ELSEVIER INTEGRATION, the VLSI journal 28 (1999) 101–118

INTEGRATION the VLSI journal

www.elsevier.com/locate/vlsi

New efficient totally self-checking Berger code checkers

X. Kavousianos, D. Nikolos*, G. Foukarakis, T. Gnardellis

Department of Computer Engineering and Informatics, University of Patras, 26500 Rio, Patras, Greece

Received 7 August 1997

Abstract

This paper presents a new method for designing totally self-checking (TSC) Berger code checkers for any number k of information bits, even for $k = 2^{r-1}$, taking into account realistic faults as stuck-at, transistor stuck-open, transistor stuck-on and resistive bridging faults. Double- as well as single-output TSC checkers are given. The proposed single-output TSC Berger code checkers are the first in the open literature. The checkers designed following the proposed method are significantly more efficient, with respect to the required area and speed, than the corresponding already known from the open literature checkers. © 1999 Elsevier Science B.V. All rights reserved.

Keywords: Self-checking circuits; Totally self-checking circuits; Berger codes; Unidirectional errors

1. Introduction

Self-checking circuits (SCC) [1] are widely used in applications with high reliability requirements, due to their ability to detect errors on line during the normal system operation. The errors covered include those caused by permanent, transient as well as intermittent faults. A SCC consists of a functional circuit, whose output words belong to a certain code, and a checker that monitors the output of the functional circuit and indicates if it is a code or a noncode word. The structure of such a SCC is shown in Fig. 1.

The reliability of a SCC depends on the ability of its checker to behave correctly despite the possible occurrence of internal faults. It has been shown that this is achieved when the checker satisfies either the totally self checking (TSC) [2] or the strongly code-disjoint (SCD) [3] property. In this paper we will take into account the TSC property. The TSC checker is a circuit which satisfies the self-testing, fault secure and code disjoint properties [2,4].

* Corresponding author. Fax: + 30-61-991-909.
E-mail addresses: kabousia@ceid.upatras.gr (X. Kavousianos), nikolosd@cti.gr (D. Nikolos)

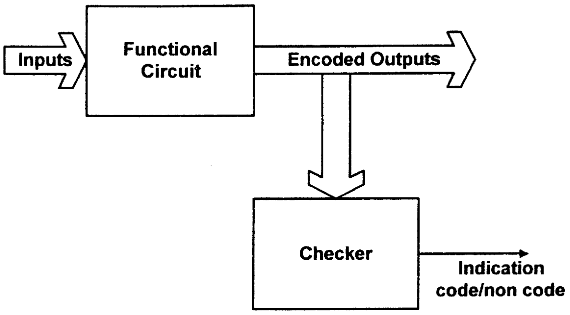


Fig. 1. Self-checking circuit.

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.

Some authors believe that the fault secure property is meaningless for checkers [5,6] while some others believe that it is useful for all others except for the final checker [7].

It has been observed for many years that a large number of errors in VLSI circuits and compact laser disks are of unidirectional type [8–10]. Berger codes [11] are the least redundant separable codes among the all unidirectional error detecting (AUED) codes [12]. For k information bits, the check part has length $r = \lceil \log_2(k+1) \rceil$. There are two different encoding schemes for Berger code: B_0 and B_1. The B_0 encoding scheme uses the binary representation of the number of 0's in the information bits as the check symbol, whereas the B_1 encoding scheme uses the ones complement of the number of 1's in the information bits.

Due to the wide use of Berger codes, several design methods of Berger code checkers were proposed in the open literature [13–19]. Among them only the design methods given in [16–19] can be used for designing Berger code checkers when the number of information bits is equal to 2^r. Taking into account that in most cases the information length is a power of two, we conclude that this is a great disadvantage of the methods given in [13–15]. The checkers given in [13–17] are TSC only with respect to single stuck-at faults; only the checkers designed in [18,19] are TSC with respect to a realistic fault model [20] including node stuck-at, transistor stuck-open, transistor stuck-on and resistive bridging faults. Apart from the above the checkers given in [19] have the advantage that require less area and feature higher speed than the checkers proposed in [16–18]. However, the checkers proposed in [19] have static power consumption.

In this paper a new method for designing TSC Berger code checkers is proposed. The checkers designed according to this method are TSC with respect to all node stuck-at, transistor stuck-on, transistor stuck-open and resistive bridging faults. The proposed checkers are significantly more efficient, with respect to area and speed, than the already known TSC Berger code checkers. Apart from this the proposed checkers have significantly lower power consumption than the checkers given in [19]. We use the B_1 encoding scheme, but the modifications for B_0 scheme are straightforward.

The rest of the paper is organised as follows. In Section 2 we give the proposed design method, Section 3 presents the testability analysis, while comparisons are given in Section 4.

2. Design method

The design of the proposed Berger code checkers is based on the use of the k-weight threshold circuit which is defined as follows.

Definition. A single output circuit is called a k-weight threshold circuit if when k or more of its inputs are high, its output is high else its output is low.

Consider the circuit of Fig. 2. In this circuit all nmos transistors have the same sizes W_{nm} and L_{nm}. The circuit of Fig. 2 is similar to the threshold function generator used in [19]. However, a systematic method for designing such a circuit has not been given in [19]. A systematic method to design a k-weight threshold circuit has been given in [21]. In [21] we have shown that the circuit of Fig. 2 is a k-weight threshold circuit as long as the ratio of the transistor aspect ratios

$$\frac{W}{L}=\frac{W_{\mathrm{pm}}/L_{\mathrm{pm}}}{W_{\mathrm{nm}}/L_{\mathrm{nm}}}=\frac{W_{\mathrm{pm}}\cdot L_{\mathrm{nm}}}{W_{\mathrm{nm}}\cdot L_{\mathrm{pm}}}$$

satisfies the following relation:

$$(k-1)\frac{\mathrm{KP_n}}{\mathrm{KP_p}}\cdot\frac{2(V_{\mathrm{dd}}-V_{\mathrm{tn}})V_{\mathrm{IHMIN}}-V_{\mathrm{IHMIN}}^2}{(V_{\mathrm{dd}}+V_{\mathrm{tp}})^2}\leqslant\frac{W}{L}\leqslant k\frac{\mathrm{KP_n}}{\mathrm{KP_p}}\cdot\frac{2(V_{\mathrm{dd}}-V_{\mathrm{tn}})V_{\mathrm{ILMAX}}-V_{\mathrm{ILMAX}}^2}{(V_{\mathrm{dd}}+V_{\mathrm{tp}})^2}. \tag{1}$$

The following notation has been used:

- V_{IHMIN} (V_{ILMAX}) is the minimum HIGH (maximum LOW) input voltage which is recognised as logic 1 (0) from a driven gate.
- V_{tn} (V_{tp}) is the threshold voltage of nmos (pmos) transistor.
- $\mathrm{KP_n}$($\mathrm{KP_p}$) is the Spice parameter for $\mu_{\mathrm{n}}\cdot C_{\mathrm{ox}}(\mu_{\mathrm{p}}\cdot C_{\mathrm{ox}})$.

