Efficient Design of CMOS TSC Checkers

Anita Biddappa, Manjunath K. Shamanna, Gary Maki and Sterling Whitaker
NASA Space Engineering Research Center
for VLSI System Design
University of Idaho
Moscow, Idaho 83843

Abstract - **This paper considers the design of an efficient, robustly testable CMOS Totally Self-Checking (TSC) Checker for k-out-of-$2k$ codes. Most existing implementations use primitive gates and assume the single stuck-at fault model. The self-testing property has been found to fail for CMOS TSC checkers under the stuck-open fault model especially due to timing skews and arbitrary delays in the circuit. A new four level design using CMOS primitive gates (NAND, NOR, INVERTERS) is presented, which retains its properties under the stuck-open fault model. Additionally this method offers an impressive reduction ($> 70\%$) in gate count, gate inputs and test set size when compared to the existing method. This implementation is easily realizable and is based on Anderson's technique[1]. A thorough comparative study has been made on the proposed implementation and Kundu's[8] implementation and the results indicate that the proposed one is better than Kundu's[8] in all respects for k-out-of-$2k$ codes.**

1 Introduction

Totally Self-Checking (TSC) checkers are circuits which detect errors concurrently under normal operation. They are used to check the validity of various codes of which m-out-of-n codes are very important. An m-out-of-n code is one in which all valid code words have exactly m ones and $n - m$ zeros. An k-out-of-$2k$ code is a special case of m-out-of-n code and is the least redundant code for error detection[11] and hence is of particular interest. This paper deals only with k-out-of-$2k$ codes.

Carter and Schnieder[4] were the first to propose Self-Checking circuits. Anderson and Metze[1] extended this concept to develop TSC checkers for k-out-of-$2k$ codes based on the conventional stuck-at fault model. The conventional stuck-at fault model is inadequate in defining certain failures in CMOS circuits. Hence, the fault model has been augmented to include stuck-open and stuck-on faults[14,3,5]. Furthermore, it has been shown that tests for stuck-open faults in CMOS combinational logic circuits could potentially be invalidated by arbitrary delays or due to timing skews in the input changes[7,12,8]. Tests which do not suffer from this potential invalidation are called *Robust Tests.*

Until now, only two approaches have been reported to realize robustly testable CMOS TSC checkers[6,8]. The approach followed by Jha and Abraham[6] has been found to be

inadequate[9]. Kundu and Reddy[8] have proposed a four-level realization of robust CMOS TSC checkers for k-out-of-$2k$, k-out-of-$2k+1$, $k+1$-out-of-$2k+1$ and $k+1$-out-of-$2k+1$ codes. However, a robustly testable complex gate realization of TSC checker is yet to be reported. The main contribution of this paper is the design of a four level robustly testable CMOS TSC implementation based on Anderson's technique for k-out-of-$2k$ codes. This offers considerable savings ($> 70\%$) in gate count, gate inputs and test set size over the existing method.

2 Notations and Definitions

This section summarizes the notations and definitions used in this paper. The notation used by Kundu and Reddy[8] and Anderson and Metze[1] will also be followed here. A few of the notations are as follows:

A:	set of all valid m-out-of-$2m$ code words.
vertex:	binary word.
n-vertex:	binary word with n ones.
f, g:	outputs of the TSC checker.
f_1:	subset of A such that f_1=1 and g_1=0 for any input from f.
g_1:	subset of A such that f_1=0 and g_1=1 for any input from g.
$< T_1, T_2 >$:	two pattern Stuck-open fault test.
T_1 :	initializing input for the stuck-open fault test.
T_2 :	test input for the stuck-open fault.
$T(k_a > i)$:	majority function which is true if the number of ones in the vertex k_a is greater than i.

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

Definition 2 *A circuit is fault secure if for every fault from the fault set, the circuit never produces an incorrect code output for every code input.*

Definition 3 *A circuit is totally self checking if it is both self testing and fault secure.*

Definition 4 *A binary n-tuple is said to 1-cover (0-cover) another binary n-tuple if the former has ones (zeros) in every position the latter has ones (zeros).*

Definition 5 *A vertex X is a true vertex of a function F iff F(X)=1.*

Definition 6 *A true vertex V_p of a function F is called a minimal true vertex iff there does not exist another true vertex Y of the function, $Y \neq V_p$ such that V_p 1-covers Y.*

Definition 7 *A 0-cofactor (1-cofactor) of order n of a product term of uncomplemented variables is obtained by substituting n zeros (ones) for n ones (zeros) in the product.*

If a minimal true vertex V_p has n literals and if a 1 at position k of V_p is changed to a 0 to obtain a 0-cofactor of V_p it is denoted by $V_p(k)$.

