 CMES, vol.120, no.1, pp.25-43, 2019

Region-Aware Trace Signal Selection Using Machine Learning Technique for Silicon Validation and Debug

R. Agalya[1], R. Muthaiah[2,*] and D. Muralidharan[3]

Abstract: In today's modern design technology, post-silicon validation is an expensive and composite task. The major challenge involved in this method is that it has limited observability and controllability of internal signals. There will be an issue during execution how to address the useful set of signals and store it in the on-chip trace buffer. The existing approaches are restricted to particular debug set-up where all the components have equivalent prominence at all the time. Practically, the verification engineers will emphasis only on useful functional regions or components. Due to some constraints like clock gating, some of the regions can be ignored during execution. Likewise, some of these regions can be verified deeply and have minimum errors compared to other control regions. The proposed system focusses on random signals that identify more errors which are prone to signal selection technique with low area overhead. To enhance the observability, a machine learning technique is developed. Based on the training samples of smaller designs, a model is developed to find out the contiguous neighbours of each flip-flop. This can eliminate the obstacles of unknown signals. This system demonstrates using Opencores and ISCAS'89 benchmark circuits that result in easy and fast error detection compared to the state-of-the-art of other methods. This is also verified using gate-level error models by cross-validation of each debug run.

Keywords: Controllability, error propagation, machine learning, observability, signal selection.

1 Introduction

As mentioned earlier, the main challenge of verification or validation is to observe and control the internal signal states. To improve those signal states, only a small set of signals [Ko and Nicolici (2009); Liu and Xu (2012)] are traced and stored it in the online trace buffer because of design overhead constraint. These stored data can be helpful for debugging by off-line debug. The problem of limited observability is more challenging so that the restoration ratio technique has evolved [Basu and Mishra (2013); Liu and Vemuri

[1] Junior Research Fellow, School of Computing, SASTRA Deemed University, Thirumalaisamudram, Thanjavur-613401, Tamilnadu, India.

[2] Associate Dean, School of Computing, SASTRA Deemed University, Thirumalaisamudram, Thanjavur-613401, Tamilnadu, India.

[3] Assistant Professor, School of Computing, SASTRA Deemed University, Thirumalaisamudram, Thanjavur-613401, Tamilnadu, India.

[*] Corresponding Author: R. Muthaiah. Email: sjamuthaiah@core.sastra.edu.

(2017)]. Unfortunately, previous approaches are not applicable due to various reasons since each region of the design is equally important for debugging and it selects the signals accordingly [Basu, Mishra, Patra et al. (2013)]. These methods may assume a spatial and temporal uniform distribution for finding the errors. However, in reality, it may not be useful for certain regions from various motives. For example, if certain timeframes using clock gating [Zhang, Li, Shi et al. (2018)] in multicore architecture, there will be no error during that particular frame. Likewise, some regions may have fewer errors associated with intensive regions. Sometimes the logical errors can affect the internal memory rudiments that can propagate towards the output [Ko and Nicolici (2010)]. During debug, only a small number of signals is pertinent for certain time. Hence, a verification engineer likes to trace a diverse set of signals at the diverse time frame. The signal selection can be done dynamically and it was proposed based on error-prone zones [Prabhakar and Hsiao (2009)]. However, this approach is not appropriate for dynamic/random error locations. Therefore, this paper focused on dynamic errors to achieve accurate error findings.

In this article, we recommend an efficient trace signal selection that allied to control the design for accurate trace error detection. This makes two significant contributions. So the authors proposed a regional based random signal selection technique which helps to select a group of lucrative signals based on functional regions associated to error-prone zones with low area overhead based on active regions (A_r). The second thing is, we use the machine learning algorithm that is suitable to enhance the observability based on mock simulation with injected design errors. A contiguous neighbour model is established for the circuit design which is independent to characteristics of a particular design. Henceforth, this method helps to find out the neighbours of each flip-flop (FF) of any random design. The non-traced signals states can be categorised based on contiguous neighbours. The main contribution to this approach is to identify the closely related neighbour of each flip-flop using a machine learning structural features. The debug methodology is developed to pinpoint the errors in the netlist for the smaller region.

The remaining work of this paper is organized as follows. Section 2 describes the related work and background of the research. Section 3 is discussing the methodology used in this paper and it illustrates clearly. The Proposed method and experimental setup were explained in Sections 4&5 respectively. Finally, this paper concludes in Section 6.

2 Background and related works

Limited observability is the major concern while selecting the internal signals of the circuit. It is associated with the problem of data mining. Till now, the existing approaches were tried to solve the problem using design netlist, untraced signal states (state restoration) with the assistance to structural dependencies [Basu and Mishra (2013)].The important aspect is to trace the signal in both forward and backward to justify the untraced signal states. Conversely, the preeminent signals are selected based on design or netlist by the heuristic approach. Selecting the signals based through state restoration is not adequate. The clustering algorithms like contiguous neighbours can help to choose the unknown signals. These are achieved in this fact to maximize internal visibility. The selection techniques like Chatterjee et al. [Chatterjee, McCarter and Bertacco (2011); Prabhakar and Hsiao (2010); Han, Yang and Abraham (2013)] have been proposed which is focused only on state

restoration, not on error detection. The timing errors have been detected recently and these approaches are based on the same set of signals (i.e., static signals) for complete execution of the system. Without considering static signal, an alternate signal method was developed by Mitra et al. [Mitra and Park (2008)] which is very specific to temporal distribution of errors. After that, a multiplexed based signal selection method was developed. However, their approach doesn't consider the heuristic approach of both temporal and spatial distribution of errors. The dynamic method describes how to generate the input sequence and forgets to focus on trace signal selection that results in an error. Later, this dynamic method [Yigit, Zhang, Li et al. (2017)] considers only on state restoration rather than error findings. Finally, it concludes that these approaches are based on test vectors, not on active regions. Later the dynamic method focused on active region but it forgets to concentrate on dynamic errors. Hence, the proposed method focused on the random signal selection that considers a different set of active regions which helps to detect the dynamic errors easily.