The following relation gives all the possible values of k for which a k-threshold circuit can be designed:

$$k\leqslant\frac{1}{1-(2(V_{\mathrm{dd}}-V_{\mathrm{tn}})V_{\mathrm{ILMAX}}-V_{\mathrm{ILMAX}}^2)/(2(V_{\mathrm{dd}}-V_{\mathrm{tn}})V_{\mathrm{IHMIN}}-V_{\mathrm{IHMIN}}^2)}.$$

The relation above, is a reformulation of the relation we have given in [21]. From the above relation we can see that the values of k depend not only on the values of the noise margins but also on the transition voltage of the inverter. Therefore, in order to achieve large values of k with acceptable noise margins we can suitably select the transition voltage, that is, $\beta_{\mathrm{p}}/\beta_{\mathrm{n}}$ of the inverter.

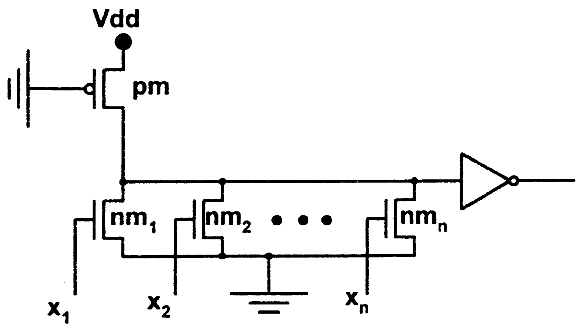


Fig. 2. k-weight threshold circuit.

Hereafter we will use the abbreviations

$$C_H = \frac{KP_n}{KP_p} \cdot \frac{2(V_{dd} - V_{tn})V_{IHMIN} - V_{IHMIN}^2}{(V_{dd} + V_{tp})^2},$$

$$C_L = \frac{KP_n}{KP_p} \cdot \frac{2(V_{dd} - V_{tn})V_{ILMAX} - V_{ILMAX}^2}{(V_{dd} + V_{tp})^2},$$

therefore from relation (1) we get

$$C_H \cdot (k-1) \leqslant \frac{W}{L} \leqslant C_L \cdot k. \tag{2}$$

In the sequel the k-weight threshold circuit will go through successive modifications in order to take a circuit suitable for the implementation of the Berger code checker.

Consider now the circuit of Fig. 3. In this circuit we have chosen all nmos transistors to have sizes $W_{nm}/L_{nm} = 1$. Then we can choose the size of the i pmos transistor, $1 \leqslant i \leqslant q$, so that the ratio W/L satisfies relation (2) for $k = i$. That is, the aspect ratio of the pmos transistors were chosen according to the following relations

$$C_H \cdot (i-1) \leqslant \frac{W_{pm_i}}{L_{pm_i}} \leqslant C_L \cdot i \tag{3}$$

for $i = 1, 2, \ldots, q$.

Therefore setting a line d_i equal to zero, $i \in \{1, 2, \ldots, q\}$, and the rest lines d_j equal to one, for $j = 1, 2, \ldots, q$ and $j \neq i$, the circuit of Fig. 3 functions as the i-weight threshold circuit. In other words, we can specify the weight of this circuit by setting the proper line d_i to low. This circuit will be called a (1 to q)-programmable-weight threshold circuit.

It is easy to modify the (1 to q)-programmable weight threshold circuit to a (0 to q)- programmable weight threshold circuit. To this end we append one pmos transistor, pm_0, and one nmos transistor, nm_0, to the (1 to q) - programmable - weight threshold circuit of Fig. 3, so as to get the circuit of Fig. 4. The sizes of the transistors pm_0 and nm_0 have been chosen equal to the sizes of pm_1 and nm_1 transistors, respectively. For $d_0 = 1$ both pm_0 and nm_0 are not conductive, hence the circuit behaves as a (1 to q) - programmable - weight threshold circuit. When $d_0 = 0$,

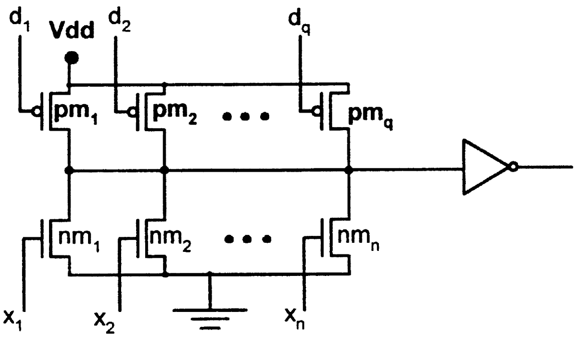


Fig. 3. (1 to q)-programmable weight threshold circuit.

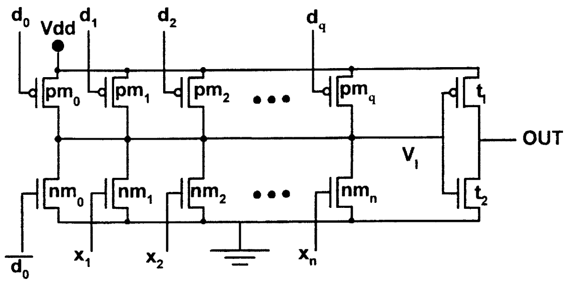


Fig. 4. (0 to q) programmable weight threshold circuit.

$d_1 = d_2 = \cdots = d_q = 1$, then from the pmos transistors $\mathrm{pm}_0, \mathrm{pm}_1, \ldots, \mathrm{pm}_q$, only pm_0 is conductive. Transistor nm_0 is also conductive. Then taking into account that $W_{\mathrm{pm}0}/L_{\mathrm{pm}0} = W_{\mathrm{pm}1}/L_{\mathrm{pm}1}$ and $W_{\mathrm{nm}0}/L_{\mathrm{nm}0} = W_{\mathrm{nm}1}/L_{\mathrm{nm}1}$ (nm_0 dominates pm_0) we conclude that regardless of how many input lines among of $X_1, X_2, \ldots, X_n$ are high, OUT remains high, so the circuit behaves in fact as a 0-weight threshold circuit. Therefore the circuit of Fig. 4 is a (0 to q) - programmable weight threshold circuit.

Adding a pmos transistor pm_I in the circuit of Fig. 4 we get the circuit of Fig. 5. The operation of the circuit of Fig. 5 depends on the value of the input I. The aspect ratio of pm_I was chosen according to the following relation.

$$C_\mathrm{H} \cdot i \leqslant \frac{W_{\mathrm{pm}_i}}{L_{\mathrm{pm}_i}} + \frac{W_{\mathrm{pm}_\mathrm{I}}}{L_{\mathrm{pm}_\mathrm{I}}} \leqslant C_L \cdot (i+1). \tag{4}$$

Relation (4) describes the operation of the circuit when both transistors pm_i, $i \in \{0, 1, \ldots, q\}$, and pm_I are conductive. It is well known that the parallel connection of two transistors p_1, p_2 with conductances G_1 and G_2 is equivalent to a single transistor with conductance $G = G_1 + G_2$. The conductance G_i of a transistor p_i is given in [[22] p. 61] as $G_i = D(W_i/L_i)$, where D is a factor which is the same for all transistors connected in parallel, so we have

$G = G_1 + G_2$ or equivalently,

$G = D(W_1/L_1 + W_2/L_2)$.

