Scalable Trace Signal Selection using Machine Learning

Kamran Rahmani and Prabhat Mishra
Department of Computer & Information Science & Engineering
University of Florida, Gainesville FL 32611, USA
{kamran, prabhat}@cise.ufl.edu

Sandip Ray
Strategic CAD Labs
Intel Corporation, USA
sandip.ray@intel.com

***Abstract*—A key problem in post-silicon validation is to identify a small set of traceable signals that are effective for debug during silicon execution. Structural analysis used by traditional signal selection techniques leads to poor restoration quality. In contrast, simulation-based selection techniques provide superior restorability but incur significant computation overhead. In this paper, we propose an efficient signal selection technique using machine learning to take advantage of simulation-based signal selection while significantly reducing the simulation overhead. Our approach uses (1) bounded mock simulations to generate training vectors set for the machine learning technique, and (2) an elimination approach to identify the most profitable signals set. Experimental results indicate that our approach can improve restorability by up to 63.3% (17.2% on average) with a faster or comparable runtime.**

I. Introduction

The goal of post-silicon validation is to ensure that the fabricated, pre-production silicon functions correctly while running actual applications under on-field operating conditions. Post-silicon validation is a complex activity performed under aggressive schedule, accounting for more than 50% of the overall validation cost of a modern integrated circuit [10], [16]. A fundamental challenge in post-silicon validation is limited observability and controllability. Limitations in the number of output pins, coupled with restrictions imposed by area and power constraints on internal trace buffer sizes imply that only a few hundreds among the millions of internal signals can be traced during a silicon execution. Furthermore, in order for a signal to be observed, the design must be instrumented a priori with appropriate hardware that routes the signal to an observation point. It is therefore crucial to develop techniques to identify trace signals that maximize design visibility and debug information under the constraints imposed by the post-silicon observability restrictions.

Research in post-silicon validation has attempted to address the above issue by developing algorithms for selecting trace signals through automatic analysis of pre-silicon (RTL or gate-level) designs. The idea is to select a set of signals S that maximizes *state restorability*, *i.e.*, the set of states that can be reconstructed based on the observation of the signals in S. Traditionally, signal selection has entailed defining a metric based on the structure, which is then used in a (typically greedy) selection process to evaluate a candidate signal set [9], [1], [6], [13]. These approaches are fast but the restoration quality, *i.e.*, the number of design states that can be reconstructed based on the observation of traced signals, has been low. Recent work on simulation-based signal selection [4] provides superior restoration quality but incurs prohibitive computation overhead. A hybrid signal selection approach [8] has been proposed which incorporated a combination of metric-based and simulation-based signal selection approaches. However, using less simulation to save selection time sacrifices the restoration performance.

In this paper, we develop a novel signal selection technique that retains (and improves upon) the restoration quality of simulation-based signal selection while in a faster or comparable signal selection time.

This work was partially supported by NSF grant CCF-1218629.

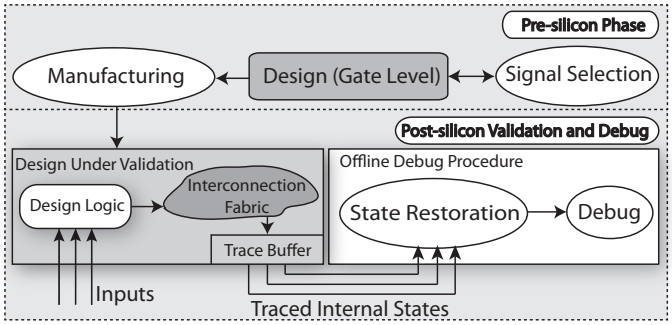

Fig. 1. Simplified overview of post-silicon validation flow and role of trace signal selection. Signals selected through pre-silicon analysis are routed to trace buffer from which signal states are restored offline to assist in debug.

Our approach is characterized by two key components: (1) a machine learning technique to model the restoration strength of the signals, and (2) an elimination based selection technique to find the most profitable set of signals. Our experiments demonstrate that our approach can improve restoration quality by 63.3% (17.2% on average).

The remainder of the paper is organized as follows. Section II presents the relevant background. We present the technical details of our approach in Section III followed by experimental results in Section IV. Section V discusses related work. We conclude in Section VI.

