TAIR: Testability Analysis by Implication Reasoning

Shih-Chieh Chang, Wen-Ben Jone, and Shi-Sen Chang
Department of Computer Science and Information Engineering National Chung Cheng University.Chiyai, Taiwan 621, Republic of China.

Abstract

To predict the difficulty of testing a wire stuck-at fault, testability analysis algorithms provide an estimated testability value by computing controllability and observability. In most common previous work such as COP [4] and SCOAP [10], signal correlation between controllability and observability is not well handled. As a result, the estimated values can be quite inaccurate. On the other hand, some previous work [16][17][19][24] can take into account signal correlation but may require more CPU time. This paper discusses an efficient method for testability analysis improvement. Our algorithm starts with results obtained from conventional testability analysis such as COP [4]. For each stuck-at fault, we gradually refine these results by recursively applying some simple signal correlation rules. Experimental results show that, with reasonable run-time overhead, significant improvement for testability analysis can be achieved.

1 Introduction

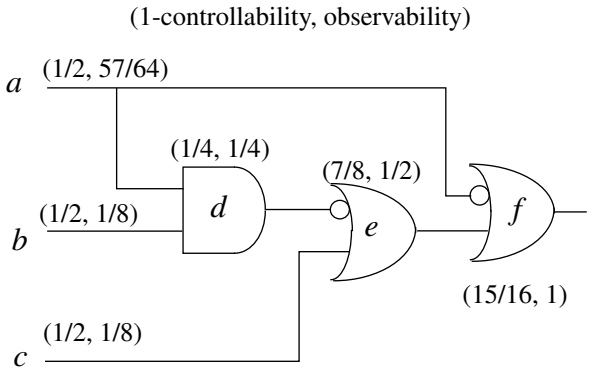
The objective of testability measurement is to predict the difficulty for testing a node or a wire. A good measurement can give an early warning about testing problems so as to provide guidance in improving the testability of a circuit by adding test circuits [6] [20] [22] [25]. In addition, the testability measurement can also provide information for random test length analysis.

Normally, testability measurement assigns each node a testability value by computing the node's controllability and observability. The computation can be done by two sweep of the circuit under test (CUT) with some rules. Since only approximate values are provided, testability measurement is less accurate than the time-consuming test generation and fault simulation processes. As a result, for test-

ability analysis to be useful, its run time must be very fast. Usually, the analysis needs to be linear or almost linear to a circuit size.

There has been research [2] [4] [5] [7] [8] [10] [11] [16] [17] [19] [23] [24] in the area of testability analysis. The algorithm COP [4] computes a signal probability (controllability) value for each node from primary inputs to primary outputs using a set of formulae. These signal probability values are then used to derive the observability of each node by some rules. For the example in Fig. 1, COP estimates the 1-controllability of node *d* to be *1/4* and the observability to be *1/4*. The testability for wire *d->e* stuck-at-1 fault is calculated to be *(1/4) * {1 - (1/4)} = 3/16*. Due to the reconvergent fanout problem, signal probabilities computed by COP are generally inaccurate. The algorithm PREDICT [24], on the other hand, tries to improve signal probabilities using the concept of super gates. This results in much higher complexity with more precise testability estimation. The testability analysis tool SCOAP [10] does not provide probability-like values; instead, it gives a numeric number which represents the difficulty for testing a node. Without computing the exact value for each signal probability, the cutting algorithm [19] calculates the lower and upper bounds for each signal probability by cutting all multiple fanout branches in the CUT. The Parker-McCluskey algorithm [16] manipulates the symbolic expression for each output to compute both signal and detection probabilities, though the equations may soon become too complex to analyze for large circuits. The improved cutting algorithm [17] combines the cutting algorithm and the Parker-McCluskey algorithm. The algorithm chooses a subset of fanout branches for cutting and others for symbolic manipulation. Both algorithms [19] [17] are designed to compute signal probabilities which can be further modified to compute the lower bounds of testability measurements by adding an auxiliary AND gate. The AND-gate output will be 1 if the target fault is sensitized and propagated by a pre-selected path. The signal correlation between controllability and observability can be considered for detection probability analysis in [16][17][19][24].

In the example of Fig. 1, although the controllability values are exact for all the nodes except *f*, all testability measurements (detection probabilities) are **incorrect**. This is due to the ignorance of signal correlation between controllability and observability. In addition, two redundant faults, *a->d* and *a->f* stuck-at-1 faults, are both assigned non-zero values. In this paper, we address the problem of testability analysis by efficiently taking into account signal correlation. Our algorithm called TAIR (Testability Analysis by Implication Reasoning) starts with testability analysis results such as COP [4]. For each single stuck-at fault, the accuracy of testability will be improved by recursively applying some



Testability		
Wire	COP	Exact
a–>d (s-a-1)	1/16	0
b–>d (s-a-1)	1/16	1/8
d–>e (s-a-1)	3/16	1/8
c–>e (s-a-0)	1/16	1/8
e–>f (s-a-0)	7/16	3/8
a–>f (s-a-1)	1/16	0

Fig. 1 An example of testability analysis from COP and exact solution.

simple rules. Each rule is associated with a formula which characterizes signal correlation. When the condition of a rule is met, we modify the old testability analysis result by using the associated formula. For the example in Fig. 1, to test the *d->e* stuck-at-1 fault, node *a* must be assigned *1* and node *d* must be assigned *0*. TAIR can recognize that these two signal assignments are correlated and then can correct the old result by multiplying a factor of *2/3* to get the exact result, *1/8*. The rules are derived from the implication analysis during testing a wire stuck-at fault. Further, these rules are complete in the sense that we always get the same final results regardless of how an implication algorithm is implemented.

Since our emphasis is on detecting efficiently correlation between controllability and observability of a fault, TAIR can benefit from existing algorithms, such as the cutting algorithm [19] and PREDICT [24], which attempt to improve the accuracy of signal probabilities. Although, in the following context, the method is suggested to improve COP's results, the same principle can be applied to other methods such as PREDICT. Another advantage of TAIR is that the testabilities of many redundant faults can be naturally set to zero.

This paper is organized as follows. Section 2 gives the required background. Section 3 presents the relationship between testability analysis and signal correlation. Section 4 describes the major idea of TAIR and Section 5 summarizes the algorithm by an example. In Section 6, we discuss the experimental results obtained by computer simulation. Finally, conclusions are given in Section 7.

2 Background and Definitions

Our algorithm TAIR uses the reasoning of *Automatic Test Pattern Generation (ATPG)* [12] to derive formulae for testability analysis improvement. In this section, we present the background review for testing a wire stuck-at fault and also give some definitions related to the testing area. Here, we only consider circuits with AND, OR, and NOT gates. For a complex gate, it can be decomposed into these gates first.

