Observability-aware Directed Test Generation for Soft Errors and Crosstalk Faults

Kanad Basu
Syst. and Tech. Dev.
IBM India Private Limited
email: kanbasu2@in.ibm.com

Prabhat Mishra
Comp. and Info. Sc. and Engg.
University of Florida, USA
email: prabhat@cise.ufl.edu

Priyadarsan Patra
Post-Si Validation Architecture
Intel Corporation, USA
email: priyadarsan.patra@intel.com

Abstract—**Post-silicon validation has emerged as an important component of any chip design methodology to detect both functional and electrical errors that have escaped the pre-silicon validation phase. In order to detect these escaped errors, both controllability and observability factors should be considered. Soft errors and crosstalk faults are two important electrical faults that can adversely affect the correct functionality of the chip. A major bottleneck with the existing approaches is that they do not consider the inter-dependence of the selected trace signals and test generation. In this paper, we explore the synergy between trace signal selection and observability-aware test generation to enable efficient detection of electrical errors including soft errors and crosstalk faults. Our experimental results demonstrate that our approach can significantly improve error detection performance - on an average 58% for crosstalk faults and 48% for soft errors compared to existing techniques.**

I. Introduction

An Integrated Circuit (IC) must be error free before it is delivered to a customer. Due to reduced time-to-market and increasing design complexity, a lot of errors escape the pre-silicon validation phase and manifest themselves in the manufactured chip. Post-silicon validation is used to capture these bugs. An overview of post-silicon validation is shown in Figure 1. A small set of signals are traced during execution. During debug, these signal states are used for failure analysis and error localization.

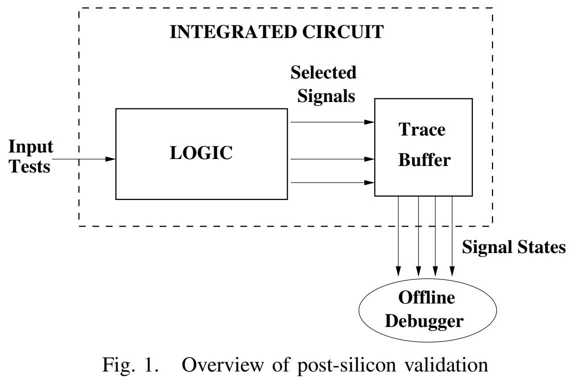


Fig. 1. Overview of post-silicon validation

Electrical defects manifest as important errors in modern SoCs. Soft errors and crosstalk faults are two important defects that can adversely affect the correct functionality of the chip. While soft errors are caused by radioactive effects on design impurities, crosstalk faults occur due to imperfect coupling capacitance between two lines in the chip. Soft errors can be modeled using single stuck-at-faults. Effects of crosstalk can be represented as glitches and delay faults. Effective directed test generation strategies need to be employed in order to detect these faults. The tests should be able to activate the faults and propagate them towards the observation points, e.g., primary outputs or internal trace signals.

The two primary challenges governing efficient error detection are controllability and observability. During post-silicon validation, not all the primary outputs of a design block are visible (since they may be internally connected to some other components of the design). Also, the number of primary outputs of a circuit is typically larger than the trace buffer width, which determines the number of signal states that can be stored per cycle. Hence, the primary outputs alone can not be used as observation points. Existing methods [2], [7], [3] on signal selection assume that the input tests are always random in nature. However, once the trace signals are known, Automatic Test Pattern Generation (ATPG) tools can be used to generate efficient directed tests for error detection if the probable error locations are available. In modern System-on-Chip (SoC) design methodology, it is found that regions where errors are detected during pre-silicon verification are more likely to be erroneous during post-silicon validation. Therefore, the pre-silicon engineer can provide information about the probable erroneous locations or zones for post-silicon validation. These observations can be utilized for efficient observability-aware test generation as outlined in Figure 2.

