Error Correction codes Derived from Orthogonal Latin Square codes

Ande Bhargav, Y.Varthamanan

Abstract— The developments in IC technology and rapid increase of transistor densities and scaling factor, the use of ECC’s acquired prominence. Multiple bit errors in memories due to technology scaling demands advanced error correction codes. SEC-DEC, DEC, burst error detection, Golay code, Reed Solomon codes etc. have much decoding complexity and latency. The above drawbacks can be reduced with OLS codes. OLS codes with majority logic decoding technique, modular construction and simple decoding mechanisms it enables low delay improvements. MBU’S can be addressed using OLS-MLD codes. This paper presents a detail study of developments in multibit ECC’s using OLS-MLD mechanism.

Keywords: -- component; Error correction codes; Single error correction – double error detection; orthogonal latin squares; majority logic decodable.

I. INTRODUCTION

Communication systems of deep space exploration are one of the impressive applications of ECC’s. On the other hand ECC’s are extensively used in storage device systems. A typical audio CD can accept one and a half million bits to represent just one second of music. At the same instant 6 billion or so bits are added to protect the recorded music form scratches, dust… etc. Due to scaling of technology multiple bit upsets came into existence as the floor planning and routing became more complex. When technology scaled down from 180nm to 22nm, the ratio of multiple bit upsets (MBU) to single event upset (SEU) increases from 0.5% to 3.9% [11]. Soft errors lead to loss of information, which ultimately effects the function of the devices, especially in avionics and space applications. Internal radiation can also cause soft errors (bit change).

Early developments like SEC’s, DEC’s, SEC-DED, are unable to address the multiple bit upsets due to their decoding complexity. Memories like SRAM is not exceptional form MBU’s. Decoding and encoding has to be done in parallel with SRAM memories for fast access of data. Developments like Hamming code, Bose – chaudhuri – Hocquenghem (BCH) codes, and Reed Solmon codes are more complex in parallel decoding [3][14]. Due to delay overheads and high area requirements the codes are limited to smaller applications. And most of the ECC’s have serial decoding circuitry. An alternative for parallel decoding limitation is one step majority logic decodable (OS-MLD) codes such as Euclidean geometry (EG) codes, Difference set (DS) codes [13]. But both the EG, DS codes have limited number of error correction capabilities and data size [8]. In addition to that, available data bits are not in the powers of two [3]. So OS-MLD is not suitable for SRAM memories as the data size is generally in the powers of two like 16, 32. This limitation has opened a new class of codes i.e. orthogonal latin square (OLS) codes. OLS codes are derived from latin squares [1]. OLS codes are also a type of OS-MLD codes with wide range of word sizes and error correction capabilities. However it requires more number of parity bits compared to other ECC’s. With increase in number of error correction bits, parity bits increase rapidly. Research is being done to reduce the parity bits and a method has been proposed recently [6]. In this paper different advancements in OLS-MLD codes are present.

The rest of the paper is organized as follows. Introduction and basic implementation of OLS-MLD code is included in Section II. Recent developments of OLS-MLD codes are discussed in Section III. Section IV deals with a comparative study of different parameters. Conclusion and References are included in Section V.

II. ORTHOGONAL LATIN SQUARE CODES

Before A Latin Square is a matrix structured grid with the numbers ‘1 to n’ occurs only once in each row and column. For example: Latin Square of order 4.

\[
\begin{array}{cccc}
2 & 3 & 1 & 4 \\
4 & 1 & 3 & 2 \\
1 & 4 & 2 & 3 \\
3 & 2 & 4 & 1 \\
\end{array}
\]

Two Latin Squares are said to be mutually orthogonal when two squares are superimposed, every order of co-ordinates should occur exactly once. The concept of mutually orthogonal latin squares (MOLS) can extend to a set of more than two squares, provided that every square must be mutually orthogonal to every other one.

\[
\begin{array}{cccc}
1 & 2 & 3 & 1 \\
2 & 3 & 1 & 3 \\
3 & 1 & 2 & 3 \\
\end{array}
\]

Revised Version Manuscript Received on August 19, 2019.

