SCALABLE SIGNAL SELECTION FOR POST-SILICON DEBUG

By

KAMRAN RAHMANI

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2017

© 2017 Kamran Rahmani

I dedicate this to my family.

ACKNOWLEDGMENTS

First of all, I would like to thank my adviser Prof. Prabhat Mishra for all his help and support throughout this journey. It was him who led me to explore unknowns and to grow and push the boundaries. He taught me how to explore new directions and how to present my findings. He is the person who made this dissertation come true. I would also like to thank my Ph.D. committee members: Prof. Sartaj Sahni, Prof. Sanjay Ranka, Prof. Shigang Chen, and Prof. Nima Maghari for their valuable input and suggestions.

I would like to take this opportunity to thank those who helped me during different stages of my research. I would like to thank Dr. Sandip Ray from NXP (Qualcomm) who helped me understand the industry challenges in post-silicon debug. I sincerely thank Prof. Swarup Bhunia for helping me understand various aspect of memory-based computing. I am also thankful to my labmates Hadi Hajimiri, Sudhi Proch, Kanad Basu, Weixun Wang, and Xiaoke Qin for their help and support.

Last but not least, I would also like to extend my gratitude towards my family and their unconditional support and help.

.

TABLE OF CONTENTS

page

ACKNOWLEDGMENTS . . . 4

LIST OF TABLES . . . 7

LIST OF FIGURES . . . 8

ABSTRACT . . . 10

CHAPTER

1 INTRODUCTION . . . 11

1.1 Signal Restoration in Post Silicon Validation . . . 13
1.2 Challenges . . . 14
1.3 Research Contributions . . . 16
1.4 Dissertation Organization . . . 17

2 BACKGROUND AND RELATED APPROACHES . . . 18

2.1 Trace Signal Selection . . . 19
2.2 Dynamic Signal Selection . . . 20
2.3 Supervised Learning and Prediction . . . 21
2.4 Trace Data Compression . . . 22
2.5 Observability-Aware Test Generation . . . 22
2.6 Post-silicon Debug Techniques . . . 23
2.7 Combination of Trace and Scan Signals . . . 24

3 TRACE SIGNAL SELECTION USING AUGMENTATION AND ILP TECHNIQUES . . . 26

3.1 Augmentation-based Selection . . . 28
3.1.1 Augmentation-based Signal Selection . . . 29
3.1.2 ILP Optimization . . . 31
3.1.3 Complexity and Scalability . . . 33
3.2 Experiments . . . 34
3.2.1 Experimental Setup . . . 34
3.2.2 Results . . . 35
3.2.2.1 Restoration quality . . . 35
3.2.2.2 Selection time . . . 36
3.3 Summary . . . 38

4 EFFICIENT COMBINATION OF TRACE AND SCAN SIGNALS . . . 39

4.1 Fine-grained Combinations . . . 43
4.1.1 Debug Architecture . . . 43
4.1.2 Constrained Signal Selection (CSS) . . . 44
4.1.3 Flexible Signal Selection (FSS) . . . 49

4.2 Experiments . . . 53
4.2.1 Experimental Setup . . . 53
4.2.2 Restoration Quality . . . 54
4.2.3 Selection Time . . . 56
4.2.4 Hardware Overhead . . . 56
4.3 Summary . . . 62

5 SCALABLE TRACE SIGNAL SELECTION USING MACHINE LEARNING . . . 63

5.1 Learning-based Signal Selection . . . 63
5.1.1 Problem Formulation . . . 64
5.1.2 Overview and Motivation . . . 65
5.1.3 Signal Selection Algorithm . . . 68
5.1.3.1 Linear pruning . . . 71
5.1.3.2 Generating training vectors . . . 72
5.1.3.3 Final model selection . . . 73
5.1.3.4 Elimination-based signal selection . . . 75
5.1.3.5 Augmentation-based signal selection . . . 76
5.1.3.6 Signal selection using random initial set . . . 77
5.2 Feature-based Signal Selection . . . 78
5.2.1 Selection Model Generation . . . 79
5.2.1.1 Feature selection . . . 81
5.2.1.2 Model selection . . . 82
5.2.2 Signal Selection . . . 82
5.3 Experiments . . . 83
5.3.1 Learning-based Signal Selection . . . 85
5.3.1.1 Experimental setup . . . 85
5.3.1.2 Model selection . . . 85
5.3.1.3 Restoration quality . . . 86
5.3.1.4 Selection time, complexity, and scalability . . . 88
5.3.2 Feature-based Signal Selection . . . 90
5.3.2.1 Experimental setup . . . 90
5.3.2.2 Model Selection . . . 90
5.3.2.3 Restoration quality . . . 90
5.3.2.4 Selection time and scalability . . . 92
5.4 Summary . . . 93

6 CONCLUSIONS AND FUTURE WORK . . . 95

6.1 Conclusions . . . 95
6.2 Future Research Directions . . . 96

APPENDIX: LIST OF PUBLICATIONS . . . 98

REFERENCES . . . 99

BIOGRAPHICAL SKETCH . . . 105

LIST OF TABLES

Table — page

1-1 Illustration of restored signals for the simple circuit shown in Figure 1-4 15

3-1 Illustration of restored signals for the simple circuit shown in Figure 1-4 26

3-2 Restored signals for circuits in Figure 1-4 27

3-3 Restored signals from Chatterjee *et al.* [7] for the circuit in Figure 3-1 28

3-4 Restored signals using our method for the circuit in Figure 3-1 28

3-5 Restoration ratios using our approach compared with existing selection approaches . 36

4-1 Restored signals for the circuit shown in Figure 4-1 41

4-2 Trace buffer slots and shadow flip-flops values in our proposed debug architecture in Figure 4-2 41

4-3 Restored signals using our proposed debug architecture in Figure 4-2 42

4-4 Different restoration power values of our algorithm in example circuit of Figure 4-1 . 49

4-5 Restoration impact values of our flexible selection algorithm in example circuit of Figure 4-6 52

4-6 Different parameters used in CSS 54

4-7 Restoration ratios of different trace only approaches compared with our proposed architecture 55

4-8 Restoration ratios using our approach compared with Basu et al. [3] 56

4-9 Runtime comparison of different approaches 57

4-10 Configurations for fine-grained architectures 58

4-11 Area overhead of fine-grained signal selection compared to trace-only approach . . . 60

4-12 Power overhead of fine-grained signal selection compared to trace-only approach . . 61

5-1 Time complexity of our approach and existing signal selection techniques 79

5-2 Restoration ratios using our approach compared with existing selection approaches . 88

5-3 Runtime comparison of our approach compared with existing selection approaches . 89

5-4 Restoration ratios using our approach compared with existing selection approaches . 92

5-5 Runtime comparison of our approach compared with existing selection approaches . 94

LIST OF FIGURES

Figure		page
1-1	Different phases of validation and testing in IC design flow	11
1-2	An overview of post-silicon validation flow	13
1-3	Basic restoration rules for common logic gates	14
1-4	A simple circuit to illustrate restorability	14
1-5	Overview of research contributions	16
2-1	Simulation-based trace signal selection flow	20
2-2	Two different type of supervised learning	21
3-1	Example circuit to compare our approach with Chatterjee *et al.* [7]	27
3-2	Proposed signal selection process	29
3-3	Selection times of our approaches compared and normalized to Chatterjee *et al.* [7]	37
4-1	Simple circuit to illustrate restorability used in Basu et al. [3]	40
4-2	Proposed debug architecture for example circuit in Figure 4-1	42
4-3	Spectrum of existing and proposed debug architectures	43
4-4	Proposed debug architecture	45
4-5	An example of trace buffer partitioning	46
4-6	Sample circuit to illustrate our flexible selection algorithm	51
4-7	Sample scan-trace configuration	58
4-8	Area overhead of our approach for different circuit and trace sizes, benchmarks are ordered based on the number of flip-flops (shown in brackets)	60
4-9	Power overhead of our approach for different circuit and trace sizes, benchmarks are ordered based on the number of flip-flops (shown in brackets)	61
5-1	Overview of our approach and its relation to existing simulation based approaches	65
5-2	Real values of $r_m(v)$ versus predicted values for different random vectors in s38417 benchmark	67
5-3	Proposed signal selection process	68
5-4	Different signal selection techniques for exploring the search space	69

5-5 Number of profitable signals remaining after linear pruning 73

5-6 Overview of our proposed approach and its relation to existing simulation based approaches 80

5-7 Actual versus predicted values of selection ranks in s38584 benchmark 84

5-8 Real versus predicted restoration states for different models in *S38584* benchmark . 86

5-9 Mean prediction errors (MPE) of different models on the set of our benchmarks . . 87

5-10 Real versus estimated values of selection rank on S38584 benchmark 91

6-1 Overview of our proposed anomaly-based debug architecture 97

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

SCALABLE SIGNAL SELECTION FOR POST-SILICON DEBUG

By

Kamran Rahmani

May 2017

Chair: Prabhat Mishra
Major: Computer Engineering

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 challenge in post-silicon validation is limited observability and controllability. Design overhead considerations impose restrictions 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 under post-silicon observability restrictions. This dissertation proposes novel techniques to enhance the observability during post-silicon debug. My research has three major contributions. First, it proposes efficient signal selection techniques to enhance the observability of the circuit. Next, to improve the observability further, a new fine-grained scan and trace combination architecture has been proposed. Finally, the application of machine learning techniques has been extensively explored to improve the scalability of selection techniques. Extensive experimental results exhibit significant improvement in both overall signal observability and signal selection time.

CHAPTER 1
INTRODUCTION

Design complexity is rapidly increasing with drastic growth of number of transistors in each technology cycle. This has caused a significant increase in validation complexity. In spite of extensive effort on design validation using simulation and formal methods, it is not always possible to detect all the functional and electrical errors and bugs in pre-silicon validation. Post-silicon validation is used to capture the design flaws and escaped errors.

Figure 1-1 illustrates the different phases of validation and testing in a typical chip design flow. Pre-silicon validation includes validation of different functional and timing requirements of the specification and implementation. Manufacturing testing is used to detect the physical defects of each IC. In addition, post-silicon validation is used to detect design errors that have escaped pre-silicon validation phase. It should be noted that most of the electrical bugs such as crosstalk and transient faults are captured during post-silicon validation as it is difficult to model and fix those in pre-silicon validation. Post-silicon validation is a complex activity performed under aggressive schedules, representing more than 50% of the overall validation cost [1]. A fundamental challenge in post-silicon validation is limited observability and controllability. Due to limitations in the number of output pins as well as area and power overhead constraints of internal trace buffer, only a small percentage of internal signals in the design can be observed during silicon execution. Furthermore, in order for a signal to be observed, the design must be instrumented a priori with appropriate control hardware that

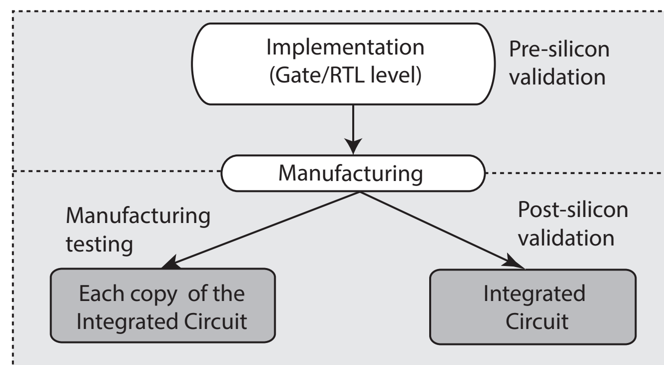


Figure 1-1. Different phases of validation and testing in IC design flow

routes a signal to an observation point. It is therefore crucial to identify trace signals that maximize design visibility and debug information under the observability constraints.

Figure 1-2 provides an overview of post-silicon validation and debug process. Signal selection and trace buffer design are done in pre-silicon phase. If an error occurs during post-silicon validation phase, the traced values of internal signals are dumped. During off-line trace analysis, a restoration procedure tries to reconstruct unknown signal states from the traced signal values. During off-line debug process both dumped signals and restored signals are used to locate the error. The number of signals that can be traced per cycle is limited to trace buffer width. In addition, the maximum number of values recorded per signal is limited to the trace buffer depth. Due to design overhead considerations, trace buffer size tends to be very small. In a million-gate design, a typical trace buffer width is 128 and depth is 2048. In other words, a 128×2048 trace buffer can trace 128 signals (out of millions of possible signals) for 2048 cycles. Therefore, the primary objective is to select a small set of profitable signals that can maximize restoration performance. A major challenge in efficient signal selection is that the exploration space (number of potential alternatives) can be prohibitively large even for small circuits. For example, s35932 circuit of ISCAS'89 benchmarks suite has 1728 flip-flops. If the trace buffer width is 32, we need to choose 32 signals out of the total 1728 flip-flops. It is easy to observe that there are more than 10^{69} such combinations. This makes exhaustive exploration infeasible[1] .

The focus of this dissertation is to reduce the overall effort of post-silicon validation and debug. The remainder of this chapter is organized as follows. First, we describe the basics of signal restoration which is an important concept in post-silicon validation. We then explain

[1] If each simulation for evaluating one combination takes only 1 second, more than 10^{60} years is needed in order to find the best 32 trace signals in s35932 circuit using one single machine.

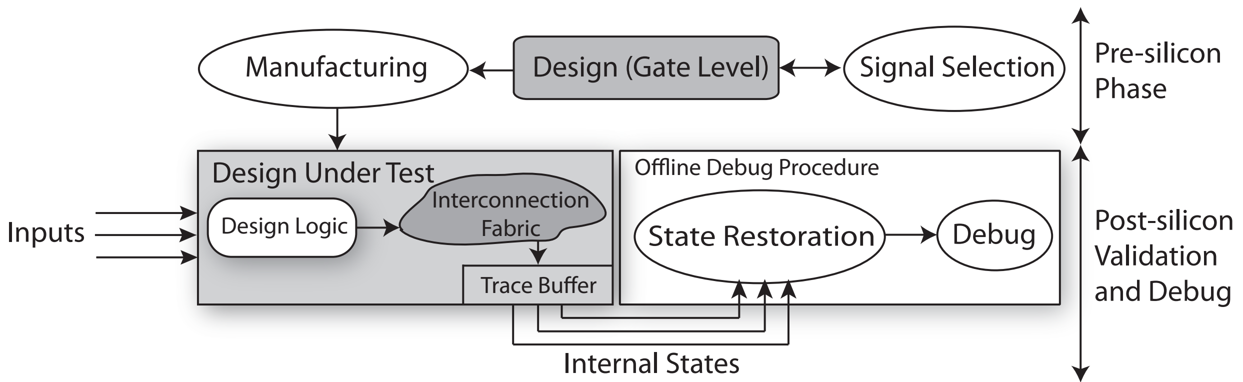


Figure 1-2. An overview of post-silicon validation flow

various challenges associated with post-silicon validation and our contributions to address them.

