VLSI Design of Systematic Odd-Weight-Column Byte Error Detecting SEC-DED Codes

Luca Penzo^*, Donatella Sciuto^*, Cristina Silvano*
^ Politecnico di Milano, Dipart. di Elettronica, P.za Leonardo da Vinci 32, 20133 Milano, Italy
* Bull HN Information Systems Italia, R&D Lab., 20010 Pregnana M. (Milano), Italy

Abstract

The paper introduces an extension to previous single error correcting (SEC) - double error detecting (DED) - single byte error detecting (SBD) codes. The proposed approach constructs systematic odd-weight-column SEC-DED-SBD codes whose correctible errors also include any odd number of erroneous bits. Main purpose of this paper is to show that the proposed codes are suitable for high performances VLSI implementations in computer applications, using high speed encoding/decoding circuits and parallel data manipulation. Furthermore, the paper will show how such codes can be easily designed from the specifications using a software tool, which generates the VHDL (VHDL Hardware Description Language) description of the circuits.

1. Introduction

In the actual computer applications, the memory sub-systems are commonly organized as b-bit-per-chip or b-bit-per-card. If a failure occurs, the resulting information read out of the memory is likely to have up to b erroneous bits within a byte, where a byte represents the cluster of b data bits that are fed by the same memory chip or card ([4]). In byte organized memory sub-systems, error control codes (ECCs) capable of correcting/detecting bit errors as well as byte errors are suitable to increase reliability and data integrity.

Several codes with byte error control capabilities have been proposed in literature ([5], [6], [7], [8], [9], [10], [11], [13], [14], [15]). Single error correcting - single byte error detecting (SEC-SBD) codes (indicated also as SEC-ShED where b is the byte length) have been introduced in [6], [7], [11]. Errors are corrected if they are single random errors, but the class of detectable errors includes also double random errors (SEC-DED-SBD) in [8] and, for byte lengths b greater than or equal to 5, in [7]. SEC-DED-SBD codes have been proposed also in [15] for b=4, in [13] for b=5, in [11] for b ≥ 7 and in [14] for even byte length. Other approaches ([5], [9], [10]) have presented single byte error correcting - double byte error detecting (SBC-DBD) codes (indicated also as SBC-ED-Ded). However, the construction of optimal SEC-DED-SBD codes or SBC-DBD codes for the general case with the minimum number of check bits is still an open and challenging problem.

The present paper introduces an extension to previous SEC-DED-SBD codes. The proposed techniques construct systematic odd-weight-column SEC-DED-SBD codes in which the class of correctible errors also includes any odd number of erroneous bits per byte. This additional capability increases the level of reliability of a computer memory system with respect to those systems employing the conventional SEC-DED-SBD codes.

However, to support such possibility, an overhead in terms of the number of check bits is required. Table I provides a first comparison among the codes proposed in the rest of this paper and the SEC-DED-SBD codes presented in [7], [8], [13], [15] and the SBC-DBD codes reported in [4], [5], [9], [12] and [17]. Each table entry represents the check bits lengths (r) for some practical values of byte lengths (b) and data bit lengths (k) for the specified class of codes. In particular for SBC-DBD codes each entry is the minimum of check bits lengths required by the referred codes. As shown in Table I, a few additional check bits (at most four) are required to the proposed codes with respect to SEC-DED-SBD codes in order to correct at least 90% of the possible multiple errors per byte. On the other hand such overhead is reasonable if compared with the requirements in terms of check bits of SBC-DBD codes (almost half for b > 8 for the data in Table I).

The proposed codes have been designed to be applied in a multiprocessor computer system to achieve high reliability in the communication between processors and the memory sub-system. In addition to high reliability, the proposed codes are systematic. Thus the information bits are maintained separated from the check bits, so that the encoding/decoding and the data processing can be performed in parallel.

The main purpose of this paper is to demonstrate that the proposed codes are suitable for high performance VLSI implementations in computer applications, using high speed encoding/decoding circuitries and parallel data processing. These characteristics are mainly due to the fact that the codes belong to the class of systematic odd-weight-column codes with a modular structure. The paper will also show how this code can be easily designed from the specifications using a new software tool.

The paper is organized as follows. First Section II briefly describes the new codes construction techniques, then Section III shows the main advantages offered by the codes and how these codes can be implemented in VLSI circuits. Finally the automatic code generation tool is proposed in Section IV, while some application results are given in Section V.

