Deterministic Test for the Reproduction and Detection of Board-Level Functional Failures*

Hongxia Fang¹, Zhiyuan Wang², Xinli Gu² and Krishnendu Chakrabarty¹

¹ECE Dept., Duke University, Durham, NC, USA ²Cisco Systems Inc., San Jose, CA, USA
 {hf12, krish}@ee.duke.edu {zhiywang, xgu}@cisco.com

Abstract—A common scenario in industry today is “No Trouble Found” (NTF) due to functional failures. A component on a board fails during board-level functional test, but it passes the Automatic Test Equipment (ATE) test when it is returned to the supplier for warranty replacement or service repair. To find the root cause of NTF, we propose an innovative functional test approach and DFT methods for the detection of board-level functional failures. These DFT and test methods allow us to reproduce and detect functional failures in a controlled deterministic environment, which can provide ATE tests to the supplier for early screening of defective parts. Experiments on an industry design show that functional scan test with appropriate functional constraints can adequately mimic the functional state space well (measured by appropriate coverage metrics). Experiments also show that most functional failures due to stuck-at, dominant bridging, and crosstalk faults can be reproduced and detected by functional scan test.

I. INTRODUCTION

Functional test is commonly used in industry to target defects that are not detected by structural test [1] [2]. An advantage of functional test is that it can be used to evaluate functional mode power consumption, signal integrity, etc., since it is performed in normal functional mode. In contrast, structural test is accompanied by additional power consumption and di/dt problems. It may also lead to test escapes due to clock stretching [3]. Although structural test suffers from the above disadvantages, a major advantage of structural test is that it is applied in a controllable and deterministic manner, i.e., its clocks and signal values in each clock cycle is determined and repeatable in multiple runs with the same test. On the other hand, functional test is non-repeatable; the clock signal and signal values may show appreciable shifts in multiple runs for the same test. Therefore, it is hard to identify the exact time when a functional test fails in native (mission) mode and thus it is hard to diagnose functional failures.

A board may fail structural test or functional test, or both. Various diagnosis techniques [4] [5] [6] have been developed to find the root cause for chip-level failures. However, test and diagnosis are more complex at board-level when functional test is applied. Also, external memory, noise, power-supply variations, and clock skew on a board test aggravate the problem of board-level functional failure diagnosis [7].

Moreover, in many cases in practice, a board does not fail until functional test is performed. A common scenario in industry is “No Trouble Found” (NTF) due to functional failures. The problem here is that a component on a board fails board-level functional test, but it passes Automatic Test Equipment (ATE) test when it is returned to the component supplier for warranty replacement or service repair. Board

manufacturers are especially concerned about the complexity of such diagnosis problems. It is therefore no longer sufficient to simply use boundary scan testing to localize faulty ASICs on a board.

Little work has been published in the literature on techniques that can help to locate the root cause of a board-level functional failure to a faulty wire/gate inside the component. This is necessary because additional test patterns can be generated once we have located the root cause to a faulty wire/gate. These additional test patterns can be ported to automatic-test-equipment (ATE) at the supplier side for screening defective parts at an earlier stage, thus reducing defective-parts-per-million (DPPM) and product cost.

The trace buffer technique [8] [9] and freezing/dump technique [10] [11] [12] [13] used in silicon debug are useful for localizing a board-level functional failure. However, trace buffers can only provide real-time at-speed observation for a few signals across a limited number of clock cycles. In the freezing/dump technique reported by Intel [10], use is made of the fact that microprocessors and SOCs have the ability to stop the execution of normal functional mode and freeze internal clocks. However, this method is only effective for synchronous designs and it is not applicable to multiple clock domains driven by different PLLs, which is common in modern boards and systems. The freezing/dump feature is also implemented in Sun’s microprocessors [11] [12] [13]. A drawback with this approach is that the internal clock used during normal functional operation is non-deterministic and non-repeatable, which makes it difficult to identify the exact failing moment and debug/diagnose functional failures.

