A Novel Combinational Testability Analysis by Considering Signal Correlation[1]

+Shih-Chieh Chang **+Shi-Sen Chang** **+Wen-Ben Jone** ***Chien-Chung Tsai**

+Department of Computer Science and Information Engineering National Chung Cheng University.

*Mentor Graphics Corp.

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 all previous work, signal correlation between controllability and observability is generally ignored. As a result, the estimated value can be inaccurate. This paper discusses an efficient method to take into account signal correlation for testability analysis. Our experimental results have shown that, with little run time overhead, significant improvement of 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 [6] [15] [17] [20] [21] [22]. In addition, the testability measurement can also give hints about the "hardest" inputs in a test generation algorithm to improve efficiency.

Normally, testability measurement assigns each node a testability value by computing the node's controllability and observability. The computation can be done by one sweep of the circuit with some rules. Since only an approximation value is provided, testability measure is less accurate than the time-consuming test generation or fault simulation process. As a result, for testability analysis to be useful, its runtime must be very fast. Usually, the analysis needs to be linear or almost linear to a circuit size.

There has been research [2] [5] [7] [8] [9] [10] [14] [18] [19] in the area of testability analysis. The algorithm COP [2] computes the signal probability (controllability) value of 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 a node by some rules. Due to the reconvergent fanout problem, signal probabilities computed by COP are generally inaccurate. The algorithm PREDICT [19], on the other hand, tries to improve the signal probability using the concept of super gate. This results in much higher complexity. The testability analysis tool SCOAP [9] does not provide probability-like values; instead, it gives a numeric number that represents the difficulty for testing a node.

In this paper, we address the problem of testability analysis by taking into account signal correlation. Our algorithm starts with testability analysis results such as COP [4]. For each stuck-at fault, the accuracy of testability will be improved by recursively applying some simple rules. Each rule is associated with a formula which characterizes signal correlation. When the con-

1. This work was partially supported by National Science Council, Taiwan R.O.C., Contract NSC 87-2215-E-194-008.

dition of a rule is met, we modify the old testability analysis result by using the associated formula.

Since our emphasis is on detecting correlation between controllability and observability of a fault, the proposed algorithm can benefit from existing algorithms which attempt to improve the accuracy of signal probabilities such as the cutting algorithm [14] or the super-gate analysis [19]. Although, in the following context, the method is suggested to improve COP's results, the same principle can be applied to methods such as SCOAP. Another advantage of our algorithm is that the testabilities of many redundant faults can be naturally set to zero.

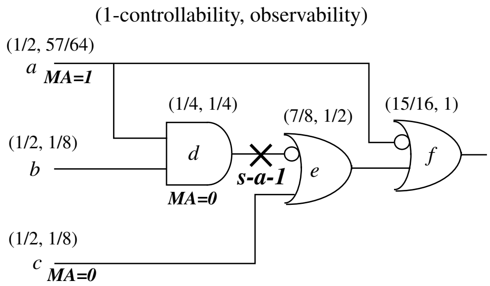
This paper is organized as follows. Section 2 discusses the background. Section 3 reviews the COP algorithm. Section 4 describes our main algorithm and Section 5 summarizes our algorithm by an example. In Section 6, we discuss the experimental results. Finally, conclusions are given in Section 7.

2 Background and Definitions

Our algorithm uses the reasoning of *Automatic Test Pattern Generation (ATPG)* to derive formulae for testability analysis improvement. In the section, we present the background review for testing a wire stuck-at fault and also give some definitions related to testing areas. Here, we only consider circuits with AND, OR, and INV gates. For a complex gate, it can be decomposed into AND and OR gates first.

The *dominators* [11] 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 are all immediate inputs of n not in the transitive fanout of w. The value v of an input to a node is said to be *controlling* if v uniquely determines the value of n regardless of the values of the 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 *noncontrolling* 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 target wire to be the *activating* value. The process of implication is 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 target wire is set to the activating value. These MAs can then be propagated. If the output of 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 *simple implication*. Additional MAs can be found by more complicated approaches such as recursive learning [12].



