An Improved Double Byte Error Correcting Code Using Cellular Automata

Jaydeb Bhaumik, Dipanwita Roy Chowdhury, and Indrajit Chakrabarti

Indian Institute of Technology,
Kharagpur 721302, India
{jaydeb.gssst,drc.cse,indrajit.ece}@iitkgp.ernet.in

Abstract. Cellular Automata(CA) based VLSI implementation of t-byte errors correcting code has been established by previous research to be superior to the other existing techniques employed for realizing Reed-Solomon(RS) code. However, the scheme suffers from the limitation that it can correct $t-$byte errors ($t \geq 2$) provided errors are confined either wholly to the information bytes or entirely to the check bytes. The work reported in the present paper overcomes this limitation and corrects the errors likely in both information and check bytes. Moreover one weakness found in an earlier similar work has been identified and rectified using a modified check symbol expressions.

Keywords: Byte Error Correcting Code, Cellular Automata, RS Code and VLSI.

1 Introduction

RS code has found many applications in storage devices (CD, DVD), wireless communications, high speed modems and satellite communications. The complexity of RS encoder and decoder increases with the error correcting capability of the code. Hence many researchers have put their effort to minimize the complexity of RS encoder/decoder for communication applications. A number of general encoding and decoding schemes of the RS code is available in the literature [7, 8].

But VLSI system designer always prefers to have simple, regular, modular and cascadable structure with local interconnection for reliable high speed operations of the circuit. It has been found that these parameters are supported by local neighborhood Cellular Automata (CA). In [1] CA based byte error correcting code has been proposed. The proposed design in [1] requires less hardware compared to the existing techniques used for RS code. A new scheme has been proposed for pipeline implementation of CA based tbEC - tbED codes that are analogous to the conventional RS code in [2]. Another design scheme has been reported for parallel implementation of CA based SbEC/DbED and DbEC/DbED code that is also analogous to the conventional RS code in [3]. A new high speed VLSI architecture for decoding RS codes with Berlekamp-Massey (BM) algorithm has been published in [4]. In this scheme, the speed bottleneck in BM

H. Umeo et al. (Eds): ACRI 2008, LNCS 5191, pp. 463–470, 2008.

algorithm is eliminated by using fully systolic architecture. A new degree computationless modified Euclid (DCME) algorithm and its dedicated architecture for RS decoder has been reported in [5]. The architecture has low hardware complexity compared with conventional modified Euclid architecture.

However, the scheme in [1] can correct t-byte errors ($t \geq 2$) provided errors are totally confined to information or check byte only. The scheme[1] fails if one error is in information byte and another in check byte. Another weakness of the scheme in [1] is that single byte error in k^{th} information byte and double byte errors one in k^{th} information byte and another in the last information byte correspond to the same equation for error location identification.

In this paper, an improved double byte error correcting code using CA has been proposed. The new scheme over comes the limitation of [1] and can correct errors even if one error occurs in the information byte and another is in the check byte. Also the scheme can unambiguously determine error locations. CA-based VLSI design is attractive because of its simplicity, regularity and higher throughput.

The rest of this paper is organized as follows. In the next section, a brief overview of existing CA-based double byte error correcting code is described. Then, we describe the weakness and limitation of the existing CA-based double byte error correcting code in section 3. Section 4 discusses the proposed new scheme and finally the paper is concluded in section 5.

2 CA-Based Double Byte Error Correcting Code

In this section, we discuss the preliminaries of CA and CA based double byte error correcting code which has been proposed in [1].

