Self - Exercising Self Testing k-order Comparators

X. Kavousianos & D. Nikolos

Department of Computer Engineering and Informatics
University of Patras, 26500, Patras, Greece
e-mail : kabousia@ceid.upatras.gr, nikolosd@cti.gr

Abstract

In this paper we give a systematic method to design Self - Exercising (SE) [1] self testing k-order comparators. The k-order comparator is defined as a combinational circuit that compares two operands and decides if these differ in less than k bits. According to this definition the usual equality comparator is the 1st-order comparator. Also in this paper we discuss the applicability of the k-order comparators in the implementation of (k-1)-EC/AUED, (k-1)-EC/d-ED/AUED, (k-1)-EC/d-UED and (k-1)-EC/d-ED/f-UED codes [2-7] as well as in the design of a fault tolerant cache memory [9] and broadcast networks.

I. Introduction

Self Checking Circuits (SCC) [10] are widely used in applications with high reliability requirements, due to their ability to detect errors on line during the normal system operation. The type of errors covered include those caused by permanent, transient as well as intermittent faults. A SCC consists of a functional circuit, the output words of which belong to a certain code, and a checker that monitors the output of the functional circuit and indicates whether a code word or a noncode word has appeared.

To achieve the totally self-checking goal (i.e., the first erroneous output of a functional block is signaled by the checker) [8] the checker was proposed to be Totally Self-Checking (TSC) [11] or strongly Code Disjoint (SCD) [12]. However the achievement of the totally self-checking goal in practice depends on the actual input vectors that the checker receives during the operation of the functional unit, which usually differs from application to application.

In [1] it was shown that Self-Exercising (SE) Self-Testing or Strongly Code Disjoint checkers are more close to achieve the Totally Self-Checking goal than TSC and SCD checkers. Besides a SE checker have the advantage that can be designed to be self-testing or strongly code disjoint for a more realistic fault model than TSC and SCD checkers. In this work we give a systematic method to design SE self-testing k-order comparators. SE self-testing checkers were defined in [1] as follows:

Definition. The self-exercising checker is self-testing with respect to a fault set F if for each fault f in F, either the checker receives during normal operation a code input that produces a noncode output, or a noncode output is produced to primary outputs (Z_o, Z_1) (figure 1) due to the test phase.

SE self-testing k-order comparator is suitable for a wide range of applications. As we will see in section IV the k-order comparator can be used in the implementation of (k-1)-EC/AUED, (k-1)-EC/d-ED/AUED, (k-1)-EC/d-UED and (k-1)-EC/d-ED/f-UED codes. Also the k-order comparator can be used for the implementation of a new way of exploitation of the (k-1)-EC/d-ED code, well suited to the tag part of cache memories [9]. As it was shown in [9] this implementation is faster and requires significantly less hardware than the classical implementation of these codes.

The implementation of the (k-1)-EC/d-ED code using a k-order comparator is also suitable in broadcast networks where in order to cope with errors occurring during the transfer of the packets the destination address is encoded in a (k-1)-EC/d-ED code. The classical implementation of the code implies that each host includes a (k-1)-EC/d-ED error decoder where the possible errors in the destination address are corrected and then the corrected destination address is compared with the host address. On the contrary using the k-order comparator the encoded destination address is compared in each host with the host address encoded in the (k-1)-EC/d-ED code. If the compared addresses differ in less than k bit positions a match is signaled.

An obvious design of the k-order comparator consists of three modules. The first module is a comparator implemented as a row of XOR gates, whose outputs drive the inputs of a weight generator. The weight generator is implemented as a tree of full and half adders (this is the faster implementation among the already known) and its

0-8186-7810-0/97 $10.00 © 1997 IEEE

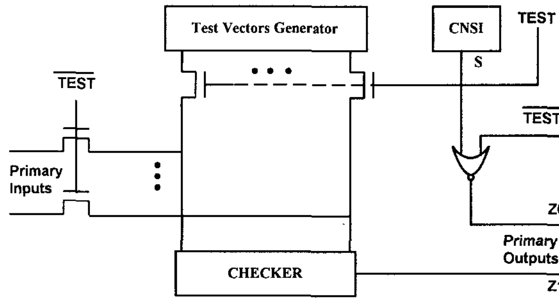

Figure 1. Structure of a Self-Exercising checker.

output is compared with the number k using a magnitude comparator. The propagation delay of this implementation as well as its hardware cost is large. The k-order comparators we propose in this work are much more efficient, with respect to the required hardware and delay than the above described k-order comparators as well as the comparators used in [9].