The debug and diagnosis of functional failures based on the trace buffer and freezing/dump techniques described above all rely on a given functional test. Such an approach is labor-intensive due to the lack of automated and efficient diagnostic tools. Furthermore, even if the root cause has been found based on the given functional test, it is hard to recover and port the functional test into an ATE test program. To tackle these problems, a promising approach is to reproduce and detect functional failures using deterministic (i.e., repeatable) test. In this way, we can translate a board-level functional failure into a structural failure. The subsequent diagnosis procedure can be easily performed based on the structural failure because it is deterministic and repeatable. Furthermore, we can derive appropriate tests that can be ported to the ATE.

In this paper, we propose a new technique, which we call *functional scan*, to bridge the gap between functional test and structural test. This technique can provide test conditions close to functional test in a controlled and repeatable manner. We also propose a new framework based on this technique to reproduce and detect the board-level functional failure using

*This research was supported in part by a research grant from Cisco Systems, Inc.

functional scan test. The proposed technique adopts the similar ideas of freezing and dumping of states. However, in prior work, internal clocks during normal functional mode are still non-controllable and non-deterministic when functional test is applied. In contrast, the proposed technique can provide controlled and deterministic internal clocks for both functional test and functional scan test. Furthermore, in the methods reported by Intel and Sun, the debug and diagnosis procedures are based on the given functional test. In this work, we first reproduce the functional failure using functional scan test and next debug/diagnose the failure detected by the functional scan test. The advantages of our proposed technique are: 1) we can reproduce and detect the functional failure in a controlled and deterministic environment; 2) we can diagnose the root cause to a faulty wire/gate inside a component; 3) we can provide the final functional scan test to the supplier, who can port it to the ATE.

Our goal is not to replace functional test or traditional structural test with functional scan test. Instead, we use functional scan test to reproduce and detect functional failures. This approach is different from pseudo-functional test [14], which has been proposed to address the yield loss problem in at-speed scan testing. The goal of pseudo-functional tests is to avoid functionally unreachable states. On the other hand, the purpose of functional scan test is to reproduce and detect functional failures in a deterministic environment when a board fails a functional test. Moreover, there are only two capture cycles in pseudo-functional tests while there are millions of capture cycles in functional scan test so that the functional scan test can adequately mimic the test condition of the given functional test.

The remainder of this paper is organized as follows. Section II describes the motivation for this research with the problems being tackled in this paper. Section III introduces the hardware implementation of the proposed functional scan design and defines functional scan test. Section IV introduces some key definitions. Section V presents the proposed reproduction and detection procedure. The design of experiments and experimental results are reported in Section VI. Section VII concludes the paper.

II. MOTIVATION AND PROBLEM STATEMENT

We use Figure 1 as a motivating example. The dots represent the states close to functional state. Using a functional state or a close-to-functional state as initial state, we can run deterministic test for a reasonable number of clock cycles and obtain a new deterministic-test state space S . Intuitively, if we can use this state space to mimic the functional state space, it is possible for us to use deterministic test to reproduce and detect functional failures.

For example, suppose a fault occurs when a circuit traverses state S_1 in functional test mode. It is possible that this failure will also be detected by deterministic test if S_1 belongs to the deterministic-test state space. Furthermore, let S_1 be the state 1101 and $\bar{S}_1$ be another state 1100. Since the Hamming distance between these two states is small, there is high likelihood that the fault will be detected in deterministic test if $\bar{S}_1$ belongs to the deterministic-test state space.

In this paper, we focus on solving two problems. The first problem is stated as follows:

Fig. 1. An illustration of the state space for different test methods.

Given: A board design including several components;

Goal: Add DFT hardware to each component of the board, such that we can

- provide controlled and deterministic internal clocks;
- lock the circuit state of a component when a functional test fails. The locking can include multiple states before the test fails. The states are determined by the flip-flops in each component;
- dump the desired circuit state;
- recover to a desired circuit state.