II. Background And Motivation

A. Post-silicon Validation Overview

Figure 1 provides a very simplified overview of post-silicon validation and debug process focusing on the role of signal selection. A modern IC design includes debug mechanisms such as *embedded logic analyzers* (ELA) to record values of internal design signals during silicon execution. An ELA consists of trigger and sampling units; trigger units are used to specify events that trigger recording initiation, and sampling units then record a small set of signals in the trace buffer for a specified number of cycles. The sampled signals can then be transferred from the trace buffer for off-chip analysis. In particular, the off-chip analysis can apply restoration algorithms using the sampled signals to infer the values of other design signals and reconstruct internal states. The traced and restored signal values can be used together to detect design errors. To make this possible, the set of signals to be sampled is selected *a priori* by pre-silicon analysis of the design. Note that the number of sampled signals is restricted by the width of the trace buffer and typically represents a very small fraction of the internal signals in the design. Thus ideally one would like to choose the set of signals that permit maximum reconstruction of design states. Unfortunately, exhaustive exploration of all signal subsets to determine this optimum is computationally intractable; most signal selection approaches [9], [1], [6], [4] involve developing heuristics that are efficient in practice while still yielding signals with good restorability properties.

B. Signal Restoration

Restoration entails inferring values of untraced signal states from a sequence of traced signals sampled over a period of time. This is achieved by forward and backward propagation of signal values of circuit elements (*e.g.*, gates, latches, etc.): forward propagation involves reconstructing the output of a circuit element from traced inputs, while backward propagation involves inferring input values from the observed output. Note that backward reconstruction may be partial. Restoration Ratio (RR), defined below, is a popular metric for measuring the quality of a set of selected trace signals.

$$\text{Restoration Ratio} = \frac{\text{No. of traced and restored signals}}{\text{No. of traced signals}}$$

TABLE I. ILLUSTRATION OF RESTORED SIGNALS FOR THE SIMPLE CIRCUIT SHOWN IN FIGURE 2. THE TRACED SIGNALS A AND C ARE SHADED. AN X INDICATES THAT THE SIGNAL VALUE CANNOT BE RESTORED AT THAT CYCLE USING THE KNOWN SIGNAL STATES.

Signal/Cycle	1	2	3	4	5	6	7	8
A	0	1	0	0	0	1	1	1
B	0	X	1	1	1	X	X	X
C	0	0	1	1	1	1	1	1
D	X	0	0	0	0	0	1	1
E	X	0	0	1	1	1	X	X
F	X	X	0	0	1	1	1	1
G	X	0	X	0	0	0	1	1
H	X	0	X	0	0	1	X	X

Consider the simple circuit shown in Figure 2. Suppose that the width of the trace buffer is 2 (*i.e.*, only two signals can be traced at any clock cycle), and the trace buffer depth is 8 (*i.e.*, selected signals are traced for 8 cycles). Suppose that A and C are selected as trace signals. Table I shows the signal states that can be restored: 52 signal values can be restored while 16 are traced, yielding a restoration ratio of 3.25.

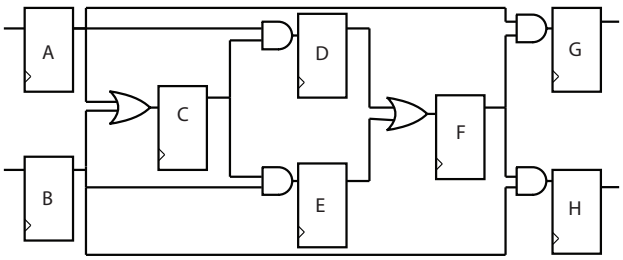

Fig. 2. A simple circuit to illustrate restorability.

C. Simulation-based Signal Selection

Our work is motivated by simulation-based signal selection proposed by Chatterjee *et al.* [4]. They showed that mock simulations are more effective in identifying trace signals than metrics based on the circuit structure. Simulation-based signal selection is tractable for two key reasons. First, there is negligible overall variation in restoration quality over different random input vectors. Second, restoration ratio is insensitive to the trace buffer depth beyond a certain size. The first observation permits the use of a single simulation with one random input vector for evaluation of restoration quality; the latter reduces the evaluation time by using a smaller trace buffer depth in mock simulations. Chatterjee *et al.*'s approach involves an iterative removal process. They start with a set of candidate signals which is initialized with all the flip-flops. In each iteration, their algorithm attempts to remove one of the signals which seems to be least important based on simulation results. The process continues until the number of remaining candidates equals to the trace buffer width.

Note that if the initial candidates set includes N flip-flops, O(N^2) simulations are required to reach the final set. The cost of simulations makes the approach computationally prohibitive for large circuits. To address this issue, they propose a pruning phase prior to running the algorithm. In each iteration of their pruning phase, d_{step} flip-flops are removed from the candidates set, instead of just one flip-flop. The pruning phase continues until the average restorability of the candidates set drops below $PT \times R_{max}$, where PT is the cutoff threshold (typically $PT = 0.95$) and R_{max} is the maximum restorability when all the flip-flops are present in the candidates set.

