Scalable Computing: Practice and Experience, ISSN 1895-1767, http://www.scpe.org
© 2020 SCPE. Volume 21, Issue 1, pp. 33–40, DOI 10.12694:/scpe.v21i1.1607

TOPOLOGICAL ORDERING SIGNAL SELECTION TECHNIQUE FOR INTERNET OF THINGS BASED DEVICES USING COMBINATIONAL GATE FOR VISIBILITY ENHANCEMENT

AGALYA RAJENDRAN[*] AND MUTHAIAH RAJAPPA[†]

Abstract. In modern System on Chip (SoC) design consist of intelligence of products, Internet of Things (IoT) based devices, mobile phones, laptops, servers etc. This shrinking market reduces the design automation validation process. Signal selection is the most effective and challenging technique in post-silicon validation and debug. The vital problem prevailing in this method is that it has limited observability and controllability due to the minimum number of storage space in the trace buffer. This tends to select the signals prudently in order to maximize the state reconstruction. To identify the trace signals, signal restoration is the extensive metric that has been used so far. Topology-based restoration method is proposed here to minimize the error detection latency which helps to select the trace signals with minimum error or even errorless. This method aid to detect more number of errors within limited number of clock cyclesthan the restoration only selection techniques.

Key words: Design bugs, Error detection, Internet of Things (IoT), Restorability, Topology, Trace signal, validation

AMS subject classifications. 94A12

1. Introduction.

Recent System-on-Chip (SoC) design studies specifies the extensive usage of SoCs in the electronic industry with increasing complexity in design circuits and its influence to meet the demands in ever growing technologies from various applications containing Internet of Things (IoT). Since there is a growing in the technology it is difficult to meet the functional correctness of the circuits during pre-silicon validation. The escaped errors (both functional and electrical errors) must be captured during posts-silicon validation. Fig. 1.1 represents the key components in the design industry.

The main challenge in post-silicon validation (psv) is that it has limited observability and controllability of internal signals after the first silicon is available [1]. Because of poor observability of internal signals, it is very difficult to debug the design and electrical errors in this atmosphere is laborious [2, 3]. Electronic design automation (EDA) tools are used to automate the product development cycle and validation to achieve the anticipated results with minimum cost and time that ensures the efficiency. Controllability defines the ability of moving an internal signals from input to output. On-chip buffers are used to control these signals.Compare to several methods, scan chain is the most effective technique to enhance both the observability and controllability of internal signal states for the determination of manufacturing. However, it is very less effective during chip execution of debugging and validation [4]. At that time of verification, it need to halt the process to readout the each state of flip-flops.Few signal states are stored in the trace buffer within a limitedno.of clock cycles by incorporating it to the on-chipto attain the non-destructive state of internal signals. Only small set of signals can be traced due to the constraint of area overhead. There is a difficulty in the selection of correct signals since they are selected at the design stage. For smaller deign circuit, it is easy to recommend few signals based on the designers own knowledge but for larger design circuit like automated signal selection, it is difficult. Hence, restoration of signals helps to increase the visibility of untraced signals from the value of traced flip-flops [5]. This is the most common method for finding the signals automatically. From [6, 7, 8, 9], none of the papers has discussed about the effectiveness of detecting or localizing the errors.

[*]Research scholar, School of Computing, SASTRA Deemed University, Thanjavur, Tamilnadu. India. (agalyasastra@gmail.com).

[†]Associate Dean, School of Computing, SASTRA Deemed University, Thanjavur, Tamilnadu. India (sjamuthaiah@core.sastra.edu).

The authors sincerely thank DST-INSPIRE Fellowship and SASTRA Deemed University for providing a great support.

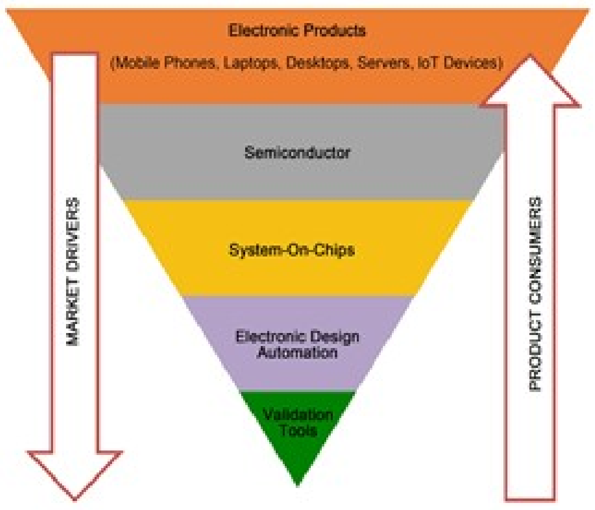

Fig. 1.1. *Interconnected market propellers*

This paper is focusing on the efficacy of bugs and latency that present in the traced signals that may affect observability and controllability enhancements. These effects can be observed at the gate-level selection. An alternative way is proposed at this level of selection that considers a path along the traced gates and their topological sort rather than considering flip-flops only. By this way, it reduces the error detection latency and increases the erroneous responses from the traced gates.