The second problem can be stated as follows.

Given: We are given the following:

- a board design with each component having the characteristics described in the first problem;
- the components of the board pass all structural tests;
- the board fails when a functional test is applied;
- there is a golden board available that passes the target functional test;
- the root cause of the functional failure has been located to a specific faulty component.

Goal: Reproduce and detect the functional failure in a controlled deterministic environment and obtain the final test that can be returned to the supplier side and ported to ATE.

III. PROPOSED FUNCTIONAL SCAN DESIGN AND FUNCTIONAL SCAN TEST

A. Functional scan design

To address the first problem, we add functional scan design to each component of the target board. The hardware implementation of functional scan design includes four parts: clock controller, circuit state load and unload, memory contents load and unload, and pseudorandom bit sequence (PRBS) generator. Fig. 2 shows an example of functional scan design for a ASIC, namely ASIC1.

Fig. 2. An example of functional scan design.

1) *Clock controller*: The clock controller is a key module that is needed to support the locking and recovering of a component's circuit state when board-level functional failure occurs. It allows us to start counting and stop the clock at predetermined times. The On/Off of clock pulses is controlled by user-defined trigger signals (such as hardware interrupt or ECC failure) and a programmable counter. For multiple clock domain designs, the clock controller provides the flexibility to adjust clock phase and the latency relationship to mimic the relationship between clocks in functional test. Consider Fig. 2 as an example. Given the reference clock $refclk$, the PLL unit generates free running PLL clocks pll_clks . The signal FS_mode indicates whether the system is in functional scan mode. The signal $trigger$ is used to indicate the beginning of propagating clock pulses of pll_clks to the output clocks $clks_out$. In functional scan mode, once $trigger$ is asserted, the counter inside the clock controller begins to count. After it has counted for a desired number of clock cycles, the clock controller stops propagating clock pulses of pll_clks to $clks_out$ and the output $lock$ is asserted. (Note that $clks_out$ may consist millions of clock cycles thus the power/noise condition of functional scan mode is quite close to that of functional mode.) Since all the clocks to the Core Logic unit will be stopped and there will be no pulses on them when $lock$ is asserted, we can easily lock the circuit state for dumping.

2) *Circuit state load and unload*: This hardware provides the capability to load and unload the circuit state using scan chains. When a board fails in functional test, we use the unload feature to dump the locked functional circuit state of the faulty component. Next we use the load feature to recover the functional circuit state of the component in a controlled deterministic environment. Moreover, the unload feature can also be used to lock and dump the circuit states in the controlled deterministic environment. In Fig. 2, there is a JTAG interface for the circuit state load and unload unit. Under the control of the signal to indicate a beginning of load procedure, which comes from the JTAG interface, and the control of the signal $lock$, the circuit state load/unload unit generates the scan enable signal SE and feeds it to the Core Logic. When the signal SE is asserted, the Core Logic begins to load or unload the circuit state through the scan chain ports SIs or SOs .

3) *PRBS generator*: A PRBS is a statistically random bit sequence. It is usually generated using a linear-feedback-shift-register (LFSR). A PRBS is useful for mimicking the random nature of defects due to the package, data I/O, and the power/noise variation. Here we use a PRBS generator to generate the pseudorandom patterns for the inputs of the faulty component for use in diagnosis. In Fig. 2, the outputs of PRBS generator and the normal functional data are selected to feed the primary inputs (PIs) of the Core Logic unit through a MUX, which is controlled by $fs_capture$. When it is 1, the values for PRBS generator are passed to $Core_in$ of the Core Logic.

B. Functional scan test

Based on the functional scan design, we now define a new type of test, namely functional scan test. It is defined to be a test that involves the following procedure:

- 1) The board enters functional scan mode.

Fig. 3. An example of clock timing for functional scan test.

