Optimizing Dynamic Trace Signal Selection Using Machine Learning and Linear Programming

Charlie Shucheng Zhu, Sharad Malik
Princeton University, Princeton, NJ 08544 USA

***Abstract*—The success of post-silicon validation is limited by the low observability of the signals on the chip under debug. Trace buffers are used to enhance visibility of a subset of the internal signals during the chip's operation. These trace signals can be selected statically, i.e. the same trace signals are used through an entire debugging run, or dynamically where a different set of signals can be used in different parts of a debugging run. The focus of this work is on dynamic trace signal selection. Our technique uses machine learning for classification of different groups of inputs that are likely to trigger different faults, and a linear programming based optimization method for selecting the different sets of trace signals for different combinations of inputs and states. In contrast to existing methods, this technique is applicable to both transient and permanent faults.**

I. Introduction

The goal of ***Post-Silicon Validation*** is to find bugs in manufactured prototype chips. In post-silicon validation, the debugging run can execute long tests at speed. However, the limited observability in the post-silicon setting hinders the debugging process. Improving the observability of a silicon chip is critical for the effectiveness of silicon debug [5].

To address the observability issue, trace buffers are used to obtain the circuit's internal status for a number (typically thousands) of cycles without interrupting the chip's test run [1]. However, due to the limited space for trace buffers, only a small portion of the circuit's internal register values can be recorded by trace buffers. Therefore, it is important to select good trace signals to facilitate the debugging process.

There are two main types of trace signal selection techniques: ***static*** and ***dynamic trace signal selection***. Due to the limited size of trace buffers, only a limited number of signals can be traced at a time. Static trace signal selection uses the same set of traced signals through the entire debugging run. In contrast, dynamic trace signal selection allows different groups of trace signals, which are called ***trace groups***, to be chosen at different cycles in a single run. Because the trace groups are usually selected through multiplexers, dynamic trace signal selection is also referred to as ***multiplexed trace signal selection*** [2]. Due to its flexibility in tracing more signals in different time frames, dynamic trace signal selection outperforms static trace signal selection in many scenarios. The focus of this work is on dynamic trace signal selection.

A ***permanent fault*** is one that occurs in all cycles during circuit operation. Such a fault is considered detected if it is detected in any cycle during the execution. A ***transient fault***, in contrast, is only activated in some specific cycles. Capturing such a fault during the test run is more challenging. Our goal in this work is to *optimize transient fault coverage, while maintaining high permanent fault coverage*.

Our approach makes the following contributions:

(i) We propose a novel dynamic trace signal selection approach based on Machine Learning and Integer Linear Programming (ILP). Higher coverage is achieved by using tailored trace signals for different inputs and state values during a debug run. Our approach can capture 5% more permanent faults and 18% more transient faults, compared with an existing competing dynamic approach. (ii) In our dynamic approach, the number of groups is adjustable to provide debug flexibility. (iii) Our algorithm works on the gate level design. No extra high level insight, e.g., the error zones and activation regions as in [2], is needed. (iv) Our approach is scalable and is able to analyze circuits with tens of thousands of gates.

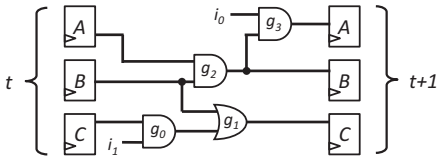


Fig. 1: Example circuit under debug

II. Preliminaries

In a ***sequential circuit***, the outputs depend on both the input signals and the state in the registers $r_1, \ldots, r_l$, whose outputs and inputs are connected via a combinational circuit C. The values of all nodes in C at a particular cycle t is determined by the primary inputs and register values at t. We define these values as the ***in-state*** v_t. Similarly, the ***out-state*** consists of the output signals and the register values when cycle t ends.