1.1 Signal Restoration in Post Silicon Validation

Restoration entails inferring values of untraced signal states from a sequence of traced signals sampled over a period of time. This is achieved using forward and backward propagation of signal values of circuit elements (*e.g.*, gates, latches, etc.). Figure 1-3 illustrates forward and backward restoration for common logical gates. Forward propagation involves reconstructing the output of a circuit element from traced inputs. For example, if one of the inputs of the *OR* gate is '1', the output value would be '1'. If all the inputs are known, the unknown output can be definitely determined. On the other hand, backward propagation involves inferring input values from the observed output. For example, if the output of the *OR* gate is '0', both of the inputs would be '0'. Backward reconstruction might fail in certain scenarios. For example, if the output of a 2-input *OR* gate is '1' and one of the input has a known value of '1', the other input still cannot be reconstructed. During signal value reconstruction, forward and backward restoration are repeated for all the gates in the circuit 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.

$$\text{Restoration Ratio} = \frac{\text{Number of traced and restored signals}}{\text{Number of traced signals}}$$

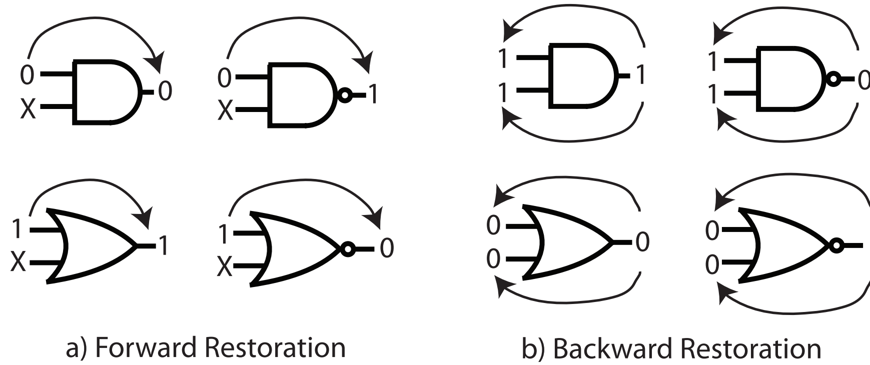


Figure 1-3. Basic restoration rules for common logic gates

Consider the simple circuit shown in Figure 1-4 used in [3]. 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 1-1 shows the signal states that can be restored using the traced values of A and C over eight cycles. In this example, 32 signal values can be restored (entries with '0' or '1' in the table) while 16 are traced (two signals for eight cycles, i.e., $2 \times 8 = 16$), yielding a restoration ratio of $(32 + 16)/16 = 3$.

1.2 Challenges

There are many challenges in efficient post-silicon validation and debug. This section describes three important challenges that I have addressed in my dissertation.

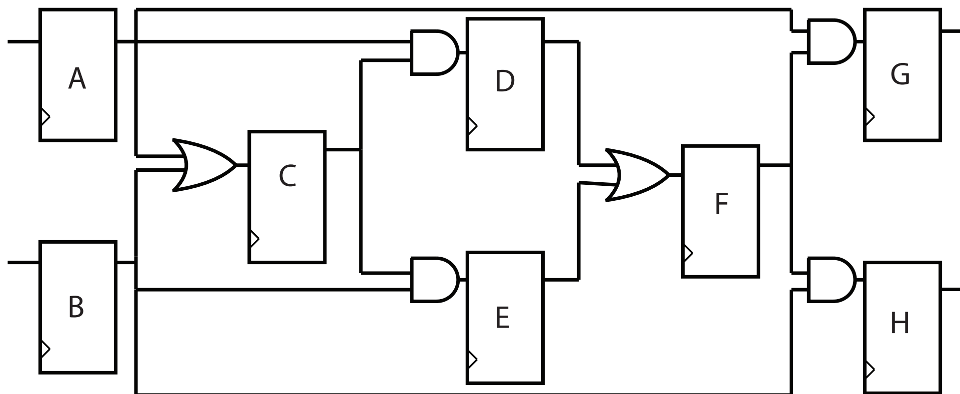


Figure 1-4. A simple circuit to illustrate restorability

Table 1-1. Illustration of restored signals for the simple circuit shown in Figure 1-4

Signal	Cycle 1	Cycle 2	Cycle 3	Cycle 4	Cycle 5	Cycle 6	Cycle 7	Cycle 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

Challenge 1: There is a limited number of signals to be traced which should be selected carefully in order to maximize the restoration ratio during the post silicon debug. Existing approaches either use some metrics based on the circuit structure to select the signals [3–6] or they use expensive simulations/restorations [7]. The former can sacrifice the restoration quality and the latter is not scalable if it is not done carefully, due to the prohibitive computation time requirements. An efficient selection algorithm is needed which can select trace signals to provide a high restoration ratio.

Challenge 2: As discussed in *Challenge 1*, metric-based selection techniques are fast and scalable but provide a low restoration performance. On the other side, simulation-based techniques provide a high restoration performance but generally are not scalable for large circuits. A scalable selection technique is needed which can be applied to large industry scale circuits.

Challenge 3: The use of scan chains in post-silicon debug has been extensively studied in [8, 9]. Ko et al. [10] proposed an architecture that divides trace buffer bandwidth into two parts, one for the trace signals and the other one for the scan signals. In order to find the most beneficial partitioning, they proposed an exhaustive exploration. However, exhaustive exploration is not practical in real designs with large number of flip-flops. Basu et al. [11] proposed an efficient algorithm that chooses trace and scan signals based on connectivity graph of flip-flops. They reduce the scan chain length by pruning the graph in each iteration. However, both of these techniques divide the signals in two extreme categories. One set of

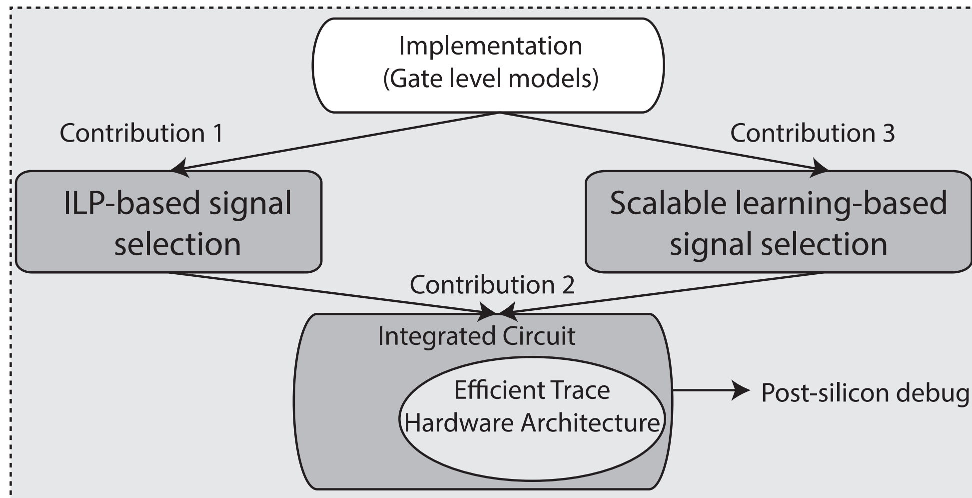


Figure 1-5. Overview of research contributions

signals are traced every cycle. The other signals are dumped in a relatively large period. They do not consider other profitable fine-grained scenarios. An efficient fine-grained architecture is needed that shares the trace buffer bandwidth between several scan chains with different lengths to significantly improve restoration performance.

1.3 Research Contributions

My research proposes novel techniques to address challenges in post-silicon validation and debug described in Section 1.2. The objective of my research is to develop efficient signal selection techniques as well as trace hardware architectures. The major contributions of my research are summarized as follows. Figure 1-5 highlights these contributions in the IC design methodology.

ILP-based signal selection: This contribution addresses the first challenge outlined in Section 1.2. Most existing signal selection techniques rely on a metric based on circuit structure. Simulation-based signal selection is promising but have major drawbacks in computation overhead and restoration quality. In this dissertation we propose an efficient simulation-based signal selection technique to address these bottlenecks. Our approach uses (1) bounded mock simulations to determine state restoration effectiveness, and (2) an ILP-based algorithm for refining selected signals over different simulation runs.

Efficient selection of fine-grained combination of trace and scan signals: This contribution addresses the first and third challenge outlined in Section 1.2. One of the key challenges in post-silicon validation is the limited observability of internal signals in manufactured chips. A promising direction to improve observability is to combine trace and scan signals - a small set of trace signals are stored every cycle, whereas a large set of scan signals are dumped across multiple cycles. Existing techniques are not very effective since they explore a coarse-grained combination of trace and scan signals. In this dissertation, we propose a fine-grained architecture that addresses this issue by using various scan chains with different dumping periods. We also propose efficient algorithms to select beneficial signals based on this architecture.

Scalable Trace Signal Selection using Machine Learning: This contribution addresses the first and second challenge outlined in Section 1.2. 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 dissertation, we propose efficient signal selection techniques using machine learning to take advantage of simulation-based signal selection while significantly reducing the simulation overhead. Our approaches use (1) bounded mock simulations to generate training vectors set for the machine learning technique, and (2) train a machine learning model and apply it to the circuit to identify the most profitable signals set.

1.4 Dissertation Organization

The dissertation is organized as follows. Chapter 2 describes background and the related work. Chapter 3 presents our ILP-based trace signal selection technique. Chapter 4 describes our fine-grained trace and scan signal architecture and corresponding selection techniques. Chapter 5 presents our scalable selection algorithms using machine learning techniques. Finally, Chapter 6 concludes the dissertation.

CHAPTER 2
BACKGROUND AND RELATED APPROACHES

A primary problem for post-silicon debug is the limited observability of internal signals since the chip has already been fabricated. Once the signal states are known, they can be analyzed using techniques like failure propagation tracing [12] to identify the errors in the circuit. Koushanfar et al. [13] proposed a technique to obtain the internal states of a system using a concept called golden cut. However, their method is not applicable for post-silicon debug since it is difficult to stop execution of a process running on a chip and get all the current signal states. Formal analysis for post silicon debug proposed by De Paula [14] is not applicable to circuits with a large number of gates. Physical probing techniques were proposed by Nataraj et al. [15]. Decrease in feature size and growing complexity of IC designs have made it difficult to implement these techniques in practice. A method for validation of memory subsystem in CMPs was proposed by DeOrio et al. [16], where it only focuses on the memory subsystem. Scan based debugging techniques such as [17] are not appropriate since they require to stop the circuit functionality when the scan data is being written. This is particularly not beneficial when the functional errors are drastically apart. Double buffering [18] of scan elements helps to mitigate this problem, but with a large area penalty. Design-for-Debug (DfD) techniques have been used extensively to increase the observability of internal signals of the silicon. Generally this is performed by sampling the data which is stored in on-chip trace buffers. Various DfD techniques like embedded logic analyzer (ELA) [19] and shadow flip flops [18] have been proposed over the years for post-silicon debug. ELA can be used to probe into the chip and record some internal logic states. The trace is then recorded in an on-chip trace buffer. During debug, the contents of trace buffer is transferred to an offline debugger via some Joint Test Action Group (JTAG) interface.

The notion of restorability is based on the execution of the circuit on an input sequence; when the input sequence represents an on-field execution scenario for the circuit during post-silicon validation. However, signals must be selected *a priori* based on the circuit structure.

Heuristics for selecting signals need to comprehend and encapsulate overlaps and interactions between different signals, and anticipate how such interactions might affect restorability on-field — an intrinsically difficult task. In this chapter, we present the background and related work for trace signal selection techniques as well as other methodologies to improve the overall post-silicon debug effort.

2.1 Trace Signal Selection

Trace buffers have been widely studied in post-silicon debug [6, 20–24]. Existing signal selection approaches can be classified in two categories, *metric-based (structural)* and *simulation-based*. Approaches in the first category use greedy heuristic to iteratively select signals optimizing a metric based on the circuit structure [3–5]. They are relatively efficient in computation speed, but have poor restoration quality compared to simulation-based algorithms. Simulation-based algorithms are based on the intuition that if a set of signals works well for some random input vectors then it is also likely to provide high state reconstruction on other inputs and therefore a high restorability ratio. In particular, Chatterjee *et al.* [7] showed that mock simulations are more effective in identifying trace signals than metrics based on the circuit structure. Their approach involves an iterative removal process. They start with a set of candidate signals which is initialized with all flip-flops. In each iteration, their algorithm attempts to remove one of the signals which appears to be least important based on simulation results. The process continues until the number of remaining candidates equals to the trace buffer width. Figure 2-1 illustrates the approach for a sample circuit with a total of 4 flip-flops and a trace buffer of width 2. This selection technique is promising but it 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.* [25] proposed a hybrid (metric-based

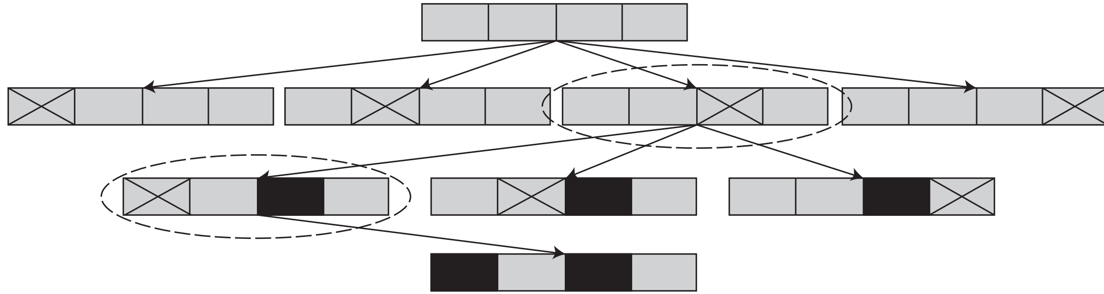


Figure 2-1. Simulation-based trace signal selection flow

and simulation-based) signal selection technique. However, to save selection time, [25] uses simulation for a small fraction of the signals and thereby sacrifices restoration performance. Ma et al. [26] proposed a selection technique based on a new metric called Pagerank which tries to maximize behavioral coverage during post-silicon debug. Zhu et al. [27] proposed a signal selection technique with the aim of facilitating the subsequent automated localization of faults using consistency-based diagnosis. BeigMohammadi et al. [28] proposed a signal selection method that selects combinational gates in addition to flip-flops in the circuit with the goal of further improving restoration capability.

To address the challenges of the simulation based approach proposed in [7], we present a top-down simulation-based technique in Chapter 3. Our approach has two components: (1) an iterative approach to signal selection based on mock simulations, and (2) a filtering scheme based on Integer-Linear Programming (ILP) to refine the selected set.

2.2 Dynamic Signal Selection

Existing trace signal selection techniques focus on selecting a set of signals that are traced every cycle during the debug time. Prabhakar et al. [29] proposed a technique where they alternate between two set of signals every other cycle based on implication-based correlation. Liu et al. [30] proposed a multiplexed signal selection for error detection using a heuristic based on error-visibility metric. Basu et al. [31] proposed a dynamic selection approach where it considers both spatial and temporal distribution of errors. Han et al. [32] proposed a dynamic signal selection based on the characteristic of the circuit under test to maximize the visibility.

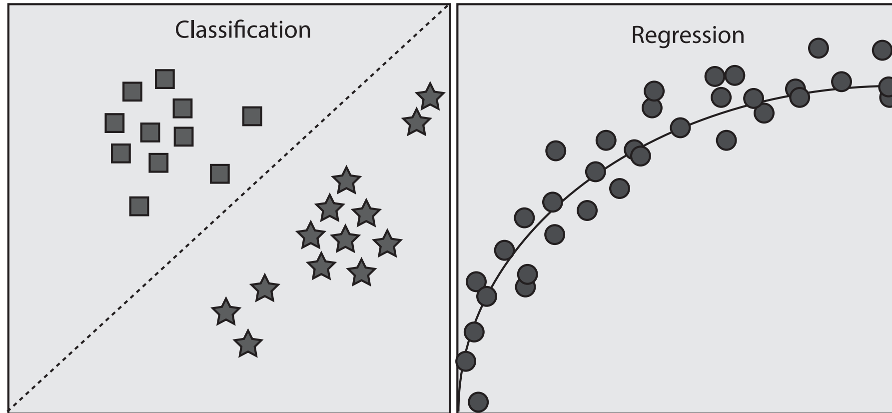


Figure 2-2. Two different type of supervised learning

Zhu et al. [33] proposed a dynamic selection technique which utilizes machine learning for classification of the groups of signals that are likely to trigger different faults.

