Post-silicon Trace Signal Selection Using Machine Learning Techniques

Kamran Rahmani, Sandip Ray, *Senior Member, IEEE*, and Prabhat Mishra, *Senior Member, IEEE*

***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. The basic idea is to train a machine learning framework with a few simulation runs and utilize its effective prediction capability (instead of expensive simulation) to identify beneficial trace signals. Specifically, our approach uses (1) bounded mock simulations to generate training vectors for the machine learning technique, and (2) a compound search-space exploration approach to identify the most profitable signals. Experimental results indicate that our approach can improve restorability by up to 143.1% (29.2% on average) while maintaining or improving runtime compared to state-of-the art signal selection techniques.**

***Keywords*—Post-silicon debug, simulation, feature selection, supervised learning**

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 [1], [2]. A fundamental constraint in post-silicon validation is *limited observability*: 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.

Post-silicon trace signal selection in current industrial practice is primarily manual and guided by the designer's experience and insight, often with no objective techniques for qualifying the observability quality of a selected set of signals. Critical observability holes manifest themselves only during silicon debug, typically in the form of inadequacy of the set of traced signals for diagnosis or localization of a bug. However, this is too late for redesign of the debug infrastructure or selection of new trace signals (with associated routing hardware), which would require a significant hardware change. Thus one has to contend with costly escapes, complex workarounds, and in many cases, more silicon respins.

There has been significant recent research to address the above issue by developing algorithms for signal selection through automatic analysis of pre-silicon (RTL or gate-level) designs. The focus is to identify 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. A common class of signal selection techniques involves defining a metric based on the design structure, which is then used in a (typically greedy) selection process to evaluate a candidate signal set [3], [4], [5], [6]. These approaches are fast but provide a low value of state restoration. Recent work on simulation-based signal selection [7] 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.

The key contribution of this paper is a novel signal selection technique that retains (and improves upon) the restoration quality of simulation-based signal selection while achieving faster or comparable selection time complexity. Our approach is characterized by two key components: (1) for the first time to our knowledge, a machine learning technique is applied to model the restoration strength of the signals; and (2) the raw machine learning algorithm has been augmented with a compound back-end selection technique to find the most profitable set of signals using the circuit model. The basic idea is to run only a small number of simulations to train the machine learning framework. Subsequently, our approach will utilize the predication capability of the machine learning replacing the need for costly simulation runs. Our proposed approach address three important challenges in using machine learning for signal selection. First, we have to identify a machine learning algorithm that is suitable for signal selection. We also need to determine the minimum number (as well as specific types) of training vectors (simulation runs) that will provide an effective trade off between the cost (time) and prediction accuracy. Finally, we need to develop a signal selection algorithm that utilizes the best use of our model to select the best set of signals. Our experiments demonstrate that our approach can improve restoration quality by 143.1% (29.2% on average).

The remainder of the paper is organized as follows. Section II

The authors are affiliated to University of Florida, Gainesville FL 32611, USA, and NXP Semiconductors, Austin, TX 78735, USA (email: kamran@cise.ufl.edu, sandip.ray@nxp.com, and prabhat@cise.ufl.edu.)

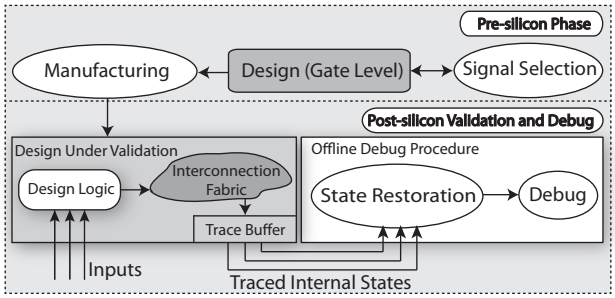


Fig. 1. Simplified overview of post-silicon validation flow and role of trace signal selection. Signals selected through pre-silicon analysis are routed to trace buffer from which signal states are restored offline to assist in debug.

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

Fig. 1 provides a very simplified overview of the 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 [3], [4], [5], [7] 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. Figure 2 illustrates forward and backward restoration for common logical gates. Forward restoration entails reconstructing the output from the input. For example, if one of the inputs of the *AND* gate is '0', the output value would be '0'. If all the inputs are known, the unknown output can be definitely

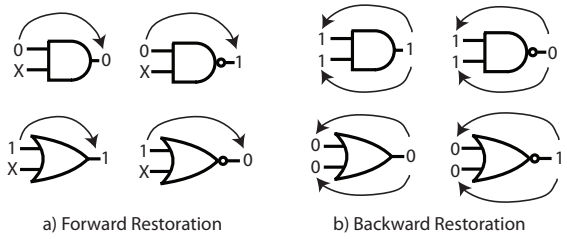


Fig. 2. Basic restoration rules for common logic gates. (a) Forward restoration: knowledge of inputs can reconstruct the output. (b) Backward restoration: knowing the output can restore the unknown inputs.

TABLE I. Illustration of restored signals for the simple circuit shown in Figure 3. 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

determined. On the other hand, backward restoration entails reconstructing the inputs from the output. For example, if the output of the *AND* gate is '1', both of the inputs would be '1'. Backward restoration might fail in certain scenarios. For example, if the output of a 2-input *AND* gate is '0' and one of the input has known value of '0', the other input still cannot be inferred. During signal value reconstruction, forward and backward restoration are repeated for all the gates in the circuit back and forth until no more states can be restored. Restoration Ratio (RR), defined below, is a popular metric for measuring the quality of a set of selected trace signals.

$$\text{Restoration Ratio} = \frac{\text{\# of traced signals + \# of restored signals}}{\text{\# of traced signals}}$$

Consider the simple circuit shown in Figure 3. Suppose that the width of the trace buffer is 2 (*i.e.*, two signals can be traced at each 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 then shows the signal states that can be restored: 48 signal values can be restored while 16 are traced, yielding a restoration ratio of 3.0.