There are two primary issues with such a pruning approach. First, the coarse-grained elimination parameter d_{step} can be detrimental to the quality of selection process, as some important signals may be removed during preprocess. Furthermore, even with pruning, the complexity of the approach is $\Omega(N^2/d_{step})$; since for most large circuits $N >> d_{step}$, the algorithm is still computationally prohibitive for industial-scale circuits.

III. LEARNING-BASED SIGNAL SELECTION

Our approach makes use of machine learning techniques to ameliorate the cost of mock simulations identified above in simulation-based signal selection. In particular, we propose a two-step signal selection approach using semi-supervised learning: in the first (pre-processing) step, a small number of mock simulations is used as a training set; in the second (selection) step, we use prediction to replace expensive mock simulations with simple calculations.

A. Problem Formulation

The goal of a selection algorithm is to construct a set S of w flip-flops (out of N flip-flips in the circuit) so that restoration ratio during post-silicon debug is maximized. Here w is the width of the trace buffer and is a parameter to the algorithm. To motivate our approach, we first provide a rigorous formulation of signal selection as a constrained optimization problem. First note that the selected signal set S can be mapped to a *feature vector* $v = \langle f_1, f_2, ..., f_N \rangle$, with $f_i \in \{0, 1\}$. Informally, $f_i = 1$ if and only if the i-th flip-flop is selected in S, otherwise 0. Note that v completely identifies the set S and vice versa; we will refer to S as the *candidate signal set* of v and v as the *candidate feature set* of S. We then define $r_m(v)$ to be the number of signal states that can be restored over a window of m cycles by tracing the candidate signal set of v. We then formulate the problem of signal selection as the following constrained optimization problem.

$$\begin{aligned} &\max \quad r_m(v) \\ &\text{under constraint} \quad \sum_{k=1}^{N} f_k = w \end{aligned} \tag{1}$$

The problem as posed above includes both the trace buffer width (w) and simulation window (m) as parameters. Clearly, a larger value of m yields more accurate restoration estimation, and consequently, higher restoration ratio during debug. However, previous work [4] showed that even choosing a small value of m (*e.g.*, for $m = 64$), there is a strong correlation between the restoration quality in m cycles and that in a real post-silicon debug scenario. Thus, for the rest of this paper, we treat m as a small constant.

B. Overview of Our Approach

Solving the above optimization problem requires an estimation of $r_m(v)$ given a feature vector v. Indeed, both metric-based and simulation-based selection approaches can be seen as approaches to

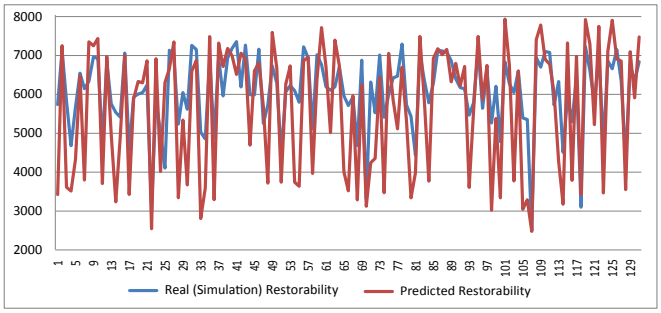

Fig. 3. Real values of $r_m(v)$ versus predicted values for 130 random vectors in s5378 benchmark using support vector regression. Each random vector represents a set of randomly selected trace signals.

estimate this function, through structural analysis of the circuit, and applying mock simulation with restoration, respectively. The lower restoration quality of metric-based approaches are attributed to the fact that extracting this function from circuit structure alone is often infeasible due to complicated overlaps between restorable states of different flip-flops. On the other hand, simulation-based techniques are expensive for industrial circuits, even for a small simulation window, since the circuit size (and therefore the size of the feature vector v) is large.

Our approach uses machine learning techniques to estimate $r_m(v)$. Many machine learning techniques, *e.g.*, regression analysis, use a small set of *training vectors* to estimate a model of the function, which is then used to predict the results. In our case, the training vectors come from restoration estimates obtained from mock simulations for given feature vectors. If the training set is small (*i.e.*, only a small set of mock simulations is necessary), and the predicted model is accurate, then the technique can provide high restoration quality at low computation cost. Regression analysis techniques are effective in predicting the parameter estimates in cases where (1) the number of parameters is large, and (2) estimation through exhaustive (or even significant) simulation of all the parameters is infeasible. Thus these techniques are appropriate for solving the signal selection problem as posed in our formulation.

Nevertheless, applying these techniques directly on the problem is challenging. In particular, the regression analysis techniques require generation of training vectors such that (1) generation time is reasonable, and (2) a reasonable number of vectors is generated to avoid deviation of the estimated model of the function from the (unknown) actual model. Note that if the number of training vectors is too small, it is possible to fit a large number of regression models to this set, and can lead to underfitting (*i.e.*, fitting a model that is too simplistic), or overfitting (fitting a model that is too complex), leading to high prediction error. Furthermore, the class of curves that model the function is another important factor. For example, linear fitting may not be a good choice for modeling the complicated non-linear relationships between the flip-flops of the circuit.