Ande Bhargav, Research Scholar, Dept. of Electronics and Communication Engineering, Sathyabama Institute of Science and Technology, Chennai, Tamil Nadu, India

Dr. Y. Varthamanan, Assoc.Prof, Dept. of Electronics and Communication Engineering, Sathyabama Institute of Science and Technology, Chennai, Tamil Nadu, India

Published By:
Blue Eyes Intelligence Engineering & Sciences Publication

DOI:10.35940/ijrte.B1205.0882S819

1948
Error Correction codes Derived from Orthogonal Latin Square codes

The superimposing of two latin squares is shown below:
Mutually OLS condition:  
\[
\begin{array}{ccc}
11 & 22 & 33 \\
23 & 31 & 12 \\
32 & 13 & 21 \\
\end{array}
\]
OLS codes are constructed using these mutually orthogonal latin squares. Primary considerations for OLS codes are as follows:
Order of the matrix = m
Number of error correcting bits = t
Number of data bits that a code can protect = \(k = m^2\)
Number of check bits / parity bits = \(2tm\)

In encoding technique each bit involves \(2^t\) parity bit computations and each other bit participates in at most one of those parity bits. This creates a simple correction when the number of bits in error is \(t\) or less. Decoding starts with recomputing the parity bits. The recomputed check parity bits are fed to one step majority logic and a majority vote is taken. If the majority value is one, it results in bit error and correction is needed else the decoded bit is true. If \(t+1\) bits need to be corrected then simply \(2m\) parity bits are added to the code.

Modeling a parity check matrix \(H\):
Parity check matrix is modeled depends on error correction capabilities of the code. For single error correction (SEC) using OLSC. Consider an \(H\) matrix for \(m = 4; k = 16\) data bits, \(t=1\) error correcting capability, \(2tm = 8\) parity bits.

\[
H = \begin{bmatrix} M_1 & I_{2m} \\
M_2 & \end{bmatrix}
\]

Where \(I_{2m}\) is identity matrix of size \(2m\). \(M_1, M_2\) are matrices of size \(m\times m^2\). The matrix \(M_1\) has \(m\) ones in each row. For the \(r^{th}\) row the one are at positions: \((r-1) \times m + 1, (r-1) \times m+2, \ldots (r-1) \times m+m-1, (r-1) \times m + m\)
The matrix \(M_2\) is constructed as shown below:

\[
M_2 = \begin{bmatrix} I_m & I_m & \ldots & I_m \end{bmatrix}
\]

The encoding matrix \(G\) is just the \(H\) matrix on which the check bits are removed.

\[
G = \begin{bmatrix} M_1 \\
M_2 \end{bmatrix}
\]

In brief, the encoder with \(k=m^2\) data bits will generate \(2tm\) check bits using generator matrix \(G\), derived from OLS's.

\[
M = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

Fig. 1. Parity check matrix for OLS (24, 16) having \(k=16\) and \(t=1\)

Generation of parity bits:

Each row of the matrix \(M\) should have only \(m\) number of one’s. As shown in the Fig. 1. OLS (24, 16) parity bits are calculated by performing an XOR operation of data bits present in each row of generator matrix. \(P_0\) is calculated by XOR of first 4 data bits \((d_0, d_1, d_2, d_3)\) of row one in \(M_1\). \(P_1\) is calculated by XOR operation between \(d_4, d_5, d_6, d_7\) of 2\(^{nd}\) row in \(M_1\). Likewise all other parity bits are obtained with XOR operation of data bits in respective rows of parity check matrix. Total 8 parity bits will be obtained as \(2m = 8\) in OLS (24, 16) code. In the decoding process: each group of data bits involved in encoding mechanism will be verified against their respective parity bits to generate syndrome bits \(S_i\). \(S_0\) is an XOR operation of data bits along with its parity bit \(d_0\). Thus obtained syndrome bits are fed to majority logic decodable gate for error detection and correction.

Fig. 2. Parity check matrix for OLSC (32, 16) having \(k=16\) and \(t=2\)

Fig. 3. Encoder for OLS code \(k=16\) and \(t=2\)
The above figure shows that three XOR gates are required to generate one parity bit. Encoder of OLS (32, 16) requires 48 two- input XOR gates [2]. As number of error correction bits increases, parity bits increases, that results in increase in XOR gates. Developments are going on to reduce the number of parity bits and logic gates which will be discussed later. The basic decoder along with correction logic of OLS code computes the syndrome \(S_i\) as shown in Fig.4.
Fig. 4. Decoder for OLS code $k=16$ and $t=2$

Syndrome bits thus calculated are connected to majority logic and correction module. To detect / correct a data bit $d_i$, syndrome bits having the same $d_i$ as one of its input, are given to the OS-MLD gate [1] [6]. Inputs of OS-MLD correction logic for data bit $d_0$ are shown in Fig.5. Encoder requires $2tm(m-1)$ two input XOR gates and syndrome computation requires $2tm^2$ two input XOR gates [9].