2. New code construction techniques

A general description of the code construction techniques and the error detection/correction capabilities of the codes are
<table>
<thead>
<tr>
<th>k</th>
<th>16</th>
<th>32</th>
<th>64</th>
<th>128</th>
<th>256</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>6</td>
<td>9</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
<td>9</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
</tbody>
</table>

Table I: Comparison among check bit lengths of some class of byte error control codes with data bit lengths \( k = 16, 32, 64, 128 \) and 256.

A: SEC-SDH codes for \( b \leq 3 \) and SEC-DED-SDH codes for \( b \leq 5 \) proposed in [7];
B: SEC-DED-SDH codes proposed in [8] and also in [16] for \( b = 4 \);

\[ A: \text{SEC-SDH codes for } b \leq 3 \text{ and SEC-DED-SDH codes for } b \leq 5 \text{ proposed in [7];} \]
\[ B: \text{SEC-DED-SDH codes proposed in [8] and also in [16] for } b = 4; \]

The parity check matrix \( H \) is in systematic form as in (2.2) with the matrix \( B(r \times k) \) composed of a set of \( K \) matrices as:

\[ B(r \times k) = \begin{bmatrix} B_1 & B_2 & \ldots & B_k \end{bmatrix} \]

where the matrices \( B_i \) are non zero distinct \( (r \times k) \) binary matrices. By construction, the number \( K \) of non zero distinct matrices \( B_i \) is equal to the number of data bytes in the codeword \( (K = k / b) \) where \( k \) is supposed to be a multiple of \( b \), without any loss of generality.

The structure of the generic matrix \( B_i \) is composed of two sub-matrices. The first sub-matrix is a \((b \times b)\) identity matrix \( I_b \), allowing to uniquely identify the single bit within the byte.

The second sub-matrix of \( B_i \) is a \((r \times b)\) matrix, indicated as \( H_b \), having the property that its column vectors are equal to each other and the generic column is stated to be a nonzero \((r \times b)\)-tuple of even weight over GF(2). A set of nonzero distinct matrices \( H_b \) can be obtained defining its generic column vector as one of all the possible distinct nonzero \((r \times b)\)-tuples of even weight over GF(2). This part of \( B_i \) allows to uniquely identify the single byte.

The two sub-matrices composing \( B_i \), indicated as \( I_b \) and \( H_b \), have to be placed vertically in \( B_i \) following three different techniques depending on \((r \times b)\) being equal, greater or less than \( b \). In all cases, the code is an odd-weight-column code (\( [1] \)), in fact the column vectors corresponding to the check bits are one-weight columns and the column vectors corresponding to the data bits are odd-weight columns, being the sum of the generic even-weight column of \( H_b \) plus the generic one-weight column of \( I_b \).

Furthermore the code requires a number of check bits \( r \geq b + 2 \), being the number of rows of \( B \) \((r)\) the sum of the number of rows of \( I_b \) \((b)\) and the number of rows of \( H_b \) \((at least 2)\). Globally codes with the given properties for arbitrary code length \((n)\) and byte length \((b)\) within the range \( r \geq b + 2 \) and \( b > 2 \) can be constructed using the three proposed techniques. In particular the first technique (called \( C_1 \)) requires \( r = 2b \), the second technique \( (C_2) \) requires \( r > 2b \) and the third technique \( (C_3) \) requires \( b + 2 \leq r < 2b \).

The maximum number of data bytes in the codeword \( (K) \) as a function of \( (r, b) \) is reported hereafter for the three constructions and it is fully derived in [16]:

\[ K(r, b) = 2^b - 2 \quad \text{for } C_1 \]
\[ K(r, b) = 2^{r-b-1} + 2^{r-b-2} - 2 \quad \text{for } C_2 \]
\[ K(r, b) = 2^{r-b-1} - 1 \quad \text{for } C_3 \]

From these values, the maximum length of the proposed codes can be easily derived as: \( n(r, b) = bK(r, b) + r \).
a detailed description of the other two construction techniques and the related theorems are reported in [16]. However, the codes properties, in terms of error correction and detection, as reported in Theorem C1, are the same also for the codes obtained by C2 and C3.