II. Design Method

The general structure of a self-exercising checker is given in Figure 1 (Figure 3 in [1]). In our case the module named checker is the k-order comparator. In subsection A we give a systematic method to design the k-order comparator and in subsection B we give some experimental results.

A. k-order Comparator Design.

The structure of the k-order comparator is given in Figure 2. Module D is similar to the threshold function generator given in [16]. However a systematic method for designing such a circuit has not been given in [16]. In the sequel we will analyze the operation of the D module and we will derive the conditions under which the comparator can be designed, as well as the proper nmos and pmos transistor sizes (width and length).

Let V_{IHMIN} (V_{ILMAX}) be the minimum HIGH (maximum LOW) input voltage which is recognized as logic 1 (0) from a gate.

Taking into account the above definitions and the definition of the k order comparator we conclude that module D must be designed in such a way that:

$V_{out} \geq V_{IHMIN}$ when less than k of the transistors q_1, q_2, ..., q_n are conductive and $V_{out} \leq V_{ILMAX}$ when at least k of them are conductive, where V_{IHMIN} and V_{ILMAX} refer to the buffer consisting of two inverters in figure 2. We can see that when none of the transistors q_1, q_2, ..., q_n is conductive then V_{out}=5 volts because transistor t1 is permanently conductive. When one transistor q_i, $i \in \{1, 2, ..., n\}$ starts to conducting then V_{out} is dropping by a value that depends on the resistance ratio between the transistors

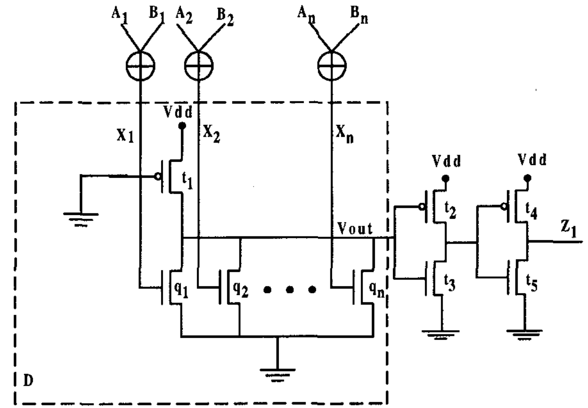

Figure 2. k-order comparator

q_i and t_1. For each additional conductive transistor q_j, $j \in \{1, 2, ..., n\}$, V_{out} drops a little more and the above design targets can be redefined as follows:

$V_{out} \geq V_{IHMIN}$ when k-1 of the transistors q_1, q_2, ..., q_n are conductive and $V_{out} \leq V_{ILMAX}$ when k of the transistors q_1, q_2, ..., q_n are conductive. It is obvious that the range of our interest is between the boundaries [V_{ILMAX}, V_{IHMIN}].

The following analysis is based on the basic DC equations given in [13, pp 51-52]. We start from the behavior of an nmos transistor. The threshold voltage of nmos is $V_{tn}>0$. We consider that $V_g=V_{dd}=5$ volts, $V_s=V_{gnd}=0$ volts and $V_d=V_{out}$.

In the linear region we have $V_{gs}-V_{tn}>V_{ds}$ or equally $V_{dd}-V_{tn}>V_{out}$ and the current is

$$I_{dsn} = \beta_n \cdot \left[(V_{dd} - V_{tn}) \cdot V_{out} - \frac{V_{out}^2}{2} \right].$$

In the saturation region the condition is $V_{out}>V_{dd}-V_{tn}$ and the current is

$$I_{dsn} = \beta_n \cdot \frac{(V_{dd} - V_{tn})^2}{2}, \quad \text{where } \beta_n = KP_n \cdot \frac{W_n}{L_n}$$

(KP is the Spice parameter for $\mu \cdot Cox$).