A node *n* is in the *transitive* fanout of a wire *w* if there is a path from *w* to node *n*. The *dominators* [13] of a wire *w* is a set of nodes which all paths from *w* to any primary output must pass through. Given a dominator *n* of a wire *w*, the *side inputs* of dominator *n* with respect to *w* are all immediate inputs of *n* not in the transitive fanout of *w*. The value *v* of an input to a node *n* is said to be *controlling* if *v* uniquely determines the value of *n* regardless of the values of all other inputs. The controlling value is *1* for an OR gate and *0* for an AND gate. The inverse of a controlling value is called a *non-controlling* or *sensitizing* value. The *mandatory assignments* (MAs) are the value assignments required for a test to exist and must be satisfied by any test vector. The process of computing these MAs and checking their consistency is referred to as *implication* [1].

We define the MA assigned to the source node of the faulty wire to be the ***activating*** value; i.e., to activate the fault. The process of implication can be as follows. The MAs on the side inputs of a dominator are set to sensitizing values and the MA on the source node of the faulty wire is set to the activating value. These MAs can then be propagated. For example, if the output of an AND {OR} gate is *1* {*0*}, the inputs are assigned *1* {*0*}. Similarly, if all the inputs of an AND {OR} gate are *1* {*0*}, the output is given *1* {*0*}. This process is called *direct implication*. Additional MAs can be found by more complicated approaches such as recursive learning [14].

If the MAs of a single stuck-at fault test cannot be consistently justified, the fault is undetectable and, therefore, the wire is redundant. For example, consider wire *a->d* stuck-at-1 fault in Fig. 1. To activate the fault, node *a* must be assigned *0* while, to propagate the fault through *f*, node *a* must be assigned the sensitizing value, 1. Both assignments conflict and the fault is thus undetectable.

3 Testability Analysis and Its Relation with Mandatory Assignments

In this section, let us revisit the testability analysis algorithm COP and also discuss its assumption of signal independencies. To simplify the discussion, let n_i be a node in the circuit and v_i be *0* or *1*.

Definition 1: The *signal probability* or *v-controllability* of a node *n, p(n=v),* is the probability to evaluate node *n* to *v,* and the *multiple signal probability* $p(n_0=v_0, n_1=v_1, ..., n_k=v_k)$ is the probability to simultaneously evaluate each node n_i to v_i.

Note that, if all signal assignments $\{n_0=v_0, n_1=v_1, ..., n_k=v_k\}$ are structurally independent, we have

$$p(n_0=v_0, n_1=v_1, ..., n_k=v_k) = p(n_0=v_0) * p(n_1=v_1) * .... * p(n_k=v_k).$$

In a tree-structured circuit, because signal assignments for the immediate fanins of a node are all independent, the signal probability of the node can be exactly computed by multiplying the signal probabilities of the node's fanins. The observability of a node can also be exactly computed by some rules. Based on the assumption of signal independencies, the COP algorithm estimates the controllability and the observability of a node using the same tree formulae. Let the COP observability of a node *n* be represented by *obv(n)*. The observability of node *d* shown in Fig. 2 is recursively computed by the following formula:

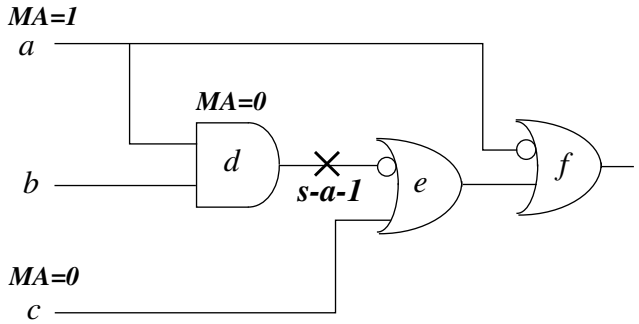


Fig. 2 The relation between COP testability and MAs for a stuck-at fault

$$obv(d) = p(c=0) * obv(e) = p(c=0) * p(a=1) * obv(f) = p(c=0) * p(a=1)$$

where *obv(f)* is assigned *1* because node *f* is a primary output. Since *obv(d) = obv(d->e),* the COP testability for fault *d->e* s-a-1 is equal to *obv(d->e)*p(d=0).* Therefore, the COP testability for *d->e* s-a-1 is

$$obv(d) * p(d=0) = p(c=0) * p(a=1) * p(d=0).$$

Note that the circuit given in Fig. 2 is the same circuit as in Fig. 1.

After briefly describing the COP algorithm, we will discuss a way of using the implication process to improve the testability analysis.

Definition 2: The *root* MAs of a wire stuck-at fault are the MAs assigned to the side inputs of a dominator (sensitization) and the MA assigned to the source of the wire (activation).

For example, in Fig. 2, the root MAs for wire *d->e* s-a-1 fault are *{c=0, a=1, d=0}* where MAs *{c=0, a=1}* are to sensitize the fault and MA *{d=0}* is to activate the fault. The implied MA *{b=0}*, however, is not a root MA.

Lemma 1: For a wire *w* stuck-at fault, let MAs $\{n_0=v_0, n_1=v_1, ..., n_k=v_k\}$ be the root MAs. The testability analysis computed by COP for *w* is

$$Const * p(n_0=v_0) * p(n_1=v_1) * .... * p(n_k=v_k) = Const \times \prod_{(n_i = v_i) \in \text{Root MAs}} p(n_i = v_i)$$

Proof. This can be proved directly by the algorithm of COP.

Given the same example in Fig. 2, the COP testability for *d->e* s-a-1 fault is determined as *p(c=0)*p(a=1)*p(d=0)*. We emphasize again that MAs *{c=0, a=1, d=0}* are exactly the root MAs for *d->e* stuck-at-1 fault.

From Lemma 1, COP computes the testability of a single stuck-at fault by multiplying the signal probability of each root MA with some constant which comes from the formula for multiple fanouts by COP. However, the signal assignments for the root MAs are not independent because all test vectors must simultaneously satisfy the root MA conditions. Therefore, a more accurate estimation for testability analysis should compute the multiple signal probability $p(n_0=v_0, n_1=v_1, ..., n_k=v_k)$, instead of computing $p(n_0=v_0)*p(n_1=v_1)*....*p(n_k=v_k)$ by assuming signal independencies of root MAs. For example in Fig. 2, COP gives a wrong testability of *3/16* for *d->e* s-a-1 from *p(c=0)*p(a=1)*p(d=0)*. The exact result can be obtained from the multiple signal probability *p(c=0, a=1, d=0) = 1/8*.

