Symbolic Fault Simulation for Sequential Circuits and the Multiple Observation Time Test Strategy

R. Krieger B. Becker M. Keim
Computer Science Department
J.W. Goethe-University
Frankfurt am Main, Germany

Abstract— **Fault simulation for synchronous sequential circuits is a very time-consuming task. The complexity of the task increases if there is no information about the initial state of the circuit. In this case an unknown initial state is assumed which is usually handled by introducing a three-valued logic. As it is well-known fault simulation based on this logic only determines a lower bound of the fault coverage. Recently it has been shown that fault simulation based on the multiple observation time test strategy can improve the accuracy of the fault coverage. In this paper we describe how this strategy can be successfully implemented based on Ordered Binary Decision Diagrams. Our experiments demonstrate the efficiency of the fault simulation procedure developed.**

I. Introduction

Despite numerous attempts to create automatic test pattern generators capable of testing sequential logic this problem remains inherently intractable if a three-valued logic is used. The major cause of the difficulty is the unknown initial state of the circuit. By using the concept of synchronizing sequences, it is shown in [11] that entire classes of testable sequential circuits exist which cannot be tested by algorithms based on the three-valued logic. With regard to fault simulation the unknown initial state prevents from computing the fault coverage accurately. Based on the three-valued logic only a lower bound for the fault coverage can be determined. An improvement of the accuracy either requires that functional information is provided to the fault simulation or that circuit modifications are made to permit setting the circuit into a known initial state. But in general the information or modifications only concern the fault-free circuit. Consequently, the computation of synchronizing sequences for the faulty circuits remains intractable using the three-valued logic. Therefore in [8] a hybrid fault simulator is proposed which improves the accuracy by performing a symbolic simulation based on Ordered Binary Decision Diagrams (*OBDD*s) [4]. In many cases this simulation strategy improves the accuracy considerably but until now it determines the fault coverage only with respect to the single observation time test strategy (*SOT*) which is inaccurate in itself.

To overcome the limitation of *SOT* a more general definition of detectability has to be considered [1]. This definition leads to the multiple observation time test strategy (*MOT*) which is used in [12, 14] to increase the efficiency of test generation. In [5, 9] test generation with respect to the *MOT* strategy is combined with *OBDD*-based techniques. From "ordinary" test generation it is well-known that test generation has to be accelerated by use of an efficient fault simulator. Of course, *MOT*-based test generation should be supported by a *MOT*-based fault simulation to obtain the full power of the *MOT* strategy and to improve on the time and space behavior. Until now, only Pomeranz and Reddy realized this necessity and proposed a three-valued fault simulator based on the *MOT* strategy in [13]. For the "complete" *MOT* strategy it is necessary to compare the sets of fault-free responses with the set of responses obtained in the presence of faults (for all possible states of the circuit). This is especially complicated if long test sequences and a large number of memory elements exist. To overcome these problems several methods are presented in [13], which basically try to reduce the size of the response set (for the fault-free circuit and for the faulty circuit). This leads to a "less accurate" *MOT* test which nevertheless is more accurate than *SOT* in many cases.

In this paper we extend the hybrid fault simulation as proposed in [8] to also work for the *MOT* strategy and thereby avoid the explicit state enumeration. At first, we describe a procedure that identifies faults which cannot be detected by a conventional fault simulation based on the three-valued logic and the *SOT* strategy. The procedure works in polynomial time. Eliminating faults identified by this procedure can accelerate the three-valued fault simulation considerably. We then show how the *MOT* strategy can be integrated efficiently in the *OBDD*-based part of the hybrid fault simulator. In order to reduce the complexity of the *MOT* approach a three-valued fault simulation is carried out first. During the *MOT*-based fault simulation only faults which are not detected by the three-valued fault simulation are considered. We study the improvement in terms of detected faults and for the first time we also investigate the performance of a *MOT*-based fault simulation. By the use of *OBDD*s it becomes possible to handle a large number of output sequences. Thus, we succeed in computing the *exact MOT* fault coverage for many of the considered benchmark circuits. In case that the space requirements of the *OBDD*-based approach exceed a given limit which is determined by the working environment the hybrid fault simulator changes to the *SOT* strategy based on the three-valued logic for some simulation steps as described in [8] and then again returns to the symbolic evaluation and the *MOT* strategy. This guarantees that the *MOT* strategy can be applied even to large circuits. We also show that the *MOT* strategy is

32nd ACM/IEEE Design Automation Conference ®

very attractive if the circuit considered has a reset state or if it is restricted to outputs with a well-defined value in the fault-free case ($rMOT$). The usefulness of the $rMOT$ strategy in the area of test generation for fully synchronizable circuits was already investigated in [5]. In contrast to the general MOT strategy the $rMOT$ strategy allows a test evaluation by comparing the output sequence of the circuit-under-test with the unique output sequence of the fault-free circuit as usually. We show that the accuracy of fault simulation based on $rMOT$ is identical with that based on MOT for many circuits. With regard to the performance it can be observed that for some circuits a symbolic $rMOT$ fault simulation works even more efficiently than a symbolic SOT fault simulation. Clearly, compared with a three-valued fault simulation the increase of accuracy must be paid for with an increase of the execution time.