C. Simulation-based Signal Selection

Chatterjee *et al.* [7] developed simulation-based signal selection, *i.e.*, the use of mock simulations to identify trace signals. They demonstrated that simulation-based signal selection provided better restoration ratio than older approaches of optimizing metrics based on circuit structure. This approach is tractable for small designs for two key reasons. First, there is negligible overall variation in restoration quality over different random input vectors. Second, restoration ratio is insensitive to

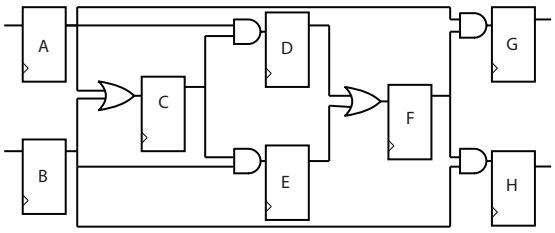


Fig. 3. A simple circuit to illustrate restorability.

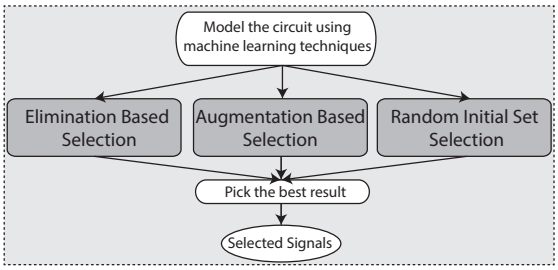


Fig. 4. Overview of our approach and its relation to existing simulation based approaches [7], [8]. We use machine learning techniques to replace mock simulations with fast predictions. This allows us to run both elmination-based and augmentation-based algorithms as well as our newly proposed random initial set selection technique. Running all these techniques together expands our search space and increase the chance of finding a better global solution.

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 whose removal will have the least impact on restoration ratio. 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, then O(N^2) simulations are required to reach the final selection 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 key problems 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. This happens when each of the signals removed in a pruning iteration does not contribute significantly in the restoration ratio but their combination might do. 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.

To mitigate this limitation, Li *et al.* [8] proposed a hybrid approach which incorporated a combination of metric-based and simulation-based signal selection approaches. In each step of their algorithm, the most beneficial flip-flop is added to the trace signals candidate set, instead of removing the least beneficial one. However, using less simulation to find the top candidates sacrifices the restoration performance.

Another drawback of these approaches is that they both start at a fixed starting point in the search space (all flip-flops for Chatterjee *et al.*, no flip-flops in Li *et al.*) and iterate until they reach a local maximum; the local maximum may not be the globally optimum selection. Exploring more combinations in the search space will be computationally expensive as it needs more mock simulations/restoration processes.

The learning-based signal selection approach we propose in this paper addresses these issues. We first model the circuit using machine learning techniques and bounded simulations as training vectors. After that, our approach replaces mock simulations with faster prediction. This enables efficient exploration of a larger search space and thereby produces comparable or better results with a faster or comparable time complexity. Although there is no gaurantee that our approach is able to find the global maximum, however, due to fast estimations instead of simulations, it can find several local maximums by starting from different random initial sets. This allows us to explore bigger search space and find a better solution.

III. Learning-based Signal Selection

Fig. 4 shows the overview of our approach and its relation to existing simulation based approaches [7], [8]. Both elimination-based [7] and hybrid augmentation-based [8] approaches use mock simulations to evaluate the quality of candidate signals. However, using mock simulations is expensive and it can limit the search space exploration. Our approach makes use of machine learning techniques to meliorate the cost of mock simulations. In particular, we first model the circuit using machine learning techniques and bounded mock simulations. After that, the model can be used to explore a bigger search space as we are replacing mock simulations with fast predictions. This allows us to run both elmination-based and augmentation-based algorithms as well as our newly proposed random initial set selection technique. Running all these techniques together expands our search space and increase the chance of finding a better global solution.

In order to reduce the number of mock simulations and also increasing the accuracy in modeling, we propose a two-step signal selection approach using supervised learning: in the first (pre-processing) step, a small number of mock simulations is used as a training set to build a linear model of the circuit and eliminate non beneficial signals; in the second (selection) step, we use a non-linear and more accurate prediction model to find the final selected signals using different selection techniques.

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. 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.

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

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 [7] 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 and Motivation

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 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 regression supervised machine learning techniques to estimate $r_m(v)$. Supervised learning algorithm is inferring a function from training data. Training data is a set of input vector and the desired output which is number of restored states in our case. Once the model is trained using training examples, it can be used to predict the output value of any new input vector. In our case, the training vectors come from restoration estimates obtained from mock simulations for given feature vectors. If the training set is selected carefully

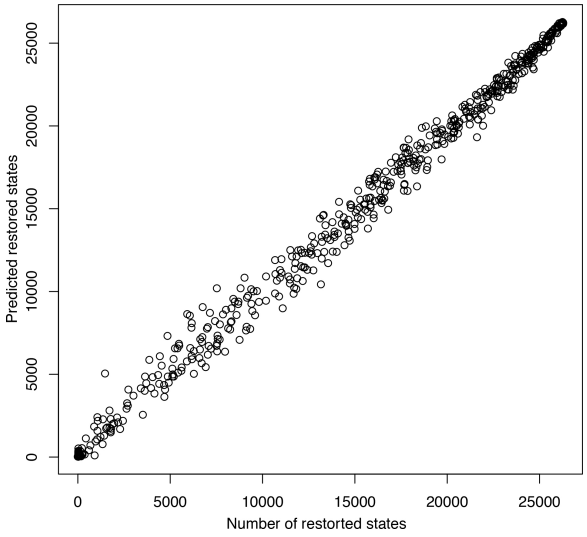


Fig. 5. Real values of $r_m(v)$ versus predicted values for different random vectors in s38417 benchmark using cubist model from *caret* package in *R* [9]. Each random vector represents a set of randomly selected trace signals and is showed as a circle in the graph.

to be effective and 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, 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 having too few vectors can lead to underfitting, and too many vectors can lead to overfitting. Underfitting happens when the model is too simplistic (generlized), resulting in a low accuracy in both training and new data. On the other hand, overfitting happens when the model is too specific to the training data, resulting in a high accuracy in trianing data and low accuracy in new data. Thus both overfitting and underfitting lead to high prediction error for unknown input vectors. Furthermore, the class of regression model being used is another important factor. There are many regression models that each of them are a good fit for a prticular application or domain. For example, linear fitting may not be a good choice for modeling the complicated nonlinear relationships between the flip-flops of the circuit.