Definition 8 *A 0-cofactor (1-cofactor) is a false (true) vertex.*

Definition 9 *A hazard is defined as the appearance of an undesired logic value (transient) at the output of a circuit which is produced due to one or more changes at the inputs of the circuit.*

3 TSC Checkers

Smith[13] and Anderson and Metze[1] have proposed the following conventions for TSC checkers;

1. All invalid code words of weight less than m are mapped to the checker outputs: $f = g = 0$.
2. All invalid code words of weight greater than m are mapped to the checker outputs: $f = g = 1$.
3. All valid code words of weight equal to m are mapped to the checker outputs: $f = 1$, $g = 0$ or $f = 0$, $g = 1$.

Let F be a minimal sum of products expression realized by a two level NAND-NAND logic network N, where the first level gates correspond to the minterms of F. Then we have the following properties.

Property 1: *If a minimal true vertex V_p is input to N, then the output of the circuit and the outputs of all first level NAND's except the gate covering V_p are 1.*

Property 2: *If $V_p(k)$, a 0-cofactor, is input to N, then the outputs of all first level gates are 1 and that of the second level gate is 0.*

Property 3: *If a two pattern input sequence $< V_p, V_p(k) >$ or $< V_p(k), V_p >$ is applied to N, then only one input to any gate changes. Furthermore, the outputs of all first level gates except the one covering V_p are 1 (hazard-free) for both inputs in the two pattern sequence.*

Property 4: *If a two pattern input sequence consisting of two 0-cofactors of different orders is presented to N, such that one 0-cofactor covers the other, then the output of every first level NAND gate and the second level NAND gate is 1 (hazard-free) and 0 (hazard-free) respectively for both inputs in the two pattern sequence.*

Property 5: *If a two pattern input sequence consisting of a true vertex and a 1-cofactor or two 1-cofactors of different orders, such that one 1-covers the other is applied to N, then there exists a gate whose inputs are all 1's for both inputs in the two pattern sequence*

implying that the output of the second level NAND is always 1 (hazard-free) for both the inputs.

3.1 Kundu's Technique

This technique yields a four level implementation which is robustly testable. Here the k-out-of-$2k$ code is partitioned into subsets f_1 and g_1 according to Smith's Criterion[13]. The checker outputs f and g are then expanded with respect to an input variable x_i, chosen arbitrarily such that,

$$f = x_i f_{1i} + \hat{f}_{1i}$$

$$g = x_i f_{2i} + \hat{f}_{2i}$$

where f_{1i}, f_{2i}, $\hat{f}_{1i}$, $\hat{f}_{2i}$ are functions of the variable x_i through x_n excluding x_i. The functions $\hat{f}_{1i}$, $\hat{f}_{2i}$, and f_{1i}, $f_{2i}i$ are realized by NAND-NAND and NOR-NOR circuits respectively.

This design is hard to realize for large values of k and also necessitates the use of a large number of gates with very high fan-in.

3.2 Anderson's Technique

To design a k-out-of-$2k$ TSC, the $2k$ input bits are first divided into two groups of equal length ($n_a = n_b = k$). The checker output functions f and g are realized as follows:

$$f = \sum_{i=0}^{k} T(k_a \geq i) T(k_b \geq k - i) \mid_{i=odd}$$

$$g = \sum_{i=0}^{k} T(k_a \geq i) T(k_b \geq k - i) \mid_{i=even}$$

where k_a and k_b will refer to the number of 1's occurring in each group. This technique was limited to two level realization using AND and OR gates and was developed under the stuck-at fault assumption. Under the extended fault model the self-testing property, which is a necessary condition for a circuit to retain its TSC properties, is difficult to satisfy for TSC CMOS circuits. In fact, it was shown by Manthani and Reddy[10] that a two level realization of Anderson and Metze's TSC checker[1] is not robustly testable. We now proceed to present a four level CMOS gate implementation based on this technique.

4 CMOS Checker Implementation

First we present a robustly testable CMOS checker design using primitive gates.

Procedure 1

1. Partition the $2k$ input bits $(a_1, \ldots, a_k, \ldots, a_{2k})$ into two groups of equal length $A_1 = \{a_1, \ldots, a_k\}$ and $A_2 = \{a_{k+1}, \ldots, a_{2k}\}$.

2. Find f and g using the equations:-

$$f = \sum_{i=0}^{k} T(k_a \geq i)T(k_b \geq k-i) \mid_{i=odd}$$

$$g = \sum_{i=0}^{k} T(k_a \geq i)T(k_b \geq k-i) \mid_{i=even}$$

where k_a and k_b refers to the number of one's in A_1 and A_2 respectively.