The paper is structured as follows: We start in Section II by presenting some definitions and important properties of synchronous sequential circuits. In Section III we describe the procedure for the identification of faults which cannot be detected by the test sequence due to the inaccuracy of the three-valued logic. In Section IV we show how fault simulation based on the MOT strategy can be realized by the use of $OBDD$s. In Section V experimental results are given, which demonstrate the efficiency of the fault simulation procedure with respect to the MOT strategy. Finally we draw some conclusions.

II. Definitions

A synchronous sequential circuit can be considered to be a *Finite State Machine* (FSM) as introduced in [7].

Definition 1: A finite state machine M is defined as a quintuple $M = (I, O, S, \delta, \lambda)$, where I is the input set, O is the output set and S is the set of states, $\delta : S \times I \rightarrow S$ is the next state function, and $\lambda : S \times I \rightarrow O$ is the output function.

Since we consider a gate level realization of the FSM, we have $I = \mathbf{B}^k$, $O = \mathbf{B}^l$ and $S = \mathbf{B}^m$ with $\mathbf{B} = \{0, 1\}$. k denotes the number of primary inputs, l the number of primary outputs and m the number of memory elements. The functions δ and λ are computed by a combinational circuit. The inputs (outputs) of this circuit which are connected to the outputs (inputs) of the memory elements are called secondary inputs (outputs).

For the description of fault simulation we use the following notations: $Z = (z(1), \ldots, z(n))$ denotes an input sequence of length n. $z_i(t)$, $1 \leq i \leq k$, is the value that is assigned to the i-th primary input before starting simulation at time t, $1 \leq t \leq n$. $S(p, Z) = (s(p, 0), s(p, 1), \ldots, s(p, n))$ denotes the state sequence defined by the initial state $p = s(p, 0)$ and Z. $s_i(p, t)$, $1 \leq i \leq m$, is the state of the i-th memory element after simulation step t. $O(p, Z) = (o(p, 1), o(p, 2), \ldots, o(p, n))$ denotes the output sequence defined by the initial state p and Z. $o_i(p, t)$, $1 \leq i \leq l$, is the value at the i-th primary output after simulation step t. Using these notations the next state is given by

$$s(p,t) \quad = \quad \begin{cases} p & \text{if } t = 0 \\ \delta(s(p, t-1), z(t)) & \text{otherwise} \end{cases}$$

Analogously, the output $o(p, t)$ is defined by the function λ.

A single stuck-at fault f transforms a machine M into a machine $M^f = (I, O, S, \delta^f, \lambda^f)$. The functions δ^f and λ^f are defined analogously.

If there is no knowledge about the initial state of the circuit using the SOT strategy the detectability of a single stuck-at fault with respect to an input sequence Z can be defined according to [1] as follows:

Definition 2: A single stuck-at fault is detectable with respect to the SOT strategy by an input sequence $Z = (z(1), z(2), \ldots, z(n))$ if $\exists b \in \{0, 1\}, \exists t \leq n, \exists i \leq l, \forall p, q$ $o_i(p, t) = b$ and $o_i^f(q, t) = \overline{b}$ with p an initial state of the fault-free circuit and q an initial state of the faulty circuit.

As shown in [11] fault simulation based on the three-valued logic determines a lower bound for the fault coverage with respect to the definition given above. The exact fault coverage can be determined by using the $OBDD$-based symbolic fault simulation as presented in [8]. In many cases the application of the symbolic techniques leads to more accurate fault coverages than that obtained by using the three-valued logic.

III. Elimination of X-redundant Faults

In this section we describe a procedure that efficiently identifies faults which are undetectable with respect to a given test sequence due to the inaccuracy of the three-valued logic and the limitation of the SOT strategy. We denote these faults as X-redundant faults as proposed in [2]. Eliminating these faults before starting three-valued fault simulation leads to a considerable speed-up. Similar methods are used to accelerate test generation. For instance the testability measure $SCOAP$ [6] allows the identification of such faults. An extension of this approach is described in [15]. All these methods identify faults that cannot be detected by any test sequence due to the inaccuracy of the three-valued logic. Using the fact that in case of fault simulation a test sequence is given we developed a more sophisticated procedure called *ID_X-red*. It works in three steps.