The remaining paper is organized by research background and how this work motivates us are described in Section II. In Section III and IV, the authors proposed the signal section techniques and their findings based on the topological sorting respectively. In section V they have conclude their research work.

2. Research Backgroundand Motivation.

2.1. Background. The state restoration (SRR) can be done either by forward restoration or backward restoration of unknown signal states that are traced during silicon debug [1]. This helps to decide which signal are to be traced in an automatic manner to increase the rate of post- silicon validation part. It can be defined as

$$\text{SRR} = \frac{\text{Number of states restored + traced signals}}{\text{Number of states traced}} \tag{2.1}$$

The visibility of internal signals increases if the state restoration value increases. It may be helpful to find the root cause of internal design bugs when debugging. There is a wide variety of selection techniques were existed and it may cause a serious issue while debugging. One such factor is that assuming all the signals are having equal importance in the selection process and it is really unrealistic from the point of debugging. But in reality, which internal signals are ought tobe observed that varies between design structure and design feature. Existing methods like [10], attempted to link the trace signal selection method with debugging, error detection and localization. But their methods were not appropriate to find the location of random errors. If few flip-flops are traced, it can affect the other flip-flop when the bit flip is detected if they consider different restoration scenarios [12]. However in [11], they critically analyzed about the state restoration but they strongly suggest the usage of assertion coverage to sort out the trace signal problems. This method is restricted to one type of error model and their signal selection methodology is commendable for missing error models.So there is lack in standardizing the error models in psv is the biggest weakness for the evaluation of various signal selection algorithms and deign bugs are categorized from few of reported works [13]. Some bugs were missed out at the gate level abstraction like exchange of wires, inverters, etc. This is the starting point to propose this paper by taking the above said bugs as our motivating example. The bugs in the RTL, translated into gate level manifestations. In our experiment we consider different instances to represent a single design bug.

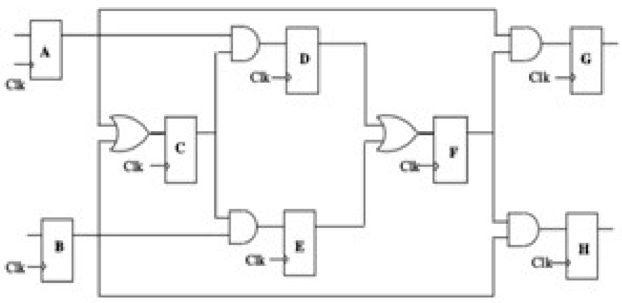

Fig. 2.1. *Example circuit from [4,5]*

2.2. Motivation. To elucidate the irrelevances between the actual error localization and the signal selection algorithms, we simulated the gate-level description of Fig. 2.1 [6]. For example, here we are simulating 2 different types of errors. Primarily, haphazard gate replacement is consider as type 1 bugs and exchange of wires between the circuit is consider as type 2 bugs. As we know that some bugs are not identifiable at the pre-silicon stage under some circumstances. For example, if the design has two input Ex-OR gate that can be replaced by two input OR gate but the only identification of replacement of the gate is '11'. This is the place where we can fails to identify the state at pre-silicon stage when the corresponding state is activated rarely in the state machine. Similarly, type 2 bugs are also justified in this section.