3. Implement both f and g in NAND-NAND form with each individual majority function implemented in two level NAND-NAND logic except the terms $T(k_a \geq 1)$ or $T(k_b \geq 1)$ which are to be realized using NOR logic.

The four level implementation obtained using Procedure 1 is robustly testable. If the expressions for f and g in the above procedure are now expanded in the form of a true sum of products expression to yield a two level realization, then the TSC checker fails to be self-checking under stuck-open faults.

Testing Strategy

Consider Figure 1. To test for stuck-open faults at the inputs of the circuit block implementing $T(k_a \geq i)$, it is to ensured that $T(k_b \geq k-i)$ should be 1 for $< T_1, T_2 >$. Under these conditions, $T(k_a \geq p)|_{(p>i)} = 0$ and $T(k_b \geq q)|_{(q>k-i)} = 0$. Hence, all the other inputs of the NAND gate (N_4), excepting the path under test, are 1 (hazardfree) during $< T_1, T_2 >$ ensuring robust testing. Similarly for testing stuck-open faults at the inputs of $T(k_b \geq k-i)$, $T(k_a \geq i)$ should be 1 for $< T_1, T_2 >$. This implies that $T(k_a \geq p)|_{(p>i)}$ and $T(k_b \geq q)|_{(q>k-i)}$ are 0 during $< T_1, T_2 >$ which ensures that all the inputs of the NAND gate (N_4), except the path under test, are 1 (transientfree). Thus the tests are robust in nature.

Theorem 1 *The four level circuit designed using Procedure 1 is robustly testable for all single stuck-open and stuck-at faults.*

Proof: Consider the expressions for f and g:

$$f = T(k_a \geq 1)T(k_b \geq k-1) + T(k_a \geq 3)T(k_b \geq k-3) + \ldots + T(k_a \geq k-1)T(k_b \geq 1)$$

..... if k is even.

$$f = T(k_a \geq 1)T(k_b \geq k-1) + T(k_a \geq 3)T(k_b \geq k-3) + \ldots + T(k_a \geq k)T(k_b \geq 0)$$

..... if k is odd.

$$g = T(k_a \geq 0)T(k_b \geq k) + T(k_a \geq 2)T(k_b \geq k-2) + \ldots + T(k_a \geq k)T(k_b \geq 0)$$

..... if k is even.

$$g = T(k_a \geq 0)T(k_b \geq k) + T(k_a \geq 2)T(k_b \geq k-2) + \ldots + T(k_a \geq k-1)T(k_b \geq 1)$$

..... if k is odd.

Each of the different implementations of f and g when k is even or odd will be considered separately for proving the circuit's robustness during testing. Chandramouli's[3] procedure for testing stuck-open faults is followed here. This envisages the application of a sequence of a pair of test vectors to detect the stuck-open faults in CMOS logic circuits. The first test vector initializes the output of the faulty gate and the second sensitizes and propagates the fault to the output. Furthermore, Chandramouli[3] has shown that the tests for all input stuck-open (ISOP) faults and output stuck-open faults (OSOP) at all primary gate inputs will test cover all the stuck-open faults in an Non-Internal Reconvergent fan-out circuit. Also, the test vectors for detecting the ISOP and OSOP faults are identical but reversed in their order of application. It should also be noted that, since the implementation is four level, when testing a primary input fault all the four gate levels must be sensitized.

4.1 Realization of g when k is even

Consider the realization of g with k even. Each majority function, $T(k_a \geq i) \mid_{i \neq 1,k}$ and $T(k_b \geq i) \mid_{i \neq 1,k}$, is realized by two levels of NAND gates. The third level NAND gate realizes $T(k_a \geq i)T(k_b \geq k-i)$ and the fourth level NAND gate realizes g. To simplify the proof, the whole circuit is partitioned into a finite number of subcircuits and then each one of these subcircuits is shown to be robustly testable.