In other words, the parallel connection of two transistors p_1, p_2 behaves in the same way with a single transistor p with ratio equal to the sum of the ratios of the transistors p_1 and p_2. Then it is easy to see that when the input I is equal to one then due to relation (3) the circuit behaves identically to the circuit of Fig. 4. When the input I is equal to zero then for d_i equal to zero, $i \in \{0, 1, 2, \ldots, q\}$, and the rest inputs d_j equal to one, the circuit behaves as an $(i+1)$-weight threshold circuit, due to relation 4. In other words, for $I = 1$ the circuit of Fig. 5 behaves as a (0 to q)-programmable-weight threshold circuit, while for $I = 0$ it behaves as a (1 to $q+1$)-programmable-weight threshold circuit. We will call this circuit a (0 to q)/(1 to $q+1$)-programmable-weight threshold circuit.

For the implementation of the Berger code checker another module, the decoder shown in Fig. 6 will be used. The truth table of the decoder is shown in Table 1 for the case that we have seven information bits. The input CP is the system clock signal and it is used to isolate the outputs from the power supply (when CP = 1) in order to accelerate the $1 \to 0$ transition of the output that is to fall according to the information bits.

The proposed Berger code checker is shown in Fig. 7. $C_0 \ldots C_{r-1}$ is the check part of the Berger code word and $X_1, \ldots, X_n$ are the information bits. Let W denote the Hamming weight (number of ones) of the input vector $X_1, X_2, \ldots, X_n$ and d_m the output of the decoder that is equal to zero. Then the outputs of the checker for each possible input are shown in Table 2. It can be easily verified that

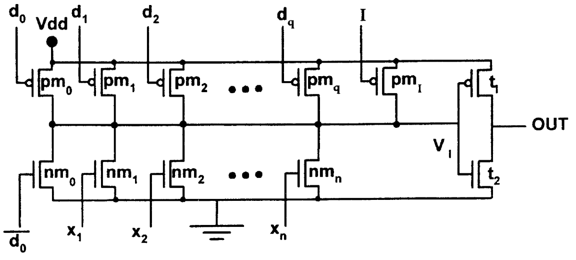


Fig. 5. (0 to q)/(1 to $q+1$) programmable weight threshold circuit.

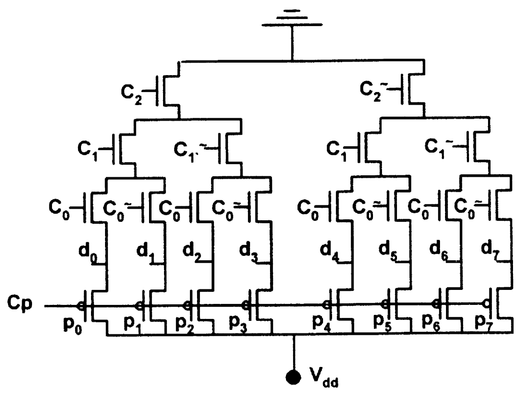


Fig. 6. Decoder design.

Table 1
Truth table for the 1-out-of-8 decoder

Check part			Decoder outputs								Weight
$c2$	$c1$	$c0$	$d7$	$d6$	$d5$	$d4$	$d3$	$d2$	$d1$	$d0$	$(X_1 \ldots X_7)$
1	1	1	1	1	1	1	1	1	1	0	0
1	1	0	1	1	1	1	1	1	0	1	1
1	0	1	1	1	1	1	1	0	1	1	2
1	0	0	1	1	1	1	0	1	1	1	3
0	1	1	1	1	1	0	1	1	1	1	4
0	1	0	1	1	0	1	1	1	1	1	5
0	0	1	1	0	1	1	1	1	1	1	6
0	0	0	0	1	1	1	1	1	1	1	7

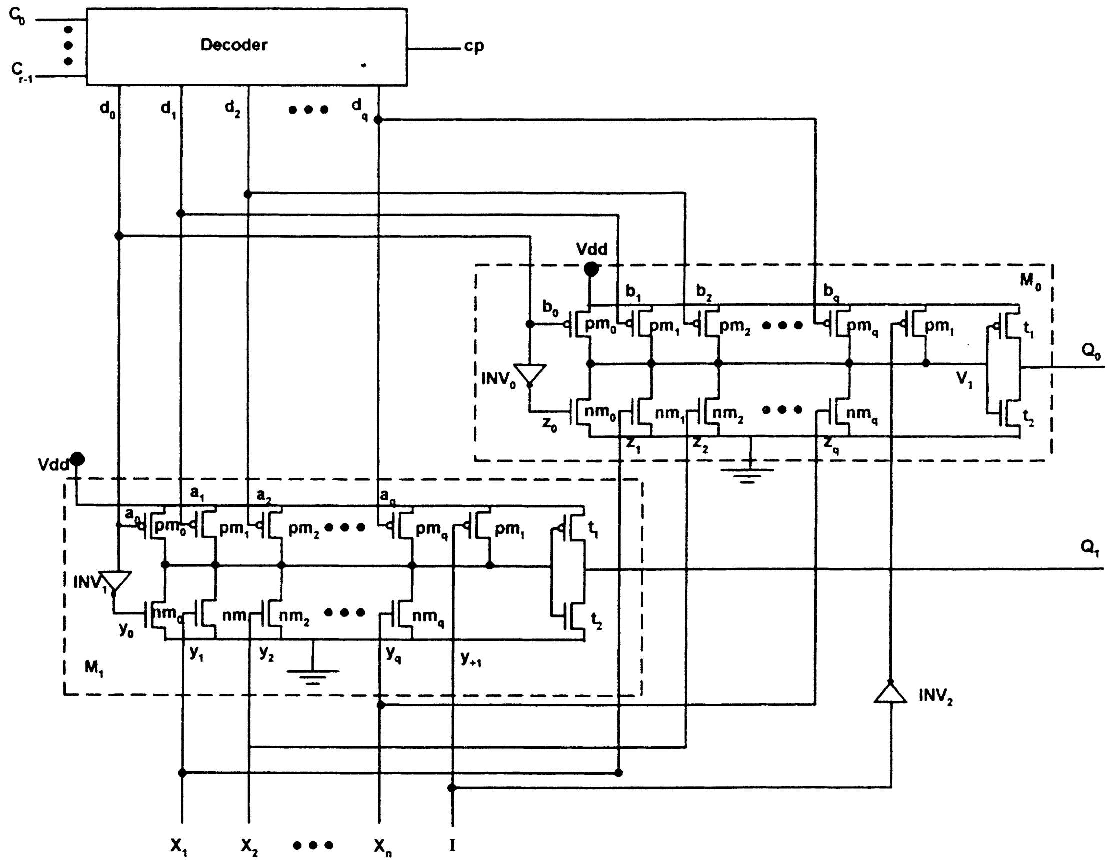


Fig. 7. Berger code checker for $n = 2^r - 1$.

when $I = 0$ then module M_0 operates as the (0 to q) - programmable - weight threshold circuit and module M_1 operates as the (1 to $q + 1$) - programmable - weight threshold circuit so if $d_i = 0, i \in \{0, 1, \ldots, q\}$, then module M_0 is the i-weight threshold circuit and module M_1 is the $(i + 1)$-weight threshold circuit. For $I = 1$, M_0 operates as a (1 to $q + 1$) - programmable - weight