We investigated the effectiveness of three regression techniques; *linear regression*, *neural network regression*, and *support vector regression*. As expected in theory, support vector regression produced most accurate results for the set of our benchmarks.[1] Figure 3 shows the relationship between the real value of $r_m(v)$ (calculated using simulation) and the predicted value for 130 random vectors where $m = 64$ in *s5378* benchmark. Each random vector represents a set of randomly selected trace signals. The support vector regression is used for modeling and prediction. Observe that although the real and predicted values do not match exactly, the relative pattern over the vectors is consistent for both of them. In fact, in a selection process, only the relative relation between different $r_m(v)$ values is enough to choose the most effective vector. This enables us to use the predicted values instead of the real ones without significant loss in quality.

[1]The circuit modeling problem is not linear separable to be modeled using *linear regression*. In addition, *neural network regression* may get stuck in a local extreme, if it can find such a point.

C. Signal Selection Algorithm

In order to increase the accuracy of the prediction as well as to reduce the runtime of modeling/prediction in large circuits, we propose a two-step modeling scheme. Figure 4 illustrates the framework. In the first step, a linear modeling is applied to eliminate less important flip-flops and to reduce the size of feature vector. Although the accuracy of linear modeling is low, it is fast and can be used to quickly prune out the non-beneficial signals and determine top candidates using simple calculations. In the second step, a non-linear regression is applied on the reduced set to produce a finer model of the remaining flip-flops. The reduced number enables us to use a more accurate non-linear model with fewer training vectors. Finally, a further elimination-based selection is applied to remaining flip-flops to select top w candidates. We now discuss the different steps of the algorithm in more detail.

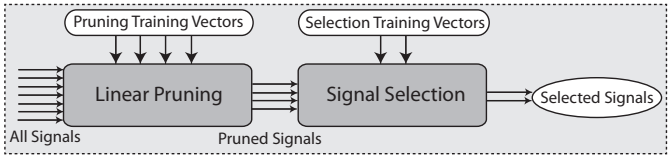

Fig. 4. Proposed signal selection process. A quick linear model is used to eliminate most of the non-beneficial flip-flops. A more accurate non-linear model is used to select *w* flip-flops out of the remaining flip-flops where *w* is the trace buffer width.

1) Generating Training Vectors: Algorithm 1 outlines the pseudo-code for training vector generation. Our implementation entails an X-simulator in C++ which can conduct the simulation as well as forward/backward restoration in the circuit. To consider the effect of each flip-flop on total restorability, two vectors are generated. First a vector in which only a particular flip-flop is selected, and second a vector in which all the flip-flops are selected except that particular flip-flop. In addition, to include the vectors with different number of flip-flops, $N-1$ vectors with $2, 3, \ldots, N$ randomly chosen flip-flops are generated. This process continues until a total number of t vectors are generated. This unbiased random vector can model the correlation between the effect of different flip-flops. After generating training vectors, the corresponding $r_m(v)$ is generated using a mock simulation followed by a restoration process over m cycles. Finally, we have t pairs $\langle v_i, r_m(v_i)\rangle$ that are used as training vectors for the regression technique. The set of generated vectors S and corresponding restorability R are returned as the output of algorithm.

2) Linear Pruning: In order to improve the prediction accuracy and also decrease the runtime of simulation/modeling we apply a pruning phase which is equivalent to feature selection in machine learning. In this step, a linear modeling is used to quickly eliminate most of the non-beneficial flip-flops (in term of restorability effectiveness). Given the training set $\langle v_i, r_m(v_i)\rangle$, the support vector regression solution is a set of j support vectors which is used for predicting new vectors. Denoting the predicted $r_m(v)$ as $\hat{r}_m(v)$, we have the following equation.

$$\hat{r}_m(v) = \hat{w}_0 + \sum_{k=1}^{j} \alpha_k k(v_k, v) \quad (2)$$

In Equation 2, v is the vector whose restorability we wish to predict, v_k is the k^{th} support vector, and α_k is the corresponding coefficient. In addition, $k(v_k, v)$ is the output of the kernel function

