Fault Simulation under the Multiple Observation Time Approach using Backward Implications

Irith Pomeranz and Sudhakar M. Reddy +
Electrical and Computer Engineering Department
University of Iowa
Iowa City, IA 52242

Abstract

We present an improved procedure for fault simulation under the multiple observation time approach based on state expansion. Under state expansion, an incompletely specified state reached during fault simulation is replaced by 2^k states, each one assigning a different combination to k unspecified present state variables. For each expanded state, additional output values are then implied. As a result, a fault that cannot be identified as detected using conventional simulation may now be identified as detected. The procedure proposed here enhances state expansion by backward implications to take advantage of every present state variable value specified under state expansion. As a result of using backward implications, fewer states need to be considered after state expansion, fewer state expansions are potentially needed for every fault, and the number of faults that can be efficiently considered is increased. Experimental results are presented to support these claims.

1. Introduction

Given a test sequence, a fault simulator should accurately determine the faults detected by the sequence. For synchronous sequential circuits, conventional fault simulation procedures [1] may fail to identify faults as detected, due to the loss of precision in three value logic simulation and the use of the single observation time approach [2]. To improve the accuracy of fault simulation, it is possible to use the multiple observation time approach proposed in [2]. This approach allows accurate determination of detected faults by considering responses of different initial states separately. For example, if the fault free output sequence of a circuit is (000) and the output sequence in the presence of a fault f is either (010) or (101) depending on the initial state of the faulty circuit, then conventional simulation would derive the output sequence (xxx) for the faulty circuit, and the fault would not be declared detected. By considering each initial state of the faulty circuit separately and deriving the output sequences (010) and (101), the fault is identified as detected under the multiple observation time approach.

+ Research supported in part by NSF Grant No. MIP-9220549, and in part by NSF Grant No. MIP-9357581

DAC 97, Anaheim, California
(c) 1997 ACM 0-89791-920-3/97/06 ..$3.50

The multiple observation time approach in its most general form requires the use of multiple responses of the fault-free circuit to the given test sequence. However, conventional test application to the circuit can only accommodate a single output response of the fault-free circuit. In order to admit this practical situation, the restricted multiple observation time approach was proposed in [2], [3]. This approach employs a single fault-free circuit response; however, it allows multiple output responses to be considered for the faulty circuit. Several fault simulation methods under the restricted multiple observation time approach were proposed in [4]-[7]. The procedure of [4] uses *state expansion*, defined as follows. Consider a test sequence T that brings the circuit to an incompletely specified state s at time unit u. Suppose that present state variable y_i is unspecified in s. Under state expansion, we may replace s by two different states, $s(y_i = 0)$ which is identical to s except that y_i is set to 0, and $s(y_i = 1)$ which is identical to s except that y_i is set to 1. After state expansion, fault simulation continues for each state separately. As a result of state expansion, additional next state variables and primary output values become specified for the various starting states at time unit u, helping to determine whether the fault is detected. Other fault simulation methods under the restricted multiple observation time approach are based on symbolic simulation using a *BDD* representation of the circuit [5], the use of implications [6], and probablistic analysis using limited state expansion [7]. All these methods attempt to resolve the unknown values in the response of the faulty circuit to determine that the faulty circuit response for every one of its initial states conflicts with the fault-free response.

The state expansion procedure of [4] does not use *BDD*s and hence is applicable to circuits for which *BDD*s cannot be derived. Additionally, it provides an accurate implementation of the restricted multiple observation time approach, in the sense that it is capable of determining accurately if a fault is detected by the given test sequence when conventional test application is employed. The procedure of [6] is not accurate in this sense. However, the procedure of [4] is computationally intensive and existing implementations cannot handle all the faults in large circuits. In this work, we present a method to improve the effectiveness of state expansion. We demonstrate that the proposed method allows handling of faults and circuits that were not handled before.

The unique feature of the procedure proposed here that distinguishes it from previous approaches is that it uses implications done backward in time to take full advantage of every flip-flop value set under state expansion. Previously [4], implications following state expansion were done only forward in time. For example, if a present state variable y was specified at time unit u, then its values were implied at time units $u, u+1, \cdots$, in this order, to compute output sequences with fewer unspecified values. Under backward implications, we observe that setting

$y = \alpha$ at time unit u implies that the corresponding next-state variable Y must be α at time unit $u-1$. This may imply additional values at time unit $u-1$, and may help specify the values of additional next state variables at time unit $u-1$. This in turn would specify additional present state variables at time unit u. Backward implications may also reveal that a value α on state variable y at time unit u conflicts with the values of other lines in the circuit; it may also assign primary output values at time unit $u-1$ that conflict with the corresponding fault free values, resulting in fault detection. In both of these cases, the states with a value α on state variable y at time unit u do not need to be considered further. Thus, fewer states need to be considered after state expansion, and the states that need to be considered have more specified state variables, implying that fewer state expansions are potentially needed for every fault. Consequently, the number of faults that can be efficiently considered is increased.