Circuit corresponding to any $T(k_a \geq i)T(k_b \geq k-i)$

Consider an arbitrary lead y of any first level NAND gate corresponding to $T(k_a \geq i)$. Each first level NAND corresponds to a true vertex of the unate function $T(k_a \geq i)$. Select a two pattern test $< V_p, V_p(y) >$. The test vector V_p is chosen such that all the i leads of the gate are 1's during T_1 implying that the weight of n_a is i and that of n_b is $k-i$. $V_p(y)$ applies a zero to the input y while all other inputs of this gate remain at 1, so that the weights of n_a and n_b are $i-1$ and $k-i+1$ respectively. According to Property 3, the second level NAND gate is sensitized robustly (all the sensitized leads remain at one during T_1 and T_2). Since V_p and $V_p(y)$ are 0-cofactors of the majority functions $T(k_a \geq p) \mid_{p \geq i+2}$, all the other inputs feeding the fourth level NAND are 1 during the test sequence $< V_p, V_p(y) >$ and are hazard-free. Also the output of the second level NAND, $T(k_b \geq k-i)$, is 1 and is hazard-free. Hence, $< V_p, V_p(y) >$ is a robust test for ISOP on the lead y. Similarly the reverse sequence $< V_p(y), V_p >$ forms a robust test for OSOP fault corresponding to the lead y. In a similar manner, it can be shown that any input/output stuck-open fault corresponding to the first level NAND gates of the majority functions $T(k_b \geq k-i)$ can be robustly tested.

Circuit corresponding to $T(k_b \geq k)$ *term*

The term $T(k_b \geq k)$ corresponds to a single k input NAND from which an arbitrary lead u is selected. The test pattern $< V_p, V_p(u) >$ is then applied. The test vectors are such that weights of n_a are 0 and 1 and that of n_b are k and k-1 during T_1 and T_2 respectively.

Since the weight of n_a is never greater than 1, during the test sequence the outputs of all second level NANDs corresponding to $T(k_a \geq p) \mid_{p \geq 2}$ are 0 and hence the outputs

of all third level NAND's are 1 (hazard-free). Also the output of the gate corresponding to $T(k_a \geq k)$ is a 1 (hazard-free) during the test sequence. Since no transients occur in the circuit, the test pattern $< V_p, V_p(u) >$ is a robust ISOP test for the lead u. Similarly, $< V_p(u), V_p >$ is a robust OSOP fault test corresponding to lead u. On the same lines, it can be shown that the input gate implementing $T(k_a \geq k)$ is also robustly testable.

4.2 Realization of g when k is odd

This implementation is similar to the implementation of g when k is even. As usual, the terms $T(k_a \geq k)$ and $T(k_a, T(k_b \geq i)\ |_{i \neq 1}$ are realized by NAND gates except $T(k_b \geq 1)$ which is realized by a k input NOR gate.

Circuits corresponding to any $T(k_a \geq i)T(k_b \geq k-i)\ |_{i \neq 1}$, *and* $T(k_b \geq k)|_{k \neq 1}$
The arguments when k is even are also valid here. Hence, these circuits can be robustly tested.

Circuit corresponding to $T(k_b \geq 1)$
Consider an arbitrary lead m of the NOR gate corresponding to $T(k_b \geq 1)$. A two pattern test $< U_s, U_s(m) >$ is selected such that U_s applies 0 to all k leads of the NOR gate such that weight of n_b is 0 and $U_s(m)$ applies a 1 to the lead under question and 0 to the rest of the leads. Since the weight of n_b is 0 and 1 in T_1 and T_2 respectively, the output of all second level NAND gates corresponding to all $T(k_b \geq p)\ |_{p \geq 2}$ is always 0. Hence the outputs of all third level NAND gates except the one corresponding to the NOR gate under consideration are 1 (hazard-free) during the test sequence. Also the output of the NAND gate corresponding to $T(k_b \geq k)$ is 1. Hence, the test sequences $< U_s(m), U_s >$ do not suffer from test invalidation. So, the circuit is robustly testable.

4.3 Realization of f when k is odd

This case is similar to that of g when k is odd. Hence using arguments similar to the one where k is odd in the realization of g, it can be shown that the circuit is robustly testable.

4.3.1 Realization of f when k is even

The majority functions $T(k_a \geq 1)$ and $T(k_b \geq 1)$ are implemented by NOR gates while the other majority functions are implemented by two level NAND logic.

Circuit corresponding to any $T(k_a \geq i)T(k_b \geq k-i)\ |_{i \neq 1, k-1}$
The arguments corresponding to the case when k is even in the realization of g are valid here. Hence this circuit is robustly testable.

Circuit corresponding to $T(k_b \geq 1)$
The arguments corresponding to the realization of g when k is odd are valid here. Hence this circuit is robustly testable.

Circuit corresponding to $T(k_a \geq 1)$

Using arguments similar to the case of $T(k_b \geq 1)$, it can be easily shown that this circuit is robustly testable.

QED.

The following example will demonstrate the design of the proposed TSC checker.