Our proposed approach takes as input the circuit and the fault list. In the first step, we perform test generation considering the primary outputs as observation points to obtain a set of directed tests for error detection. Next, we use these tests to determine the profitable trace signals. This process continues until the fault-coverage reaches 100% or does not improve in subsequent iterations. The overview of our proposed approach is shown in Figure 3. To the best of our knowledge, it is the first attempt to study the synergy between signal selection and test generation in the context of post-silicon validation

This work was partially supported by NSF grants CNS-0746261 and CCF-1218629.

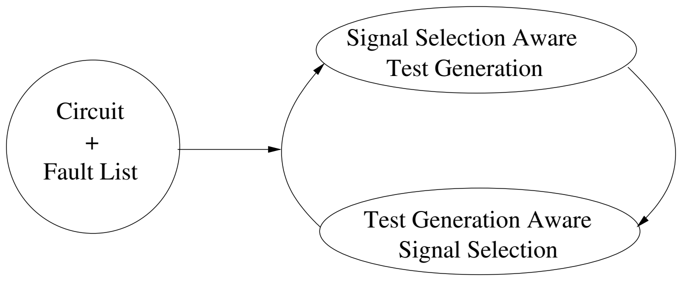


Fig. 2. Proposed framework

and debug. We have modified existing ATPG techniques to make them suitable in the presence of selected trace signals, and vice versa.

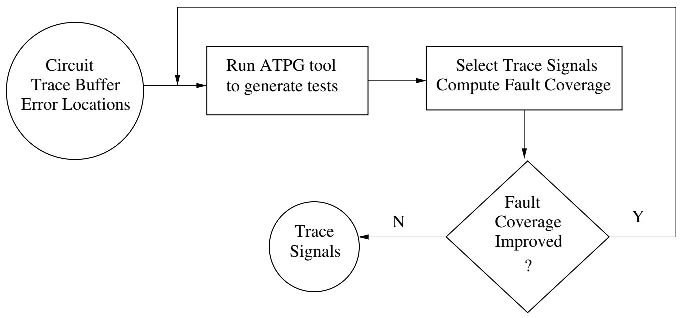


Fig. 3. Observability-aware test generation flow

The rest of the paper is organized as follows. Section II presents related work on post-silicon validation and debug. Section III describes our test-aware signal selection algorithm. Section IV presents our observability-aware test generation algorithm. Section V presents our experimental results. Finally, Section VI concludes the paper.

II. Related Work

Limited observability is a primary concern during post-silicon debug. Once the internal signal states are known, debug algorithms like failure propagation tracing [4] can be used to identify the errors in the circuit. Design-for-Debug (DfD) techniques such as embedded logic analyzer (ELA) [1] have been used extensively to increase the observability of internal signals of the circuit. Using ELA, some of the internal signal states of a circuit are stored in an on-chip trace buffer. Since the trace buffer size is limited, efficient trace signal selection techniques [2], [3], [7], [5] are necessary to improve the overall observability of the SoC.

Soft errors and crosstalk faults are two major electrical errors found in a fabricated SoC. Effect of soft errors on memory devices had been studied by May et al. [11]. Over the years, researchers [6], [12] have studied various aspects of soft errors. Sanyal et al. [13] have proposed different methods for directed test generation for soft errors. Crosstalk faults occur when two lines in a circuit are so near that their mutual capacitance affects their state. Effects of crosstalk faults on digital circuits [14], [9] have been studied extensively. There are various test generation algorithms for crosstalk faults [15], [10]. These approaches assume that all the output signals of a logic block are visible. However, during post-silicon validation, since the chip is fabricated, observing the output signals of every component may not be feasible since these components can be embedded in an SoC. The only observable points would be the trace signals. The test generation algorithms need to be modified to take this into account.

III. Test-aware Signal Selection

Traditional ATPG tools generate tests assuming all the primary outputs as observation points. Once the tests are determined, a set of trace signals need to be determined to improve the error detection performance. In general, during selection of trace signals, the input tests are assumed to be random. We would like to look at a special case when the input test sets are known prior to signal selection. Knowledge of input tests can be used to determine the signals very efficiently, specially, when the main focus is error detection. Our signal selection procedure is presented in Algorithm 1. The remainder of this section describes the three important steps of the signal selection algorithm.