A. Permanent and Transient Fault Coverage

A fault in the sequential circuit is modeled as a bit-flip (negation of a signal) on a node for one or multiple cycles during its execution. For a ***permanent fault***, the bit-flip may occur in many cycles on a particular node. For a ***transient fault***, the bit-flip may only occur once during execution. A fault ***triggered*** by an in-state v_t is a bit-flip which can propagate to the out-states given the values of v_t. Consider a set of in-states V, a fault is triggered by V if any in-state in V can trigger this fault. A ***triggered region*** of v_t (or V) is the set of nodes on which faults can be triggered by v_t (or V respectively). They are annotated as $tr(v_t)$ (or $tr(V)$). In Figure 1, given an in-state $v_0 = \{i_0 \mapsto 0, i_1 \mapsto 0, A \mapsto 0, B \mapsto 0, C \mapsto 0\}$, the triggered region is $tr(v_0) = \{B, g_0, g_1, g_2, g_3\}$.

Given a particular in-state v_t, we denote the property of fault propagation by a function $P(f, r, v_t)$. It is set to be 1 if a fault on node f can propagate to a register r under v_t. Otherwise it is 0. A fault on f is ***covered*** by the selected trace signals under v_t if the fault can propagate to any of the selected trace signals under v_t . This property is described as $C(f, v_t) = \bigvee_{r \in selected} P(f, r, v_t)$, where $selected$ is the set of selected trace signals.

Suppose V is a set of in-states. We define $P(f, r, V) = \bigvee_{v_t \in V} P(f, r, v_t)$, that is, $P(f, r, V)$ is 1 if a fault on f can propagate to a register r under any in-state of V. Similarly, a fault f is ***covered*** by the trace signals under V if the fault can propagate to any selected trace signals under any in-state of V. This property is described as $C(f, V) = \bigvee_{r \in selected} P(f, r, V)$, or equivalently, $C(f, V) = \bigvee_{v_i \in V} C(f, v_i)$. Note that a fault that is covered under an in-state must be triggered by the in-state. However, a triggered fault is not necessarily covered. Given the selected (either statically or dynamically) trace signals and a set V with $|V|$ in-states, there are two types of metrics to consider:

(i) Suppose V is the set of all possible in-states during debugging. A permanent fault f is covered if $C(f, V) = 1$. The ***permanent fault coverage*** is the ratio of covered permanent

faults under V divided by all possible permanent faults:

$$pfc = \frac{\sum_{f \in N} C(f, V)}{|N|} \quad (1)$$

(ii) To ensure that a transient fault on node f at cycle t is detected, the corresponding $C(f, v_t)$ must be 1. Since there are $|N|$ possible locations for f and $|V|$ possible in-states, all possible transient faults are covered only when $\sum_{v_t \in V} \sum_{f \in N} C(f, v_t) = |V| \times |N|$, The ***transient fault coverage*** is defined as the the ratio of covered transient faults under every in-state v_t divided by all possible transient faults:

$$tfc = \frac{\sum_{v_t \in V} \sum_{f \in N} C(f, v_t)}{|V| \times |N|} \quad (2)$$

Our goal is to optimize the transient fault coverage, while maintaining a high permanent fault coverage, in other words, maximizing (2) and maintaining high value for (1).

B. Supervised and Unsupervised Machine Learning

Machine learning is extensively used in many applications in computer science. One typical task of machine learning is to predict the ***labels*** of samples by building a statistical model from the ***features*** of the samples. Depending on whether training instances with pre-determined labels are available, machine learning algorithms can be categorized as ***supervised learning (classification)*** or ***unsupervised learning (clustering)***.

Recently, machine learning techniques have been explored in hardware verification [4]. Trace signal selection also fits well as a machine learning application in the sense that there is a large amount of data to be classified for a circuit with many in-states. In our novel approach, an unsupervised learning algorithm is used to cluster those in-states into different groups (as labels) by different triggered regions (as features). Those grouped in-states (as labeled features) are further utilized as the training data set for another supervised learning algorithm to predict the group label of a new in-state during a debug run. In our experiment the ***K-means clustering*** technique is used for the unsupervised learning, while ***decision tree learning*** is applied as the supervised learning algorithm for the ease of implementing the multiplexer tree in hardware for trace signal selection.