Table 2

Input (I)	Hamming weight (W)	Output (Q_0)	Output (Q_1)
	$W < m$	0	0
0	$W = m$	1	0
	$W > m$	1	1
	$W < m$	0	0
1	$W = m$	0	1
	$W > m$	1	1

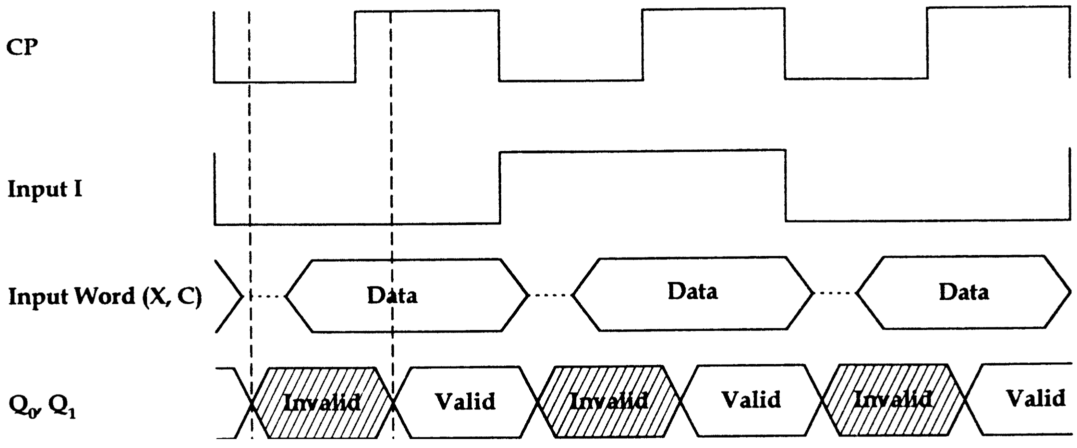


Fig. 8. Waveforms.

threshold circuit and M_1 as a (0 to q) - programmable - weight threshold circuit. The input I of the checker is driven by a clock signal with frequency equal to the half of the operation frequency of the checker, that is to the half of the system clock. The signal I can be easily generated from the system clock using T flip-flop. The operation waveforms of the checker are shown in Fig. 8. It is obvious that the Data ($X_1, X_2, \ldots, X_n, C_0, C_1, \ldots, C_{r-1}$) must be applied during the low phase of the clock CP and remain stable during the high phase. The response of the checker (Q_0, Q_1) is valid during the high phase of the clock.

From Table 2 we conclude that the circuit of Fig. 7 is a Berger code checker since for each encoded input, it produces the 2-rail encoded outputs 01 or 10 and for each non code input it produces the non-2-rail encoded outputs 00 or 11.

We have to note here that when the number of the information bits of the implemented Berger code is n, $n < 2^r - 1$, the decoder that we use has $q = n + 1$ outputs, where $n + 1 < 2^r$, that is, the decoder is not complete. In that case if the checker receives a noncodeword with check part $C_{r-1}C_{r-2} \ldots C_0$ where

$$\bar{C}_{r-1} \cdot 2^{r-1} + \bar{C}_{r-2} \cdot 2^{r-2} + \cdots + \bar{C}_0 \cdot 2^0 > n$$

then all the outputs of the decoder are inactive (high). This error is signaled, $(Q_0, Q_1) = (1, 1)$, if the information part of the noncodeword has at least one bit X_i high. If all the information bits X_i are low ($1 \leqslant i \leqslant n$), due to a unidirectional error, then the above error is not signaled. For that case we use one pmos transistor "test" in each module as shown in Fig. 9, with size equal to pm_1, which conducts only when $\mathrm{CP} = 0$. This transistor forces both modules to the state $(Q_0, Q_1) = (0, 0)$ when $\mathrm{CP} = 0$ and $W(X) = 0$ and this state remains unchanged for $\mathrm{CP} = 1$, under the presence of the above noncodeword, and consequently the error is signaled. Transistor "test" is conductive only in the phase $\mathrm{CP} = 0$ therefore it does not affect the normal operation of the circuit in any other way. Therefore, when the number of information bits n is equal to $2^r - 1$ either the checker of Fig. 7 or 9 can be used, while for $n \neq 2^r - 1$ the checker of Fig. 9 must be used.

In the sequel we will give two alternative designs that require less implementation area and exhibit lower power consumption, however they are slower than the checkers given in Figs. 7 and 9.

In Fig. 10 the module M_1 is identical to the module M_1 of Fig. 7. The input I is driven by a periodic signal with frequency equal to the double of the frequency of the CP signal. This is the reason that the test transistors used in the checker of Fig. 9 are unnecessary in this design. From Table 2 we can see that the design of Fig. 10 is a Berger code checker. With this structure we have

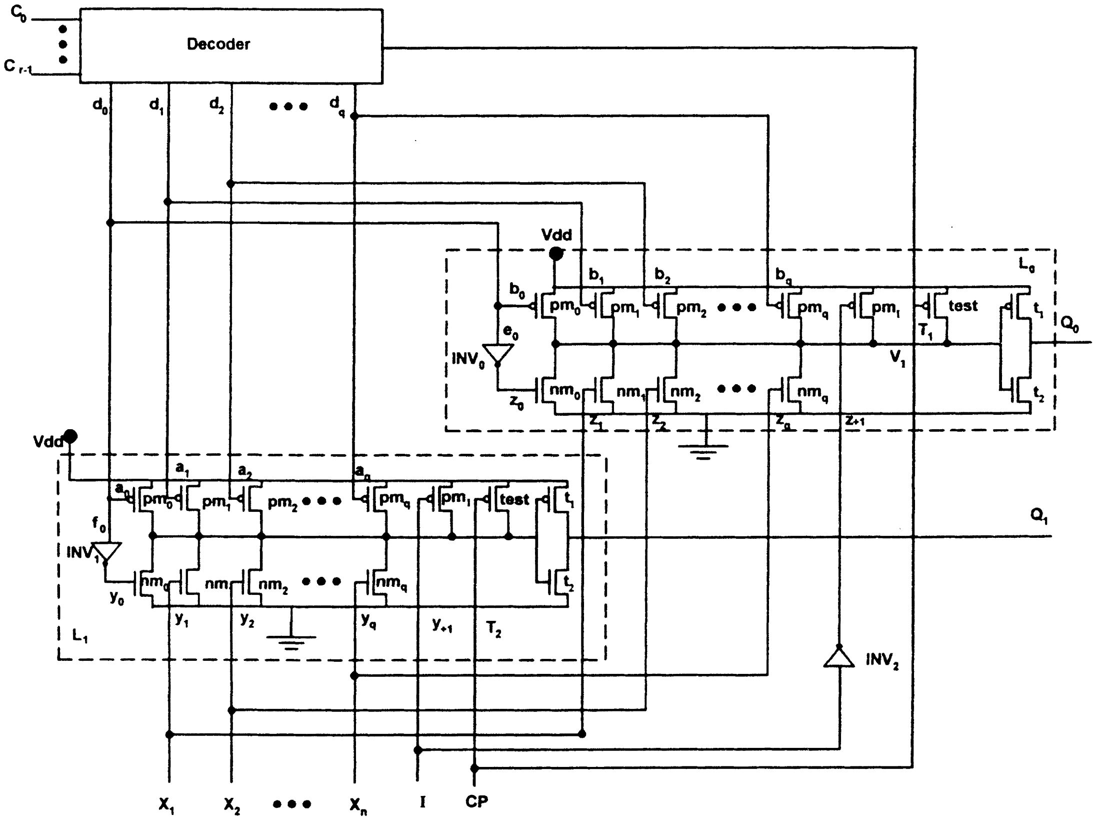


Fig. 9. Berger code checker for $n \neq 2^r - 1$.