Step 1: For the given test sequence a true value simulation of the circuit using the three-valued logic is performed. For each lead the values assigned to the lead during simulation are stored. The result of this simulation is encoded into a four-valued logic containing the elements $\{X\}$, $\{X, 0\}$, $\{X, 1\}$, $\{X, 0, 1\}$. The value $\{X\}$ means that a lead always has value X, i.e. the lead does neither assume the value 0 nor the value 1 during simulation. The other logic elements are interpreted analogously. The value assigned to a lead l in this step is denoted by $I_X(l)$.

Step 2: A backward pass is started from the primary and secondary outputs towards the primary and secondary inputs. Thereby the I_X-values of the leads are recomputed. During the recomputation we distinguish between fanouts and gates. Let l be a fanout stem with branches l_1 and l_2. Then we perform the assignment $I_X(l) \leftarrow \{X\}$ if and only if $I_X(l_1)$ and $I_X(l_2)$ are equal to $\{X\}$. In case of gates the I_X-values of the inputs are set to $\{X\}$ if the output of the gate has the value $\{X\}$. After this step a lead has the value $\{X\}$ if either the lead does not assume the value 0 or 1 during the simulation performed in step 1 or if every path from the lead to a primary or secondary output contains a lead l with $I_X(l) = \{X\}$. The backward pass is iterated until no changes occur at the secondary inputs.

Step 3: A backward traversal inside the fanout-free regions is performed that determines the observability OB of each lead inside the region at the output of the region. The backward traversal starts at the output l of the region by set-

ting $OB(l) = 0$ if $I_X(l) = \{X\}$ and $OB(l) = 1$ otherwise. The observabilities at the inputs of a gate are determined as follows: An input of an *AND* gate is observable if its output is observable and if the other inputs of the gate have the values $\{X, 1\}$ or $\{X, 0, 1\}$. An input of an *OR* gate is observable if its output is observable and if the other inputs have the values $\{X, 0\}$ or $\{X, 0, 1\}$. An input of an inverter is observable if its output is observable.

Step 4: Faults which cannot be detected by the test sequence are identified by checking the following sufficient condition: A stuck-at 0 fault at l (stuck-at 1 fault at l) is undetectable if $I_X(l) = \{X, 0\}$ ($I_X(l) = \{X, 1\}$) or $I_X(l) = \{X\}$ holds or if $OB(l) = 0$ holds.

For shortness of the paper the correctness proof of the procedure described above is omitted.

The procedure has the following run time behaviour. The simulation in step 1 can be performed in time $O(|C| \cdot |Z|)$ where $|C|$ is the circuit size and $|Z|$ is the length of the test sequence. Steps 2-4 can be performed in $O(|C|)$. Therefore the procedure works very efficiently and its run time is negligible compared with the run time needed for three-valued fault simulation. If it is performed before starting the three-valued fault simulation, the number of faults which have to be considered can be reduced for a lot of circuits.

IV. Multiple Observation Time

Faults which are eliminated by the procedure described in the previous section cannot be detected by a fault simulation based on the three-valued logic. On the one hand this fact can be explained by the inaccuracy of the three-valued logic. In [8] a hybrid fault simulation is described which eliminates this inaccuracy. Nevertheless there are some circuits for which this fault simulator does not achieve a reasonable accuracy because the simulator is based on the *SOT* strategy. This strategy shows several limitations with respect to the detection of faults by a test sequence. For illustration consider Fig. 1 which is taken from [13]. It shows two fault simulation steps

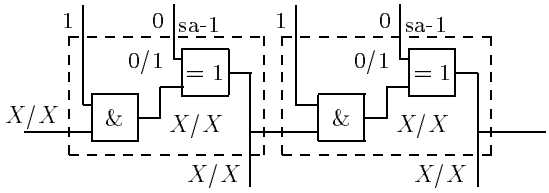

Fig. 1. Stuck-at fault not detected with respect to the *SOT* strategy.

for the test sequence ([1, 0], [1, 0]). As it can easily be seen the simulated fault cannot be detected with respect to the *SOT* strategy. Even if the fault-free circuit is initialized similar effects can occur as outlined in Fig. 2. As it can be seen, this sequence drives the fault-free circuit in a defined state but not the faulty circuit. Therefore the fault is undetectable with respect to definition 2. To overcome the limitations of the *SOT* approach the multiple observation time test (*MOT*) strategy can be applied.