In the region of our interest [V_{ILMAX}, V_{IHMIN}], we have $V_{IHMIN}<V_{dd}-V_{tn}$ therefore among the transistors q_1, q_2, ..., q_n the conductive ones are in the linear region, while the rest are in the cutoff region.

The threshold voltage of a pmos transistor is $V_{tp}< 0$. We consider $V_g=V_{gnd}$, $V_s=V_{dd}$ volts and $V_d=V_{out}$. In the linear region we have $V_{gs} - V_{tp} > V_{ds}$ or equally $-V_{dd} - V_{tp} > V_{out} - V_{dd}$ or $V_{out} < -V_{tp}$ and the current is

$$I_{dsp} = \beta_p \cdot \left[(-V_{dd} - V_{tp}) \cdot (V_{out} - V_{dd}) - \frac{(V_{out} - V_{dd})^2}{2} \right].$$

In the saturation region we have $V_{out}>-V_{tp}$ so

$$I_{dsp} = \beta_p \cdot \frac{(V_{dd} + V_{tp})^2}{2}, \quad \text{where } \beta_p = KP_p \cdot \frac{W_p}{L_p}.$$

In our case we have V_{ILMAX} >-V_{tp} and -V_{tp}<1 volt so in the region of interest [V_{ILMAX}, V_{IHMIN}] the transistor t_1 is in the saturation region.

Assuming that λ of the transistors $q_1, q_2, .., q_n$ are conductive we get $I_{dsp} = \lambda \cdot I_{dsn}$ or equivalently

$$\frac{\beta_p}{\lambda \cdot \beta_n} = \frac{2 \cdot (V_{dd} - V_{tn}) \cdot V_{out} - V_{out}^2}{(V_{dd} + V_{tp})^2} \quad (1)$$

The function $f(x) = \frac{-x^2 + 2 \cdot (V_{dd} - V_{tn}) \cdot x}{(V_{dd} + V_{tp})^2}$ is maximized at the point $x=V_{dd}-V_{tn}$ and f(x) is monotone increasing for $x<V_{dd}-V_{tn}$. We are interested in the region [V_{ILMAX},V_{IHMIN}] so we assume that $V_{out} \le V_{dd}-V_{tn}$. Then taking into account the monotony of the function we conclude that when $V_{out} \ge V_{IHMIN}$ we have $f(V_{out}) \ge f(V_{IHMIN})$. According to the definition of the k-order comparator when k-1 of the transistors $q_1, q_2, \ldots, q_n$ are conductive the output voltage must satisfy the condition $V_{out} \ge V_{IHMIN}$. Setting λ=k-1 in equation (1) we get

$$\frac{\beta_p}{\beta_n} \ge (k-1) \frac{2 \cdot (V_{dd} - V_{tn}) \cdot V_{IHMIN} - V_{IHMIN}^2}{(V_{dd} + V_{tp})^2} \quad (2)$$

For $V_{out} \le V_{ILMAX}$ we get $f(V_{out}) \le f(V_{IHMIN})$. According to the definition of the k-order comparator when k of the transistors $q_1, q_2, \ldots, q_n$ are conductive the output voltage must satisfy the condition $V_{out} \le V_{ILMAX}$. Setting λ=k in equation (1) we get

$$\frac{\beta_p}{\beta_n} \le k \cdot \frac{2 \cdot (V_{dd} - V_{tn}) \cdot V_{ILMAX} - V_{ILMAX}^2}{(V_{dd} + V_{tp})^2} \quad (3)$$

Taking into account that $\beta_p/\beta_n=(KP_p/KP_n) \cdot W/L$ with $W=W_p/W_n$ and $L=L_p/L_n$, from relations (2) and (3) we get :

$$(k-1) \cdot \frac{KP_n}{KP_p} \cdot \frac{2 \cdot (V_{dd} - V_{tn}) \cdot V_{IHMIN} - V_{IHMIN}^2}{(V_{dd} + V_{tp})^2} \le \frac{W}{L} \le$$

$$\le k \cdot \frac{KP_n}{KP_p} \cdot \frac{2 \cdot (V_{dd} - V_{tn}) \cdot V_{ILMAX} - V_{ILMAX}^2}{(V_{dd} + V_{tp})^2} \quad (4)$$

Relation (4) implies

$$k \le \frac{2 \cdot (V_{dd} - V_{tn}) \cdot V_{IHMIN} - V_{IHMIN}^2}{(V_{IHMIN} - V_{ILMAX}) \cdot [2 \cdot (V_{dd} - V_{tn}) - V_{IHMIN} - V_{ILMAX}]}$$

The relation above gives all the possible values of k for which a k-order comparator can be designed.

