Usable Circuits with Imperfect Scan Logic

Irith Pomeranz
School of Electrical and Computer Engineering
Purdue University
West Lafayette, IN 47907, U.S.A.
E-mail: pomeranz@ecn.purdue.edu

***Abstract*—The approximate computing paradigm supports yield improvement by allowing imperfect components to be used for computations that can tolerate imprecision. Applying the same arguments to the scan logic, a circuit with faults in the scan logic does not need to be discarded if it is possible to ascertain that the circuit will work correctly during functional operation. This article takes several steps toward a solution that allows circuits with faults in the scan logic to be used. It provides conditions under which a circuit with faults in the scan logic is usable. It also suggests how to extend a test set for fault detection targeting the functional logic to provide the fault detection capabilities needed to ascertain that a circuit with faults in the scan logic is usable. The transparent-scan approach is used for this purpose. Experimental results for benchmark circuits demonstrate some of the effects of this approach.**

I. Introduction

The approximate computing paradigm allows imperfect components to be used for computations that can tolerate imprecision [1]-[10]. It thus supports yield improvement by reducing the number of dice that need to be discarded because of faults.

Applying the same arguments to the scan logic, a circuit with faults in the scan logic does not need to be discarded if it is possible to ascertain that the circuit will work correctly during functional operation. With approximately 50% of the area of a design being occupied by scan logic [11]-[12], the effect on the yield can be significant.

This article suggests that a circuit with a fault f_s in the scan logic does not need to be discarded if two conditions are satisfied: (1) it is possible to place the circuit in functional mode when f_s is present in the scan logic, and (2) tests can be found for all the detectable faults in the functional logic in the presence of f_s. The set of detectable faults in the functional logic is denoted by F_L. The required tests detect faults in F_L when f_s is present in the circuit.

A unit that satisfies these conditions is said to be usable even though it has an imperfect scan logic. Such a unit can be used in one of several ways as discussed next.

A unit that passes all the tests for F_L can be considered fault-free since faults in the functional logic have been tested for, even though its scan logic is imperfect. Compared with approximate computing, the advantage of allowing imperfect scan chains is that it is still possible to ensure that the functional logic passes all the tests.

The work was supported in part by NSF Grant No. CCF-2041649.

In addition, scan is a DFT approach aimed at making the design easier to test by improving its controllability and observability. An imperfect scan chain that still makes the design more controllable and observable makes it more testable, and thus achieves the goal of inserting DFT logic.

Even if the unit is eventually discarded, it can be used for further diagnosis of defects in the functional logic. This is important in the early stages of yield learning when the yield is low, and defect diagnosis is needed for providing information about the types of defects that occur using as many faulty units as possible.

This article takes the following steps toward a solution that allows units with faults in the scan logic to be used.

The first step is to identify a set of faults in the scan logic that allow the circuit to be placed in functional mode. This set of faults is denoted by F_S. Of all the faults in the scan logic, only faults from F_S are considered further in this article. Experimental results for benchmark circuits demonstrate that F_S contains significant numbers of faults.

The second step is related to the test set used for the circuit. A conventional scan-based test set C for the detectable faults in F_L is generated under the assumption that the scan logic is fault free. Such a test set is not always sufficient for detecting the faults in F_L when a fault from F_S is present in the circuit. Starting from C, the article suggests a procedure for generating tests for F_L that address the presence of a fault from F_S.

Tests are simulated and generated in this article under the approach referred to as transparent-scan [13]. This ensures that faults in the scan logic are accounted for correctly. Under transparent-scan, a conventional scan-based test set is translated into a sequence of scan shift and functional capture cycles. An input vector in the transparent-scan sequence specifies values for the primary inputs, the scan enable and scan chain inputs. The corresponding output vector specifies values for the primary outputs and scan chain outputs. When the transparent-scan sequence is simulated cycle by cycle, the effects of faults in the scan logic are included in the simulation.

The test generation procedure suggested in this article uses a transparent-scan sequence T obtained from a conventional scan-based test set C that targets the detection of the faults in F_L. The procedure computes several different scan-enable sequences for T to detect faults from F_L in the presence of faults from F_S. In general, a fault $f_s \in F_S$ is activated by a scan shift cycle. Activation of f_s may prevent a fault f_l in

the functional logic from being detected. Without the ability to detect f_l, the circuit cannot be ascertained to be usable. By replacing scan shift cycles with functional capture cycles, the procedure described in this article produces a new scan-enable sequence with fewer scan shift cycles. This sequence allows f_l to be detected in the presence of f_s. Experimental results for benchmark circuits show that, for significant numbers of faults from F_S, scan-enable sequences exist for detecting all the detectable faults in the functional logic.