Fig. 5 shows the relationship between the real value of $r_m(v)$ (calculated using simulation) and the predicted value for different random vectors where $m = 64$ in *s38417* benchmark. Each random vector represents a set of randomly selected trace signals and is represented by a circle in the graph. The cubist model (a rule based regression model) from *caret* package in

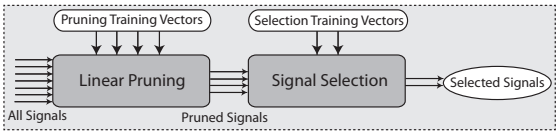


Fig. 6. Proposed signal selection process. A quick linear model is used to eliminate most of the non-beneficial flip-flops. A more accurate non-linear model is used to select w flip-flops out of the remaining flip-flops where w is the trace buffer width.

R [9] is used for modeling the circuit in this experiment. It should be noted that the vectors used for training the model were all different from the one used for this experiment. It can be observed that if the right model and training vectors are used, there is a high accuracy in prediction and strong correlation between predicted and real values. This permits the use of predicted values instead of the real ones without significant loss in quality. The same high accuracy was observed for other benchmarks as well which is discussed in Section IV in details.

C. Signal Selection Algorithm

In order to increase the accuracy of the prediction while simultaneously reducing the runtime of modeling/prediction in large circuits, we propose a two-step modeling scheme. Fig. 6 illustrates the framework. In the first step, a linear model is applied to eliminate less important flip-flops and 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 for selecting the final set of signals.

Since we are replacing the expensive mock simulation/restoration with prediction, we can explore a larger search space compared to existing approaches [7], [8]. Fig. 7 illustrates the search space exploration using different techniques. The horizantal axis is the number of signals being traced and the vertical axis is the number of restored states. The circle is an initial state of the selection approach and the square is the end state. Note that the elimination-based technique [7] (shown in green) starts with all the flip-flops and stops when number of remaining flip-flops is equal to the trace buffer width w; the hybrid augmentation approach [8] (shown in red) starts with no flip-flops and stops when w signals are selected. It can be observed that these are just two ways of exploring the search space and they will end up in a local maximum. We propose a new way of exploration - random initial set - which can help to explore significantly larger search-space. In our approach, we start with a random set of w signals and in each iteration we remove the least beneficial flip-flop and add the most beneficial flip-flop to the candidates set. This process terminates once it is not beneficial to do this removal-addition anymore. This process is shown by blue in the Fig. 7. It can be observed that running all these techniques at the same time explores more of

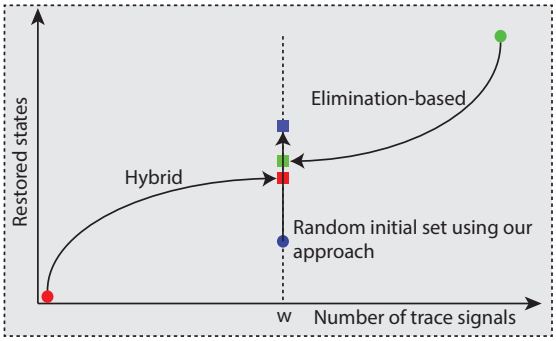


Fig. 7. Different signal selection techniques for exploring the search space. The circle is an initial state of the selection approach and the square is the end state. The elimination-based [7] is shown in green, the hybrid [8] in red, and our proposed machine learning based approach is presented in blue.

search space and increases the chance of finding a better local maximum which yields to a better set of selected signals.

Algorithm 1 outlines the major steps involved in our proposed learning-based signal selection technique. First, we start by pruning the set of candidate signals. In this step, most of the non-beneficial flip-flops (in term of restorability effectiveness) are identified and removed using a linear model. Next, an accurate model of the $r_m(v)$ is created using the remaining signals. Once the final model is created, we run both elimination-based and augmentation-based techniques to generate two sets of final candidate signals and choose the one with better result. Finally, we run our proposed random initial set technique r times ($r = 10$ in our experiments [1]) and return the best result of these runs, elimination-based, and augmentation-based techniques as the selected signals. It should be noted that each run of random initial set algorithm starts from a completely random initial set. Combining all the techniques along with multiple run of our proposed random initial select approach can increase the explored search space which will boost the final result. Next, we will explain each step of our approach in more details.

1) 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). For our explaination here, we use support vector regression with linear kernel which is the simplest form of this well-known model. However, any other linear regression technique can be used as well. 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:

[1] **In our experiments, we did not see any new end state that further expands the search space for r bigger than 10. In addition, this is a tunable parameter of our approach and depending on the time/performance constraints can be tuned during the signal selection.**

Algorithm 1 Learning-based Signal Selection

procedure LEARNINGSELECTION($circuit, m, t_{pruning}, p, r, t_{selection}, w, candidateModels$)
  Prune the least useful signals by calling LinearPruning
  Create the final model of the circuit by calling SelectFinalModel
  Set v as the selected signals by calling EliminationBased
  $maxRestorability = \hat{r}_m(v)$
  Set v_{new} as the selected signals by calling AugmentationBased
  if $\hat{r}_m(v_{new}) > \hat{r}_m(v)$ **then**
    $v = v_{new}$)
    $maxRestorability = \hat{r}_m(v)$
  end if
  for $i = 1; i <= r; i++$ **do**
    Set v_{new} as the selected signals by calling RandomInitialSet
    if $\hat{r}_m(v_{new}) > \hat{r}_m(v)$ **then**
      $v = v_{new}$)
      $maxRestorability = \hat{r}_m(v)$
    end if
  end for
  return v
end procedure

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

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 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 as those will get a smaller coefficient in the $\hat{w}$ vector. 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. The 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. The output of the process is the preserved flip-flops set S.