4 Derivation of Signal Correlation from Implication

In this section, we discuss an efficient algorithm to improve the estimation for $p(n_0=v_0, n_1=v_1, ..., n_k=v_k)$. We assume that the controllability and the observability for each node are derived a *priori* by COP. Our algorithm TAIR then improves the accuracy of testability analysis for each fault. Given a single stuck-at fault, we first perform implication by forward and backward propagating root MAs of the fault. Using the information from these new implied MAs, we derive a set of rules to model signal correlation. When the condition of a rule is met, the COP testability is multiplied by a corresponding correlation factor. The process is recursively applied until no rule is met. These rules are complete in the sense that they are derived from exhaustively examining all possible situations of direct implication. We will also show that the order of applying these rules will not affect the final results and therefore, TAIR is independent of how the implication process is implemented. Before we further describe these rules, let us discuss how an implication process can give hints for signal correlation.

Consider again the example, *d->e* s-a-1 fault in Fig. 2. The COP testability of the fault is determined as *p(c=0)*p(a=1)*p(d=0)* where *{c=0, a=1, d=0}* are root MAs. This testability is incorrect because MAs *{a=1, d=0}* are dependent. One can check that *{a=1, d=0}* leads to *{b=0}*. As a result, *p(c=0, a=1, d=0)* should be equal to *p(c=0, a=1, b=0)*. Suppose the new signal assignments *{c=0, a=1, b=0}* are **independent**. It is easy to see that

p(c=0, a=1, b=0)
*= p(c=0) * p(a=1) * p(b=0)*
*= p(c=0) * p(a=1) * {p(b=0)/p(d=0)} * p(d=0)*
*={COP testability result} * {p(b=0)/p(d=0)}*

Consequently, to correct the COP testability, we multiply the old COP result *3/16* by *p(b=0)/p(d=0) = 2/3,* the *correlation factor.* Thus, we obtain the exact value *1/8*. Note that assignments *{c=0, a=1, b=0}* are independent since nodes *a, b,* and *c* are all primary inputs. If they are not independent, TAIR may recursively apply rules to obtain more accurate results. We will elaborate this point later.

The success in the above example is to recognize that *{a=1, d=0}* leads to *{b=0}*. This reasoning can be obtained by using implication of MAs. It is intuitive that during implication when two signal assignments "collide" on a node, there is correlation between these two signal assignments. We now discuss a set of useful rules to resolve the signal correlation problem. Each rule is associated with a

condition and a correlation formula. Though only the case of a two-input AND gate is illustrated in the rules, the same principle can be extended to rules for other gate types with any number of inputs. Our goal is to closely approximate multiple signal correlations. Let us consider $p(n_i=v_i, n_j=v_j)$ of a two-input AND gate for simplification, and the general case can be easily deduced.

4.1 Rules for Improving Testability Analysis

All the rules are shown in Table 1 where the second column shows the condition and the third column describes the corresponding correlation factor for each rule. The condition for Rule 1 in Table 1 checks whether an AND gate n_i has a logic *"0"* MA while one of its fanins, n_j, has a logic *"1"* MA[1]. The corresponding correlation factor for this rule is $p(n_k=0)/p(n_i=0)$ where n_k is the other input of node n_i. The reason for this correlation factor is as follows: Since n_j is equal to *1*, the only possibility for n_i being *0* is that n_k must be *0*. Therefore, $p(n_i=0, n_j=1) = p(n_j=1, n_k=0)$. Our starting point, $p(n_i=0, n_j=1)$, from the COP result was calculated by $p(n_i=0)*p(n_j=1)$. After multiplying the COP result $p(n_i=0)*p(n_j=1)$ by the correlation factor $p(n_k=0)/p(n_i=0)$, we can obtain $p(n_j=1)*p(n_k=0)$ which is more accurate. We emphasize that this rule has been successfully applied to the example in Fig. 2 for deriving the exact testability of fault *d->e* stuck-at 1. Although this new result $p(n_j=1)*p(n_k=0)$ may not equal to the exact solution $p(n_j=1, n_k=0)$, we resolve this problem by recursively applying rules which will be discussed later.

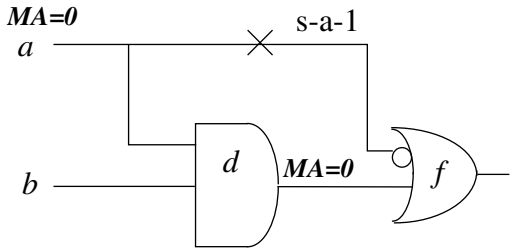


Fig. 3 An example for Rule 2.

The condition for Rule 2 in Table 1 checks whether the AND gate n_i has a *"0"* MA and one of its fanins, n_j, also has a *"0"* MA. In this case, we must ensure that the MA $\{n_i=0\}$ is not an implied value of $n_j=0$. Since the MA $\{n_j=0\}$ dominates $n_i=0$, we have $p(n_i=0, n_j=0) = p(n_j=0)$. In order to obtain $p(n_j=0)$ from $p(n_i=0)*p(n_j=0)$ which is the COP result, the correlation factor $1/p(n_i=0)$ must be used. For example, in Fig. 3, consider *a->f* s-a-1 fault. The MA $\{d=0\}$ is derived by the fact that *d* is the

1. For an OR gate, we check the output of n_i has a logic *"1"* MA and one of its fanin has a logic *"0"* MA.

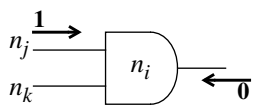
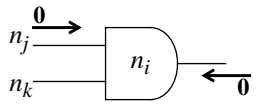
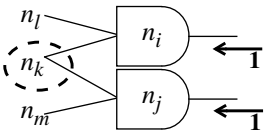
side input of the dominator *f*. Further, to activate the fault, we must have the MA *{a=0}*. Since the MA *{d=0}* is not derived from the MA *{a=0}*, the condition of Rule 2 is met and the correlation factor *1/p(d=0)* must be used. Rule 2 can also be generalized to the case where all the inputs of an AND gate have logic "1" MAs and its output also has a logic "1" MA which is not implied by the inputs.

Sometimes, several MAs can imply the same MA and this causes inaccuracy. In Rule 3 of Table 1, we check whether there is a node whose MA can be implied by several of its fanout MAs. Consider the case in Table 1 where two MAs $\{n_i=1, n_j=1\}$ can both imply the same MA $\{n_k=1\}$. The COP testability assumes $p(n_i=1, n_j=1) = p(n_k=1)*p(n_l=1)*p(n_k=1)*p(n_m=1)$ so $p(n_k=1)$ has been over-calculated. Using the same principle as in Rule 1, we have

$$
\begin{aligned}
& p(n_i=1, n_j=1) \\
& = p(n_l=1) * p(n_k=1) * p(n_m=1) \\
& = 1/p(n_k=1) * \{p(n_k=1) * p(n_l=1) * p(n_k=1) * p(n_m=1)\} \\
& = 1/p(n_k=1) * (\text{COP testability result}).
\end{aligned}
$$

Thus, if there are q such fanouts, we must multiply $q-1$ times of the correlation factor $1/p(n_k=1)$.

Table 1 The condition and correlation factor of each rule.

