On TSC Checkers for *m*-out-of-*n* Codes

V. V. Dimakopoulos, G. Sourtziotis,
A. Paschalis, and D. Nikolos

***Abstract*—Paschalis et al. in [12] have given a structured method to design TSC *m*-out-of-2*m* code checkers suitable for VLSI implementation. In this correspondence we give sufficient conditions so that the method given in [12] can be used to design checkers for classes of *m*-out-of-*n* codes with $n \neq 2m$.**

***Index Terms*—Fault detection, fault tolerance, MOS transistor implementation, *m*-out-of-*n* code, totally self-checking checkers.**

I. Introduction

The codes considered here are the well known m-out-of-n codes which are useful in detecting unidirectional errors, a type of errors occurring frequently in VLSI MOS circuits. A code word of such a code has exactly m 1s and $n - m$ 0s. In designing highly reliable systems, the totally self-checking (TSC) checkers play an important role as they are capable of detecting faults in the functional circuits they check, as well as faults in themselves. The concept of TSC checkers was introduced in [1] and formalized in [2] as follows:

Definition 1. *A circuit is self-testing for a set of faults F if, for every fault in F, the circuit produces a noncode output for at least one code input.*

Definition 2. *A circuit is fault secure for a set of faults F, if for every fault in F, the circuit never produces an incorrect code output for all code inputs.*

Definition 3. *A circuit is code disjoint if, during fault-free operation, code inputs map into code outputs and noncode inputs map into noncode outputs.*

Definition 4. *A circuit is a TSC checker if at the same time it is self-testing, fault secure and code disjoint.*

Under the assumption of the stuck-at fault model, a number of TSC checker design methods have been reported for the general case of *m*-out-of-*n* codes [2], [5], [6], [7], [8], [9]. The methods given in [2], [5], [6], [8], [9] are fully unstructured and thus they are not suitable for VLSI implementation. "Without the introduction of some structure, i.e., hierarchy and regularity, the problem of VLSI system design would be unmanageable" [14]. Efstathiou and Halatsis [7] have given a modular method to design TSC checkers for *m*-out-of-*n* codes, however a large number of different modules are used and the implementation cost is excessively large [10]. Other structured methodologies to design TSC *m*-out-of-*n* code checkers are not known from the open literature.

Structured methods to design TSC *m*-out-of-2*m* code checkers have been given in [3], [4], [12], [13]. The TSC checkers designed by the methods given in [3], [4], [13] have a cellular form while the TSC checker designed in [12] has the form of a binary tree. The root is a TSC two-rail checker while any one of the leaves is a weight generator. The TSC two-rail checker can be implemented as a tree of two-variable TSC two-rail code checkers [15]. Any one of the weight

Manuscript received July 29, 1993; revised Apr. 15, 1994.

V.V. Dimakopoulos is with the Department of Electrical and Computer Engineering, University of Victoria, P.O. Box 3055, Victoria, BC, Canada, V8W 3P6.

G. Sourtziotis and D. Nikolos are with the Department of Computer Engineering and Informatics, University of Patras, 26 500 Patra, Greece.

A. Paschalis is with the Institute of Informatics and Telecommunications, NCSR "Demokritos," 15 310 Aghia Paraskevi, Attiki, Greece; e-mail: paschali@iit.nrcps.ariadne-t.gr.

IEEECS Log Number C95086.

generators is implemented as a tree of full-adders and half-adders [16]. Therefore, very few different modules are necessary: half-adders, full-adders, and two-variable TSC two-rail code checkers. The method given in [12] as a structured method decreases the design time (increases productivity) and the chance of mistakes [17]. Moreover it has been shown in [12] that the TSC checkers designed by this method are superior, with respect to implementation cost, speed and test-set to the corresponding TSC checkers known from the literature. In this correspondence we extend the idea of [12] to classes of m-out-of-n codes with $n \neq 2m$ without sacrificing the above mentioned advantages.