Example 1 *Design a CMOS TSC checker for a 3-out-of-6 code using Procedure 1*

The six input bits are partitioned into two groups: $A_1 = (a_1, a_2, a_3)$ and $A_2 = (a_4, a_5, a_6)$ where $a_1, a_2, \ldots, a_6$ represent the input code bits. The expressions for f and g are given by:-

$$\begin{aligned} f &= T(k_a \geq 1)T(k_b \geq 2) + T(k_a \geq 3)T(k_b \geq 0) \\ &= (a_1 + a_2 + a_3)(a_4a_5 + a_4a_6 + a_5a_6) + (a_1a_2a_3) \\ g &= T(k_a \geq 0)T(k_b \geq 3) + T(k_a \geq 2)T(k_b \geq 1) \\ &= (a_4a_5a_6) + (a_1a_2 + a_1a_3 + a_2a_3)(a_4 + a_5 + a_6) \end{aligned}$$

The majority functions $T(k_a \geq 1)$ and $T(k_b \geq 1)$ in the realizations of f and g respectively are implemented using NOR logic while the remaining majority functions are implemented using NAND logic as shown in Figures 1 and 2. This four level implementation can be robustly tested using the guidelines presented in [3].

5 Comparisons

As mentioned before, there is only one method[8] currently available, for the design of robustly testable CMOS TSC checkers wherein the cases of k-out-of-$2k$, $k-1$-out-of-$2k+1$, $k+1$-out-of-$2k+1$ and k-out-of-$2k+1$ codes have been covered. The proposed method covers the case of k-out-of-2k codes. A comprehensive study of the number of gates used by our method and Kundu's method[8] has been made and general relations have been established which are shown in Tables 1 through 4.

Table 5 illustrates the comparison between the proposed and Kundu's method[8] for k=2, 3, 4, 5 and 6. The comparisons have been made with respect to gate inputs and test vector pairs. Figures 3 and 4 illustrate the percentage savings in test vector pairs and gate inputs respectively with increasing values of k. These graphs show an almost exponential nature with about 32 % and 60 % savings in gate inputs and test vector pairs for even the trivial case of $k = 2$ and increases to well over 90 % for cases $k \geq 6$. The tremendous increase in gate inputs is not only a result of increased number of gates but also gates with high fan-in index. For a typical case of 5-out-of-10 code TSC checker, Kundu and Reddy's method[8] uses 258 gates with four gates having a fan-in of over 60. On the contrary, the proposed design uses 102 gates with 3 gates involving a maximum fan-in of 10.

Testing of the circuits as designed in [8] is complex and offers fewer choices of test vectors for testing a stuck-open fault, where as the proposed one is easily testable and offers more choices of test vectors for testing the same fault. Also, the probability that a fault is tested sooner is high. Assuming a random occurrence of test vectors, the probability

distribution that the circuit is tested is multinomial in nature . It can be easily shown that the probability of the proposed realization being tested completely in a shorter period of time is considerably more compared to Kundu and Reddy's realization[8]. Also, the waiting time is considerably less.

The proposed design is very simple and has been acknowledged by various researchers in this field. The partition of all valid code words into f_1 and f_2 as described in [8] is a bit tedious for larger values of k. Additionally converting f_{1i}, f_{2i} into the product of sums form from the sum of products form is highly cumbersome and laborious for larger values of k (> 4). No such Boolean simplification is needed for our realization. Also, the proposed design is very simple and is based on Anderson and Metze's technique[1], the simplicity of which has been acknowledged by various researchers in this field.

As opposed to our method, the amount of fan-in of the second level NOR and NAND gates in Kundu's method[8] is considerable and increases dramatically with k. Furthermore, the proposed method offers a large reduction in chip area because of the considerably fewer number of gates and gate inputs.

6 Conclusions

In this paper we have considered the problem of designing robustly testable CMOS TSC checkers for k-out-of-$2k$ codes. The robust testability criterion under the extended fault model has been achieved by following a four level implementation rather than the regular two level implementation. A comparison between the proposed method and the presently available method[8] has been carried out and results tabulated. Savings of over 90 % in the number of gate inputs and test vector pairs have been achieved for even simple cases (k=6). The savings in the number of gate inputs and test vector pairs appear to be exponentially increasing reaching over 90 % for even small values of k ($k \geq 6$). The proposed design not only offers considerable savings in gates, test vectors, chip area, etc. as compared to the presently available method, but it also offers the benefit of simplicity of design.

References

[1] Anderson, D. A., and Metze, G.," Design of Totally Self-Checking Check Circuits for m-out-of-n Codes", *IEEE Transactions on Computers*, vol. 22, 1973, pp. 263-269.

[2] Bose, B., and Lin, D. J., "PLA Implementation of k-out-of-n Code TSC Checker", *IEEE Transactions on Computers*, vol. 6, 1984, pp. 43-68.

[3] Chandramouli, R., " On Testing Stuck-Open Faults", *Proceedings of 13th Fault Tolerant Computing Symposium*, 1983, pp. 258-265.