2.3 Supervised Learning and Prediction

Supervised learning is a technique of inferring an unknown output for an input vector using a set of training examples consisting of pairs of input vectors and corresponding known outputs. There is two main categories for supervised learning in machine learning, classification and regression. Classification is predicting whether an element belongs to a set of discrete values (or classes) whereas, regression is predicting the value for a continuous function. An example for classification is classifying an email as spam or legitimate email based on some features of its content and meta data (feature vector). On the other side, predicting the price of a house based on its features like location, size, and age (feature vector) would be a regression prediction as the price is a continues value. In fact, classification is a special case of regression where each class is assigned to a range of values (or probabilities) of the prediction function. This is illustrated in Figure 2-2. The classification predictor divides the outputs to different classes, square and star. On the other hand, the regression is a continuous function trying to fit the best line passing through the inputs and outputs. In Chapter 5, we develop novel signal selection techniques based on regression to retain (and improve upon) the restoration quality of simulation-based signal selection with a drastically faster selection time.

2.4 Trace Data Compression

To increase the amount of data that can be stored in a trace buffer while keeping the trace buffer size constant, trace compression techniques have been proposed [34–37], which compress the trace data before storing them into trace buffer. This enables us to observe more trace data while keeping the trace buffer size constant. The trace buffer has two parameters, width and depth. Width refers to the number of signals whose states can be stored every cycle, while depth refers to the number of cycles over which the trace is stored. Existing trace compression approaches differ in terms of compression objectives. While [36] compresses the width of the trace buffer, [34, 35] compress the depth. Muthyala et al. [38] proposed a feedforward techniques for improving the compression in scan based decompressors.

2.5 Observability-Aware Test Generation

A major problem in post-silicon validation is that the regression tests may not be observability-aware. While the tests are likely to activate the errors, but there effects may not propagate to the observable points (e.g., trace signals). For example, if the result of a test affects a signal and it is not observable, it is difficult to determine whether the test has executed as expected.

Soft errors and crosstalk faults are two major electrical errors found in a fabricated SoC. Effect of soft errors on memory devices had been studied as early as in 1979 by May et al. [39]. Over the years, researchers [40–42] have studied the various aspects of soft errors. Sanyal et al. [43, 44] have proposed different methods for directed test generation for soft errors. However, these approaches are not designed for post-silicon validation purposes, that is, they assume all the output signals of a logic block are visible. However, during post-silicon validation, since the chip is fabricated, observing the output signals of every component may not be feasible since these components can be embedded in an SoC. The only observable points would be the trace signals. The test generation algorithms need to be modified to take this into account. Crosstalk faults occur when two lines in a circuit are so near that their mutual capacitance affects their state. Effects of crosstalk faults on digital circuits [45–47] have been studied

extensively. Existing test generation algorithms for crosstalk faults [48–50] suffer from the same problem as the corresponding test generation algorithms for soft errors - that is, they are not suitable for application in post-silicon validation due to limited observability through trace signals. There has also been recent approaches [51, 52] proposed which exploit an on-chip hardware component to generate real-time stimuli to satisfy the functionally-compliant stimuli requirements.

In addition to soft errors and Crosstalk faults, there has been several works on detecting functional errors. Adir et al. [53] proposed an approach to detect functional errors using random test generation. On the other side, directed test generation techniques [54, 55] try to generate tests that target a specific functional scenario. However, they are effective only if the list of the bugs are available a priori. Farhmandi et al. proposed a test generation approach [56] that is guaranteed to activate unknown bugs. Their technique also addresses the scalability concern of existing bug localizations approaches [57–59] with a faster bug localization. However, none of these approaches consider observability (trace signals) during test generation. Recently, Farahmandi et al. [60] proposed a framework for observability-aware test generation using transaction-level models.

2.6 Post-silicon Debug Techniques

Trace buffers are used to capture the signals during post-silicon validation. Recent works [34, 35, 61, 62] has been proposed with the goal of reducing the validation time by reducing the debug iterations. The main idea is that it is not necessary to store all the cycles specially for those signals that can be restored using simulation or those that are most likely from a non-buggy cycle. Their approach involve a multiple-step debug session where it allows the debugger to expand the visibility window. Effectively, it allows to zoom in and zoom out during the debugging session by storing a signature of trace signal values in a larger block of cycles. This means, data might get lost as we compress several bits in a few parity bits. In fact, during the debug session we do not know which cycles are the erroneous ones to store and we rely on a multiple-step offline software analysis to find those based on the input. There has also

been other approaches [63–65] proposed where they use trace data for detecting electrical errors (such as bit-flips), in addition to functional design errors. Goossens et al. [66] proposed a debug technique where it focuses on observing the communications among the IP blocks and mapping it to the transactions. Similar approaches have been proposed in [67, 68]. Zheng et al. [69] proposed a technique to infer system-level transactions from trace data to have a better high-level understanding of the system during the debug.

2.7 Combination of Trace and Scan Signals

Scan based debugging is widely used in manufacturing test. It is used to detect the fabrication defects in the chip. It would be beneficial to use scan dump in post-silicon debug. The problem is that the data can only be dumped in scan or debug mode. During these modes the circuit execution needs to be interrupted. This prevents real-time observability of the internal states. Enhanced scan chains are proposed in [10] to address this issue in which shadow flip-flops are used to form a shadow scan chain. Shadow flip-flops enable storing scan signals without interrupting normal execution of the circuit.

The use of scan chains in post-silicon debug has been extensively studied in [8, 9]. Ko et al. [10] proposed an architecture that divides trace buffer bandwidth into two parts, one for the trace signals and the other one for the scan signals. In order to find the most beneficial partitioning they proposed an exhaustive exploration. However, exhaustive exploration is not practical in real designs with large number of flip-flops. Basu et al. [11] proposed an efficient algorithm that chooses trace and scan signals based on connectivity graph of flip-flops. They reduce the scan chain length by pruning the graph in each iteration. However, both of these techniques divide the signals in two extreme categories. One set of signals are traced every cycle. The other signals are dumped in a relatively large period. They do not consider other profitable fine-grained scenarios. It would be beneficial if we divide the signals in a large number of categories in terms of dumping period. This enables us to select a promising signal with a profitable dumping period. In Chapter 4, we propose an efficient fine-grained architecture that shares the trace buffer bandwidth between several scan chains with different

dumping periods. We also propose two different signal selection algorithms which can be used based on hardware constraints. Our signal selection algorithms assign the signals to different scan chains in order to maximize the number of states that can be restored.

CHAPTER 3
TRACE SIGNAL SELECTION USING AUGMENTATION AND ILP TECHNIQUES

In this chapter, we describe our ILP-based signal selection which is inspired by simulation-based signal selection techniques, but includes a refinement technique to address the weaknesses of previous simulation-based approaches. Before presenting the technical details of our approach, we motivate it by comparing its results using illustrative examples with state-of-the-art metric-based and simulation-based approaches, *viz.*, Basu *et al.* [3] and Chatterjee *et al.* [7]; these experiments expose some key features of our approach which we then discuss.

For the circuit in Figure 1-4, Basu *et al.* [3] select signals A and C, thus yielding the restoration ratio of 3 as shown in Table 3-1. On the other hand, both our approach and the simulation-based approach of Chatterjee *et al.* [7] selects signals A and B. The corresponding restorability calculations are shown in Table 3-2. From the table, 40 states are restored from tracing 16 states, yielding a restoration ratio of 3.5.

On the other hand, to illustrate the distinction between our approach and Chatterjee *et al.* [7] consider the circuit in Figure 3-1. For a trace buffer width of 2, Chatterjee *et al.* [7] produce signals B and C. From Table 3-3, this leads to a restoration of 13 states from a total of 16 traced states, yielding a restoration ratio of 1.81. On the other hand, our approach selects signals C and K. From Table 3-4, this allows restoration of 18 states from 16 traced states, resulting in a restoration ratio of 2.13.

Table 3-1. Illustration of restored signals for the simple circuit shown in Figure 1-4

Signal	Cycle 1	Cycle 2	Cycle 3	Cycle 4	Cycle 5	Cycle 6	Cycle 7	Cycle 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

Table 3-2. Restored signals for circuits in Figure 1-4

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

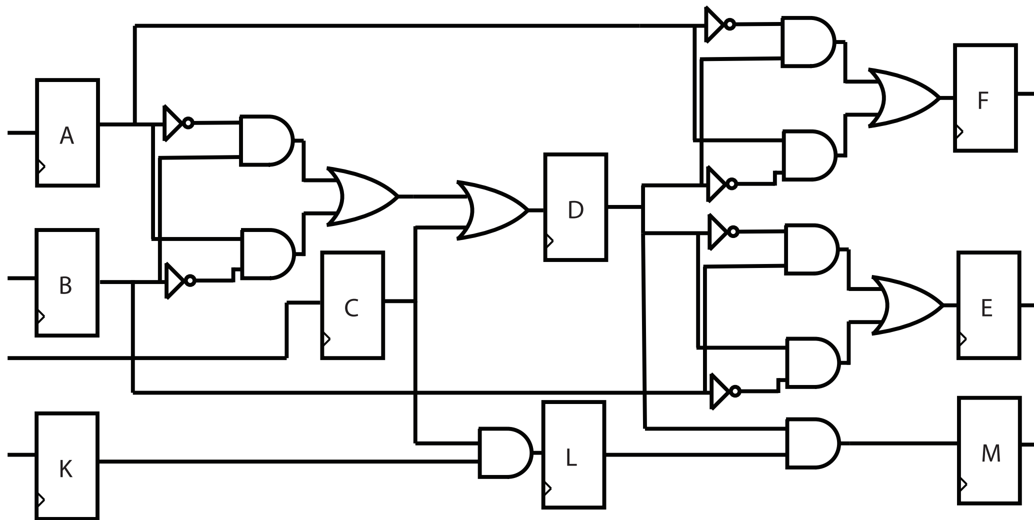


Figure 3-1. Example circuit to compare our approach with Chatterjee *et al.* [7]

It is illuminating to understand the source of the differences between the different approaches on these simple examples. The high restoration ratio achieved by both our approach and that of Chatterjee *et al.* [7] for the circuit in Figure 1-4 represents a general trend of superior signal quality achieved by simulation-based selection techniques; our observations here match the conclusions of Chatterjee *et al.* [7] as well. The comparison with Chatterjee *et al.* [7] for the circuit in Figure 3-1 is more interesting. Their approach is based on *greedy elimination*: starting with the set of all signals, they iteratively remove signals one at a time. In each iteration the objective is to select a candidate signal whose elimination minimizes the number of states which become unrestorable as a result; this signal is then eliminated and the algorithm iterates. The problem with this approach is that the candidate computation

Table 3-3. Restored signals from Chatterjee *et al.* [7] for the circuit in Figure 3-1

Signal	Cycle 1	Cycle 2	Cycle 3	Cycle 4	Cycle 5	Cycle 6	Cycle 7	Cycle 8
A	X	X	X	X	X	X	X	X
B	0	0	1	1	0	0	1	0
C	0	1	1	1	0	1	1	0
D	X	X	1	1	1	X	1	1
E	X	X	X	0	0	1	X	0
F	X	X	X	X	X	X	X	X
K	X	X	X	X	X	X	X	X
L	X	0	X	X	X	0	X	X
M	X	X	0	X	X	X	0	X

Table 3-4. Restored signals using our method for the circuit in Figure 3-1

Signal	Cycle 1	Cycle 2	Cycle 3	Cycle 4	Cycle 5	Cycle 6	Cycle 7	Cycle 8
A	X	X	X	X	X	X	X	X
B	X	X	X	X	X	X	X	X
C	0	1	1	1	0	1	1	0
D	X	X	1	1	1	X	1	1
E	X	X	X	X	X	X	X	X
F	X	X	X	X	X	X	X	X
K	0	0	0	0	0	0	1	1
L	X	0	0	0	0	0	0	1
M	X	X	0	0	0	0	0	0

assumes that all the remaining signals are available for state restoration, an assumption that is flawed precisely by virtue of the iterative elimination algorithm itself. Thus it is possible that a profitable signal s is eliminated in an early iteration when the states reconstructible from s can also be restored by other signals available at that iteration; however, these states can no longer be reconstructed when a subsequent iteration eliminates other signals. In the example, the signal K is eliminated in an early iteration since the states restorable from K can be restored without K as long as the signal L is available; however, when a subsequent iteration eliminates L as well, the set of states that can be restored gets drastically reduced.

3.1 Augmentation-based Selection

Our algorithm exploits the advantages of simulation-based signal selection while avoiding the drawbacks discussed above. Figure 3-2 illustrates the framework. We apply an iterative

approach based on augmentation rather than elimination. In particular, we maintain a set S of signal candidates (initially empty), which we "grow" at each iteration by identifying the most promising signal based on mock simulations; the objective is to maximize the set of states that can be restored from the signals in the candidate set. The key observation here is that in this approach restorability of the candidate set is never over-estimated at any iteration since each member of S is guaranteed to be in the final trace selection set. Furthermore, note that the number of iterations in this approach is bounded by buffer size, which is very small precisely because of the observability limitation in post-silicon validation. On the other hand, the number of iterations in the elimination-based selection is bounded by the number of signals which can be large. Thus our approach achieves much better run-time performance compared to the elimination-based selection.

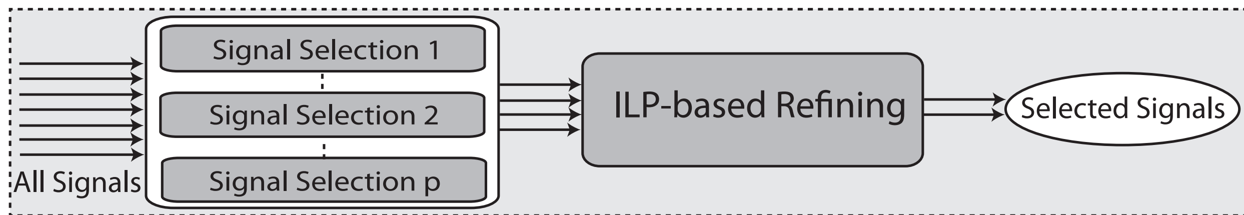


Figure 3-2. Proposed signal selection process

Our second observation is that any selection algorithm based on random simulation is susceptible to perturbations based on the randomness in the input vectors. To eliminate the influence of randomness, our approach makes use of multiple simulation runs using an ILP-based refinement algorithm to consolidate the results from these runs.

3.1.1 Augmentation-based Signal Selection

We first describe our augmentation-based selection algorithm; we will discuss the ILP-based refinement in the next subsection. Algorithm 1 outlines the major steps of the signal selection process. The inputs of the algorithm are the *circuit, trace buffer width (w),* and *the number of cycles in mock simulations (c)*. To understand the workings of the algorithm we need two key concepts: *restoration influence* and *restoration difference*.

Given a set of candidate signals s, an input vector I, and the number of simulation cycles c, we define the *Restoration Influence* $RI(s, I, c)$, as the total number of states that can be restored if we do a mock simulation over c cycles using input vector I and the signals set S. The *restoration difference* between two candidates s_1 and s_2 with respect to I and c, denoted by $RD(s_1, s_2, I, c)$, is then given by the following formula:

$$RD(s_1, s_2, I, c) = RI(s_1, I, c) - RI(s_2, I, c)$$

Data: circuit, w, c
Result: Selected set of signals S
Create flip-flops graph of circuit;
Create list of selected signals S ▷ initially empty ;
while $|S| < w$ **do**
  Generate a random input vector I ;
  foreach *flip-flop* f *that is not in the* S **do**
    Calculate $RD(S \cup \{f\}, S, I, c)$;
  end
  Find flip-flop f with maximum RD. If two or more flip-flops have same RD, find the one with higher connectivity ;
  Add f to the list S ;
end
return S ;

Algorithm 1: SelectSignals Procedure

Informally, for a given c-cycle mock simulation I, the restoration difference between two candidate signal sets s_1 and s_2 measures the observability improvement achieved by selecting s_2 over s_1. In particular, if $s_2 = s_1 \cup \{f\}$ for some design signal f, then it measures the observability improvement achieved by augmenting s_1 with f. Algorithm 1 is a greedy algorithm that uses this metric to iteratively grow the set S of currently selected signals. At each iteration, it (1) performs a new simulation for c cycles using a random input vector I, (2) computes the restoration difference between S and $S \cup \{f\}$ for each design signal f, and (3) augments S with the signal that maximizes the restoration difference. If two or more

signals have identical restoration difference, then the tie is broken in favor of the signal that has the highest connectivity.[1] The process is continued until w signals have been selected.