Testability			
Wire	COP	Exact	root MA
a–>d (s-a-1)	1/16	0	conflict at *a*
b–>d (s-a-1)	1/16	1/8	*a=1, c=0, b=0*
d–>e (s-a-1)	3/16	1/8	*c=0, a=1, d=0*
c–>e (s-a-0)	1/16	1/8	*d=1, a=1, c=1*
e–>f (s-a-0)	7/16	3/8	*a=1, e=1*
a–>f (s-a-1)	1/16	0	conflict at *a*

Fig. 1 The relation between COP testability and MAs for a stuck-at fault

If the MAs of a stuck-at fault test cannot be consistently justified, the fault is untestable and, therefore, the wire is redundant. For example, let us consider wire *a->d* stuck-at-1 fault as shown 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 untestable.

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 logic 0 or 1.

Definition 1 The *signal probability* or *v-controllability, 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-structure 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 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. 1 is recursively computed by the following formula:

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

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 d->e s-a-1 testability is*

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

After briefly describing the COP algorithm, we will discuss a way of using implication 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. 1, 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 $p(n_0=v_0)*p(n_1=v_1)*...\ *p(n_k=v_k)*Constant.$

Given the same example in Fig. 1, the COP testability for *d->e* s-a-1 fault is determined as *p(c=0)* p(a=1)*p(d=0).* Note 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 by individually multiplying the signal probability of each root MA with some constant. However, because all test vectors must simultaneously satisfy the root MA conditions, a more accurate estimation for testability analysis can be done by computing the multiple signal probability $p(n_0=v_0,\ n_1=v_1,\ ...,\ n_k=v_k),$ instead of just computing $p(n_0=v_0)*p(n_1=v_1)*....*p(n_k=v_k)$ by assuming signal independencies of root MAs.

For the example in Fig. 1, COP gives a wrong testability of *3/16* for *d->e* s-a-1 fault while the exact result should be 1/8. The reason mainly comes from that $p(c=0, a=1, d=0)$ is not equal to $p(c=0)*p(a=1)*p(d=0)$. One can easily verify that the exact result is $p(c=0, a=1, d=0) = 1/8$.

4 Derive Signal Correlation from Implication

As mentioned, COP estimates testability by assuming signal independencies of root MA conditions. When those conditions are tightly correlated, the estimation can be very inaccurate. 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 observability for each node are derived a *priori* by COP. Our algorithm then improves the accuracy of testability analysis of each fault. For each 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 some correlation factor. The process is recursively applied until no rule is met. Before we describe those 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. 1. 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

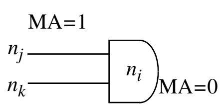
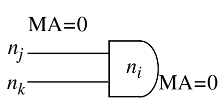
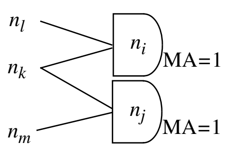
$$\begin{aligned}&p(c=0, a=1, b=0)\\&= p(c=0) * p(a=1) * p(b=0)\\&= p(c=0) * p(a=1) * \{p(b=0)/p(d=0)\} * p(d=0)\\&= \{COP\ testability\ result\} * \{p(b=0)/p(d=0)\}\end{aligned}$$

Consequently, to correct the testability result, we multiply the old value $(3/16)$ with the correlation factor, $p(b=0)/p(d=0) = 2/3$. Thus, we obtain the exact value $1/8$. Note that those assignments $\{c=0, a=1, b=0\}$ are independent since nodes *a, b,* and *c* are primary inputs. If they are not independent, our algorithm 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 develop a set of useful rules to resolve the signal correlation problem. The rules are classified into three different groups. In the following, we describe the condition for each rule along with its associated correlation formula. Though only the case of 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 correlation, let us consider $p(n_i=v_i, n_j=v_j)$ for simplification. The general case can be easily implied.

Table 1 The condition and correlation factor of a rule.

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

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 that 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 $p(n_i=0)*p(n_j=1)$ with the correlation factor $p(n_k=0)/p(n_i=0)$, we can obtain $p(n_j=1)*p(n_k=0)$. 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. This point will be discussed later.