[4] Carter, W. C., " Design of dyanamically checked Computers", *IFIP*, vol. 2, 1968, pp. 878-883.

[5] El-Ziq, Y. M., and Cloutier, R. J., "Functional-level Test Generation for Stuck-open Faults in CMOS VLSI", *International Test Conference*, 1981, pp. 536-546.

[6] Jha, N. K., and Abraham, J. A., "Totally Self-Checking CMOS Circuits Using a Hybrid Realization", *International Symposium on Fault Tolerant Computing*, 1985, pp. 154-158.

[7] Jain, S. K., and Agrawal, V. D., "Test Generation for MOS Circuits Using D-Algorithm", *Proceedings of the Twentieth Design Automation Conference*, 1983, pp. 64-70.

[8] Kundu, S., and Reddy, S. M., "Design of TSC Checkers for Implementation in CMOS Technology", *International Conference on Computer Design*, 1989, pp. 116-119.

[9] Kundu, S., and Reddy, S. M., "On the Design of TSC CMOS Combinational Logic Circuits", *International Test Conference*, 1986, pp. 486-499.

[10] Manthani, S. R., and Reddy, S. M., "On CMOS Totally Self-Checking Circuits", *International Test Conference*, 1984, pp. 866-877.

[11] Paschalis, A. M., Nikolos, D., and Halastsis, C., "Efficient Modular Design of TSC Checkers for M-out-of-2M Codes", *IEEE Transactions on Computers*, vol. 3, 1988, pp. 301-309.

[12] Reddy, S. M., Reddy, M. K., and Kuhl, J. G., "On Testable Design for CMOS Logic Circuits", *Proceedings of 1983 International Test Conference*, 1983, pp. 435-445.

[13] Smith, J. E., "The design of totally self-checking check circuits for a class of unordered codes", *Journal of Design Automation and Fault-Tolerant Computing*, vol. 1, 1977, pp. 321-342.

[14] Wadsack, R. L., "Fault Modeling and Logic Simulation of CMOS and MOS Integrated Circuits", *Bell System Technical Journal*, vol. 57, 1978, pp. 1449-1474.

k	Gate inputs		Test vector pairs	
	Kundu's	Proposed	Kundu's	Proposed
2	24	15	30	12
3	81	40	103	34
4	344	99	415	88
5	1497	226	1749	214
6	6230	515	7364	500

k	% Savings in Gate inputs	% Savings in Test vector pairs
2	31.8%	60%
3	50.6%	67%
4	71.2%	78.8%
5	84.9%	87.8%
6	91.7%	93.2%

Table 1: Comparison of the Proposed method and Kundu's method

Gate type	*Inputs*	*Number of gates*
NOR	k	$1 + \sum^{k} C_r{}^{k-1}C_{k-r}\|_{2\leq r\leq k}$
NOR	k-1	1
NOR	$1 + \sum^{k} C_r{}^{k-1}C_{k-r}\|_{r=even}^{2\leq r\leq k}$	1
NOR	$1 + \sum^{k} C_r{}^{k-1}C_{k-r}\|_{r=odd}^{3\leq r\leq k}$	1
NAND	$(^{2k}C_k/2)$ -1 - $\sum^{k} C_r{}^{k-1}C_{k-r}\|_{2\leq r\leq k-2}$	1
NAND	k	$(^{2k}C_k/2)$
NAND	n	1
NAND	2	4
Invertors		2

Table 2: Hardware requirements for Kundu's Realization when k is even

Gate type	*Inputs*	*Number of gates*
NOR	k	$1+\sum^{k} C_r{}^{k-1}C_{k-r}\|^{2\leq r\leq k}$
NOR	k-1	1
NOR	$\sum^{k} C_r{}^{k-1}C_{k-r}\|^{2\leq r\leq k}_{r=even}$	1
NOR	$2 + \sum^{k} C_r{}^{k-1}C_{k-r}\|^{3\leq r\leq k}_{r=odd}$	1
NAND	$1 + \sum^{k} C_r{}^{k-1}C_{k-r}\|^{1\leq r< k}_{r=odd}$	1
NAND	$({}^{2k}C_k/2)$-1 - $\sum^{k} C_r{}^{k-1}C_{k-r}\|^{1\leq r< k}_{r=odd}$	1
NAND	k	m=$({}^{2k}C_k/2)$
NAND	$({}^{2k}C_k/2)$	1
NAND	2	4
Invertors		2