Algorithm 1 Training Vector Generation

procedure GENERATEVECTORS($circuit, m, t$)
  Create training vectors set S
  Create restoration power set R
  $totalGenerated = 0$
  for each flip-flop f in $circuit$ **do**
    Add a vector to S in which only f is selected
    Add a vector to S in which only f is omitted
    totalGenerated=totalGenerated+2
  end for
  for $i = 2; i <= N; i++$ **do**
    Add a vector to S in which exactly i random flip-flops are chosen
    $totalGenerated++$
  end for
  while $totalGenerated < t$ **do**
    $length$= a random number between 1 and N
    $randomVector$=a vector in which exactly $length$ random flip-flops are chosen
    if $randomVector \notin S$ **then**
      Add $randomVector$ to S
      $totalGenerated++$
    end if
  end while
  for each vector v in S **do**
    $R(v)$ = Restoration power of v using a mock simulation followed by a restoration process over m cycles
  end for
  return S, R
end procedure

used in support vector regression. In linear mode, the kernel function is of the form $k(v_k, v) = v_k^T.v$, where v_k^T is the transpose of v_k. Then we can rewrite Equation 2 as follows.

$$\hat{r}_m(v) = \hat{w}_0 + \sum_{k=1}^{j} \alpha_k v_k^T.v \tag{3}$$

$$\Rightarrow \hat{r}_m(v) = \hat{w}_0 + \hat{w}^T.v \ (where\ \hat{w} = \sum_{k=1}^{j} \alpha_k v_k) \tag{4}$$

Equation 4 illustrates the simplified version of the prediction formula when a linear kernel is used. In fact, the model is a simple hyperplane which has the minimum error amongst all the hyperplanes over the training set. Although this linear model may not be the best fit for the non-linear function $r_m(v)$, it can be used to quickly detect and eliminate non-beneficial flip-flops. Algorithm 2 outlines the linear pruning process. First, a set of training vectors is generated followed by a linear modeling using support vector regression. Next, the weight vector $\hat{w}$ of predicted function is calculated as illustrated in Equation 4. Those flip-flops with most effect on restorability have largest values in corresponding index of weight vector. Therefore, the index of $p \times N$ largest values in weight vectors are kept as most useful flip-flops in terms of restorability and the rest is removed. Here, N is the number of flip-flops in the circuit and p is the pruning factor. Smaller p means less features in next step which leads to a more accurate and faster non-linear model. However, due to lower accuracy of linear model, lower value of p will also increase the chance of eliminating a useful flip-flop by mistake. In our experiments, we set $p = 0.15$. The output of the process is the preserved flip-flops set S.

Algorithm 2 Linear Pruning Algorithm

procedure LINEARPRUNING($circuit, m, t, p$)
  Create selected features set S
  $trainVectors$ =GenerateVectors($circuit, m, t$)
  Model $\hat{r}_m(v)$ using support vector regression with $trainVectors$ and linear kernel
  Calculate the weight vector $\hat{w} = \sum_{k=1}^{j} \alpha_k v_k$
  S = the index of top $p \times N$ values in vector $\hat{w}$
  return S
end procedure

3) Signal Selection: The reduced number of flip-flops in feature vector enables us to create a more accurate non-linear model of the circuit with significantly less number of training vectors. The effective number of required training vectors in second step is reduced by $1 - p$, where p is the pruning factor. The second step is the actual signal section procedure. Therefore, in this step, having an accurate model of the $r_m(v)$ is essential in order to select most profitable set of signals. Using kernel trick along with a non-linear kernel function, we can model complex non-linear functions using support vector regression. This function is replaced by linear function ($k(v_k, v) = v_k^T.v$), which is used in pruning step. Although new kernels are being proposed by researchers, there are general-purpose well-known functions which are shown to be effective in most of the scenarios. We investigated *polynomial*, *radial basis function (RBF)*, and *sigmoid* kernel functions. However, *sigmoid* function demonstrated to be the best fit for restoration prediction in our experiments. This function is defined as $k(v_k, v) = tanh(\gamma v_k^T.v + r)$[2], where γ and r are the kernel parameters.

Algorithm 3 outlines the step involved in our proposed signal selection algorithm. First, a linear pruning is applied to circuit using $t_{pruning}$ training vectors. Next, a new set of $t_{selection}$ training vectors is generated and a non-linear model is created using pruned features set. After the pruning and modeling phases, all the remaining flip-flops are set to be selected in signals vector v (i.e., are set to 1). In each iteration of the algorithm, a signal which has the minimum impact on restoration performance of the v is eliminated from the vector (i.e., is set to 0). Here, instead of evaluating $r_m(v)$ using mock simulations, the predicted value $\hat{r}_m(v)$ is used. This enables the algorithm to proceed very fast, while utilizing the high prediction accuracy of a non-linear model. This process continues until the number of remaining flip-flops is equal to trace buffer width w. The set of selected signals S is returned as the algorithm output.