III. Dynamic Approach to Optimize Permanent and Transient Fault Coverage

As proposed in [8], ***error transmission matrices*** (***ETM***) represent how faults propagate under different in-states.

A. Optimizing Permanent & Transient Fault Coverage Using Integer Linear Programming

***Example* 1.** *Let us work on the sequential circuit in Figure 1. Suppose the number of selected trace signals is restricted to be one. Let us consider three in-states from the set V:*

$$\begin{aligned} v_0 &= \{i_0 \mapsto 0, i_1 \mapsto 0, A \mapsto 0, B \mapsto 0, C \mapsto 0\} \\ v_1 &= \{i_0 \mapsto 1, i_1 \mapsto 1, A \mapsto 1, B \mapsto 1, C \mapsto 1\} \\ v_2 &= \{i_0 \mapsto 1, i_1 \mapsto 0, A \mapsto 1, B \mapsto 0, C \mapsto 1\} \end{aligned} \quad (3)$$

Figures 2a, 2b and 2c illustrate the three ETMs corresponding to v_0, v_1 and v_2. Each row in an ETM represents an injected fault and each column represents a register. Each entry of the matrix is the corresponding fault propagation function $P(f, r, v_t)$. For example, the entry of row 7 column 3 of Figure 2a is 1, since a fault on g_1 is able to propagate to register C given the in-state v_0. A $\star$ sign annotating a row indicates that the particular fault is triggered.

A ***cumulative ETM*** can be built for V as shown in Figure 2d. In this new ETM, each entry is set to be $P(f, r, V)$ instead of $P(f, r, v_t)$, to represent whether a particular permanent fault is able to propagate to a particular register with respect to a set of in-states in V. To distinguish the original ETMs from this cumulative ETM, the original ETMs are called ***distributed ETMs*** in this paper.

$$\begin{array}{c|ccc} & A & B & C \\ (\mathbf{i_0}, v_0) & 0 & 0 & 0 \\ (\mathbf{i_1}, v_0) & 0 & 0 & 0 \\ (\mathbf{A}, v_0) & 0 & 0 & 0 \\ \star(\mathbf{B}, v_0) & 0 & 0 & 1 \\ (\mathbf{C}, v_0) & 0 & 0 & 0 \\ \star(\mathbf{g_0}, v_0) & 0 & 0 & 1 \\ \star(\mathbf{g_1}, v_0) & 0 & 0 & 1 \\ \star(\mathbf{g_2}, v_0) & 0 & 1 & 0 \\ \star(\mathbf{g_3}, v_0) & 1 & 0 & 0 \end{array}$$

(a) Distributed ETM of v_0

$$\begin{array}{c|ccc} & A & B & C \\ \star(\mathbf{i_0}, v_1) & 1 & 0 & 0 \\ (\mathbf{i_1}, v_1) & 0 & 0 & 0 \\ \star(\mathbf{A}, v_1) & 1 & 1 & 0 \\ \star(\mathbf{B}, v_1) & 1 & 1 & 0 \\ (\mathbf{C}, v_1) & 0 & 0 & 0 \\ (\mathbf{g_0}, v_1) & 0 & 0 & 0 \\ \star(\mathbf{g_1}, v_1) & 0 & 0 & 1 \\ \star(\mathbf{g_2}, v_1) & 1 & 1 & 0 \\ \star(\mathbf{g_3}, v_1) & 1 & 0 & 0 \end{array}$$

(b) Distributed ETM of v_1

$$\begin{array}{c|ccc} & A & B & C \\ (\mathbf{i_0}, v_2) & 0 & 0 & 0 \\ \star(\mathbf{i_1}, v_2) & 0 & 0 & 1 \\ (\mathbf{A}, v_2) & 0 & 0 & 0 \\ \star(\mathbf{B}, v_2) & 1 & 1 & 1 \\ (\mathbf{C}, v_2) & 0 & 0 & 0 \\ \star(\mathbf{g_0}, v_2) & 0 & 0 & 1 \\ \star(\mathbf{g_1}, v_2) & 0 & 0 & 1 \\ \star(\mathbf{g_2}, v_2) & 1 & 1 & 0 \\ \star(\mathbf{g_3}, v_2) & 1 & 0 & 0 \end{array}$$

