

Automatic Generation of Error Control Codes for Computer Applications

Franco Fummi, Donatella Sciuto, and Cristina Silvano

Abstract—This paper proposes a methodology, implemented in a tool, to automatically generate the main classes of error control codes (ECC's) widely applied in computer memory systems to increase reliability and data integrity. New code construction techniques extending the features of previous single error correcting (SEC)–double error detecting (DED)–single byte error detecting (SBD) codes have been integrated in the tool. The proposed techniques construct systematic odd-weight-column SEC–DED–SBD codes with odd-bit-per-byte error correcting (OBC) capabilities to enhance reliability in high speed memory systems organized as multiple-bit-per-chip or card.

Index Terms—Computer memory systems, error control codes, memory reliability, VLSI architectures.

I. INTRODUCTION

TO meet the increasing requirements of system reliability and data integrity, error control codes (ECC's) have been widely exploited in the design of computer memory subsystems [1]. This work aims at providing a methodology, implemented in a modular and efficient tool, to automatically design ECC's and the corresponding encoding/decoding logic from the code specifications. The proposed tool, called GECO (GEnerator of COdes), integrates the main classes of codes suitable for high performance computer applications, as identified in literature. However, its modularity allows an easy introduction of new classes of codes. In particular, the tool automatically generates the parity check matrix, the VHDL description of the encoder/decoder and the related performances.

Several codes with byte error control capabilities have been proposed in literature [1]. Single error correcting–single byte error detecting (SEC–SBD) codes (indicated also as SEC–SbED where b is the byte length) have been introduced in [2], [3]. Errors are corrected if they are single random errors, but the class of detectable errors includes also double random errors (SEC–DED–SBD) in [4] and, for byte lengths $b \geq 5$ in [2]. SEC–DED–SEC codes have been proposed also in [5] for $b = 4$, in [3] for $b \geq 7$, while other techniques are reported in [1]. The approaches [6], [7] have presented single byte error correcting–double byte error detecting (SBC–DBD) codes (indicated also as SbEC–DbED). However, the construction of optimal SEC–DED–SEC or SBC–DBD codes for the general case with the minimum number of check bits is still an open and challenging problem.

To increase the reliability level of a computer memory system with respect to those systems employing conventional SEC–DED–SEC codes, the authors proposed new code

construction techniques [8] providing systematic odd-weight-column SEC–DED–SEC codes in which the class of correctable errors also includes any odd weight error pattern in a single byte by adding redundancy. A few additional check bits (at most four) are required by the proposed codes with respect to SEC–DED–SEC codes in order to extend the protection including the correction of at least 50% of the possible multiple errors per byte. However such an overhead is reasonable with respect to the redundancy of SBC–DBD codes: the proposed codes require almost half of the redundant bits with respect to those required by SBC–DBD codes for $b > 8$.

This paper extends the works presented in [9] and in [10], and shows also that the proposed codes are suitable for high performance VLSI implementations in computer applications, by using high speed encoding/decoding circuits and parallel data processing. The paper is organized as follows. Section II proposes an overview of the methodology and the derived architecture of the software advisor, while the main strategies applied to implement the code classes considered in GECO are examined in Section III. The same section briefly describes the new codes construction techniques (a formal description of these techniques can be found in [8]), while Section IV shows the main advantages offered by these codes and how these codes can be implemented as VLSI circuits. Finally, some application results are given in Section V.

II. AUTOMATIC INSERTION OF ECC'S INTO VLSI'S

The aim behind the use of the proposed tool GECO is to provide system designers with an easy-to-use software advisor to increase the design time during the development of ECC's and the corresponding logic. The user-friendly interactive interface allows the user to choose among the code classes and code parameters, mainly the number of data bits (k), the number of check bits (r) and the byte length (b). GECO has been developed by using the object oriented methodology and the C++ language. It is composed of the following main modules: the user interface, the controller, the generator, the separator, the VHDL translator, and the internal functions manager, such as the Galois field manager.