The paper is organized as follows: Section II presents a summary of the method given in [12]. We assume that the reader is familiar with most of the details of the method in order to avoid unnecessary repetitions. Section III presents the extension of the method and Section IV concludes the work.

II. The TSC Checkers for m-out-of-$2m$ Codes

Efstathiou and Halatsis [11] presented a method for constructing TSC checkers for the special case of m-out-of-$2m$ codes with $m = 2^k - 1$. The $2m$ inputs are separated into two m-line groups and for each of the groups, a tree of adders is used to count the number of 1s. These adders are based on the ones presented in [16] for the design of TSC checkers for Berger codes. Since, under fault-free operation, the outputs of the two trees should add up to $m = 2^k - 1$, they must be complementary to each other. A two-rail checker is thus used to ensure this complement condition.

In the case that $m = 2^k - p - 1$, $0 < p < 2^{k-1}$, the outputs of the two trees are not complementary to each other. The sum of these outputs (i.e., the total number of 1s in the input of the checker) will be less than $2^k - 1$ by the constant p. A method to indirectly "add" the number p to the output of the adder trees, and thus make them complementary, was given by Paschalis et al. in [12]. The authors utilized a set of normal and complemented full-adders and half-adders to achieve the desired addition. The resulting checkers were characterised by a very small number of required gates and a very small test-set. They also have the additional advantage of straightforward VLSI MOS implementation using the techniques of [10].

Let the two outputs of an adder be denoted by C (carry) and S (sum). The binary number (CS) corresponds to the number of 1s in the inputs of the adder. The following definitions are necessary.

Definition 5. *The normal full-adder (NFA) is a full-adder with normal inputs x_1, x_2, and x_3, and complemented outputs $\overline{C}$ and $\overline{S}$.*

Definition 6. *The complemented full-adder (CFA) is a full-adder with complemented inputs $\overline{x}_1$, $\overline{x}_2$, and $\overline{x}_3$, and normal outputs C and S.*

Definition 7. *The normal half-adder (NHA) is a half-adder with normal inputs x_1 and x_2, and complemented outputs $\overline{C}$ and $\overline{S}$.*

Definition 8. *The complemented half-adder (CHA) is a half-adder with complemented inputs $\overline{x}_1$ and $\overline{x}_2$, and normal outputs C and S.*

Definition 9. *The normal half-adder plus one (NHAp1) is a circuit with normal inputs x_1 and x_2, and complemented outputs $\overline{C}$ and $\overline{S}$, where the binary number (CS) corresponds to the number of 1s in the inputs x_1 and x_2, plus one.*

Definition 10. *The complemented half-adder plus one (CHAp1) is a circuit with complemented inputs $\overline{x}_1$ and $\overline{x}_2$, and normal outputs C and S, where the binary number (CS) corresponds to the number of 1s in the inputs x_1 and x_2, plus one.*

The key observation is that the CHA is identical to NHAp1 and the CHAp1 is identical to the NHA, so that substituting CHAs for NHA's (and vice versa), at appropriate places, has the effect of adding the desired number p at the output of the adder tree.

III. The Proposed TSC m-out-of-n Code Checkers

Consider an adder tree T, as those used in [12], [16], that operates as an 1s count generator. T has $K > 1$ inputs and k outputs a_{k-1}, a_{k-2}, ..., a_0, where $k = \lceil \log_2(K+1) \rceil$, that is, $2^{k-1} \leq K \leq 2^k - 1$. Each of the adders in T receives inputs of the same weight 2^w and produces an output S of weight 2^w and an output C of weight 2^{w+1}, for $w = 0, 1, ..., k-2$. The output a_{k-1} comes from the C output of the (unique) adder that adds inputs of weight 2^{k-2}, termed hereafter Most Significant weight Adder (MSA). In case that $2^{k-1} \leq K \leq 3 \times 2^{k-2} - 1$ the MSA is a half-adder, while in case that $3 \times 2^{k-2} \leq K \leq 2^k - 1$ the MSA is a full-adder.