It is interesting to note that the need for tests other than conventional scan-based tests also exists when considering the problem of scan chain diagnosis [14]-[21]. Here, tests with scan-enable sequences different from conventional scan-based tests provide additional fault diagnosis capabilities. Scan chain diagnosis is needed for identifying a fault in the scan logic. The identification must be precise to determine with confidence that the fault belongs to F_S. This it not likely to be possible for defects with complex behaviors. However, if the fault belongs to F_S, the circuit is usable if the second condition can be ascertained as well.

The entire scan-based test set C is translated into a single transparent-scan sequence T for the discussion in this article. Since T requires sequential fault simulation, this limits the discussion to logic blocks for which sequential fault simulation of T is feasible. Partitioning C will result in shorter transparent-scan sequences, and allow larger logic blocks to be considered. In addition it is assumed that it is possible to store scan-enable sequences on the tester, and use them for test application. Excluding the scan-enable sequence, the procedure described in this article does not modify the test set C. If C is compressed, the same on-chip decompression logic can be used for applying it.

The article is organized as follows. Section II provides the conditions under which a circuit with a fault in the scan logic is usable. Section III describes the type of tests considered in this article. Section IV describes a procedure for generating scan-enable sequences. Section V presents experimental results.

II. Usable Circuits with Faults in the Scan Logic

Two conditions need to be satisfied for a fault f_s in the scan logic that leaves the circuit usable.

(1) In the presence of f_s, it is possible to place the circuit in functional mode by setting the scan-enable input to zero. The set of all the faults in the scan logic that satisfy this condition is denoted by F_S.

(2) In the presence of f_s, it is possible to detect all the detectable faults in the functional logic, i.e., the set of faults F_L. If the circuit in the presence of f_s passes a test set that detects all the faults in F_L, the functional logic can be considered fault-free.

The two conditions together ensure that the circuit can work correctly during functional operation, which is the condition for a usable circuit. For simplicity of discussion, the sets of faults F_S and F_L consist of single stuck-at faults.

To define the set F_S, the structure of a multiplexer-based scan flip-flop is shown in Figure 1. The flip-flop has index i. Its present-state variable is y_i, and its next-state variable is Y_i. The functional logic driving Y_i is not shown in Figure 1. The input s_{en} is the scan-enable input. We have that $s_{en} = 0$ during functional operation, and $s_{en} = 1$ during scan shifting.

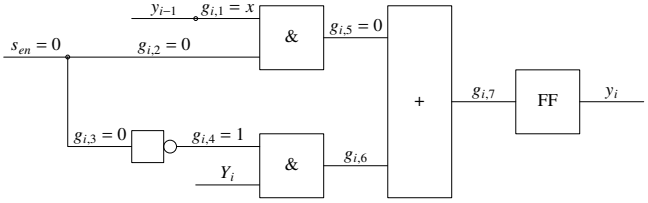

Fig. 1. Faults in the scan logic.

TABLE I
Example Output Responses

circuit	$-$	f_s	$f_s, f_{l,0}$	$f_s, f_{l,1}$	$f_s, f_{l,2}$
response	0000	1100	0011	0000	1100

Since functional operation occurs when $s_{en} = 0$, the fault s_{en} stuck-at 0, denoted by $s_{en}/0$, may be present without preventing the circuit from being placed in functional mode.

When $s_{en} = 0$, additional values are implied as shown in Figure 1. The stuck-at faults that result in the values shown in Figure 1 can be present in the circuit without affecting the ability to place it in functional mode. In addition, with $s_{en} = 0$, the value of $g_{i,1}$ does not affect the ability to place the circuit in functional mode. Therefore, the faults $g_{i,1}/0$ and $g_{i,1}/1$ do not affect this ability.

Overall, the set of faults F_S includes the fault $s_{en}/0$, and for every flip-flop i, the faults $g_{i,1}/0$, $g_{i,1}/1$, $g_{i,2}/0$, $g_{i,3}/0$, $g_{i,4}/1$ and $g_{i,5}/0$.

A fault $f_s \in F_S$ satisfies the first condition given above for a usable circuit. For the second condition, a test set C needs to detect all the detectable faults of the functional logic, included in F_L, when f_s is present in the circuit. This implies that the fault-free circuit is replaced with the circuit that contains f_s for the purpose of fault detection. More accurately, let the response of the fault-free circuit to C be $Z(-)$. Let the response of the faulty circuit to C in the presence of a fault f be $Z(f)$. When the circuit is expected to be fault-free, the condition for detecting a fault $f_l \in F_L$ is $Z(f_l) \neq Z(-)$. If f_s leaves the circuit usable, it is acceptable for f_s to be present in the circuit, and the response of the fault-free circuit is replaced with $Z(f_s)$. The condition that needs to be satisfied for every fault $f_l \in F_L$ is $Z(f_s, f_l) \neq Z(f_s)$, where $Z(f_s, f_l)$ is the response of the faulty circuit when f_l is present in addition to f_s. If this condition is satisfied for a fault $f_s \in F_S$, the circuit in the presence of f_s is considered to be usable.