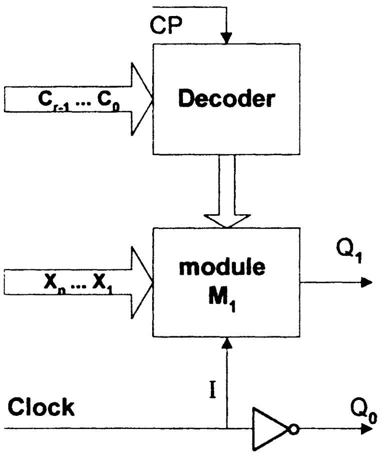


Fig. 10.

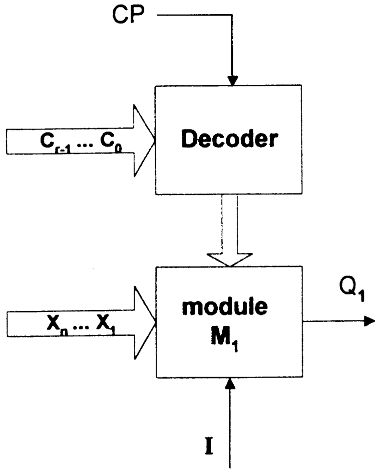


Fig. 11. Single output berger code TSC checker.

a further reduction in area as well as reduction in power consumption in comparison with the checker of Fig. 7 or 9.

There are cases that a single output TSC checker with its output two-rail encoded in time may have some advantages over the double output checker [30,31]. For example the routing of the error signals coming from different checkers would be simplified. There are also applications, for example automotive controllers and smart cards, where the system poses particular constraints on the number of possibly used input/output signals [23]. To this end we propose the single output checker of Fig. 11. In this design the frequency of the clock signal I should be equal to the double of the frequency of the CP signal, and the module M_1 is identical to the module M_1 of Fig. 7 checker. We can easily verify, using Table 2, that the design of Fig. 11 is a single output Berger code checker.

When the input vector is a code word and the checker is fault free then during a period of CP the output Q_1 gets the values (0, 1). When the input vector is not a code word then during a period the output Q_1 gets the values (1, 1) or (0, 0).

3. Testability analysis

We will firstly investigate the testability of the checker of Fig. 9.

In the appendix we give a detailed testability analysis for the node stuck-at, transistor stuck-on and transistor stuck-open faults. The conclusion is that all the faults of this type are detectable during the normal operation, except of a stuck-on fault on the pmos transistor of the inverter INV_0 or INV_1 or INV_2 or $\mathrm{Inverter}_i$ or transistor t1 which is undetectable but the checker behaviour remains unchanged under this fault. After the occurrence of any one of the undetectable faults or of any sequence of them the checker remains code disjoint. Furthermore if any sequence of them is followed by a detectable fault, the resulting fault is detectable.

The checker of Fig. 9 is not self-testing with respect to a stuck-open fault at one of the two transistors test. However, we have to note here that stuck-open faults have been verified to be less likely to occur than the other faults [20,24,25]. Moreover, the possibility for such faults to occur can be reduced further if the checker layout is suitably designed [26–28]. Additionally, these faults can be detected off-line, using the following two noncode words, $C'X'$ and $C''X''$, where $W(C') = W(C'') = W(X'') = 0$ and $W(X') > 0$. Applying the vector $C'X'$ for $I = 0$ and in the sequence the vector $C''X''$ for $I = 1$ we get after the application of the second input vector $(Q_0, Q_1) = (0, 0)$ for fault free operation and $(Q_0, Q_1) = (1, 0)$ when the test transistor of module M_1 is stuck-open. To detect a stuck-open fault on the test transistor of module M_0 we apply the input vector $C'X'$ for $I = 1$ and in the sequence the vector $C''X''$ for $I = 0$. Then, after the application of the second vector we get for the fault free operation $(Q_0, Q_1) = (0, 0)$ while under the stuck-open fault we get $(Q_0, Q_1) = (0, 1)$.

The self-checking capability of the proposed designs with respect to resistive bridging faults has been evaluated with extensive circuit-level simulations. Resistive bridging faults (RBFs) between two transistor terminals or between two inputs or between d_i and d_{i+1}, or a_i and a_{i+1}, or b_i and b_{i+1} lines, with $i \in \{1, 2, \ldots, q-1\}$, have been considered. All RBFs with connecting resistance $R \in [0, R\mathrm{max}]$ are detected, where Rmax depends on the sizing of the transistors. For an implementation in 1 μm technology we used the transistor aspect ratios (W/L) shown in Table 3. We are interested for resistances in the range [0, 6 KΩ] [24]. During the simulation the inputs of the checker are driven by standard cell inverters with aspect ratios $(W/L)p = 12$ and $(W/L)n = 6$. Specifically we found that all the resistive bridging faults are detected in the range [0, 6 K], except those shown in Table 4.

The testability analysis for the checkers of Figs. 7, 10 and 11 is similar to the analysis made for the checker of Fig. 9. However, the checkers of Figs. 7, 10 and 11 have the advantage that they do not use the transistors test. Furthermore, the checker of Fig. 11 has one more advantage that a stuck-at 0 or 1 fault on line I is detected without the use of a checker for periodic signals.

It is very easy to verify that for any single fault the output of the Berger code checker (Figs. 7, 9 and 10 or 11) is the correct code word or a noncode word, therefore the checker is fault secure.

Table 3

Transistor	Module	W/L
pmos of Inverter$_i$	Decoder	4/1
nmos of Inverter$_i$	Decoder	2/1
nmos driven by C_i, $C_i^{\sim}$	Decoder	6/1
p_i	Decoder	2/1
pm$_0$, pm$_1$, pm$_{\text{test}}$	L_0 or L_1	10/2
pm$_2$	L_0 or L_1	22/2
pm$_3$	L_0 or L_1	16/1
pm$_4$	L_0 or L_1	22/1
pm$_5$	L_0 or L_1	26/1
pm$_6$	L_0 or L_1	32/1
pm$_7$	L_0 or L_1	38/1
pm$_8$	L_0 or L_1	42/1
nm$_i$, $i \in \{1, \ldots, n\}$	L_0 or L_1	2/1
pmos of INV$_0$, INV$_1$, INV$_2$	L_0 or L_1	4/1
nmos of INV$_0$, INV$_1$, INV$_2$	L_0 or L_1	2/1
t$_1$	L_0 or L_1	4/1
t$_2$	L_0 or L_1	2/1

Table 4

Transistor	Type of bridging fault	$R_{\max}$ (KΩ)
p*i* of decoder	Source–gate	1.1
nmos of Inverter$_i$ of decoder	Source–gate	4
nmos of Inverter$_i$ of decoder	Source–drain	1.2
nm$_i$	Gate–drain	4
pm$_0$	Source–drain	3
pm$_0$	Gate–source	3
pm$_1$	Gate–source	1.5
pm$_2$	Source–drain	5

4. Comparisons and discussion

Among the already known TSC checkers for Berger codes the checkers proposed in [19] are the most efficient with respect to the required area and speed as well as the fault model. These Berger code checkers take into account apart from single stuck-at faults, stuck-on, stuck-open and resistive bridging faults too. The checkers proposed in this paper, as we will show, are more compact, faster and have significantly lower power consumption than the checkers given in [19] while they use the same fault model.