We consider that T is completely tested if any adder in T receives all possible input combinations. Depending on the value of K, T may not need the all-ones input in order to be completely tested. In fact, only a maximum L of ones with $L < K$ may be enough. The following theorem gives a sufficient test-set for T.

Theorem 1. *Let T be an adder tree with K inputs (where $2^{k-1} \leq K \leq 2^k - 1$) that generates the binary number (a_{k-1}, a_{k-2}, .., a_0) corresponding to the number of 1s in its inputs. If T receives the code words of the following codes:*

- *the 0-out-of-K code;*
- *the 2^w-out-of-K code, for all $w = 0, 1, ..., k-1$;*
- *the (3×2^w)-out-of-K code, for all $w = 0, 1, ..., k-3$;*
- *the $(3 \times 2^{k-2})$-out-of-K code, only if MSA is a full-adder;*

then T is completely tested.

Proof. Any adder in T, that adds inputs of weight 2^w ($0 \leq w \leq k-2$), receives the all 0s input combination when the code word of the 0-out-of-K code is appearing in the K inputs of T.

- Any adder in T, that adds inputs of weight 2^w ($0 \leq w \leq k-2$), receives the combinations with exactly one 1 when specific code words of the 2^w-out-of-K code is appearing in the K inputs of T, for all $w = 0, 1, ..., k-2$.
- Any adder in T, that adds inputs of weight 2^w ($0 \leq w \leq k-2$), receives the combinations with exactly two 1s when specific code words of the (2×2^w)-out-of-K (i.e., 2^{w+1}-out-of-K) code is appearing in the K inputs of T, for all $w = 0, 1, ..., k-2$.
- Any full adder in T, that adds inputs of weight 2^w ($0 \leq w \leq k-3$), receives the all 1s input combination when a code word of the (3×2^w)-out-of-K code is appearing in the K inputs of T, for all $w = 0, 1, ..., k-3$.
- If the MSA in T is a full-adder, then MSA receives the all 1s input combination when a code word of the $(3 \times 2^{k-2})$-out-of-K code is appearing in the K inputs of T. □

The following corollaries are derived from Theorem 1:

Corollary 1. *If an adder tree T with K inputs, where $2^{k-1} \leq K \leq 3 \times 2^{k-2} - 1$, receives the r-out-of-K codes (for all $r = 0, 1, ..., 2^{k-1}$) then it is completely tested.*

Corollary 2. *If an adder tree T with K inputs, where $3 \times 2^{k-2} \leq K \leq 2^k - 1$, receives the r-out-of-K codes (for all $r = 0, 1, ..., 3 \times 2^{k-2}$) then it is completely tested.*

At this point we remark that both Corollaries 1 and 2 are still valid in case that T generates the binary number (a_{k-1}, a_{k-2}, .., a_0) corresponding to the number of 1s in its inputs plus the constant p, that is, when some CHAs in T have been accordingly replaced by NHAs (and vice versa) as proposed in [12].

Based on Corollaries 1 and 2, we present here TSC m-out-of-n code checkers with $n \geq 2m$ whose structure is shown in Fig. 1. The proposed checkers consists of two adder trees T1 and T2 with k_1 and k_2 inputs, respectively, and a k-variable TSC two-rail code checker, where $n = k_1 + k_2$ and $\lceil \log_2(m+1) \rceil = k$. T1 and T2 must have the same number of output lines to be compared and thus the following relation should be satisfied:

$$\lceil \log_2(k_1+1) \rceil = \lceil \log_2(k_2+1) \rceil = k \tag{1}$$

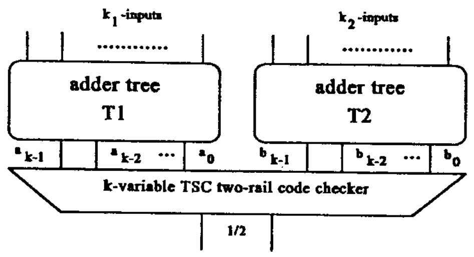

Fig. 1. The proposed TSC m-out-of-n code checker.