3.1.2 ILP Optimization

Experiments show that most of the selected trace signals are identical in different runs of our signal selection. However, in any simulation-based signal selection approach, signals may be different in different runs depending on generated random input vector seed. The goal of our refinement algorithm is to eliminate the influence of randomness and also to cover more states of the circuit through selected signals. To do so, we use multiple runs of the signal selection algorithm which are then processed by ILP to select the best signal set among all outcomes.

```
Data: circuit, w, c, p
Result: The matrices associated to the ILP problem
Create S[1..p][1..w] and R[1..p][1..w] ;
Create k and j, initialize to 1 ;
Create list of all selected signals A          ▷ initially empty ;
Create list of selected signals S              ▷ initially empty ;
while k <= p do
    T = Signal selection algorithm with (circuit, w, c) ;
    Generate a random input vector I ;
    j = 1 ;
    foreach flip-flop f in the T do
        S[k][j] = f ;
        A = A ∪ {f} ;
        RD_f = RD(T, T − {f}, I, c) ;
        R[k][j] = RD_f ;
        j++ ;
    end
    k++ ;
end
return A, S, and R ;
```

Algorithm 2: InitializeMatrices Procedure

[1] The connectivity of a flip-flop is the number of flip-flops connected to it through other combinational gates in both backward and forward directions.

To perform the refinement, we first create ILP formulation matrices from the signal selection algorithm. Algorithm 2 outlines the steps involved. The inputs of the algorithm are the *circuit*, *trace buffer width (w)*, *the number of cycles in mock simulations (c)*, and *refinement precision (p)*. The refinement precision specifies the number of runs of the signal selection algorithm used in the refinement process. The algorithm returns two matrices S and R, and a set A, which are then used as the basis of the ILP optimization. A is the set of all flip-flops selected in the p runs of our selection algorithm. The matrices S and R record the importance of the selected flip-flops in state reconstruction: $S[k][j]$ records the j-th selected flip-flop in the k-th run of our selection algorithm; $R[k][j]$ records the number of states that is lost in the mock simulation corresponding to the k-th run if $S[k][j]$ is removed from the final selected set. The algorithm executes p runs of our selection algorithm, filling out the entries $S[k][j]$ and $R[k][j]$ at the k-th run. Recall that the perturbation caused to the selection set is typically small. Thus, for the set T of flip-flops computed in the k-th run and any $f \in T$, most of the signals in $T - \{f\}$ end up in the final selected signal set; thus, the value of $R[k][j]$ is a reliable estimate of the importance of flip-flop $S[k][j]$.

Once the required matrices are initialized, we can model our refinement process as an ILP optimization problem in a fairly standard manner. For each flip-flop in A, we create a variable which can be 0 or 1. $A_i = 1$ indicates that A_i is eliminated; $A_i = 0$ indicates that it is not removed and therefore exists in final trace signals set. Note that since A is a cumulative superset of all selected flip-flops during p runs, for each $1 \leq i \leq p$ and $1 \leq j \leq w$, we have $S[i][j] \in A$. Equation 3–1 shows the objective function which should be minimized. L_i is the number of states that is lost in i_{th} run, based on signal assignments in A. The aim is to minimize the total number of lost states in all the runs.

$$min : \sum_{i=1}^{p} L_i \tag{3–1}$$

Equation 3–2 shows how L_i is calculated. Recall that $S[i][j]$ is the assignment of signal j in A (which is 0 or 1), and $R[i][j]$ is the number of states that is lost in i-th run if j-th signal

is removed (i.e., is equal to 1). Therefore, L_i is the total number of states that is lost due to removed flip-flops of i-th run.

$$L_1 = \sum_{i=1}^{w} S[1][i] * R[1][i] \ ..., \ L_p = \sum_{i=1}^{w} S[p][i] * R[p][i] \tag{3–2}$$

The constraints of ILP optimization problem are shown in Equation 3–3. Recall that A is the superset of all selected signals in different runs. However, $|A|$ may be larger than w as some selected signals may be different during signal selection runs. It means $|A| - w$ signals must be removed from A. These signals are removed in such a way that the total number of lost states in all runs is minimized. The remaining w flip-flops in A which are assigned to 0 correspond to the final trace signals set.

$$\sum_{i=1}^{|A|} A_i = |A| - w$$

$$A_1, A_2, ..., A_{|A|} \in \{1, 0\} \tag{3–3}$$

3.1.3 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 required in the computation. Note that although our approach involves ILP-based optimization, the running time for solving the ILP in practice is still negligible compared to the time for mock simulations. The reason is that the perturbation caused by randomization in simulations in practice to the selected set of signals is small, so that there is a large overlap between the signals selected at different runs. Thus, the selected set A of flip-flops over all the different runs in our ILP-based refinement is of the order of the width of the trace buffer, independent of the number p of iterations of the selection algorithm actually performed. Consequently, we compute the complexity of our algorithm in terms of the number of required mock simulations.

Assume that there are N flip-flops in the circuit and the trace buffer width is w. Number of needed simulation in each run of signal selection algorithm is $N + (N-1)+...+(N-w+1)$. Note that $N >> w$ for large circuits, since the trace buffer size is bounded by the observability limitations. The complexity of Algorithm 1 is thus $\theta(Nw)$. Algorithm 2 consists of a main loop which runs signal selection algorithm followed by w additional simulations to fill in matrix R. Consequently, each iteration needs $\theta(Nw) + \theta(w) = \theta(Nw)$ simulations. Therefore, the complexity of our algorithm is $p * \theta(Nw) = \theta(Npw)$. However, our experiments show that in practice $p << N$ is enough to cover most of the input vectors. Consequently, in most cases, our algorithm requires fewer simulations than the previous simulation-based approach of Chatterjee *et al.* [7], which has a complexity of $O(N^2)$ — with the lower bound of $\Omega(N^2/d_{step})$ which is still computationally expensive since $N >> d_{step}$ in large industry-scale circuits ($d_{step} = 50$ in their experiments). On the other hand, the hybrid approach [25] 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. Note that once the parameters are fixed, both our approach and the hybrid approach have the same asymptotic complexity $\theta(N)$), with different constant coefficients.

In addition, not only all the simulations in each iteration of our selection algorithm are independent, but the iterations of initialization algorithm are also independent tasks. This makes our approach scalable for very large industry-level circuits by running them in parallel in a multi-processor environment.

3.2 Experiments

3.2.1 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 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* [70]. We also used *lp_solve* 5.5 [71] to solve the ILP optimization part of our approach.

In the results reported below, the comparisons with related work [7, 25] are based on our implementation of their results. The reason is that their reported results used their own synthesized/optimized version of the ISCAS'89 benchmarks, while we used the standard, publicly available versions. Moreover to make the comparison fair for comparing restorability, identical input vectors should be used in all the approaches. We used the same parameters $c = 64$ and $PT = 95\%$ as reported in Chatterjee *et al.* [7]. In addition, we used the same parameters $M = 64$, $k = 5\%$, and an initialization simulation of 10K cycles as reported in Li *et al.* [25]. We also used $c = 32$ and $p = 6$ for our approach in our experiments. Our experiments demonstrate that restoration ratio shows no improvement for $p > 6$ in the set of used benchmarks. After signal selection and 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. To make the comparison fair, these random input vectors are different from those which are used in signal selection process.

3.2.2 Results

3.2.2.1 Restoration quality

Table 3-5 presents the restoration ratios of our approach compared with previous techniques [7, 25] using the 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 in most cases; in particular we

Table 3-5. Restoration ratios using our approach compared with existing selection approaches

Circuit	Flip-flops	Buffer Width	Simulation-based [7]	Hybrid [25]	Our Approach	Improvement over the best
		8	**13.41**	13.32	14.63	9.1%
s5378	179	16	**7.35**	7.26	9.26	26.0%
		32	**4.47**	4.27	5.11	14.3%
		8	13.98	**14.58**	15.97	9.5%
s9234	228	16	8.30	**8.55**	9.32	9.0%
		32	4.46	4.46	5.53	24.0%
		8	26.33	**27.38**	45.89	67.6%
s15850	597	16	19.89	**20.65**	25.82	25.0%
		32	13.19	13.19	13.97	5.9%
		8	35.52	**39.21**	52.22	33.2%
s13207	669	16	20.13	**22.47**	34.89	55.3%
		32	11.25	**12.52**	16.37	30.8%
		8	19.73	**25.87**	159.1	515.0%
s38584	1452	16	28.39	**29.01**	48.39	66.8%
		32	32.45	**34.62**	44.46	28.4%
		8	29.23	**51.01**	53.47	4.8%
s38417	1636	16	17.02	**19.22**	26.87	39.8%
		32	**15.14**	13.25	17.22	13.7%
		8	132.00	**139.52**	185.1	32.7%
s35932	1728	16	67.45	**71.36**	93.2	30.6%
		32	34.63	**35.08**	47.13	34.4%

achieve improvement in restoration performance is up to 515% (in *s38584*). Note however that the restoration ratio is heavily dependent on the circuit structure, and such high restoration in isolated cases may be an anomaly. Nevertheless, our approach performs better on most cases, with an improvement of 51.23% in restoration quality. Compared to original simulation-based signal selection [7], our fine-grained pruning reduces the chance of removing effective flip-flops prior to selection itself. On the other hand hybrid selection [25] incorporate simulations for only top 5% of the candidate flip-flops, which sacrifices the precision of the selection process; our approach performs better by addressing this weakness through refinement.

3.2.2.2 Selection time

In addition to restoration ratio, we compared the runtime between our approach and Chatterjee *et al.* [7]. Figure 3-3 illustrates the selection time of our approach compared and normalized to [7] using different ISCAS'89 benchmarks. Since selection complexity of [7] is

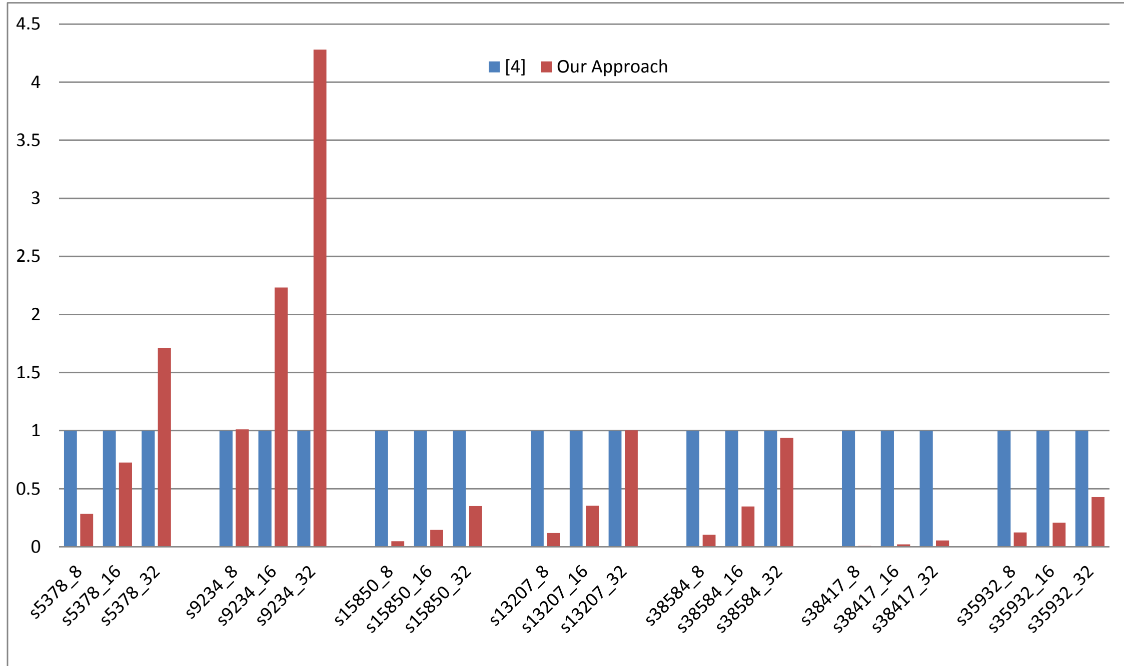


Figure 3-3. Selection times of our approaches compared and normalized to Chatterjee *et al.* [7]

$O(N^2)$ ($\Omega(N^2/50)$ in the best case) and ours is $\theta(Npw)$, as expected, for smaller benchmarks where pw is comparable to or larger than N our approach takes comparable time or longer than [7] (for example *s5378* benchmark and buffer width of 16 and 32 respectively). However, our approach demonstrates consistent speed-up for larger benchmarks (*s15850, s13207, s38584, s38417, and s35932*). The reason is that even after pruning phase of [7], number of conducted simulations in [7] is significantly larger than our approach. In fact, once p and w are fixed, our approach grows linearly with respect to number of flip-flops in circuit. In short, our approach not only produces better restoration quality, but also it is more feasible in terms of selection runtime in large circuits. This makes our approach a better fit for large-scale industry circuits where $N >> pw$. Our signal selection time speed-up is up to 127.6X (in *s38417* with buffer width of 8) and 12.9X on average. Note however that the hybrid approach of Li *et al.* [25] also reports significant speed-up over simulation-based techniques. However, their runtime results are reported for a multi-threaded implementation running on a specific quad-core machine, and are difficult to reproduce in our framework to provide a fair comparison.

3.3 Summary

Post-silicon validation is an expensive phase in the production of integrated circuits, and crucially depends on signal selection to effective use of the limited available observability. Thus it is critical to develop 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 simulation-based signal selection technique that yields signals with higher restorability than current approaches while still being computationally efficient. Our key contribution is the observation that simulation-based signal selection can be significantly improved by augmentation through ILP-based refinement, together with the insights to smoothly integrate the augmentation phase into the selection framework resulting in a unified scalable infrastructure. Our experiments demonstrate that our approach provides up to 515% (51.23% on average) improvement in restoration ratio compared to existing signal selection techniques.

CHAPTER 4
EFFICIENT COMBINATION OF TRACE AND SCAN SIGNALS

To improve the observability in post-silicon debug further, various approaches [10, 11] explored a profitable combination of trace and scan signals. The idea is to divide the trace buffer (width) into two parts. The first part stores the trace signals and the second part stores the scan signals. There is a very small set of important control signals that would be traced every cycle. The remaining slots of the trace buffer will be filled with a portion of a large set of scan signals that would be dumped across several cycles. Existing approaches divide signals into two extreme categories - very important and less important. They lose opportunity from scenarios where some other partitioning is useful such as very important, important, less important, and so on.

It would be beneficial if we divide the signals in a large number of categories in terms of dumping period. This enables us to select a promising signal with a profitable dumping period. In this chapter, we propose an efficient fine-grained architecture that shares the trace buffer bandwidth between several scan chains with different dumping periods. We also propose two different signal selection algorithms which can be used based on hardware constraints. Our signal selection algorithms assign the signals to different scan chains in order to maximize the number of states that can be restored.

Figure 4-1 shows a simple circuit with 8 flip-flops [11] with associated shadow flip flops. Shadow flip-flops are shown in shades. For example, *DS* is the shadow flip-flop for *D*. In addition, proposed debug architecture is shown using two scan chains. Shadow flip-flops of an identical scan chain are shown in same pattern. For example, *AS* and *FS* create a scan chain of length 2 which is shown in dotted pattern. We use this example to show the benefit of using different fine-grained scan chains.