Algorithm 1: Test-aware signal selection

```
Input: Circuit, Trace Buffer Width, Test Set $T$
Output: Trace Signals
for Each Test Vector do
    1: Simulate each fault in the circuit.
    2: For each signal in the circuit, determine whether it
    can detect the fault.
end
3: Compute the error detection ability of each signal.
while Trace buffer width is not reached do
    4: Select the signal with the highest error detection
    capability.
    5: Remove overlap.
end
Return Selected trace signals.
```

A. Fault Simulation

The best way to know whether a fault can be detected using a particular observation point and a test vector is to simulate the fault and notice the state of the observation point. Since we already have a set of test vectors, the fault simulation is straightforward. For each test vector, we first do a simulation of the golden (correct) design and observe the correct states of the various signals. Now, we perform simulation for every fault with the same test vector. For each fault, the signal states of the circuits are observed. If they are different from the ideal simulation, it is obvious that the fault is propagated to that signal. This process is repeated for each test case and each fault. For example, if there are m test vectors and n faults, there will be a total of $m \times n$ simulations. For each signal, we note the faults that it can detect. This is recorded as a binary variable Error Propagation Probability (EPP). For example, in Figure 4, if c can detect an error in a using any of the test vectors, $EPP_{c,a} = 1$. On the other hand, since d can never detect any error in a, $EPP_{d,a} = 0$.

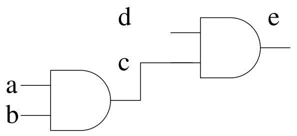


Fig. 4. Example circuit

B. Error Detection Ability Computation

Error Detection Ability (EDA) of a node (signal) is a measure of the errors that a particular node can detect. A node can only detect errors in its fan-in cone. For example, if we consider Figure 4, any error in *c* can only propagate to *e* and not to *a*, *b* or *d*. Therefore, the only nodes whose errors *c* can detect are *a* and *b*. EDA of a node is the sum of all the errors that are detected using fault simulation.

$$EDA_c = EPP_{c,a} + EPP_{c,b} \tag{1}$$

It should be noted that a node can detect an error using multiple test cases, however, it should be counted only once. We enforce this by ensuring that EPP is a Boolean number. For example, if by simulating 2 test cases, *c* can detect *a* and *b* in both cases, EDA_c would be 2 and not 4. Once EDA value for each node is computed, the node with the highest EDA value is selected for tracing. The next section describes how to remove the overlap of already selected signals before determining the next profitable signal.

C. Overlap Removal

This part of the signal selection algorithm is used to remove effects of already selected signals and thus, select appropriate signals for improved error detection. In order to explain this, let us re-visit Figure 4. Let us consider node *c*, which is the first node to be selected for tracing. If $EPP_{c,a} = 1$ and $EPP_{c,b} = 1$, that is, the errors in *a* and *b* can propagate to *c*, contributions of $EPP_{e,a}$ and $EPP_{e,b}$ should not be included while computing EDA_e. In this case,

$$EDA_e = EPP_{e,c} + EPP_{e,d} \tag{2}$$

Thus, overlapping nodes, whose contributions have already been accounted for, should not be taken into account when computing the EDA value of a node. The process of signal selection continues until the trace buffer is full.

IV. Observability-aware Test Generation

Once the set of selected signals are known, the next step would be to generate another set of tests based on these signals that can maximize the error detection ability. We used Atalanta [16] as an ATPG tool that generates tests depending on the fault list and considering the output nodes as observation points. In order to generate tests based on the selected trace signals, we modify the netlist to replace the trace signals as observation points. The ATPG engine will generate the tests assuming the trace signals as the observation points. The fault coverage using these new set of tests and the selected signals is computed. If the coverage does not improve, the set of selected signals are reported as trace signals. The tests generated using the ATPG tool are used as directed input tests. If the fault coverage improves, the process in Section III is repeated to generate better trace signals and associated tests to further improve error detection performance. The time and memory requirements for the test generation process depends on the performance of the *ATPG* tool used. The remainder of this section describes modeling and test generation to detect soft errors and crosstalk faults.