The controller is a virtual class from which as many classes as the corresponding classes of codes have been derived. Each controller class contains the knowledge on the parameter relations and possible code construction techniques. When the user interface receives a request from the user to change the value of a code parameter, the Interface addresses the request to the controller. First, the controller verifies if the request is in the allowed parameter range, then the relations among the code parameters are recomputed. When the parameter values satisfy the user, the controller calls the generator to activate the corresponding optimal construction algorithm.

Manuscript received April 26, 1995; revised September 30, 1997.

F. Fummi and D. Sciuto are with the Department of Electronics and Information, Politecnico di Milano, Milano I-20133 Italy.

C. Silvano is with the Università di Brescia, Brescia I-25123 Italy.

Publisher Item Identifier S 1063-8210(98)06001-6.

The generator module consists of a set of functions implementing the code construction algorithms for every code class and parameter configuration. Up to now, the five classes of codes described in Section III are included in the generator. Some algorithms use specialized objects to implement different algebras such as the Galois fields algebra.

The finite fields of 2^m elements ($m > 1$), called Galois fields $GF(2^m)$, play a primary role in algebraic coding theory [1], [11], thus a dedicated object has been developed to manage the $GF(2^m)$, based on both the element representations: the power representation and the polynomial representation. Any function included in GECO can interact with the $GF(2^m)$ manager, asking for elements of $GF(2^m)$ and functions representing possible operations among the elements.

The separator algorithm transforms a nonsystematic code into a systematic or at least a separable code. This operation is always possible [2], thus all code construction schemes generating nonsystematic codes, call, as last operation, the separator to derive the final parity check matrix in systematic or, at least, separable form. In both cases, the information bits are maintained separated from the check bits, so that the encoding/decoding and the data processing can be performed in parallel and all information bits read out of the memory appear unchanged.

As basic definition, a binary linear block code C can be described as the null space of a binary vector space generated by the row vectors of an $(r \times n)$ matrix called parity check matrix and indicated as $H_{(r \times n)}$, with n being the length of the codeword composed of k data bits and r check bits. An n -bit row vector X is a codeword in C if and only if $HX^T = 0$, where X^T denotes the transpose of X . When the H matrix is expressed as $H = [BI_r]$, being B an $(r \times k)$ matrix and I_r the $(r \times r)$ identity matrix, then the code is called systematic. Given r , the separator algorithm searches for a set of r columns of $H_{(r \times n)}$ in order to find a submatrix of H , called $A_{(r \times r)}$ that is nonsingular, thus invertible. Once the submatrix A is obtained, the inverse matrix A^{-1} is computed and finally the $A^{-1}H$ product is performed to obtain the parity check matrix in separable/systematic form.

The algorithm can require a large amount of computation time for high values of n or r : in fact for a $H_{(r \times n)}$ it can require, in the worst case, the computation of the rank of $\binom{n}{r}$ matrices of size $(r \times r)$. Thus, for some classes of codes some heuristic methods have been identified and implemented, such as in the case of SEC-DED-SEC codes proposed by Reddy in [2].

Finally, the VHDL Translator reads the code characteristics and the parity check matrix and generates the VHDL hierarchical description of the basic blocks implementing the code: check bit generator, syndrome generator, syndrome decoder, and error corrector. Two architecture bodies are defined for each of these entities, to have both structural and behavioral descriptions.

III. CODE CONSTRUCTION TECHNIQUES

The main features of the five classes of codes considered in GECO are examined in this section, to outline their application

advantages and the implementation strategies adopted. In general, 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 [1]. In particular, in systematic codes the total number t_i 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 [12].

Assuming the use of a v -inputs module 2 adder, the number of logic levels required to generate C_i and S_i are, respectively, given by $l_{C_i} = \lceil \log_v(t_i - 1) \rceil$ and $l_{S_i} = \lceil \log_v t_i \rceil$, where $\lceil x \rceil$ indicates the smallest integer greater than, or equal to, x . Therefore, to obtain the fastest generation of check and syndrome bits, all t_i for $i = 1, 2, \dots, r$ should be minimum and equal, or as close as possible, to the average number given by the total number of 1s in H divided by the number of rows (r). A class of codes satisfying such criteria is called a minimum-equal-weight code [1]. Finally the n -modularized codes, whose encoding/decoding logic can be organized as n identical modules [7], have been preferred for their implementation advantages.