Table 3: Hardware requirements for Kundu's Realization when k is odd

Gate type	*Inputs*	*Number of gates*
NAND	k	4
NAND	$r\|_{r=even}^{2\leq r<k}$	${}^{k}C_{r}$
NAND	$r\|_{r=odd}^{3\leq r<k}$	${}^{k}C_{r}$
NAND	${}^{k}C_{r}$	4
NAND	(k-r)	$2({}^{k}C_{r})$
NAND	2	$(k-1)$
NAND	$k/2$ +1	1
NAND	k -1	2k
NAND	$k/2$	1
NOR	k	2
Invertors		2

Table 4: Hardware requirements for Proposed Realization when k is even

Gate type	*Inputs*	*Number of gates*
NAND	k	4
NAND	k-1	2k
NAND	$r\|^{1\leq r<k}$	$({}^kC_r)$
NAND	${}^kC_r$	4
NAND	(k-r)	$2({}^kC_r)$
NAND	2	$(k-1)$
NAND	$(k+1)/2$	1
NAND	$(k)/2$	1
NOR	k	2
Invertors		2

Table 5: Hardware requirements for Proposed Realization when k is odd

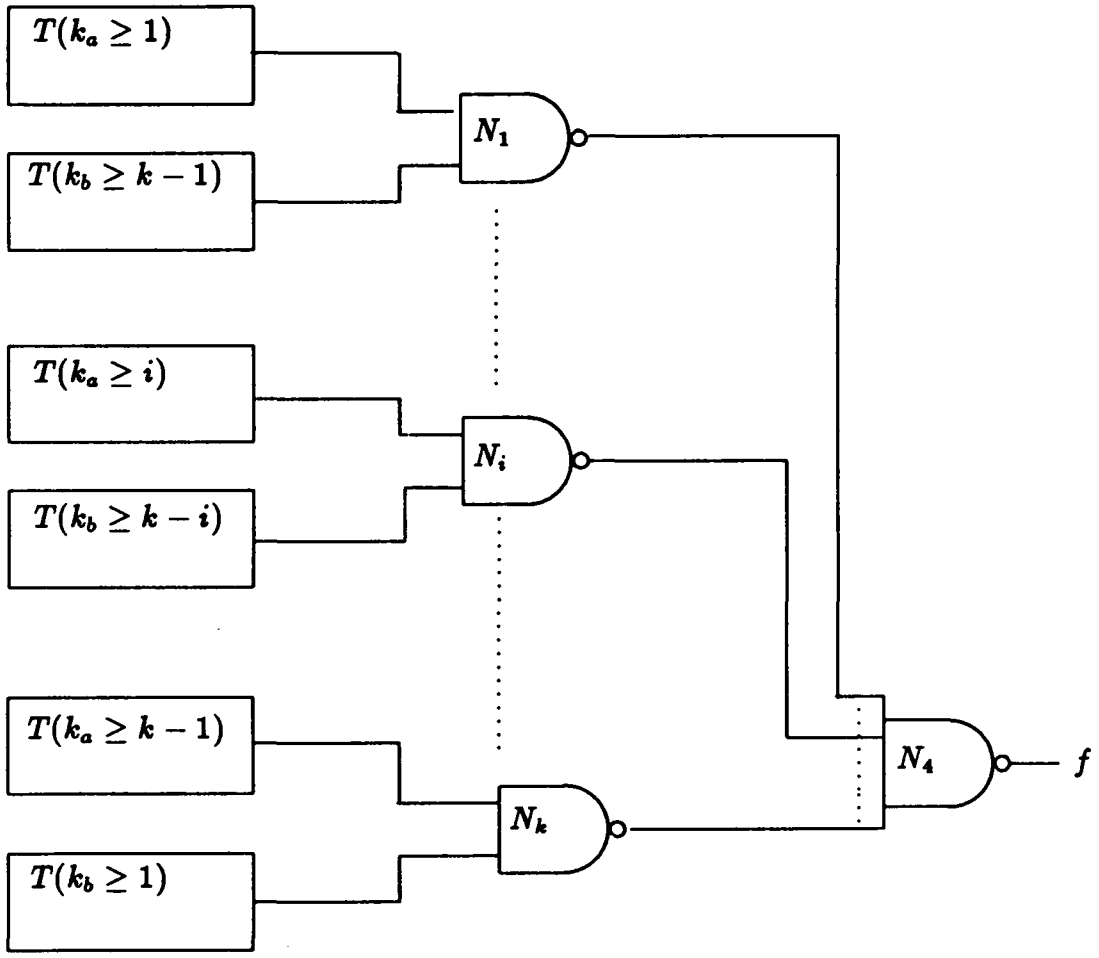

Figure 1: Illustration of the General Testing Strategy

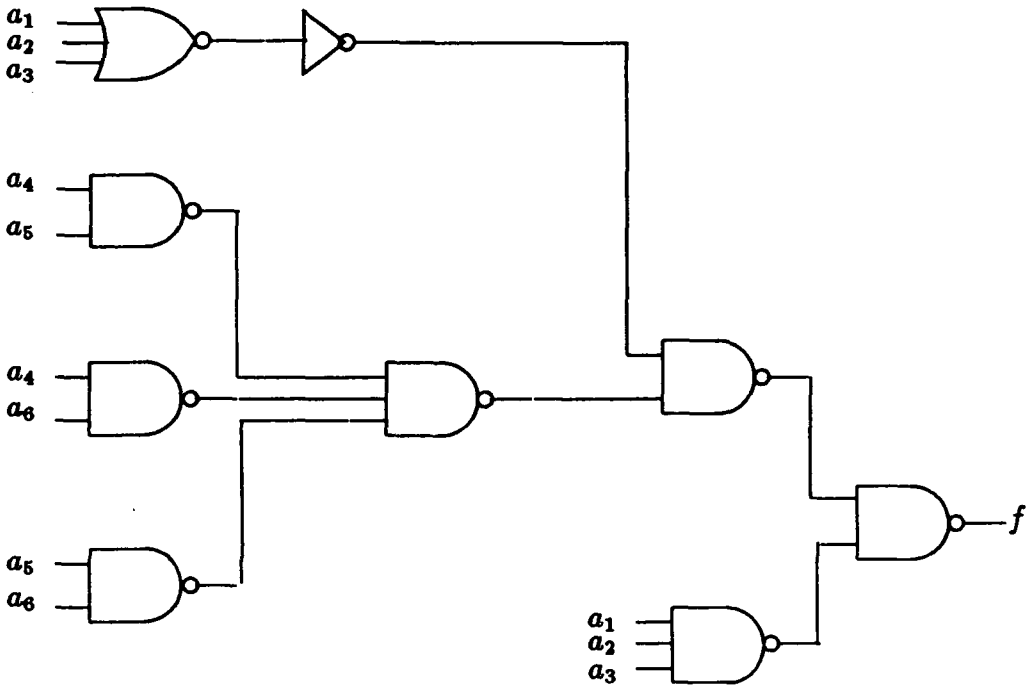

Figure 2: TSC Checker output f for Example 1

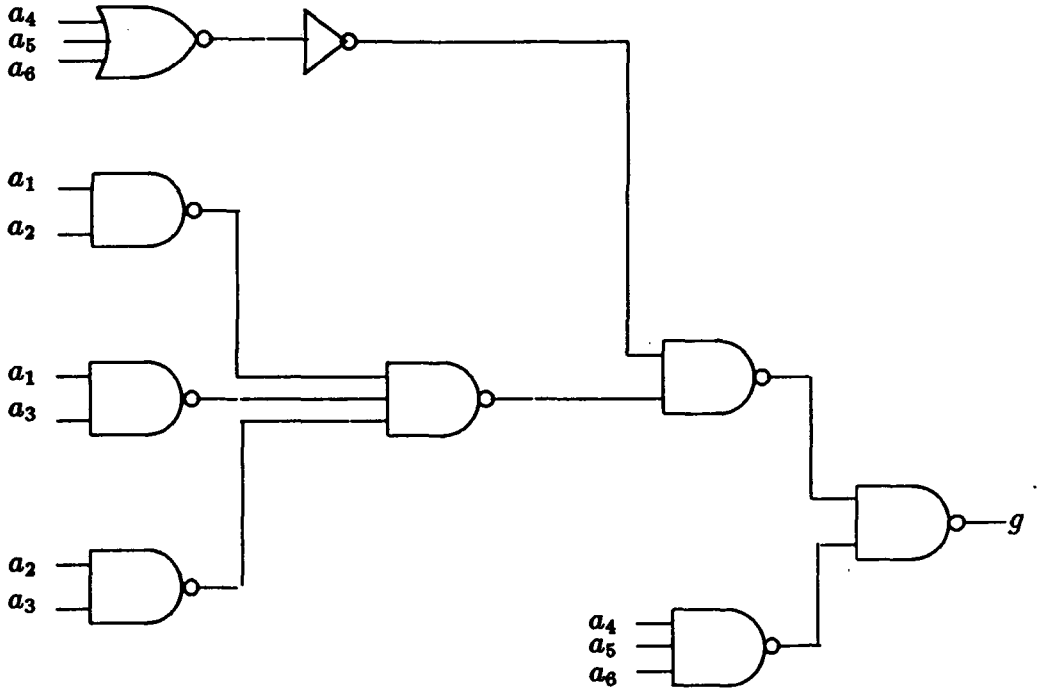

Figure 3: TSC Checker output g for Example 1