D. Complexity and Scalability

Simulation of large industrial designs incurs high cost in running time. Indeed, simulation time is the primary bottleneck in the usability of simulation-based signal selection on large-scale designs. Therefore, a good metric of the complexity of such algorithms is the number of mock simulations/restoration processes required in the computation. Assume that there are N flip-flops in the circuit. In our approach, mock simulations are required in generating the training vectors, including pruning and the selection steps. Therefore, a total number of $t_{pruning} + t_{selection}$ simulations are conducted. In pruning phase, since the model is a simple linear one, our experiments demonstrated that choosing $t_{pruning} = 4 \times N$ is enough in order to eliminate non-beneficial flip-flops when $p = 0.15$. We also discovered that $t_{pruning} = 10 \times N_{select}$ is the beneficial number of training vectors for non-linear model

[2] $tanh(x) = \dfrac{1 - e^{-2x}}{1 + e^{-2x}}$

Algorithm 3 Learning-based Signal Selection

procedure SELECT($circuit, m, t_{pruning}, p, t_{selection}, w$)
  Call LinearPruning ($circuit, m, t_{pruning}, p$)
  $trainVectors$ =GenerateVectors($circuit, m, t_{selection}$)
  Model $\hat{r}_m(v)$ with pruned features using support vector regression, $trainVectors$, and a non-linear kernel
  Create selected signals set S
  Create initial vector of $v =< 1, 1, ..., 1 >, |v| = N \times p$
  $remainedSignals = N \times p$
  while $remainedSignals > w$ **do**
    $maxRestorability = -\infty$
    $maxIndex = -1$
    for $i = 1; i <= N \times p; i++$ **do**
      if $v[i] = 1$ **then**
        $v[i] = 0$
        if $\hat{r}_m(v) > maxRestorability$ **then**
          $maxRestorability = \hat{r}_m(v)$
          $maxIndex = i$
        end if
        $v[i] = 1$
      end if
      $v[maxIndex] = 0$
    end for
    $remainedSignals = remainedSignals - 1$
  end while
  for $i = 1; i <= N \times p; i++$ **do**
    if $v[i] = 1$ **then**
      Add i to S
    end if
  end for
  return S
end procedure

which prevents overfitting and underfitting, where $N_{select} = p \times N$. Therefore, in our experiments, $t_{pruning} = 10 \times 0.15 \times N = 1.5 \times N$. It means, the total number of mock simulations in our approach is $5.5 \times N$, which is much less than $\Omega(N^2/d_{step})$ reported in previous work [4], where $d_{step} = 50$ in their experiments. On the other hand, the hybrid approach [8], uses simulation/restoratin computation only for top $k\%$ of the candidate signals, where $k = 5\%$ in their experiments. The complexity of their approach is $O(kwN)$ where w is the trace buffer width. It can be observed that once the parameters are fixed, the asymptotic complexity of our approach and [8] is identical ($\theta(N)$), with potentially different constant coefficients.

IV. EXPERIMENTS

A. Experimental Setup

In order to investigate the effectiveness of our proposed approach, we have developed a cycle-accurate simulator for ISCAS'89 benchmarks using C++. Our simulator also conducts restoration in both forward and backward directions. The simulator iterates on the unknown signals queue and attempts to restore them leveraging both forward and backward restoration techniques. This process terminates when it is not possible to restore any more states. In addition, we checked the correctness of our simulator by comparing its output with the output of Verilog simulation of the identical circuits using *Icarus Verilog* [14]. We used *LIBSVM* [3] as the support vector regression modeling/prediction tool. In addition, we used *5-fold cross validation* technique to choose the best set of parameters in the support vector regression and the kernel functions.

In our experiments, we compared our approach with our implementation of [4] and [8]. We used our implementation of [4], [8] for the comparison for two reasons. First, their reported results used their own synthesized/optimized version of the ISCAS'89 benchmarks which are inaccessible to us, while we used the standard, publicly available versions of ISCAS'89 benchmarks.. In addition, to make the comparison of the runtime fair, same simulation/restoration platform needs to be used in all the approaches. We used the same parameters $c = 64$ and $PT = 95\%$ as reported in [4]. In addition, we used the same parameters $M = 64$, $k = 5\%$, and an initialization simulation of 10K cycles as reported in [8]. We also used $m = 64$, $p = 0.15$, $t_{pruning} = 4 \times N$, and $t_{selection} = 1.5 \times N$ as our approach parameters where N is the number of flip-flops in the circuit. For reporting the restoration ratios, we fed the simulator with 100 sets of random input vectors and noted the average restoration ratios for the selected set of signals. However, we forced the circuits to operate in their normal mode by fixing the relevant control (reset) signals, while assigning random values to all the other inputs. The control signals include active low reset signals *RESET* in *s35932* and *g35* in *s38584* which was set to 1 in our experiments.