- 2) We bring the target component to an initial state that is close to a functional state using the load capability. (The “closeness” metric is quantified in Section IV.) Ideally, the initial state is a functional state. However, it is hard to lock and recover to an exact functional state due to the existence of memories.
- 3) Vary PIs and internal flip-flops values of the target component and run desired capture cycles, where the capture clock is from the customized clock controller. In this paper, we only vary values of PIs.
- 4) Dump and observe the circuit state and primary outputs (POs) of the target component using the unload capability.

Figure 3 shows an example of clock timing for a functional scan test. The signals pll_clk1 and pll_clk2 are free running clocks from the PLL. The signal TCK is the slow clock from JTAG. The clocks $clk1$ and $clk2$ are fed to internal cells. In functional scan program mode, we configure the appropriate parameters for the faulty component of the board, such as the number of capture clock cycles, which clock domains to enable, and so on. Next, the FS_mode signal is asserted and the board enters functional scan load mode. In this mode, the target component is brought to the desired initial state. Next, the $FS_capture$ signal is asserted and the board enters functional scan capture mode. In this mode, we vary PIs and internal flip-flops values and run a desired number of capture cycles. Finally, the $FS_capture$ signal is de-asserted and the board enters functional scan unload mode. In this mode, the values of the circuit state and POs are dumped and observed. In the functional scan program mode and functional scan capture mode, clocks to internal cells $clk1$ and $clk2$ come from the PLL clocks. In the functional scan load and unload mode, $clk1$ and $clk2$ come from the slow clock TCK .

Note that the capture procedure of functional scan mode may consist of millions of cycles and the capture clocks are similar to that of normal functional mode. In view of this, functional scan test has a similar clocking scheme compared to functional test. Thus, functional scan test provides similar test conditions as functional test, involving power supply, voltage levels, temperature, and timing. Also, functional scan test can cover the exactly same paths as that covered by functional test, some of which may not be covered in traditional ATPG mode. We can therefore see that functional scan test bridges the gap between functional test and ATPG test. It provides the controllable and deterministic characteristics of ATPG test. On the other hand, it uses the same capture path as in functional test. It can thus provide a test condition close to functional test

in a controlled manner. Once the functional failure has been reproduced and detected in functional scan test, it will be easier to perform the subsequent diagnosis procedure as well as recover and port the test to ATE due to the controllable and deterministic characteristics of functional scan test.

IV. KEY DEFINITIONS

Degree of similarity between two circuit states: Given two states S_i and S_j (with the same number of bits in the binary encoding), the degree of similarity between S_i and S_j , $SD(S_i, S_j)$, is defined as $M(S_i, S_j)/B(S_i)$, where $B(S_i)$ is the number of bits in S_i and $M(S_i, S_j)$ is the number of matched bits between S_i and S_j .

For example, suppose S_1 is 1101 and S_2 is 1100. Then $B(S_1)$ is 4 and $M(S_1, S_2)$ is 3 (the first three bits are matched). Thus the similarity degree $SD(S_1, S_2)$ is 3/4.

State sequence of depth k : Suppose that in k consecutive clock cycles, the circuit state changes as follows: $S_1 \rightarrow S_2 \rightarrow S_3 \dots \rightarrow S_k$. Then $S_1 \rightarrow S_2 \rightarrow S_3 \dots \rightarrow S_k$ is defined to be a state sequence of depth k .