T1 generates the binary number $(a_{k-1}, a_{k-2}, ..., a_0)$ corresponding to the number m_1 of 1s in its inputs plus some constant p_1. T2 generates the binary number $(b_{k-1}, b_{k-2}, ..., b_0)$ corresponding to the number m_2 of 1s in its inputs plus some constant p_2. The adders utilized are the ones given in Definitions 5–8. The trees consist of alternating layers of normal and complementary adders, with inverters inserted where appropriate. The construction details can be found in [12]. The constants p_1 and p_2 have to be chosen so that the following relation is satisfied:

$$p_1 + p_2 = 2^k - 1 - m \tag{2}$$

The satisfaction of (2) assures that during normal operation, when the proposed checker receives inputs encoded in the m-out-of-n code, the binary numbers $(a_{k-1}, a_{k-2}, ..., a_0)$ and $(b_{k-1}, b_{k-2}, ..., b_0)$ are complementary to each other.

In some cases (e.g., see Fig. 2), T1 and T2 have different polarities in their corresponding outputs due to the fact that they have different number of adder layers. The problem is solved without any effort by putting inverters at the outputs of the adder tree with the least adder layers.

The k-variable TSC two-rail checker monitors the outputs of T1 and T2 and checks whether the binary numbers $(a_{k-1}, a_{k-2}, ..., a_0)$ and $(b_{k-1}, b_{k-2}, ..., b_0)$ are complementary to each other. The structure of the k-variable TSC two-rail checker is given in Fig. 3 [12], [16]. It consists of two 2-variable TSC two-rail code checkers (termed TRC1 and TRC2, respectively) and a $(k-2)$-variable TSC two-rail code checker (termed TRC3). The latter can be implemented in a tree structure or in two gate levels [15].

Theorem 2 gives sufficient conditions to be satisfied in order for the adder trees T1, T2, and the k-variable TSC two-rail checker to constitute a TSC m-out-of-n code checker.

THEOREM 2. *The proposed circuit of Fig. 1 is a TSC m-out-of-n code checker, with respect to the single stuck-at fault model, where* $n = k_1 + k_2$, $m \leq k_1 \leq k_2$ *and* $\lceil \log_2(m+1) \rceil = k$, *if both of the following conditions are satisfied:*

C1. *Either*

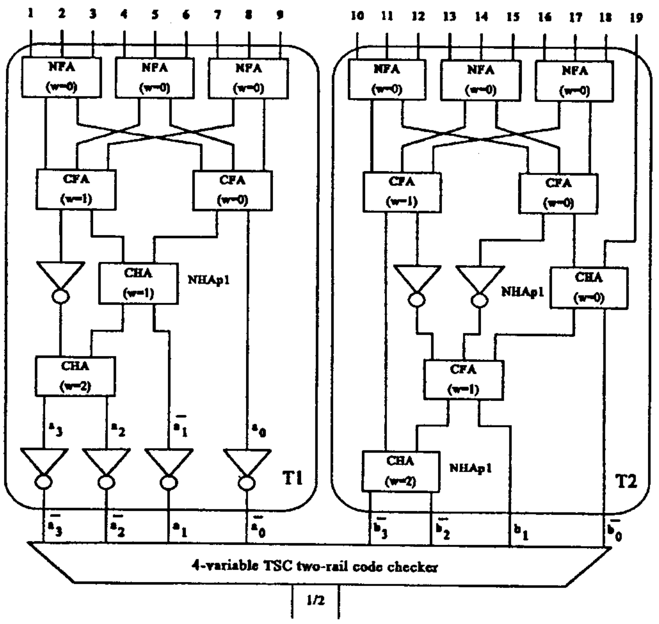

Fig. 2. The TSC 8-out-of-19 code checker with $k_1 = 9$ and $k_2 = 10$.