A quick identification of a bug location is observed and its latency report is shown in table 2.1 and 2.2 based on gate replacement before Flip-flop 'C' and exchange of wires between flip-flops 'C' and 'F' respectively. Here, we experiment 4 number of iterations by injecting an error. 'NO' represents either the error doesn't propagate through the flip-flop or cannot be observed (Not observed). In paper [6], they consider AC and AB and in [5, 9] they consider C&F are the combination to achieve high restoration. If SRR helps to find out the erroneous response then these combinations can capture all kind of errors as soon as possible. But in the above table it is evident that the gates which helps to get high SRR value will not observe the errors at all times. If two signals gives high restoration value but if one is traced and the other is restored fully then it will be ineffective for error identification. Because tracing of one correlated signal is ineffective then the restored states were also

Table 2.1
Type 1 bug latency (Gate replacement)

Errors	Flipflops							
	A	B	C	D	E	F	G	H
$Error_1$	NO	NO	NO	1	1	2	3	3
$Error_2$	NO	NO	1	3	2	3	4	4
$Error_3$	NO	NO	1	2	3	3	4	4
$Error_4$	NO	NO	2	3	4	4	1	5
$Error_5$	NO	NO	2	4	3	4	1	5
$Error_6$	NO	NO	NO	NO	NO	NO	NO	NO

Table 2.2
Type 2 bug latency (Exchange of wires)

Errors	Flipflops							
	A	B	C	D	E	F	G	H
$Error_1$	NO	NO	1	2	NO	2	3	4
$Error_2$	NO	NO	1	NO	2	2	3	4
$Error_3$	NO	NO	NO	NO	NO	1	2	2
$Error_4$	NO	NO	1	3	NO	NO	2	NO
$Error_5$	NO	NO	1	NO	3	NO	2	NO
$Error_6$	NO	NO	1	NO	3	NO	NO	2
$Error_7$	NO	NO	1	3	NO	NO	NO	2

Table 2.3
Buggy restored states of Fig. 2.1

Clock cycles/ Flip-flops	A	B	C	D	E	F	G	H
$Cycle_1$	0	0	1	0	0	0	ND	ND
$Cycle_2$	0	0	1	1	0	0	0	0
$Cycle_3$	0	0	1	0	0	0	0	0
$Cycle_4$	0	0	1	0	0	0	0	0
$Cycle_5$	ND	ND	0	0	0	0	0	0
$Cycle_6$	ND	ND	0	0	0	0	0	0
$Cycle_7$	0	0	1	0	0	0	0	0
$Cycle_8$	0	0	1	0	0	0	0	0

follows the same tendency.

As mentioned earlier, restoration is the only criteria that fails to address the debug and validation in an appropriate manner. In [9], some of the restored sates does not illustrate for the erroneous design. Sometimes these are different from the actual state that demonstrates in Fig. 2.1 Type 1 bugs are performed to introduce the design bugs. If the trace buffer width is 2, then C and F are the selected to trace the signal combinations in [5]. Here Table 2.3 shows the buggy design of eight restored states of flip-flops. The highlighted position indicates the conflict value of actual response of design bugs. This is done for 8 cycles, at least one of values of each cycle have the conflict values instead of actual restoration value. This is point we consider to propose this system with the following points since it is undesirable for circuit misbehavior in comparison to actual response. One is the multiple occurrences of false restoration which can affect the root cause of the problem and can increase the debug time. And the second is, the incorrect values may lead to wrong path when try to localizing the error during backtracing.

3. Proposed Method. To improve the signal observability, restoration plays a major role since the untraced signals are restored from the traced signal states. In these approaches, flip-flops are not considered for error propagation whereas it is purely depends on the presence and absence of combinational paths in the circuit between the flip-flops (internal signals). So we adventure this principle in our proposed methodology by allowing circuit topology.

At first, we create an S-graph to examine the error propagation from one gate to the other since the actual bugs are present on the combinational path is the main issue. However the reported work says it is a tedious process hence they have used flip-flops instead of gates present on the path. Therefore the proposed work concentrates on the error propagation through the gates using topological sort is the main concept to reduce the complexity. So that it reduces the time and identifies the errors that propagates through the combinational path. While creating an S-graph, gates are represented as nodes and the flip-flops between the gates are represented as path with directed weighted edges. To optimize the error detection and latency, we are ordering the gates in design structure (Algorithm 1). The traced signals must capture the bugs/errors within limited number of clock cycles, so that this method can ensure the error detection latency is curtailed.

It is not necessary to trace the neighborhood of primary output, it can be done at the primary output with high latency. The vicinity of primary input (PI) can be ignored from a set of traced signals that can observe erroneous behavior through flip-flops. After performing the topological sort (S-graph), the list of gates are pruned until the elements are compared with the width of the trace buffer. The main intension behind the above sorting is to select the gate that can capture higher number of erroneous behavior as early as possible. But this method has major limitation that the topological order can be done only on acyclic graph but the real circuits have more number of cycles. Because of this reasons, group of gates forming cycles into single region and then it becomes a directed acyclic graph. This definitely causes implications on bug propagation in order to bug detection. Hence, this aids to give rudimentary solution incurred by limited topological order. A node in a graph (SG) has definite no.of inputs and outputs. The weights are calculated between the edges of two gates along the path. SG'' indicates the conversion of building SG by grouping the nodes as acyclic graph which forms a cycles into a single node. For larger cycles this conversion of SG'' can break the cycle. To select the particular node we need to find out which node has more priority that gives higher visibility. The outline of the procedure is given in Algorithm 2.