(c) Distributed ETM of v_2

$$\begin{array}{c|ccc} & A & B & C \\ \mathbf{i_0} & 1 & 0 & 0 \\ \mathbf{i_1} & 0 & 0 & 1 \\ \mathbf{A} & 1 & 1 & 0 \\ \mathbf{B} & 1 & 1 & 1 \\ \mathbf{C} & 0 & 0 & 0 \\ \mathbf{g_0} & 0 & 0 & 1 \\ \mathbf{g_1} & 0 & 0 & 1 \\ \mathbf{g_2} & 1 & 1 & 0 \\ \mathbf{g_3} & 1 & 0 & 0 \end{array}$$

(d) Cumulative ETM of v_0, v_1, v_2

Fig. 2: Fault propagation prepresented in ETMs

$$\begin{array}{c} \forall f \in N,\ C(f,V) \in \{0,1\} \\ S_{\mathbf{A}}, S_{\mathbf{B}}, S_{\mathbf{C}} \in \{0,1\} \\ S_{\mathbf{A}} + S_{\mathbf{B}} + S_{\mathbf{C}} = 1 \\ \forall f \in N, \\ \left.\begin{array}{l} P(f,A,V)S_A + \\ P(f,B,V)S_B + \\ P(f,C,V)S_C \end{array}\right\} \geq C(f,V) \\ \text{max: } \sum_{f \in N} C(f,V) \end{array}$$

(a) Optimizing permanent fault coverage

$$\begin{array}{c} \forall f \in N,\ C(f,V) \in \{0,1\} \\ S_{\mathbf{A}}, S_{\mathbf{B}}, S_{\mathbf{C}} \in \{0,1\} \\ S_{\mathbf{A}} + S_{\mathbf{B}} + S_{\mathbf{C}} = 1 \\ \forall f \in N, \\ \left.\begin{array}{l} P(f,A,V)S_A + \\ P(f,B,V)S_B + \\ P(f,C,V)S_C \end{array}\right\} \geq C(f,V) \\ \text{max: } \sum_{f \in N} \alpha(f) C(f,v_1) \end{array}$$

(b) Optimizing transient fault coverage

Fig. 3: ILP problem for optimizing fault coverage

The ILP formulation of selecting an optimized trace signal in order to maximize permanent fault coverage is shown in Figure 3. Each variable $C(f, V)$ represents whether the corresponding fault is covered under V, given the trace signal selection. Consequently, the objective is to maximize the sum $\sum_{f \in N} C(f, V)$, which provides the permanent fault coverage. Whether an error is captured in a register depends on the selected register. In our example, we restrict the trace buffer to only one register; accordingly, $S_{\mathbf{A}} + S_{\mathbf{B}} + S_{\mathbf{C}} = 1$. Finally, each row in the transmission matrix determines which errors can be captured. The values of $P(f, r, V)$ can be obtained from the cumulative ETM in Figure 2d. By solving this ILP problem, we obtain the optimized trace signal as A.

Although the trace signal A is optimized for permanent fault coverage in Example 1, it does not result in the best coverage for all in-states separately. A more challenging problem is selecting trace signals to enhance both transient fault coverage and permanent fault coverage. We propose a novel trace signal selection approach to maximize transient fault coverage, while maintaining a high permanent fault coverage.