Let us assume that the trace buffer width is 2 which means state of only two signals can be stored in each clock cycle. Table 4-1 shows the signal states that can be restored using the selected signals by [11]. Traced signals are shown in shades/bold. Signal C is traced every

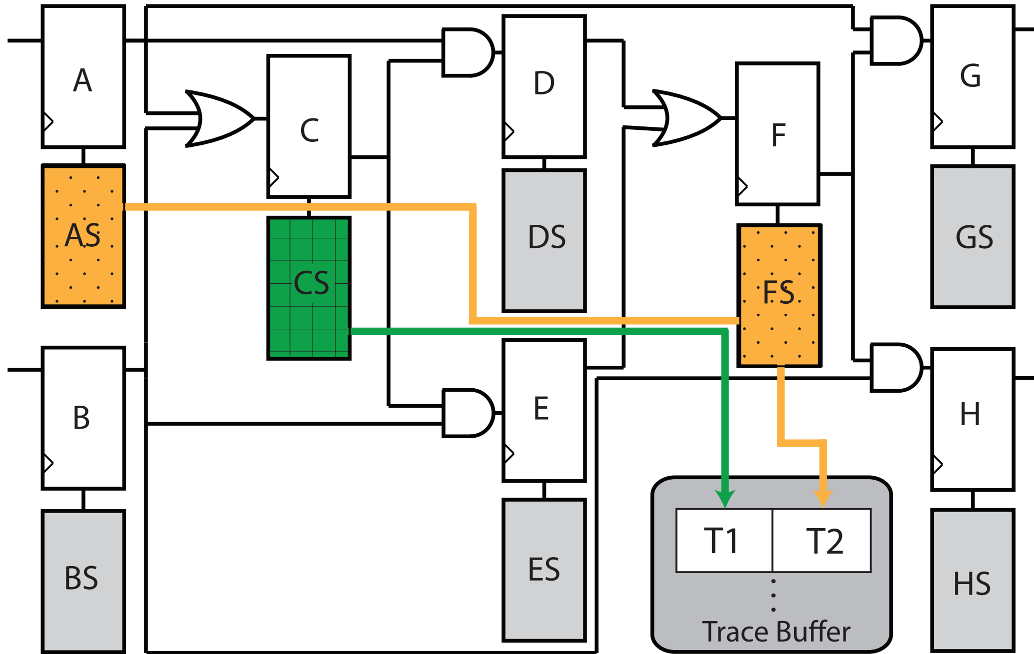


Figure 4-1. Simple circuit to illustrate restorability used in Basu et al. [3]

cycle whereas A and F are dumped in alternate cycles. Although scan signals are dumped in alternate cycles, the table shows states for both A and F in cycle 1, cycle 3, and so on. This is because in cycle 1 the state of signal A is dumped whereas in cycle 2 the state of signal F is dumped. However, the scan chain (i.e., A and F shadow flip-flops) holds the state for the same cycle, although different parts were dumped in different cycles. In other words, the signal state of F captured at cycle 1 is dumped in cycle 2. Forward and backward restoration are used to reconstruct the values for the signals that were not traced. For example, the entry corresponding to D in cycle 2 will be '0' because C was '0' in cycle 1 (forward restoration). The symbol 'X' represents the state that cannot be restored using known signal states. It can be seen that in this case a total number of 36 states can be restored and a total number of 16 states are traced. Therefore the restoration ratio is 2.25.

We now show how different scan chains can help in signal restoration using the same circuit. Figure 4-2 shows an illustrative example of our proposed partitioning of trace buffer of width 2 for the same circuit. It can be observed that trace buffer width is shared between

Table 4-1. Restored signals for the circuit shown in Figure 4-1

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

Table 4-2. Trace buffer slots and shadow flip-flops values in our proposed debug architecture in Figure 4-2

Buffer	Cycle 1	Cycle 2	Cycle 3	Cycle 4	Cycle 5	Cycle 6	Cycle 7	Cycle 8
CS	C_1	A_1	C_3	A_3	C_5	A_5	C_7	A_7
AS	A_1	A_1	A_3	A_3	A_5	A_5	A_7	A_7
ES	E_1	B_1	B_1	E_4	B_4	B_4	E_7	B_7
DS	D_1	E_1	B_1	D_4	E_4	B_4	D_7	E_7
BS	B_1	B_1	B_1	B_4	B_4	B_4	B_7	B_7
T1	C_1	A_1	C_3	A_3	C_5	A_5	C_7	A_7
T2	D_1	E_1	B_1	D_4	E_4	B_4	D_7	E_7

two different scan chains of length 2 and length 3 showed in dotted and grid patterns. In this case there are no trace signals and the buffer width is partitioned between two scan chains. We apply our method to select efficient signals of the sample circuit for this debug architecture. Consequently, we assign signals A and C to the first scan chain while signals B, D, and E to the second scan chain. Scan chains consist of corresponding shadow flip-flops of selected signals. Table 4-2 shows the values of trace buffer and shadow flip-flops in each cycle. The subscript indicates the value in that clock cycle. For example, A_3 implies the value of flip-flop A in cycle 3. It can be observed that signals A and C are stored in trace buffer in alternate cycles whereas B, D, and E are dumped in every third cycle. In other words, signals A and C are dumped with period (T) equals to 2 while *dumping period* for signals B, D, and E is equal to 3.

Table 4-3 shows the signal states that can be restored using the signals chosen by our method (described in Section 4.1). It can be seen that a total number of 55 states can be

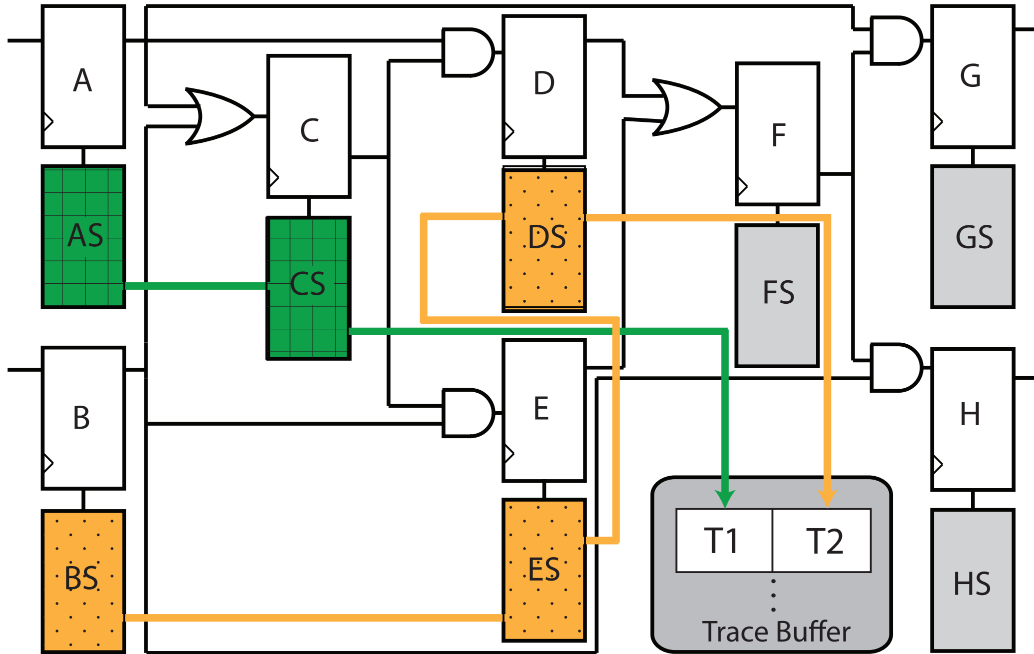


Figure 4-2. Proposed debug architecture for example circuit in Figure 4-1

Table 4-3. Restored signals using our proposed debug architecture in Figure 4-2

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

restored and a total number of 17 states are traced. The restoration ratio is 3.24 which is higher than 2.25 [11]. Thus, more states give a more detailed view of the internal state of the circuit.

The primary problem of using different scan chains is to determine the length of scan chains and signals to select for each scan chain. Signals should be chosen such that more important signals are assigned to smaller scan chains with small dumping period. Less important signals on the other hand, should be distributed among the scan chains with larger dumping periods. In addition, minimizing the overlaps between the states that can be restored

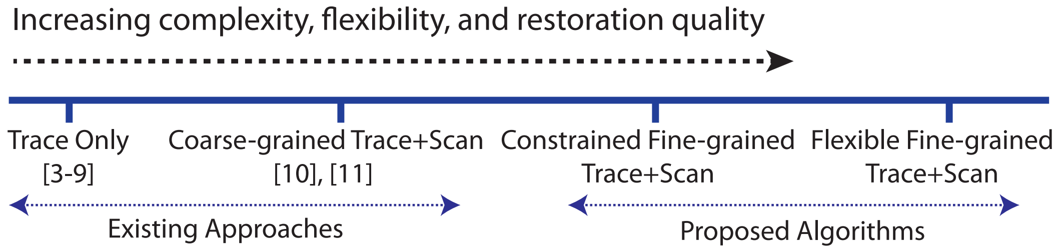


Figure 4-3. Spectrum of existing and proposed debug architectures

by different scan chains should be considered in selection algorithm in order to maximize the restoration ratio in a debug scenario. In this chapter, we propose two algorithms to select profitable signals for each scan chain.

4.1 Fine-grained Combinations

In this section, we first propose our fine-grained debug architecture. Next, we present signal selection algorithms for constrained and flexible debug architectures. In constrained debug architecture, the length of each scan chain is determined prior to the signal selection process. This approach is used when the designer fixes the scan chains lengths or there is hardware constraints in the system. The goal of constrained hardware signal selection algorithm is to assign the best possible signals to each scan chain. On the other hand, in flexible hardware architecture there are no constraints on the and types (length) of scan chains. Therefore, the primary objective of flexible selection algorithm is to maximize the restorability, regardless of scan chains lengths. Figure 4-3 illustrates the spectrum of existing and proposed architectures. It can be observed that trace only and flexible hardware architectures are two extremes of this spectrum. Trace only is the simplest with less complexity, while flexible hardware architecture is more efficient in terms of observability but may introduce minor hardware overhead.

4.1.1 Debug Architecture

Our fine-grained architecture is motivated by the coarse-grained design of [10, 11]. They proposed an architecture that divides the trace buffer into two parts, one for trace signals and the other one for scan signals. However, this partitioning is coarse-grained. Very important

signals are traced every cycle whereas the less important ones are assigned to a scan chain. The dumping period for scan signals depends on trace buffer width and number of signals in the scan chain. Putting more signals in scan chain increases the dumping period for signals. On the other hand, putting less signals may not be desirable as it decreases the coverage of the circuit. As discussed earlier, the limitation of the existing approaches is to consider only two extremes and losing the opportunity for not considering in-between scenarios. We consider a fine-grained approach by allowing multiple partitions of trace buffer width.

Figure 4-4 illustrates our proposed fine-grained architecture. It can be observed that width *bw* of the trace buffer is partitioned for ω trace signals and n different scan chains i.e., $bw = \omega + n$. Each of these scan chains comprises of different number of signals denoted by identical color which determines the dumping period for those signals. In each cycle, the shadow flip-flops of a particular scan chain capture the value of their corresponding flip-flops. It has to be noted that if a particular scan chain contains only one signal it is essentially a trace signal that is traced every cycle. These different signal chains provide more fine-grained dumping periods. Thus, each signal can be assigned to the appropriate scan chain based on its importance. These fine-grained scan chains enable us to dump larger number of signals which improves the observability in the circuit compared with coarse-grained scan signals. The next two sections describe two variations (constrained and flexible) of our proposed algorithms which try to select the best signal in each iteration considering all the signals that have been selected before.

4.1.2 Constrained Signal Selection (CSS)

In this section, we propose a greedy heuristics for constrained selection algorithm which selects profitable signals for each scan chain of determined length in order to maximize the observability. We define $P_0(f)$ and $P_1(f)$ for flip-flop f in the circuit that define the probability of its value being 0 and 1, respectively. These values can be calculated by feeding the simulator with circuit graph and random input vectors and running it for numerous times. We also use *connectivity* information similar to [11]. The connectivity of a flip-flop is the number of

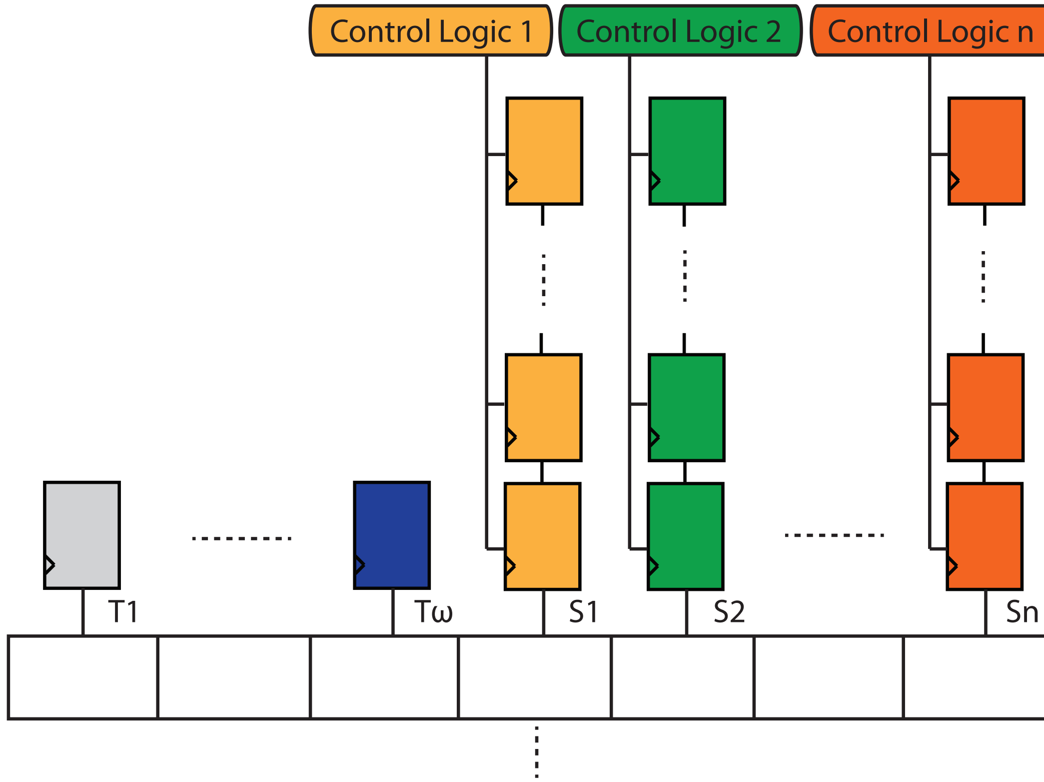


Figure 4-4. Proposed debug architecture

flip-flops connected with it through other combinational gates in both backward and forward directions.

In order to partition the trace buffer in different scan chains we define *buffer width (bw)*, *trace signals* (ω), *partition factor* (α), and *step function* (ϕ). The buffer is divided into two parts. First part consists of ω trace signals that are dumped every cycle. The remaining $bw - \omega$ buffer entries are further divided into α partitions. Each partition consists of $(bw - \omega)/\alpha$ scan chains with identical length. The step function determines the length of scan chains in each partition. In other words, assume l_i and l_{i+1} are the lengths of scan chains in two successive partitions, then we would have: $l_{i+1} = \phi(l_i)$. It has to be noted that l_0 is equal to initial value of 1.