2.1 Cellular Automata Preliminaries

A cellular automata (CA) consists of a number of cells arranged in a regular manner, where the state transitions of each cell depends on the state of its neighbors and each cell consists of a storage element (D flip-flop) and a combinational logic implementing the next-state function. The next-state function for a three-neighborhood CA cell can be expressed as follows.

$$q_i(t+1) = f[q_{i-1}(t), q_i(t), q_{i+1}(t)]$$

where $q_i(t)$ is the output state of the i^{th} cell at time t and f is the next state function also called the rule of the automata[6]. An n-cell CA can be characterized by an $n \times n$ characteristic matrix T. S_{t+1} can be computed by multiplying $[S_t]$ with T, where S_t and S_{t+1} represent the states of the CA at t^{th} and $(t+1)^{th}$ time instant respectively. This matrix has additional properties of being tridiagonal and having a primitive characteristic polynomial. This guarantees that linear recurrence has maximum period $2^n - 1$. Detailed information on CA may be found in [6]. Only linearity property of CA is used in byte error correcting code.

2.2 Overview of the CA-Based Double Byte Error Correcting Code

CA-based byte error correcting code [1] is similar to extended RS code. But compared to RS code, the proposed code is much simpler and requires much less hardware. In two byte error correcting code, encoder generates four check bytes from a block of N-byte information. After that the check symbols are appended to information bytes to form the code word. Now four check bytes should be generated by running the CA for N cycles, while sequentially feeding the N information bytes (D_k), where $0 \leq k \leq (N-1)$. The four check bytes are as follows:

$$C_0 = D_{N-1} \oplus D_{N-2} \oplus ... \oplus D_0 \tag{1}$$

$$C_1 = D_{N-1} \oplus T[D_{N-2}] \oplus ... \oplus T^{N-1}[D_0] \tag{2}$$

$$C_2 = D_{N-1} \oplus T^2[D_{N-2}] \oplus ... \oplus T^{2(N-1)}[D_0] \tag{3}$$

$$C_3 = D_{N-1} \oplus T^3[D_{N-2}] \oplus ... \oplus T^{3(N-1)}[D_0] \tag{4}$$

The primary objective of decoding is to retrieve the correct information byte. Decoding is done by employing the properties of maximum length group CA. The syndrome corresponding to the q^{th} check byte, S_q is defined as

$$S_q = C_q \oplus C_q^{'} \quad ; \quad 0 \leq q \leq 3. \tag{5}$$

where C_q is the q^{th} received check byte and $C_q^{'}$ is the q^{th} check byte recomputed from the received information bytes (with possibility of error present).

Decoding Algorithm

step 1: If all the syndrome bytes S_0, S_1, S_2, S_3 are zeros, then there is no error in the received information block.

step 2: If any one or two of the syndrome byte(s) is/are nonzero but the other are zeros, then the check byte(s) is/are in error.

step 3: If more than two syndrome bytes are nonzero then any one of the following three cases may occur.

1) One error is in the information byte

2) Double byte errors are in the information byte.

3) One error is in the information byte and other error is in the check byte.

step 4: In case none of the above conditions regarding the syndrome bytes hold, then more than two errors have occurred.

Suppose two byte errors have occurred in two information bytes. If E_k and E_l are the errors in k^{th} and l^{th} information bytes, then the corresponding syndrome equations are

$$S_0 = E_k \oplus E_l \tag{6}$$

$$S_1 = T^i[E_k] \oplus T^j[E_l] \tag{7}$$

$$S_2 = T^{2i}[E_k] \oplus T^{2j}[E_l] \tag{8}$$

$$S_3 = T^{3i}[E_k] \oplus T^{3j}[E_l] \tag{9}$$

Also $i + k = N - 1$ and $j + l = N - 1$, where $0 \leq k, l \leq N - 1$. If at least three syndromes are non-zero and there exist two integers i and j $(0 \leq i, j \leq (N-1))$ such that

$$T^i[S_2] \oplus S_3 = T^{2j}(T^i[S_0] \oplus S_1) \tag{10}$$

$$T^{2i}[S_1] \oplus S_3 = T^j(T^{2i}[S_0] \oplus S_2) \tag{11}$$

then the k^{th} and the l^{th}information bytes where $k = N-1-i$ and $l = N-1-j$ are erroneous. Let $T^i \oplus T^j = T^y$. Again, if $L = 2^n - 1$ is the cycle length of a n-cell maximum length group CA, then $(T^y)^{-1} = T^{-y} = T^{(L-y)} = T^p$. The error magnitudes are determined using the two equations given below.