Definition 3: A fault is detectable with respect to the multiple observation time strategy by an input sequence Z if for all states p and q there are $t \leq n$, $i \leq l$ and $b \in \{0, 1\}$ such

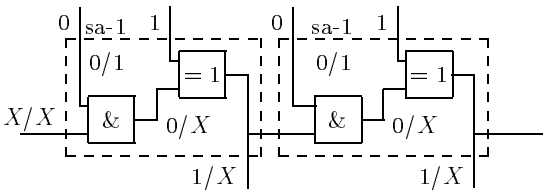

Fig. 2. Stuck-at fault not detected with respect to the *SOT* strategy despite of initialization.

that we get $o_i(p, t) = b$ and $o_i^f(q, t) = \overline{b}$. ($p$ denotes an initial state of the fault-free circuit and q an initial state of the faulty circuit.)

This means a fault is detectable if the output sequences determined by the fault-free and faulty circuits are different for any pair of initial states. In [13] fault simulation based on this approach is implemented by using the three-valued logic in combination with an enumeration of a subset of all possible initial states. To reduce the complexity it is proposed to use the value X as initial value of a latch if this does not cause a negative effect on the fault detection. Nevertheless the main disadvantage of this implementation is the explicit state enumeration. The authors restrict their enumeration technique to less than 6 memory elements which corresponds to 64 different initial states. A way to overcome this limitation is presented in the sequel. We define the detection function $D_{f,Z} : \mathbf{B}^m \times \mathbf{B}^m \rightarrow \mathbf{B}$ by

$$D_{f,Z}(x, y) := \prod_{t=1}^{n} \prod_{j=1}^{l} [o_j(x, t) \equiv o_j^f(y, t)]$$

for each fault f and test sequence Z. x and y denotes the initial states of the fault-free and faulty circuit, respectively.

Lemma 1: A fault f is detectable by the input sequence $Z = (z(1), z(2), \ldots, z(n))$ according to Definition 3 iff

$$D_{f,Z} \equiv \mathbf{0}$$

Proof: Let p and q be two arguments with $D_{f,Z}(p, q) \neq 0$. Then it follows from the definition of the detection function $D_{f,Z}$ that for the initial states p and q the output sequence $(o(p, 1), \ldots, o(p, n))$ of the fault-free circuit and the output sequence $(o^f(q, 1), \ldots, o^f(q, n))$ of the faulty circuit are equal. Therefore the fault f is undetectable according to Definition 3. □

To illustrate the application of the *MOT* strategy based on the detection function consider the circuit shown in Fig. 3. In

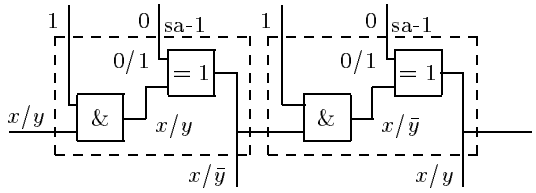

Fig. 3. Example.

order to check whether the test sequence detects the fault f located at the second primary input of the circuit we have to

compute its detection function. In doing so we determine the output sequence of the fault-free and faulty circuits. Obviously we get $o_1(x,1) = x$, $o_1^f(y,1) = \bar{y}$, $o_1(x,2) = x$ and $o_1^f(y,2) = y$. According to the definition given above we get for the detection function $D_{f,Z}$

$$\begin{aligned} D_{f,Z}(x,y) &= [x \equiv \bar{y}] \cdot [x \equiv y] \\ &= 0 \quad \forall x,y \end{aligned}$$

Consequently, with Lemma 1 and the MOT strategy it follows that the fault can be detected by the test sequence.

A. Fault Simulation

To perform fault simulation based on the MOT strategy we introduce variables x_i for each memory element of the circuit. These variables are used for the encoding of the unknown initial state which is defined by $x = [x_1, \ldots, x_m]$. Moreover for each fault a detection function $\tilde{D}_{f,Z}$ is defined. It is initially set to the constant function **1**. During a symbolic fault simulation this function will be used for the construction of the detection function $D_{f,Z}$. At the end of fault simulation it holds $\tilde{D}_{f,Z} = D_{f,Z}$. More in detail, the symbolic fault simulation works as follows: First a symbolic true value simulation is carried out which determines the symbolic value of each lead implied by the test vector and the present state vector. Subsequently an explicit fault simulation is carried out. For each fault f and time frame t $o^f(x,t)$ and $s^f(x,t)$ are determined by an event-driven single fault propagation. This means that the faults are injected one by one. The effects are propagated towards the primary outputs and the memory elements. If in course of this process the i-th primary output is reached we check whether the fault is observable. Depending on the different observation time test strategies there are three possibilities:

1. SOT Test Strategy: The fault is marked as detectable if $o_i(x,t), o_i^f(x,t) \in \{\mathbf{0},\mathbf{1}\}$ and $o_i(x,t) \not\equiv o_i^f(x,t)$.
2. Restricted MOT ($rMOT$) Test Strategy: We compute

   $$\tilde{D}_{f,Z}(x,x) \leftarrow \tilde{D}_{f,Z}(x,x) \cdot [o_i(x,t) \equiv o_i^f(x,t)]$$

   if $o_i(x,t) \in \{\mathbf{0},\mathbf{1}\}$. If $\tilde{D}_{f,Z}$ is evaluated to **0** the fault is marked as detectable.
3. MOT Test Strategy: First we have to compute $o_i^f(y,t)$ for all $i, 1 \le i \le l$, where $y = [y_1, \ldots, y_m]$ denotes the unknown initial state of the faulty circuit. Since the $OBDD$-representation for $o_i^f(x,t)$ is given this can efficiently be done by a compose operation. (Another possibility to compute $o_i^f(y,t)$ would be to initialize the state of the faulty circuit with y at the beginning of the fault simulation. But in doing so we would not profit by the event-driven single fault propagation.) Then we compute

   $$\tilde{D}_{f,Z}(x,y) \leftarrow \tilde{D}_{f,Z}(x,y) \cdot \prod_{i=1}^{l} [o_i(x,t) \equiv o_i^f(y,t)]$$

   If $\tilde{D}_{f,Z}$ is evaluated to **0** the fault is marked as detectable.

Of course faults which are marked as detectable are dropped and will not be considered during following simulation steps. If the symbolic values are represented by Ordered Binary Decision Diagrams, the fault simulation procedure as described above works efficiently for many circuits. Moreover the integration of this procedure in the hybrid fault simulator allows the application of the MOT strategy even to large circuits. If the space requirements of the symbolic fault simulation exceed a given limit the hybrid fault simulator changes to the SOT strategy and works as described in [8]. After a few simulation steps using the three-valued logic which can reduce the space requirements of the symbolic simulation the hybrid fault simulator returns to the MOT strategy again. In doing so the detection function $\tilde{D}_{f,Z}$ has to be re-initialized with the constant function **1**.

The difference between MOT and $rMOT$ lies in the space requirements and in the accuracy which can be achieved. MOT works more accurately than $rMOT$. On the other hand the space requirement of MOT is larger than that of $rMOT$ because MOT requires different variables for encoding the initial state of the fault-free and the faulty circuit. A further important advantage of rMOT is that it allows the standard test evaluation.

B. Test Evaluation

The problem of test evaluation with respect to a given input sequence Z can be described as follows: Let $(c(1), \ldots, c(n))$ be the output sequence which is obtained by applying Z on the circuit-under-test. Decide whether or not the circuit-under-test is faulty.

In case of a test sequence which is determined with respect to the SOT approach or the $rMOT$ approach the test evaluation can easily be done. The circuit-under-test is faulty if there are $t \le n$ and $i \le l$ with $c_i(t) \not\equiv o_i(x,t)$ where $o_i(x,t) \in \{\mathbf{0},\mathbf{1}\}$ denotes the value at the i-th primary output after simulation step t. In case of a test sequence which is determined with respect to the MOT approach the test evaluation is more complicated. The implementation proposed in [13] requires to check whether the output sequence $(c(1), \ldots, c(n))$ is contained in a set of output sequences which can be computed by a fault-free circuit depending on the initial state. Since the number of output sequences may be exponential in the number of memory elements test evaluation may be very time-consuming. To reduce the time requirements we propose the comparison of the sequence $c(1), \ldots, c(n)$ with the symbolic representation of the output sequence $o(1), \ldots, o(n)$. The comparison can be done by evaluating step by step the product

$$\prod_{t=1}^{n} \prod_{j=1}^{l} [o_j(x,t) \equiv c_j(t)]$$

If the result of this computation is **0** the circuit-under-test is faulty. Obviously, the tractability of this approach depends on the size of the symbolic representation and the time needed for evaluating the product of the symbolic output values. It turns out that, if the symbolic representation is based on $OBDD$s, both space and time are of moderate size for many circuits.

V. Experimental Results

To investigate the performance and the accuracy of our $OBDD$-based approach and the advantage of the MOT strategy we implemented the fault simulation procedures as previously described. For the implementation we used C++. Our measurements were performed on a SUN SPARC station 10 Mod.

Table I: Influence of *ID_X_red* on the run time of three-valued fault simulation for random test sequences of length 200.