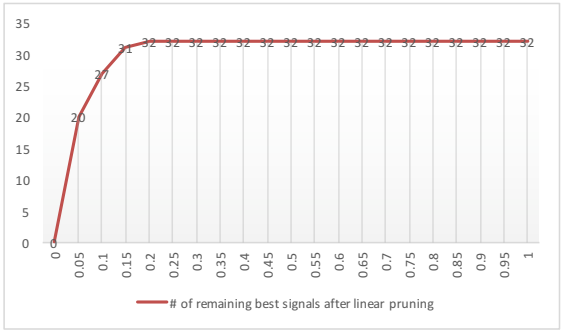


Fig. 8. **Number of profitable signals remaining after linear pruning (out of 32 signals selected when linear pruning is not applied or $p = 1$) for different values of p in *S38417* benchmark.**

The linear model has a higher prediction error; however, we compensate for this by selecting a bigger set (compared to the buffer width) of the top signals in the pruning phase. The more accurate non-linear model in the second step enables us to pick the most profitable signals from this set with a more accurate and fine-grained selection. To illustrate the fact that top signals are not removed in the linear pruning, Figure 8 shows how many of the 32 top signals are kept when we keep reducing the p value for benchmark *S38417*. As we can see that even for $p = 0.05$, we have most of the profitable signals left. In our experiments, we set $p = 0.15$.

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

2) Generating Training Vectors: Algorithm 3 outlines the pseudo-code for training vector generation used in both pruning and final model. 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, in order to calculate the corresponding $r_m(v)$, we first run a mock simulation over m cycles assuming that the signals in training vector are being traced. We then apply forward/backward restoration techniques decribed in Section II-B to get the total number of restored states. 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 $trainingSet$ and corresponding restorability R are returned as the output of algorithm.

Algorithm 3 Training Vector Generation

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

3) Final Model 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 this step is reduced by $1-p$, where p is the pruning factor. There are several nonlinear models available to use, each of which can be a good fit in a specific domain. Mean Prediction Error (MPE) can be used to measure the quality of a model on a test vector set of size n, is defined as below.

$$MeanPredictionError = 1/n * \sum_{k=1}^{n} |\hat{r}_m(v_k) - r_m(v_k)| \quad (5)$$

Algorithm 4 outlines the final model selection process after the pruning. First, a set of $t_{selection}$ training vectors is generated for final model training. In order to find the best non-linear model in the *candidateModels* set, we do a quick training followed by an MPE calculation on a small set of vectors randomly selected from the bigger training vectors set. It should be noted that we do not use the same set of vectors for quick training and testing (MPE calculation); this makes our model selection unbiased and yields a better result for new input vectors. After choosing the best model with minimum MPE, we retrain it with all the training vectors and return it as the result.

Algorithm 4 Final Model Selection Algorithm

procedure SELECTFINALMODEL(*circuit*, m, $t_{selection}$, w, *candidateModels*)
  trainVectors = GenerateVectors(*circuit*, m, $t_{selection}$)
  quickTrainVectors =10% of trainVectors randomly selected
  quickTestVectors =10% of trainVectors randomly selected, exclusive with quickTrainVectors
  for each model *model* in *candidateModels* **do**
    Model $\hat{r}_m(v)$ with pruned features using *model* and *quickTrainVectors*
    Calculate MPE for $\hat{r}_m(v)$ on *quickTestVectors*
  end for
  bestModel = model with minimum MPE
  result = Model $\hat{r}_m(v)$ with pruned features using *bestModel* and *trainVectors*
  return *result*
end procedure

4) Elimination-based Signal Selection: Now that we have the final model of the circuit, we can use it to select the final set of signals. Algorithm 5 outlines the steps involved in selecting the signals using the circuit model and elimination-based technique described by Chatterjee *et al.* [7]. 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. It should be noted that our approach is not identical to Chatterjee *et al.*'s. Because of computational limitation, they use a coarse-grained prunning pre-processing to

remove most of the signals from the candidates set which can degrade the performance of the final set of signals. Our approach does not have this limitation as we use quick predictions instead of expensive simulation/restorations.

Algorithm 5 Elimination-based Signal Selection

procedure ELIMINATIONBASED($circuit, \hat{r}_m(v), w$)
  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
    end for
    $v[maxIndex] = 0$
    $remainedSignals = remainedSignals - 1$
  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

5) Augmentation-based Signal Selection: Algorithm 6 outlines the steps involved in selecting the signals using the circuit model and augmentation-based technique similar to the approach described by Li *et al.* [8]. In this technique, instead of removing the least profitable flip-flop in each iteration, we add the most beneficial one and continue the process until total number of w flip-flops are selected. The set of selected signals S is returned as the algorithm output. Our approach is slightly different than Li *et al.*, as they use simulation only for top 5% of the candidates, which can degrade the selection performance of their approach.

6) Random Initial Set Signal Selection: Algorithm 7 outlines the steps involved in our proposed random initial set selection technique. First we start with a random set of w selected signals. In each iteration, we remove the least beneficial signal and add the most profitiable one. We continue this process until removing a signal and adding back another one does not improve the predicted restoration in $\hat{r}_m(v)$. The random initial set can expand our search space and helps us finding a better global maximum point.

Algorithm 6 Augmentation-based Signal Selection

procedure AUGMENTATIONBASED($circuit, \hat{r}_m(v), w$)
  Create selected signals set S
  Create initial vector of $v =< 0, 0, ..., 0 >, |v| = N \times p$
  for $selected = 1; selected <= w; selected++$ **do**
    $maxRestorability = -\infty$
    $maxIndex = -1$
    for $i = 1; i <= N \times p; i++$ **do**
      if $v[i] = 0$ **then**
        $v[i] = 1$
        if $\hat{r}_m(v) > maxRestorability$ **then**
          $maxRestorability = \hat{r}_m(v)$
          $maxIndex = i$
        end if
        $v[i] = 0$
      end if
    end for
    $v[maxIndex] = 1$
  end for
  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

Algorithm 7 Random Initial Set Signal Selection

procedure RANDOMINITIALSET($circuit, \hat{r}_m(v), w$)
  Create selected signals set S
  Create initial vector of $v =< 0, 0, ..., 0 >, |v| = N \times p$
  Randomly set w elements of v to 1
  Set v_{new} as v
  do
    Set v as v_{new}
    Find the signal in v_{new} which its removal has the minimum effect on $\hat{r}_m(v_{new})$ and set it to 0
    Find the signal in v_{new} which its addition has the maximum effect on $\hat{r}_m(v_{new})$ and set it to 1
  while $\hat{r}_m(v_{new}) > \hat{r}_m(v)$
  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

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* [10]. We used the set of largest circuits in ISCAS'89 as has been studied by previous works. We used *caret* package in *R* [9] as the modeling/prediction tool. In addition, we used *10-fold cross validation* and *normalization and scaling* while training our models.