A. Test Generation for Soft Errors

Soft errors are caused due to ionizing radiations from radioactive impurities present in a chip during manufacture. These may result in ionizing radiations like alpha-particles. When these alpha particles come in contact with a semiconductor, their kinetic energy gets converted to electrical energy [8], which results in a large number of free electrons and holes. This leads to a creation of an inversion layer as well as a voltage glitch on the affected transistor. If the glitch is of sufficient magnitude, a faulty logic value is introduced temporarily on a node in the circuit. This is known as Single Event Transient (SET). If the faulty value is propagated to a primary output or an observation point, the event is known as Single Event Upset (SEU). We try to generate directed tests to detect all SETs resulting in possible SEUs.

The error model that is used for modeling soft-errors is a simple stuck-at fault model. The nodes affected by radiations get stuck at certain fixed values depending on the amount of free electrons or holes created. The effect of soft errors on a node value depends on output capacitance as well as pull-up and pull-down networks. A weaker capacitance makes a node more susceptible to soft errors. Weak pull-up and pull-down networks can lead to stuck-at-0 and stuck-at-1 faults respectively.

Detection of soft errors require generation of test cases that would activate the particular errors and propagate them to the observation points. As discussed before, during post-silicon validation, the erroneous values should propagate towards the traced signals and not to the primary outputs. The test generation problem should focus on generating a set of test cases that would activate and propagate a maximal number of soft errors (if possible, all of them) to the observation points, that is, the trace signals.

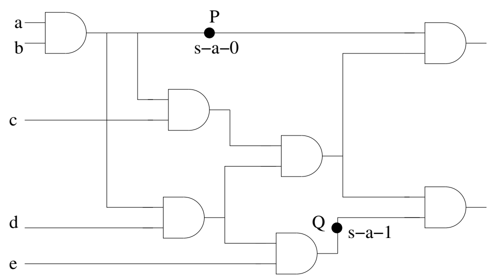


Fig. 5. Example circuit illustrating test generation for soft errors

Let us consider the example in Figure 5 to explain the test generation problem for soft errors. Consider two error points *P* and *Q*, where the errors can be represented as $s-a-0$

and $s-a-1$, respectively. We would like to generate tests that would activate them as well as propagate them to the observation points. The 5 input signals are $< a,b,c,d,e >$. To detect the $s-a-0$ fault at P, the input tests should be $< 1,1,1,1,X >$ whereas the test required to detect the $s-a-1$ fault at Q, the input tests should be $< 1,1,1,1,0 >$. Therefore, the test that can detect both faults is $< 1,1,1,1,0 >$. ATPG algorithms can be designed to generate tests that would detect these faults.

Algorithm 2 describes our test generation procedure for soft-errors. It should be noted that this algorithm is applicable for test generation when multiple soft errors are present simultaneously. In the first step, we identify all the internal signals (gate signals and fan-out branches) in the soft error affected zone. For each of the signals in the fault list, we perform test generation for $s-a-0$ and $s-a-1$ faults using ATPG.

Algorithm 2: Test generation for soft error detection

```
Input: Circuit, Trace signals, Soft-error affected zone $Z$
Output: Test set to detect the faults
1: Find the signals corresponding to $Z$.
Signals corresponding to nodes as well as fan-out signals.
2: Create a fault list with stuck-at-0 and stuck-at-1 at
each node.
3: Use ATPG to generate tests for these faults.
Use the trace signals as observation points.
Return the set of tests.
```

B. Test Generation for Crosstalk Faults

Crosstalk faults are caused by parasitic coupling capacitances between adjacent lines in a chip [9]. With decrease in feature size, effect of coupling capacitances and hence, crosstalk faults become more prominent, thus leading to signal integrity problems [10]. Crosstalk faults are caused when the coupling capacitance between two lines exceeds a certain threshold. In such a case, if there are transitions on either or both the lines, the transition on one will influence the other and hence, the voltage levels change causing either a delay or a glitch. The line whose voltage level changes is known as victim, while the line which changes the voltage level is called aggressor. We will explain crosstalk glitches and delays using the example circuit in Figure 6 which has 5 lines (signals), namely a,b,c,d and e. Let us assume the coupling capacitances between lines c and d exceed the threshold so that they can act as probable aggressor-victim pairs.