Circ.	$\|F\|$	X-red.	$\|F_d\|$	$X01$	$X01_p$	*ID_X-red*
s208.1	217	195	15	1.58	0.09	0.05
s298	308	71	168	1.04	0.91	0.05
s344	342	17	291	1.10	1.10	0.07
s349	350	18	297	1.14	1.10	0.07
s382	399	174	49	2.05	1.64	0.07
s386	384	63	179	0.57	0.48	0.06
s400	424	51	51	2.23	1.76	0.08
s420.1	455	419	22	4.70	0.22	0.11
s444	474	211	53	2.42	1.98	0.08
s510	564	564	0	5.35	0.09	0.10
s526	555	283	48	3.20	2.52	0.10
s641	467	72	345	0.64	0.51	0.10
s713	581	94	417	0.94	0.78	0.13
s820	850	114	236	2.14	2.02	0.18
s832	870	116	235	2.23	2.11	0.20
s838.1	931	867	38	15.11	0.51	0.27
s953	1079	852	90	23.31	1.85	0.24
s1196	1242	31	807	2.11	2.09	0.31
s1238	1355	43	822	2.58	2.46	0.32
s1423	1515	368	333	9.66	8.54	0.43
s1488	1486	51	820	4.31	4.27	0.37
s1494	1506	51	817	4.61	4.48	0.40
s5378	4603	1647	2327	23.68	18.44	1.35
s9234.1	6927	4417	366	183.25	132.21	2.56
s13207.1	9815	7476	858	318.53	67.58	3.85
s15850.1	11725	6138	1645	326.11	223.12	4.61
s35932	39094	4306	22527	267.34	264.94	11.82
s38417	31180	29172	1098	1034.19	183.17	12.07
s38584.1	36303	6634	12585	2321.08	2065.98	20.35

20 with 32 MBytes of memory. For our experiments we considered a subset of the ISCAS-89 benchmark circuits [3]. A space limit of $30,000$ *OBDD* nodes was used to guarantee that our procedures work efficiently. At first we investigated the effect of the procedure *ID_X-red* for eleminating the X-redundant faults in order to accelerate three-valued fault simulation. Table I shows the results. We simulated 200 random vectors. $|F|$ denotes the number of faults. X-red denotes the number of faults that are identified by the procedure *ID_X-red* as undetectable with respect to the three-valued logic and the SOT strategy. On average 38% of the faults are undetectable if a three-valued fault simulation is used. $|F_d|$ denotes the number of faults that are detected by the test sequence using the three-valued logic. In order to investigate the influence of the elimination of the X-redundant faults on the performance of fault simulation the run times for fault simulation without performing *ID_X-red* ($X01$) and with performing *ID_X-red* ($X01_p$) are given. Moreover the run times for *ID_X-red* are given. As it can be seen the application of *ID_X-red* leads to a considerable acceleration of the three-valued fault simulation. Even if the time needed by *ID_X-red* is added to $X01_p$ this acceleration can still be observed for most circuits. If the execution time increases the run time of *ID_X_red* is negligible.

To compare the performance and the accuracy of the different observation strategies we performed a symbolic fault simulation based on the SOT, $rMOT$ and MOT strategies, respectively. Thereby we considered the X-redundant faults and the faults which are not detected by the three-valued fault

Table II: Comparison of SOT with $rMOT$ and MOT for random test sequences of length 200.

			faults detected			CPU time [sec]		
Circ.	$\|F\|$	$\|F_u\|$	SOT	$rMOT$	MOT	SOT	$rMOT$	MOT
s208.1	217	202	0	10	51	47.52	48.26	49.07
s298	308	140	5	6	*6	6.71	7.08	58.94
s344	342	51	4	6	*6	29.84	7.61	336
s349	350	53	4	6	*6	30.13	7.54	307
s382	399	350	0	1	1	31.56	25.81	35.10
s386	384	205	0	0	0	0.58	0.64	0.75
s400	424	373	0	1	1	33.21	27.11	36.62
s420.1	455	433	0	13	*13	533	529	401
s444	474	421	0	1	*1	71.91	64.05	56.37
s510	564	564	395	477	531	507	440	585
s526	555	507	0	*1	*1	95.32	105	101
s641	467	122	4	4	4	1.77	5.64	8.75
s713	581	164	4	4	4	2.15	7.93	11.39
s820	850	641	1	1	1	1.91	2.55	3.68
s832	870	635	1	1	1	1.94	2.65	3.92
s838.1	931	893	*0	*12	*11	1801	1759	1041
s953	1079	989	513	516	516	86.90	116	182
s1196	1242	435	0	0	0	1.39	1.49	1.63
s1238	1355	533	0	0	0	1.77	1.88	2.16
s1423	1515	1182	*2	*6	*6	34.77	51.50	62.18
s1488	1486	666	2	2	2	2.56	3.31	9.82
s1494	1506	689	2	2	2	2.72	3.34	12.59
s5378	4603	2276	*7	*12	*99	115	401	651
$\sum$			944	1082	1263	3441	3618	3957