To illustrate this condition, Table I shows hypothetical output responses of a circuit (an example based on an actual circuit is given in the next section). The fault-free response $Z(-) = 0000$ is shown in the second column of Table I, followed by the response obtained in the presence of f_s, and three responses obtained when a fault in the functional logic is present in addition to f_s.

Once it is established that f_s is present in the circuit, the fault-free response $Z(-) = 0000$ is replaced with $Z(f_s) = 1100$. The fault $f_{l,0}$ is detected in the presence of f_s since $Z(f_s, f_{l,0}) \neq Z(f_s)$.

The same applies to $f_{l,1}$. Even though $Z(f_s, f_{l,1}) = Z(-)$, the fault-free circuit is not relevant to the discussion once it is determined that f_s is present in the circuit. The fact that $Z(f_s, f_{l,1}) \neq Z(f_s)$ implies that $f_{l,1}$ is detected in the presence of f_s.

The fault $(f_s, f_{l,2})$ is detected relative to the fault-free circuit. However, with $Z(f_s, f_{l,2}) = Z(f_s)$, the fault $f_{l,2}$ is not considered detected when the presence of f_s is accepted.

It is possible to add conventional scan-based tests to a test set C to detect faults from F_L in the presence of a fault $f_s \in F_S$. However, conventional scan-based tests have a limited ability to detect faults in the functional logic in the presence of faults in the scan logic. The approach in this article relies on several different scan-enable sequences to detect the faults in F_L when f_s is present in the circuit. The scan-enable sequences are associated with a transparent-scan sequence T obtained from a conventional scan-based test set C that detects the faults in F_L. This is discussed in the next sections.

III. Scan-Enable Sequences

This section starts with an example to illustrate the use of transparent-scan and the importance of the scan-enable sequence. The circuit under consideration is benchmark circuit $s27$. The circuit has a set A of four primary inputs, one primary output, z_0, and three state variables included in a single scan chain. In addition, the scan circuit has a scan-enable input s_{en}, a scan chain input s_{in}, and a scan chain output s_{out}. The primary output z_0 and the scan chain output s_{out} are included together in a set of outputs denoted by Z. The output response of the circuit is obtained from Z.

An input (output) vector of a transparent-scan sequence specifies values for all the inputs (outputs). A transparent-scan sequence for $s27$ is shown in the left part of Table II. The sequence is obtained from a conventional scan-based test set that consists of six tests. Each test starts with a scan-in operation of three clock cycles, and ends with a scan-out operation of three clock cycles that is overlapped with the scan-in operation of the next test. This results in the scan-enable sequence 11101110...0111. This scan-enable sequence is shown under column $S_{en,0}$ of Table II.

Table II shows the sequence of primary input vectors under column A, and the scan chain input sequence under column S_{in}. The sequence of output vectors is shown under column $Z_0(-)$ for the fault-free circuit, and under column $Z_0(f)$ for a fault f. The right part of Table II will be discussed later.

A fault f is detected, in the conventional sense, when $Z_0(f) \neq Z_0(-)$ at a clock cycle u. The output vectors at clock cycle u are denoted by $Z_0(-)(u)$ for the fault-free circuit, and $Z_0(f)(u)$ in the presence of a fault f. The condition for fault detection at clock cycle u is $Z_0(f)(u) \neq Z_0(-)(u)$. Under transparent-scan, fault detection can occur at any clock cycle, including both functional capture and scan shift cycles.