**Construction C1**

Assume \( r = 2b \), let the matrix \( B_{1}(2 \times k_{0}) \) in (2.3) be defined as:

\[
B_{1}(2 \times k_{0}) = \begin{bmatrix}
B_{2} & B_{3} & \ldots & B_{K_{0}} & B_{1} & B_{2} & \ldots & B_{1} & B_{K_{0}}
\end{bmatrix}
\]

(2.7)

where:

\[
B_{1} = \begin{bmatrix}
H_{1} \\
1_{b}
\end{bmatrix}
\text{for } i \in \{1, 2, \ldots, K_{0}\}
\]

(2.8)

and \( 1_{b} \) denotes the \((b \times b)\) identity matrix, \( H_{1} \) is the \((b \times b)\) matrix described above and \( K = 2K_{0} \).

Let \( b_{i} \) be the generic b-bit column vector of \( H_{1} \) and \( S_{b_{i}} \) the (n-1)-dimensional subspace of the finite field \( GF(2)^{b} \) composed of the \( 2^{b} \) distinct b-tuples of even weight over \( GF(2) \). It is possible to define a set of \((2^{b+1}-1)\) vectors \( c_{i} \) and consequently a set of \((2^{b+1}-1)\) matrices \( H_{i} \) corresponding to the \((2^{b+1}-1)\) distinct nonzero b-tuples of \( S_{b_{i}} \). Hence the function \( K(b, r) \) is represented by equation (2.4).

**Theorem C1**

For \( r = 2b \) the codes constructed by C1 are systematic odd-weight-column SEC-DED-SBD codes capable of correcting any odd number of erroneous bits per byte and \( K(b, r) \) is given by equation (2.4).

For example, Construction C1 can be applied to \( b = 4 \) to yield a systematic \((64, 56)\) SEC-DED-SBD triple-bit-per-byte code represented by the following matrix:

\[
H_{(64 \times 56)} = \begin{bmatrix}
H_{1} & H_{2} & H_{3} & H_{4} & H_{5} & H_{6} & H_{7} & I_{8} & I_{9} & I_{10} & I_{11} & I_{12} & I_{13} & I_{14} & 0_{15}
\end{bmatrix}
\]

where:

\[
\begin{bmatrix}
H_{1} & H_{2} & H_{3} & H_{4} & H_{5} & H_{6} & H_{7} & I_{8} & I_{9} & I_{10} & I_{11} & I_{12} & I_{13} & I_{14} & 0_{15}
\end{bmatrix}
\begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{bmatrix}
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]

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

\[
\begin{bmatrix}
H_{1} & H_{2} & H_{3} & H_{4} & H_{5} & H_{6} & H_{7} & I_{8} & I_{9} & I_{10} & I_{11} & I_{12} & I_{13} & I_{14} & 0_{15}
\end{bmatrix}
\]

(3.1)

3. **VLSI implementation**

In this Section the main characteristics of the codes derived in the previous Section are examined to outline their advantages from the VLSI implementation point of view. High-speed encoding/decoding circuits are proposed too.

The implementation advantages offered by the codes are mainly related to the fact that the codes are systematic with odd-weight-columns and modular structure. Being systematic, the information bits remain unchanged and separated from the check bits, therefore the codes offer the advantages that the encoding/decoding and the data processing can be performed in parallel, avoiding propagation delay on the data path.

The overall complexity of the parity check circuits required by the given codes can be roughly estimated by examining the structure of the parity check matrix \( H \). Basically, the global number of 1's in \( H \) determines the complexity of the hardware required: a lower number of 1's requires a less complex circuit ([12]). In particular, in systematic codes the total number of 1's in the i-th row of \( H \) is related to the number of logic levels necessary to generate the corresponding check bit \( (C_{i}) \) or syndrome \( (S_{i}) \), as described in [1]. Therefore to obtain the fastest generation of check and syndrome bits, all \( t_{i} \) for \( i = 1, 2, \ldots, r \) should be minimum and equal, or as close as possible, to the average number given by the total number of 1's in \( H \) divided by the number of rows \( (r) \) ([11]).

In the construction of the proposed \( H \) matrix, for a given number of check bits, the odd-weight-column vectors having an odd number of 1's in the first b bits and all 0's in the last b bits or vice versa cannot be used, to maintain the property of correcting any odd number of erroneous bits per byte also for the check bytes.