The paper is organized as follows. Section 2 demonstrates the advantages of backward implications. Section 3 presents the fault simulation procedure. Experimental results are included in Section 4. Section 5 concludes the paper.

2. State expansion and backward implications

In this section, we first review the state expansion procedure of [4]. We then introduce backward implications and show the advantages of adding backward implications to state expansion.

Both state expansion and backward implications are useful in increasing the information available regarding line values in the circuit under test. Specifically, they result in output sequences with increased numbers of specified values. By maximizing the number of specified values, we also maximize the ability to identify that a given fault is detected by the test sequence being simulated.

State expansion is applied to a single sequence of states and a single output sequence obtained using conventional simulation starting from the all-unspecified initial state. State expansion is used to specify values of state variables until fault detection can be established or a limit on the number of state expansions is reached. This limit is necessary since every state expansion of a state variable y_i at time unit u duplicates the state and output sequences, setting state variable y_i at time unit u to 0 in one copy and to 1 in the other. To illustrate the advantages of state expansion, we consider a circuit whose state and output sequences after conventional simulation are as shown in Table 1(a). Suppose that state expansion is performed based on the first state variable y_1 at time unit 1 in the faulty circuit. Then two state (output) sequences are obtained, one where $y_1 = 0$ at time unit 1, and one where $y_1 = 1$ at time unit 1. Let the sequences obtained after implying the value of y_1 at time unit 1 be the ones shown in Table 1(b). Note for example that setting $y_1 = 1$ at time unit 1 results in specifying the value of one of the outputs at time unit 1, and the value of the second present state variable at time unit 2. The second output sequence allows the fault to be detected (at time unit 2 on the second output). Thus, we do not need to consider it further. For the first sequence, additional state expansion steps are needed before it is possible to declare the fault as detected.

To demonstrate the advantages of adding backward implications to state expansion, we use ISCAS-89 benchmark circuit *s*27 shown in Figure 1 under the input pattern (1001) assuming that its state is fully unspecified (*xxx*). Considering the input pattern (1001) as the first pattern of a test sequence (the input pattern applied at time unit 0), our goal is to compare the number of next-state variables and outputs that can be specified under various simulation schemes when (1001) is applied. For simplicity, we consider only the fault free circuit. The conclusions we draw hold for faulty circuits as well.

Table 1: An example of expansion
(a) Conventional simulation

	time	0	1	2	3
fault free	state	*xx*	*x*0	1*x*	00
	output	*xx*0	0*x*1	111	011
faulty	state	*xx*	*xx*	0*x*	*x*1
	output	*x*0*x*	*xxx*	1*x*1	011

(b) After expansion

	time	0	1	2	3
faulty	state1	*xx*	0*x*	0*x*	01
	output1	*x*0*x*	*x*11	1*x*1	011
	state2	*xx*	1*x*	01	11
	output2	*x*0*x*	*xx*1	101	011

The results of conventional three-value simulation of the input pattern (1001) when the present-state variables are unspecified are shown in Figure 1. Values are shown in parentheses. The next-state variables and the output are all unspecified.

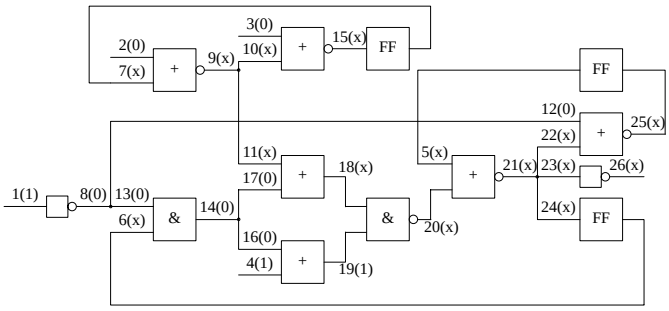

Figure 1: Conventional simulation of *s*27

The results of state expansion using present-state variable 7 at time unit 0 are shown in Figure 2. The value of line 7 is denoted by (0,1) corresponding to two different partially specified initial states, one where line 7 is 0 and one where line 7 is 1. The values (α_0, α_1) on a line g indicate that α_0 results from setting line 7 to 0, and α_1 results from setting line 7 to 1. A single value (α) on a line g indicates that $g = \alpha$ regardless of the value of line 7. As a result of this expansion, the primary output becomes partially specified (specified only when line 7 assumes the value 1), next-state variable 15 is fully specified, and next-state variables 24 and 25 are partially specified. Considering the next-state variables and the outputs, we have five specified values compared to conventional simulation that yielded no specified values. For time unit 0, state expansion using state variable 7 yields the largest number of specified values, compared to the other state variables. State expansion using state variable 6 does not result in any specified values, and expansion of state variable 5 results in three specified values.