Figure 4-5 illustrates an example of trace buffer partitioning in a debug architecture with $bw = 8, \omega = 2, \alpha = 3$, and $\phi(i) = \phi(i-1) + 1$. It can be seen that there are two trace signals that are dumped every cycle. The rest of the trace buffer is shared between

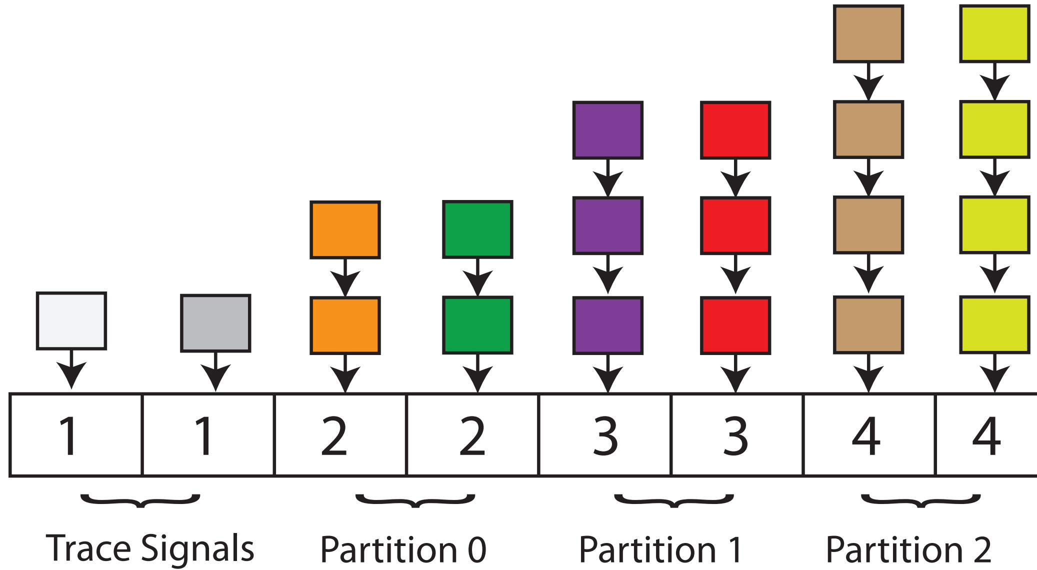


Figure 4-5. An example of trace buffer partitioning

fine-grained scan chains. For example, first partition consists of two scan chains each of them with identical length of 2. In other words, two signals that are assigned to scan chain 1 will be dumped in alternate cycles. We also define *dumping period (T)* for each scan chain. Clearly, dumping period for a particular scan chain is equal to its length. For example, four signals are assigned to last scan chain in Figure 4-5. Each of these signals will be dumped every four cycles. Fine-grained partitioning enables us to assign signals to different dumping periods based on their importance. For example in Figure 4-5, important (control) signals are assigned to trace slots (T=1). On the other hand, less important signals are assigned to scan chains with larger lengths (T=2, T=3, and T=4), based on their impact on the restoration performance.

We use *restoration power (RP)* as a selection metric in our algorithm. Assume S is the current set of assigned flip-flops to scan chains. In addition, f_T implies that flip-flop f is dumped every T cycles. We show the dumped values of f using v that can be either 0 or 1.We define $\delta(S \cup \{f_T\}, v)$ as the number of additional states that can be restored using $S \cup \{f_T\}$ (compared to restored stated using only S) over a window of c cycles when we dump f each T cycles with the assumption that the value of f is fixed to v throughput the dumping cycles. $\delta(S \cup \{f_T\}, v)$ is calculated for both $v = 0$ and $v = 1$. These values are then weighted averaged

based on the probabilities of being 0 or 1 in flip-flop f and are used in the selection metric of our algorithm. Clearly, larger c is more desirable as it yields more precise result. However, large c in real scenarios is not practical as there are numerous number of flip-flops that make the restoration process computationally expensive. Our experimental results demonstrate that $c = LCM(l_0, l_1, ...l_{n-1})$ is large enough where l_i is the length of scan chain i and LCM is least common multiple. The reason is that restoration pattern in whole circuit is repeated over each $c = LCM(l_0, l_1, ...l_{n-1})$ cycles. For example, in Figure 4-5 $c = LCM(1, 2, 3, 4) = 12$ is used in our algorithm. For a particular flip-flop f with a dumping period of T (f_T), RP is defined as follows.

$$RP(f_T) = T * (P_0(f) * \delta(S \cup \{f_T\}, 0) + P_1(f) * \delta(S \cup \{f_T\}, 1))$$

In other words, RP is the number of probable additional states that can be restored by adding f_T to the S. There is a multiplication by T in RP because we would like to take into account the resources that f_T uses for these additional restored states. Larger T means smaller resource usage. In fact, the intuition of RP is that in each iteration we try to choose a flip-flop that makes the best trade-off between the maximum newly restored states and minimum resource usage.

Algorithm 3 outlines the major steps in our constrained signal selection algorithm. First, we create a graph of flip-flops using the same methodology described in [3]. This graph is used to compute the connectivity of each flip-flop. Next, we calculate P_0 and P_1 for each flip-flop, and partition the trace buffer using input parameters. We create an empty list S to hold the list of selected flip-flops. In each iteration, RP is calculated for all the remaining flip-flops (flip-flops that are not in S). If two or more flip-flops have equal RP we choose the one with higher connectivity. This increases the chance of restoring more states during the real debug scenario as it is connected to more flip-flops. We continue assigning one flip-flop in each iteration until all the scan chains get full.

We now show how our algorithm works for the example circuit in Figure 4-1. Assume that trace buffer width is 2. We partition the trace buffer using $bw = 2, \omega = 0, \alpha = 2$, and

Data: *circuit*, bw, ω, α, ϕ
Result: The set of selected signals
Create a graph model of the circuit ;
Calculate P_0 and P_1 for all the flip-flops ;
Create list of selected signals S ▷ initially empty ;
Partition trace buffer using $(bw, \omega, \alpha, \phi)$;
while *there is an available scan chain* **do**
    foreach *flip-flop f that is not in the S* **do**
        foreach *available scan chain of length T* **do**
            Calculate RP for f_T using $S \cup \{f_T\}$;
        end
    end
    Find flip-flop f with dump period T (f_T) that has the maximum RP. If two or more flip-flops have same RP, find the one with higher connectivity assign f to a scan chain with length T ;
    Add f_T to the list S ;
end
return S ;

Algorithm 3: Constrained Signal Selection (CSS)

$\phi(i) = 1 + \phi(i-1)$. In other words, we have two scan chains of length 2 and 3, respectively. Hence, from our algorithm we would have $c = LCM(2, 3) = 6$, which is used in restoration power calculation. Table 4-4 summarizes intermediate results of our algorithm in each iteration. First column is the candidate flip-flops from the example circuit. Second and third columns are P_0 and P_1 of each flip-flop which are calculated by feeding our simulator with 100 different random inputs. The rest of the columns are the restoration power of flip-flops in each iteration. Each cell in these columns contains two rows which are the RP values if we assign the flip-flop to the scan chain of length 2 (T=2) or length 3 (T=3), respectively. In the first iteration, signal C has the highest RP for T=2 and is assigned to the first scan chain of length 2 (the RP value for the selected signal in each iteration is shown in bold). In the second iteration, both signals A and B yield the maximum RP when they are assigned to scan chain of length 2. In addition, since both of them have same connectivity (3), our algorithm selects one of them (signal A) randomly. Till now, signals A and C are assigned to first scan chain of length 2. Using the same procedure, signals B, D and E are assigned to second scan chain of length 3 in the remaining iterations.

Table 4-4. Different restoration power values of our algorithm in example circuit of Figure 4-1

Signal	P_0	P_1	RP (I1)	RP (I2)	RP (I3)	RP (I4)	RP (I5)
A	0.5	0.5	15.00	**10.00**	-	-	-
			15.00	6.00	-	-	-
B	0.5	0.5	15.00	10.00	-	-	-
			15.00	6.00	**13.50**	-	-
C	0.29	0.71	**21.19**	-	-	-	-
			19.15	-	-	-	-
D	0.6	0.4	14.70	4.37	-	-	-
			11.93	8.37	10.20	4.20	**6.00**
E	0.6	0.4	14.68	4.37	-	-	-
			11.92	8.37	13.18	**7.18**	-
F	0.45	0.55	14.99	6.49	-	-	-
			14.09	5.70	10.35	3.00	3.00
G	0.68	0.32	9.80	2.63	-	-	-
			8.85	4.90	9.00	3.00	3.00
H	0.68	0.32	9.81	2.63	-	-	-
			8.86	4.90	9.95	3.95	3.00

From Table 4-4, it can be observed that our algorithm covers a large part of the circuit by assigning more resources to important signals. This procedure continues by assigning less resources to signals that can cover other parts of the circuit. As a result, it gets benefit of both spatial and temporal observability of fine-grained sets of signals.

4.1.3 Flexible Signal Selection (FSS)

In this section, we describe our flexible signal selection algorithm which is used when there are no pre-defined architectural constraints on scan chains. FSS starts with initially empty scan chains. In each iteration, a flip-flop is added to one of the scan chains. This process stops once adding more flip-flops is not profitable anymore. Algorithm 4 outlines the major steps of proposed flexible selection algorithm. The inputs of the algorithm are the *circuit, trace buffer width (w),* and *the number of cycles in mock simulations (c)*. To understand the workings of the algorithm we need a key concept: *restoration impact*.

Given a scan chain configuration s, an input vector I, and the number of mock simulation cycles c, we define the *restoration impact* $RI(s, I, c)$, as the total number of states that can be restored if we do a mock simulation over c cycles using input vector I and the scan chain

configuration s. For a given scan chain configuration s and flip-flop f, $s \cup \{f_k\}$ is a new configuration which is same as s except that flip-flop f is added to k_{th} scan chain of s.

```
Data: circuit, bw, c
Result: The set of selected signals
Create a graph model of the circuit ;
Create initial configuration S                    ▷ empty scan chains ;
continue = true ;
while continue do
    Generate a random input vector I ;
    currentImpact = RI of (S, I, c) ;
    foreach each flip-flop f that is not in the S do
        for k_th scan chain; 1 ≤ k ≤ w do
            Calculate RI of (S ∪ {f_k}, I, c) ;
        end
    end
    Find f_k with maximum RI. If two or more flip-flops have same RI, find the one with
    higher connectivity ;
    if RI of S ∪ {f_k} > currentImpact then
        Add f to k_th scan chain of S ;
        continue = true ;
    end
    else
        continue = false ;
    end
end
return S ;
```

Algorithm 4: Flexible Signal Selection (FSS)

Informally, for a given c-cycle mock simulation on I, the restoration impact shows how scan chain configuration s is profitable in terms of observability performance. In particular, if $s_2 = s_1 \cup \{f_k\}$ for some design signal f, then $RI(s_2, I, c) - RI(s_1, I, c)$ measures the observability improvement achieved by augmenting k_{th} scan chain of s_1 with f. Algorithm 4 is a greedy heuristics that uses this metric to iteratively grow the set S of current scan chain configuration. At each iteration, it (1) performs a new simulation for c cycles using a random input vector I and computes the restoration impact of current configuration S, (2) computes the restoration impact of $S \cup f_k$ for each design signal f which is not selected before when f is added to k_{th} scan chain, and (3) augments k_{th} scan chain of S with the signal that maximizes

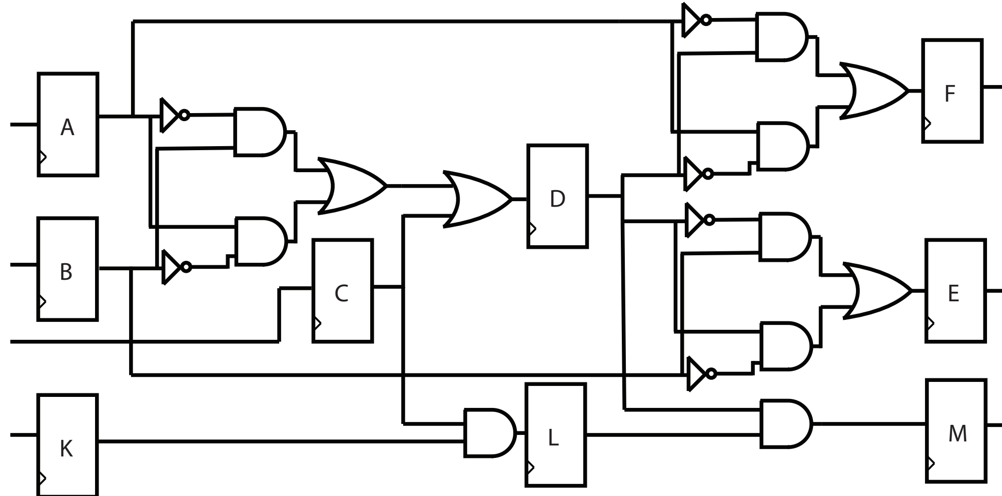


Figure 4-6. Sample circuit to illustrate our flexible selection algorithm

the restoration impact, if it increases the restoration impact of current configuration. If two or more signals have identical restoration impact, then the tie is broken in favor of the signal that has the highest connectivity. The process is continued until augmenting scan chains is not profitable anymore.

We now show how our algorithm works for the example circuit in Figure 4-6. Assume that trace buffer width is 2 and $c = 32$ cycles is used in mock simulations of the algorithm. Table 4-5 summarizes intermediate results of our algorithm in each iteration. Second row shows the current configuration S in each iteration and third row demonstrates the corresponding restoration impact. The remaining rows are the restoration impact values of the circuit flip-flops in each iteration. Each cell in these rows contains two numbers which are the RI values if we add the flip-flop to the first or second scan chain of current configuration, respectively. In the first iteration, signal *L* has the highest RI and is added to the first scan chain (the RI value for the selected signal in each iteration is shown in bold). Using the same procedure, signals *A, D, B,* and *C* are added to the second scan chain in the following iterations. It can be observed that in the last iteration adding a new flip-flop to any of the scan chains is not profitable anymore. In other words, none of the RI values is greater than the RI of

Table 4-5. Restoration impact values of our flexible selection algorithm in example circuit of Figure 4-6

Variable	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5	Iteration 6
S	{}	{L}	{L}	{L}	{L}	{L}
	{}	{}	{A}	{A,D}	{A,D,B}	{A,D,B,C}
currentImpact	0	84	123	130	142	144
A	32	64	-	-	-	-
	32	**127**	-	-	-	-
B	32	60	117	128	-	-
	32	122	129	**142**	-	-
C	81	78	119	136	142	-
	81	121	123	137	**149**	-
D	60	61	114	-	-	-
	60	118	**141**	-	-	-
E	32	66	92	105	125	130
	32	122	119	133	133	134
F	32	60	127	103	116	123
	32	116	122	123	146	128
K	66	84	115	125	148	144
	66	113	114	131	140	133
L	**96**	-	-	-	-	-
	96	-	-	-	-	-
M	80	77	97	130	149	144
	80	88	98	131	140	131

current configuration *S*. The algorithm stops with *L* assigned to the first scan chain and *A,D,B,* and *C* assigned to the second scan chain.

It can be observed that our flexible selection algorithm is different from constrained selection in terms of selection goal and time. The flexible selection continues augmenting scan chains with new flip-flops until adding new flip-flops degrades restoration performance. In other words, there are no constraints on scan chain lengths as they can grow when profitable. On the other hand, the goal of the constrained selection (discussed in Section 4.1.2) is to find the best possible fit for a fixed hardware architecture. In other words, lengths of various scan chains are fixed in constrained selection. The constrained hardware may degrade the restoration performance compared to flexible hardware as we put constraint on scan chain

lengths. However, constrained signal selection can reduce the selection time and hardware overhead.

4.2 Experiments

4.2.1 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 conducts restoration in both forward and backward directions following the mechanism outlined in [4]. The simulator iterates on the unknown signals queue and tries to restore them using both forward and backward directions. 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 same circuits using *Icarus Verilog* [70].

We fed the simulator with 100 sets of random values and noted the average restoration ratios. 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. Table 4-6 shows the set of parameters that we used for each benchmark and different trace buffer widths for constrained signal selection algorithm.

In addition, we used $c = 64$ cycles in mock simulations of flexible signal selection algorithm. Our experiments show that by using $c = 64$, relative restoration performance is consistent between different set of trace signals and input vectors. In other words, $c = 64$ is enough to remove the random behaviors from our selection process. Important signals always perform better compared to other signals (for almost all random input vectors). Therefore, these signals are always selected in final result. We use different random signals to make sure that different states of the circuits are covered (different areas enabled). These random signals may affect borderline signals in the final result. However, the effect on restoration performance in borderline signals is negligible.