Degree of similarity between two state transitions: Given two state transitions $S_i \rightarrow S_j$ and $\bar{S}_i \rightarrow \bar{S}_j$, the degree of similarity between them is defined as:

$$SD(S_i \rightarrow S_j; \bar{S}_i \rightarrow \bar{S}_j) = \frac{SD(S_i, \bar{S}_i) + SD(S_j, \bar{S}_j)}{2}. \quad (1)$$

Degree of similarity between two state sequences: Given two state sequences of depth k : $S_1 \rightarrow S_2 \dots \rightarrow S_k$ and $\bar{S}_1 \rightarrow \bar{S}_2 \dots \rightarrow \bar{S}_k$, their degree of similarity is defined as

$$DSS(S_1 \dots \rightarrow S_k; \bar{S}_1 \dots \rightarrow \bar{S}_k) = \frac{\sum_{i=1}^k SD(S_i, \bar{S}_i)}{k}. \quad (2)$$

For a given functional test, we can obtain the set of functional states, the set of functional state transitions, and the set of functional state sequences of depth k for any given k . In order to check whether the functional scan state space can adequately mimic the functional state space, we define the following three terms: α -coverage of functional states, α -coverage of functional state transitions, and α -coverage of functional state sequences.

α -coverage of functional states: Suppose we are given a threshold α ($0 \leq \alpha \leq 1$), where α is a user-defined parameter. For a functional state S_i in the functional state set, if we can find a state $\bar{S}_i$ in the set of functional scan states such that $SD(S_i, \bar{S}_i) \geq \alpha$, we say that the functional state S_i is covered with degree of similarity α . Suppose the number of functional states covered with degree of similarity α is $n(state_alpha)$ and the number of states in the functional state set is $n(func_state)$. The α -coverage of functional states is defined by:

$$Cov_S_alpha = \frac{n(state_alpha)}{n(func_state)} \quad (3)$$

α -coverage of functional state transitions: Once again, suppose we are given a threshold α ($0 \leq \alpha \leq 1$), where α is a user-defined parameter. Consider a functional state transition $S_i \rightarrow S_j$ in the set of functional state transitions. If we can find another state transition $\bar{S}_i \rightarrow \bar{S}_j$ in the set of functional scan state transition such that $SD(S_i \rightarrow S_j, \bar{S}_i \rightarrow \bar{S}_j) \geq \alpha$, we say that the functional state transition $S_i \rightarrow S_j$ is

covered with degree of similarity α . Suppose the number of functional state transition covered with degree of similarity α is $n(statetrans_alpha)$ and the number of state transitions in the set of functional state transition is $n(func_state_trans)$. The α -coverage of functional state transition is defined as follows:

$$Cov_ST_alpha = \frac{n(statetrans_alpha)}{n(func_state_trans)} \quad (4)$$

α -coverage of functional state sequence of depth k : Consider a functional state sequence $S_1 \rightarrow S_2 \dots \rightarrow S_k$ of depth k . If we can find a state sequence $\bar{S}_1 \rightarrow \bar{S}_2 \dots \rightarrow \bar{S}_k$ of depth k in the set of functional scan state sequence such that $DSS(S_1 \rightarrow S_2 \dots \rightarrow S_k, \bar{S}_1 \rightarrow \bar{S}_2 \dots \rightarrow \bar{S}_k) \geq \alpha$, we say that the functional state sequence $S_1 \rightarrow S_2 \dots \rightarrow S_k$ is covered with degree of similarity α . Suppose the number of functional state sequence of depth k covered with degree of similarity α is $n(state_seq_k_alpha)$ and the number of state sequences of depth k in the set of functional state sequence is $n(func_state_seq_k)$. The α -coverage of functional state sequence of depth k is defined as follows:

$$Cov_SS_k_alpha = \frac{n(state_seq_k_alpha)}{n(func_state_seq_k)} \quad (5)$$

Our premise is that if we can obtain high Cov_S_alpha , high Cov_ST_alpha , and high $Cov_SS_k_alpha$ for a large value of α , the functional scan state space can mimic the functional state space well. This implies that the functional scan test adequately reproduces the functional failures.

V. METHODOLOGY FLOW

In this section, we describe how functional scan test is used to reproduce and detect functional failures. Three major steps are included in the proposed flow:

Step 1: Divide the functional test into stages and find the first failing stage. A functional test sequence may contains millions of clock cycles. We first partition the given functional test sequence into several stages according to its functionality. At the end of each stage, we check the values of related control and status registers (CSRs). In this way, we can easily identify the first failing stage for the given functional test.