Next, we consider the expansion of state variable 6 at time unit 1 (this is the only expansion assumed in this example). The effects of this expansion on time unit 1 are unknown, since the input pattern is unknown. Our interest is in the information that can be derived from this expansion at time unit 0. Setting state variable 6 at time unit 1 to (0,1) implies that next-state variable 24 has the values (0,1) at time unit 0. Setting line 24 to (0,1) implies that lines 21, 22 and 23 are (0,1). Additional values can then be implied as shown in Figure 3. It can be seen that the primary output and next-state variable 25 become fully specified. In addition, next-state variable 15 becomes partially specified. We

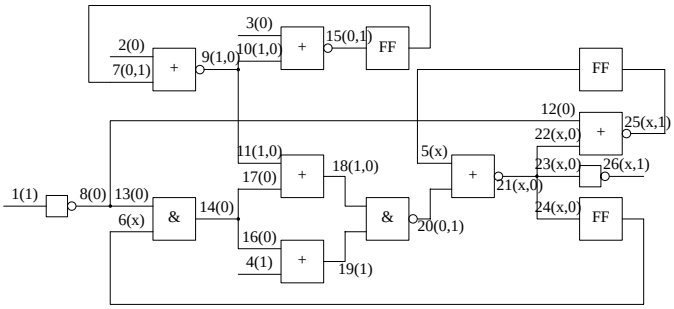

Figure 2: State expansion of state variable 7 at time 0

have a total of seven specified values at time unit 0 compared to at most five when state expansion was done at time unit 0 and backward implications were not used. Note that when we say *backward* we mean backward in time.

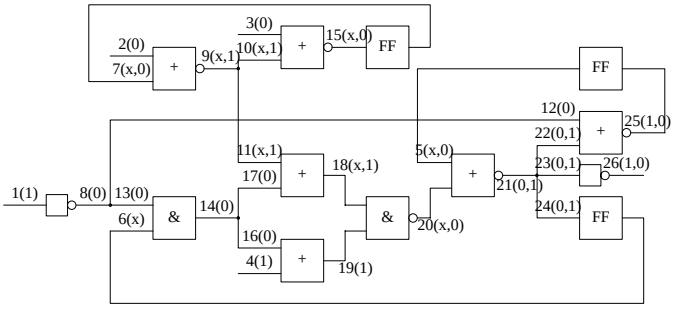

Figure 3: Backward implication of state variable 6 at time 1

The process of backward implications consists of the following steps. First, the value of a present state variable y at time unit u is assigned to the corresponding next state variable Y at time unit $u-1$. Then, additional values are assigned at time unit $u-1$ in two directions: from outputs to inputs and then from inputs to outputs. Several passes over the circuit at time unit $u-1$ may be required to determine all the implications. To keep the computation time low, we use only two passes in our implementation, one pass from outputs to inputs and one pass from inputs to outputs. Backward implications may also be done over multiple time units. For example, suppose that backward implication of next-state variable Y_i at time unit $u-1$ results in a specified value on present-state variable y_j at time unit $u-1$. Then we can assign the same value to next-state variable Y_j at time unit $u-2$ and continue to perform backward implications. In our implementation we consider only one time unit during backward implications.

Previous procedures based on state expansion [4] did not take advantage of backward implications to increase the amount of information available regarding circuit state and primary output values. As demonstrated by the example above, backward implications can help specify additional circuit values, thus increasing the effectiveness of state expansion. Another important consequence of backward implications, which is ignored by existing expansion techniques, is that it can identify cases where some state variable values are inconsistent with the input sequence. These cases do not need to be further simulated. The following example demonstrates such a case and shows how backward implications can help identify it. Again, the example uses only the fault free circuit, however, the same observations are used during fault simulation for faulty circuits.