$$E_l = T^p(T^i[S_0] \oplus S_1) \tag{12}$$

$$E_k = S_0 \oplus E_l \tag{13}$$

If $D_m^{'}$, E_m are the received m^{th} information byte and the calculated m^{th} error byte respectively, then the correct information byte can be obtained as

$$D_m = D_m^{'} \oplus E_m; \qquad where\ 0 \leq m \leq (N-1) \tag{14}$$

The next section reports the weakness and limitation of scheme [1].

3 Weakness and Limitation of the CA-Based Double Byte Error Correcting Code

CA based byte error correcting code proposed in [1] is very good from VLSI implementation point of view. But the scheme has a weakness and one limitation. This section explains the weakness and limitation of the scheme proposed in [1].

3.1 Weakness of the CA-Based Double Byte Error Correcting Code

Case 1 : Single byte error in the information byte
Assume that a single byte error has occurred in the k^{th} information byte. If E_k is the error in k^{th} byte, then according to [1] corresponding syndrome equations are

$$S_0 = E_k; \quad S_1 = T^i[E_k]; \quad S_2 = T^{2i}[E_k]; \quad S_3 = T^{3i}[E_k] \tag{15}$$

where S_0, S_1, S_2, S_3 are the four syndrome bytes. From (15) we get

$$S_1 \oplus S_3 = T^i(S_0 \oplus S_2) \tag{16}$$

$$S_2 \oplus S_3 = T^{2i}(S_0 \oplus S_1) \tag{17}$$

Equations (16) and (17) may be used to determine the single byte error location in information byte. But we can't use the two equations if the error is in the last information byte. For an error in the last information byte the syndromes are all equal i.e. $S_0 = S_1 = S_2 = S_3 = E_{N-1}$, where E_{N-1} is the error in the last information byte. Therefore, $S_2 \oplus S_0 = 0;\ S_1 \oplus S_3 = 0;\ S_1 \oplus S_0 = 0;\ S_3 \oplus S_2 = 0$

and equations (16) and (17) will be satisfied for all i and we can not determine error location uniquely.

Case 2 : One error in the $(N-1)^{th}$ ***i.e. the last information byte and the other in the*** k^{th} ***information byte***
The syndrome equations according to [1] are

$$S_0 = E_k \oplus E_{N-1} \tag{18}$$

$$S_1 = T^i E_k \oplus E_{N-1} \tag{19}$$

$$S_2 = T^{2i} E_k \oplus E_{N-1} \tag{20}$$

$$S_3 = T^{3i} E_k \oplus E_{N-1} \tag{21}$$

Using above four equations we get

$$S_1 \oplus S_3 = T^i (S_0 \oplus S_2) \tag{22}$$

$$S_2 \oplus S_3 = T^{2i} (S_0 \oplus S_1) \tag{23}$$

It is observed that the equations (16), (17) in case1 are same as equations (22), (23) in case2. So it is impossible to compute error location uniquely. But we can separate the case1 and case2 by checking any one of the additional conditions.

$$T^i S_0 \neq S_1;\ T^i S_1 \neq S_2;\ T^i S_2 \neq S_3 \tag{24}$$

When any one of the above three conditions and equations (16), (17) are satisfied then one error in the k^{th} information byte and another in the $(N-1)^{th}$ information byte is identified. So it requires only one extra checking.

3.2 Limitation of the CA-Based Double Byte Error Correcting Code

The scheme [1] suffers from the limitation that it can correct errors provided errors are totally confined to information or check bytes only. If one error is in the information byte and another is in the check byte, then the error is located in maximum of three information byte position. In the next section, we modify the scheme [1] to overcome the above weakness and limitation.

4 Our Improved Scheme

In our improved double byte error correcting code, check bytes are generated by running the CA for N cycles while sequentially feeding the N information bytes.