Instruction Footprint Recording and Analysis (IFRA) technique was proposed [Rahmani and Mishra (2017)] to detect the errors using on-chip observability mechanism with minimum cost. This mechanism contains thousands of logic gates that results difficulty in debugging at the gate-level. For bug localisation, the Machine Learning Method (MLM) have been used recently [Basu and Mishra (2013); Jindal, Kumar, Jindal et al. (2018)] in both the stages of verification and validation. During testing, a large amount of information can be collected using some learning techniques like regression, clustering, etc. Based on the failure states the clustering technique is applied. Hence, this method helps to extract the bugs from the obtained results. This paper focused on observability expansion of debug data through gate-level error localization technique. These data can be accessed via on-chip trace buffers. The contiguous neighbor helps to make a decision on unknown signal states by increasing the restored signal states and with the traced data. This methodology gives a reasonable solution compared to other methods. The first approach helps to find the neighbors using distance calculation. The next approach helps to find out the feature sets based on the physical structure. These approaches can be extended to localize the errors in electrical nature. Finding the neighbors through the distance calculation and through the machine learning model helps to achieve accuracy. It will be a promising way for further investigation.

3 Methodology & illustration

3.1 Signal restoration vs. error detection

Signal restoration plays a vital role in signal selection for the past few years. Let's contemplate the two-input OR gate, having the inputs of x, y and its output is z. If the output is 1, then any one of the inputs should be 1. Fig. 1 shows an example circuit that comprises 12 number of flip-flops, the signals are traced between input and output. Tracing of flip-flops between A and C in the first cycle, it aids to reinstate the D flip-flop value in the subsequent cycle since the OR gate input is detached. Likewise, I and K are connected through a NOT gate and it can be easily traced in t-1 cycles. From this, we can understand that the tracing can be done in both the forward and backward direction. It is evident that backward restoration is meaningful for error detection. For e.g., When the flip-flop D is having an error,

then it will propagate through forward direction from F to G flip-flops of fan-out cone. To identify these bugs, those flip-flops should be traced in a backward direction. Tracing the fan-in cone of D is not useful to find out those errors. The errors can be detected with the help of restoration property. Concentrating only on restoration performance does not help to improve the internal signal observability and controllability.

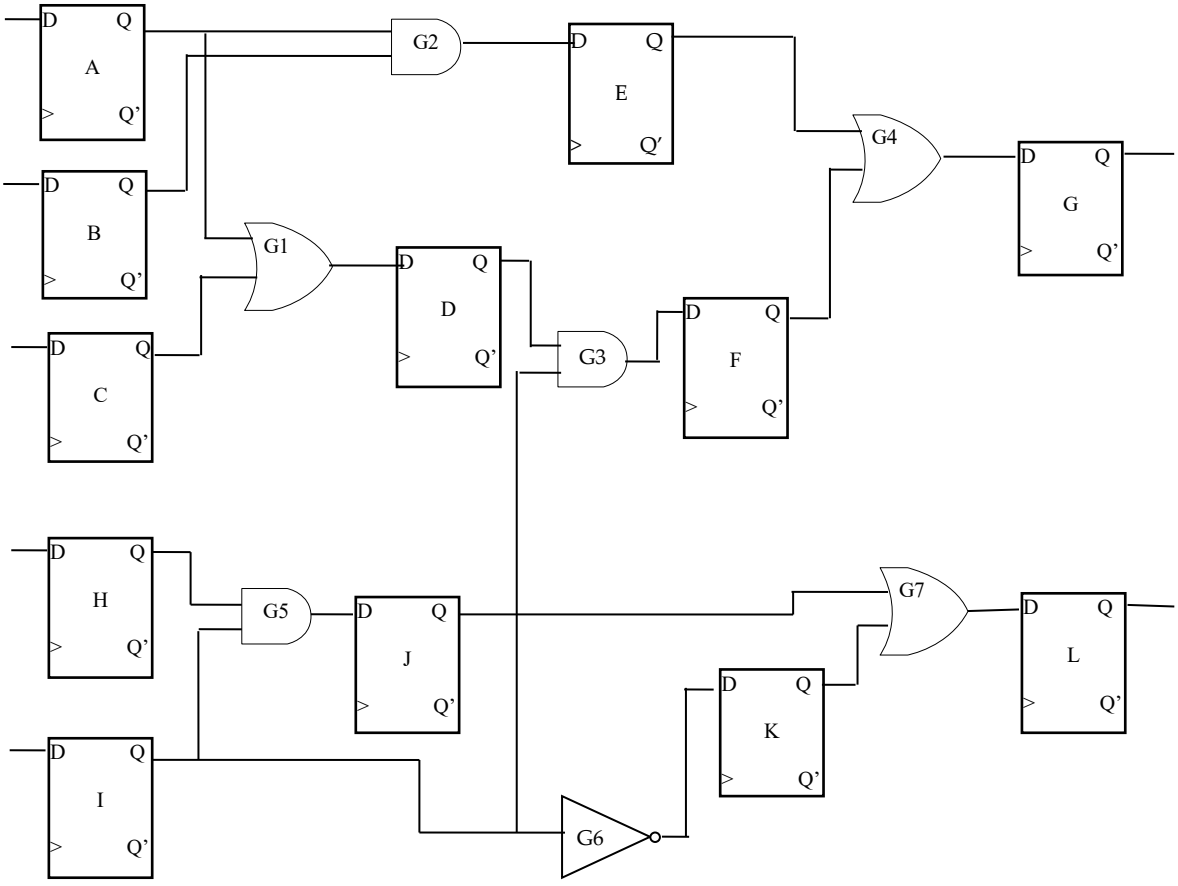

Figure 1: An example circuit

As discussed earlier in this section, observability and controllability are the two main obstacles for error localisation. This is the most stimulating part at the granularity of FF or at the gate-level visibility improvement that delivers very few FFs. There is a large mismatch between the simulation phase and validation phase that results the debug process more difficult. To manage the disparity of signal value, a matrix completion is used based on the machine learning problem. This supports the obtained signal states with the known flip-flop values. If the contiguous neighbour of a particular flip-flop is identified then its corresponding state also can be identified easily. The state may be either 0 or 1 that aids to find all other neighbour flip-flop values. This can find all the signal values in each clock cycles of signal restoration that outcomes full internal signal visibility. The flip-flop states

may not be identified in all the states. Those states are mentioned as X. Because of excessive storage requirements, it is common to assume in the debug stage that not to trace the inputs. The contiguous algorithm decides the contiguous flip-flops based on some features that are related to the particular circuit design. It may be associated with the design structure or simulated values. The netlist is used to perform the simulation of 5 clock cycles to identify its behaviour and state restoration compared to original values.