Consider the circuit of Figure 4 under the input combination (0). Setting line 1 to 0 implies only that lines 3 and 4 are set to 0. Let us expand the present-state variable at time unit 1. Backward implications take place at time unit 0 with line 11 set to (0,1). The results of backward implication are shown in Figure 4 in parentheses. When line 11 is set to 1, we obtain that line 5 must be 1 and line 6 must be 0. This implies two different specified values on line 2, and a conflict is identified. The conflict is marked with a C in Figure 4. We conclude that line 11 can only assume the value 0 at time unit 0, and that the present-state variable can only assume the value 0 at time unit 1. State expansion and backward implications result in this case in a single state, 0, that needs to be considered further.

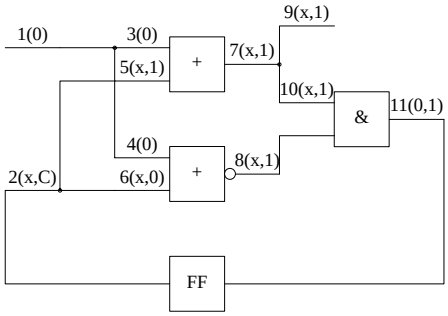

Figure 4: An example of a conflict

Fault detection for certain states can also follow from backward implications. For example, consider a case where the state s of the faulty circuit at time unit u is replaced by two states, $s(y_i=0)$ and $s(y_i=1)$ corresponding to setting present-state variable y_i to these values. Suppose that when next-state variable Y_i is set to 0 at time unit $u-1$ in the faulty circuit, an output o is set to 0. If the value produced by the fault free circuit on output o at time unit $u-1$ is 1, then the fault is detected for $s(y_i=0)$. In this case, only state $s(y_i=1)$ needs to be considered further.

Backward implications are also used in this work to help identify effective state variables and effective time units for state expansion. We describe this heuristic in more detail in Section 3.

During fault simulation, state expansion and backward implications can be used both in the fault free and in the faulty circuit. If state expansion is performed in the fault free circuit, multiple fault free responses may be obtained. In this work, we use state expansion and backward implications only in the faulty circuit. The result is that faults are detected under the restricted multiple observation time approach [2], [3].

3. The fault simulation procedure

In this section, we describe the proposed fault simulation procedure. To simplify the discussion and the implementation of the procedure, we do not consider methods to speed up the simulation process. Thus, faults and input patterns are considered sequentially. The test sequence to be simulated is denoted by T and its length is denoted by L. The input pattern included in T at time unit u is denoted by $T[u]$, where $0 \le u \le L-1$. For ease of notation we may sometimes refer to time unit L of T. The input pattern at this time unit is undefined, however, the circuit state can be determined by simulating T.

The fault simulation procedure starts with conventional simulation of the fault free circuit under the given test sequence T. Every fault is then considered separately and the following procedures are applied.

First, the fault is simulated using a conventional fault simulation procedure. If the fault is detected, it is marked as such

and dropped from further consideration.

Next, we check the following necessary condition for fault detection under the restricted multiple observation time approach. Under this approach, state expansion is done only in the faulty circuit, potentially specifying additional output values in this circuit. The output sequence of the fault free circuit is not affected by state expansion. Consequently, to be able to detect a fault, there must exist a time unit u and an output o such that o is specified in the fault free circuit at time unit u and unspecified in the faulty circuit. State expansion at time unit $u_0 \leq u$ or backward implications due to state expansion at time unit $u_0 \leq u+1$ may then specify o at time unit u and allow to determine that the fault is detected. To state this condition more formally, we use the following notation.

• $N_{sv}(u)$ is the number of unspecified state variables in the faulty circuit at time unit u.

• $N_{out}(u)$ is the number of pairs (u', o) such that output o at time unit $u' \geq u$ is specified in the fault free circuit and unspecified in the faulty circuit. For example, for the output sequences of Table 1(a) we have $N_{out}(0) = 4$, $N_{out}(1) = 3$, $N_{out}(2) = 1$ and $N_{out}(3) = 0$.

A necessary condition for a fault to be detectable by state expansion and backward implications at time unit $0 < u \leq L$ is the following.

(C1) $N_{sv}(u) > 0$ and $N_{out}(u-1) > 0$

(we check for unspecified outputs at time unit $u-1$ since backward implications may allow setting these values, thus detecting the fault at time unit $u-1$). For time unit $u = 0$, the following is a necessary condition for fault detection under the restricted multiple observation time approach.

(C2) $N_{sv}(u) > 0$ and $N_{out}(u) > 0$

We observe that $N_{sv}(u) > 0$ implies that $N_{sv}(u-1) > 0$ (if $N_{sv}(u-1) = 0$, the state at time unit $u-1$ is fully specified, implying that $N_{sv}(u) = 0$ as well). Based on this observation, we consider $u = 0$ and $u > 0$ together by replacing (C1) and (C2) by the following condition.