B. Signal Selection Time

The run-time result comparison used an Ubuntu 10.04.4 machine with a Dual-Core AMD Opteron 222SE (3000 MHz) processor and 16 GB of memory for all the experiments. Although [4] and [8] used a multi-thread or GPU-based implementation for their simulation core, to make the comparison fair, we used a single thread program for all the techniques. This enables us to highlight the significant runtime differences of the approaches. The runtime of our approach is calculated as the summation of required time for generating training vectors (simulations), modeling, and signal selection process itself.

TABLE II. RUNTIME COMPARISON OF OUR APPROACH COMPARED WITH EXISTING SELECTION APPROACHES

Circuit	#Flip-flops	Buffer Width	Simulation-based [4]	Hybrid [8]	Learning-based
s5378	179	8	00:01:53	00:00:03	00:00:13
		16	00:01:52	00:00:14	00:00:13
		32	00:01:48	00:00:19	00:00:13
s9234	228	8	00:08:52	00:00:15	00:00:47
		16	00:08:43	00:00:33	00:00:47
		32	00:08:10	00:00:52	00:00:47
s15850	597	8	03:44:12	00:00:59	00:07:19
		16	03:44:04	00:03:39	00:07:19
		32	03:43:39	00:04:30	00:07:19
s13207	669	8	01:21:41	00:01:08	00:05:46
		16	01:21:35	00:03:17	00:05:45
		32	01:21:13	00:04:07	00:05:44
s38584	1452	8	28:43:02	00:13:54	01:31:59
		16	28:42:16	00:42:56	01:31:00
		32	28:38:59	01:01:33	01:30:03
s38417	1636	8	196:51:50	00:16:07	02:00:32
		16	196:50:44	00:47:52	02:00:25
		32	196:48:27	01:15:09	02:00:16
s35932	1728	8	11:39:36	00:17:27	01:41:34
		16	11:39:09	00:52:12	01:41:30
		32	11:38:01	01:22:08	01:41:23

Table II presents the runtime of our approach compared with previous techniques [4], [8] using different ISCAS'89 benchmarks. The reported runtime format is 'hour:minute:second'. From the table, as expected, it is clear that our approach is significantly faster than pure simulation-based approach presented in [4]. Moreover, we note that our approach runtime is comparable to hybrid approach [8], specially for the larger trace buffer widths. The reason is that once the circuit is modeled in our approach, the selection process can be done in negligible time using simple calculations. This makes our approach runtime independent of the trace buffer width which is not the case in

[8]. This makes our approach more scalable in industry-scale circuits where larger trace buffer widths are used.

Finally, iterations in pure simulation-based and hybrid approaches are interdependent and cannot be executed concurrently. In contrast, all the simulations in our approach are independent and can be conducted at the same time. Therefore, we expect that our approach would be even faster if a paralleled implementation is incorporated. This makes our approach more scalable for very large industry-level circuits by running them in parallel in a multi-processor environment.

C. Restoration Quality

Table III presents the restoration ratios of our approach compared with previous techniques [4], [8] using different ISCAS'89 benchmarks. The trace buffer sizes used in our experiment are $8 \times 4k$, $16 \times 4k$, and $32 \times 4k$. The corresponding restoration ratio for each technique is reported. The last column indicates the percentage of improvement using our approach compared with the best (shown in bold) result provided by existing approaches. The results indicate that our approach performs significantly better compared to existing approaches. Compared to [4], our fine-grained pruning reduces the chance of removing effective flip-flops prior to selection itself. Similarly, [8] incorporated simulations for only top 5% of the candidate flip-flops, which sacrifices the precision of the selection process. The improvement in restoration performance is up to 63.3% in *s38584* and 17.2% on average. In short, our approach not only produces better restoration quality, but also it is significantly faster than [4] and has a comparable runtime to [8].

TABLE III. RESTORATION RATIOS USING OUR APPROACH COMPARED WITH EXISTING SELECTION APPROACHES

Circuit	#Flip-flops	Buffer Width	Simulation-based [4]	Hybrid [8]	Learning-based	Imp. over the best
s5378	179	8	**13.41**	13.32	13.84	3.2%
		16	**7.35**	7.26	7.83	6.5%
		32	**4.47**	4.27	4.47	0.0%
s9234	228	8	13.98	**14.58**	15.33	5.1%
		16	8.3	**8.55**	8.76	2.5%
		32	4.46	4.46	4.84	8.5%
s15850	597	8	26.33	**27.38**	42.48	55.1%
		16	19.89	**20.65**	21.9	6.1%
		32	13.19	13.19	13.92	5.5%
s13207	669	8	35.52	**39.21**	47.18	20.3%
		16	20.13	**22.47**	29.00	29.1%
		32	11.25	**12.52**	15.42	23.2%
s38584	1452	8	19.73	**25.87**	29.36	13.5%
		16	28.39	**29.01**	47.37	63.3%
		32	32.45	**34.62**	43.66	26.1%
s38417	1636	8	29.23	**51.01**	52.33	2.6%
		16	17.02	**19.22**	24.25	26.2%
		32	**15.14**	13.25	16.73	10.5%
s35932	1728	8	132.00	**139.52**	157.18	12.7%
		16	67.45	**71.36**	81.00	13.5%
		32	34.63	**35.08**	44.64	27.3%