In our experiments, we did not use the reported numbers of Li *et al.* [8] and Chatterjee *et al.* [7], since they used modified versions of ISCAS89 benchmarks (with some specific optimizations). To perform a fair comparison, we tried to obtain the executables of [8] and [7]. Li et al. [8] provided us with their signal selection framework and we used it for the selection process. Unfortunately we were not able to get the implementation of Chatterjee *et al.* [7] and we used our own implementation of their approach in this revision, but used the same parameters $c = 64$ and $PT = 95\%$ as they reported. We also used $m = 32$, $p = 0.15$, $r = 6$, $t_{pruning} = 3 \times N$, and $t_{selection} = 0.75 \times N$ 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. Model Selection

In order to choose the best non-linear after pruning model for the benchmarks, we explored several models available in *caret* package [9]. Fig. 10 illustrates Mean Prediction Errors of different models on the set of our benchmarks caluculated using Equation 5. It can be observed that *cubist* is the best model in our experiments with minimum prediction error. This can be also clearly seen in Fig. 9 which illustrates the real versus predicted restoration states for different models in *S38584* benchmark. It should be noted that the MPE is bigger for larger benchmarks in *cubist*; however, it still maintains the relative relationship between the restoration values. In other words, the percentage of error ($|Predicted - Actual|/Actual$) will not grow linearly as the actual restoration absolute value grows in larger benchmarks. For this experiment, we used 80% of our training vector for actual training and the other 20% for the testing. This can prevent us from biasing while training the models. We selected *cubist* model as our non-linear model for the rest of our experiments. In fact, the predicted values in *cubist* match the real values in most of the cases. This enables us to have high quality signal selection without any further real simulation.

C. Restoration Quality

Table II presents the restoration ratios of our approach compared with previous techniques [7], [8] using different ISCAS'89 benchmarks. The trace buffer sizes used in our experiment are $8 \times 4k$, $16 \times 4k$, and $32 \times 4k$. We used the forward and backward restoration technique described in Section II where we conduct forward and backward restoration repeadetly until there is no more restoration possible. The corresponding restoration ratio for each technique is reported. The letters in parentheses for learning-based numbers show the algorithm that yielded the best result for our run. *E* stands for elimination based, *A* for augmentation-based, and *R* for random-initial set. 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 Chatterjee *et al.* [7], our fine-grained pruning reduces the chance of removing effective flip-flops prior to selection itself. Similarly, Li *et al.* [8] incorporated simulations for only top 5% of the candidate flip-flops, which sacrifices the precision of the selection process. In addition, replacing mock simulations with fast predictions allows us to run all the selection techniques (elimination based, augmentation-based, and random-initial set) at the same time and pick the best one as the final result. It can be observed that the best approach depends on the benchmark structure and also the buffer width. For example, elimination-based yields the best result for *s9234* benchmark with buffer width of 8. However, random-initial set yields the best result for the same benchmark and buffer widths of 16 and 32. Running all these techniques together increases the chance of having a better local maxima and consequently having a better restoration ratio. It can also be observed that our newly introduced random initial set selection technique yielded the best result in several benchmarks. The improvement in restoration peformance is up to 143.1% in *s38584* and 29.2% on average. In summary, our approach not only produces better restoration quality, but also it is significantly faster than [7] and has a comparable runtime to [8].

Table III presents the runtime of our approach compared with previous techniques [7], [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 [7]. 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.

D. Selection Time, Complexity, and Scalability

Simulation of large industrial designs incurs high cost in running time. Indeed, simulation time is the primary bottleneck