TABLE II
Example Transparent-Scan Sequence

u	A	$S_{en,0}$	S_{in}	$Z_0(-)$	$Z_0(f_{46})$	$S_{en,1}$	$Z_1(f_{46})$	$Z_1(f_{46}, f_5)$
0	0000	1	1	xx	xx	1	xx	xx
1	0000	1	1	1x	1x	1	1x	1x
2	0000	1	0	1x	1x	1	1x	1x
3	0000	0	0	01	01	0	01	01
4	1001	1	0	11	11	1	11	11
5	1001	1	1	11	11	1	11	11
6	1001	1	0	10	11	1	11	11
7	1001	0	0	00	11	0	11	11
8	0100	1	0	00	11	0	11	11
9	0100	1	1	11	01	1	11	11
10	0100	1	1	10	11	1	11	11
11	0100	0	0	10	11	0	11	11
12	0111	1	1	11	11	0	11	11
13	0111	1	0	10	11	1	10	10
14	0111	1	0	00	01	1	11	01
15	0111	0	0	11	11	0	11	-
16	1101	1	1	10	10	1	10	-
17	1101	1	1	10	11	1	10	-
18	1101	1	0	10	11	1	10	-
19	1101	0	0	11	11	0	11	-
20	1010	1	0	11	11	1	11	-
21	1010	1	0	10	11	1	10	-
22	1010	1	0	11	11	1	11	-
23	1010	0	0	10	11	0	10	-
24	xxxx	1	x	10	10	1	10	-
25	xxxx	1	x	x0	x1	1	x0	-
26	xxxx	1	x	x1	x1	1	x1	-

The fault $f_{46} \in F_S$ of $s27$ is considered in the left part of Table II. With $Z_0(f_{46})(6) \neq Z_0(-)(6)$, the fault is detected at clock cycle $u = 6$. The faults in F_L are simulated in the presence of f_{46}. Fault simulation shows that nine faults remain undetected, including $f_5 \in F_L$.

To detect $f_5 \in F_L$ in the presence of $f_{46} \in F_S$, the procedure described in Section IV produces the scan-enable sequence shown under column $S_{en,1}$ of Table II. The procedure from Section IV decided to change scan shift cycles into functional capture cycles at clock cycles $u = 8$ and 12. The primary input and scan chain input sequences are not modified.

The output sequences obtained with $S_{en,1}$ are shown under column $Z_0(f_{46})$ for $f_{46} \in F_S$, and $Z_0(f_{46}, f_5)$ for the case where $f_5 \in F_L$ is present in addition to $f_{46} \in F_S$. In the presence of f_{46}, the fault f_5 is detected at clock cycle $u = 14$, i.e., $Z_0(f_{46}, f_5)(14) \neq Z_0(f_{46})(14)$. The sequence is not simulated beyond this clock cycle.

IV. Generation of Scan-Enable Sequences

This section describes a procedure for generating scan-enable sequences for a transparent-scan sequence T based on the faults in F_S.

A. Preliminaries

The transparent-scan sequence T is obtained from a conventional scan-based test set C that targets the detection of the faults in F_L when the scan logic is fault-free. The transparent-scan sequence is not modified except to produce additional scan-enable sequences. The goal is to ensure that, with as many faults in F_S as possible, each one considered alone, all the detectable faults in F_L can be detected.

The procedure generates a set of scan-enable sequences, denoted by S_{EN}, based on all the faults in F_S. If scan chain

diagnosis determines that a particular fault $f_s \in F_S$ is present, it is possible to select a subset of the sequences in S_{EN} for detecting the faults from F_L. If a scan-enable sequence $S_{en,i} \in S_{EN}$ is selected, the transparent-scan sequence with the scan-enable sequence $S_{en,i}$ needs to be applied. The resulting transparent-scan sequence is denoted by T_i.

Initially, S_{EN} contains only the original scan-enable sequence of T, denoted by $S_{en,0}$. For a circuit with K flip-flops in its longest scan chain, $S_{en,0} = 1...101...10...01...1$, with subsequences of K scan shift cycles (1s) for scan operations, and a single functional capture cycle (0) between every two scan operations.

The procedure for computing scan-enable sequences uses a fault simulation process that accepts a fault $f_s \in F_S$ and a set of scan-enable sequences S_{EN}. For every scan-enable sequence $S_{en,i} \in S_{EN}$, every fault $f_l \in F_L$ is simulated under T_i in the presence of f_s. The output sequence $Z_i(f_s, f_l)$ is compared to the output sequence $Z_i(f_s)$ obtained in the presence of f_s alone to determine whether f_l is detected by T_i when it is present together with f_s. If f_l is detected, it is marked as such and not simulated further. The set of faults from F_L that are left undetected in the presence of f_s is denoted by $U_L(f_s)$.

If the procedure obtains $U_L(f_s) = \emptyset$, all the faults in F_L are detected in the presence of f_s, and the circuit in the presence of f_s satisfies the conditions of a usable circuit. In this case, f_s is moved from F_S to a set denoted by F_{USE}.

B. Procedure Overview

Initially for the generation of scan-enable sequences, $F_{USE} = \emptyset$ and $S_{EN} = \{S_{en,0}\}$. In addition, $U_L(f_s) = F_L$ for every fault $f_s \in F_S$ indicates that the fault has not been simulated yet (the entire set F_L is not stored in this case, only an indication that $U_L(f_s) = F_L$).