Among SEC-DED encodings, the Hsiao codes [12], representing the optimal minimum odd-weight-column SEC-DED codes, have been implemented.

The SEC-DED-SEC codes are useful to maintain data integrity in byte-organized memories, where the probability of byte errors is high. Both Reddy codes [2] and Chen codes [4] can be generated by GECO. Being these codes nonsystematic, the Separator is called to convert them into a separable form. For a given set of values (k, r, b) , the construction technique of $H_{(r \times n)}$ for Reddy codes is based on the composition of as many submatrices $H_{i_{(r \times b)}}$ as the number of bytes in the codeword. Each submatrix H_i is composed of two submatrices following two techniques, for b even or odd, respectively, as illustrated in [2]. Chen codes can be constructed following several techniques as described in [4]. In particular, the first two techniques can be applied to a predefined SEC-DED-SEC code to obtain a new code with the same properties, but with $(r + 1)$ redundant bits and $(b + 1)$ bits per byte, respectively.

The new class of SEC-DED-SEC-OBC codes described in [8] has been included in GECO, since it extends the protection provided by SEC-DED-SEC codes by adding few redundant bits and offering implementation advantages, as shown in Section IV. As in the Reddy codes, the construction technique of $H_{(r \times n)}$ is based on the composition of as many submatrices $H_{i_{(r \times b)}}$ as the number of bytes in the codeword. Depending on the values of (r, b) , three techniques have been defined to get the H_i as in [8].

In particular, the first technique (C_1) requires $r = 2b$, the second technique (C_2) requires $r > 2b$ and the third technique (C_3) requires $b + 2 \leq r < 2b$. In C_1 , the matrix $B_{(r \times k)}$ is defined as

$$B_{(r \times k)} = [B_1 B_2 \cdots B_i \cdots B_{K_0} B'_1 B'_2 \cdots B'_i \cdots B'_{K_0}]$$

$$B_i = \begin{bmatrix} H_i \\ I_b \end{bmatrix}, \quad \text{for } i \in [1, 2, \dots, K_0]$$

$$B'_i = \begin{bmatrix} I_b \\ H_i \end{bmatrix}, \quad \text{for } i \in [1, 2, \dots, K_0]$$

where I_b denotes the $(b \times b)$ identity matrix, H_i is a $(b \times b)$ matrix having its column vectors equal to each other [the generic column of H_i is a nonzero b -tuple of even weight over $\text{GF}(2)$] and $K = 2K_0$. Let c_i be the generic b -bit column vector of H_i and S_{b-1} the $(b-1)$ -dimensional subspace of the finite field $\text{GF}(2^b)$ composed of the 2^{b-1} distinct b -tuples of even weight over $\text{GF}(2)$, 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-1} . Hence, the function $K(r, b)$ is given by $K(r, b) = 2^b - 2$. For example, the C_1 scheme can be applied to $b = 4$ to yield a systematic $(64, 56)$ SEC-DED-SEC triple-bit-per-byte ECC represented by (1) shown at the bottom of the page

$$H_1 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad H_2 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$H_3 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \quad H_4 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$H_5 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \quad H_6 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$H_7 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}.$$

The single byte error correcting-double byte error detecting (SBC-DBD) codes implemented in GECO are based on the Reed-Solomon techniques [7] to get systematic codes. Extension techniques, defined in [6] and [7], are used to derive nonsystematic codes with greater values of r and n . The construction of Reed-Solomon codes is based on the $\text{GF}(2^b)$, having as elements the power matrices $(b \times b)$ of a nonsingular binary matrix $T(b \times b)$, called "companion matrix" of an irreducible polynomial of degree b , with binary coefficients. A dedicated object, Transformator, has been developed from the primitive polynomial of degree b . The elements of the $\text{GF}(2^b)$ are addressed by an integer i , representing the exponent of the matrix T^i in the field $\Gamma = \{O_b, T^1, T^2, \dots, T^{2^b-2}, T^{2^b-1} = I_b\}$. First, a matrix of integer elements H_i is generated, where each integer i corresponds to an element of the $\text{GF}(2^b)$. Then the Transformator receives as input H_i and b and it returns a binary matrix H , by substituting each integer i of H_i with the