Rule #	The condition of a rule	Correlation Factor (CF)
Rule 1		$CF1 = p(n_k=0)/p(n_i=0)$
Rule 2		$CF\ 2 = 1/p(n_i=0)$
Rule 3		$CF3 = 1/p(n_k=1)$

4.2 Recursive Application of Rules

In Rule 1, we have derived the correlation factor by assuming that the modified multiple signal probability $p(n_k=0, n_j=1)$ is equal to $p(n_k=0)*p(n_j=1)$. However, $\{n_k=0, n_j=1\}$ may be dependent. In TAIR, this problem is resolved by recursively applying the rules. Let us illustrate this scenario by an example for computing $p(d=0, a=1)$ in Fig. 4. It is easy to see that $p(a=1, d=0)$ is equal to $p(a=1, b=0)$. We start with the COP testability $p(a=1)*p(d=0)$ and try to get the result in terms of $p(a=1)*p(b=0)$.

First, we have the *COP testability* = $p(a=1)*p(d=0)$. Immediately, Rule 1 condition is met on node *d*, and the corresponding correlation factor must be applied as follows:

Modified testability
*= {CF1 on node d} * COP testability*
*= {p(c=0)/p(d=0)} * p(d=0)* p(a=1)*
*= p(c=0) * p(a=1).*

In the second step, Rule 1 condition again is met on node *c*, and we have

Final testability
*= {CF1 on node c} * Modified testability*
*= {CF1 on node c} * [{CF1 on node d} * COP testability]*
*= {p (b=0)/p(c=0)} * [p (c=0) * p(a=1)]*
*= p(b=0) * p(a=1)*

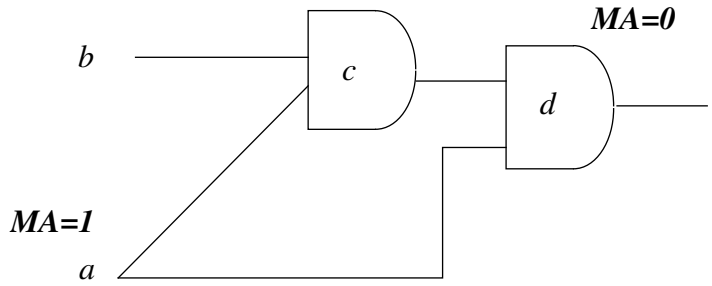


Fig. 4 Recursive application of the rules.

Therefore, by recursively applying Rule 1 twice, we can gradually modify *p(d=0)*p(a=1)* into *p(b=0)*p(a=1)* which is equal to the target signal probability, *p(d=0, a=1)*. In this way, we are able to little by little approximate the multiple signal probability to a more accurate result.

4.3 The Proposed Algorithm

```
/* assume COP is done before this routine */
Improve_testability_analysis_for_a_fault(f_i)
{
    Push_rootMAs_into_stack(f_i)
    while (stack_is_not_empty) {
        (n_i=v_i) = pop(stack)
        imply(n_i=v_i)
        If (MA_is_inconsistent)   /* the fault is redundant */
            return 0
        switch (condition of MAs) {
            case rule1 : CF = CF * CF1
            case rule2 : CF = CF * CF2
            case rule3 : CF = CF * CF3
        }
    }
return CF;
}
```

Fig. 5 The TAIR algorithm.

The entire testability enhancement algorithm is shown in Fig. 5. Given the COP results, TAIR first computes the root MAs for a target single stuck-at fault. Then, we perform direct implication on these MAs. Note that the idea of direct implication has been presented in Section 2. During the implication process, if the MAs cannot be consistently justified, we return zero for the fault. Otherwise, when the condition of a rule is met, the corresponding correlation factor is applied. The procedure for each target fault continues until no rule can be met. Following the above process, we are able to enhance the COP testability for each fault under consideration. In addition, the testabilities of many redundant

faults can be naturally set to zero. Note that our algorithm does not consider the multiple-path sensitization of a fault and neither do other non-exact methods for testability measurement.

4.4 Rule Application Order

The application conditions of all rules are checked during the direct implication process for each single stuck-at fault test. It can be found that the order of processing MAs for each fault might not be the same for different implementations. This can result in applying different rules to the same node and therefore applying different correlation factors. Fortunately, although different correlation factors are used, we will show that the final correlation factor (*CF* in Fig. 5), which is the multiplication of all correlation factors, is still the same. In other words, the resultant *CF* of TAIR is independent of the order in processing MAs and of the order in implementing the implication rules.

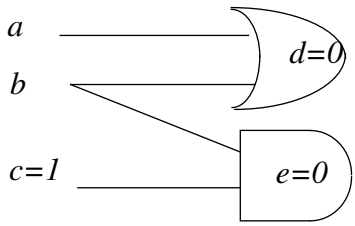


Fig. 6 The order of processing MAs.

In Fig. 6, consider the case of *p(c=1, d=0, e=0)*. Because MAs *{e=0, c=1}* leads to MA *{b=0}*, one may first apply the correlation factor *CF1 =p(b=0)/p(e=0)* of Rule 1 to node *e*. Then, MA *{b=0}* can be implied again from MA *{d=0}* so the correlation factor *CF3 =1/p(b=0)* of Rule 3 is applied to node *b*. The final correlation factor in this way is *CF = CF1*CF3={p(b=0)/p(e=0)}* {1/p(b=0)} = 1/ p(e=0)*. On the other hand, the MAs can be processed in a different order. Suppose we start by implying MA *{d=0}* to obtain MAs *{a=0, b=0}* and no rule condition is met at this step. It can then be found that MAs *{b=0, e=0}* satisfy the condition of Rule 2 on node *e* and the correlation factor, *CF =CF2=1/p(e=0),* of this rule can be applied. Although the processing order of MAs is different in both ways, we end up with the same final correlation factor, *CF=1/p(e=0)*. In fact it can be proved that our results from Fig. 5 are independent of the processing order of MAs.

5 A Complete Example

We summarize our algorithm by an example using the well-known Schneider circuit [21] in Fig. 7. Initially, the testability analysis is performed by the COP algorithm. Consider wire *f->j* s-a-0 fault whose COP observability is *125/1024* and the 1-controllability of node *f* is *1/4*. Thus, before applying TAIR, we have the COP testability equal to *125/4096*. TAIR first finds the root MAs *{f=1, d=1, k=0, i=0, h=0}*. The implication process is then applied to these MAs with the detailed steps shown in Table 2.