Relation (4) is used to determine the boundaries for W/L, so as the k-order comparator to satisfy the conditions that initially have been set. After the specification of these boundaries, the designer has only to select the appropriate values of W_p, L_p, W_n, L_n .

Taking under consideration the fact that the circuit consists of only one pmos transistor (t_1) and n nmos transistors ($q_1, q_2, \ldots, q_n$), W_n, L_n must be both minimum for area optimization. On the other hand, large values of W_n/L_n reduce the transition time, optimizing in that way the design for delay.

B. Experimental Results and Discussion.

For our experiments we used the following tools :
Alliance Cad System 3.0 (Graphic Layout Editor V1.10, Netlist extractor V1.10).
Cazm: circuit analyzer using macromodeling.
Sigview: X11 tool for displaying Analog and Digital Simulation Data.

The technology that we used for our experiments is the SCN08H with minimum feature size 1.0 micron. Some typical values for this technology are V_{tn}=0,7522 volts V_{tp}=-0,8433 volts, $KP_n=1.207 \cdot 10^{-4}$ and $KP_p=3.434 \cdot 10^{-5}$.

We simulated the output buffer consisting from two-inverters in figure 2 to find its transition region [V_L, V_H]. Then we selected the values V_{IHMIN} = 2,5 volts and V_{ILMAX} = 1,9 volts in order to satisfy the condition $V_{IHMIN}>V_H$ and $V_{ILMAX}<V_L$. For these values we have noise margins NM_L=1,90 volts and NM_H=2,49 volts.

With the above values for noise margins we can design k-order comparators with $k \in [1, 6]$. Here we laid emphasis on large noise margins. Shortening the noise margins we can design comparators for greater values of k.

For k-order comparators with k=2 and k=3 we calculated the theoretical boundaries of W/L. For each such range we selected specific values for W/L with step 0.25 and for each value we designed two k-order comparators, one for area and the other for delay optimization. The selected values for W_n, L_n, W_p, L_p for each case of optimization are :
area optimization: W_n=1μ, L_n=1μ, W_p/L_p = W/L
delay optimization: W_n=4μ, L_n=1μ, W_p/L_p= 4W/L

For the area optimized designs we have selected for W/L values with step 0.5 because a step equal to 0.25 leads to very large pmos transistor, and thus excessive area requirements. For example if we select the value W/L=3.75 then for the area optimized designs we have W_p/L_p=3.75 and the minimum values for W_p, L_p are 15 and 4 respectively, while for W/L=3.5 we have W_p=7, L_p=2.

The timing analysis of the k-order comparator [17] implies that the circuit is getting faster when the value of W_n/L_n increases. So for the delay optimization we should select the maximum possible value for W_n/L_n. For L_n we selected the minimum possible value 1μ. For W_n we selected the value 4μ to avoid excessive increase of W_p.

For each circuit we derived with simulation the values of D_0 and D_1 where D_0 and D_1 are the delays for the transitions 1→0 and 0→1 respectively. D_0 time was

Table 1 2-order comparators for n=16

W/L	Area optimization					Delay optimization				
	Wp	Lp	D0 (ns)	D1 (ns)	Area (μm)	Wp	Lp	D0 (ns)	D1'(ns)	Area (μm)
3,25						13	1	2,09	1,78	77
3,5	7	2	3,35	2,68	30	14	1	2,16	1,73	78
3,75						15	1	2,24	1,7	79
4	4	1	3,67	2,42	20	16	1	2,35	1,67	80
4,25						17	1	2,5	1,64	81
4,5	9	2	4,14	2,37	34	18	1	2,6	1,61	82
4,75						19	1	2,73	1,59	83
5	5	1	5,13	2,24	21	20	1	3,12	1,57	84
Area Optimization : Wn=1 Ln=1										
Delay Optimization : Wn=4 Ln=1										
Theoritical boundaries : [3.046 , 5.096]										