For the same three in-states in V, let us consider the problem of optimizing transient fault coverage, that is, to maximize $\sum_{v_t \in V} \sum_{f \in N} C(f, v_t)$ in Equation 2. A comprehensive ILP formulation of optimizing transient fault coverage in Equation 2 is based on the distributed ETMs. However, the total number of rows in these ETMs is $|N| \times |V|$ and thus at least $|N| \times |V|$ variables are to be optimized in the corresponding ILP. Even for a circuit with thousands of gates fed by thousands of in-states, the number of variables generated in the resulting optimization problem is beyond the capacity of modern optimization solvers. The following theorem provides an approximate solution for maximizing transient fault coverage by using a single cumulative ETM.

$$\begin{array}{cc} & \begin{matrix} A & B & C \end{matrix} \\ \begin{matrix} (\mathbf{i_0}, V_1) \\ \star(\mathbf{i_1}, V_1) \\ (\mathbf{A}, V_1) \\ \star(\mathbf{B}, V_1) \\ (\mathbf{C}, V_1) \\ \star(\mathbf{g_0}, V_1) \\ \star(\mathbf{g_1}, V_1) \\ \star(\mathbf{g_2}, V_1) \\ \star(\mathbf{g_3}, V_1) \end{matrix} & \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \end{array}$$

(a) ETM of group $V_1 = \{v_0, v_2\}$

$$\begin{array}{cc} & \begin{matrix} A & B & C \end{matrix} \\ \begin{matrix} \star(\mathbf{i_0}, V_2) \\ (\mathbf{i_1}, V_2) \\ \star(\mathbf{A}, V_2) \\ \star(\mathbf{B}, V_2) \\ (\mathbf{C}, V_2) \\ (\mathbf{g_0}, V_2) \\ \star(\mathbf{g_1}, V_2) \\ \star(\mathbf{g_2}, V_2) \\ \star(\mathbf{g_3}, V_2) \end{matrix} & \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \end{array}$$

(b) ETM of group $V_2 = \{v_1\}$

Fig. 4: Cumulative ETMs grouped by trigger regions

Theorem 1. *The upper-bound of the transient fault coverage* $\sum_{v_t \in V} \sum_{f \in N} C(f, v_t)$ *is* $\sum_{f \in N} \alpha(f) C(f, V)$*, where* $\alpha(f)$ *is the number of in-states, by which* f *gets triggered in* V*.*

Proof: A transient fault can only be covered by v_t if it is triggered by v_t. We can obtain $\sum_{v_t \in V} \sum_{f \in N} C(f, v_t) = \sum_{v_t \in V} \sum_{f \in tr(v_t)} C(f, v_t)$. From Section II, we have $C(f, V) = \bigvee_{v_t \in V} C(f, v_t)$. Thus, for every $v_t \in V$, we have $C(f, v_t) \leq C(f, V)$. Further, $\sum_{v_t \in V} \sum_{f \in N} C(f, v_t) \leq \sum_{v_t \in V} \sum_{f \in tr(v_t)} C(f, V) = \sum_{f \in N} \alpha(f) C(f, V)$ ■

Intuitively, the parameter α can be regarded as the weight of each node in the circuit. A higher weight indicates that a fault on a node is triggered by more in-states and thus more important for optimizing transient coverage. By optimizing the upper-bound of transient fault coverage instead of the exact transient fault coverage, our technique can handle more complex circuits with a large number of in-states.

***Example* 2.** *Figure 3b gives the formulation for optimizing transient fault coverage based on the cumulative ETM. Nodes are assigned with different weights, e.g* $\alpha(A) = 1$ *and* $\alpha(B) = 3$ *because* A *is triggered in only one distributed ETM, while* B *is triggered in all of the three distributed ETMs.* C*, however, is never triggered as a transient fault and thus assigned a zero weight. By solving this ILP problem, the resulting optimized trace signal for transient fault coverage is* A*.*

B. Clustering In-states Based on Triggered Regions Using Unsupervised Learning

The above static approach to maximize transient fault coverage is limited by the constraint that only one set of trace signals can be traced through the entire debugging run. In our novel dynamic approach, we propose to group the in-states according to their triggered regions. In-states with similar triggered regions are merged as a cumulative ETM with relatively fewer triggered rows. This effectively yields a smaller ETM and allows each group to be optimized by focusing on covering the relatively small triggered region and ignoring the non-triggered regions.

The in-state clustering is a standard clustering problem and can be solved by the K-means technique of unsupervised machine learning. The labels are the groups, while the features are triggered regions.