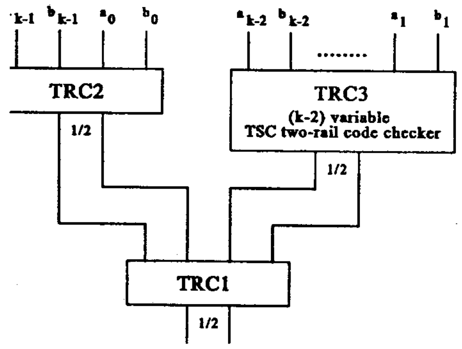

Fig. 3. The k-variable TSC two-rail code checker.

$$2^{k-1} \leq m \leq k_1 \leq k_2 \leq 3x2^{k-2} - 1 \tag{3}$$

or

$$3x2^{k-2} \leq m \leq k_1 \leq k_2 \leq 2^k - 1 \tag{4}$$

C2. *If '$\vee$' and '$\wedge$' are the bitwise* OR *and* AND *operations, correspondingly, on the binary representation of the numbers involved, then*

$$(k_1 \wedge k_2) \vee m = m \tag{5}$$

PROOF. The proposed circuit of Fig. 1 is a TSC m-out-of-n code checker if it is code disjoint self-testing and fault secure with respect to the single stuck-at fault model.

Code disjoint property. Following the same reasoning given in [12] we conclude that if (1) and (2) are satisfied the adder trees T1 and T2 constitute a code disjoint circuit that translates from the m-out-of-n code into the k-variable two rail code. We can see easily that (1) is satisfied if condition C1 is satisfied. Let us examine under which condition (2) is satisfied. In the case of the TSC m-out-of-$2m$ code checker proposed in [12], when there exists a 0 at the wth bit of the binary representation of m, a half adder that adds inputs of the weight 2^w is changed from CHA to NHA or vice versa, in either of the two adder trees; the replacement occurs for all such

w. These replacements have the desired effect of the constants p_1 and p_2 being added at the outputs of the adder trees, and thus make the outputs complementary. While there always exist half adders that add such weights w when both trees have m inputs, this is not always the case when k_1 and k_2 are not equal to m. Thus, (2) is satisfied if the binary representations of k_1 and k_2 have 0s in (at least) all the bit positions for which the binary representation of m has a 0, that is, if condition C2 is satisfied. This is seen by observing that $(k_1 \wedge k_2)$ gives a 0 at all the bit positions at which k_1 or k_2 have a 0. Bitwise ORing with m preserves the 0s as long as they coincide with the 0s in the binary representation of m.

The k-variable TSC two-rail checker is code disjoint by construction. Consequently, the whole TSC m-out-of-n code checker is code disjoint.

Self-testing property. If (3) is satisfied then the adder tree T1 receives at least the r-out-of-k_1 codes (for all $r = 0, 1, ..., 2^{k-1}$) and the adder tree T2 receives at least the r-out-of-k_2 codes (for all $r = 0, 1,.. ., 2^{k-1}$). Thus, from Corollary 1 it is derived that T1 and T2 are completely tested. Alternatively, if (4) is satisfied then the adder tree T1 receives at least the r-out-of-k_1 codes (for all $r = 0, 1, ..., 3 \times 2^{k-1}$) and the adder tree T2 receives at least the r-out-of-k_2 codes (for all $r = 0, 1, ..., 3 \times 2^{k-1}$). Thus, from Corollary 2 is derived that T1 and T2 are completely tested. Therefore, if condition C1 is satisfied then both adder trees T1 and T2 are completely tested. Following the procedure proposed by Marouf and Friedman [16], it is seen that for each of the adder trees T1 and T2, $8(k - 1)$ code words are enough to exhaustively test all single stuck-at faults. For the case of m-out-of-$2m$ codes the same number of code words is enough to test the two trees simultaneously. In the case of m-out-of-n codes, where $n > 2m$, the same number of test inputs may not be enough, so that in general, we have an upper bound of $16(k - 1)$ code words in order for both adder trees T1 and T2 to be completely tested.