Thus, given \( r \), the maximum number of data bits of Hsiao codes ([11]) is greater than the maximum number of data bits of the proposed codes. But, when it is necessary to control a number of data bits lower than the maximum number of data bits, some other codes can be simply derived from the global H matrix by discarding some selected sub-matrices \( B_{i} \). The choice of which one of these sub-matrices have to be discarded follows the criteria of having all \( t_{i} \) minimum and equal. In this way it is possible to reach the values of \( t_{i} \) equal to the minimum number as given by Hsiao codes.

Given \( b = 4 \) and \( r = 8 \), the above \( H_{(8 \times 64)} \) matrix controls the maximum number of data bits (56). From this code, other codes, having the same error control capabilities but controlling a number of data bits lower than 56, can be simply derived by discarding some of the sub-matrices \( B_{i} \) \((i \neq 4)\). Hence, a set of matrices \( H_{(8 \times 40)} \) can be derived from \( H_{(8 \times 64)} \), having 32 data bits and a number of 1's in the rows of \( H \) equal to 13, that also corresponds to the minimum number given by optimal minimum odd-weight-column SEC-DED codes ([11]) for the same \( r \) and \( k \). One of these matrices is the following matrix:

\[
H_{(8 \times 40)} = \begin{bmatrix}
H_{1} & H_{2} & H_{3} & H_{4} & H_{1} & I_{4} & I_{4} & I_{4} & I_{4} & I_{4} & I_{4} & I_{4} & I_{4} & I_{4}
\end{bmatrix}
\]

that has already been reported in [4], as an example of SEC-DED-SBD code for \( b = 4 \) and \( k = 32 \). However this code was derived in [4] from a SEC-DED code through the application of the column reconfiguration method to detect also single byte errors.

Another class of codes, suitable for VLSI implementations, is the class of n-modularized codes, whose encoding/decoding circuits can be partitioned into n identical modules ([19]). For example, for a 2-modularized code, the \( H_{(r \times k_{0})} \) matrix can be divided into two parts, called Module 0 (\( M_{0} \)) and Module 1 (\( M_{1} \)); the j-th row of \( M_{0} \) (\( j = 1, 2, \ldots, r \)) is equal to the j-th row of \( M_{1} \) (\( j = 1, 2, \ldots, r \)), where i and j can be equal or different (obviously in order to distinguish between the two modules there must exist at least one value of i which differs from j).

The \( n \) modules have the property that the same logic block can be applied for all the modules, resulting in a great flexibility and simplicity into the logical and physical design phases of its VLSI implementation.

The codes defined by Construction C2 are 2-modularized codes, in fact the H matrix, including the check bytes, can be partitioned into Module 0 (\( M_{0} \)) and Module 1 (\( M_{1} \)) respectively, observing that, being the H matrix expressed as:

\[
H = \begin{bmatrix}
H_{1} & \ldots & H_{r} & I_{0} & H_{1} & \ldots & I_{0} & I_{0} & I_{0} & I_{0} & 0_{0}
\end{bmatrix}
\begin{bmatrix}
I_{0} & I_{0} & I_{0} & I_{0} & I_{0} & H_{1} & \ldots & H_{r} & I_{0} & I_{0} & I_{0}
\end{bmatrix}
\]

158
where the \( I_b \) denotes the \((b \times b)\) identity matrix and the matrices \( H_b \) have been defined in \( C_1 \). It can be considered as composed of two modules, as follows:

\[
H = \begin{bmatrix}
H_1 & H_2 & \ldots & H_{K_0} & I_b & \ldots & I_b \\
I_b & I_b & \ldots & I_b & H_1 & \ldots & H_{K_0}
\end{bmatrix}
\]

keeping in mind that the \((K_0 + 1)\)-th byte and the \((2K_0 + 1)\)-th byte are the check bytes. The first \( b \) rows of \( M_0 \) are equal to the last \( b \) rows of \( M_1 \) and vice versa, thus the \( H \) matrix defined by \( C_1 \) has a 2-modularized organization.