The procedure performs iterations where it attempts to generate new scan-enable sequences for more and more faults from F_S. This is motivated by the following observation. Even with the initial set $S_{EN} = \{S_{en,0}\}$, there are faults for which $U_L(f_s)$ is small, and new scan-enable sequences are required for a small number of faults. After these faults are considered, and new scan-enable sequences are added to S_{EN}, the sets $U_L(f_s)$ decrease for other faults, and these faults are targeted for the generation of new scan-enable sequences. Overall, the procedure considers faults with small sets of undetected faults, and postpones other faults to later iterations.

Two constants are used for deciding how a fault $f_s \in F_S$ will be handled, N_{STORE} and N_{GEN}. In an arbitrary iteration, when the procedure considers a fault $f_s \in F_S$, it first performs fault simulation for f_s with the current set S_{EN} to update the set $U_L(f_s)$. If $U_L(f_s) = \emptyset$, f_s is moved to F_{USE} and not considered further. If $|U_L(f_s)| \leq N_{GEN}$, the procedure attempts to generate new scan-enable sequences for f_s. Otherwise, the procedure postpones the consideration of f_s to a later iteration.

In addition, if $|U_L(f_s)| \leq N_{STORE}$, the procedure stores $U_L(f_s)$ for later use. This will avoid the need to simulate the entire set F_L when f_s is considered later. Otherwise, the indication that $U_L(f_s) = F_L$ is kept.

With $|U_L(f_s)| \leq N_{GEN}$, the procedure attempts to generate a new scan-enable sequence for every fault $f_l \in U_L(f_s)$. If a new scan-enable sequence $S_{en,i}$ is generated for f_l, the procedure simulates all the faults in $U_L(f_s)$ under T_i, and removes detected faults from $U_L(f_s)$. It then adds $S_{en,i}$ to S_{EN}. If the procedure is unable to generate a new scan-enable sequence for a fault $f_l \in U_L(f_s)$, it does not consider additional faults from $U_L(f_s)$ in the same iteration.

To take full advantage of newly generated scan-enable sequences, the procedure considers the faults of F_S from low to high size of $U_L(f_s)$ (the size is considered before it is updated). This affects the order of the faults starting in the second iteration. Among faults with the same size of $U_L(f_s)$, the procedure prefers the one closest to the scan chain output. This fault affects a smaller portion of the scan chain, and is more likely to leave the circuit usable. A higher index indicates that a fault is closer to the scan chain output.

The procedure terminates after consideration of all the faults in F_S does not result in the generation of any additional scan-enable sequence.

C. Generating a Scan-Enable Sequence

The generation of a new scan-enable sequence for a fault $f_l \in U_L(f_s)$ is described next. The new scan-enable sequence is denoted by $S_{en,i}$. The length of $S_{en,i}$ is equal to the length of T, and it is denoted by L. For $0 \leq u < L$, the value at clock cycle u of $S_{en,i}$ is denoted by $S_{en,i}(u)$.

For a constant N_{INIT}, the sequence $S_{en,i}$ is initialized up to N_{INIT} times using N_{INIT} different sequences from S_{EN}. The procedure always uses $S_{en,0}$ to initialize $S_{en,i}$. In addition, it uses up to $N_{INIT} - 1$ sequences that detect f_l when it is present alone in the circuit. After initialization, the procedure attempts to modify $S_{en,i}$ to detect f_l in the presence of f_s.

The modification is based on three premises, discussed next.

1) The Set $ACT(f_s)$: The first premise is that the activation of f_s, which occurs during scan shift cycles, prevents f_l from being detected in its presence. If f_s is activated at fewer clock cycles, f_l may be detected.

Based on this premise, the procedure simulates the fault-free circuit and the faulty circuit in the presence of f_s under T_i to search for scan shift cycles where f_s is activated. The activation of f_s occurs at a clock cycle u if the fault site of f_s assumes different values in the fault-free and faulty circuits at clock cycle u. If this condition is satisfied, clock cycle u is included in a set denoted by $ACT(f_s)$. The set $ACT(f_s)$ is updated as $S_{en,i}$ is modified by considering the activation of f_s under T_i.

2) The Set $DET(f_l)$: The second premise is that a clock cycle $u \in ACT(f_s)$ is important to the detection of f_l if it occurs close to a clock cycle where f_l is detected by T_i. Since f_l is not detected in the presence of f_s, detection of f_l is considered when it is present alone in the circuit.

Based on this premise, the procedure simulates the fault-free circuit and the faulty circuit in the presence of f_l under T_i to search for clock cycles where f_l is detected. The detection of f_l occurs on the outputs at clock cycle u when $Z_i(f_l)(u) \neq Z_i(-)(u)$. If this condition is satisfied, clock cycle u is included in a set denoted by $DET(f_l)$.