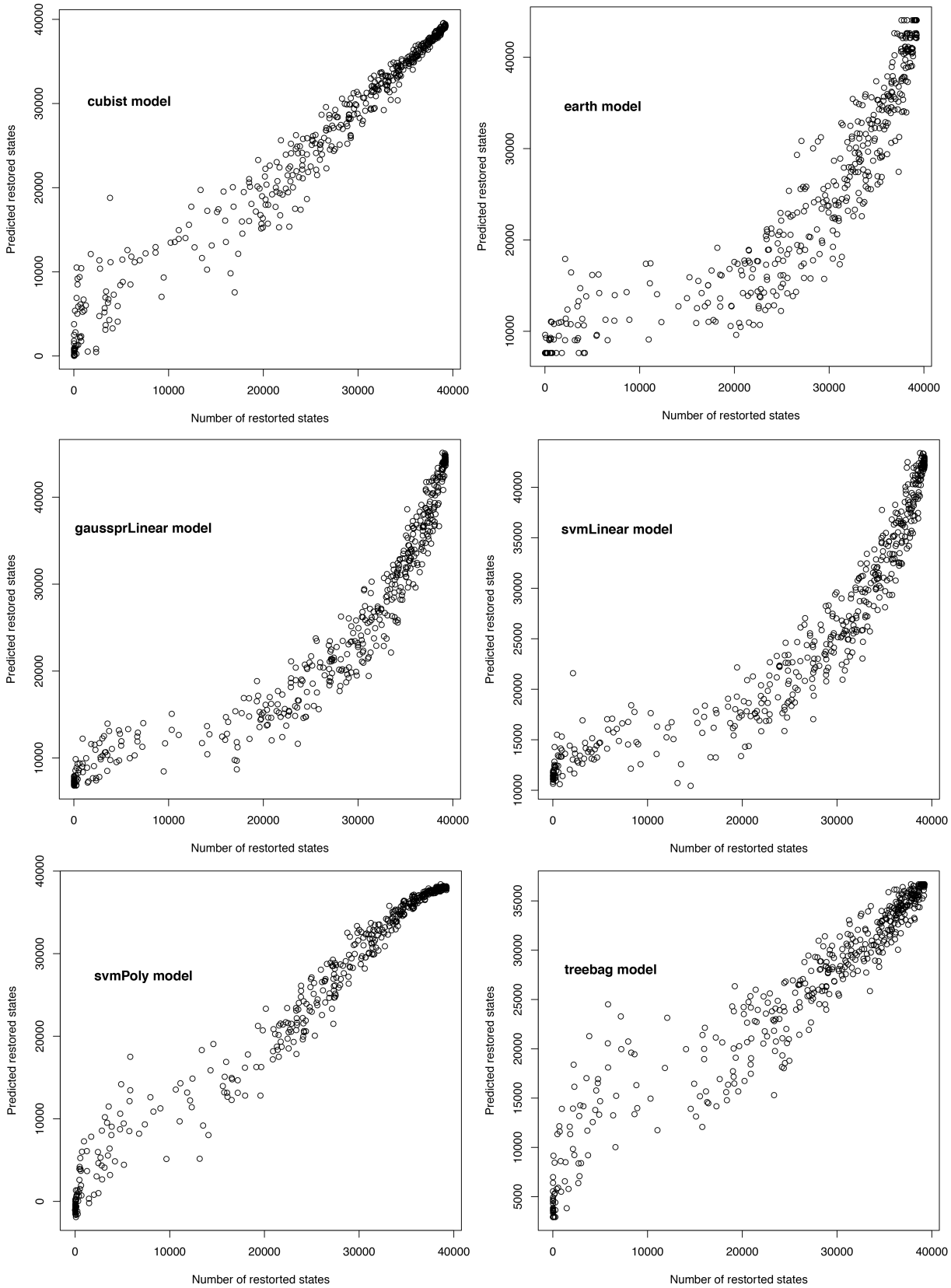
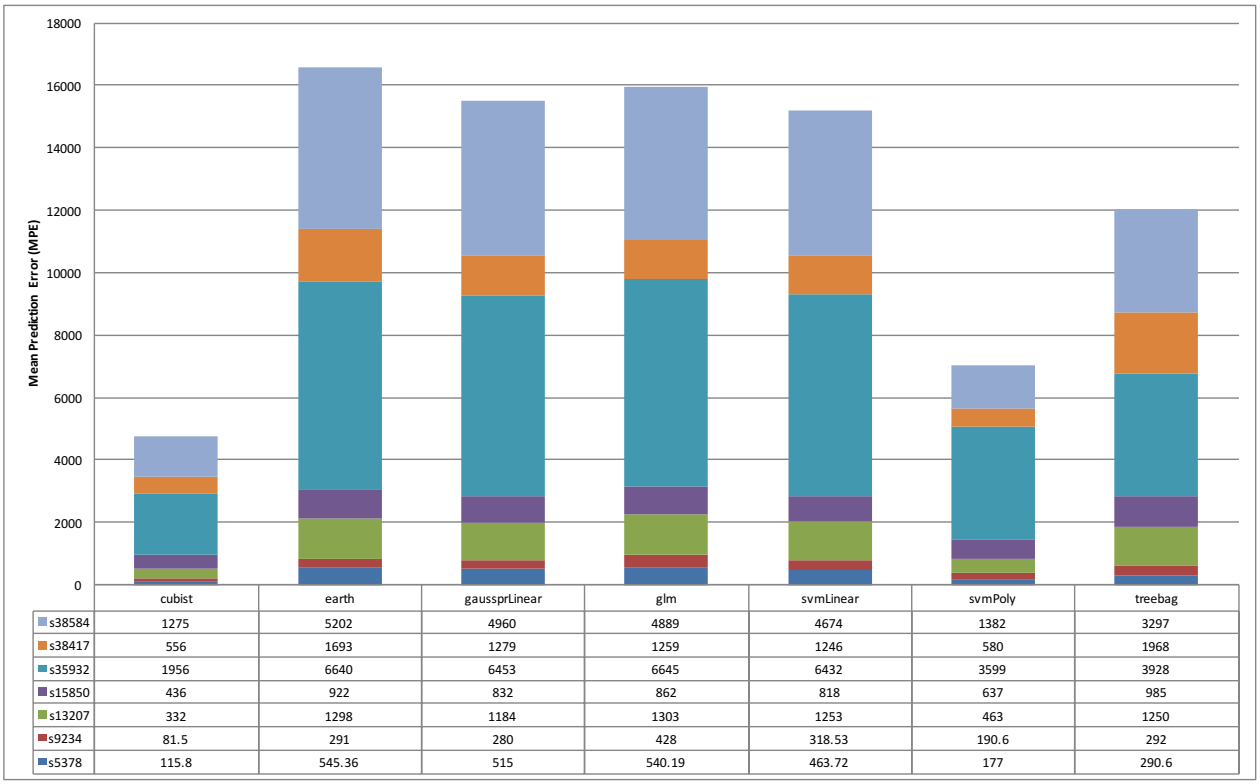


Fig. 9. Real versus predicted restoration states for different models in *S38584* benchmark. Each random vector represents a set of randomly selected trace signals and is showed as a circle in the graph.



	cubist	earth	gausssprLinear	glm	svmLinear	svmPoly	treebag
s38584	1275	5202	4960	4889	4674	1382	3297
s38417	556	1693	1279	1259	1246	580	1968
s35932	1956	6640	6453	6645	6432	3599	3928
s15850	436	922	832	862	818	637	985
s13207	332	1298	1184	1303	1253	463	1250
s9234	81.5	291	280	428	318.53	190.6	292
s5378	115.8	545.36	515	540.19	463.72	177	290.6

Fig. 10. **Mean prediction errors (MPE) of different models on the set of our benchmarks.**

TABLE II. RESTORATION RATIOS USING OUR APPROACH COMPARED WITH EXISTING SELECTION APPROACHES. THE LETTERS IN PARENTHESES FOR LEARNING-BASED NUMBERS SHOW THE ALGORITHM THAT YIELDED THE BEST RESULT. *E* STANDS FOR ELIMINATION BASED, *A* FOR AUGMENTATION-BASED, AND *R* FOR RANDOM-INITIAL SET