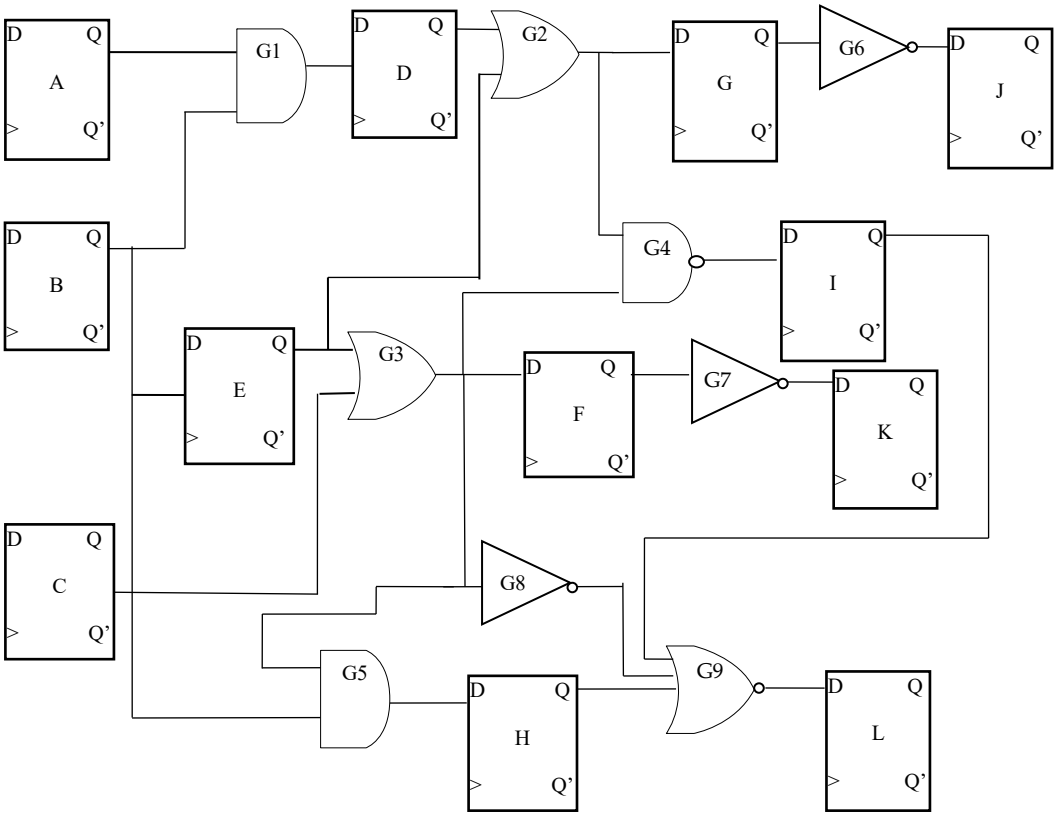

Figure 2: An example circuit using flip-flops at the inputs and outputs

Fig. 2 shows an example circuit of with the flip-flops placed at the inputs and outputs. The state restoration ratio (SRR) can be defined as the ratio of the sum of number of traced states and number of restored states to the number of traced states. Tab. 1 shows the state restoration (SR) that considers flip-flop and it gives the ratio of 2.6 whereas Tab. 2 shows the proposed state restoration that considers gate value gives 3.8 as the restoration ratio. One can think using of flip-flops at the input and output side can be an area overhead. In Fig. 2 and Tab. 2, it is added for better understanding that considers G3=0.

Table 1: SRR of [Jindal, Kumar, Jindal et al. (2018)]

FF/CC	C1	C2	C3	C4	C5
A	0	0	1	0	0
B	x	x	1	x	x
C	x	x	X	x	X
D	0	0	1	0	0
E	x	x	1	x	1
F	x	x	1	1	1

Table 2: SRR of Proposed System

FF/CC	C1	C2	C3	C4	C5
A	**x**	**x**	**X**	**x**	**X**
B	**x**	**0**	**0**	**0**	**0**
C	**0**	**0**	**0**	**0**	**0**
D	*x*	*x*	*0*	*0*	*0*
E	*0*	*0*	*0*	*0*	*0*
F	*x*	*0*	*0*	*0*	*0*
G	*x*	*x*	*x*	*0*	*0*
H	*x*	*x*	*0*	*0*	*0*
I	*x*	*x*	*x*	*1*	*1*
J	**x**	**x**	**X**	**1**	**1**
K	**x**	**x**	**1**	**1**	**1**
L	**x**	**x**	**X**	**x**	**0**

3.1 Problem Formulation

The main aim of this paper is to progress the signal selection process in a random manner to maximize the dynamic error detection in each active region. The error detection can be done by $\sum_{i=0}^{n} D_i$ where i is the number of signals that can be detected using D_i with the trace buffer width 'w'. According to several industrial studies, errors are not equally disseminated in the circuit, whereas they are grouped in several regions. Assume that these regions in the post-silicon validation are similar to the pre-silicon stage. The circuit is divided into several regions (called functional regions) either based on the highest node value or by its components and each region has one or more error zones (E_z).