(C) $N_{sv}(u) > 0$ and $N_{out}(u) > 0$

If this condition is not satisfied at any time unit $0 \leq u < L$, the fault is dropped from further consideration.

Faults that pass the above check are simulated under the proposed fault simulation procedure. The overall structure of the procedure is given below as Procedure 1. Each one of the steps of Procedure 1 is described next.

Procedure 1: Fault simulation for fault f

(1) For every present-state variable y_i and every time unit u such that y_i is unspecified at time unit u in the faulty circuit, collect information regarding backward implications at time unit $u-1$, including fault detections, conflicts and additional state variables that become specified.

(2) Check if the information collected in Step 1 is sufficient to conclude that f is detected. If f is detected, stop.

(3) Select the best state variables and time units for state expansion and perform the expansions followed by backward implications.

(4) Fault simulate the test sequence after state expansion and backward implications and check if f is detected.

3.1 Collecting backward implications

In this step of the procedure, we collect information about pairs of time units and present-state variables (u, i) such that state expansion using u and y_i may help establish fault detection. We collect the information for each pair separately, starting in every case from the state and output sequences computed for the faulty circuit using conventional simulation.

We consider every time unit $0 < u < L$ and present-state variable y_i such that y_i is unspecified at time unit u and $N_{out}(u-1) > 0$ (i.e., there are unspecified output values at time unit $u-1$ or afterwards that can be specified to establish fault detection). For $\alpha \in \{0,1\}$, we perform the following procedure. We set next-state variable Y_i corresponding to y_i to α at time unit $u-1$ and perform implications at time unit $u-1$. We then record one of the following results (the first one that applies is recorded).

(1) Backward implications result in a conflict. We record this fact by setting a variable $confl(u, i, \alpha)$ to 1.

(2) Backward implications assign a value β to a primary output o at time unit $u-1$, and in the fault free circuit, T results in the value $\bar{\beta}$ on o at time unit $u-1$. We conclude in this case that the fault is detected for $y_i = \alpha$. We record this fact by setting a variable $detect(u, i, \alpha)$ to 1.

(3) If backward implications do not result in a conflict or in fault detection, we record in a set $extra(u, i, \alpha)$ all the present state variables that become specified at time unit u when next-state variable Y_i is set to α at time unit $u-1$. We have

$extra(u, i, \alpha) = \{(j, \beta)$: $Y_j = \beta$ *when* $Y_i = \alpha$ *at time unit u-1*$\}$.

The set $extra(u, i, \alpha)$ contains the pair (i, α). We also record in a variable $N_{extra}(u, i, \alpha)$ the size of the set $extra(u, i, \alpha)$.

The procedure above does not apply to state expansion at time unit $u = 0$, where backward implication is not required. We observe that state expansion at time unit $u = 0$ is possible, however, it cannot result in conflicts or detection at time unit $u-1$, and it cannot result in additional specified present-state variables at time unit 0. To allow state expansion at time unit $u = 0$, we add the following information to the information collected above. For every present-state variable y_i and every value $\alpha \in \{0,1\}$, we set $confl(0, i, \alpha) = 0$, $detect(0, i, \alpha) = 0$, $extra(0, i, \alpha) = \{(i, \alpha)\}$ and $N_{extra}(0, i, \alpha) = 1$.

3.2 Identifying detected faults

Consider a present-state variable y_i at time unit u such that $confl(u, i, \alpha) = 1$ or $detect(u, i, \alpha) = 1$ for some $\alpha \in \{0,1\}$. Suppose in addition that $detect(u, i, \bar{\alpha}) = 1$. In this case, setting next-state variable Y_i to every possible value at time unit $u-1$ results either in a conflict (indicating that the value is not possible) or in detecting the fault. Thus, we can conclude that the fault is detected in this case. We perform this check for every unspecified state variable at every time unit $0 < u < L$ where $N_{sv}(u) > 0$ and $N_{out}(u) > 0$.

3.3 Selecting pairs for expansion

State variables and time units for expansion are selected in two phases, described next. The starting point for state expansion are the state and output sequences obtained by conventional simulation of the test sequence T.

In the first phase (Step 2 of Procedure 2 below), we select every pair (u, i) for which $confl(u, i, \alpha) = 1$ or $detect(u, i, \alpha) = 1$ for $\alpha \in \{0,1\}$. Expanding y_i at time unit u, we need only consider the states where $y_i = \bar{\alpha}$. This can be accomplished by setting state variable y_i to $\bar{\alpha}$ in the state sequence obtained after conventional simulation. Thus, in this case, state expansion does not increase the number of state sequences that need to be considered. We use the set $extra(u, i, \bar{\alpha})$ to update the faulty circuit state at time unit u with all the values implied by setting $y_i = \bar{\alpha}$ at time unit u.