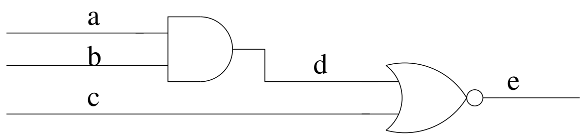
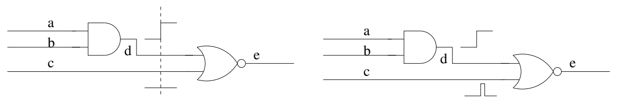


Fig. 6. Example circuit illustrating crosstalk faults

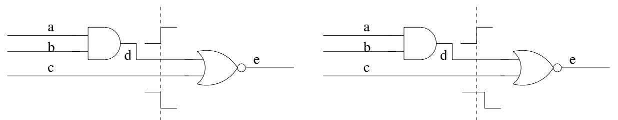
During crosstalk glitch, the victim line stays at a static state, while the aggressor undergoes a transition. If the transition effect is opposite to the state of the victim, a glitch is created. For example, if the victim is at a state of 0, while the aggressor has a positive transition, a positive glitch is formed on the victim line. Similarly, if the victim line is in a state of 1 and a negative transition is formed on the aggressor line, a negative glitch is created. Figure 6 has been redrawn in Figure 7(a) to show that when line c is in a steady state and line d transits, a positive glitch on line c is formed as shown in Figure 7(b).



(a) Source of crosstalk glitch (b) Crosstalk glitch

Fig. 7. Positive glitch on c

On the other hand, delays are created when both aggressor and the victim undergo transition. If the transitions are in the same direction, the overall delay is reduced. If the transitions are in opposite direction, the signal propagation delay is increased. Figure 8(b) shows a positive delay on line c due to transitions on both lines c and d (Figure 8(a)).



(a) Source of crosstalk delay (b) Crosstalk delay

Fig. 8. Positive delay on c

It should be noted that both the transitions need to be simultaneous in order for the delay to take effect. As can be seen in Figure 9, if the two transitions are not simultaneous, there will not be any delay.

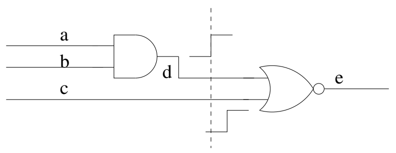


Fig. 9. Non-simultaneous transitions

The effect of crosstalk fault, that is, delay or glitch will be propagated to fan-out gates. In case of sequential circuits, if the glitch duration or delay is less than the clock frequency, it gets suppressed. However, for combinational circuits, the effects get propagated to the outputs.

The test generation algorithm for crosstalk faults is shown in Algorithm 3. The first step of the algorithm is to find all the aggressor-victim pairs by observing their coupling capacitances. In this case, we consider single aggressor-single victim pairs only. However, the algorithm can be extended to multiple aggressors as well. Information on coupling capacitances is obtained from the layout information of the chip. Once we have identified all the pairs, the next step would be to generate

Algorithm 3: Test generation for crosstalk fault detection

```
Input: Circuit, list of coupling capacitances, threshold
Output: Test set to detect the faults
1: Find all the pair of lines that contribute to crosstalk faults.
2: Duplicate the circuit.
3: Use ATPG to generate tests for these faults.
Return Test set.
```

tests that would provide transitions on either or both the lines depending on the desired type of crosstalk effect.