Fig. 5. OLS-OS-MLD gate with correction logic

OLS codes can be implemented for different values of $m$, $t$, and a few are listed in the Table-1. It is clear that with increase in correctable errors, parity bits needed to encode the data also increase in the same phase. With increase in parity bits, parameters like area, power and delay may get effected.

### Table 1. Orthogonal Latin square codes

<table>
<thead>
<tr>
<th>Latin square size (m)</th>
<th>No. of errors (t)</th>
<th>Data bits (m²)</th>
<th>Parity bits (2tm)</th>
<th>Total bits (m² + 2tm)</th>
<th>No. of XOR gates (encoder)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2</td>
<td>9</td>
<td>12</td>
<td>21</td>
<td>36</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>16</td>
<td>8</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>16</td>
<td>16</td>
<td>32</td>
<td>48</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>25</td>
<td>20</td>
<td>45</td>
<td>60</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>64</td>
<td>32</td>
<td>96</td>
<td>96</td>
</tr>
</tbody>
</table>

Fig. 6. Parity check matrix for OLSC (36, 20) for $m=4$ and $t=2$

### III. ADVANCEMENTS IN OLS CODES

An extended method in OLS codes have been proposed to protect the large number of data blocks without changing parity bits. For example, OLS code with $m=4$, $t=2$ will have we have $k=16$ data bits. For same $m=4$, $t=2$, $k=20$ data bits have been proposed in [6]. In both the cases parity bits $2tm=16$ are same, but the number of data bits increased by 4. Extended parity check matrix with same of parity bits is shown in Fig. 6. Design of encoder and decoder will be same but the input of encoder will increase by one number.

### Table 2. Extended DEC-OLS Parameters

<table>
<thead>
<tr>
<th>$K_{OLS}$</th>
<th>$K_{Extended}$</th>
<th>n-k</th>
<th>m</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>20</td>
<td>16</td>
<td>4</td>
</tr>
<tr>
<td>64</td>
<td>72</td>
<td>32</td>
<td>8</td>
</tr>
<tr>
<td>256</td>
<td>336</td>
<td>64</td>
<td>16</td>
</tr>
</tbody>
</table>

OLS codes are being studied to protect memories against MCUs. For 32 bit data it is not possible to implement OLS codes as data bits are defined by $k=m^2$ and 32 is not a perfect square. In [1] it was proposed to reduce the $(77, 49)$ OLS code to $(60, 32)$ OLS code. And extended $(55, 32)$ OLS code was proposed for 32-bit data bits in [7]. Extended OLS codes for 32-bit data block $(54, 32)$, $(53, 32)$ are discussed and decoder was modified accordingly [16].

Reduction of parity bits is the prominent problem to work with OLS codes. A new scheme was developed in this aspect [2]. A self – checking encoder and syndrome generator was designed in [9]. In this method two additional bits are generated with XOR of selected parity bits at encoder and the same was verified at syndrome computation of the receiver for $S$ and $t=1$. Parity prediction in this scheme was proposed as
Error Correction codes Derived from Orthogonal Latin Square codes

\[ r_1 = s_1 \oplus s_2 \oplus s_3 \oplus \ldots \oplus s_{2m} \]

\[ r_2 = c_1 \oplus c_2 \oplus c_3 \oplus \ldots \oplus c_{2m} \]

Syndrome computation \( O_{\text{syn}} = \frac{(4m-2)}{(2m^2)} \)

Overhead of encoder and syndrome for \((24,16)\) OLS code is 29.17% and 43.75 respectively, which are less when compared to other error correction codes [9].