Table 3.1
Algorithm 1

```
1. Create S-graph (Input: Design; Output: Graph (SG))
Total_gates (G)= count the total no.of gates
2. for each gate Gi do
$Gates_{remaining} = Gates_{total} - G_i$
3. for each gate $G_j$ in $G_{remaining}$ do
$node \leftarrow 0$;
4. if path exists between Gj to Gi then
$node \leftarrow node + 1$; end if
end for
5. edge weight =node
6. $node \leftarrow 0$;
7. link $G_i$ and $G_j$ using edge weight
8. end for
9. SG'=directed weight of the design graph
10. SG =Convert SG"
```

Table 3.2
Algorithm 2

```
1. Priority node allocation (Input: Graph (SG); Output: priority list)
2. For each node Ni of graph do
3. $Edge_{input}$ = no.of incoming edges of each node
4. $Edge_{output}$ = no.of outgoing edges of each node
5. $Weight_{input}$ = incoming edge weight
6. $Weight_{output}$ = outgoing edge weight
7. Notch = $Edge_{input}$ ( $\Sigma$ $Weight_{input}$ ) + $Edge_{output}$ ($\Sigma$ $Weight_{output}$)
8. List = priority of each node
9. end for
```

Table 3.3
Algorithm 3

```
1. Find pruning_factor pf ( Input: Graph (SG), Trace_buffer bandwidth (TB),
list of priority node values; Output : Pruning factor)
2. Graph_list = Topological order (SG)
3. Temp_list = Graph_list
4. Pruning_factor= first_pf
5. While Temp_list > TB do
6. Temp_list' = pruned factor of Temp_list
7. Total_priority node values =  Temp_list'
8. Number = no.of Temp_list' entries
9. Avg_value = Tot_priority node values/ number
10. Store pruning_factor and avg_score in a list
11. Pf = pf+pf_increment
12. Temp_list = Temp_list'
13. end while
14. pf = pf of highest avg_score
```

The node values are calculated and it assigns the priority to each gates for the persistence of ranking with regards to error collection. These are calculated using how many number of connections are there from one flip-flop to the other. By multiplying the number of incoming and outgoing edges with their corresponding weights are summed up to known the total number of connections present. After sorting of gates in the topological order these should be cut down from the both the ends of primary input and primary output. The pruning factor can be find by using Algorithm 3.

Here pruning factor (pf) signifies the node which are detached from the list of S-graph. It plays a major role for the selection of trace signal. The proposed fine-grained algorithm find out the optimal solution that begins with complete topological order depends on first_pf and pf_increament. The remaining list is represented as Temp_list', the priority node value calculation has done for the Temp_list' and avg_score that obtained

Table 3.4
Algorithm 4

```
1. Trace_SS (Input: Graph_list, pf, list of priority nodal values,
TB_Bandwidth; Output: FF_list)
2. Node value = value of each node from the priority list
3. Graph_list' = graph_list based on pf
4. FF_list' = sort graph_list by corresponding node value
5. FF_list = FF_list' entries based on TB_Bandwidth
```

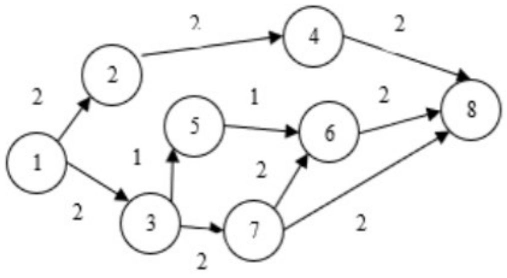

Fig. 3.1. *Directed Acyclic Graph (DAG)*

during calculation. Here pf incrimination is based on the both pf and pf_increment. At last, highest value of pruning factor of avg_score has chosen as pf. These are all based on the window size (ω) of the trace buffer. Based on the percentage of the size of the window, the topological order and the priority list; the arrangements were done and it is shown in Algorithm 4. For example, the pruning factor of 80% and 60% are 0.1 and 0.2 respectively. There is no pruning value for 100%. The topological sorting should be done before pruning factor findings. With the help of both priority assignment and pruning factor, this proposed technique can find more number of bugs within limited clock cycles. To select the signals, the signal selection are done based on the topological order. The remaining nodes of pruning order are arranged using priority node values. Based on the width of the trace buffer the signal selections are done with highest node value.