Step 2: Obtain the initial cycle and dump initial state in functional test. We first define the following terms.

Faulty cycle: Suppose a failing stage contains N cycles $C_1, \dots, C_N$. Suppose at the strobe point of C_i ($i = 1, 2, \dots, N$), the good board and faulty board have different values on observation points (CSRs or scan flip-flops or POs). Then we refer to C_i as a faulty cycle.

k -cycle fault: If k cycles are needed to activate a fault and capture the fault effect, the fault is referred to as a k -cycle fault.

Initial cycle and initial state: Suppose C_i is a faulty cycle in a functional test. The initial cycle of C_i is C_j , where j equals $i - k$ for a k -cycle fault. The circuit state corresponding to the initial cycle is defined as the initial state.

It is hard to apply functional vectors on the ATE due to the following reasons: 1) High cost of functional vector translation/transformation; 2) Strict timing constraints on ASIC boundary signals are required, which requires a high-capacity tester. Therefore, we propose to use functional scan test to reproduce the failure. In order to do so, we first use binary

search to find a narrowed-down faulty cycle in functional test for the first failing stage. Then we find the initial cycle corresponding to this faulty cycle. Next, we dump the circuit state of the initial cycle from the good board. This dumped state will be used in the next step to reproduce the functional failures.

Step 3: Reproduce and detect functional failures in functional scan test. We next reproduce and detect the functional failure in functional scan test starting from the initial state (dumped state in Step 2). By running functional scan test with varying PI values and applying appropriate functional constraints (extracted according to the given functional test and the specification of the design) for a desired number of clock cycles, we can check whether the functional failure is reproduced in functional scan test. Details about the strategy of varying PI and extraction of functional constraints can be found in [15].

VI. EXPERIMENTAL RESULTS

To evaluate the proposed method, we performed experiments on an industry design (called *design_A* in this paper). The industry design is a multi-port Ethernet network interface to the HULC stacking system. It has been implemented in 65 nm technology. In the final gate-level netlist, there are about 176K flip-flops and 4M gates. A test board has been built by putting two instances of *design_A* together. The functional test for *design_A* comes from the design-verification suites, provided by its developers.

Our first goal is to show that with proper functional constraints, functional scan test can adequately mimic the functional state space, measured by Cov_{S_α} , Cov_{ST_α} , and Cov_{SSk_α} . Recall that this is the theoretical basis for using functional scan test to reproduce and detect functional failures. Our second goal is to show that the proposed flow can efficiently reproduce and detect functional failures by using functional scan test. Thus it can be used to return the narrowed-down test to the supplier for ATE use.

A. Experimental setup

All experiments were performed on a 64-bit Linux server with 4 GB memory. Synopsys Verilog Compiler (VCS) was used to run Verilog simulation. All other programs were implemented using C++ and Perl scripts.

B. Mimicking of functional state space

1) *Substate group partition:* It is impractical to consider the complete state space of a board or even the complete state space of a component due to design complexity. Furthermore, sometimes it is unnecessary and meaningless to consider all the states. In this paper, we determine the substate groups at register transfer (RT)-level by considering the functionality of the design. Our key idea is to place registers that share common conditions for assignment operations in one substate group.

2) *Simulation results:* First, we partition the state space into several substate groups. Next, for each substate group, we record the number of unique functional states, state transitions, and state sequences traversed by the given functional test. Then, we apply functional scan test and calculate the α -coverage for functional states, state transitions, and state

Fig. 4. Distribution of the number of unique functional states (*design_A*).

Fig. 5. Distribution of the number of unique functional state transitions (*design_A*).

sequences. In functional scan test, we use a PRBS generator to generate patterns for inputs of *design_A*. Also, for *design_A*, we extract and apply functional constraints during functional scan test. Finally, we observe whether we can obtain high values of α -coverage metrics with a large value of α .