Considering a clock cycle $u \in DET(f_l)$, the presence of f_s is likely to prevent f_l from being detected if f_s is activated within the clock cycles of a conventional scan-based test preceding clock cycle u. This is because T_i is constructed from conventional scan-based tests for F_L. For a circuit with K state variables in its longest scan chain, a conventional scan-based test has $2K+1$ clock cycles. The $2K+1$ clock cycles preceding clock cycle u are added to $DET(f_l)$.

3) The Set MOD: A scan shift cycle is considered important to the detection of f_l if $u \in ACT(f_s)$ implies that f_s is activated at clock cycle u, and $u \in DET(f_l)$ implies that f_l is detected within $2K+1$ clock cycles from u. Together, the procedure considers the clock cycles in $MOD = ACT(f_s) \cap DET(f_l)$ important to change into functional capture cycles. As the set MOD evolves, a clock cycle that was already considered earlier is excluded.

4) Fault Detection: The third premise is that any change to $S_{en,i}$ should ensure that f_l is detected by T_i when it is present alone in the circuit. Although the fault is not detected in the presence of f_s, its detection alone ensures that T_i maintains its ability to detect it. With fewer activations of f_s, it is possible that f_l will be detected in its presence.

5) Procedure: With these premises, the procedure considers the clock cycles from MOD one by one in a random order. When clock cycle u is considered, the procedure assigns $S_{en,i}(u) = 0$. It then performs fault simulation of f_l under T_i in the presence of f_s. If f_l is detected, the procedure has $S_{en,i}$ as the required scan-enable sequence. In this case, it simulates all the faults in $U_L(f_s)$ under T_i and removes detected faults from $U_L(f_s)$. It then adds $S_{en,i}$ to S_{EN}.

If T_i does not detect f_l in the presence of f_s, the procedure simulates f_l alone to check whether it is detected by T_i. If it is, the procedure keeps $S_{en,i}(u) = 0$, and updates the set MOD. Otherwise, it restores $S_{en,i}(u) = 1$.

The use of $MOD = ACT(f_s) \cap DET(f_l)$ results in relatively small numbers of clock cycles that the procedure considers for the generation of a new scan-enable sequence.

The procedure is not guaranteed to find a scan-enable sequence for f_l even if one exists. It compensates for this issue by performing several iterations where it considers every fault $f_s \in F_S$ with $0 < |U_L(f_s)| \leq N_{GEN}$.

V. Experimental Results

The procedure for generating scan-enable sequences was applied to benchmark circuits as described next.

The target faults are single stuck-at faults. The transparent-scan sequence T was obtained from a compact scan-based test set C for single stuck-at faults. The length of T is equal to the number of clock cycles required for applying it, which is also the number of clock cycles required for applying C.

The procedure was run with a limit of $N_{STORE} = 256$ on the number of undetected faults in a stored set $U_L(f_s)$. To generate new scan-enable sequences for T, the size of $U_L(f_s)$ was initially required to be at most $N_{GEN} = 16$. This value was selected based on the experimental observation that benchmark circuits have large numbers of faults for which $U_L(f_s) \leq 16$ when T_0 is simulated. After the procedure terminates with this value, it is also run with $N_{GEN} = 64$ and 256. The increased values of N_{GEN} are needed for several circuits.

The number of scan-enable sequences used for initialization of a new scan-enable sequence is $N_{INIT} = 8$. Experimental results show that a higher value of N_{INIT} is typically not needed.

The procedure was run with a runtime limit. Additional runtime may increase the number of faults that leave the circuit usable.

The results are shown in Table III. The circuits are arranged from low to high percentage of faults in F_{USE} relative to the number of faults in F_L. This percentage is computed as $|F_{USE}|/|F_L| \cdot 100$. It provides an indication of the importance of identifying faults in the scan logic that leave a circuit usable.

Table III is organized as follows. After the circuit name, column sv shows the number of state variables, and column pi shows the number of primary inputs.

Column C shows the number of tests in the conventional scan-based test set C. Column T shows the length of the transparent-scan sequence T, which is also the number of clock cycles required for applying it.

Column F_S shows the number of faults in F_S, which is the number of faults in the scan logic that allow the circuit to be placed in functional mode. Column $\%F_S$ shows the percentage of faults in F_S relative to the number of faults in F_L, computed as $|F_S|/|F_L| \cdot 100$.

