_m \oplus T^{2j} E_n \oplus T^{2k} E_o \oplus T^{2l} E_p \\
S_3 &= T^{3i} E_m \oplus T^{3j} E_n \oplus T^{3k} E_o \oplus T^{3l} E_p \\
S_4 &= T^{4i} E_m \oplus T^{4j} E_n \oplus T^{4k} E_o \oplus T^{4l} E_p \\
S_5 &= T^{5i} E_m \oplus T^{5j} E_n \oplus T^{5k} E_o \oplus T^{5l} E_p \\
S_6 &= T^{6i} E_m \oplus T^{6j} E_n \oplus T^{6k} E_o \oplus T^{6l} E_p \\
S_7 &= T^{7i} E_m \oplus T^{7j} E_n \oplus T^{7k} E_o \oplus T^{7l} E_p
\end{aligned} \tag{20}$$

Using the syndrome equations in (20), the following four equations can be formulated to compute four error locations, provided all the errors are concentrated in the information bytes only.

$$\begin{aligned}
&T^{2k}(T^{2j}(T^i S_2 \oplus S_3) \oplus T^i S_4 \oplus S_5) \oplus T^{2j}(T^i S_4 \oplus S_5) \oplus T^i S_6 \oplus S_7 = \\
&T^{2l}(T^{2k}(T^{2j}(T^i S_0 \oplus S_1) \oplus T^i S_2 \oplus S_3) \oplus T^{2j}(T^i S_2 \oplus S_3) \oplus T^i S_4 \oplus S_5); \\
&T^k(T^j(T^{4i} S_1 \oplus S_5) \oplus T^{4i} S_2 \oplus S_6) \oplus T^j(T^{4i} S_2 \oplus S_6) \oplus T^{4i} S_3 \oplus S_7 = \\
&T^l(T^k(T^j(T^{4i} S_0 \oplus S_4) \oplus T^{4i} S_1 \oplus S_5) \oplus T^j(T^{4i} S_1 \oplus S_5) \oplus T^{4i} S_2 \oplus S_6); \\
&T^k(T^j(T^{3i} S_2 \oplus S_5) \oplus T^{3i} S_3 \oplus S_6) \oplus T^j(T^{3i} S_3 \oplus S_6) \oplus T^{3i} S_4 \oplus S_7 = \\
&T^{2l}(T^k(T^j(T^{3i} S_0 \oplus S_3) \oplus T^{3i} S_1 \oplus S_4) \oplus T^j(T^{3i} S_1 \oplus S_4) \oplus T^{3i} S_2 \oplus S_5); \\
&T^{2k}(T^j(T^{2i} S_2 \oplus S_4) \oplus T^{2i} S_3 \oplus S_5) \oplus T^j(T^{2i} S_4 \oplus S_6) \oplus T^{2i} S_5 \oplus S_7 = \\
&T^{2l}(T^{2k}(T^j(T^{2i} S_0 \oplus S_2) \oplus T^{2i} S_1 \oplus S_3) \oplus T^j(T^{2i} S_2 \oplus S_4) \oplus T^{2i} S_3 \oplus S_5) \tag{21}
\end{aligned}$$

Simultaneous solution of the equations in (21) determines the four error positions within the information bytes, where  $m = N - i$ ,  $n = N - j$ ,  $o = N - k$  and  $p = N - l$  and  $N$  is the number of information bytes in the code word. The four error magnitudes  $E_m, E_n, E_o$  and  $E_p$  are calculated using the following equations.