Algorithm to compute Check byte C_i
begin
$X := 0$; (X denotes the CA state)
for $k = 0$ *to* $N-1$ **do**

begin
$X := X \oplus D_k$;
Run the CA for one cycle;(CA with characteristic matrix T^i)
end;
$C_i := X$;
end;

For double byte error correcting code, the expression for check symbols C_0, C_1, C_2 and C_3 are given below.

$$C_0 = D_{N-1} \oplus D_{N-2} \oplus ... \oplus D_0 \tag{25}$$

$$C_1 = TD_{N-1} \oplus T^2[D_{N-2}] \oplus ... \oplus T^N[D_0] \tag{26}$$

$$C_2 = T^2 D_{N-1} \oplus T^{2(2)}[D_{N-2}] \oplus ... \oplus T^{2(N)}[D_0] \tag{27}$$

$$C_3 = T^3 D_{N-1} \oplus T^{3(2)}[D_{N-2}] \oplus ... \oplus T^{3(N)}[D_0] \tag{28}$$

Next we will derive the equations to identify the error locations from the received information and check bytes. Suppose two byte errors have occurred in k^{th} and l^{th} information bytes with $k \neq l$. The corresponding syndrome equations are

$$S_0 = E_k \oplus E_l \tag{29}$$

$$S_1 = T^i[E_k] \oplus T^j[E_l] \tag{30}$$

$$S_2 = T^{2i}[E_k] \oplus T^{2j}[E_l] \tag{31}$$

$$S_3 = T^{3i}[E_k] \oplus T^{3j}[E_l] \tag{32}$$

where S_0, S_1, S_2, S_3 are the four syndrome bytes and E_k and E_l are the corresponding errors in the k^{th} and l^{th} bytes. Also $i + k = N$ and $j + l = N$, where $1 \leq i, j \leq N$. From the above four syndrome equations we get

$$T^i[S_2] \oplus S_3 = T^{2j}(T^i[S_0] \oplus S_1) \tag{33}$$

$$T^{2i}[S_1] \oplus S_3 = T^j(T^{2i}[S_0] \oplus S_2) \tag{34}$$

So using the above two equations we can determine the error locations if both the errors occur in the information byte.

Theorem 1. The scheme identifies the error locations uniquely if single/double byte(s) error occurred and it is independent of error position.

Proof We establish the result for the following cases where indistinguishability of error location identification equations can happen.

Case 1 : One error in the $(N-1)^{th}$ ***i.e. the last information byte and the other in the*** k^{th} ***information byte***

According to proposed scheme the equations for error location identification are

$$T[S_2] \oplus S_3 = T^{2i}(T[S_0] \oplus S_1) \tag{35}$$

$$T^2[S_1] \oplus S_3 = T^i(T^2[S_0] \oplus S_2) \tag{36}$$

Case 2 : Single byte error in the k^{th} information byte
The equations for error location identification are

$$T^i(S_2 \oplus S_0) = S_1 \oplus S_3 \tag{37}$$

$$T^{2i}(S_1 \oplus S_0) = S_3 \oplus S_2 \tag{38}$$

If single byte error occurs in the last information byte then also we can identify the error location using equations (37) and (38). According to our scheme $S_0 \neq S_1 \neq S_2 \neq S_3$, so above two equations will be satisfied for a particular i.

It is observed that equations (35), (36) in case1 are different from equations (37), (38) in case 2. Thus, it overcomes the weakness of scheme [1] given in section 3.1. Note that we can also derive the equations in case1 and case2 from equations (33) and (34). Next we describe the situation where one error is in the information byte and other is in the check byte.

Two Byte Error location identification when one information byte and one check byte are erroneous

Theorem 2. The scheme decodes correctly if one error is in the information byte and another is in check byte.

Proof. Assume e_0, e_1, e_2 and e_3 are the errors in the 1st, 2nd, 3rd and 4th check byte respectively. If one error is in k^{th} information byte and another is in any one of the four check bytes, then any one of the four different cases may occur.