Based on the above four algorithms (table 3.1-3.4), the erroneous response can be find out easily. Consider a design having 8-combinational gates and its directed acyclic graph is shown in Fig.3.1. This graph contains 8-nodes and its reported results are shown in this section. Based on the priority, the node values are assigned and its corresponding edge values from G1 to G8 are 16, 4, 12, 9, 3, 32, 11 and 14 respectively. These values are arranged in decreasing order to compute the procedure. The pruning factor decides at the previous step is represented as the highest node value after detachment of gates at both the ends. Gate values and its ordering are represented below:

$$Gates(G1 - G8) \rightarrow 1, 2, 3, 4, 5, 6, 7, 8$$
$$Values \rightarrow 16, 4, 12, 9, 3, 32, 11, 14$$

Table 3.5
Topological order (TO) based on priority of node values

Window size (%)	TO	By its priority node value (descending order)
100	1, 3, 7, 2, 4, 5, 6, 8	6, 1, 8, 3, 7, 4, 2, 5
80	1, 3, 7, 2, 4, 5, 60	1, 8, 3, 7, 4, 2, 5
60	3, 7, 2, 4, 5	1, 3, 7, 4, 5

Employing the distributed trace buffer concept rather than using a single trace buffer that helps to debug the modules efficiently. This method can be applied to any larger circuit. For better understanding the authors has taken the ISCAS '89 benchmark circuit to represent it.

4. Experimental Results. To evaluate the trace signal selection, some parameters should be identify. They are: State Restorability (SR), Error Detection Latency (EDL) and Non-Detected Bugs after injecting (NDB). The restoration principle can be applied to any type of signal selection like [5, 6, 7, 8]. However, these techniques were insufficient to increase the visibility of internal signal states than the proposed system. To estimate the weakened state of restoration, the above mentioned parameters are used to derive from the following equation.

$$\triangle\, SR = SR\{SRR\ based\ Trace\ \_SS\}\ -\ SR\{obtained\,Trace\,\_SS\} \tag{4.1}$$

$$\triangle\, EDL = EDL\{SRR\ based\ Trace\ \_SS\}\ -\ EDL\{obtained\,Trace\,\_SS\} \tag{4.2}$$

$$\triangle\, NDB = NDB\{SRR\ based\ Trace\ \_SS\}\ -\ NDB\{obtained\,Trace\,\_SS\} \tag{4.3}$$

To evaluate the diminished states, the above mentioned parameters are used. $\triangle SR$ shows the difference between state restorations for the selected set of trace signals. The difference of error detection latency and the number of errors are reported by $\triangle EDL$ and $\triangle BND$. These three parameters are evaluated more than thousand times on type 1 bug by injecting an error. The signals are selected using either by restoration method or by our proposed topological method. Table 4.1 shows the evaluation of larger benchmark circuits of ISCAS'89 using different parameters as follows. Since the restoration and error restoration approach incompetent, the principle operation of state restoration is applied on the above mentioned parameters.

Table 4.1
Evaluation of proposed system with the bandwidth size of 16

Benchmark t Circuit	No.of FFO	SR	EDL	NDB
s38584	1426	0.906	3.86	152
s38417	1636	-0.263	5.28	491
s35932	1728	-0.01	39.4	223

There is an alternative method to find out the efficacy of trace signal selection using this proposed technique using different bandwidth of trace buffer. These can be calculated using combinational score value by summing the EDL and NDB for different window sizes (ω). The negative sign indicates that this method has higher restorability than existing methods.

$$A\,(wi) = EDL_i/EDL'_i + NDB_i/NDB'_i \tag{4.4}$$

These can be applied to various modules of [6, 9]. To search the highest value the pruning factor should be searched iteratively. It helps to find out the optimal solution from the pruning factor. As mentioned in previous section, it is applicable to distributed modules. If the flip-flop of window size (ω) is traced, then the proposed system can detect the errors in limited cycles. It can be normalized with the help of pruning factor with different window size. The proposed topological sorting method can be applied to automatically generate various learning paths for the mapping analysis concept.