simulation. Table II shows the results for 200 random test vectors. Results on the largest benchmarks are not listed in the table because the hybrid fault simulator mainly uses the SOT strategy due to the space requirement of $rMOT$ and MOT. $|F|$ denotes the number of faults. $|F_u|$ denotes the number of faults that were not classified as detected by the three-valued fault simulation. For each strategy execution times with respect to these faults and the number of faults marked as detectable are given. Results which were obtained by a temporary change to the three-valued logic during hybrid fault simulation are indicated by an asterisk. Due to the *OBDD*-based simulation all strategies permit a further classification of detectability of faults. In general fault simulation based on the $rMOT$ strategy detects more faults than SOT-based fault simulation. In 14 out of 23 examples we even succeeded in computing the exact MOT fault coverage of the test sequences. In 12 out of these 14 examples already the $rMOT$ strategy computed the exact fault coverage. Only for three of all circuits considered additional faults were detected by MOT (s208.1, s510, s5378). On the other hand, in all cases where MOT was not exact we at least succeeded in improving the accuracy compared to the three-valued fault simulation and the SOT approach as well. In case of s838.1 MOT detects fewer faults than $rMOT$. This can be explained by the space requirement of MOT which is larger than the space requirement of $rMOT$. Due to this fact the hybrid fault simulator based on MOT selects more frequently the fast three-valued fault simulation which works less accurately.

Comparing the execution times for some circuits an acceleration can be observed by using $rMOT$ instead of SOT. Moreover SOT and $rMOT$ are generally faster than MOT.

The same measurements were repeated for deterministic test sequences which are also considered in [10]. Table III shows

Table III: Comparison of SOT with $rMOT$ and MOT for deterministic test sequences.

				faults detected			CPU time [sec]		
Circ.	$\|T\|$	$\|F\|$	$\|F_u\|$	SOT	$rMOT$	MOT	SOT	$rMOT$	MOT
s208.1	111	217	200	0	4	46	35	35	36
s298	162	308	44	4	7	7	3.23	1.73	4.11
s344	91	342	13	4	6	6	3.68	1.08	1.13
s349	91	350	15	4	6	6	3.86	1.07	1.17
s382	2463	399	36	3	12	12	377	22	24
s400	1282	424	73	6	13	13	208	30	35
s420.1	173	455	432	0	10	*6	672	667	417
s510	200	564	564	549	549	549	265	250	380
s526	754	555	137	2	11	11	201	32	41
s641	133	467	64	4	4	4	0.89	2.84	3.57
s713	107	581	111	4	4	4	1.15	3.45	5.14
s820	411	850	154	2	2	2	1.35	1.94	2.41
s832	377	870	162	1	1	1	1.04	1.29	1.58
s953	16	1079	995	132	143	171	27	31	73
s1238	349	1355	72	0	0	0	0.85	0.87	0.88
s1488	590	1486	110	3	3	3	3.10	2.54	3.40
s1494	469	1506	134	5	5	5	2.51	2.58	3.79
s5378	408	4603	1196	*11	*19	*19	61	347	543
$\sum$				734	799	865	1867	1433	1576

Table IV. Results on symbolic test evaluation.

		Random Sequences			Deterministic Sequences		
Circ.	PO	$\|T\|$	BDD Size	Time	$\|T\|$	BBD Size	Time
s208.1	1	200	250	0.02	111	111	0.02
s510	7	200	439	0.05	200	339	0.07
s953	23	200	179	0.23	16	198	0.05
s5378	49	200	*69	0.36	408	*21	0.90