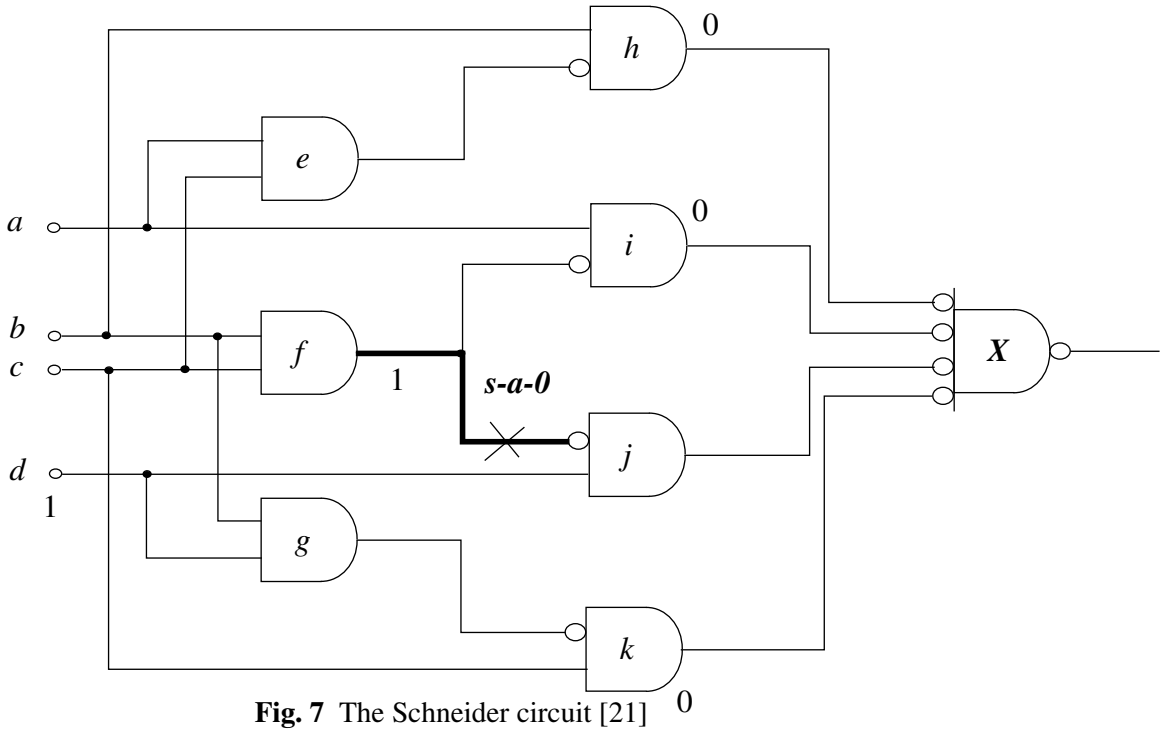


Fig. 7 The Schneider circuit [21]

In Step 1, we imply MA *{f=1},* which results in the assignments of *1* to both nodes *b* and *c*. Obviously, no rule condition is met since nodes *b* and *c* were not given any value before Step 1. In Step 2, we imply MAs *{c=1, k=0},* which results in the assignment of *1* to node *g*. The Rule 1 condition is met on node *k* and the correlation factor, *p(g=1)/p(k=0) = 0.25/0.625 = 2/5,* must be multiplied later. Note that due to the existence of an inverter at the output of *g,* the equation for Rule 1 just applied is slightly different from that in Table 1. In Step 3, after implying MA *{g=1},* we do not have any new MA assigned. However, MAs *{b=1, d=1}* are deduced again. The MA *{b=1}* is deduced from MA *{f=1}* and MA *{d=1}* is a root MA. Thus, the Rule 3 condition is met on nodes *b* and *d* and this must be corrected by the correlation factors *1/p(b=1) = 2* and *1/p(d=1) = 2*. In Step 4, we imply MAs

Table 2 Detail steps of the implication process

Step	Implication	Apply Rule	On Node(s)	Correlation Factor
1	*{f=1} => {b=1, c=1}*	None	*N/A*	*N/A*
2	*{c=1, k=0} => {g=1}*	Rule 1	*k*	*cf = p(g=1)/p(k=0) = 2/5*
3	*{g=1} => {b=1, d=1}*	Rule 3	*b,* *d*	*cf = 1/p(b=1) = 2,* *cf = 1/p(d=1) = 2*
4	*{b=1, h=0} => {e=1}*	Rule 1	*h*	*cf = p(e=1)/p(h=0) = (1/4)/(5/8) = 2/5*
5	*{e=1} => {a=1, c=1}*	Rule 3	*c*	*cf = 1/p(c=1) = 2*
6	*{f=1) => {i=0}*	Rule 2	*i*	*cf = 1/p(i=0) = 8/5*

Table 3 : control. and obv.

Node	1-Controllability	Observability
a	0.5	0.233
b	0.5	0.321
c	0.5	0.321
d	0.5	0.233
e	0.25	0.122
f	0.25	0.229
g	0.25	0.122
h	0.375	0.244
i	0.375	0.244
j	0.375	0.244
k	0.375	0.244
X	0.847	1

{b=1, h=0}, which results in the assignment of logic *1* to node *e*. Again, the Rule 1 condition is met on *h*, and the correlation factor *p(e=1)/p(h=0) = (1/4)/(5/8) = 2/5* must be applied. In Step 5, MA *{e=1}* is implied and logic *1* is assigned to both nodes *a* and *c*. Since node *c* has been assigned logic *1,* the Rule 3 condition is met on node *c* and the correlation factor, *1/p(c=1) =2* must be multiplied to the COP testability. Finally, in Step 6, we find that MA *{f=1}* dominates the root MA *{i=0}*. As a result, the Rule 2 condition is met on node *i* and the correlation factor *1/p(i=0) = 8/5* must be applied. Let us now multiply all these correlation factors with the original testability:

$$\frac{2}{5} \times 2 \times 2 \times \frac{2}{5} \times 2 \times \frac{8}{5} \times \frac{125}{4096} = \frac{1}{16}$$

The new result *1/16 =0.0625* is the exact detection probability for *f->j* s-a-0 fault.

Table 4 Testability comparisons.

Wire	Fault	COP	Exact values	Ours
g->k	s-a-0	0.0305	0.0625	0.0625
c->k	s-a-1	0.0915	0.0625	0.0625
d->j	s-a-1	0.0915	0.0625	0.0625
a->i	s-a-1	0.0915	0.0625	0.0625
f->j	s-a-0	0.0305	0.0625	0.0625
e->h	s-a-0	0.0305	0.0625	0.0625
f->i	s-a-0	0.0305	0.0625	0.0625
c->f	s-a-1	0.0573	0.0	0.0
j->X	s-a-0	0.0915	0.0625	0.0625
c->e	s-a-1	0.0305	0.0	0.0
i->X	s-a-0	0.0915	0.0625	0.0625
b->f	s-a-1	0.0573	0.0	0.0
d->g	s-a-1	0.0305	0.0625	0.0625
b->g	s-a-1	0.0305	0.0	0.0
a->e	s-a-1	0.0305	0.0625	0.0625
b->h	s-a-1	0.0915	0.0625	0.0625
h->X	s-a-0	0.0915	0.125	0.0915
k->X	s-a-0	0.0915	0.125	0.0915

ours are the same as the exact values

For the circuit in Fig. 7, we list the testability improvement results in Table 4. As can be seen, our testability results are all better than the COP results. In addition, all but two of our testability results are the same as the exact values, and four redundant wires have also been identified while COP cannot. For the last fault in Table 4, the exact testability for *k->X* s-a-0 should consider *p(h=0, i=0, j=0, k=1)*. Since there is no rule matched, our algorithm cannot detect the correlation among these signals. As a result, ours is the same as COP which assumes signal independence. The cause of inaccuracy for *h->X* s-a-0 is similar to that for *k->X* s-a-0. This example also demonstrates that COP may give high testability estimation for a fault with low testability such as *b->f* and *c->f* s-a-1 and also may give very low testability estimation for a fault with high testability such as *f->j, e->h* and *f->i* s-a-0.