Fig.4.1 shows the error detection ration between the three larger benchmark circuits. The error detection is defined by

$$\text{Error detection ratio} = \frac{\text{No. detected errors}}{\text{No. errors detectable}} \tag{4.5}$$

Consider the circuit is divided into different regions and each region has one error-prone and we have introduced 50 random errors to the active regions and the error density is directly proportional to region size. Here, we executed two kinds of simulation one with an ideal case and the other with the erroneous signals. We have pragmatic our algorithm to various active regions.

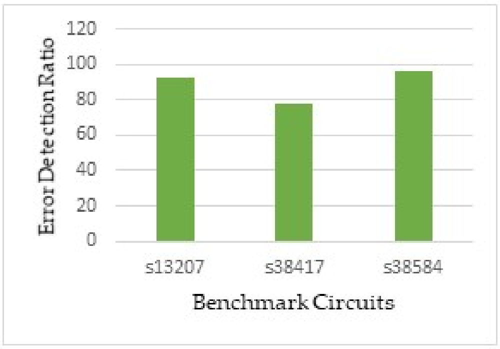

Fig. 4.1. *Comparison of error detection ratio for single active region*

5. Conclusion. Trace buffers are used to store the traced value and to find out the root cause of errors during post-silicon validation and debug. State restoration technique fails to find out the errors present in the traced signals whereas the proposed technique identifies the useful signals that can detect errors. To avoid the problem of false restoration, we can consider the region-separation method. So that we can ensure the selection of signals regardless of topological order.

Acknowledgement. The authors sincerely thank DST-INSPIRE Fellowship and SASTRA Deemed University for providing a great support.

REFERENCES

[1] R. Agalya, and S. Saravanan, *Recent trends on Post-Silicon validation and debug: An overview*, International Conference on Networks & Advances in Computational Technologies (NetACT).,2017.
[2] J. Keshava, N. Hakim, and C. Prudi , *Post-silicon validation challenges: How EDA and Academia can help*, ACM/IEEE Design Automation Conference (DAC), (2010), pp. 3-7.
[3] J. Goodenough, and R. Aitken , *Post-Silicon Is Too Late Avoiding the $50 Million Paperweight Starts with Validated Designs*, ACM/IEEE Design Automation Conference (DAC), (2010), pp. 8-11.
[4] M. Abramovici, P. Bradley, K. Dwarakanath, P. Levin, G. Memmi, and D. Miller, *reconfigurable design-for-debug infrastructure for socs*, 43rd ACM/IEEE Design Automation Conference, (2006), pp. 7–12.
[5] H. F. Ko and N. Nicolici, *Algorithms for state restoration and trace signal selection for data acquisition in silicon debug*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems., 28, (2009), pp. 285–297.
[6] K. Rahmani, P. Mishra, and S. Ray, *Efficient trace signal selection using augmentation and ilp techniques*, Fifteenth International Symposium on Quality Electronic Design, (2014), pp. 148–155.
[7] K. Basu and P. Mishra, *Rats: Restoration-aware trace signal selection for post-silicon validation*, IEEE Transactions on Very Large Scale Integration (VLSI) Systems., 21, (2013), pp. 605–613.
[8] M. Li and A. Davoodi, *A hybrid approach for fast and accurate trace signal selection for post-silicon debug*, Design, Automation Test in Europe Conference Exhibition (DATE), (2013), pp. 485–490.
[9] X. Liu and Q. Xu, *On signal selection for visibility enhancement in trace-based post-silicon validation*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems., 31, (2012), pp. 1263–1274.
[10] K. Basu, P. Mishra, P. Patra, A. Nahir, and A. Adir, *Dynamic selection of trace signals for post-silicon debug*, 14th International Workshop on Microprocessor Test and Verification, (2013), pp. 62–67.
[11] S. Ma, D. Pal, R. Jiang, S. Ray, and S. Vasudevan, *Can't see the forest for the trees: State restoration's limitations in post-silicon trace signal selection*, 2015 IEEE/ACM International Conference on Computer- Aided Design (ICCAD), (2015), pp. 1–8.
[12] A. Vali and N. Nicolici, B*it-flip detection-driven selection of trace signals*, 221th IEEE European Test Symposium (ETS), (2016), pp. 1–6.
[13] D. V. Campenhout, T. Mudge, and J. P. Hayes, *Collection and analysis of microprocessor design errors*, IEEE Design Test of Computers., 17, (2000), pp. 51–60.

Edited by: Swaminathan JN
Received: Sep 24, 2019
Accepted: Oct 30, 2019