The condition for Rule 2 in Table 1 checks that the AND gate n_i has a "0" MA and one of its fanins, n_j, also has a "0" MA. Note that 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 $p(n_i=0)*p(n_j=0)$ that is the COP result, the correlation factor $1/p(n_i=0)$ must be used.

Sometimes, several MAs can imply the same MA. In Rule 3 of Table 1, we check whether there is a node whose MA can be implied by several MAs such as its fanout MAs or its fanin MAs. Consider the case in Table 1 where two MAs $\{n_i=1, n_j=1\}$ can both imply the same MA $\{n_k=1\}$. Note that the COP testability assumes $p(n_i=1, n_j=1) = p(n_k=1)*p(n_l=1)*p(n_k=1)*p(n_m=1)$. Using the same principle as in Rule 1, we have

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

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

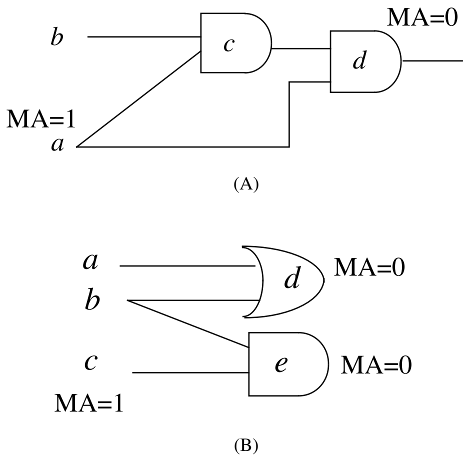


Fig. 2 (A) Recursively applying the rules. (B) Order of processing MAs

4.2 Recursive Application of Rules

In Rule 1, we derive the correlation factor by assuming 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 our algorithm, 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. 2(A). 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

$$COP\ testability = p(a=1) * p(d=0)$$

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

Immediately, Rule 1 condition is met on node d, and the corresponding correlation factor must be applied as follows.

Modified testability
*= {the Rule 1 CF on node d} * COP testability*
$= \{p(c=0)/p(d=0)\} * p(a=1) * p(d=0)$
$= p(c=0) * p(a=1)$

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

Final testability
*= {the Rule 1 CF on node c} * Modified testability*
$= \{p(b=0)/p(c=0)\} * p(c=0) * p(a=1)$
$= p(b=0) * p(a=1)$
$= \{p(c=0)/p(d=0)\} * \{p(b=0)/p(c=0)\}$ * COP testability

Therefore, by recursively applying Rule 1 twice, we can gradually modify $p(d=0)*p(a=1)$ into $p(b=0)*p(a=1)$ that 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 Results are Independent to Implementations

The entire algorithm is shown in Fig. 3. Given the COP result, our algorithm first computes the root MAs for a target stuck-at fault. Then, we perform simple implication on these MAs. Note that the simple implication has been defined 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. This procedure for the 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.

The conditions of rules are checked during the implication process of a stuck-at fault test. Note that the order of processing MAs for different implementations can be different even though the final MAs are the same. This can result in applying different rules on the same node. Fortunately, we have the following theorem.

```
/* assume COP is done before this routine */
Improve_testability_analysis_for_a_fault(fi)
{
    Put_rootMAs_into_stack(fi)
    while (stack_is_not_empty) {
        (ni=vi) = pop(stack)
        imply(ni=vi)
        If (MA_is_inconsistent)
            return 0. /* the fault is redundant */
        switch (condition of MAs) {
            case rule1 : apply CF1
            case rule2 : apply CF2
            case rule3 : apply CF3
        }
    }
}
```

Fig. 3 The testability analysis enhancement algorithm

Theorem 2 The results of our algorithm are **independent** of the order in processing MAs.

For example, in Fig. 2(B), let us 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 $p(b=0)/p(e=0)$ of Rule 1 on node e. Then, MA $\{b=0\}$ can be implied again from MA $\{d=0\}$ so the correlation factor $1/p(b=0)$ of Rule 3 can be applied to node b. The final correlation factor in this way is $\{p(b=0)/p(e=0)\} * \{1/p(b=0)\} = 1/p(e=0)$. On the other hand, the MAs can be processed in 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 be found that MAs $\{b=0, e=0\}$ satisfy the condition of Rule 2 on node e and the correlation factor, $1/p(e=0)$, 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, $1/p(e=0)$.