Column N_{GEN} shows the value of the parameter N_{GEN}. Column $iter$ shows the iteration of the test generation procedure with N_{GEN}. Column S_{EN} shows the number of scan-enable sequences in S_{EN}. Column F_{USE} shows the number of faults from F_S that leave the circuit usable since they allow all the faults in F_L to be detected. Column $\%F_{USE}$ shows the percentage of faults in F_{USE} relative to F_L, $|F_{USE}|/|F_L| \cdot 100$, when S_{EN} is used.

Column tg shows the total number of attempts made to generate a scan-enable sequence for a fault $f_l \in F_L$ in the presence of a fault $f_s \in F_S$. Column $ntime$ shows the normalized runtime, which is the cumulative runtime divided by the runtime for fault simulation with fault dropping of $F_S \cup F_L$ under T_0.

The following points can be seen from Table III. A conventional scan-based test set, using only $S_{en,0}$, is typically not sufficient for determining that a circuit is usable. With the scan-enable sequences included in S_{EN}, the number of faults that leave the circuit usable is significant for many of the circuits considered.

The number of faults in F_{USE} as a percentage of F_L varies with the circuit. For many of the circuits it is sufficiently high

TABLE III
EXPERIMENTAL RESULTS

circuit	sv	pi	C	T	F_S	$\%F_S$	N_{GEN}	iter	S_{EN}	F_{USE}	$\%F_{USE}$	tg	ntime
systemcaes	670	258	121	81861	4021	18.527	16	1	1	4	0.018	0	67.80
s9234	228	19	111	25647	1369	19.763	16	1	5	12	0.173	5	1832.51
wb_dma	523	215	66	35107	3139	34.506	16	1	13	20	0.220	12	601.51
s13207	669	31	235	158119	4015	40.907	16	1	16	22	0.224	15	117.05
s15850	597	14	97	58603	3583	30.559	16	1	58	88	0.751	57	382.97
b11	30	8	59	1859	181	16.621	64	5	123	14	1.286	807	43188.97
des_area	128	239	118	15350	769	7.742	16	1	23	180	1.812	22	291.12
s5378	179	35	100	18179	1075	23.354	16	1	124	155	3.367	222	31679.40
b08	21	10	38	857	127	25.971	64	6	70	17	3.476	640	25837.35
b05	34	2	61	2169	205	11.289	64	12	206	72	3.965	2461	148174.03
b07	51	2	52	2755	307	25.951	64	5	191	55	4.649	1539	126008.30
systemcdes	190	130	79	15279	1141	17.283	16	1	19	465	7.043	18	706.85
s526	21	3	50	1121	127	22.883	64	6	157	47	8.468	1296	70219.70
b04	66	12	44	3014	397	29.495	256	2	248	126	9.361	5415	280647.41
s382	21	3	25	571	127	31.830	16	5	85	47	11.779	300	10754.76
b09	28	2	21	637	169	40.238	16	4	58	55	13.095	202	8794.03
i2c	128	17	45	5933	769	32.905	16	4	334	313	13.393	1295	143270.89
s953	29	16	76	2309	175	16.219	64	2	67	148	13.716	91	2406.90
s1423	74	17	26	2024	445	29.373	16	5	246	281	18.548	1074	61358.84
simple_spi	131	15	36	4883	787	37.458	16	6	237	483	22.989	1430	176787.61
b03	30	5	24	774	181	40.044	16	4	58	114	25.221	283	13417.58
usb_phy	98	14	32	3266	589	45.308	256	2	239	423	32.538	2815	204262.28
sasc	117	15	22	2713	703	41.696	16	3	107	575	34.104	449	34281.91

to make a difference in the number of usable circuits.

The normalized runtime does not increase with the size of the circuit. This implies that the procedure scales similar to a fault simulation procedure.

VI. CONCLUDING REMARKS

This article took several steps to suggest that, analogous to approximate computing, a circuit with faults in the scan logic does not need to be discarded if it is possible to ascertain that the circuit will work correctly during functional operation. The article provided conditions under which a circuit with faults in the scan logic is usable. It also suggested how to identify faults that satisfy these conditions. As part of this process, faults in the scan logic need to be diagnosed, and faults in the functional logic need to be detected in the presence of faults in the scan logic. The article described a procedure that generates several different scan-enable sequences for the same transparent-scan sequence obtained from a conventional scan-based fault detection test set for faults in the functional logic. Experimental results were presented for benchmark circuits to demonstrate some of the effects of this approach.

REFERENCES

[1] C. Li, W. Luo, S. S. Sapatnekar and J. Hu, "Joint Precision Optimization and High Level Synthesis for Approximate Computing", in Proc. Design Automation Conf., 2015, Art. 104, pp. 1–6.

[2] Q. Xu, T. Mytkowicz and N. S. Kim, "Approximate Computing: A Survey", in IEEE Design & Test, Vol. 33, No. 1, 2016.