Table 4-6. Different parameters used in CSS

Circuit	Buffer Width	ω	α	ϕ
	8	4	1	$\phi(i) = 1 + \phi(i-1)$
s5378	16	8	4	$\phi(i) = 2 * \phi(i-1)$
	32	8	3	$\phi(i) = 1 + \phi(i-1)$
	8	4	4	$\phi(i) = 2 * \phi(i-1)$
s9234	16	8	4	$\phi(i) = 2 * \phi(i-1)$
	32	12	4	$\phi(i) = 2 + \phi(i-1)$
	8	2	3	$\phi(i) = 2 * \phi(i-1)$
s15850	16	2	7	$\phi(i) = 1 + \phi(i-1)$
	32	8	6	$\phi(i) = 1 + \phi(i-1)$
	8	2	3	$\phi(i) = 2 + \phi(i-1)$
s38584	16	4	3	$\phi(i) = 2 + \phi(i-1)$
	32	8	3	$\phi(i) = 1 + \phi(i-1)$
	8	2	3	$\phi(i) = 2 * \phi(i-1)$
s38417	16	8	4	$\phi(i) = 2 * \phi(i-1)$
	32	16	4	$\phi(i) = 2 + \phi(i-1)$
	8	4	1	$\phi(i) = 1 + \phi(i-1)$
s35932	16	8	1	$\phi(i) = 1 + \phi(i-1)$
	32	16	1	$\phi(i) = 1 + \phi(i-1)$

We used the original reported restoration quality numbers in [5] and [3]. We did not use the reported numbers of [25] and [7] as they used modified (performed some optimizations) version of ISCAS89 benchmarks. To perform a fair comparison, we tried to obtain the executables of [25] and [7]. Li et al. [25] provided us with their signal selection framework and we used it for the selection process in this revision. Unfortunately we were not able to get the implementation of [7] and we used our implementation of their approach in this revision. It should be noted that we used our framework for simulations and restoration quality calculations.

4.2.2 Restoration Quality

Table 4-7 compares the restoration ratios of our approach with several previous trace only techniques [3, 5, 7, 25] using different ISCAS'89 benchmarks. The trace buffer used in our experiment are $8 \times 4k$, $16 \times 4k$, and $32 \times 4k$. The corresponding restoration ratio for each technique (if available) is reported. Seventh and ninth columns indicate the percentage improvement using CSS and FSS techniques respectively, compared with the best (shown in

Table 4-7. Restoration ratios of different trace only approaches compared with our proposed architecture

Circuit	Buffer Width	Ko et al. [5]	Basu et al. [3]	Chatterjee et al. [7]	Li et al. [25]	CSS	CSS Imprv.(%) over best	FSS	FSS Imprv.(%) over best	FSS Imprv.(%) over CSS
	8	**14.67**	-	13.24	14.35	14.65	0	14.84	1	1
s5378	16	**8.99**	-	7.83	8.36	8.64	-4	9.33	4	8
	32	4.72	-	4.89	**4.99**	5.00	0	5.30	6	6
	8	4.76	-	**10.68**	9.25	20.43	91	24.28	127	19
s9234	16	**7.18**	-	7.16	6.13	12.31	71	12.93	80	5
	32	**4.67**	-	4.18	4.38	6.78	45	6.92	48	2
	8	19.93	-	**39.54**	21.90	47.35	20	50.96	29	8
s15850	16	24.22	-	**24.85**	14.78	26.00	5	26.87	8	3
	32	13.30	-	**13.60**	10.88	14.71	8	15.23	12	4
	8	19.23	78.00	**84.10**	27.00	146.64	74	159.65	90	9
s38584	16	13.96	40.00	**47.04**	13.97	80.85	72	85.90	83	6
	32	8.68	20.00	**26.97**	7.50	43.22	60	45.00	67	4
	8	18.63	**55.00**	45.21	37.71	55.40	1	56.41	3	2
s38417	16	18.62	29.00	**30.77**	23.80	33.41	9	33.73	10	1
	32	14.20	16.00	**20.25**	11.83	21.33	5	22.11	9	4
	8	64.00	95.00	96.12	**144.00**	178.51	24	186.60	30	5
s35932	16	38.13	60.00	67.45	**72.00**	89.25	24	94.80	32	6
	32	21.06	35.00	**43.23**	36.00	45.01	4	47.85	11	6

bold) result provided by existing approaches. The improvement in restoration performance is up to 91% in *s9234* (28% on average) for constrained selection algorithm. Likewise, this improvement is up to 127% in *s9234* (36% on average) for flexible selection algorithm. It can be observed that our approach performs significantly better because existing trace only approaches only take advantage of temporal observability of a small set of signals while miss the opportunity of both spatial and temporal observability of a large set of signals. The last column shows the percentage improvement by relaxing the scan chain lengths (FSS) over constrained hardware approach (CSS). As expected, FSS outperforms CSS approach as there is no constraints on scan chains hardware. This improvement is up to 19% in *s9234* and 5% on average.

We also compared our approach with the existing trace+scan approach proposed by Basu et al. [11] in Table 4-8. It is important to note that we did not compare with other trace+scan approaches (such as [10]) since [11] has shown to perform better than other approaches. It can be observed that our approach outperforms [11] consistently. The improvement in restoration performance is up to 116% in *s38584* (54.7% on average) for CSS, and up to 125% in *s38584*

Table 4-8. Restoration ratios using our approach compared with Basu et al. [3]

Circuit	Buffer Width	Basu et al. [11]	CSS	CSS Imprv.(%)	FSS	FSS Imprv.(%)
s38584	32	20.00	43.22	116	45.00	125
s38417	32	18.00	21.33	19	22.11	23
s35932	32	35.00	45.01	29	47.85	37

(61.7% on average) for FSS. The reason for significant improvement is that their approach is limited by coarse-grained partitioning (two fixed partitions) of signals.

4.2.3 Selection Time

Table 4-9 presents the runtime of our approach compared with previous simulation-based/hybrid techniques [7, 25] using different ISCAS'89 benchmarks. The reported runtime format is 'hour:minute:second'. For this comparison, we used an Ubuntu 12.04.5 machine with a 64-Core AMD Opteron 6378 (1400 MHz) processor and 189 GB of physical memory. We measured the runtime of Li et al. [25] using their provided multi-thread binary file. In addition, to make the comparison fair, we used a multi-thread implementation of both [7] and our approach. It can be observed that compared to [7], our approach reduces the selection runtime significantly. Compared to [25], our approach demonstrated significant improvement in restoration quality, the runtime (of CSS, more specifically) is still comparable. In addition, since CSS enforces constraints on scan chain lengths, as expected, it demonstrates consistent speed-up compared with FSS approach. In short, although FSS outperforms CSS in terms of restoration performance, it needs more time to run the signal selection. CSS is beneficial when there is constraints on hardware architecture or selection time. On the other hand, FSS is beneficial when the restoration quality is the primary objective. In addition, in each step of our approach all the evaluations (RP in CSS and RI in FSS) can be done simultaneously. This makes our approach scalable for large industrial designs by using MapReduce or similar programming models.

4.2.4 Hardware Overhead

In order to investigate the hardware overhead of our approach, we developed Verilog Register-Transfer Level (RTL) design for each of the profitable scan-trace configurations

Table 4-9. Runtime comparison of different approaches

Circuit	Buffer Width	Chatterjee et al. [7]	Li et al. [25]	CSS	FSS
	8	00:00:14	00:00:04	00:00:01	00:00:04
s5378	16	00:00:14	00:00:08	00:00:19	00:00:33
	32	00:00:13	00:00:11	00:00:17	00:00:55
	8	00:01:06	00:00:07	00:00:15	00:00:28
s9234	16	00:01:05	00:00:19	00:00:29	00:00:51
	32	00:01:01	00:00:22	00:00:44	00:01:06
	8	00:28:01	00:01:40	00:00:45	00:01:17
s15850	16	00:28:00	00:01:58	00:04:11	00:07:02
	32	00:27:57	00:02:07	00:05:32	00:06:00
	8	03:20:22	00:01:20	00:02:57	00:06:45
s38584	16	03:20:17	00:05:40	00:06:29	00:15:08
	32	03:19:52	00:08:30	00:13:47	00:31:59
	8	02:49:00	00:07:10	00:01:42	00:24:03
s38417	16	02:49:00	00:33:30	00:09:25	00:21:18
	32	02:49:00	00:35:30	00:11:55	00:13:22
	8	01:27:27	00:01:21	00:01:28	00:09:04
s35932	16	01:27:23	00:03:36	00:03:14	00:10:31
	32	01:27:15	00:04:47	00:05:59	00:15:48

(shown in Table 4-10) and performed logic synthesis. For each benchmark design, we selected the most profitable scan-trace combinations for different trace buffer widths. We fixed the trace buffer depth to 128 for all the experiments. The RTL design has been developed as a standalone module that accepts the scan-trace configuration as a parameter. In each scan chain, shadow flip-flops are connected through a chain and are stored in the trace buffer. For example, Figure 4-7 shows a sample configuration for a trace buffer of width 4. In this example, each entry of the trace buffer is connected to a specific scan chain. The first entry is connected to a scan chain of length 1 consisting of only one flip-flop (A). Similarly, the second entry is connected to a scan chain consisting of a flip-flop (B). In other words, A and B are essentially trace signals that are traced every cycle. The third entry is connected to a scan chain of length 2 consisting of flop-flops C and D that are traced in alternate cycles. The last entry is connected to a scan chain of length 3 consisting of flip-flops E, F and G that are traced in every third cycles. This configuration can be termed as *2T-1S2-1S3* to indicate that it consists of two trace signals (2T), one scan chain of two flip-flops (1S2) and one scan chain of three flip-flops (1S3).

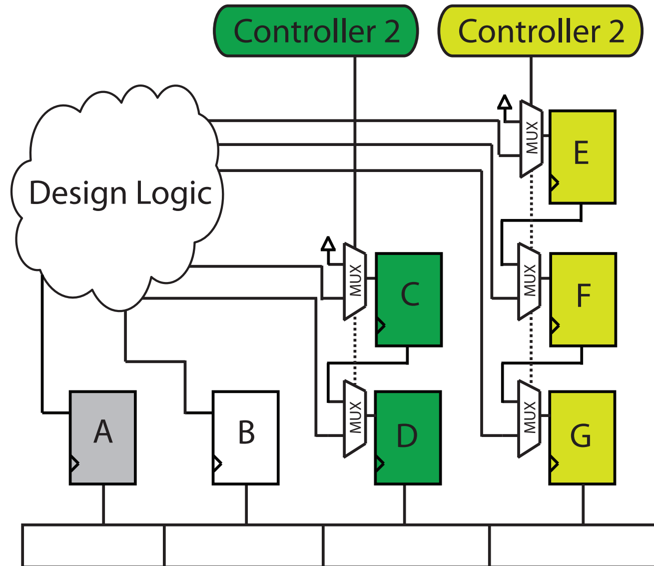


Figure 4-7. Sample scan-trace configuration

Table 4-10. Configurations for fine-grained architectures

Circuit	Buffer Width	Fine-grained Architecture
	8	7T-1S5
s5378	16	10T-1S2-4S3-1S4
	32	21T-4S2-6S3-1S6
	8	6T-1S3-1S5
s9234	16	12T-3S2-1S10
	32	26T-4S2-2S3
	8	3T-4S2-1S4
s15850	16	9T-4S2-3S3
	32	29T-2S2-1S3
	8	7T-1S2
s38584	16	15T-1S2
	32	30T-2S2
	8	2T-3S2-1S3-2S4
s38417	16	11T-4S2-1S5
	32	30T-2S2
	8	6T-2S3
s35932	16	13T-3S2
	32	30T-2S2

Table 4-10 shows the profitable configurations used in different benchmarks. The first column indicates the benchmark. The second column provides different trace buffer widths. The last column lists the configurations. For example, consider the configuration *21T-4S2-6S3-1S6* for benchmark s5378 with buffer width 32. This configuration consists of 21 trace signals (21T), 4 scan chains each having 2 flip-flops (4S2), 6 scan chains each having 3 flip-flops (6S3) and 1 scan chain of 6 flip-flops (1S6).

We present both area and power overhead for performing fine-grained signal selection compared to existing trace-only approaches in Table 4-11 and 4-12, respectively. The area overhead computation includes the area of all the required scan registers (shadow flip-flops that capture the state of the signals) as well as the necessary logic to control the timing of the signal storage for each scan chain. Scan control logic in this case is essentially a counter for dumping the values of each scan chain by generating suitable control signals to enable shift operation in each cycle. In order to synthesize the designs, we used Synopsys Design Compiler version F-2011.09 and FreePDK45 target library (45 nm technology) [1].

Table 4-11 presents the hardware area overhead of our approach for different benchmarks and trace buffer widths. The first and second column indicate the benchmarks and trace buffer widths, respectively.

The third column presents the area (in μm^2) for the circuit including the trace buffer. The fourth column presents the area (in μm^2) of the circuit with trace buffer including our fine-grained trace controller. The last column indicates the percentage of area overhead which is calculated as *100* $\times$*(column 4 - column 5) / column 4*. It can be observed that in most of the cases the area overhead of our approach is negligible. This overhead is up to 1.49% in *s9234* and 0.57% on average. It can also be observed that the area overhead is more noticeable in smaller circuits (*s5378*, *s9234*, and *s15850*). This can also be seen in Figure 4-8 which shows the area overhead (in percentage) of different circuit sizes (in terms of number of flip-flops) for different trace buffer widths. The reason is that the size of the trace buffer is dominant in these scenarios. However, the area overhead is much smaller for larger benchmarks where the circuit area is dominant compared to the trace buffer area. Therefore, area overhead of our approach would be negligible when fine-grained signal selection is applied on large industrial designs with larger trace buffers.