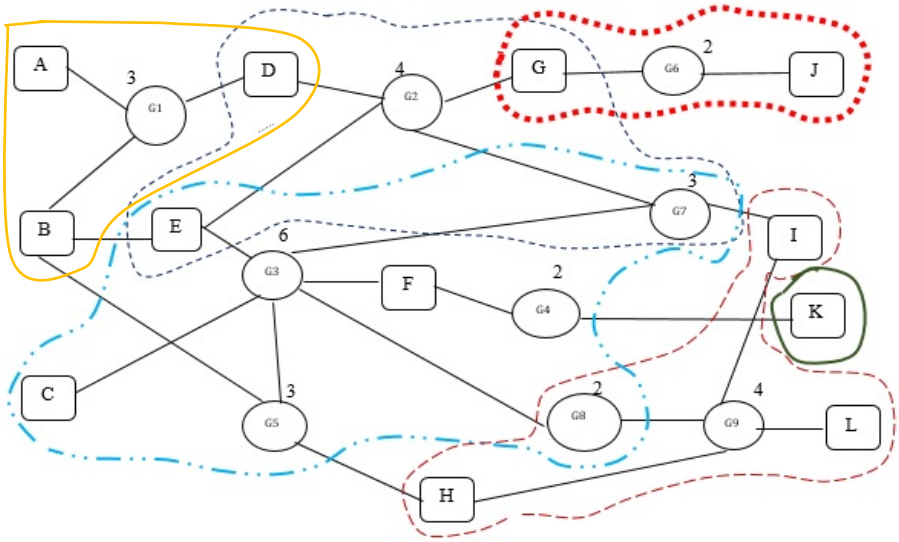

Figure 3: Graphical representation with region separation of Fig. 2

Assume that pre-silicon engineer gives the information. For example, in a multi-core System-on-Chip (SoC) each core can customize the area. If one region has more error zones, then it can be divided into several regions provided some functional boundary and these zones are present inside the circuit. If the fetch and the decode units are in one region then remaining things are in another region. There is a tradeoff between error-prone zones and number of regions. If the circuit is divided like one error zone for a single region then there will be too many regions and thus creates a computational complexity with area overhead. Conversely, a larger region having many error zones can reduce the complexity of random signal selection. Fig. 3 shows the graphical representation of Fig. 2 and it is divided into q regions and each is having one or more error zones and here we consider single error per single region. Contemplate 'n' signal states can be stored in the trace buffer for the width 'w'. Some of the functional regions are being active for any specific cycle due to clock-gating or due to some other cause. If the signal transition does not happen in some regions, then that region is considered as an inactive region. These are the two extremes. If all the regions are active then the useful signals are traced with the help of tracing algorithm and it gives proportional importance to all the regions whereas if one region is active then only that region will be focused to trace the signals. This methodology follows that select the best of 's' signals from q regions from the total set of s×q signals. It should be noted that the error detection in E_{zi} from any of 'q' regions. These signal selection techniques were implemented in efficient hardware for without area overhead. Fig. 3 shows the region separation of Fig. 2 based on highest node value. After separating the regions, we can get the restoration ratio in 3 clock cycles because of parallel computation. Each region may depend on the other region or an independent region like Fig. 4. Each region may have the same or different error zones. These are interconnected to any other regions.

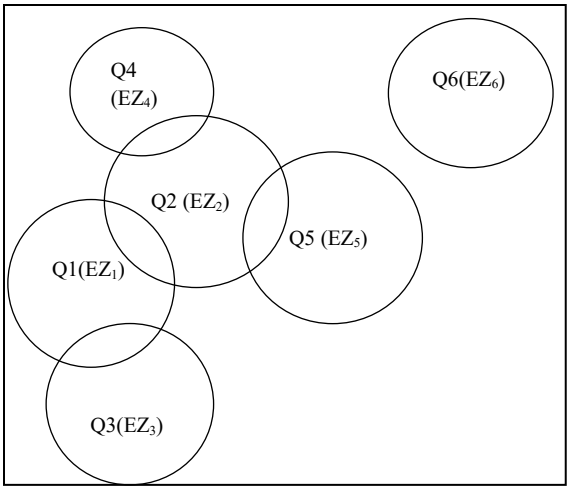

Figure 4: Illustrative of regions with error zones

The contiguous algorithm determines the metric between the points (node). One such metric is Euclidean Distance (ED) that measures the dissimilarity between the two points in Jindal et al. [Jindal, Kumar, Jindal et al. (2018)]. The distance between the two points a and b having the line segment of $\overline{ab}$. The coordinates of a and b are (a_1,a_2,….a_n) and (b_1,b_2….b_n) respectively. The distance between these two points are in Pythagorean format is given by,

$$d(a,b) = d(b,a) = \sqrt{(b_1 - a_1)^2 + (b_2 - a_2)^2 + \cdots + (b_n - a_n)^2}$$
$$= \sqrt{\sum_{j=1}^{j=n} (b_j - a_j)^2}$$

Since ED is a straight line distance calculating method, it is not suitable for finding the length between the two points of the circuit. Because the circuit on the chip may not be connected in a straight line. For finding the adjacent neighbours, the existing method used K-D tree, Ball tree, etc. But these methods perform poorly for the high dimensional matrix (>60) than a brute force linear method. So, the proposed system uses locality sensitive hashing that effectively used for the high dimensional matrix with the Manhattan Distance (MD) for accurate length/distance calculation. This metric gives accurate measures which are having less complexity. For this, we need to pre-process the points in a graph, and the new point is assigned to k-nearest neighbours. To find the contiguous neighbour, start from a random point and move to the neighbour nodes closest to your target. When its distance cannot be improved then that node is your nearest target. Manhattan that finds absolute differences between the coordinates. It is also recognized as taxicab, city block distance and rectilinear distance. It defines the sum of the length of the projections between the coordinates of the line segment. The calculation is given by,

$$distance = \sum_{j=1}^{n} |x_j - y_j|$$

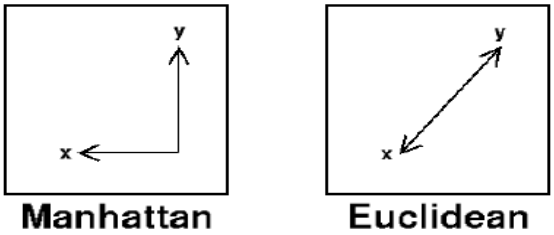

Figure 5: Difference between Manhattan distance & Euclidean distance

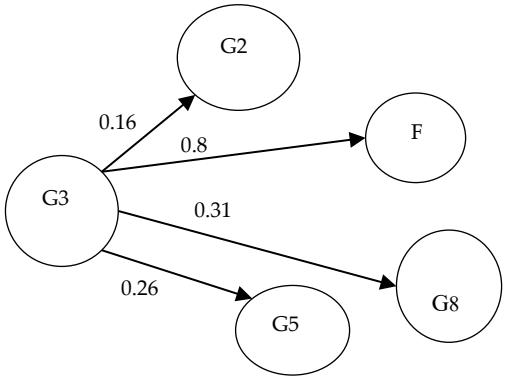

Figure 6: Distance between the neighbours using Manhattan distance

The Manhattan distance between the gates and the flip-flops are identified using a contiguous neighbour algorithm. Fig. 5 shows the basic difference between the two metrics. The distance between the flip-flops and the gates has identified in the forward direction for G3 and it is shown in Fig. 6 using the Manhattan metric. Based on this calculation the neighbours are chosen. This distance helps to determine the unknown values of X. For e.g., if 3-input AND gate is having the inputs of 1,X,X respectively the output cannot be determined. To trace the unknown value of this gate can be identified using the algorithm. If the second input distance is longer than the third input, then the third input value can be identified and taken first. If the value of third input is 0, then the output can be traced as 0. For cases like this, the proposed technique will be useful. If the third input is 1, then the second input should also be identified. In most of the cases, the minimum distance will find the value of unknown, so that more number of signals can be traced within the limited bandwidth. Fig. 6 has 4 neighbours and the signal states are depicted by alpha. But the derived states vary with the neighbours and it is depicted by beta. Tab. 3 shows the expanded internal signal state values that illustrations number of disparate states values are fewer for less neighbours.