The k-variable TSC two-rail checker consists of two 2-variable TSC two-rail code checkers (termed TRC1 and TRC2, respectively) and a $(k - 2)$-variable TSC two-rail code checker (termed TRC3) as it is shown in Fig. 3. Since condition C1 is satisfied, TRC3 receives all possible 2^{k-2} code words, independently of p_1 and p_2. The same holds for TRC1 and TRC2, when $2^{k-1} < m$. If $2^{k-1} = m$ then TRC1 and TRC2 are TSC only when both $p_1 \neq 0$ and $p_2 \neq 0$ [12]. In the TSC m-out-of-n code checker proposed here this can be achieved only if both k_1 and k_2 have some zero bits in the positions corresponding to zero bits in the binary representation of m. But this is always the case, because, since (3) is satisfied, all three numbers (m, k_1, and k_2) have their $(k - 2)$th bit equal to zero. Therefore, if condition C1 is satisfied then the k-variable TSC two-rail checker is self-testing and thus TSC with respect to single stuck-at faults. Consequently, the whole TSC m-out-of-n code checker is self-testing with respect to single stuck-at fault model.

Fault secure property. Following the same reasoning as in [12] we can see easily that the proposed TSC m-out-of-n code checker is fault secure with respect to single stuck-at fault model. □

For a given m-out-of-n code, the implementation can be done in more than one way, by varying the values of k_1 and k_2 as long as all the conditions are satisfied. For example, the TSC 8-out-of-19 code checker could be implemented using either two trees of 8 and 11 inputs or two trees with 9 and 10 inputs, correspondingly. Both implementations of the TSC 8-out-of-19 code checker are equivalent with respect to implementation cost and operation speed. Fig. 2 shows the checker for the code 8-out-of-19 consisting of a 9 input adder tree T1 and a 10 input adder tree T2.

Notice also that in a TSC m-out-of-n code checker designed as proposed here the replacement of normal adders (NFAs, NHAs) with complemented ones (CFAs, CHAs) and vice versa, in both trees T1 and T2, results in the corresponding TSC $(n - m)$-out-of-n code checker. Thus, our method is not only applicable in the design of TSC m-out-of-n code checkers with $n \geq 2m$, but also in the design of the corresponding TSC $(n - m)$-out-of-n code checkers.

Sample values for m and n, for which the circuits described here are TSC m-out-of-n code checkers, are given in Table I. From Table I it is derived that TSC m-out-of-n code checkers with values of m in the neighbourhood of $\lfloor n/2 \rfloor$ are presented in this paper. Thus, taking into account that among all m-out-of-n codes these codes have less redundancy, we conclude that in this paper TSC code checkers for the most useful m-out-of-n codes are given.

TABLE I
SAMPLE m-OUT-OF-n CODES FOR WHICH TSC CHECKERS ARE DESIGNED

m	n	m	n
4	8*, 9	16	32*, 33 ~ 39
5	9, 10*	17	33, 34*, 35 ~ 40
6	12*, 13	18	34, 35, 36*, 37 ~ 41
7	13, 14*	19	35 ~ 37, 38*, 39 ~ 42
8	16*, 17 ~ 19	20	36 ~ 39, 40*, 41 ~ 43
9	17, 18*, 19, 20	21	37 ~ 41, 42*, 43, 44
10	18, 19, 20*, 21	22	38 ~ 43, 44*, 45
11	19 ~ 21, 22*	23	39 ~ 45, 46*
12	24*, 25 ~ 27	24	48*, 49 ~ 55
13	25, 26*, 27, 28	...	....
14	26, 27, 28*, 29	30	54 ~ 59, 60*, 61
15	27 ~ 29, 30*	31	55 ~ 61, 62*

* *The TSC m-out-of-2m code checkers are designed as proposed in* [12].