We will present firstly a qualitative comparison between the proposed checkers and the checkers given in [19].

The proposed design is much smaller than that of [19] because:

1. It is obvious that we have compacted the TFG module of [19] in only two rows of the n used there (where n is the number of information bits) using the two programmable weight threshold circuits. So we use only $2 \cdot n$ nmos transistors while TFG in [19] uses n^2 nmos transistors.
2. The decoder module of the proposed checkers is much smaller from the rest modules, TSFC, BDEC, CDEC and TRC, of the checkers given in [19].

With respect to speed we can say that module L_0 or L_1 have similar delay with the module TFG in [19], while the delay of the decoder is expected to be smaller than the aggregate delay of modules TSFC, BDEC and TRC.

We have implemented the proposed Berger code checkers, the design of Fig. 9, as well as that given in [19] for 8 information bits in 1 μm technology. The proposed here Berger code checker of Fig. 9 features reductions in area, delay and power consumption equal to 53.9%, 50% and 73.1%, respectively.

A further reduction of power consumption and the required area can be achieved using the TSC checker given in Fig. 10 or 11. However, the maximum frequency of operation of these checkers is half of that of the checkers of Figs. 7 and 9, that is, it is the same with the maximum frequency of operation of the checker given in [19].

5. Conclusion

In this paper we presented a novel method for designing double and single output TSC Berger code checkers under a realistic fault model including stuck-at, stuck-on, stuck-open and resistive bridging faults.

The checkers designed following our method are significantly more efficient with respect to area and speed in comparison to the corresponding already known checkers. The proposed single output TSC Berger code checkers are the first known in the open literature.

Appendix A

We present the testability analysis for the checker of Fig. 9, where the input I is driven by a clock signal with frequency equal to half of the frequency of the system clock signal. We consider the node stuck-at, transistor stuck-on and transistor stuck-open faults in each module of the checker. Throughout the appendix X and X' will denote the information parts of two different Berger code words.

A.1. Faults at the decoder

The decoder is shown in Fig. 6 for the case that we have 3 check bits and 7 information bits. The inverted inputs $C_i^{\sim}$ are driven by inverters (Inverter$_i$) which are not shown in the figure. The analysis is similar for all the possible cases.

1. Line d_i stuck-at 0. When the checker receives a code word with $W(X)=j, j \neq i$, then $(Q_0, Q_1)=(0,0)$.
2. Line d_i stuck-at 1. When the checker receives a code word with $W(X)=i$ then for $i>0$ we have $(Q_0, Q_1)=(1,1)$, while for $i=0$ we have $(Q_0, Q_1)=(0,0)$ due to the existence of transistors "test". In both cases the fault is detected.
3. Line CP stuck-at 0. It is the same with 2.
4. Line CP stuck-at 1. When the checker receives two successive code words with $W(X) \neq W(X')$, then for the second code word we get $(Q_0, Q_1)=(0,0)$.
5. Line C_i stuck-at 0. When the checker receives a code word with $C_i=1$ then if the information part has weight $W(X)>0$ then $(Q_0, Q_1)=(1,1)$, else $(Q_0, Q_1)=(0,0)$ due to the existence of the "test" transistor. In both cases the fault is detected.
6. Line $C_i^{\sim}$ stuck-at 0. When the checker receives a code word with $C_i=0$ then if the information part has weight $W(X)>0$ then $(Q_0, Q_1)=(1,1)$ else $(Q_0, Q_1)=(0,0)$ due to the existence of the "test" transistor. In both cases the fault is detected.
7. Line C_i stuck-at 1 or line $C_i^{\sim}$ stuck-0. When the checker receives a vector with $C_i=0$ then $(Q_0, Q_1)=(0,0)$.
8. Stuck-on fault on one of the nmos transistors driven by signals C_i $(C_i^{\sim})$ shown in Fig. 6. When the checker receives a code vector with $C_i=0$ $(C_i=1)$ then $(Q_0, Q_1)=(0,0)$.
9. Stuck-open fault on one of the nmos transistors driven by signals C_i $(C_i^{\sim})$ shown in Fig. 6. When the checker receives a code word with $C_i=1$ $(C_i=0)$ then if the information part has weight $W(X)>0$ then $(Q_0, Q_1)=(1,1)$ else $(Q_0, Q_1)=(0,0)$ due to the existence of the "test" transistor. In both cases the fault is detected.
10. Transistor p_i stuck-open. When the checker receives two successive code words with $W(X) \neq W(X')$ and $W(X)=i$ then for the second code word we get $(Q_0, Q_1)=(0,0)$.
11. Stuck-on fault on the nmos transistor of the inverter Inverter$_i$. The Inverter$_i$ is constructed with n-dominate logic so when the checker receives a code word with $C_i=0$ then if the information part has weight $W(X)>0$ then $(Q_0, Q_1)=(1,1)$ else $(Q_0, Q_1)=(0,0)$ due to the existence of the "test" transistor. In both cases the fault is detected.
12. Stuck-open fault on the nmos transistor of the inverter Inverter$_i$. When the checker receives two successive code words with C_i equal to 0 and 1, respectively, then for the second word we get $(Q_0, Q_1)=(0,0)$.
13. Stuck-open fault on the pmos transistor of the inverter Inverter$_i$. When the checker receives two successive code words with C_i equal to 1 and 0, respectively, then if the information part of the second vector has weight $W(X)>0$ then $(Q_0, Q_1)=(1,1)$ else $(Q_0, Q_1)=(0,0)$ due to the existence of the "test" transistor. In both cases the fault is detected.

A.2. Faults that affect both modules L_0 and L_1.

Such faults are stuck-at 0/1 on lines $d_0 \ldots d_q$ and $X_1 \ldots X_n$.

1. Line X_i stuck-at 0. When the checker receives a code word with $X_i=1$ then $(Q_0, Q_1)=(0,0)$.
2. Line X_i stuck-at 1. When the checker receives a code word with $X_i=0$ then $(Q_0, Q_1)=(1,1)$.
3. Line d_i stuck at 0/1. These faults are analysed in (1) and (2) of Section A.1 respectively.
4. Line I stuck-at 0 or 1. These faults are detected with a checker for periodic signals [29].

A.3. Faults that affect only module L_0.