Table 2 3-order comparators for n=16

W/L	Area optimization					Delay optimization				
	Wp	Lp	D0 (ns)	D1 (ns)	Area (μm)	Wp	Lp	D0 (ns)	D1 (ns)	Area (μm)
6,25						25	1	2,16	1,51	89
6,5	13	2	3,23	2,03	42	26	1	2,29	1,5	90
6,75						27	1	2,38	1,48	91
7	7	1	3,59	1,9	23	28	1	2,44	1,47	92
7,25						29	1	2,59	1,47	93
7,5	15	2	3,94	1,91	46	30	1	2,83	1,46	94
7,75						31	1	2,95	1,44	95
Area Optimization : Wn=1 Ln=1										
Delay Optimization : Wn=4 Ln=1										
Theoritical boundaries : [6.092 , 7.641]										

measured for the case where exactly k nmos transistors started to conducting simultaneously. This is the worst case since from the timing analysis [17] we derived that the discharging time is reduced as the number of conductive nmos increases.

Looking at the experimental results (Tables 1,2) we can make some observations :

1. As it was expected, the design for delay optimization gives smaller delay than the design for area optimization. However, the difference becomes smaller as the order of the comparator increases. Besides that, W/L increases as the order of the comparator increases and for the case of delay optimization (where $W_n/L_n=4$) W_p is getting excessively large.
2. As the value of W_p/L_p increases the value of D_1 becomes smaller as it was commented in [17].
3. For each k-order comparator we observe that as W/L increases, D_0 increases too, so for area as well as for delay optimized designs. This happens because in both cases we keep β_n constant and we change the value of β_p. As we can see from (5) in [17] β_p influences load capacity C_L, so when β_p increases, C_L increases and D_0 is getting worse.

III. Test Vector Generator.

In the case of the k-order comparator the test Vector Generator (Figure 1) consists of two n-bit shift registers A and B (Figure 3), where n is the length of each input vector. In each shift register the output of the last cell drives the input of the first cell through an inverter. Shift register B is initialized with the all-zero state while the shift register A is initialized with the state (1....10.......0) where the count of ones and zeroes is respectively equal to k and n-k and the least significant bit is at the left.

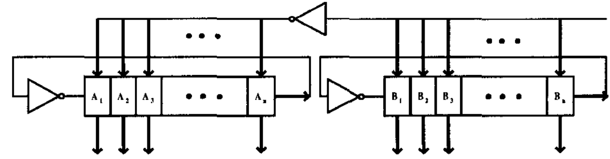

Figure 3. Test Vectors Generator

In Table 3 we give the sequences that are generated by the shift registers A and B. When the XOR gates receive as inputs the vectors of Table 3 generate the sequence of vectors presented in Table 4. From Table 6 we can see that module D receives in turn vectors with Hamming weight k and k-1 respectively. Therefore, during the test phase when D is fault free its output will be in turn equal to zero and one. Taking into account the above and Figure 1 we conclude that the module CNCI can be realized by a flip-flop. This flip-flop changes its state at every clock pulse and generates the sequence 0101. The period of the clock input of the flip-flop should be half the period of the clock input of the shift registers A and B. Then during the test phase and for fault free operation the outputs Z_0, Z_1 will be double-rail encoded. From Table 3 we can see that the self-exercising k-order comparator of length n is tested by a test set consisting of 4n vectors.