***Example* 3.** *We continue working on Example 2. The in-states are clustered using the K-means technique according to triggered regions. The triggered regions,* $tr(v_0)$*,* $tr(v_1)$ *and* $tr(v_2)$*, are represented by three vectors,* tr_0*,* tr_1 *and* tr_2*, each of which has* $|N|$ *entries and each entry is 1 if the corresponding fault is triggered (annotated with* $\star$ *sign in 4). It is obvious that the euclidean distance of* $\|tr_0 - tr_2\|$ *is smaller compared to their distances to* tr_1*. Thus, k-means algorithm returns the in-state groups* $V_1 = \{v_0, v_2\}$ *and* $V_2 = \{v_1\}$*.*

Two cumulative ETMs can be built for the two groups separately as shown in Figure 4. For each trace group, a similar ILP formulation is applied for transient fault coverage as in Figure 5a and 5b. The constraints in Figure 5c are added so that a certain level of permanent fault coverage must be achieved. Finally, the transient fault coverage is an objective function in 5d. The constant ξ *can be adjusted by the user to ensure a desired level of permanent fault coverage. This constraint can be removed if permanent fault coverage is not a concern in debug. This ILP formulation accomplishes two objectives: 1. Ensure a reasonable permanent fault coverage through a constraint. 2. Maximize transient fault coverage.*

$$\begin{array}{l} \forall f \in N,\ C(f, V_1) \in \{0, 1\} \\ S^{V_1}_{\mathbf{A}}, S^{V_1}_{\mathbf{B}}, S^{V_1}_{\mathbf{C}} \in \{0, 1\} \\ S^{V_1}_{\mathbf{A}} + S^{V_1}_{\mathbf{B}} + S^{V_1}_{\mathbf{C}} = 1 \\ \forall f \in N, \\ \left.\begin{array}{l} P(f, A, V_1) S^{V_1}_A + \\ P(f, B, V_1) S^{V_1}_B + \\ P(f, C, V_1) S^{V_1}_C \end{array}\right\} \geq C(f, V_1) \end{array}$$

(a) Transient fault coverage for V_1

$$\begin{array}{l} \forall f \in N,\ C(f, V_2) \in \{0, 1\} \\ S^{V_2}_{\mathbf{A}}, S^{V_2}_{\mathbf{B}}, S^{V_2}_{\mathbf{C}} \in \{0, 1\} \\ S^{V_2}_{\mathbf{A}} + S^{V_2}_{\mathbf{B}} + S^{V_2}_{\mathbf{C}} = 1 \\ \forall f \in N, \\ \left.\begin{array}{l} P(f, A, V_2) S^{V_2}_A + \\ P(f, B, V_2) S^{V_2}_B + \\ P(f, C, V_2) S^{V_2}_C \end{array}\right\} \geq C(f, V_2) \end{array}$$

(b) Transient fault coverage for V_2

$$\begin{array}{l} \sum_{f \in N} C(f, V) \geq \xi \\ \forall f \in N,\ C(f, V) \in \{0, 1\} \\ \sum_{i \in \{1,2\}} C(f, V_i) \geq C(f, V) \end{array}$$