Table 3: Expanded Internal Signal States

FF/CC	C_1	C_2	C_3	C_4	C_5
A	0	0	1	0	0
B	0	0	1	0	0
C	0	$1(0^{\alpha\beta})$	1	0	$1(0^{\alpha\beta})$
D	0	0	1	0	0
E	0	$1(0^{\alpha\beta})$	1	0	1
F	0	$1(0^{\alpha\beta})$	1	1	1

The signal states of the neighbours of gates and flip-flops will be updated dynamically. However, the distance will justify the choice of the neighbours. It is difficult to vitrine all the merits using this simple example here since most of the signal states are restored through state restoration. We found around 60% of internal signals are unknown in larger circuits. This indicates the necessity of improving the observability even after state restoration.

4 Proposed methodology

The first step is to build a graphical representation of a circuit (Fig. 2) to find the error propagation probability from each node to the other node in the same region or from a different region. The profitable set of signals are selected from each q region using Algorithm 1. The edge between the nodes is regardless of the type of node. G3 is the node that probable sources of error propagation in the forward direction. Any error that propagates through this node in region 1 which are regardless of fan-out cones. Consequently, error from gate 3 propagates through all the gates and the flip-flops towards the output. The probability-based node value calculation can be done based on dependent and independent paths of each node. However, based on the number of input and output connections the highest node value has found and it also gives the same restoration value like Pedregosa et al. [Pedregosa, Weiss and Brucher (2011)]. The regions are separated with the help of highest node value. The only difference is that the computation time is less when we select the highest node value selection than the probability approach. Hence, the proposed method achieves less time compared to the existing method in this step. The signal selection can be done using the following algorithm.

Algorithm 1: Region aware signal selection

In: Circuit, TBW, E_z; **Out:** list of s selected signals (s1…si)

1. Generate a graphical illustration of a circuit, then split it into q regions having an error zone Ez
2. Initialize Si=null
3. Calculate the probability of an error for each node s that propagates to Qi
4. While the region Qi is empty and signal Si is not having i number of signals do
5. Select the highest node value j and add this to the list $S_i=S_i$ U j
6. Remove the node j from the list after computation to prevent overlap from Qi
7. for each node in Qi, calculate the error propagation probability of Ezi and then sum it
8. end
9. return the list with selected signals

4.1 Error propagation computation

It defines the probability of an error that propagated from input to output of a circuit/ gate. For multiple gates, it is essential to describe the dependent and independent paths. This paths are well-defined by the sequence of logic gates from the source to destination. These probability of an error occurs in the single gate of an AND and an OR gates are $P_{i1}^{1} = \pi_{2\leq k\leq n} P_{ik}^{1}$ and $P_{i1}^{0} = \pi_{2\leq k\leq n} P_{ik}^{0}$ respectively. In detail, for any error (0/1 or 1/0) that propagates towards an output, then all the remaining inputs of AND gate must be 1. For example: if the output of AND gate is 1, then all the inputs of AND gate must be 1, if the output is 0 then any one of the input must be 0 and it is irrespective of i_1. So the error may be undetected and we can assume that all the remaining inputs of AND gate are independent. The probability of all the remaining inputs is represented in the above equation that the probability of an error i1 will propagate to the output o1. Similarly, the calculation of NAND gate can be done. In place of an OR gate, the tactics is same but the state may be held at 0 and the probability of getting an error i1 with respect to 0 is mentioned in the above equation. Similar computations for NOR gate. These computations are only for an individual gate. For multiple gates, we need to consider both dependent and independent paths.

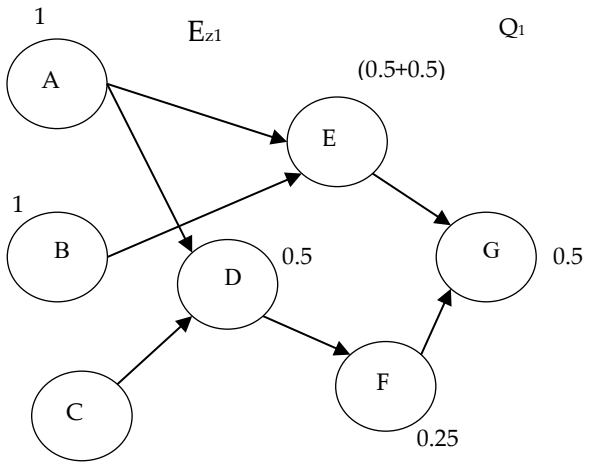

Figure 7: Probability node values for predicting errors in each region

4.1.1 Multiple gates by dependent path

In Fig. 7, from a source to destination, the internal signal probability is 50% since the state is 0 or 1. This is calculated for Fig. 1 with region Q1 for better understanding. For example, there is a single path from A to F and the probability of getting an error through D is 0.25 since there is an OR gate between A and D, AND gate between D and F. The probability of getting an error between these paths are 0.5 and 0.5 respectively. By multiplying these paths, the overall probability of getting an error is 0.25. So, the overall error propagation is $P_{(s,d)} = \pi_{1\leq k\leq n} P_k$

4.1.2 Multiple gates by independent path

Like the above-said statement, there are two paths between A and G. From those, one such path is through A,E,G and A,D,F,G. The computation is similar to the previous one. The overall probability of getting an error in the path A,E,G is 0.25 and for A,D,F,G is 0.125. The maximum probability is always dominated the detection of an error. So it can be calculated as $P_{(A,G)} = maximum\ (P_{(A,E,G)}, P_{(A,D,F,G)})$. In general, the error propagation probability is demarcated as $P_{(s,d)} = maximum\ (P_{(e1,e2\ldots\ldots en)})$. From this, the proposed system finalizes some of the features that can calculate the error propagation score of each gate. Because the training model must be generic and it is applied to Design Under Test (DUT) to get the best results. For modeling technique ML, the following features are considered.