In the second phase of the state expansion process (Steps 3-8 of Procedure 2 below), we select a limited number of pairs (u, i) for which both specified values must be considered. After selecting a pair (u, i), we immediately duplicate each state

sequence and we assign the additional values according to $extra(u, i, 0)$ in one copy, and according to $extra(u, i, 1)$ in the other copy. Thus, the selection process is sequential. We select additional pairs for expansion until the total number of state sequences reaches a predetermined constant N_{STATES}.

We select the pairs (u, i) based on the information collected in Section 3.1, where each pair was considered separately. To ensure that each pair has maximal effect on the number of specified values in the faulty circuit, we impose the following constraint. We define $sv(u, i) = \{j: (j, \beta) \in extra(u, i, \alpha)$ *for* $\alpha, \beta \in \{0, 1\}\}$, i.e., $sv(u, i)$ is the set of state variables whose values at time unit u are determined when present-state variable y_i is set to either 0 or 1 at time unit u. We allow the pair (u, i) to be selected for state expansion only if all the present-state variables in $sv(u, i)$ are unspecified at time unit u in all the state sequences under consideration.

During an iteration of the state expansion process, one of the pairs (u, i) that satisfies the constraint above is selected according to the following criteria. The criteria are listed in order of importance.

(1) u is such that $N_{out}(u)$ is maximum over all the time units with $N_{out}(u) > 0$ and $N_{sv}(u) > 0$. This ensures that there is a maximum number of output values that can still be specified to establish that the fault is detected.

(2) u is such that $N_{sv}(u)$ is minimum. This ensures that a maximum number of state variables are already specified at time unit u, and it is likely that additional specified state variables will result in specifying output values (this heuristic was also used in [4]).

(3) The minimum of $N_{extra}(u, i, 0)$ and $N_{extra}(u, i, 1)$ is as large as possible.

(4) The maximum of $N_{extra}(u, i, 0)$ and $N_{extra}(u, i, 1)$ is as large as possible. The last two criteria ensure that expansion results in as many specified state variables as possible.

The overall expansion procedure is summarized next. The procedure maintains a set of state sequences S. For a given sequence S', we denote by $S'[u][i]$ the value of present-state variable y_i at time unit u under S'.

Procedure 2: State expansion

(1) Let $S = \{S_0\}$, where S_0 is the sequence of states that the faulty circuit goes through under conventional simulation of the test sequence T.

(2) For every pair (u, i) such that $confl(u, i, \alpha) = 1$ or $detect(u, i, \alpha) = 1$ for $\alpha \in \{0,1\}$:
  For every pair $(j, \beta) \in extra(u, i, \bar{\alpha})$:
    Set $S_0[u][j] = \beta$.

(3) Let E be the set of all pairs (u, i) such that the following condition is satisfied:
  For every sequence $S' \in S$, $S'[u][j]$ is unspecified for every $j \in sv(u, i)$.

(4) Let $N_{out,max} = \max\{N_{out}(u): (u, i) \in E$ *for some* $i\}$.
  Remove from E every pair (u, i) for which $N_{out}(u) < N_{out,max}$.

(5) Let $N_{sv,min} = \min\{N_{sv}(u): (u, i) \in E$ *for some* $i\}$.
  Remove from E every pair (u, i) for which $N_{sv}(u) > N_{sv,min}$.

(6) Let $N_{extra,0} = \max\{N_{extra}(u, i, 0): (u, i) \in E\}$,
  $N_{extra,1} = \max\{N_{extra}(u, i, 1): (u, i) \in E\}$, and
  $N_{extra,A} = \min\{N_{extra,0}, N_{extra,1}\}$.
  Remove from E every pair (u, i) for which $\min\{N_{extra}(u, i, 0), N_{extra}(u, i, 1)\} < N_{extra,A}$.
  Let $N_{extra,0} = \max\{N_{extra}(u, i, 0): (u, i) \in E\}$,
  $N_{extra,1} = \max\{N_{extra}(u, i, 1): (u, i) \in E\}$, and
  $N_{extra,B} = \max\{N_{extra,0}, N_{extra,1}\}$.
  Remove from E every pair (u, i) for which $\max\{N_{extra}(u, i, 0), N_{extra}(u, i, 1)\} < N_{extra,B}$.

(7) Select any pair $(\tilde{u}, \tilde{i}) \in E$.