IV. COMPARATIVE STUDY OF ECC’S & RESULTS

This section illustrates the synthesis reports of error correction codes. Parameters such as Area, Delay, and Power are compared for recent developments in ECCs. In Encoder and Decoder designs of OLS codes delay was optimized at a great level when compared to Area and power. Area and Delay were optimized in OLS codes when compared to conventional ECCs. 16 and 32 bit data bits were compared here because the conventional codes are not flexible to implement for larger data block sizes. Encoders and Decoders are included as a part of memory design to correct the data while reading the information from it because memories are not an exception from error.

TABLE.3. Parameters comparison of Conventional Codes

<table>
<thead>
<tr>
<th>Conventional Error Correcting Codes</th>
<th>Area ((\mu m^2))</th>
<th>Delay (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>16 - SEC_DEC_BCH</td>
<td>4257.2</td>
<td>3.38</td>
</tr>
<tr>
<td>DEC_BCH</td>
<td>13606</td>
<td>5.30</td>
</tr>
<tr>
<td>32 - SEC_DEC_BCH</td>
<td>7975.2</td>
<td>3.54</td>
</tr>
<tr>
<td>DEC_BCH</td>
<td>57954.4</td>
<td>6.14</td>
</tr>
</tbody>
</table>

TABLE.4. Parameters comparison of synthesized OLS encoders and decoders (16-bit)

<table>
<thead>
<tr>
<th>Advancements of 16-bit OLS-MLD codes</th>
<th>Area ((\mu m^2))</th>
<th>Delay (ns)</th>
<th>Power (mW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encode ( r )</td>
<td>Decode ( r )</td>
<td>Encode ( r )</td>
<td>Decode ( r )</td>
</tr>
<tr>
<td>OLS (32,16) Area optimized</td>
<td>970.8</td>
<td>4408.8</td>
<td>0.34</td>
</tr>
<tr>
<td>Delay optimized encoder</td>
<td>1016.4</td>
<td>3884.0</td>
<td>0.34</td>
</tr>
<tr>
<td>Delay optimized encoder</td>
<td>1417.6</td>
<td>6674.0</td>
<td>0.36</td>
</tr>
<tr>
<td>Delay optimized encoder</td>
<td>1445.6</td>
<td>6388.8</td>
<td>0.36</td>
</tr>
</tbody>
</table>

TABLE.5. Parameters comparison of synthesized OLS encoders and decoders (32-bit)

<table>
<thead>
<tr>
<th>Advancements of 32-bit OLS-MLD codes</th>
<th>Area ((\mu m^2))</th>
<th>Delay (ns)</th>
<th>Power (mW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encode ( r )</td>
<td>Decode ( r )</td>
<td>Encode ( r )</td>
<td>Decode ( r )</td>
</tr>
<tr>
<td>For ((t=2)) OLS (60,32)</td>
<td>970.8</td>
<td>4408.8</td>
<td>0.34</td>
</tr>
<tr>
<td>Proposed (55,32)</td>
<td>1016.4</td>
<td>3884.0</td>
<td>0.34</td>
</tr>
<tr>
<td>For ((t=3)) OLS (76,32)</td>
<td>1417.6</td>
<td>6674.0</td>
<td>0.36</td>
</tr>
<tr>
<td>Proposed (68,32)</td>
<td>1445.6</td>
<td>6388.8</td>
<td>0.36</td>
</tr>
</tbody>
</table>
CONCLUSION

This paper has presented recent advancements of ECCs. Today technology has scaled below 20nm and the reduced spacing in the logic circuit causes soft errors due to internal radiation. Memories (SRAM) are not an exception from these soft errors. Design of powerful ECCs is the only solution for this problem. Multiple bit upsets (MBUs) are gradually increasing due to scaling of the technology. Earlier codes like Hamming code, BCH code, Golay code, Reed Solomon codes couldn’t address recent problems due to their decoding complexity and limited data block size. Triple adjacent errors, single byte errors… etc. MBUs have increased gradually. This created a necessity to develop more efficient ECCs. In this scenario, OLS codes created a wide range of opportunity to develop error correction codes for MBUs.

OLS codes with their modular construction and Expandable data block size contributed much in development of effective ECCs. Recent developments of OLS-MLD codes are described in this letter. At the cost of parity bits OSLC reduces the delay in decoding. Advancements has shown the possibility to control the parity bits. By reducing the parity bits and with flexibility of data block size OLS codes are efficient to design effective error correcting codes.

REFERENCES