5 A Complete Example

We summarize our algorithm by an example using the well-known Schneider circuit [16] in Fig. 4. Initially, the testability analysis is performed by the COP algorithm and the results of the 1-controllability and observability for each node are shown in the Table A of Fig. 4. Consider wire *f->j* s-a-0 fault whose COP observability is *125/1024* and 1-controllability of node *f* is *1/4*. Thus, before improvement, we have that the COP testability is equal to *125/4096*.

Our algorithm first finds the root MAs *{f=1, d=1, k=0, i=0, h=0}*. The implication process is then applied on these MAs with the detailed steps shown in Table 2.

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}* was deduced from MA *{f=1}* and MA *{d=1}* was a root MA. Thus, Rule 3 conditions are met on nodes *b* and *d* and this must be corrected by the correlation factors *1/p(b=1) = 2* and *1/p(c=1) = 2* of Rule 3. In Step 4, we imply MAs *{b=1, h=0}*, which results in the assignment of logic 1 to node *e*. Again, the Rule 1 condition is met on *h*, 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*,* 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

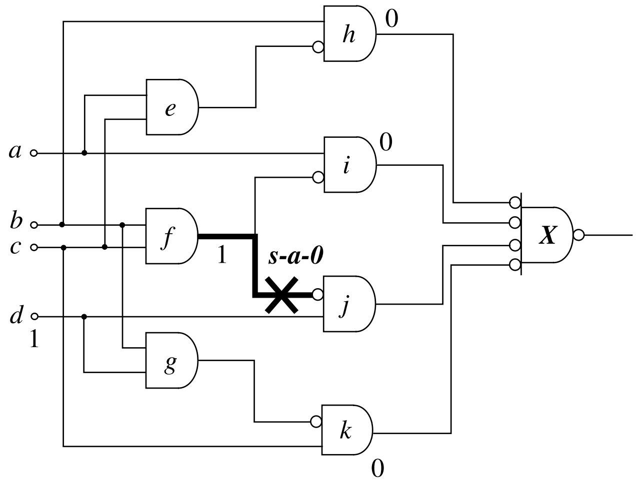


Table A : controllability & observability

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

Table B : Testability comparison.

Wire	Fault	COP	Exact	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 exacts

Fig. 4 An example from [16]

Step 6, we find that MA *{f=1}* dominates the root MA *{i=0}*. As a result, 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 testability for *f->j* s-a-0 fault.

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*

For the circuit in Fig. 4, we list the testability results in the Table B. 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 results and four redundant wires can also be identified while COP cannot.

6 Experimental Results

We have implemented the algorithm in Fig. 3 and tested on 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 test 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 algorithm [4], and our result is obtained by the algorithm in Fig. 3. The experimental results are shown in Table 3.

In Table 3, the first column shows the name of each circuit. Note that those C-circuits in ISCAS cannot be finished by the exact (BDD) method are not listed in the table. The second column shows the total number of detected irredundant faults. For each circuit, we compute the *circuit testability* by the formula $\left(\sum_{faults}\frac{1}{Pdi}\right)/|faults|$ of [13] for the purpose of performance evaluation. Here, only the testable faults are under consideration. The third column compares the circuit testability among the COP, exact, and our results. For example, the circuit testability for "count" is *30070.0* by COP, *14362.7* by the exact solution and *14853.9* by our algorithm. For this case, COP has *109%* error with respect to the exact result while our result has only *3%* error. Since the circuit testability formula tends to amplify the effect for a small testability, the results in Table 3 demonstrate that our algorithm can be much more accurate for a fault with small testability. The run time comparison is shown in the fourth column. As can be seen in the table, our algorithm only requires little run time overhead for significant improvement. For example, circuit C5315 can be done in *3.57* seconds under *296* MHz UltraSPARC-II.