6 Experimental Results

We have implemented the proposed algorithm (Fig. 5) and tested using a set of MCNC and ISCAS benchmarks. Each circuit for our experiment is first optimized by the script “script.boolean” in SIS and decomposed into AND and OR gates by using “decomp -good; tech_decomp -a 1000 -o 1000.”

The testability for each wire stuck-at fault is then computed using three different methods. The "exact" method uses Binary Decision Diagram (BDD) to obtain the exact testability. The COP result is obtained by the COP algorithm [4], while our result is obtained by TAIR in Fig. 5. The experimental results are shown in Table 5 and Table 6.

In Table 5, the first column shows the name of each circuit. The second column shows the numbers of detectable (irredundant) faults and total faults, respectively. For each circuit, we compute the *circuit testability* by the formula $\left(\sum_{faults}\frac{1}{Pdi}\right)^{1/|faults|}$ of [15] for the purpose of performance evaluation. Note that *Pdi* denotes the detection probability of detectable fault *i*. Here, only the detectable faults are under consideration. The third column compares the circuit testability among the COP, exact, and TAIR results. Since the circuit testability formula tends to amplify the effect for a small testability, the results in Table 5 demonstrate that TAIR can be much more accurate for a fault with small testability, and this result can be very useful in random test length estimation [18]. The run-time comparison is shown in the fourth column. All experiments are performed under *296* MHz UltraSPARC-II. As can be seen in the table, TAIR only requires reasonable run-time overhead with significant improvement. For example, circuit C5315 can be done in *3.57* seconds. Note that in this table, circuits marked with "*" ("**") cannot be finished by the exact (BDD) method, so they are estimated by random pattern fault simulation with 2^{19} (2^{15}) random test patterns.

Table 6 presents the results in a different form. The third column shows the percentages of COP/ TAIR testability results that are the same as the exact solutions. The fourth column shows the percentages of COP/TAIR testability results that are within *0~20%* of errors when compared to the exact results, while the ninth column shows the percentages for over *100%* of errors. For example in circuit "count" of Table 6, TAIR has *39%* of single stuck-at faults whose testability results are the same as the exact solutions, and *50%* (besides the earlier *39%*) of faults whose testability results are within *20%* of errors when compared to the exact solutions. On the other hand, COP does not have any testability value the same as the exact solution, and has only *20%* of faults whose testability values are within *20%* of errors. For better view of Table 6, we also show the bar charts for four circuits in Fig. 8. From our experimental results, it can be convinced that the proposed testability analysis is able to greatly improve the performance with reasonable overhead in computing time.

We also would like to mention that, depending on the circuit structure, the MA computations might be restricted to circuit area near the fault site and dominators, and therefore the improvements in esti-

mation quality is rather limited in some cases. However, for these cases, the run-time overhead for testability improvement is almost negligible because there are only few implications. TAIR does not waste CPU time in those cases. In addition, for some cases, which we did find, though there are only few implications, the correlation factors derived from those implications are very LARGE/SMALL. Therefore, after multiplying the correlation factors, the improvement is quite significant. In other words, one implication which results in large correlation factor can have huge improvement for testability analysis.

7 Conclusion

In this paper, we have presented an enhancement algorithm for testability analysis by implication reasoning. The TAIR algorithm starts with the COP results where signal assignments are assumed to be independent. The testability analysis results are then improved by considering signal correlations. Experimental results show that TAIR produces far more accurate testability analysis than COP using very reasonable run-time. The results can be applied to more precisely identify hard-to-detect faults which can then be dealt with by design-for-testability techniques. For example, if a fault is identified as hard-to-detect, it can be solved by inserting appropriate test logic. More precisely identifying these faults will end up with adding test logic at the right places. Also, random test pattern length generally is determined by the faults with small detection probabilities [18]. The testability analysis method proposed in this work tends to better estimate the detection probabilities for hard-to-detect faults as the simulation results demonstrated before. Thus, the random test length determined can be more accurate.

Table 5 The circuit testability comparisons.

Circuit	# of testable faults/ total faults	$\frac{\sum_{faults} \frac{1}{Pdi}}{\lvert faults \rvert}$ (*Pdi* is the testability measure for each fault and faults are those testable faults)			Run time (*sec.*)		
		COP	Exact	Ours	COP	Exact	Ours
9symml	701 / 708	180.3	150.4	139.5	0.07	43.90	0.84
C1355	1546 / 1548	100.7	78.4	87.0	0.18	9905.12	1.41
C17	28 / 28	5.7	6.3	6.5	0.03	0.24	0.03
C1908	1458 / 1462	380.5	186.1	218.4	0.19	2822.66	1.40
*C2670	1939 / 2232	104286.8	68976.3	85016.6	0.31	5952.23	1.88