1. Line Z_i stuck-at 0 or transistor nm$_i$ stuck-open. When the checker receives a code word with $X_i = 1$ and $I = 0$ then $(Q_0, Q_1) = (0, 0)$.
2. Line Z_i stuck-at 1 or transistor nm$_i$ stuck-on. When the checker receives a code word with $X_i = 0$ and $I = 1$ then $(Q_0, Q_1) = (1, 1)$.
3. Line Z_{+1} stuck-at 0 or transistor pm$_\mathrm{I}$ stuck-on. When the checker receives a code word with $I = 0$ then $(Q_0, Q_1) = (0, 0)$.
4. Line Z_{+1} stuck-at 1 or transistor pm$_\mathrm{I}$ stuck-open. When the checker receives a code word with $I = 1$ then $(Q_0, Q_1) = (1, 1)$.
5. Line b_i stuck-at 0 or transistor pm$_i$ stuck-on. When the checker receives a code word with $W(X) \neq i$ and $I = 0$ then $(Q_0, Q_1) = (0, 0)$.
6. Line b_i stuck-at 1 or transistor pm$_i$ stuck-open. When the checker receives a code word with $W(X) = i$ and $I = 1$ then $(Q_0, Q_1) = (1, 1)$.
7. Line e_0 stuck-at 0. When the checker receives a code word with $W(X) \neq 0$ and $I = 1$ then $(Q_0, Q_1) = (1, 1)$.
8. Line e_0 stuck-at 1. When the checker receives a code word with $W(X) = 0$ then $(Q_0, Q_1) = (0, 0)$.
9. Line V_1 stuck-at 0 or line Q_0 stuck-at 1. When the checker receives a code word with $I = 1$ then $(Q_0, Q_1) = (1, 1)$.
10. Line V_1 stuck-at 1 or line Q_0 stuck-at 0. When the checker receives a code word with $I = 0$ then $(Q_0, Q_1) = (0, 0)$.
11. Transistor t$_2$ stuck-on. The inverter at the output of the module is constructed with n-dominate logic, so this fault is detected when the checker receives a code word with $I = 0$ because $(Q_0, Q_1) = (0, 0)$.
12. Transistor t$_1$ stuck-open. When the checker receives two successive code words with I equal to 1 and 0, respectively, then for the second code word we get $(Q_0, Q_1) = (0, 0)$.
13. Transistor t$_2$ stuck-open. When the checker receives two successive code words with I equal to 0 and 1, respectively, then for the second code word we get $(Q_0, Q_1) = (1, 1)$.
14. Transistor nmos of the inverter INV$_0$ stuck-open. When the checker receives two successive code words with $W(X) = 0$, $W(X') \neq 0$ and I equal to 0 and 1, respectively, then for the second word we get $(Q_0, Q_1) = (1, 1)$.
15. Transistor nmos of the inverter INV$_0$ stuck-on. The inverter INV$_0$ is constructed with n-dominate logic so when the checker receives a code word with $W(X) = 0$ we have $(Q_0, Q_1) = (0, 0)$.
16. Transistor pmos of the inverter INV$_0$ stuck-open. When the checker receives two successive code words with $W(X) \neq 0$, $W(X') = 0$ and I equal to 1 and 0, respectively, then for the second code word we get $(Q_0, Q_1) = (0, 0)$.
17. Transistor nmos of the inverter INV$_2$ stuck-open. When the checker receives two successive code words with I equal to 0 and 1, respectively, then for the second code word we get $(Q_0, Q_1) = (1, 1)$.
18. Transistor nmos of the inverter INV$_2$ stuck-on. The inverter is constructed with n dominate logic, so when $I = 0$ we get $(Q_0, Q_1) = (0, 0)$.

19. Transistor pmos of the inverter INV_2 stuck-open. When the checker receives two successive code words with I equal to 1 and 0, respectively, then for the second code word we get $(Q_0, Q_1) = (0, 0)$.
20. Transistor test stuck-on or line T_1 stuck-at 0. When the checker receives a code word and $I = 0$ we have $(Q_0, Q_1) = (0, 0)$.
21. Line I_0 stuck-at 0. When the checker receives a code word and $I = 1$ then $(Q_0, Q_1) = (1, 1)$.
22. Line I_0 stuck-at 1. When the checker receives a code word and $I = 0$ then $(Q_0, Q_1) = (0, 0)$.

A.4. *Faults that affect only module L_1.*

1. Line Y_i stuck-at 0 or transistor nm_i stuck-open. When the checker receives a code word with $X_i = 1$ and $I = 1$ then $(Q_0, Q_1) = (0, 0)$.
2. Line Y_i stuck-at 1 or transistor nm_i stuck-on. When the checker receives a code word with $X_i = 0$ and $I = 0$ then $(Q_0, Q_1) = (1, 1)$.
3. Line f_0 stuck-at 0. When the checker receives a code word with $W(X) \neq 0$ and $I = 0$ then $(Q_0, Q_1) = (1, 1)$.
4. Line f_0 stuck-at 1. When the checker receives a code word with $W(X) = 0$ and $I = 1$ then $(Q_0, Q_1) = (0, 0)$.
5. Transistor t_1 stuck-open. When the checker receives two successive code words with I equal to 0 and 1, respectively, then for the second code word we get $(Q_0, Q_1) = (0, 0)$.
6. Transistor pmos of the inverter INV_1 stuck-on. The inverter INV_1 is constructed with p-dominate logic so when the checker receives a nonzero code word with $I = 0$ we have $(Q_0, Q_1) = (1, 1)$.
7. Transistor pmos of the inverter INV_1 stuck-on. When the checker receives 2 successive code words with $W(X) \neq 0$, $W(X') = 0$ and I equal to 0 and 1, respectively, then for the second vector we get $(Q_0, Q_1) = (0, 0)$.
8. Transistor nmos of the inverter INV_1 stuck-open. When the checker receives two successive code words with $W(X) = 0$, $W(X') \neq 0$ and I equal to 0 and 1, respectively, then for the second code word we get $(Q_0, Q_1) = (1, 1)$.
9. Line I_1 stuck-at 0. When the checker receives a code word and $I = 1$ then $(Q_0, Q_1) = (0, 0)$.
10. Line I_1 stuck-at 1. When the checker receives a code word and $I = 0$ then $(Q_0, Q_1) = (1, 1)$.

The rest of the faults affecting module L_1 are detected in the same way as in module L_0 with the modification that in each vector we need the complemented value of I.

A stuck-on fault on the pmos transistor of the inverter INV_0 or INV_1 or INV_2 or $\mathrm{Inverter}_i$ or transistor t1 is undetectable but the checker behaviour remains unchanged under this fault. After the occurrence of any one of them or of any sequence of them the checker remains code disjoint. Furthermore if any sequence of them is followed by a detectable fault, the resulting fault is detectable. As for the undetectable stuck open faults at the transistors test or line T_1 stuck-at 1 or line T_2 stuck-at 1 we have already referred at the testability analysis section.