1. ε_1(fan-in): number of gates that connected to its input.
2. ε_2(fan-out): number of gates that connected to its output.
3. ε_3(connectivity): number of flip-flops and combinational gates connectivity in both forward and backward direction.
4. ε_4(input distance): minimum distance between the primary input and flip-flop/gate.
5. ε_5(output distance): minimum distance between the flip-flop (gate) and primary output.
6. ε_6(ed score): $\sum_{p=0}^{p=P}(\pi_{g=0}^{g=G}(score\ of\ each\ gate))$.
7. ε_7(signal restoration): flip-flop or gates is defined as the number of restored states over the process of t cycles of trace signals.

It should be noted that the features mentioned above are fully based on the physical structure of the circuit and how fast it can calculate. Using this structural relationship, the contiguous neighbours are obtained. Observability expansion can be made either by structural analysis or by error signatures obtained from the netlist. Based on these two the neighbour states of all the flip-flops and gates are identified. The main criteria is to find the optimum neighbour (ON) to figure out the internal signal states for better visibility. To iterate this, the minimum and the maximum number of neighbours have taken as 5 and 500 respectively. To compute these, we have implemented K-D tree, Ball tree using locality sensitive hashing to find the neighbours in the fastest manner. Note that flip-flops are derived from error blocks obtained from the debug procedure. We define a localization function to quantify the efficacy of gate-level error localization with the proposed debug methodology. If the error injection and subsequent localization experiment is carried out *K* times, the function has the maximum value of *K*. We report for the two training methods (t*raining*_1 and t*raining*_2) and various error injection scenarios.

By normalising the parameter values, the variants are denoted by Vi. By training the smallest circuits that help to identify the combination of neighbours (ϕ_j). The depth of the trace buffer is denoted by Cy cycles, the number of iteration is denoted by k and the proposed learning model is ML. As mentioned earlier, the training can be performed by two ways. The algorithm is given by,

Algorithm 2:

1. **Procedure***training_1* (input: CG_e, k; output: ML)
2. ML←∅
3. T←traced and restored stated for each cycle
4. Error signature of each gate (CGe)← N- states of CGe for Cy cycles
5. notch value (ϕ_j , Vi) ← 0
6. **for** each Vi **do**
7. **for** h= ON (ϕ_j) **do**
8. **for** y=1 to K **do**
9. neighbors←ONmodel(CGe, Vi)
10. all standards ←fill values((neighbors,T)
11. notch value ← all standards – CGe
12. **end for**
13. notch value (ϕ_j , Vi)←$\sum(notch)$
14. **end for**
15. **end for**
16. neighbor score = min (notch value (ϕ_j , Vi))
17. M ← ON (ϕ_j , Vi)
18. **end procedure**

To find out the nearest neighbour is totally depends on the error signature (CGe) and its learning model (Vi). After finding the neighbour of a particular node, the traced and restored values are identified for observability expansion. The minimum difference between the expanded signature and CGe gives the best companion. The Neighbor score gives the summation of actual score and observed score of ϕ_j and Vi.

The second model is fully based on the netlist and independent error signatures. Based on the features as mentioned above, the estimation of error transmission via each input gate can be specified as ($1/N_{inputs}$). Once the gate value score is calculated, the error probability score can find out easily by calculating all the input paths to each flip-flop. Here, the number of gates are G and the number of paths are P to estimate the training model. The second training model is developed based on the number of features listed without restoring the error signatures from the simulation. The training model is given by,

Algorithm 3:

1. **Procedure***training_2* (input: netlist, n, k; output: ML)
2. ML←∅
3. feature_set←∅ and
4. T←traced and restored stated for each cycle
5. Error signature of each gate (CGe)← N- states of CGe for Cy cycles
6. notch value (ϕ_j , Vi) ← 0
7. **for** each CG 1 to x**do**
8. compute all the features
9. feature_set← feature of each CG
10. **end for**

```
for each Vi do
  for h= ON (ϕj) do
    for y=1 to K do
      neighbors←ONmodel(CGe, Vi)
      all standards ←fill values((neighbors,T)
      notch value ← all standards – CGe
    end for
    notch value (ϕj , Vi)←∑(notch)
  end for
 end for
neighbors ←ON model (feature, space, Vi)
ML ← ON (ϕj , Vi)
end procedure
```

Hence, the resultant model will be M (*training_1*, *training_2*) to find the neighbours that reduce the requirement and memory space. The missing values are identified with the help of neighbours. This helps to increase the observability that debugs efficiently. This is verified by a higher level of abstraction that localises the error exactly. The error localisation is very difficult at the gate level, so the proposed method tries to achieve at the granularity of each gates/flip-flop. The basic principle to debug is that the difference between the golden reference and obtained signatures. So the problem of unknown signal values has been resolved using the random signal selection algorithm. From this, we can understand that for localising the smaller circuit, the error signatures are divided into a certain number of blocks. Based on the dissimilarity, the error signatures are localized. It can be done by ranking of all the flip-flops or gates out of which the top most thing should be chosen.

4.2 Random signal selection

Algorithm 4: In: Circuit, tbw, A_r, related region a_r, selected signal list; Out: List of n traced signals

1. Signal to be traced (ST)←∅
2. Let ar = $\sum_{i=1}^{i=m} a_{ri}$
3. Find the influence from Ar, Cr = $\frac{n\ x\ ari}{ar}$
4. Select Cr from selected list
5. Place the selected list in ST and then repeat Step 4 and 5
6. Return selected ST

During the active region, the duration depends on (1≤k≤m) for m x n1 signals. The total number of possible states is 2^{m-1}.For example, the active region of size m=2 and n1=2, the two selected signals from each region (R_A, R_B) is represented as A_0, A_1 and B_0, B_1 respectively. If the current state is (0,1) then the selected signals will be (B_0, B_1) of Fig. 8 that uses 'x' multiplexers in Random Signal Tracing (RST). Likewise, if it is (1,0) and (1,1) then the signal states are (A_0, A_1) and (A_0, B_0) respectively. The output is given to the trace

buffer and the controller that controls the signals towards multiplexer that operates under the same clock. An external knob is given to the active error zone information of a circuit for validation.