From Table 3, it can be convinced that the proposed testability analysis is able to greatly improve the performance of available ones with quite limited computing time overhead.

7 Conclusion

We have discussed a testability analysis enhancement algorithm. Our algorithm starts with the COP results where signal assignments are assumed to be independent. Our algorithm then improves the testability analysis results by considering signal correlation. Our experiment results show that our algorithm produces far more accurate results than the results of COP.

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*, 1985, pp.562-565.

[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*, May 1984, pp. 221-225.

[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*, Apr. 1990, pp.544-548.

[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.*, Oct. 1995, pp.506-514.

[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,* Nov 1990, pp. 1369-1377.

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

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

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

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

[12] 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, Oct. 1992.

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

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

[15] 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*, Jun. 1991, pp. 1960-1963.

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

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

[18] 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.*, 1986, pp. 318-323.

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

[20] N. Tamarapalli and J. Rajski, "Constructive Multi-Phases Test Point Insertion for Scan-Based BIST," *Proc. of Int'l Test Conf.*, Oct 1996, pp. 649-658.

[21] N. A. Touba and E. J. McCluskey, "Test Point Insertion Based on Path Tracing," *Proc. of VLSI Test Symposium*, Apr. 1996, pp. 2-8.

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

Table 3 Compare the circuit testability.

Circuit	# of testable fault	$\frac{\sum_{faults} \frac{1}{Pdi}}{\lvert faults \rvert}$ (faults are those testable faults)			Run time (*sec.*)		
		COP	Exact	Ours	COP	Exact	Ours
9symml	701	180.3	150.4	139.5	0.07	43.90	0.84
C1355	1546	100.7	78.4	87.0	0.18	9905.12	1.41
C17	28	5.7	6.3	6.5	0.03	0.24	0.02
C1908	1458	380.5	186.1	218.4	0.19	2822.66	1.40
C3540	3588	1854.7	168.1	252.0	0.37	79726.65	9.98
C432	558	48.5	35.5	31.3	0.08	2555.09	0.68
C5315	5310	122.1	59.9	91.4	0.57	10485.02	3.57
C880	1222	95.2	92.1	90.3	0.13	4231.75	0.75
alu2	1184	5001.4	77.6	866.5	0.14	126.49	3.02
alu4	2254	147535.1	93.2	3521.9	0.25	517.58	8.08
apex6	2273	352.4	2591.9	1197.1	0.29	515.37	1.83
apex7	680	89.1	73.3	93.2	0.09	52.12	0.40
c8	384	112.5	73.1	74.7	0.06	13.57	0.23
cht	480	7.5	6.7	6.6	0.09	19.56	0.26
count	420	30070.0	14362.7	14853.9	0.07	17.69	0.29
example2	903	136.3	226.3	125.2	0.13	93.86	0.97
f51m	426	30.3	20.9	20.7	0.07	16.85	0.37
frg1	366	1643.5	1770.9	1717.5	0.05	24.36	0.28
frg2	2515	388.7	265.2	262.3	0.29	746.77	3.90
i6	1314	11.3	8.8	8.8	0.18	151.28	2.27
i7	1709	12.4	9.1	9.0	0.21	254.52	3.84
my_adder	576	9.9	6.3	6.6	0.08	46.75	0.27
pair	4782	188.3	406.9	206.3	0.48	5606.68	3.82
rot	1967	190.6	267.4	200.1	0.23	1229.41	1.20
t481	2122	40336.4	14515.1	12969.0	0.26	444.51	12.66
term1	643	1294.2	614.7	712.6	0.09	43.87	0.49
too_large	1226	7642.8	234498.8	34361.4	0.12	467.91	1.55
ttt2	656	65.8	115.2	89.2	0.07	44.20	0.49
x1	806	494.7	660.9	842.3	0.11	77.30	0.55
x2	118	48.6	43.6	41.9	0.04	1.62	0.08
x3	2257	706.8	213.2	199.1	0.26	472.41	1.68
x4	1070	34.8	35.3	26.4	0.13	125.78	0.74
Average Ratio		***52.998***	***1***	***2.423***	***1***	***12776.62***	***10.10***