corresponding binary matrix T^i in the field. A list of primitive polynomials [11] with the smallest number of terms (for b in the range from 3 to 24) are stored in a file as a string of bit representing the binary coefficients of each polynomial $g(x)$. From each $g(x)$, the non singular companion matrix $T(b \times b)$ is derived from the binary coefficients of $g(x)$ as follows:

$$T_{(b \times b)} = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 & g_0 \\ 1 & 0 & 0 & \dots & 0 & g_1 \\ 0 & 1 & 0 & \dots & 0 & g_2 \\ 0 & 0 & 1 & \dots & 0 & g_3 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 1 & g_{b-1} \end{bmatrix}.$$

Then, the transformation process requires the computation of the power of the binary matrix $T_{(b \times b)}$ to obtain the field Γ . The final parity check matrix $H_{(r \times b)}$ is as follows:

$$H_{(r \times b)} = \begin{bmatrix} I_b & I_b & \dots & I_b & \dots & I_b & I_b & O_b & O_b \\ I_b & T^1 & \dots & T^i & \dots & T^{2^b-2} & O_b & I_b & O_b \\ I_b & T^2 & \dots & T^{2i} & \dots & T^{2(2^b-2)} & O_b & O_b & I_b \end{bmatrix}.$$

A class of double error correcting-triple error detecting (DEC-TED) codes can be constructed according to the theory of BCH codes [11] that, for a given set of parameter values (k , r , and d_{\min}), can be obtained as cyclic codes from a table containing the corresponding generator polynomial. Then, the generator matrix G can be derived, from which the parity check matrix can be computed by $GH^T = 0$. To simplify the operation to get the H matrix, it is convenient to transform the G matrix in systematic form $G(k \times k) = [I_k P(k \times r)]$. In this case, the H matrix is in systematic form and it is given by $H(r \times n) = [P^T(k \times r) I_r]$.

IV. VLSI IMPLEMENTATION OF THE PROPOSED CODES

Coding for high-performance computer systems requires design techniques aiming at not only high reliability, but also high-speed encoding/decoding and correction circuits and parallel data manipulation to maintain high throughput. The implementation advantages offered by the proposed codes mainly relate to the fact that the codes are systematic and modular. Being systematic, the information bits are separated from the check bits, therefore the codes offer the advantage that the encoding/decoding and the data processing can be performed in parallel.

In n -modularized codes [7], the parity check matrix can be divided into n parts, called modules, composed of the same row vectors, but placed in different positions within the module. The n modules have the property that the same logic block can be applied, resulting in a great flexibility and simplicity during the VLSI implementation. In particular, the codes defined by C_1 are 2-modularized codes, in fact the

$$H_{(8 \times 64)} = \begin{bmatrix} H_1 & H_2 & H_3 & H_4 & H_5 & H_6 & H_7 & I_4 & I_4 & I_4 & I_4 & I_4 & I_4 & I_4 & I_4 & O_4 \\ I_4 & I_4 & I_4 & I_4 & I_4 & I_4 & I_4 & H_1 & H_2 & H_3 & H_4 & H_5 & H_6 & H_7 & O_4 & I_4 \end{bmatrix}. \quad (1)$$