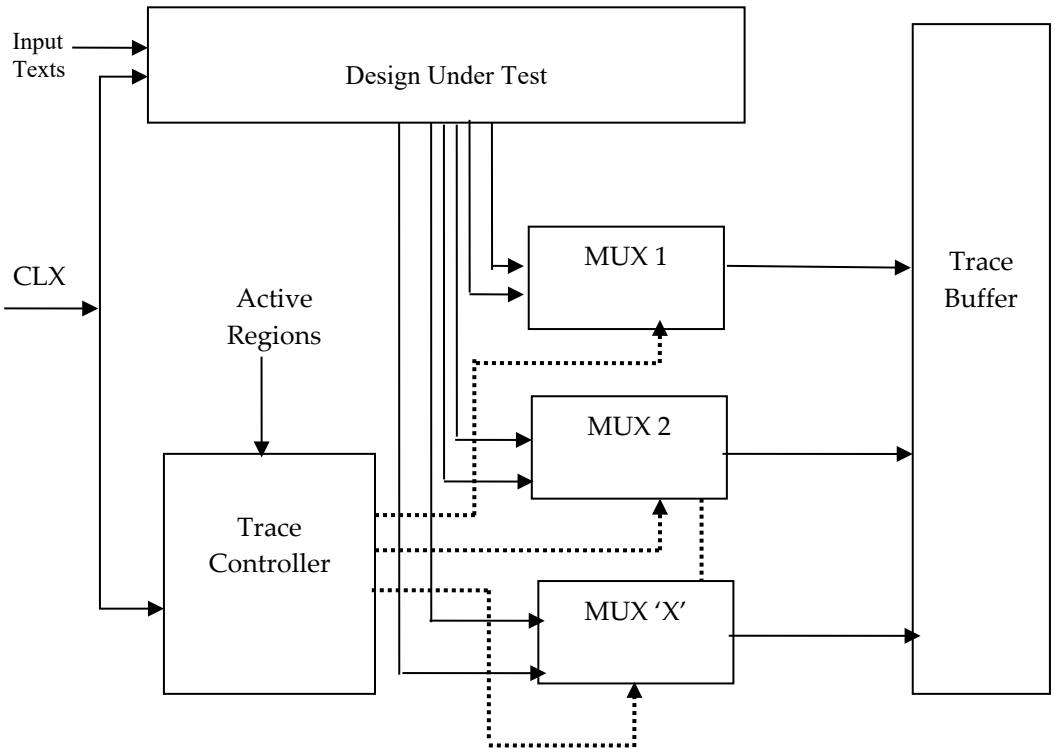

Figure 8: Block diagram for Random signal Tracing

Note that, modeling 1 is not been probable for larger extent. So the design bugs are mapped into netlist level [Choi, Jung, Oh et al. (2018)] such as unintended wire exchange, gate level random gate replacement, etc. If the two wires are exchanged due to the design error and it slips to fabricated silicon, then that error will propagate to the logical cone of a connected path. If there is an unintended inverter is present in the design then this also can propagate towards the output cone.

5 Experimental setup

We have verified the region-aware trace signal selection and random signal tracing algorithms with the help of ISCAS'89 benchmark and Opencores circuits. Consider 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. The error detection is defined by,

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

We have pragmatic our algorithm to various regions and active regions. We have obtained the results for two regions to different scenarios. In global signal selection, they assumed that the errors are uniformly distributed without considering active regions and error zones [Jindal, Kumar, Jindal et al. (2018); Kumar, Jindal, Fujita et al. (2017)] whereas the proposed technique considers error detection and restoration. After this, error-one technique has evolved and this approach was focused only on static region that considers all the regions are active. But the proposed technique focused on the active region, error zones, different or random region that can reduce the time and memory space. The comparison for error detection ratio for a single region is active is shown in Fig. 9.

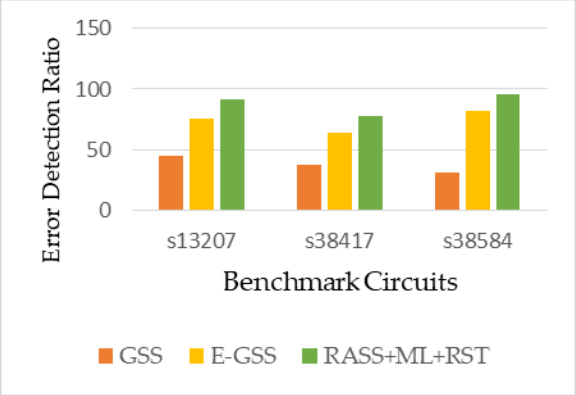

Figure 9: Comparison of error detection ratio for single active region

GSS-Gate-level Signal Selection
E-GSS-Error zone based Gate-level Signal Selection
RASS+ML+RST-Region-aware Machine Learning Random Signal Tracing

A Verilog module is developed that parameterised m regions and n1 traced signals. Then the proposed module is synthesized using Synopys design complier using lsi_10k technology. A typical size of trace buffer size is 32×1024 bits. Usually, the debug technology occupies more space than a controller, whereas the proposed technique uses only 5360 μ while using the same library. Therefore, this approach detects two times more errors compared to existing methods. With the help of Scikit-learn [Pedregosa, Weiss and Brucher (2011)] library, the above-mentioned the proposed learning model implements benchmark circuits. The training_1 model has used single design error (e.g., exchange of wire). For the same configuration of trace buffer size (32×1024 bits), the neighbours (ϕj=10) are identified using the learning model (Vi). Using the training model 1, the observability expansion is achieved. When the number of neighbours are less or if it is larger, the accuracy falls. So the neighbours should not be less than (like 5 or 10) or greater than 500, this condition occurs. Then the accuracy range lies between 8% and 15% in Fig. 10. Using the training_2 algorithm, the accuracy increases by more than 25% for 50 neighbours when choosing the combinational gate rather than the flip-flop with the help of training_2 model the accuracy increases. This simply justifies the simulation based observability technique works better than the usual reconstruction method. However, the rate

of accuracy (increases/decreases) is lower compared to training_1. After analyzing the model, the proposed method hooks the least but it catches the most important aspirants.