For *design_A*, we consider 63 substate groups that are related to the given function test. The number of flip-flop bits for 53% of the substate groups belongs to the range of 10-49. The functional test for *design_A* includes 29,317 clock cycles and the functional scan test has 100,000 capture cycles. In functional scan test, we use a PRBS generator to generate patterns for the inputs of *design_A*. Also, we extract and apply functional constraints during functional scan test.

Figures 4-6 highlight the functional state space information for *design_A*. The distribution of the number of unique functional states, unique functional state transitions, and unique functional state sequences of depth 3 under the given functional test are shown in these figures, respectively. For example, from Figure 4 we can see that 55% substate groups traverse 1-9 unique functional states under the given functional test. These figures also show that most of the substate groups traverse only a small number of unique functional states, unique functional state transitions and unique functional state sequences of depth 3. For example, although the given functional test includes 29,317 clock cycles, Figure 4 shows that only 2% substate groups traverse 1,000 or more unique functional states.

Table I lists the average values of α -coverage metrics on all groups for functional scan test of *design_A*. Column 1 shows the various value for parameter α . Columns 2-4 list the average values of α -coverage metrics when functional constraints are not applied. Columns 5-7 refer to the case when functional constraints are applied. From this table, we can see that higher the value of α , lower the value of the α -coverage metrics. For example, with no functional constraints, we obtain 86% for Cov_{S_α} when α is set to 0.8, but only

Fig. 6. Distribution of the number of unique functional state sequences of depth 3 (*design_A*).

TABLE I. α -coverage metrics for functional scan test for *design_A*.

α	No constraints			With constraints		
	$Cov_{S-\alpha}$	$Cov_{ST-\alpha}$	$Cov_{SS-k-\alpha}$ ($k=3$)	$Cov_{S-\alpha}$	$Cov_{ST-\alpha}$	$Cov_{SS-k-\alpha}$ ($k=3$)
0.8	86%	85%	84%	91%	89%	87%
0.85	81%	79%	78%	87%	84%	81%
0.9	74%	72%	70%	81%	77%	72%

81% when α is set to 0.85. This is because higher α means we need more similarity between a functional state and a functional scan state, an objective that is more difficult to achieve; thus we get a lower value for α -coverage. Therefore, a balance is needed between the values of α and the α -coverage. This table also shows that by applying appropriate functional constraints during functional scan test, we can increase values for the α -coverage metrics for the same value of α . We observe that by applying functional constraints during functional scan test, we obtain reasonably high values for α -coverage metrics with high α . The above results indicate that the functional state space can be mimicked well, thus it is possible to use functional scan test (with proper functional constraints) to reproduce and detect functional failures for *design_A*.

C. Reproducing and detecting functional failures

In this section, we use the flow in Section V to reproduce and detect functional failures in functional scan test. In the reproduction and detection procedure, for *design_A*, we divide the functional test into 3 stages. Then we find the first failing stage. Next, we obtain the initial cycle and dump the initial state in functional test. We start from the initial state and apply functional scan test (values of PIs come from the PRBS generator and the extracted functional constraints). Finally, we check whether the functional failure is reproduced and detected in functional scan test.

We first artificially create several faulty boards. Each faulty board is obtained by injecting one stuck-at, dominant bridging, or crosstalk fault on a fault-free board. Crosstalk usually results from capacitive coupling between wires [16]. We randomly generate aggressor/victim pairs in this paper. When the aggressor and victim nets fall or rise simultaneously and in opposite directions, a delay is injected to the falling or rising of the victim net.

Table II list the number of functional failures detected by using functional scan test for *design_A*, respectively. Column 2 shows the number of functional failures due to each type of fault. Column 3 shows the number of faults detected in functional scan test. Column 4 shows the percentage of detected functional failures, relative to the total functional

TABLE II. Functional failure detection using functional scan test for *design_A*.