(8) For every sequence $S' \in S$:
  (a) Copy S' into a sequence S'' and include both S' and S'' in S.
  (b) For every $(j, \beta) \in extra(\tilde{u}, \tilde{i}, 0)$, set $S'[\tilde{u}][j] = \beta$.
  (c) For every $(j, \beta) \in extra(\tilde{u}, \tilde{i}, 1)$, set $S''[\tilde{u}][j] = \beta$.

(9) If the number of sequences in S is N_{STATES}, stop.

(10) Go to Step 3.

3.4 Fault simulation after expansion

During state expansion by Procedure 2, we mark every time unit where one or more state variables were specified. For every sequence of states $S' \in S$ we then resimulate these time units. Additional time units are marked and consequently simulated if they have newly specified state variables. The resimulation procedure is described next.

For every state sequence S', we consider every time unit u starting from $u = 0$. Every marked time unit is simulated as follows. To simulate a state sequence S' at time unit u, we use the input values from T and the present-state variable values from $S'[u]$. For these input values, we compute an output pattern z and a next-state s'. We then check the following conditions. First, we compare z with the output pattern of the fault-free circuit at time unit u. If the two output patterns conflict (i.e., there is an output where they are specified to different {0,1} values), the fault is detected for state sequence S', and S' is dropped from further consideration. If the fault is not detected for S', we compare the next-state s' with $S[u + 1]$. If there is a conflict between s' and $S[u + 1]$ (i.e., $s'[i] \neq S[u + 1][i]$ for a state variable i where both are specified), we conclude that S' is not a feasible state sequence and, again, S' is dropped from further consideration. If no conflict is found, we check whether $s'[i]$ is specified for some state variable i for which $S[u + 1][i]$ is unspecified. We set $S[u + 1][i] = s'[i]$ for every such state variable, and mark time unit $u + 1$ to indicate that it has to be simulated. Simulation of a sequence S' stops when all the marked time units have been simulated or if a conflict or detection were identified. For a fault to be considered detected, either detection or a conflict have to be established for every sequence $S' \in S$.

4. Experimental results

We applied the proposed simulation procedure to ISCAS-89 benchmark circuits and to circuits from [8] using a limit of 64 on the number of state sequences of the faulty circuit after expansion. The results are shown in Table 2(a) as follows. After circuit name we show the total number of faults. We then show the number of faults detected by conventional simulation. The results of the procedure from [4] with the same limit of 64 on the number of state sequences after state expansion are shown next. In the last two columns of Table 2(a) we show the results of the proposed procedure. For each procedure we show the total number of faults detected and the number of faults detected beyond conventional simulation. All the faults identified as detected in [4] are also identified by the proposed procedure. Moreover, in many circuits, additional faults can be detected by using backward implications, compared to the numbers of faults detected in [4]. For example, for *s*5378, 11 additional faults are identified as detected by the proposed procedure, whereas no additional faults were identified by the procedure of [4]. In the case of the larger circuits, the procedure from [4] could not be applied whereas we are able to apply the proposed procedure to detect additional

faults.

Table 2: Results using random patterns

circuit	total faults	detected faults conv.	[4] tot	[4] extra	proposed tot	proposed extra
s208	215	73	86	13	86	13
s298	308	143	150	7	150	7
s344	342	314	320	6	320	6
s420	430	125	150	25	150	25
s641	467	343	347	4	347	4
s713	581	415	419	4	419	4
s1423	1515	331	338	7	338	7
s5378	4603	2352	2352	0	2363	11
s15850	11725	85	NA	NA	87	2
s35932	39094	22357	NA	NA	22367	10
am2910	2573	1234	1259	25	1272	38
mp1_16	1708	1259	1278	19	1280	21
mp2	10477	666	670	4	676	10

It is interesting to note that for *s*5378, all the faults detected by the proposed procedure were aborted by the procedure of [4] when the number of state expansions reached the limit of 64 states. This demonstrates the effectiveness of using backward implications to complement state expansion. As another indication of the effectiveness of backward implications, we collected the following information. For each fault, we maintained three counters, called $N_{confl}(f)$, $N_{det}(f)$ and $N_{extra}(f)$. The counters were initially set to zero. They were incremented during the simulation procedure according to the state variables and time units selected for expansion. Suppose that present-state variable y_i at time unit u was selected for expansion. Then the following rules were applied.

• If setting Y_i at time unit $u-1$ to $\alpha \in \{0,1\}$ resulted in fault detection (i.e., $detect(u,i,\alpha)=1$), then $N_{det}(f)$ was incremented by one and $N_{extra}(f)$ was incremented by $N_{extra}(u,i,\bar{\alpha})$.