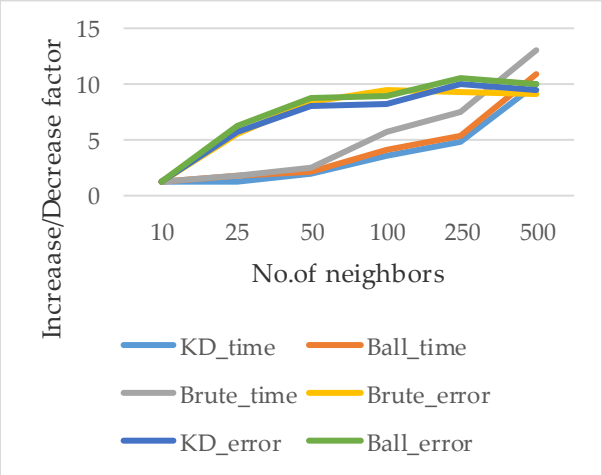

Figure 10: Training_1 observability expansion

Using the training model, the error localisation values obtained and it is shown in Tab. 4. There is a smaller variation in obtaining the error localisation between the two training models. Correspondingly, there is a smaller variation when the number of neighbours and the learning model that is used for contiguous neighbour mode M.

Table 4: Results of Two Training Models with the Number of Error Localization

Benchmark Circuit	***Training_1* Error injection**	***Training_2* Error injection**
s35932	41	39
s38417	45	42
s38584	48	46

The methodology is verified with the help of cross-validation of the debug data on designs injected with gate-level error models. To avoid over-fitting in our model, we have used 5-fold cross-validation technique (20%-test/validation vectors; 80%-training vectors). The problem of over-fitting occurs when the model is too specific to the training data that results in low accuracy of the new data and high accuracy in training data. Here we have used the smallest set of ISCAS'89 benchmark circuits to train our signal selection model and for running the modeling techniques, 5-fold cross validation have been used. This system is validated by comparing the obtained results with the Icarus Verilog simulator output.

6 Conclusion

Limited observability and controllability is the basic problem of post-silicon validation and debug. The rudimentary hypothesis of existing methods was developed using uniform errors that distributed over the circuit. The proposed technique selects the most beneficial

set of signals in each divided region based on error zones. The regions which are active are traced dynamically at a certain time with the help of contiguous neighbours. This methodology uses Manhattan distance for the distance calculation from simulated values for neighbour findings. Finding of neighbours using machine learning model helps to detect two times more errors than the existing methods.

Acknowledgement: The authors sincerely thank DST INSPIRE Fellowship (IF150623) and SASTRA Deemed University for providing a great support.

References

Basu, K.; Mishra, P. (2013): RATS: Restoration-aware trace signal selection for post-silicon validation. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, pp. 605-613.

Basu, K.; Mishra, P.; Patra, P.; Nahir, A.; Adir, A. (2013): Dynamic selection of trace signals for post-silicon debug. *14th International Workshop on Microprocessor Test and Verification*, pp. 62-67.

Chatterjee, D.; McCarter, C.; Bertacco, V. (2011): Simulation-based signal selection for state restoration in silicon debug. *IEEE/ACM International Conference on Computer-Aided Design, Digest of Technical Papers*, pp. 595-601.

Choi, I.; Jung, W.; Oh, H.; Kang, S. (2018): A debug scheme to improve the error identification in post-silicon validation. *PLoS One*, pp. 1-15.

Han, K.; Yang, J. S.; Abraham, J. A. (2013): Dynamic trace signal selection for post-silicon validation. *Proceedings of the IEEE International Conference on VLSI Design*, pp. 302-307.

Jindal, A.; Kumar, B.; Jindal, N.; Fujita, M.; Singh, V. (2018): Silicon debug with maximally expanded internal observability using nearest neighbor algorithm. *IEEE Computer Society Annual Symposium on VLSI*, pp. 46-51.

Ko, H. F.; Nicolici, N. (2010): Automated trace signals selection using the RTL descriptions. *Proceedings-International Test Conference*, pp. 1-10.

Ko, H. F.; Nicolici, N. (2009): 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*, vol. 28, pp. 285-297.

Kumar, B.; Jindal, A.; Fujita, M.; Singh, V. (2017): Combining restorability and error detection ability for effective trace signal selection. *Proceedings of the on Great Lakes Symposium on VLSI-GLSVLSI'17*, pp. 191-196.

Liu, X.; Xu, Q. (2012): On signal selection for visibility enhancement in trace-based post-silicon validation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, pp. 1263-1274.

Liu, X.; Vemuri, R. (2017): Effective signal restoration in post-silicon validation. *IEEE 35th International Conference on Computer Design*, pp. 169-176.

Mitra, S.; Park, S. B. (2008): IFRA: instruction footprint recording and analysis for post-silicon bug localization. *DAC*, pp. 373-378.

Pedregosa, F.; Weiss, R.; Brucher, M. (2011): Scikit-learn: machine learning in python. *Journal of Machine Learning Research*, vol. 12, pp. 2825-2830.

Prabhakar, S.; Hsiao, M. (2009): Using non-trivial logic implications for trace buffer-based silicon debug. *Proceedings of the Asian Test Symposium*, pp. 131-136.

Prabhakar, S.; Hsiao, M. S. (2010): Multiplexed trace signal selection using non-trivial implication-based correlation. *Proceedings of the 11th International Symposium on Quality Electronic Design*, pp. 697-704.

Rahmani, K.; Mishra, P. (2017): Feature-based signal selection for post-silicon debug using machine learning. *IEEE Transactions on Emerging Topics in Computing*, pp. 1-9.

Yigit, B.; Zhang, G. L.; Li, B.; Shi, Y.; Schlichtmann, U. (2017): Application of machine learning methods in post-silicon yield improvement. *International System on Chip Conference*, pp. 243-248.

Zhang, G. L.; Li, B.; Shi, Y.; Hu, J.; Schlichtmann, U. (2018): EffiTest2: efficient delay test and prediction for post-silicon clock skew configuration under process variations. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 70, pp. 1-14.