We now explain our algorithm using crosstalk delay, that is, transitions should be present on both lines. Crosstalk glitches can be explained in a similar way. Duplication of circuit is needed to create transitions on both aggressor and victim. For a combinational circuit, which does not have a clock signal, in order to emulate a transition, we need to make sure that the signals on a particular line change in two adjacent time units. Let us consider the example circuit in Figure 6. Suppose, lines c and d have been identified as crosstalk pairs. We would like to generate two sets of tests, such that they fire transitions on both these lines. If we want to observe the effect of crosstalk glitch, transition should be enabled on only one line. In order to generate the transitions, we have duplicated the circuit in Figure 10[1]. Corresponding to each signal in Figure 6, there is a corresponding signal in Figure 10. For example, signal a in Figure 6 will be duplicated as $a\prime$ in Figure 10. Thus all the inputs are duplicated as well. The ATPG tool is used to generate the tests. In this example, the inputs to a, b, c correspond to the test in the first time frame, while inputs to $a\prime, b\prime, c\prime$ correspond to test in the second time frame. Thus, in order to generate a transition at line c in Figure 6, the signals c and $c\prime$ should be different in Figure 10. This can be forced by connecting an exclusive-or gate, whose two inputs are c and $c\prime$. Since an exclusive-or gate will be 1 only when the two inputs are different, this ensures a transition in line c in Figure 6. Similarly, d and $d\prime$ in Figure 10 are input to another exclusive-or gate, thus, forcing a transition in d in Figure 6. We want to generate test cases that would provide transitions on both lines. This is ensured by connecting an AND gate at the output of the two XOR gates.

The ATPG tool is then used to generate tests so that the output o of the AND gate is 1. This ensures transition on both lines. The ATPG tool can be run with a fault list including the point o is $s-a-0$. In this case, the ATPG tool will generate test to force o to be 1, and hence ensure a transition on both lines.

We want the delay at the victim to be propagated to the output. In order to ensure that, o is connected to the fan-out branch of the victim and thus propagated to an observation point, or primary output. For example, if d (or $d\prime$) is the victim line in Figure 10, which incurs some delay, we add o to the fan-out cone of $d\prime$, in order to ensure that the delay in d in Figure 6 actually gets propagated to a primary output e ($e\prime$ in this case). The modified circuit is shown in Figure 11. If we have trace signals, the observation points (trace signals) are enabled such that the ATPG generates tests which propagate the delay to these trace signals. Similar test-generation procedure can be applied for crosstalk glitches, in which case, the transition should be only along the aggressors.

[1]It should be noted that the duplication of the circuit is for test generation purpose only. There is no extra hardware overhead (since the circuit is not duplicated in actual hardware) associated with it.

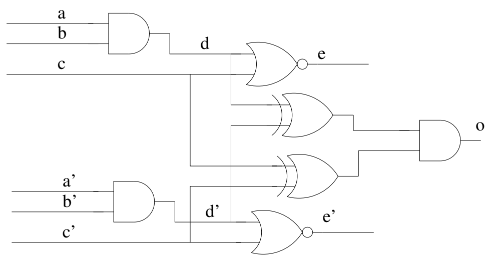


Fig. 10. Duplicated circuit of Figure 6

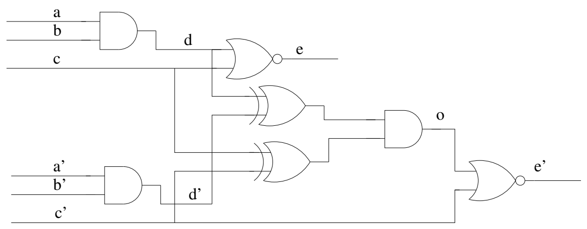


Fig. 11. Modified circuit of Figure 10

V. Experiments

We have applied our proposed approach on the ISCAS '85 combinational benchmarks. We compare the performance of our test-aware signal selection algorithm with the existing signal selection algorithm [7]. We have chosen [7] over the other trace signal selection algorithms [2], [3], [5] since the former is found to provide better performance than the others. A simulation of 1000 cycles is run assuming no error is present. The input to the circuit is fed with the test sets generated by Algorithm 2 or Algorithm 3. Then, for each error, simulation of 1000 cycles are performed assuming the error is present in the circuit. If any of the traced signal states during this simulation is different from the perfect simulation, an error is said to be detected. Error Detection Ratio (EDR) is chosen for comparing error detection performance. EDR is defined as:

$$EDR = \frac{Number\ of\ Errors\ Detected}{Number\ of\ Detectable\ Errors}$$

We present our results in two categories. First, we compare our approach with existing methods for soft error detection. Next, we present results for detection of crosstalk faults.

A. Detection of Soft Errors

First, we consider only soft errors and we applied 250 errors in each circuit. Random nodes are selected as error points. Algorithm 2 uses the ATPG tool Atalanta [16] to generate the directed tests to detect these soft errors. The comparison of EDR for 5 of the largest ISCAS '85 benchmarks is shown in Figure 12. The *proposed method* column refers to our proposed test-aware signal selection algorithm. On the other hand, the *existing approaches* column refers to existing trace signal selection algorithm. It should be noted that to make a fair comparison, we have used the same set of tests in both scenarios.

As can be seen in Figure 12, our proposed method performs consistently better than the profile-based signal selection algorithm, with an average improvement of 48.2%. Our algorithm uses tests as inputs to select signals, which gives a better insight during error propagation probability computation, and hence, subsequent selection of trace signals to detect errors.

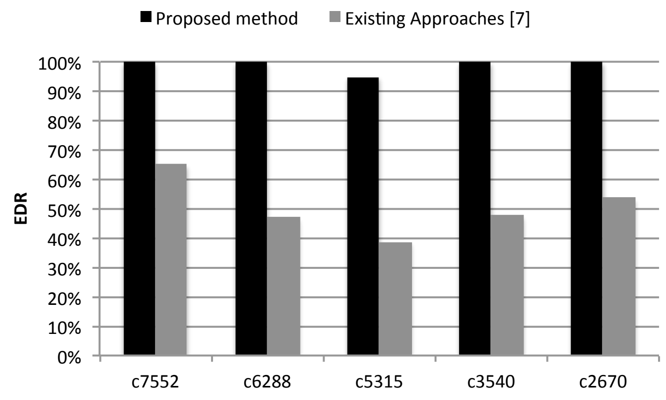


Fig. 12. Comparison of signal selection methods for soft errors

B. *Detection of Crosstalk Faults*

Now, we would like to observe the performance of our test-generation algorithm for crosstalk faults described in Section IV-B. Similar to Figure 12, we have used the 5 largest ISCAS '85 benchmarks. The pre-processing step in Algorithm 3 requires manual modification of the circuit in order to insert the additional AND and XOR gates described in Figure 11. Hence, we reduced the number of faults from 250 to 10. Similarly, the trace buffer width is also reduced to 4 instead of 32, that is, in this case, 4 signals will be stored every cycle. In order to make fair comparison, the same experiment is repeated for profile-based signal selection technique using the same set of parameters. The results are shown in Figure 13. As can be seen, our proposed method provides significant improvement over the existing technique, with an average improvement of 58%. The performance of existing method is very poor for detection of crosstalk faults. The extreme scenario is for $c3540$ where existing method could not detect any of the errors whereas our approach captured all of them. The reason behind this is the limited number of errors and limited trace buffer width of 4. These comprise of less than 1% of the total number of signals in the design.

VI. Conclusion

Limited observability is a major bottleneck in detecting errors during post-silicon validation and debug. In this paper, we presented an efficient observability-aware test generation technique that selects efficient observation points and generates corresponding test sets to improve detection of electrical errors during post-silicon debug. We evaluated the effectiveness of our approach using ISCAS '85 benchmarks. Our approach can provide on an average 58% improvement in error detection performance for crosstalk faults (48% for soft errors) compared to existing techniques.