Fault model	No. of functional failures	No. of detected faults in functional scan test	Percentage of success
stuck-at	100	100	100%
bri-dom	100	100	100%
crosstalk	65	54	83.08%

failures. We find that for *design_A*, functional scan test can detect most of the functional failures due to all three types of injected faults. In particular, all the 100 functional failures each due to stuck-at fault and bridging fault can be detected using functional scan test.

VII. CONCLUSIONS

We have proposed a DFT technique for analyzing board-level functional failures. We also presented a new method for using functional scan test to reproduce and detect board-level functional failures. The DFT technique and the associated reproducing and detection flow allow us to reproduce and detect functional failures in a controlled deterministic environment, following which we can diagnose the root cause of a board-level functional failures to obtain tests that can be ported to ATE for defect screening. Experiments for an industry design first show that functional scan test with appropriate functional constraints can mimic the functional state space adequately, thus it is feasible to use functional scan test to reproduce and detect functional failures. Experiments also show that most functional failures due to stuck-at, bridging and crosstalk faults can be detected using functional scan test with appropriate functional constraints.

REFERENCES

- [1] P. C. Maxwell, I. Hartanto, and L. Bentz, "Comparing functional and structural tests," in *Proc. ITC*, 2000, pp. 400–407.
- [2] J. Gatej *et al.*, "Evaluating ATE features in terms of test escape rates and other cost of test culprits," in *Proc. ITC*, 2002, pp. 1040–1049.
- [3] J. Rearick and R. Rodgers, "Calibrating clock stretch during AC scan testing," in *Proc. ITC*, 2005, pp. 266–273.
- [4] R. Tulloss, "Fault dictionary compression: Recognizing when a fault may be unambiguously represented with a single failure detection," in *Proc. ITC*, 1980, pp. 368–370.
- [5] J. Richman and K. Bowden, "The modern fault dictionary," in *Proc. ITC*, 1985, pp. 696–702.
- [6] S. Venkataraman and W. Fuchs, "Deductive technique for diagnosis of bridging faults," in *Proc. International Conference on Computer-Aided Design*, 1997, pp. 562–567.
- [7] T. Vo *et al.*, "Design for board and system level structural test and diagnosis," in *Proc. ITC*, 2006, pp. 409–418.
- [8] M. Abramovici *et al.*, "A reconfigurable design-for-debug infrastructure for SoCs," in *Proc. Design Automation Conference*, 2006, pp. 7–12.
- [9] J.-S. Yang and N. A. Touba, "Expanding trace buffer observation window for in-system silicon debug through selective capture," in *Proc. VTS*, 2008, pp. 345–351.
- [10] A. Carbine and D. Feltham, "Pentium®pro processor design for test and debug," in *Proc. ITC*, 1997, pp. 294–303.
- [11] H. Hao and K. Bhabuthmal, "Clock controller design in SuperSPARCTM II microprocessor," in *Proc. IEEE International Conference on Computer Design*, 1995, pp. 124–129.
- [12] H. Hao and R. Avra, "Structured design-for-debug - the SuperSPARCTM II methodology and implementation," in *Proc. ITC*, 1995, pp. 175–183.
- [13] I. Parulkar *et al.*, "DFX of a 3rd generation, 16-core/32-thread UltraSPARCTM CMT microprocessor," in *Proc. ITC*, 2008.
- [14] Y.-C. Lin *et al.*, "Pseudo-functional testing," *IEEE Trans. CAD*, vol. 25, pp. 1535–1546, 2006.
- [15] H. Fang *et al.*, "Mimicking of functional state space with structural tests for the diagnosis of board-level functional failures," in *Proc. ATS*, 2010.
- [16] C. Werner *et al.*, "Crosstalk noise in future digital CMOS circuits," in *Proc. DATE*, 2001, pp. 331–335.