[3] S. Mittal, "A Survey of Techniques for Approximate Computing", in ACM Computing Surveys, April 2016, Vol. 48, No. 4, Art. 62, pp. 1-33.

[4] M. Shafique, R. Hafiz, S. Rehman, W. El-Harouni and J. Henkel, "Cross-Layer Approximate Computing: from Logic to Architectures", in Proc. Design Automation Conf., 2016, Art. 99, pp. 1-6.

[5] S. Dutt, S. Nandi and G. Trivedi. "Analysis and Design of Adders for Approximate Computing", in ACM Trans. on Embedded Computing Systems, Feb. 2017, Vol. 17, No. 2, Art. 40, pp. 1-28.

[6] M. Gao and G. Qu, "Energy Efficient Runtime Approximate Computing on Data Flow Graphs", in Proc. Intl. Conf. on Computer-Aided Design, 2017, pp. 444–449.

[7] B. Nongpoh, R. Ray, M. Das and A. Banerjee, "Enhancing Speculative Execution With Selective Approximate Computing", in ACM Trans. on Design Automation, Feb. 2019, Vol. 24, No. 2, Art. 26, pp. 1-29.

[8] A. Wendler and O. Keszocze, "A fast BDD Minimization Framework for Approximate Computing", in Proc. Design, Automation and Test in Europe Conf., 2020, pp. 1372–1377.

[9] J. Bonnot, D. Menard and K. Desnos, "Fast Kriging-Based Error Evaluation for Approximate Computing Systems", in Proc. Design, Automation and Test in Europe Conf., 2020, pp. 1384–1389.

[10] M. Traiola, A. Virazel, P. Girard, M. Barbareschi and A. Bosio, "A Survey of Testing Techniques for Approximate Integrated Circuits", in Proceedings of the IEEE, Vol. 108, No. 12, 2020.

[11] S. R. Makar and E. J. McCluskey, "Functional Tests for Scan Chain Latches", in Proc. Intl. Test Conf., 1995.

[12] F. Yang, S. Chakravarty, N. Devta-Prasanna, S. M. Reddy and I. Pomeranz, "On the Detectability of Scan Chain Internal Faults - An Industrial Case Study", in Proc. VLSI Test Symp., April 2008.

[13] I. Pomeranz and S. M. Reddy, "Transparent Scan: A New Approach to Test Generation and Test Compaction for Scan Circuits that Incorporates Limited Scan Operations", in IEEE Trans. on Computer-Aided Design, Dec. 2003, pp. 1663-1670.

[14] X. Tang, R. Guo, W.-T. Cheng, S. M. Reddy and Y. Huang, "On Improving Diagnostic Test Generation for Scan Chain Failures", in Proc. Asian Test Symp., 2009, pp. 41-46.

[15] Z. Chen, S. Seth, D. Xiang and B. B. Bhattacharya, "Diagnosis of Multiple Scan-Chain Faults in the Presence of System Logic Defects", in Proc. Asian Test Symp., 2011, pp. 297-302.

[16] W.-L. Tsai, W.-C. Liu and J. C.-M. Li, "Structural Reduction Techniques for Logic-Chain Bridging Fault Diagnosis", in IEEE Trans. on Computers, July 2012, Vol. 61, No. 7, pp. 928-938.

[17] Y. Huang, X. Fan, H. Tang, M. Sharma, W.-T. Cheng, B. Benware and S. M. Reddy, "Distributed Dynamic Partitioning based Diagnosis of Scan Chain", in Proc. VLSI Test Symposium, 2013, pp. 1-6.

[18] W.-H. Lo, A.-C. Hsieh, C.-M. Lan, M.-H. Lin and T. Hwang, "Utilizing Circuit Structure for Scan Chain Diagnosis", in IEEE Trans. on VLSI Systems, Dec. 2014, Vol. 22, No. 12, pp. 2766-2778.

[19] S. Kundu, S. Chattopadhyay, I. Sengupta and R. Kapur, "Scan Chain Masking for Diagnosis of Multiple Chain Failures in a Space Compaction Environment", in IEEE Trans. on VLSI Systems, July 2015, Vol. 23, No. 7, pp. 1185-1195.

[20] Y. Huang, B. Benware, R. Klingenberg, H. Tang, J. Dsouza, W.-T. Cheng, "Scan Chain Diagnosis Based on Unsupervised Machine Learning" in Proc. Asian Test Symp., 2017.

[21] I.-D. Huang, P. Gupta, L. Lingappan and V. Gangaram, "Online Scan Diagnosis : A Novel Approach to Volume Diagnosis", in Proc. Intl. Test Conf., 2018.