The SE k-order comparator of Figure 2 is self testing with respect to the following faults:

a. Single or multiple stuck-at zero faults at lines X_1, X_2, ..., X_n. From Table 4 we can see that for each line X_i, $1\leq i\leq n$, there exist an input vector X with Hamming weight equal to k and $X_i=1$. Therefore, when D receives as input the vector X and X_i is stuck-at zero the output of D will be equal to one while it was expected to be equal to zero. Then the output Z_0, Z_1 is not double-rail encoded and the fault is detected. We can easily see that the vectors of Table 4 with Hamming weight equal to k detect also all multiple stuck-at zero faults.

b. Single or multiple stuck-at one faults at lines X_1, X_2, ..., X_n . From Table 4 we can see that for each line X_i, $1\leq i\leq n$, there exist an input vector X with Hamming weight equal to k-1 and $X_i=0$. Therefore, when D receives as input the vector X and X_i is stuck-at one the output of D will be equal to zero, while is was expected to be equal to one. Then the output Z_0, Z_1 is not double-rail encoded and the fault is detected. The vectors of Table 4 with Hamming weight equal to k-1 detect also all multiple stuck-at one faults.

c. A stuck-open fault at the transistor q_i is equivalent to a stuck-at zero fault at line X_i. Thus the single and multiple transistor stuck-open faults are detected.

Table 3

A_1	A_2	A_3	..	A_{k-3}	A_{k-2}	A_{k-1}	A_k	A_{k+1}	A_{k+2}	A_{k+3}	..	A_{n-2}	A_{n-1}	A_n	B_1	B_2	B_3	..	B_{n-k-2}	B_{n-k-1}	B_{n-k}	B_{n-k+1}	B_{n-k+2}	B_{n-k+3}	B_{n-k+4}	..	B_{n-3}	B_{n-2}	B_{n-1}	B_n
1	1	1	..	1	1	1	1	0	0	0	..	0	0	0	0	0	0	..	0	0	0	0	0	0	0	..	0	0	0	0
1	1	1	..	1	1	1	1	0	0	0	..	0	0	0	1	0	0	..	0	0	0	0	0	0	0	..	0	0	0	0
1	1	1	..	1	1	1	1	1	0	0	..	0	0	0	1	0	0	..	0	0	0	0	0	0	0	..	0	0	0	0
1	1	1	..	1	1	1	1	1	0	0	..	0	0	0	1	1	0	..	0	0	0	0	0	0	0	..	0	0	0	0
1	1	1	..	1	1	1	1	1	1	0	..	0	0	0	1	1	0	..	0	0	0	0	0	0	0	..	0	0	0	0
1	1	1	..	1	0	0	0	0	0	0	..	0	0	0	0	0	0	..	0	0	0	0	0	0	0	..	0	1	1	1
1	1	1	..	1	0	0	0	0	0	0	..	0	0	0	0	0	0	..	0	0	0	0	0	0	0	..	0	0	1	1
1	1	1	..	1	1	0	0	0	0	0	..	0	0	0	0	0	0	..	0	0	0	0	0	0	0	..	0	0	1	1
1	1	1	..	1	1	0	0	0	0	0	..	0	0	0	0	0	0	..	0	0	0	0	0	0	0	..	0	0	0	1
1	1	1	..	1	1	1	0	0	0	0	..	0	0	0	0	0	0	..	0	0	0	0	0	0	0	..	0	0	0	1
1	1	1	..	1	1	1	0	0	0	0	..	0	0	0	0	0	0	..	0	0	0	0	0	0	0	..	0	0	0	0

Table 4

X_1	X_2	X_3	..	X_{k-2}	X_{k-1}	X_k	X_{k+1}	X_{k+2}	X_{k+3}	X_{k+4}	..	X_{n-k-2}	X_{n-k-1}	X_{n-k}	X_{n-k+1}	..	X_{n-2}	X_{n-1}	X_n
1	1	1	..	1	1	1	0	0	0	0	..	0	0	0	0	..	0	0	0
0	1	1	..	1	1	1	0	0	0	0	..	0	0	0	0	..	0	0	0
0	1	1	..	1	1	1	1	0	0	0	..	0	0	0	0	..	0	0	0
0	0	1	..	1	1	1	1	0	0	0	..	0	0	0	0	..	0	0	0
0	0	1	..	1	1	1	1	1	0	0	..	0	0	0	0	..	0	0	0
1	1	1	..	1	0	0	0	0	0	0	..	0	0	0	0	..	0	1	1
1	1	1	..	1	0	0	0	0	0	0	..	0	0	0	0	..	0	0	1
1	1	1	..	1	1	0	0	0	0	0	..	0	0	0	0	..	0	0	1
1	1	1	..	1	1	0	0	0	0	0	..	0	0	0	0	..	0	0	0

d. A stuck-on fault at the transistor q_i is equivalent to a stuck-at one fault at line X_i. Therefore the single and the multiple transistor stuck-on faults are detected.