IV. Conclusion

We presented sufficient conditions, under which, the TSC m-out-of-$2m$ code checkers proposed by Paschalis et al. in [12] can be extended to a class of m-out-of-n codes, where $n \neq 2m$. The resulting circuits retain all the advantages of the circuits in [12], namely small gate count, small test-set, design modularity, and straightforward MOS VLSI implementation. A more detailed analysis of the proposed extension can be found in [18].

References

[1] W.C. Carter and P.R. Schneider, "Design of dynamically checked computers," *Proc. IFIP '68*, vol. 2, pp. 878-883, Edinburgh, 1968.

[2] D.A. Anderson and G. Metze, "Design of totally self-checking check circuits for m-out-of-n codes," *IEEE Trans. Computers*, vol. 22, pp. 263-269, Mar. 1973.

[3] S.M. Reddy, "A note on self-checking checkers," *IEEE Trans. Computers*, vol. 23, pp. 1,100-1,102, Oct. 1974.

[4] J.E. Smith, "The design of totally self-checking check circuits for a class of unordered codes," *J. Design Automation Fault-Tolerant Computing*, vol. 2, pp. 321-342, Oct. 1977.

[5] M.A. Marouf and A.D. Friedman, "Efficient design of self-checking checkers for any m-out-of-n code," *IEEE Trans. Computers*, vol. 27, pp. 482-490, June 1978.

[6] N. Gaitanis and C. Halatsis, "A new design method for m-out-of-n TSC checkers," *IEEE Trans. Computers*, vol. 32, pp. 273-283, Mar. 1983.

[7] C. Efstathiou and C. Halatsis, "Modular realization of totally self-checking checkers for m-out-of-n codes," *Proc. 13th FTCS*, Milan, pp. 154-161, June 1983.

[8] S. Piestrak, "Design method of totally self-checking checkers for m-out-

of-n codes," *Proc. 13th FTCS*, Milan, pp. 162-168, June 1983.

[9] T. Nanya and Y. Tohma, "A 3-level realization of totally self-checking checkers for m-out-of-n codes," *Proc. 13th FTCS*, Milan, pp. 173-176, June 1983.

[10] N. Jha and A. Abraham, "Techniques for efficient MOS implementation of totally self-checking checkers," *Proc. 15th FTCS*, pp. 430-435, June 1985.

[11] C. Efstathiou and C. Halatsis, "Efficient modular design of m-out-of-2m TSC checkers, for $m = 2^k - 1$, $k > 2$," *Electronics Letters*, vol. 21, pp. 1,083-1,084, Nov. 1985.

[12] A. Paschalis, D. Nikolos, and C. Halatsis, "Efficient modular design of TSC checkers for m-out-of-2m codes," *IEEE Trans. Computers*, vol. 37, pp. 301-309, Mar. 1988.

[13] S. Piestrak, "The minimal test-set for sorting networks and the use of sorting networks in self-testing checkers for unordered codes," *Proc. 20th FTCS*, pp. 457-464 June 1990.

[14] C.H. Sequin, "Managing VLSI complexity: An outlook," *Proc. IEEE*, vol. 71, no 1, pp. 149-166, Jan. 1983.

[15] D.A. Anderson, "Design of self-checking digital networks using coding techniques," Coordinated Science Lab., Rep. R-527, Univ. of Illinois, Oct. 1971.

[16] M.A. Marouf and A.D. Friedman, "Design of self-checking checkers for Berger codes," *Proc. Eighth FTCS*, pp. 179-184, June 1978.

[17] M. Annaratone, *Digital CMOS Circuit Design*. Kluwer Academic Publishers, 1986.

[18] V. Dimakopoulos and G. Sourtziotis, "Design of TSC circuits for m-out-of-n codes in VLSI MOS technology," Diploma thesis (in Greek), Dept. of Computer Eng. and Informatics, Univ. of Patras, 1990.