References

[1] W.C. Carter, P.F. Schneider, Design of dynamically checked computers, in Proceedings of fourth Congress IFIP, Edinburg, Scotland, Vol. 2, August 5–10, 1968, pp. 878–883.
[2] D.A. Anderson, Design of self-checking digital network, using coding techniques, Coordinated Sci. Lab. Univ. Illinois, Urbana-Champaign, Rep. R-527, June 1984.
[3] M. Nicolaidis, B. Courtois, Strongly code disjoint checkers, IEEE Trans. Comput. 37 (1988) 751–756.
[4] T.R.N. Rao, E. Fujiwara, Error-control Coding for Computer Systems, Prentice-Hall, Englewood Cliffs, NJ, 1989.
[5] Y. Tamir, C.H. Sequin, Design and application of Self-Testing Comparators Implemented with MOS PLAs, IEEE Trans. Comput. C-33 (1984) 493–506.
[6] M. Nikolaidis, Fault secure property versus strongly code disjoint checkers, IEEE Trans. CAD 13 (5) (1994) 651–658.
[7] D. Nikolos, A. Paschalis, G. Philokyprou, Efficient design of totally self-checking checkers for low cost arithmetic codes, IEEE Trans Comput. 38 (1988) 807–814.
[8] D.K. Pradhan, J.I. Stiffler, Error correcting codes and self-checking circuits in fault tolerant computers, Computer, 27–37.
[9] E.L. Leiss, Data integrity in digital optical disks, IEEE Trans. Comput. 33 (9) (1984) 818–827.
[10] E. Fujiwara, D.K. Pradhan, error-control coding in computers, Computer 23 (1990) 63–72.
[11] J.M. Berger, A note on error detection codes for asymmetric binary channels, Inform. Control 4 (1961) 68–73.
[12] C.V. Freiman, Optimal error detection codes for completely asymmetric binary channels, Inform. Control 5 (1962) 64–71.
[13] M.A. Marouf, A.D. Freidman. Design of self-checking checkers for Berger codes. In Proceedings of International Symposium on Fault-Tolerant Comput., 1978, pp. 179–184.
[14] S. Piestrak, Design of fast self-testing checkers for a class of berger codes, Trans. Comput. C-36 (1987) 629–634.
[15] J.C. Lo, S. Thanawastien, The design of fast totally self-checking Berger code checkers based on Berger code partitioning, in Proceedings of FTCS-19, June 1988, pp. 226–231.
[16] T.R.N. Rao, G. Feng, M.S. Kolluru, J.C. Lo, Novel totally self-checking berger code checker designs based on generalized berger code partitioning, IEEE Trans. Comput. 42 (1993) 1020–1024.
[17] D. Pierce, P.K. Lala, Efficient self-checking checkers for berger codes, in Proceedings of first IEEE International On-Line Testing Work, 1995, pp. 238–242.
[18] C. Metra, M. Favalli, B. Ricco, Novel berger code checker, in Proceedings of IEEE International Work. On Defect and Fault Tolerance in VLSI Systems, 1995, pp. 287–295.
[19] C. Metra, J.C. Lo, Compact and high speed berger code checker, second IEEE International On-Line Testing Workshop, Biarritz, Franze, July 8–10, 1996, pp. 144–149.
[20] J. Shen, W. Maly, F. Ferguson, Inductive fault analysis of MOS intergrated circuits, IEEE Des. Test of Comput. (1985) 26–33.
[21] X. Kavousianos, D. Nikolos, Self-Exercising, Self-testing k-order comparators, 15th IEEE VLSI Test Symposium, April 27–May 1, 1997, Monterey, California, pp. 216–221.
[22] N.H.E. Weste, K. Eshraghian, Principles of CMOS VLSI Design, A systems Perspective, 2nd Edition, Addison-Wesley, Reading, MA.
[23] C. Metra, M. Favalli, B. Ricco, Highly testable and compact single output comparator, 15th IEEE VLSI Test Symposium, April 27–May 1, 1997, Monterey, California, pp. 210–215.
[24] H. Hao, E.J. McCluskey, Resistive shorts within CMOS gates, in Proceedings of IEEE International Test Conference 1991, pp. 292–301.
[25] F.J. Ferguson, J.P. Shen, Extraction and simulation of realistic CMOS faults using inductive fault analysis, Proceedings of IEEE International Test Conference 1988, pp. 475–484.
[26] S. Koeppe, Optimal layout to avoid stuck-open faults, in Proceedings of Design Automation Conference 1987, pp. 829–835.
[27] J.M. Soden, K. Treece, M.R. Taylor, C.F. Hawkins, CMOS IC stuck-open fault electrical effects and design considerations, in Proceedings of IEEE International Test Conference, 1989, pp. 423–430.

[28] J. Teixeira, I. Teixeira, C. Almeida, F. Goncalves, J. Goncalves, A methodology for testability enhancement at layout level, J. Electron. Testing: Theory Appl. 1 (1991) 287–299.

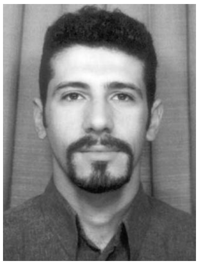
[29] A.M. Usas, A totally self-checking checker design for the detection of errors in periodic signals, IEEE Trans. Comput. C-24 (5) (1975) 483–488.

[30] C. Metra, M. Favalli, B. Ricco, Highly Testable and Compact Single Output Comparator, 15th IEEE VLSI Test Symposium, April 27–May 1, 1997, Monterey, California, pp. 210–215.

[31] S. Kundu, E.S. Sogomonyan, M. Goessel, S. Tarnick, Self-checking comparator with one periodic output, IEEE Trans. Comput. 45 (1996) 379–380.

Xrysovalantis I. Kavousianos is currently pursuing his Ph.D. at the University of Patras, Greece in the area of On-Line Testing. He holds a diploma degree in Computer Engineering from the same Department. In 1997 he was awarded a scholarship of the Technical Chamber of Greece due to his excellent student records. He is also a student member of IEEE. His other research interests include VLSI design and Low-Power Testing.

Dimitris Nikolos received the B.Sc. degree in physics in 1979, the M.Sc. degree in electronics in 1981, and the Ph. D. degree in computer science in 1985, all from the University of Athens. Dr. Nikolos is currently full professor in the Department of Computer Engineering and Informatics at the University of Patras and head of the Technology and Computer Architecture Laboratory. He served as program cochairman of the last three IEEE International On-Line Testing Workshops. He also served on the program committees for the IEEE International On-Line Testing Workshop in 1995 and 1996, for the 1997, 1998 and 1999 IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems and for the Third European Dependable Computing Conference. His main research interests are fault-tolerant computing, computer architecture, VLSI design, test, and design for testability. He is a member of the IEEE.



George N. Foukarakis received the Diploma in Computer Engineering from the University of Patras, Greece in 1998. He is currently a M. Sc. candidate at the same University, where he is doing research in the area of Array Signal Processing. His other research interests include array radar processing, and DSP architectures.

Theodore H. Gnardellis received the Diploma in Computer Engineering from the University of Patras, Greece in 1998. During the time period of 1994–1995, he worked as a systems operator at the Computer Technology Institute in Patras. From 1995 and nearly till the end of his studies, he conducted research work in VLSI design and testing. He is currently serving in the Hellenic Army, at the R & D Information Technology Center in the Pentagon, Athens.