For example the above \( H_{M \times 64} \) matrix can be seen as composed of two modules \( M_0 \) and \( M_1 \):

\[
H_M = \begin{bmatrix}
H_1 & H_2 & \ldots & H_{K_0} & I_b & \ldots & I_b \\
I_b & I_b & \ldots & I_b & H_1 & \ldots & H_{K_0}
\end{bmatrix}
\]

The proposed high-speed parallel encoding-decoding circuits consist of four main blocks: check bit generator, syndrome generator, syndrome decoder and error corrector. The check bit generator and syndrome generator blocks are constituted by trees of Exclusive-OR gates. The syndrome decoder is constituted by two main blocks. The first block, called SYNDEC, decodes the syndromes to generate the correction patterns for the single-bit errors and the odd-bit-per-byte errors. The second block, called SYNCT, decodes the syndromes for the detection of double-bit errors and even-bit-per-byte errors.

Due to the 2-modularized structure of the \( H \) matrix, the SYNDEC block can be described instancing the same logic block twice. The first instance \( (l_0) \) receives the \( r \) syndrome bits as input and outputs the Bit Error Pointers related to \( M_0 \), while the second instance \( (l_1) \) receives the \( r \) syndrome bits as input and outputs the Bit Error Pointers related to \( M_1 \). The Bit Error Pointers are used by the error corrector block, that simply executes a bit per bit Exclusive-OR between each Bit Error Pointer and the related bit read out of the memory.

The \( r \) syndromes received by \( l_0 \) are used to decode data and check byte errors (Byte Error Pointers), the single-bit errors (Single-bit Error Pointers) and the odd-bit-per-byte errors (Odd-bit-per-byte Error Pointers). Receiving the related pointers as inputs, the same logic structure, called BITDEC, is used to compute the Bit Error Pointers for each data or check bits.

Fig. 1 shows an example of the SYNDEC block for the above \( H_M \) matrix. The gate amount of the proposed code represents about six percent decrease compared to a conventional decoding logic of a \((64, 56)\) minimum odd-weight-column SEC-DED code able to correct just single errors. The increment required by the proposed code, in terms of propagation delay, is just one gate level, using a standard VLSI cell library.

In addition to the SYNDEC block, the SYNCT block receives as input the syndromes and recognizes the number of asserted bits in the syndromes. An example of the logic structure of the SYNCT block having four syndromes as input is in Fig. 2.

4. Automatic generation of VHDL description

The design of a code with the characteristics described in Section II can be automated using a software tool, called GeCo (Generator of Codes). GeCo is a more general system which generates ECCs, in fact not only it can generate the class of codes described in this paper, but the main classes of codes in literature as well. Furthermore, its modularity allows an easy introduction of new classes of codes.

The aim behind the use of the GeCo tool is to provide the system designer with an easy-to-use platform to develop rapidly ECCs from specifications and to evaluate the intermediate results of the interaction between the user and the system. In fact, chosen the desired class of codes, the user can define the code specifications in terms of requested parameters: number of data bits \( (k) \), number of check bits \( (t) \) and, eventually, byte length \( (b) \). During the interactive dialogue, the user can vary a parameter and GeCo automatically re-computes the values of

---

Fig. 1: The SYNDEC logic block for the \( H_M \) matrix.
the other parameters but, depending on the constraints existing among them, it prevents him from an illegal parameter assignment proposing only acceptable variations. After the dialogue led to an acceptable parameters configuration, the parity check matrix $H$ can be automatically generated along with a hierarchical description of the main logic blocks implementing the code using the VHDL description language. GeCo computes also the global gate amount and the estimated propagation delay.

GeCo has been developed in the Microsoft Windows framework using the object oriented approach and the C++ language. It is composed of four main modules: the User Interface, the Controller, the Generator and the Translator.

5. Application results and concluding remarks

The proposed techniques allow the generation of a class of codes which extends the protection provided by previous SEC-DED-SBD codes by constructing systematic odd-weight-column SEC-DED-SBD codes in which the class of correctable errors is enlarged to include any odd number of erroneous bits per byte. The proposed techniques are suitable for high performance computer applications and they are supported by an automatic tool.

This tool has been used to design a 64 data bit error control code inserted in an ASIC developed by Bull in the R&D Labs of Pregnana M. This ASIC interfaces the main memory, four data channels to processors and the I/O channel and it has been completely described using the VHDL language. Its main characteristics are reported in Fig. 3.

References


![Fig. 2: The SYNCNT logic block with four syndromes as inputs.](image)

![Fig. 3: Main characteristics of the ASIC](image)