Circuit	# of testable faults/ total faults	$\frac{\sum_{faults}\frac{1}{Pdi}}{\lvert faults\rvert}$ (*Pdi* is the testability measure for each fault and faults are those testable faults)			Run time (*sec.*)		
		COP	Exact	Ours	COP	Exact	Ours
C3540	3588 / 2232	1854.7	168.1	252.0	0.37	79726.65	9.98
C432	558 / 618	48.5	35.5	31.3	0.08	2555.09	0.68
C499	1546 / 1548	100.7	78.4	87.0	0.19	9926.97	1.39
C5315	5310 / 5324	122.1	59.9	91.4	0.57	10485.02	3.57
*C6288	9701 / 9796	24.1	7.1	18.7	0.99	81077.87	6.93
*C7552	6880 / 7012	162555.2	2964.6	144851.3	0.73	44826.59	6.82
C880	1222 / 1222	95.2	92.1	90.3	0.13	4231.75	0.75
alu2	1184 / 1238	5001.4	77.6	866.5	0.14	126.49	3.02
alu4	2254 / 2342	147535.1	93.2	3521.9	0.25	517.58	8.08
apex6	2273 / 2274	352.4	2591.9	1197.1	0.29	515.37	1.83
apex7	680 / 686	89.1	73.3	93.2	0.09	52.12	0.40
count	420 / 420	30070.00	14362.7	14853.9	0.07	17.69	0.29
*des	10583 / 10612	81.1	33.0	45.9	1.22	94979.54	33.53
example2	903 / 952	136.3	226.3	125.2	0.13	93.86	0.97
*i10	6512 / 6706	29708.4	2111.2	9589.4	0.72	43698.46	14.39
i6	1314 / 1314	11.3	8.8	8.8	0.18	151.28	2.27
i7	1709 / 1710	12.4	9.1	9.0	0.21	254.52	3.84
*i8	3055 / 3106	220.4	44.1	40.7	0.38	7708.32	8.47
*i9	1695 / 1696	11.4	11.1	11.0	0.21	2508.42	4.08
my_adder	576 / 576	9.9	6.3	6.6	0.08	46.75	0.27
pair	4782 / 4792	188.3	406.9	206.3	0.48	5606.68	3.82
rot	1967 / 2000	190.6	267.4	200.1	0.23	1229.41	1.20
t481	2122 / 2144	40336.4	14515.1	12969.0	0.26	444.51	12.66
term1	643 / 664	1294.2	614.7	712.6	0.09	43.87	0.49
too_large	1226 / 1226	7642.8	234498.8	34361.4	0.12	467.91	1.55
ttt2	656 / 666	65.8	115.2	89.2	0.07	44.20	0.49
s9234	1553 / 1591	81755.7	4066.4	2287.3	0.39	8719.08	2.95
**s38417	17583 / 19388	11627.8	1983.2	4177.3	6.99	44958.38	210.26
**s38584	16511 / 16794	164748.9	692.0	3404.4	8.56	34185.39	443.41
Error Percentage Average Ratio		***60.69***	***1***	***3.37***	***1***	***19972.17***	***31.83***

Table 6 The percentage of faults within certain percentage of error.

Circuit	# of total faults	eq. exact COP%/TAIR%	0%~20% COP%/TAIR%	20%~40% COP%/TAIR%	40%~60% COP%/TAIR%	60%~80% COP%/TAIR%	80%~100% COP%/TAIR%	100%~ COP%/TAIR%
9symml	708	0 / 5	27 / 29	25 / 28	22 / 17	15 / 8	3 / 4	8 / 9
C1355	1548	0 / 0	26 / 17	8 / 30	33 / 35	32 / 18	0 / 0	0 / 0
C17	28	43 / 64	36 / 36	7 / 0	14 / 0	0 / 0	0 / 0	0 / 0
C1908	1462	0 / 0	23 / 33	23 / 31	27 / 25	24 / 10	2 / 0	1 / 1
*C2670	2232	0 / 0	18 / 29	18 / 16	15 / 14	18 / 14	14 / 10	17 / 17
C3540	3734	0 / 17	21 / 35	17 / 19	18 / 14	24 / 7	13 / 4	7 / 4
C432	618	5 / 14	21 / 25	17 / 22	20 / 19	6 / 9	9 / 4	22 / 7
C499	1548	0 / 0	26 / 17	8 / 30	33 / 35	32 / 18	0 / 0	0 / 0
C5315	5324	1 / 10	23 / 28	21 / 21	18 / 11	16 / 15	20 / 15	1 / 0
*C6288	9796	0 / 1	4 / 6	10 / 14	24 / 27	48 / 44	13 / 7	1 / 1
*C7552	7012	0 / 1	11 / 15	13 / 16	16 / 14	22 / 24	35 / 27	3 / 3
C880	1222	7 / 7	40 / 51	21 / 18	11 / 7	8 / 9	11 / 7	2 / 1
alu2	1238	0 / 6	23 / 22	14 / 15	16 / 13	16 / 23	23 / 15	8 / 6

Table 6 The percentage of faults within certain percentage of error.

Circuit	# of total faults	eq. exact COP%/TAIR%	0%~20% COP%/TAIR%	20%~40% COP%/TAIR%	40%~60% COP%/TAIR%	60%~80% COP%/TAIR%	80%~100% COP%/TAIR%	100%~ COP%/TAIR%
alu4	2342	0 / 6	18 / 16	14 / 14	13 / 15	20 / 24	28 / 19	7 / 6
apex6	2274	1 / 42	53 / 37	18 / 11	17 / 3	2 / 2	3 / 1	6 / 4
apex7	686	8 / 29	45 / 52	22 / 12	17 / 5	1 / 1	1 / 1	6 / 0
count	420	0 / 39	20 / 50	65 / 11	10 / 0	5 / 0	0 / 0	0 / 0
*des	10612	0 / 0	30 / 54	25 / 20	27 / 19	15 / 6	3 / 1	0 / 0
example2	952	6 / 43	50 / 43	21 / 7	10 / 2	2 / 0	1 / 0	10 / 5
*i10	6706	0 / 2	39 / 49	17 / 15	12 / 7	8 / 8	10 / 8	14 / 11
i6	1314	1 / 62	39 / 39	60 / 0	0 / 0	0 / 0	0 / 0	0 / 0
i7	1710	1 / 63	46 / 30	34 / 5	18 / 2	0 / 0	0 / 0	1 / 0
*i8	3106	0 / 2	42 / 54	14 / 19	17 / 7	6 / 5	5 / 5	16 / 8
*i9	1696	0 / 0	37 / 53	56 / 18	5 / 29	0 / 0	1 / 0	1 / 0
my_adder	576	0 / 2	35 / 83	17 / 15	48 / 0	0 / 0	0 / 0	0 / 0
pair	4792	11 / 15	35 / 51	21 / 16	13 / 9	9 / 4	3 / 1	8 / 4
rot	2000	10 / 20	40 / 49	22 / 16	12 / 5	4 / 3	1 / 1	11 / 6
t481	2144	0 / 17	4 / 9	4 / 18	8 / 6	25 / 22	32 / 10	27 / 18
term1	664	0 / 7	25 / 45	25 / 21	23 / 13	14 / 7	3 / 3	10 / 4
too_large	1226	0 / 0	17 / 21	10 / 12	5 / 5	4 / 5	5 / 4	59 / 53
ttt2	666	1 / 34	35 / 34	28 / 16	11 / 7	8 / 4	1 / 1	15 / 4
s9234	1591	0 / 2	27 / 51	20 / 20	13 / 7	12 / 7	16 / 2	12 / 11
**s38417	19388	0 / 1	22 / 43	21 / 18	18 / 12	10 / 7	5 / 3	23 / 16
**s38584	16794	0 / 1	25 / 70	37 /14	20 / 7	7 / 2	3 / 1	7 / 4

8 Reference

[1] M. Abramovici, M.A. Breuer, A.D. Friedman, *Digital Systems Testing and Testable Design*, Computer Science Press, 1990.

[2] V. D. Agrawal and S. C. Seth, "Probabilistic testability," *Proc. of Int. Conf. Computer. Design: VLSI Computers*, pp. 562-565, 1985.