$$\begin{aligned}
E_n &= T^{L-a}(T^{2k}(T^{2l}(T^i S_0 \oplus S_1) \oplus T^i S_2 \oplus S_3) \oplus T^{2l}(T^i S_2 \oplus S_3) \oplus T^i S_4 \oplus S_5) \\
E_o &= T^{L-b}(T^{2j}(T^{2l}(T^i S_0 \oplus S_1) \oplus T^i S_2 \oplus S_3) \oplus T^{2l}(T^i S_2 \oplus S_3) \oplus T^i S_4 \oplus S_5) \\
E_p &= T^{L-c}(T^{2j}(T^{2k}(T^i S_0 \oplus S_1) \oplus T^i S_2 \oplus S_3) \oplus T^{2k}(T^i S_2 \oplus S_3) \oplus T^i S_4 \oplus S_5) \\
E_m &= S_0 \oplus E_n \oplus E_o \oplus E_p
\end{aligned} \tag{22}$$

where  $T^a = (T^j \oplus T^i)(T^{2j} \oplus T^{2k})(T^{2j} \oplus T^{2l})$ ;  $T^b = (T^k \oplus T^i)(T^{2j} \oplus T^{2k})(T^{2k} \oplus T^{2l})$ ;  $T^c = (T^l \oplus T^i)(T^{2j} \oplus T^{2l})(T^{2k} \oplus T^{2l})$  and  $L = 2^r - 1$  is the cycle length of a  $r$ -cell maximum length group CA. Similar to a 3-byte ECC, it is possible to correct the errors when the errors are distributed between information and the check bytes.

One disadvantage of the proposed error correcting code is that the error location identification block has a time complexity of  $N^t$ , where  $N$  is the number of information bytes and  $t$  is the number of errors to be corrected. But the decoding time can be reduced by duplicating the error location identification module. Therefore, the proposed scheme is suitable for a code having a smaller number of error correction capability and smaller data word length such as (23, 17), (32, 26) and (32, 24) codes, which have practical applications in wireless communications.

### 3 Conclusion

This paper presents an improved scheme for multi-byte error correcting code using CA which overcomes the weaknesses and limitation of existing scheme. As a case study full decoding algorithm for three byte error correcting code is presented. Proposed CA-based four byte error correcting code can detect and correct at most 4 errors if all errors are concentrated in information or check bytes.

### References

1. Chen, C. L.: Error-correcting codes for byte-organized memory systems. IEEE Trans. Information Theory 32(2), 181-185 (1986)
2. Roy Chowdhury, D., Basu, S., Sen Gupta, I., Chaudhuri, P.P.: Design of CAECC-Cellular Automata Based Error Correcting Code. IEEE Transaction on Computers 43(6), 759-764 (1994)
3. Roy Chowdhury, D., Sen Gupta, I., Chaudhuri, P.P.: CA-Based Byte Error-Correcting code. IEEE Transaction on Computers 44(3), 371-382 (1995)
4. Nandi, S., Rambabu, C., Chaudhuri, P.P.: A VLSI Architecture for Cellular Automata Based Reed-Solomon Decoder. In: 4th International Symposium on Parallel Architecture, Algorithm and Networks, Australia, pp. 158-165 (1999)
5. Sasidhar, K., Chattopadhyay, S., Chaudhuri, P.P.: CAA Decoder for Cellular Automata Based Byte Error Correcting Code. IEEE Transaction on Computers 45(9), 1003-1016 (1996)
6. Paul, K., Roy Chowdhury, D.: Application of  $GF(2^p)$ CA in Burst Error Correcting Codes. In: 13th International Conference on VLSI Design, India, pp. 562-567 (2000)
7. Bhaumik, J., Roy Chowdhury, D., Chakrabarti, I.: An Improved Double Byte Error Correcting Code using Cellular Automata. In: 8th International Conference on Cellular Automata for Research and Industry, Japan, LNCS 5191, pp. 463-470 (2008)
8. Bhaumik, J., Roy Chowdhury, D.: New Architectural Design of CA-Based Codec. IEEE Transactions on Very Large Scale Integration Systems (in press)

9. Bhaumik, J. Roy Chowdhury, D.: An Integrated ECC-MAC Based on RS Code. Transactions on Computational Science LNCS 5430, 117-135 (2009)
10. Lin, S., Costello, D.J.: Error Control Coding: Fundamentals and Applications. Prentice-Hall, Inc., Englewood Cliffs (1983)