(c) Permanent fault coverage

$$\max: \sum_{i \in \{1,2\}} \sum_{f \in N} \alpha(f) C(f, V_i)$$

(d) transient fault coverage

Fig. 5: ILP for optimizing dynamic trace signal selection

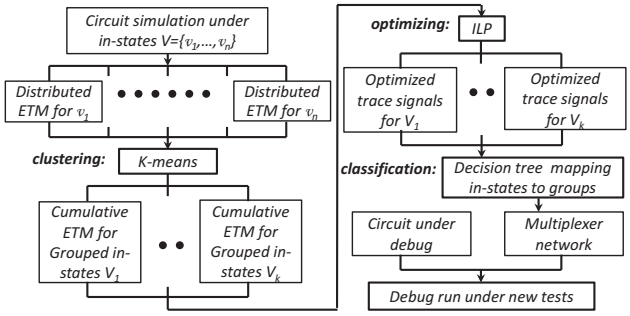


Fig. 6: Algorithm flow

C. Classifying New In-States Using Supervised Learning

The algorithm above learns a statistical model which maps a series of in-state samples to their corresponding optimized trace signal selections. These learned in-states are labeled with groups. During the test run, however, new in-states without labels will occur and a prediction for its group must be made. This can be formulated as a supervised machine learning problem, in which the groups are labels and the vector value of the in-states are features. The learned statistical model is implemented as a decision tree, which is further implemented by a multiplexer network embedded in the tested circuit to predict the group of a new in-state during debugging run.

In summary, our novel dynamic trace signal selection approach consists of three main phases as shown in Figure 6. First, in the Clustering Phase, the sample in-states are clustered according to their trigger regions, using the K-means unsupervised learning technique. Second, in the Optimization Phase, each group of the in-states is calculated for an optimized trace signal selection to optimize transient and permanent fault coverage, using an ILP formulation. Finally, in the Classification Phase, a decision tree is built based on the grouping of in-states and the corresponding multiplexer network implementing this decision tree is embedded into the chip to classify new in-states encountered during a debugging run.

D. Related Work

[5] [3] have propsed techniques for static trace buffer selection based on signal restoration computation. Based on their technique, only one set of trace signals are selected. Yang et al. [8] and Zhu et al. [9] proposed a different trace signal selection algorithm based on fault coverage analysis and using an error transmission matrix. However, all of the above approaches are for static trace signal selection.

Prabhakar et al. [7] developed a dynamic trace signal selection method by using two sets of signals alternately in even and odd cycles. Liu et al. select several trace signal groups based on a new design error visibility metric [6]. But both algorithms select trace groups independent from the input and state values and they are not optimized for transient faults. Basu et al. [2] proposed a dynamic trace signal selection approach based on active regions in a circuit. However, this is based on the knowledge of a circuit's error zones and regions, which are not available in many scenarios.

IV. Experimental Results

To determine the effectiveness of our algorithm, we conducted experiments on 20 benchmark circuits in AIG netlist format: 2 circuits from *Opencores.org*, 5 circuits from *ITC'99*, 13 circuits from *HWMCC'10 benchmarks*.

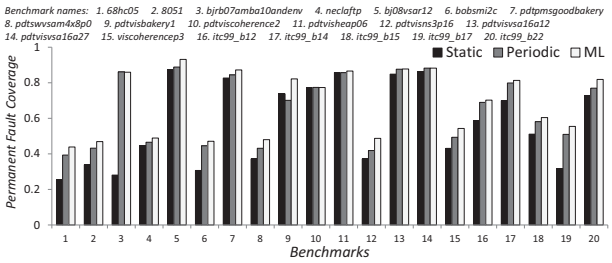
We use *CPLEX* as the ILP optimizer and the *OpenCV* library for the K-means and decision tree algorithm. In the clustering phase, each benchmark is simulated by 2000 random sample in-states to obtain 2000 distributed ETMs. Then these 2000 in-states are clustered into 5 groups (number of groups can be varied) according to different triggered regions using the K-means. In the optimization phase, we constrain that only 5% of the registers are accessible as trace signals and the CPLEX ILP solver finds the optimized trace signals for 5 trace groups. To choose a reasonably high ξ parameter for our algorithm, we run a binary search for the maximum value of ξ, such that a valid trace signal optimization can still be found by the ILP solver. Finally, in the classification phase, a decision tree is trained based on the groups' in-states. We set the maximum height of the tree to be 5 in our experiment. The resulting decision tree is implemented in a multiplexer tree to select trace groups during the debugging run. In our implementation, the multiplexer trees are constructed by 16 to 512 2-to-1 multiplexers for the different benchmarks.