[3] P. H. Bardell, W. H. McAnney, J. Savir, "Built-In Test for VLSI Pseudorandom Techniques," New York, Wiley, 1987.

[4] F. Brglez, "On Testability of Combinational Networks," *Proc. of Int'l Symposium on Circuits and Systems*, pp. 221-225, 1984.

[5] K. T. Cheng and V. D. Agrawal, "A Partial Scan Method for Sequential Circuit with FeedBack," *IEEE Trans. on Computers,* Vol. 39*,* No. 4, pp. 544-548, May 1990.

[6] K. T. Cheng and C. J. Lin, "Timing Driven Test Point Insertion for Full-Scan and Partial-Scan BIST," *Proc. of Int'l Test Conf.*, pp. 506-514, 1995.

[7] S. Chakravarty and H. B. Hunt III, "On computing Signal Probability and Detection Probability of Stuck-at Faults," *IEEE Trans. on Computer,* Vol. 39, No. 11, pp. 1369-1377*,* Nov 1990.

[8] S. K. Jain and V. D. Agrawal, "Statistical Fault Analysis," *IEEE Design & Test Computers*, Vol. 2, No 2, pp. 38-44, 1985.

[9] P. Goel, "An Implicit Enumeration AlgoritHm to Generate Tests for Combinational Logic Circuits," *IEEE Trans. on Computers,* Vol C30, No 3. pp. 215-222, Mar. 1981.

[10] L.H. Goldstein, "Controllability/Observability Analysis of Digital Circuits," *IEEE Trans. on Circuits and Systems,* vol. CAS-26, pp. 685-693, Sep. 1979.

[11] R. K. Gaede, M. R. Mercer, and B. Underwood, "Calculation of Greatest Lower Bounds Obtainable by the Cutting Algorithm," *Proc. of Int. Test Conf.*, pp. 498-505, 1986.

[12] I. Hamzaoglu, and J.H. Patel, "new Techniques for Deterministic Test Pattern Generation," *Proc*. VTS, pp. 446-452, 1998.

[13] T. Kirkland and M. R. Mercer, "A Topological Search Algorithm For ATPG," *Proc. Design Automation Conf.*, pp. 502-508, 1987.

[14] W. Kunz and D. K. Pradhan, "Recursive Learning: An Attractive Alternative to the Decision Tree for Test Generation for Digital Circuits," in *Proc. Int. Test Conf.*, pp.816-825, 1992.

[15] R. Lisanke et al, "Testability-Driven Random Test Pattern Generation," *IEEE Trans. on Computer Aided Design,* Vol. CAD-6, pp. 1082-1087, Nov. 1987.

[16] K.P. Parker and E. J. McCluskey, "Probabilistic Treatment of General combinational Networks," *IEEE Trans. Computers*, C-24, 668-670, 1975.

[17] J. Savir, "Improved Cutting Algorithm," IBM Journal of Research and Development, vol. 34, No. 2/3, pp. 381-388, March-May, 1990.

[18] J. Savir, and P. H. Bardell, "On Random Pattern Test Length," *IEEE Trans. on Computers,* Vol. C-33, No. 6, pp. 467-474, June, 1984.

[19] J. Savir, G. S. Ditlow, and P. H. Bardell, "Random Pattern testability," *IEEE Trans. Comput.,* Vol. C-33, No. 1, pp 79-90, Jan. 1984.

[20] Y. Savaria, M. Youssef, B. Kaminska, and M. Koudil, "Automatic Test Point Insertion for Pseudo-Random Testing," *Proc. of Int'l Symposium on Circuits and Systems*, pp. 1960-1963, Jun. 1991.

[21] R. R. Schneider, "On the necessity to Examine D-Chains in Diagnostic Test Generation," *IBM Journal of Research and Development*, Vol. 11, No. 1, pp. 114, Jan. 1967.

[22] B. Seiss, P. Trouborst, and M. Schalz, "Test Point Insertion for Scan-Based BIST," *Proc. of European Test Conf.*, pp. 253-262, Apr. 1991.

[23] S. C. Seth, V.D. Agrawal, "An Exact Analysis for Efficient Computation of Random-Pattern Testability in Combinational Circuits," *Proc. of 16th Int. Fault-Tolerant Computer Symp.*, pp. 318-323, 1986.

[24] S. C. Seth, L. Pan, and V. D. Agrawal, "PREDICT: Probabilistic Estimation of Digital Circuit Testability," *Proc. of 15th Int. Fault-Tolerant Computer Symp.*, pp. 220-225, Jun. 1985.

[25] H. C. Tsai, K.T. Cheng, C. J. Lin, S. Bhawmik, "A Hybrid Algorithm for Test Point Selection for Scan-Based BIST," *Proc. Design Automation Conf.*, pp. 478-483, 1997.

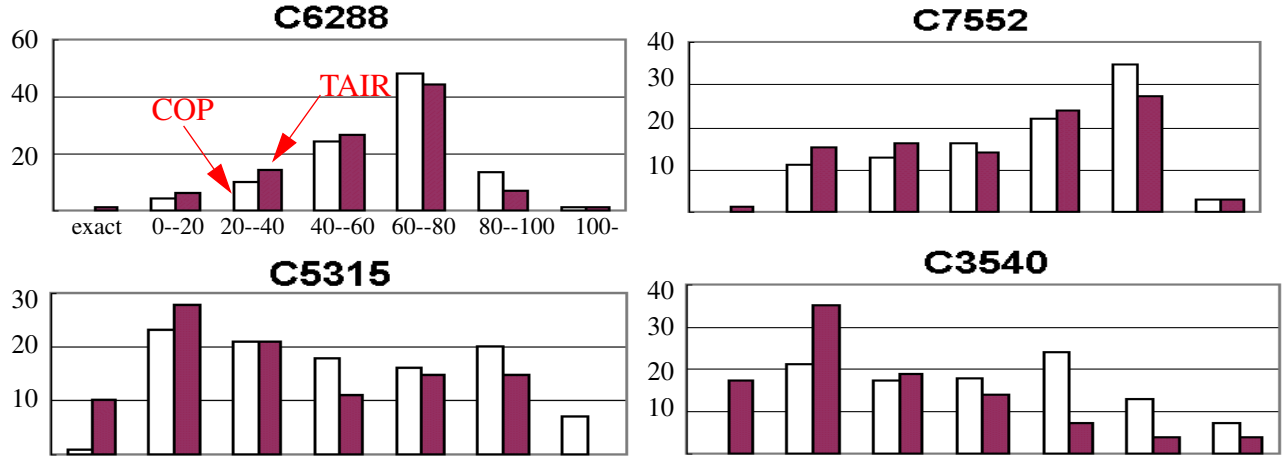


Fig. 8 Comparison of COP and TAIR in bars for some ISCAS circuits.