• If setting Y_i at time unit $u-1$ to $\alpha \in \{0,1\}$ caused a conflict (i.e., $confl(u,i,\alpha)=1$), then $N_{confl}(f)$ was incremented by one and $N_{extra}(f)$ was incremented by $N_{extra}(u,i,\bar{\alpha})$.

• In all other cases, $N_{extra}(f)$ was incremented by $N_{extra}(i,u,\alpha)+N_{extra}(i,u,\bar{\alpha})$.

In Table 3 we show the averages of the counts above, where the averages are computed over all the faults detected by the proposed method. Without backward implications, the number of detections and the number of conflicts $N_{det}(f)$ and $N_{confl}(f)$ are zero. The number of state variables whose values are specified with every state expansion is two (only the selected state variable is assigned two different values, 0 and 1). Since we allow at most six state expansions per fault until the number of state sequences reaches 64, the value of $N_{extra}(f)$ is at most 12. The values of Table 3 are significantly larger, indicating that backward implications resulted in many additional specified values. In addition, significant numbers of detections and conflicts are obtained.

Using the deterministic test sequence generated by *HITEC* [9] for *s*5378, the proposed method detected 14 additional faults, whereas the procedure of [4] detected 12 additional faults.

5. Concluding remarks

We presented an improved procedure for fault simulation under the multiple observation time approach. The procedure was based on state expansion. The proposed approach used backward implications to take advantage of every present state variable value specified under state expansion. Thus, when state variable y_i was set to a value $\alpha \in \{0,1\}$ at time unit u, the next-state variable Y_i corresponding to y_i was set to α at time unit $u-1$, and implications of this assignment were computed at time unit $u-1$. Several results are possible. (1) A conflict at time unit $u-1$ indicates that y_i can only assume the value $\bar{\alpha}$ at time unit u. (2) The assignment may help detect the fault under consideration by assigning one or more primary output values at time unit $u-1$. (3) Additional next-state variables may be specified at time unit $u-1$, allowing additional present-state variables at time unit u to be specified. Consequently, fewer states need to be considered after state expansion, fewer state expansions are potentially needed for every fault, and the number of faults that can be efficiently considered is increased. Experimental results were presented to support this claim for the case where state expansion is done only in the faulty circuit. This case is important as it is consistent with the conventional test application scheme.

Table 3: Effectiveness of backward implications

circuit	detect	confl	extra
s208	19.54	12.00	54.54
s298	6.71	36.57	60.71
s344	281.67	0.00	304.33
s420	24.88	7.60	57.60
s641	234.25	0.00	400.75
s713	178.75	0.00	219.75
s1423	10.29	91.71	195.71
s5378	616.18	142.00	1082.27
s15850	114.00	89.00	264.50
s35932	5958.00	0.00	6711.60
am2910	225.79	8.53	331.29
mp1_16	2038.57	25.38	2096.05
mp2	2996.50	50.10	3449.00

References

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

[2] I. Pomeranz and S. M. Reddy, "The Multiple Observation Time Test Strategy", IEEE Trans. on Computers, May 1992, pp. 627-637.

[3] H. Cho, S.-W. Jeong, F. Somenzi and C. Pixley, "Synchronizing Sequences and Symbolic Traversal Techniques in Test Generation", J. of Electronic Testing: Theory and Applications, Vol. 4, Feb. 1993, pp. 19-31.

[4] I. Pomeranz and S. M. Reddy, "On Fault Simulation for Synchronous Sequential Circuits", IEEE Trans. on Computers, Feb. 1995, pp. 335-340.

[5] R. Krieger, B. Becker and M. Keim, "A Hybrid Fault Simulator for Synchronous Sequential Circuits", 1994 Intl. Test Conf., Oct. 1994, pp. 614-623.

[6] I. Pomeranz and S. M. Reddy, "Low-Complexity Fault Simulation under the Multiple Observation Time Testing Approach", in Proc. 1995 Intl. Test Conf., Oct. 1995, pp. 272-281.

[7] E. M. Rudnick, J. H. Patel and I. Pomeranz, "On Potential Fault Detection in Sequential Circuits", in Proc. 1996 Intl. Test Conf., Oct. 1996.

[8] E. M. Rudnick, "Simulation Based Techniques for Sequential Circuits," Ph.D. Dissertation, also Tech. Rept. CRHC-94-14, Center for Reliable and High Performance Computing, Univ. of Illinois, Aug. 1994.

[9] T. Niermann and J. H. Patel, "HITEC: A Test Generation Package for Sequential Circuits", European Design Autom. Conf., 1991, pp. 214-218.