To evaluate the effectiveness of these multiplexer trees, we conducted two set of evaluation experiments for permanent faults and transient faults. For both experiments, we compare three algorithms: (i) A static trace signal selection algorithm as proposed in [8]. (ii) A dynamic trace signal selection algorithm which switches among the trace groups periodically every 200 cycles and is independent from in-states, as proposed in [6]. (iii) Our technique which selects trace signals based on the in-states of each cycle. The results are labeled as "Static", "Periodic" and "ML" respectively in Figure 7a and 7b.

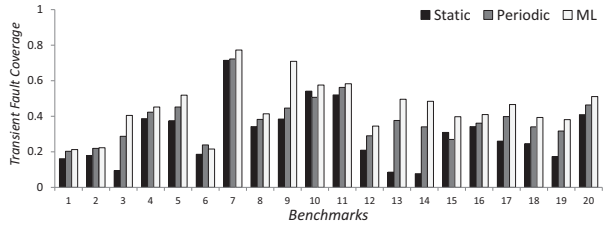
3000 single stuck-at faults and single transient faults are randomly injected into each of the 20 benchmarks. Each transient fault occurs only once during the debugging run. After a test run crashes due to an observable error, partial states in the last 1000 cycles (trace buffer depth) are available. The results of permanent and transient fault coverage are shown in Figure 7a and 7b. Our approach is able to detect 20% more permanent faults and 50% more transient faults compared to the "Static" approach, and 5% more permanent faults and 18% more transient faults compared to the "Periodic" approach.

V. Conclusions

One major challenge in post-silicon validation is improving observability, which can be achieved by using trace buffers.



(a) Permanent Fault Coverage Comparison



(b) Transient Fault Coverage Comparison

Fig. 7: Permanent and Transient Fault Coverage Experiment

The effectiveness of the debug process depends on how well the trace signals are chosen. We proposed a dynamic trace signal selection algorithm to enhance both permanent and transient fault coverage by using machine learning and integer linear programming techniques. Compared to the existing dynamic selection approaches, our algorithm dynamically selects trace signals depending on the input and state value patterns, and is able to achieve higher fault coverage, especially for transient faults. Another advantage of our approach is that its efficacy does not depends on any higher level understanding/specifications, other than the gate level design.

Acknowledgment: This work was supported in part by C-FAR, one of the six SRC STARnet Centers, sponsored by MARCO and DARPA.

References

[1] M. Abramovici, P. Bradley, K. Dwarakanath, P. Levin, G. Memmi, and D. Miller. A reconfigurable design-for-debug infrastructure for SoCs. In *DAC*, pages 7–12, 2006.

[2] K. Basu, P. Mishra, P. Patra, A. Nahir, and A. Aadir. Dynamic selection of trace signals for post-silicon debug. In *Microprocessor Test and Verification, 2013. MTV'13. 14th International Workshop on*. IEEE, 2013.

[3] D. Chatterjee, C. McCarter, and V. Bertacco. Simulation-based signal selection for state restoration in silicon debug. In *Computer-Aided Design (ICCAD), 2011 IEEE/ACM International Conference on*, pages 595–601. IEEE, 2011.

[4] A. DeOrio, Q. Li, M. Burgess, and V. Bertacco. Machine learning-based anomaly detection for post-silicon bug diagnosis. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 491–496. EDA Consortium, 2013.

[5] H. F. Ko and N. Nicolici. Algorithms for state restoration and trace-signal selection for data acquisition in silicon debug. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 28(2):285–297, 2009.

[6] X. Liu and Q. Xu. On multiplexed signal tracing for post-silicon validation. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 32(5):748–759, 2013.

[7] S. Prabhakar and M. S. Hsiao. Multiplexed trace signal selection using non-trivial implication-based correlation. In *Quality Electronic Design (ISQED), 2010 11th International Symposium on*, pages 697–704. IEEE, 2010.

[8] J.-S. Yang and N. A. Touba. Efficient trace signal selection for silicon debug by error transmission analysis. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 31(3):442–446, 2012.

[9] C. S. Zhu, G. Weissenbacher, and S. Malik. Coverage-based trace signal selection for fault localisation in post-silicon validation. In *Hardware and Software: Verification and Testing*, pages 132–147. Springer, 2013.