e. Single or multiple stuck-open faults at transistors t_1, t_3 and t_4. Two successive vectors of Table 4 are sufficient to detect this fault. The first vector must have Hamming weight equal to k and the next one equal to k-1. Then the output sequence will be (0, 0) while the expected sequence is (0, 1) and the fault is detected.

f. Single or multiple stuck-open faults at transistors t_2 and t_5. Two successive vectors of Table 4 are sufficient to detect this fault. The first vector should have weight equal to k-1 and the next one equal to k.

g. The stuck-on faults at transistors t_3, t_5. These faults are handled by an n-dominant design similar to that in [9].

h. The stuck-on faults at transistors t_2, t_4. These faults are undetectable but the k-order comparator behavior is unchanged after the occurrence of such a fault. Furthermore, if a stuck-on fault at transistor t_2 or/and t_4 is followed by a detectable fault, the resulting fault is detectable.

i. Single or multiple stuck-at one faults at lines V_{out}, Z_1. A single vector with Hamming weight k is sufficient to test these faults.

j. Single or multiple stuck-at zero faults at lines V_{out}, Z_1. A single vector with Hamming weight k-1 is sufficient to test these faults.

k. Single stuck-at faults at the XOR gates. From Table 3 we can easily see that each XOR gate receives during the test phase all possible test vectors, thus it is tested exhaustively. Besides the stuck at faults, depending on its implementation many other faults can be detected.

l. For any type of faults that affect the CNCI module and change its output sequence 0101..... The reason is that the CNCI module and the shift registers A and B do not share any circuitry.

m. For the faults that affect the shift registers A or/and B and modify the alternation of code words non-code words generated by the shift registers.

IV. Applications

The error detection and correction procedure of the k-EC/d-ED/AUED, k-EC/AUED, k-EC/d-UED and k-EC/d-ED/m-ED codes consists of three steps. The first step is a correction that takes place in the k-EC code part of the received word. The second step is the computation of the check symbols of the k-EC/d-ED/AUED, k-EC/AUED, k-EC/d-UED and k-EC/d-ED/m-ED code corresponding to the corrected part of the received word. The third step is a comparison to find whether the received word and the

corrected one differ in more than k bit positions. We can easily see that a (k+1)-order comparator is suitable for the implementation of the third step.

We have to note that during the normal, fault free operation of the error detection and correction circuit, the received word and the corrected one differ in t bits, $1 \leq t \leq k$, when a correctable error has occurred in the received word and in more than k bits when an only detectable error has occurred. The probability a correctable error to have occurred in the received word is much smaller than the probability the received word to be error free, while the probability an only detectable error to have occurred is extremely small. From the operation of the proposed k-order comparator (Figure 2) we can see that it has static power consumption when the compared words are not identical. Then from the above we conclude that when the proposed k-order comparator is used for the implementation of the above codes it rarely has static power consumption.

In the case that the k+1-order comparator is used for the implementation of a k-EC/d-ED code in the cache tag memory [9] the two operands (the search tag and the accessed tag) that are compared may differ in t positions. We consider the following cases :

- $t = 0$. In this case the search tag and the accessed tag are identical. Then the static power consumption of the (k+1)-order comparator is equal to zero.
- $0 < t \leq k$. In this case an error has occurred in the accessed or the search tag. The probability an error to have occurred in the accessed or search tag is very small thus the (k+1)-order comparator rarely has static power consumption.
- $k < t$. In this case the search tag and the accessed tag correspond to different blocks of main memory. In this case we have to distinct direct mapped caches and f-way set associative caches. In direct mapped caches just one (k+1)-order comparator is used. Since direct mapped caches with cache sizes greater than 8 Kbytes have miss ratio m, $m \leq 6.6\%$ [15, p.421] we conclude that only for the m% of the comparisons we will have static power consumption in the (k+1)-order comparator. In f-way set associative caches f (k+1)-order comparators are used. The usual value of f is 2 or 4. Then for each search f-1 (k+1)-order comparators consume static power, also another (k+1)-order comparator in the m% (in this case $m < 5.4\%$) [15, p.421] of the cases consume power. Therefore, for f-way set associative caches the static power consumption of the (k+1)-order comparators is significant and should be taken into account. Similar comments with respect the static power consumption can be made for the application of the (k+1)-order comparator in broadcast networks.