V. RELATED WORK

Limited observability of internal signals is the primary issue in post-silicon validation. Trace buffers have been widely used in post-silicon debug. The primary challenge is to compute *a priori* a small set of signals that can be traced in order to maximize reconstruction of internal states. Ko et al. [6] and Liu et al. [9] have proposed efficient signal selection algorithms based on partial restorability. Basu et al. [1] improved their methods by proposing an efficient algorithm that selects signals based on their total restorability. Shojaei *et al.* [13] proposed a metric-based signal selection technique to enhance the timing and logic visibility in the circuit. Prabhakar *et al.* [11] proposed a logic implication based trace signal selection technique that uses the primary inputs in restoration process. The use of scan chains in post-silicon debug has been extensively studied in [15], [5]. Various approaches [7], [2], [12] divided trace buffer bandwidth into two parts, one for the trace signals and the other one for the scan signals.

Chatterjee *et al.* [4] demonstrated that simulation-based signal selection is a promising approach. However, their approach requires $O(N^2)$ simulations where N is the number of flip-flops in the circuit. This makes their approach computationally expensive for large circuits. To address this issue, they propose a pre-processing phase namely pruning process, prior to running the algorithm. Basically, the pruning phase is the algorithm itself with less accuracy. The pruning phase reduces the initial candidate flip-flops set but still requires long signal selection time. In addition, it may sacrifice the signal selection quality. Li *et al.* [8] proposed a hybrid (metric-based and simulation-based) signal selection technique. However, to save selection time, [8] uses simulation for a small fraction of the signals and thereby sacrifices restoration performance.

VI. CONCLUSIONS

Post-silicon validation is an expensive phase in designing integrated circuits. Success in post-silicon validation and debug crucially depends on effective signal selection that makes effective use of the limited available observability. Thus it is critical to develop effective signal selection techniques that provide high state reconstruction and can scale to large industrial designs. Existing metric-based signal selection techniques are computationally efficient, but often yield signals with poor restorability. Simulation-based techniques, while superior in restoration quality, suffer from major computational drawbacks. We presented a learning-based signal selection approach which mitigates the computation overhead of existing simulation-based approach. Our experiments demonstrated that our fast signal selection provides up to 63.3% (17.2% on average) improvement in restoration ratio compared to existing signal selection approaches.

REFERENCES

[1] K. Basu and P. Mishra. Restoration-Aware Trace Signal Selection for Post Silicon Validation. In *TVLSI*, 2013. http://esl.cise.ufl.edu/sigsel.html

[2] K. Basu, P. Mishra, and P. Patra. Efficient combination of trace and scan signals for post silicon validation and debug. In *ITC*, 2011.

[3] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *TIST*, 2011.

[4] D. Chatterjee et al. Simulation-based signal selection for state restoration in silicon debug. In *ICCAD*, 2011.

[5] R. Datta et al. Delay fault testing and silicon debug using scan chains. In *ETS*, 2004.

[6] H. F. Ko and N. Nicolici. Algorithms for state restoration and trace-signal selection for data acquisition in silicon debug. *TCAD*, 2009.

[7] H. F. Ko and N. Nicolici. Combining scan and trace buffers for enhancing real-time observability in post-silicon debugging. In *ETS*, 2010.

[8] M. Li and A. Davoodi. A hybrid approach for fast and accurate trace signal selection for post-silicon debug. In *DATE*, 2013.

[9] X. Liu and Q. Xu. Trace signal selection for visibility enhancement in post-silicon validation. In *DATE*, 2009.

[10] A. Nahir et al. Bridging pre-silicon verification and post-silicon validation. In *DAC*, 2010.

[11] S. Prabhakar and M. Hsiao. Using non-trivial logic implications for trace buffer-based silicon debug. In *ATS*, 2009. 2009.

[12] K. Rahmani and P. Mishra. Efficient signal selection using fine-grained combination of scan and trace buffers. *VLSI*, 2013.

[13] H. Shojaei and A. Davoodi. Trace signal selection to enhance timing and logic visibility in post-silicon validation. In *ICCAD*, 2010.

[14] Stephen Williams. Icarus Verilog. http://iverilog.icarus.com/.

[15] G. Van Rootselaar and B. Vermeulen. Silicon debug: scan chains alone are not enough. In *ITC*, 1999.

[16] S. Yerramilli. Addressing post-silicon validation challenge: Leverage validation and test synergy. In *ITC*, 2006.