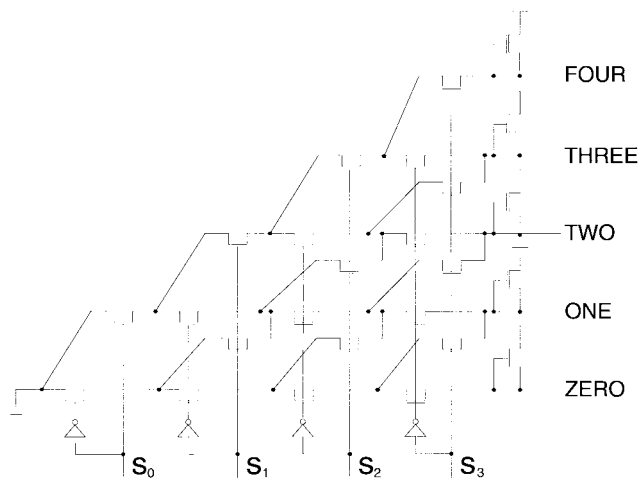


Fig. 2. The SYNCNT logic block with four syndromes as inputs.

by previous SEC-DED-SEC codes. The design of codes is supported by an automatic tool which generates the parity check matrix and the VHDL description of the logic blocks implementing the code.

This tool has been used to design a 64 data bit error control code¹ inserted in an ASIC developed by Bull Information Systems in the R&D Laboratories, Pregnana, Italy, for a shared memory multiprocessor system [13] based on the PowerPC architecture. The device implements the data cross bar architecture among the main memory, four data channels to processors and the I/O channel and it has been completely described using VHDL. The 55 000 equivalent gates ASIC

¹European Patent Application Number 938 330 254.4, 1993. United States Department of Commerce, Patent and Trademark Office, Patent Application Number 08/248140, 1994.

was manufactured using the 0.7- μm DLM CMOS technology supplied by LSI Logic Corporation and packaged into a 299 CPGA. The application specific integrated circuit (ASIC) successfully operated at full speed (75 MHz) at the first run.

REFERENCES

- [1] T. N. Rao and E. Fujiwara, *Error Control Coding for Computer Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1989.
- [2] S. M. Reddy, "A class of linear codes for error control in byte-per-card organized digital systems," *IEEE Trans. Comput.*, vol. C-27, pp. 455-459, May 1978.
- [3] L. A. Dunning and M. R. Varanasi, "Code constructions for error control in byte organized memory systems," *IEEE Trans. Comput.*, vol. C-32, pp. 535-542, June 1983.
- [4] C. L. Chen, "Error-correcting codes with byte error-detection capability," *IEEE Trans. Comput.*, vol. C-32, pp. 615-621, July 1983.
- [5] S. Kaneda, "A class of odd-weight-column SEC-DED-SbED codes for memory system applications," *IEEE Trans. Comput.*, vol. C-33, pp. 737-739, Aug. 1984.
- [6] C. L. Chen, "Error-correcting codes for byte-organized memory systems," *IEEE Trans. Inform. Theory*, vol. IT-32, Mar. 1986.
- [7] S. Kaneda and E. Fujiwara, "Single byte error correcting-double byte error detecting codes for memory systems," *IEEE Trans. Comput.*, vol. C-31, pp. 596-602, July 1982.
- [8] L. Penzo, D. Sciuto, and C. Silvano, "Construction techniques for systematic SEC-DED codes with single byte error detection and partial correction capability for computer memory systems," *IEEE Trans. Inform. Theory*, vol. 41, no. 2, pp. 584-591, Mar. 1995.
- [9] ———, "VLSI design of systematic odd-weight-column byte error detecting SEC-DED codes," in *Proc. 8th Int. Conf. VLSI Design*, New Delhi, India, Jan. 1995, pp. 156-160.
- [10] ———, "GECO: A tool for automatic generation of error control codes for computer applications," in *Proc. 1995 IEEE Int. Symp. Circuits Syst.*, Seattle, WA, Apr. 1995, pp. 912-915.
- [11] S. Lin and D. J. Costello, Jr., *Error Control Coding: Fundamentals and Applications*. Englewood Cliffs, NJ: Prentice-Hall, 1983.
- [12] M. Y. Hsiao, "A class of optimal minimum odd-weight-column SEC-DED codes," *IBM J. Res. Develop.*, July 1970, pp. 395-401.
- [13] J. O. Nicholson, "The RISC System/6000 SMP System," in *Proc. Spring CompCon '95*, San Francisco, CA, Mar. 1995.