Circuit	#Flip-flops	Buffer Width	Simulation-based [7]	Hybrid [8]	Learning-based	Imp. over the best
s5378	179	8	13.41	**14.35**	14.20 (E)	-1.0%
		16	7.35	**8.36**	8.40 (E)	0.5%
		32	4.47	**4.99**	4.93 (R)	-1.2%
s9234	228	8	**13.98**	9.25	15.33 (E)	9.7%
		16	**8.30**	6.13	8.76 (R)	5.5%
		32	**4.46**	4.38	4.84 (R)	8.5%
s15850	597	8	**26.33**	21.90	44.03 (E)	67.2%
		16	**19.89**	14.78	23.13 (E)	16.3%
		32	**13.19**	10.88	13.92 (A)	5.5%
s13207	669	8	**35.52**	33.60	47.18 (E)	32.8%
		16	20.13	**23.22**	29.00 (A)	24.9%
		32	11.25	**13.64**	15.42 (R)	13.0%
s38584	1452	8	19.73	**27.00**	54.25 (A)	100.1%
		16	**28.39**	13.97	69.03 (R)	143.1%
		32	**32.45**	7.50	43.66 (R)	34.5%
s38417	1636	8	29.23	**37.71**	52.33 (E)	38.8%
		16	17.02	**23.80**	27.12 (R)	13.94%
		32	**15.14**	11.83	16.73 (R)	10.5%
s35932	1728	8	132.00	**144.00**	186.8 (E)	29.7%
		16	67.45	**72.00**	93.60 (E)	30.0%
		32	34.63	**36.00**	46.98 (A)	30.5%

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 and 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. Based on the selected variables in our experiements, the total number of mock simulations in our approach is $3.75 \times N$, which is much less than $\Omega(N^2/d_{step})$ reported in previous work [7], where $d_{step} = 50$ in their experiments. On the other hand, the hybrid approach [8], uses simulation/restoration 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. Once the parameters are fixed, the asymptotic complexity of our approach is $\theta(N)$ and $\theta(wN)$ for [8], with potentially different constant coefficients.

To compare the runtimes in practice, we used a Octa-Core AMD Opteron 6378 (1400 MHz) machine with 188GB of memory for all the experiments. The runtime is calculated as the summation of required time for generating training vectors (simulations), modeling, and signal selection process itself. Table III presents the runtime of our approach compared with previous

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

Circuit	#Flip-flops	Buffer Width	Simulation-based [7]	Hybrid [8]	Learning-based
s5378	179	8	00:01:53	00:00:08	00:01:46
		16	00:01:52	00:00:10	00:01:52
		32	00:01:48	00:00:16	00:02:09
s9234	228	8	00:08:52	00:00:32	00:00:10
		16	00:08:43	00:00:40	00:00:10
		32	00:08:10	00:00:50	00:00:10
s15850	597	8	03:44:12	00:05:20	00:04:20
		16	03:44:04	00:06:00	00:04:35
		32	03:43:39	00:06:36	00:05:04
s13207	669	8	01:21:41	00:01:36	00:03:45
		16	01:21:35	00:02:00	00:04:01
		32	01:21:13	00:02:40	00:04:12
s38584	1452	8	28:43:02	00:05:28	00:16:52
		16	28:42:16	00:06:06	00:17:09
		32	28:38:59	00:09:02	00:17:35
s38417	1636	8	196:51:50	00:22:42	00:20:23
		16	196:50:44	00:33:04	00:21:07
		32	196:48:27	00:34:28	00:23:55
s35932	1728	8	11:39:36	00:04:28	00:16:49
		16	11:39:09	00:05:56	00:17:33
		32	11:38:01	00:08:38	00:18:21

techniques [7], [8] using different ISCAS'89 benchmarks[2]. The reported runtime format is 'hour:minute:second'. As expected, our approach is significantly faster than pure simulation-based approach presented in [7]. Moreover, 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, the selection process can be done in negligible time using simple calculations. This makes our approach runtime independent of the trace buffer width. In contrast, for [8] the runtime grows linearly with the buffer width.

Finally, iterations in pure simulation-based and hybrid approaches are interdependent and cannot be executed concurrently. In contrast, all the simulations needed for generating the training vectors in our approach are independent and can be conducted at the same time using industry techniques like MapReduce. In addition, industry level scalable machine learning modelings are available, *e.g.*, Amazon Machine Learning framework. Therefore, we expect that our approach would be faster if a parallel implementation is incorporated (for example, using Amazon Machine Learning and Amazon EC2).

V. RELATED WORK

Limited observability of internal signals is the primary issue in post-silicon validation. There has been significant work on on-chip instrumentation to ameliorate post-silicon observability [13], [14]. Trace buffers provide one of the commonest form of on-chip instrumentations. The primary challenge with trace buffers 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. [5] and Liu et al. [3] have proposed efficient signal selection algorithms based on partial restorability. Basu et al. [4] improved their methods by proposing an efficient algorithm that selects signals based on their total restorability. Shojaei *et al.* [6] proposed a metric-based signal selection technique to enhance the timing and logic visibility in the circuit. Prabhakar *et al.* [15] 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 [16], [17]. Various approaches [18], [19], [20] divided trace buffer bandwidth into two parts, one for the trace signals and the other one for the scan signals. This enabled decoupling scan-based and trace-based observabilities, and signal selection could be studied based on constraints provided by the respective architectures.

Chatterjee *et al.* [7] 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. Our work is the first paper that utilizes machine learning techniques for signal selection.

Preliminary versions of this work appeared in conference proceedings [11], [21]. This paper extends those approaches, in particular by establishing machine learning as a generic front-end for interfacing with several back-end signal selection procedures and by significantly improving the algorithms involved.

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 143.1% (29.2% on average) improvement in restoration ratio compared to existing signal selection approaches.

[2]The numbers reported here for our approach are different than the conference version [11] as we ran new experiments with the new proposed approach and regression models presented in this paper. In addition, instead of $LIBSVM$ [12] in [11], we used R for running our experiments which provides an easy way to run modeling and predictions in parallel. The numbers for the Hybrid approach [8] are also different. in the conference version we used our implementation of their approach; for this we used the multi-threaded executable we received from authors. The numbers for simulation-based approach [7] are the same since we were not able to get their implementation and used our implementation of their approach in both papers.

REFERENCES

[1] A. Nahir, A. Ziv, R. Galivanche, A. J. Hu, M. Abramovici, A. Camilleri, B. Bentley, H. Foster, V. Bertacco, and S. Kapoor, "Bridging pre-silicon verification and post-silicon validation," in *DAC*, 2010, pp. 94–95.