The above analysis implies that the proposed k-order comparators are suitable for the implementation of the (k-1)-EC/d-ED/AUED, (k-1)-EC/AUED, (k-1)-EC/d-UED, (k-1)-EC/d-ED/m-ED codes or the implementation of a (k-1)-EC/d-ED code in the tag part of a cache memory with direct mapped organization. However for cache memories with f-way set associative organization and broadcast networks k-order comparators with zero static power consumption should be designed. We are currently working to this direction.

References

[1] M. Nicolaidis, "Self-Exercising Checkers for Unified Built-in Self-Test (UBIST)", IEEE Trans. on CAD, Vol. 8, No 3, March 1989.

[2] B. Bose and D.K. Pradhan, "Optimal unidirectional error detecting correcting codes", IEEE Trans. Comput., Vol.C-31, pp.564-568, June 1982.

[3] D.J. Lin and B. Bose, «Theory and Design of t-Error Correcting and d (d>t)-Unidirectional Error Detecting (t-EC/d-UED) Codes», IEEE Trans. Comput., April 1988, pp. 433-439.

[4] T.R.N. Rao, E. Fujiwara, "Error-Control coding for computer systems." Prentice-Hall International.

[5] M. Blaum and H.V. Tilborg, "On t-Error Correcting/All Unidirectional Error Detecting Codes", IEEE Trans. Comp. Nov. 1989, pp. 1493-1501.

[6] D. Nikolos, "Theory and Design of t-Error Correcting/d-Error Detecting (t<d) and All Unidirectional Error Detecting Codes." IEEE Trans. Comp., Feb. 1991, pp.132-142.

[7] D. Nikolos, and A. Krokos, "Theory and Design of t-Error Correcting, k-Error Detecting and d-Unidirectional Error Detecting Codes with d>k>t.", IEEE Trans. on Comput., April 1992, pp. 411-419.

[8] Jien-Chung Lo and Eiji Fujiwara, "Probability to Achieve TSC Goal", IEEE Trans. on Comput., April 1996.

[9] H.T. Vergos and D. Nikolos, "Efficient Fault Tolerant Cache Memory Design", Micropr. and Microprogr., The Euromicro Journal, 41 (1995) pp. 153-169.

[10] W. C. Carter and P. F. Schneider, "Design of dynamically checked computers", in Proc. 4th Cong. IFIP, Edinburgh, Scotland, vol. 2, pp. 878-883, Aug. 5-10, 1968.

[11] D. A. Anderson, "Design of self-checking networks using coding techniques", Coord. Sci. Lab., Univ. Illinois, Urbana, IL, Tech. Rep. R-527, 1971

[12] M. Nicolaidis and B. Courtois, "Strongly Code Disjoint Checkers", IEEE Trans. Comput., June 1988.

[13] Neil H. E. Weste, Kamran Eshraghian, "Principles of CMOS VLSI Design, A systems Perspective" , 2nd ed., Addison Wesley.

[14] V.G. Oklobdzija and P.G. Konijanic, "On testability of CMOS-domino logic, " in Proc. 14th Int. Symp. Fault-Tolerant Comput., June 1984.

[15] J.L. Hennessy & D.A. Patterson, "Computer Architecture a Quantitive Approach", Morgan Kaufmann Publishers Inc.

[16] C. Metra, Jien C. Lo "Compact and High Speed Berger Code Checker" 2nd IEEE Int. On-Line Testing Workshop, Biarritz, France July 8-10, 1996, pp. 144-149.

[17] X. Kavousianos, D. Nikolos "Self-Exercising k-order Comparators: Design and Applications" CTI TR 97.1.9