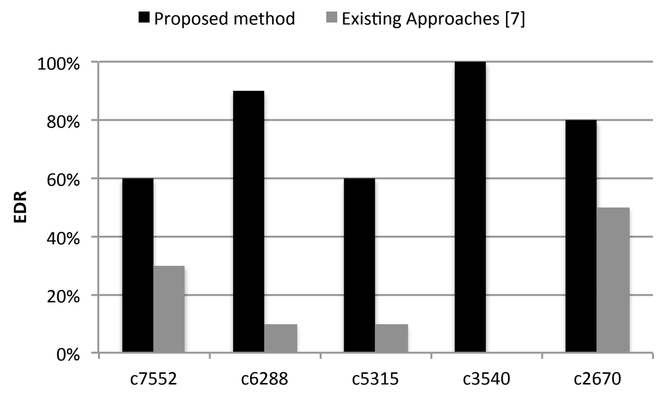


Fig. 13. Comparison of signal selection methods for crosstalk faults

References

[1] M. Abramovici et al., "A reconfigurable design-for-debug infrastructure for socs," in *DAC*, 2006, pp. 7–12.

[2] H. F. Ko et al., "Algorithms for state restoration and trace-signal selection for data acquisition in silicon debug," *IEEE TCAD*, vol. 28, no. 2, pp. 285–297, Feb. 2009.

[3] X. Liu et al., "Trace signal selection for visibility enhancement in post-silicon validation," in *DATE*, 2009, pp. 1338–1343.

[4] O. Caty et al., "Microprocessor silicon debug based on failure propagation tracing," in *ITC 2005*, pp.10pp–293.

[5] S. Prabhakar et al., "Using Non-Trivial Logic Implications for Trace Buffer-based Silicon Debug," in *ATS 2009*, pp. 131–136.

[6] P. Shivakumar et al., "Modeling the Effect of Technology Trends on the Soft Error Rate of Combinational Logic," in *DSN*, 2002, pp. 389–398.

[7] K. Basu et al., "Efficient Trace Signal Selection for Post Silicon Validation and Debug," in *Inter. Conference on VLSI Design*, 2011.

[8] P. Dodd et al., "Basic mechanisms and modeling of single-event upset in digital microelectronics," in *IEEE Trans. on Nuclear Science*, 2002, Vol. 49, No. 6, pp. 3100-3106.

[9] H. Takahashi et al., "A Method for Reducing the Target Fault List of Crosstalk Faults in Synchronous Sequential Circuits," 2005, Vol. 24, No. 2, pp. 252–263.

[10] S. Chun et al.,"ATPG-XP: Test Generation for Maximal Crosstalk-Induced Faults", in *IEEE TCAD*, 2009, Vol. 28, No. 9, pp. 1401–1413.

[11] T.C. May and M.H. Woods, "Alpha-particle-induced soft errors in dynamic memories", in *IEEE Transactions on Electron Devices*, vol. 26, no. 1, 1979, pp. 2 – 9.

[12] S. Mitra, et al., "Robust system design with built-in soft-error resilience", in *IEEE Computer*, 2005, pp. 43 – 52.

[13] A. Sanyal et al., "On Accelerating Soft-Error Detection by Targeted Pattern Generation", *ISQED*, 2007, pp. 723 – 728.

[14] R. Anglada and A. Rubio, "Brief communication. Logic fault model for crosstalk interferences in digital circuits", in *International Journal of Electronics Theoretical and Experimental*, 1989, pp. 423 – 425.

[15] A. Sanyal et al., "Test Pattern Generation for Multiple Aggressor Crosstalk Effects Considering Gate Leakage Loading in Presence of Gate Delays", in *IEEE TVLSI*, vol. 20, no. 3, 2012, pp. 424 – 436.

[16] H. K. Lee and D. S.Ha, "Atalanta: An efficient ATPG for combinational circuits", in *EE Technical Report, Department of Electrical Engineering, Virginia Tech*, 1993.