[2] S. Yerramilli, "Addressing post-silicon validation challenge: Leverage validation and test synergy," in *Keynote, Intl.Test Conf*, 2006.

[3] X. Liu and Q. Xu, "Trace signal selection for visibility enhancement in post-silicon validation," in *Design, Automation Test in Europe Conference Exhibition, 2009. DATE '09.*, april 2009, pp. 1338 –1343.

[4] K. Basu and P. Mishra, "Efficient trace signal selection for post silicon validation and debug," in *VLSI Design (VLSI Design), 2011 24th International Conference on*, jan. 2011, pp. 352 –357.

[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*, vol. 28, no. 2, pp. 285 –297, feb. 2009.

[6] H. Shojaei and A. Davoodi, "Trace signal selection to enhance timing and logic visibility in post-silicon validation," in *Proceedings of the International Conference on Computer-Aided Design*, ser. ICCAD '10. Piscataway, NJ, USA: IEEE Press, 2010, pp. 168–172. [Online]. Available: http://dl.acm.org/citation.cfm?id=2133429.2133462

[7] 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*, nov. 2011, pp. 595 –601.

[8] M. Li and A. Davoodi, "A hybrid approach for fast and accurate trace signal selection for post-silicon debug," in *Design, Automation, and Test (DATE)*, 2013, pp. 485–490.

[9] Max Kuhn, "The caret Package," http://topepo.github.io/caret/index.html.

[10] Stephen Williams, "Icarus Verilog," http://iverilog.icarus.com/.

[11] K. Rahmani, P. Mishra, and S. Ray, "Scalable trace signal selection using machine learning," in *Computer Design (ICCD), 2013 IEEE 31st International Conference on*, Oct 2013, pp. 384–389.

[12] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at http:www.csie.ntu.edu.twc̆jlinlibsvm.

[13] S. Ray and W. A. Hunt, Jr., "Connecting Pre-Silicon and Post-silicon Verification," in *Proceedings of the 9th International Conference on Formal Methods in Computer-Aided Design (FMCAD 2009)*, A. Biere and C. Pixley, Eds. Austin, TX: IEEE Computer Society, Nov. 2009, pp. 160–163.

[14] H. Yi, S. Park, and S. Kundu, "On-chip Support for NoC-based SoC Debugging," *IEEE Transactions on Circuits and Systems I*, vol. 57, no. 7, pp. 1608–1617, 2010.

[15] S. Prabhakar and M. Hsiao, "Using non-trivial logic implications for trace buffer-based silicon debug," in *Asian Test Symposium, 2009. ATS '09.*, nov. 2009, pp. 131 –136.

[16] G. Van Rootselaar and B. Vermeulen, "Silicon debug: scan chains alone are not enough," in *Test Conference, 1999. Proceedings. International*, 1999, pp. 892 –902.

[17] R. Datta, A. Sebastine, and J. Abraham, "Delay fault testing and silicon debug using scan chains," in *Test Symposium, 2004. ETS 2004. Proceedings. Ninth IEEE European*, may 2004, pp. 46 – 51.

[18] H. F. Ko and N. Nicolici, "Combining scan and trace buffers for enhancing real-time observability in post-silicon debugging," in *Test Symposium (ETS), 2010 15th IEEE European*, may 2010, pp. 62 –67.

[19] K. Basu, P. Mishra, and P. Patra, "Efficient combination of trace and scan signals for post silicon validation and debug," in *ITC*, 2011, pp. 1–8.

[20] K. Rahmani and P. Mishra, "Efficient signal selection using fine-grained combination of scan and trace buffers," *VLSI Design, International Conference on*, vol. 0, pp. 308–313, 2013.

[21] ——, "Efficient signal selection using fine-grained combination of scan and trace buffers," in *VLSI Design (VLSI Design), 2013 26th International Conference on*, jan. 2013.

Kamran Rahmani received the B.Sc. degree from the Department of Computer Engineering, Sharif University of Technology in Tehran, Iran and his M.S. degree from the Computer and Information Science and Engineering (CISE) Department, University of Florida. He is currently working as a Senior Software Engineer at Medallia Inc. and a graduate student at the Embedded Systems Lab in the CISE Department, University of Florida. His research interests include post-silicon validation and debug and reliable embedded systems.

Prabhat Mishra received his B.E. from Jadavpur University, Kolkata, M.Tech. from the Indian Institute of Technology, Kharagpur, and Ph.D. from the University of California, Irvine - all in Computer Science. He is currently an Associate Professor with the Department of Computer and Information Science and Engineering, University of Florida. His research interests include design automation of embedded systems, energy-aware computing, security and reliability, hardware/software verification, and post-silicon debug. He has published four books, ten book chapters and more than 100 research articles in premier international journals and conferences. His research has been recognized by several awards including the NSF CAREER Award from the National Science Foundation, two best paper awards (VLSI Design 2011 and CODES+ISSS 2003), and 2004 EDAA Outstanding Dissertation Award from the European Design Automation Association. Dr. Mishra currently serves as the Deputy Editor-in-Chief of IET Computers & Digital Techniques. He also serves as an Associate Editor of several journals including ACM Transactions on Design Automation of Electronic Systems, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, IEEE Design & Test, and Journal of Electronic Testing. He is a senior member of both ACM and IEEE.

Sandip Ray is a Principal Engineer at NXP Semiconductors in Austin, TX, USA. His research focuses on developing correct, dependable, and trustworthy computing systems through cooperation of specification, synthesis, verification, and validation techniques. Before joining NXP, Dr. Ray worked as a Research Scientist at Intel Strategic CAD Labs, where he led research on effective validation technologies to ensure security and correctness of next-generation SoC designs. Dr. Ray's research has found application in major semiconductor companies, including AMD, Freescale, IBM, Intel, Galois, and Rockwell Collins. Dr. Ray is the author of three books (two upcoming), as well as more than 40 peer-reviewed research articles. He has served on the technical program committee of more than 25 international meetings and conferences, as general and program chair for ACL2 2009 and FMCAD 2013, and as a guest editor for ACM TODAES and Springer JETTA. He serves as an Associate Editor for IEEE Transactions on Multi-scale Computing Systems. Dr. Ray has a Ph.D. from the University of Texas at Austin, and is a senior member of IEEE.