the results. $rMOT$ and MOT are also successful in this case: Again they classify more faults than SOT. In all but two examples we succeeded in computing the exact MOT fault coverage of the test sequences. (Only in two of these cases (s208.1 and s953) general MOT is superior to $rMOT$ and thus was really necessary.) With respect to the execution times a superiority of SOT over $rMOT$ cannot be observed. Sometimes $rMOT$ works more efficiently than SOT. The speed-up can be explained by an earlier detection of faults. In order to investigate the space needed for the test evaluation we measured the size of the symbolic output sequences with respect to the random and deterministic test sequences. We considered only the circuits for which the MOT strategy detects faults which cannot be detected neither by the SOT nor by the $rMOT$ strategy. A symbolic output sequence consists of the $OBDD$ assigned to each output during a symbolic true value simulation. In Table IV the sizes of these output sequences are given. PO denotes the number of primary outputs of each circuit. In most cases the symbolic representations have a moderate size. The asterisk indicates that in case of circuit s5378 only a partial symbolic output sequence is computed due to the space requirements. The first 7 test vectors are simulated using the three-valued logic. Then the symbolic simulation starts. (This change is also done during fault simulation for this circuit.) In order to estimate the time needed for the test evaluation we computed a possible test response of the fault-free circuit. This can be done by initializing the memory elements at the beginning of the simulation. (Note that a test response of a fault-free circuit requires the computation of the product of all symbolic output values.) The experiments show that at least in this case the test evaluation can efficiently be performed.

VI. Conclusions

In this paper we described procedures for efficient fault simulation for synchronous sequential circuits based on the multiple observation time test strategy. On the one hand we presented an algorithm for identifying faults which cannot be detected by a conventional three-valued fault simulation due to the inaccuracy of the three-valued logic and the single observation time test strategy. The run time of this procedure is negligible. The elimination of the so-called X-redundant faults based on this procedure leads to a considerable speed-up of fault simulation. On the other hand we described how fault simulation based on the multiple observation time test strategy can efficiently be handled by using Ordered Binary Decision Diagrams. We also investigated a restricted multiple observation time test strategy. The experiments have shown that these strategies can help to improve the accuracy and the efficiency of fault simulation. In addition we have shown that fault simulation based on these strategies is feasible. The test evaluation can efficiently be done by comparing the test response of the circuit-under-test with a symbolic output sequence. As a rule it can be said that fault simulation based on the restricted multiple observation time test strategy is often sufficient to determine the exact fault coverage with respect to the multiple observation time test strategy.

References

[1] M. Abramovici and M.A. Breuer. On redundancy and fault detection in sequential circuits. *IEEE Trans. on Comp.*, 28(11):864–865, 1979.

[2] B. Becker and R. Krieger. FAST-SC: Fast fault simulation in synchronous sequential circuits. In *VLSI Design Conf.*, pages 128–131, 1993.

[3] F. Brglez, D. Bryan, and K. Kozminski. Combinational profiles of sequential benchmark circuits. In *Int'l Symp. Circ. and Systems*, pages 1929–1934, 1989.

[4] R.E. Bryant. Graph - based algorithms for Boolean function manipulation. *IEEE Trans. on Comp.*, 8:677–691, 1986.

[5] H. Cho, S. Jeong, F. Somenzi, and C. Pixley. Synchronizing sequences and symbolic traversal techniques in test generation. *Jour. of Electronic Testing, Theory and Applications*, 4:19–31, 1993.

[6] L.H. Goldstein. Controllability/observability analysis of digital circuits. *IEEE Trans. on Circ. and Systems*, 26(9):685–693, 1979.

[7] Z. Kohavi. *Switching and Finite Automata Theory.* McGraw-Hill Book Company, 1978.

[8] R. Krieger, B. Becker, and M. Keim. A hybrid fault simulator for synchronous sequential circuits. In *Int'l Test Conf.*, 1994.

[9] T. Kropf and H.-J. Wunderlich. A common approach to test generation and hardware verification based on temporal logic. In *Int'l Test Conf.*, pages 57–66, 1991.

[10] H.K. Lee and D.S. Ha. HOPE: An efficient parallel fault simulator for synchronous sequential circuits. In *Design Automation Conf.*, pages 336–340, 1992.

[11] A. Miczo. The sequential ATPG: A theoretical limit. In *Int'l Test Conf.*, pages 143–147, 1983.

[12] I. Pomeranz and S.M. Reddy. Test generation for synchronous sequential circuits using multiple observation times. In *Int'l Symp. on Fault-Tolerant Comp.*, pages 52–59, 1991.

[13] I. Pomeranz and S.M. Reddy. Fault simulation for synchronous sequential circuits under the multiple observation time testing approach. In *European Test Conf.*, pages 292–300, 1993.

[14] I. Pomeranz, S.M. Reddy, and L.N. Reddy. Increasing fault coverage for synchronous sequential circuits by the multiple observation time test strategy. In *Int'l Conf. on CAD*, pages 454–457, 1991.

[15] R. Wolber, U. Gläser, and H.T. Vierhaus. Testability analysis for test generation in synchronous sequential circuits. In *Int'l Conf. on Comp. Design*, pages 350–353, 1994.