1. If 1st check byte and k^{th} information byte are erroneous, then the syndrome equations are

$$S_0 = E_k \oplus e_0; \quad S_1 = T^i E_k; \quad S_2 = T^{2i} E_k; \quad S_3 = T^{3i} E_k \tag{39}$$

$$S_3 = T^i S_2; \quad S_3 = T^{2i} S_1; \quad S_1 \neq T^i S_0 \tag{40}$$

$$E_k = T^{L-i} S_1 \tag{41}$$

Equation (40) is used to determine the error location k, where $k + i = N$ and equation (41) is used to determine error magnitude.

2. If 2nd check byte and k^{th} information byte are erroneous, then the syndrome equations are

$$S_0 = E_k; \quad S_1 = T^i E_k \oplus e_1; \quad S_2 = T^{2i} E_k; \quad S_3 = T^{3i} E_k \tag{42}$$

$$S_3 = T^i S_2; \quad S_2 = T^{2i} S_0; \quad S_1 \neq T^i S_0 \tag{43}$$

$$E_k = S_0 \tag{44}$$

Equation (43) is used to determine the error location k and equation (44) is used to determine error magnitude.

3. If 3rd check byte and k^{th} information byte are erroneous, then the syndrome equations are

$$S_0 = E_k; \quad S_1 = T^i E_k; \quad S_2 = T^{2i} E_k \oplus e_2; \quad S_3 = T^{3i} E_k \tag{45}$$

$$S_1 = T^i S_0; \quad S_3 = T^{2i} S_1; \quad S_3 \neq T^i S_2 \tag{46}$$

$$E_k = S_0 \tag{47}$$

Equations (46) are used to determine the error location k and equation (47) is used to determine error magnitude.

4. If 4th check byte and k^{th} information byte are erroneous, then the syndrome equations are

$$S_0 = E_k; \quad S_1 = T^i E_k; \quad S_2 = T^{2i} E_k; \quad S_3 = T^{3i} E_k \oplus e_3 \tag{48}$$

$$S_1 = T^i S_0; \quad S_2 = T^{2i} S_0; \quad S_3 \neq T^i S_2 \tag{49}$$

$$E_k = S_0 \tag{50}$$

Equation (49) is used to determine the error location k and equation (50) is used to determine error magnitude.

So, the proposed scheme can identify error locations unambiguously if one error is in information byte and the other is in check byte. Thus it overcomes the limitation of scheme [1] given in section 3.2.

5 Conclusion

The paper presents an improved scheme for the double byte error correcting code using CA which overcomes the weakness and limitation of the existing scheme. The proposed scheme can determine error locations which is independent of erroneous byte position and number of errors provided number of errors are less than or equal to error correcting capability of the code. The proposed code is much simpler and requires much less hardware for decoding compared with conventional RS code having two-byte error correcting capability.

References

1. Roy Chowdhury, D., Sen Gupta, I., Chaudhuri, P.P.: CA-Based Byte Error- Correcting code. IEEE Transaction on Computers 44(3), 371–382 (1995)
2. 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)
3. 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)
4. Sarwate, D.V., Shanbhag, N.R.: High-Speed Architectures for Reed-Solomon Decoder. IEEE Transaction on VLSI systems 9(5), 641–655 (2001)
5. Baek, J.H., Sunwoo, M.H.: New Degree Computationless Modified Euclid Algorithm and Architecture for Reed-Solomon Decoder. IEEE Transaction on VLSI systems 14(8), 915–920 (2006)
6. Chaudhuri, P.P., Roy Chowdhury, D., Nandi, S., Chattopadhyay, S.: Additive Cellular Automata: Theory and Applications, vol. 1. IEEE Computer Socity press, Los Alamitos (1997)
7. Rao, T.R.N., Fujiwara, E.: Error-Control Coding for Computer Systems. Prentice-Hall, Englewood Cliffs (1989)
8. Lin, S., Costello, D.J.: Error Control Coding: Fundamentals and Applications. Prentice-Hall, Inc., Englewood Cliffs (1983)