Table 4-12 presents the power overhead of our approach for different benchmarks and trace buffer widths. Power numbers include both dynamic and leakage power. Dynamic power is computed assuming frequency of 500 MHz. The first and second column indicate

Table 4-11. Area overhead of fine-grained signal selection compared to trace-only approach

Circuit	Buffer Width	Circuit with Trace Buffer	Circuit + Fine-grained	Area Overhead (%)
	8	21699.96	21906.92	0.95
s5378	16	39215.17	39639.42	1.08
	32	72404.07	73121.63	0.99
	8	20819.56	21128.83	1.49
s9234	16	38334.77	38770.28	1.14
	32	71523.67	71811.35	0.40
	8	28475.25	28779.82	1.07
s15850	16	45990.46	46331.17	0.74
	32	79179.36	79331.88	0.19
	8	49496.60	49556.67	0.12
s38584	16	67011.81	67063.44	0.08
	32	100200.71	100248.58	0.05
	8	49038.57	49491.44	0.92
s38417	16	66553.78	66871.50	0.48
	32	99742.68	99790.55	0.05
	8	45900.83	46068.37	0.37
s35932	16	63416.04	63522.57	0.17
	32	96604.94	96652.81	0.05

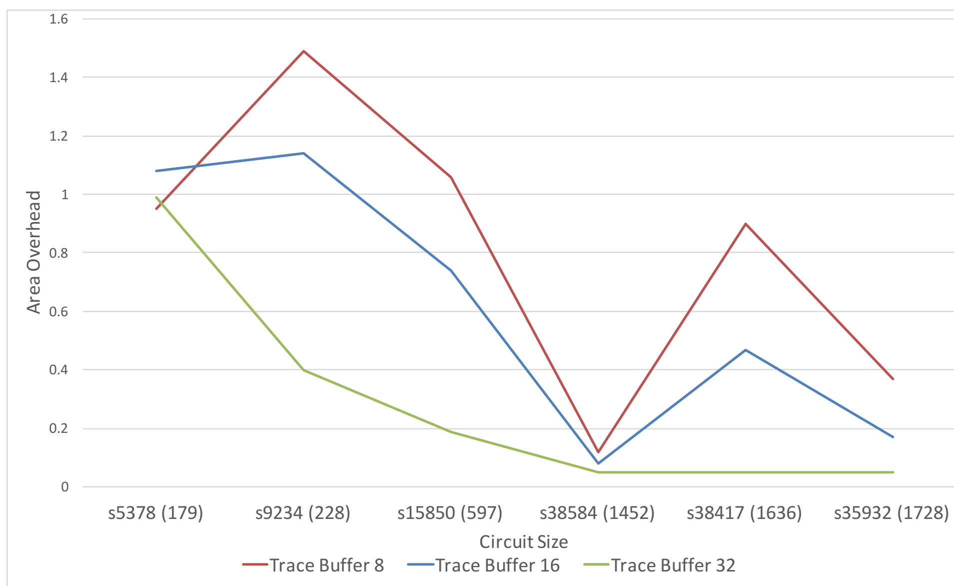


Figure 4-8. Area overhead of our approach for different circuit and trace sizes, benchmarks are ordered based on the number of flip-flops (shown in brackets)

the benchmarks and trace buffer widths, respectively. The third column presents the power (in mW) for the circuit including the trace buffer. The fourth column presents the power (in mW) of the circuit with trace buffer including our fine-grained trace controller. The last column indicates the percentage of power overhead which is calculated as *100* $\times$*(column 4 - column 5) / column 4*. It can be observed that in most of the cases the power overhead of our approach is negligible (1% or less). It can also be observed that the power overhead is more noticeable in smaller circuits (*s5378*, *s9234*, and *s15850*). The reason is that the size of the

Table 4-12. Power overhead of fine-grained signal selection compared to trace-only approach

Circuit	Buffer Width	Circuit with Trace Buffer (mW)	Circuit + Fine-grained (mW)	Power Overhead (%)
	8	9.35	9.50	1.60
s5378	16	16.88	17.20	1.90
	32	31.92	32.50	1.82
	8	8.79	9.01	2.50
s9234	16	16.32	16.65	2.02
	32	31.32	31.58	0.83
	8	12.05	12.30	2.07
s15850	16	19.58	19.90	1.63
	32	34.62	34.80	0.52
	8	21.45	21.50	0.23
s38584	16	28.98	29.04	0.21
	32	44.02	44.10	0.18
	8	22.15	22.50	1.58
s38417	16	29.68	30.00	1.08
	32	44.72	44.80	0.18
	8	25.55	25.70	0.59
s35932	16	33.08	33.20	0.36
	32	48.12	48.20	0.17

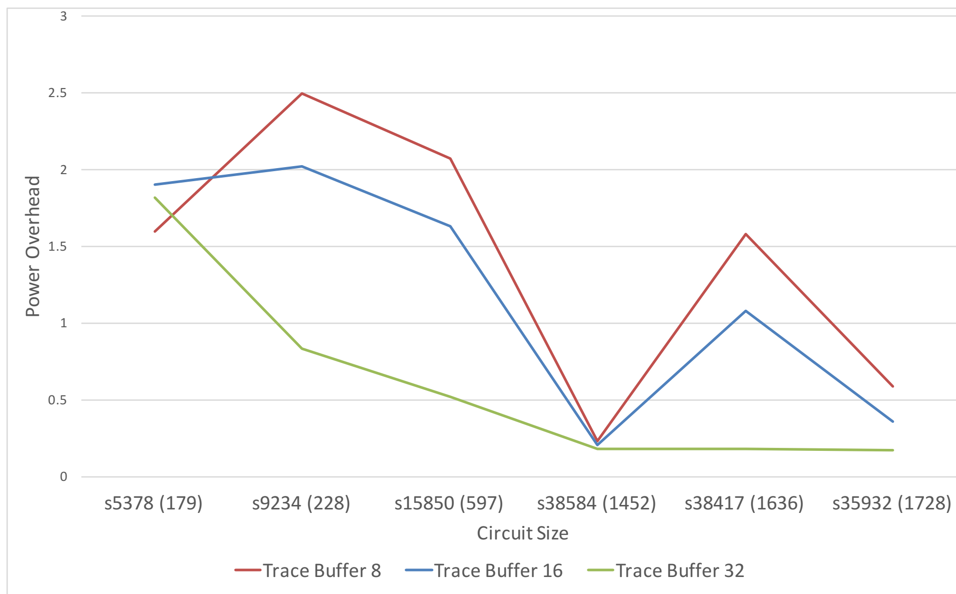


Figure 4-9. Power overhead of our approach for different circuit and trace sizes, benchmarks are ordered based on the number of flip-flops (shown in brackets)

trace buffer is dominant in these scenarios. However, the power overhead is much smaller for larger benchmarks where the circuit area is dominant compared to the trace buffer area, which is the case in real industrial designs. This can also be seen in Figure 4-9 which shows the power overhead (in percentage) of different circuit sizes (in terms of number of flip-flops) for different trace buffer widths. In summary, our promising fine-grained architecture proposes significant improvement in restoration quality with minor area and power overhead.

4.3 Summary

Signal selection is an important part of post-silicon debug. Existing techniques mainly focused on trace only signal selection. Recent techniques employed coarse-grained combination of trace and scan signals and showed limited effectiveness. In this chapter, we presented a debug architecture consisting of fine-grained combination of trace and scan signals. We developed efficient algorithms to select most profitable signals based on the proposed architecture. Our experimental results using ISCAS'89 benchmarks demonstrated that our approach shows up to 127% (36% on average) higher restoration compared to existing trace only approaches. Our approach produces up to 125% improvement (61.7% on average) compared with the state-of-the-art approaches that consider a combination of trace and scan signals. We have also demonstrated that our approach introduces minor (less than 1%) area and power overhead compared to existing trace only approaches.

CHAPTER 5
SCALABLE TRACE SIGNAL SELECTION USING MACHINE LEARNING

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 [3–5]. 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 [25] 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 chapter, we propose two novel selection techniques that uses machine learning to provide high restoration quality. Our proposed techniques are scalable - making them applicable for very large industry-scale circuits.

5.1 Learning-based Signal Selection

The key contribution of this section 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.

Fig. 5-1 shows the overview of our approach and its relation to existing simulation based approaches [7, 25]. Both elimination-based [7] and hybrid augmentation-based [25] 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 elimination-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.

5.1.1 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* $v = \langle f_1, f_2, \ldots, f_N \rangle$, 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*

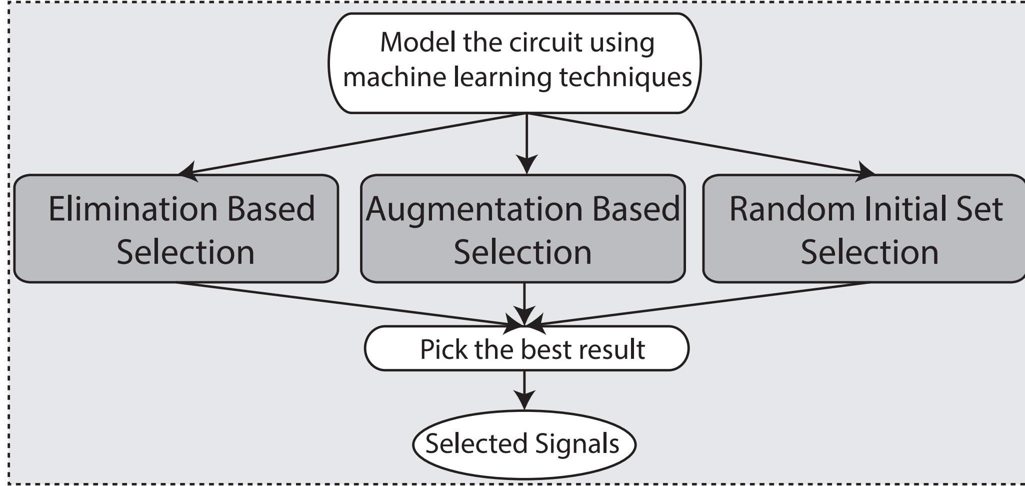


Figure 5-1. Overview of our approach and its relation to existing simulation based approaches

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.

$$\begin{aligned} \text{maximize} \quad & r_m(v) \\ \text{under constraint} \quad & \sum_{k=1}^{N} f_k = w \end{aligned} \tag{5–1}$$

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 chapter, we treat m as a small constant.

5.1.2 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 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 (generalized), 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 training data and low accuracy in

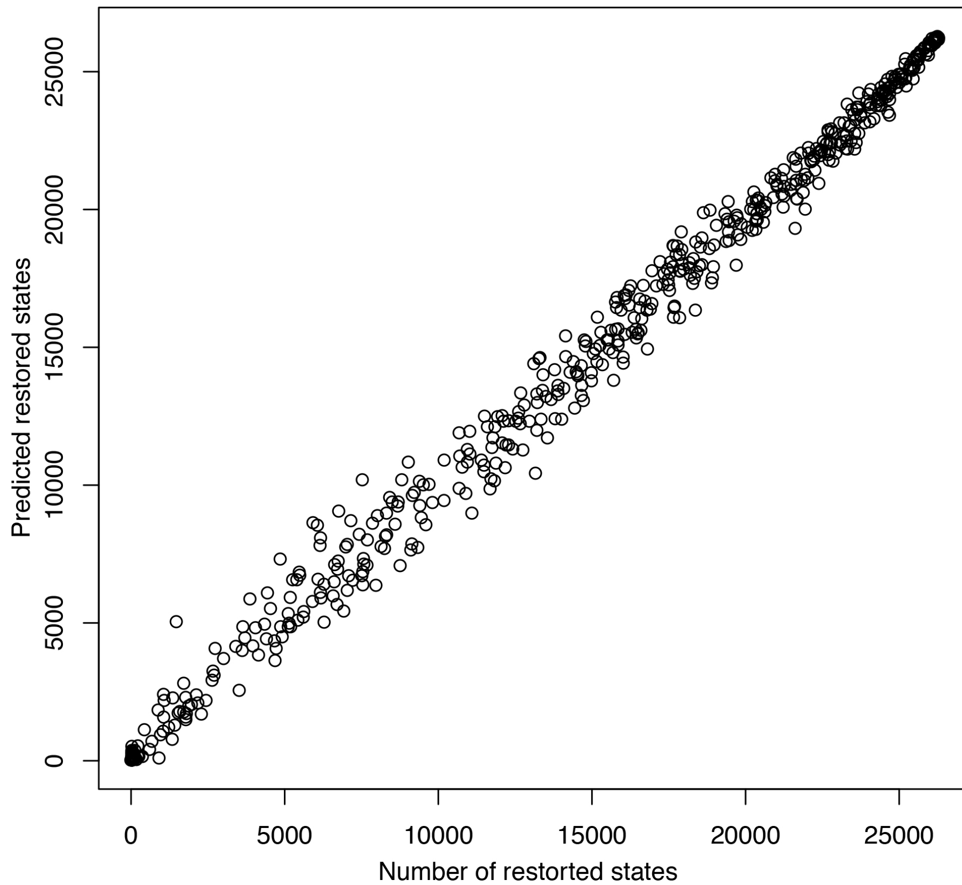


Figure 5-2. Real values of $r_m(v)$ versus predicted values for different random vectors in s38417 benchmark

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 particular 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. 5-2 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 *R* [72] 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

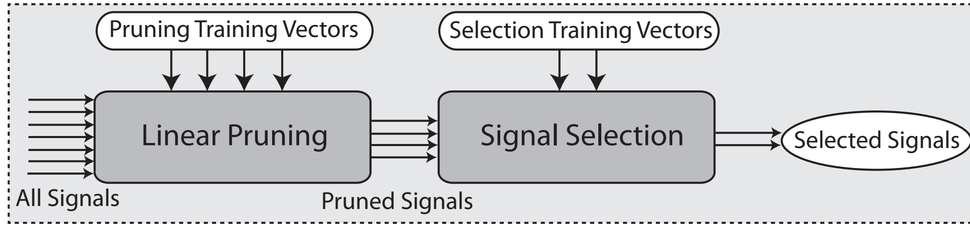


Figure 5-3. Proposed signal selection process

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 5.3 in details.

5.1.3 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. 5-3 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, 25]. Fig. 5-4 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

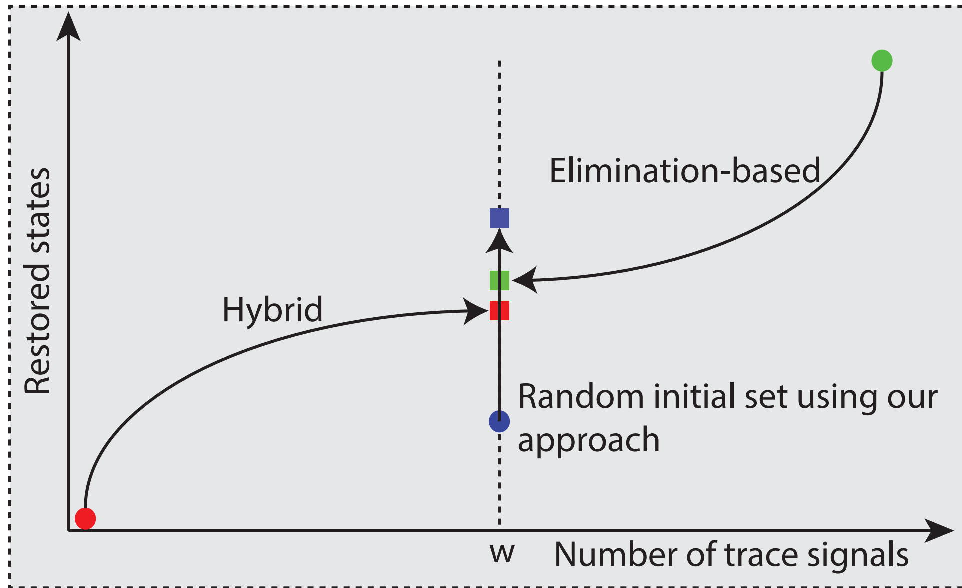


Figure 5-4. Different signal selection techniques for exploring the search space

number of remaining flip-flops is equal to the trace buffer width w; the hybrid augmentation approach [25] (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. 5-4. It can be observed that running all these techniques at the same time explores more of search space and increases the chance of finding a better local maximum which yields to a better set of selected signals.

Algorithm 5 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.

```
Data: circuit, m, t_pruning, p, r, t_selection, w, candidateModels
Result: The set of selected signals
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 = r̂_m(v) ;
Set v_new as the selected signals by calling AugmentationBased ;
if r̂_m(v_new) > r̂_m(v) then
    v = v_new ;
    maxRestorability = r̂_m(v) ;
end
for i = 1; i <= r; i++ do
    Set v_new as the selected signals by calling RandomInitialSet ;
    if r̂_m(v_new) > r̂_m(v) then
        v = v_new ;
        maxRestorability = r̂_m(v) ;
    end
end
return v ;
```

Algorithm 5: Learning-based Signal Selection

[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.

5.1.3.1 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 explanation 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:

$$\hat{r}_m(v) = \hat{w}_0 + \sum_{k=1}^{j} \alpha_k k(v_k, v) \tag{5–2}$$

In Equation 5–2, v is the vector whose restorability we wish to predict, v_k is the kth 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 5–2 as follows.

$$\hat{r}_m(v) = \hat{w}_0 + \sum_{k=1}^{j} \alpha_k v_k^T.v \tag{5–3}$$

$$\Rightarrow \hat{r}_m(v) = \hat{w}_0 + \hat{w}^T.v \; (where\ \hat{w} = \sum_{k=1}^{j} \alpha_k v_k) \tag{5–4}$$

Equation 5–4 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 6 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 5–4. 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.

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 5-5 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 $p = 0.15$.

Data: $circuit, m, t, p$
Result: The pruned set of signals
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 ;

Algorithm 6: Linear Pruning Algorithm

5.1.3.2 Generating training vectors

Algorithm 7 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

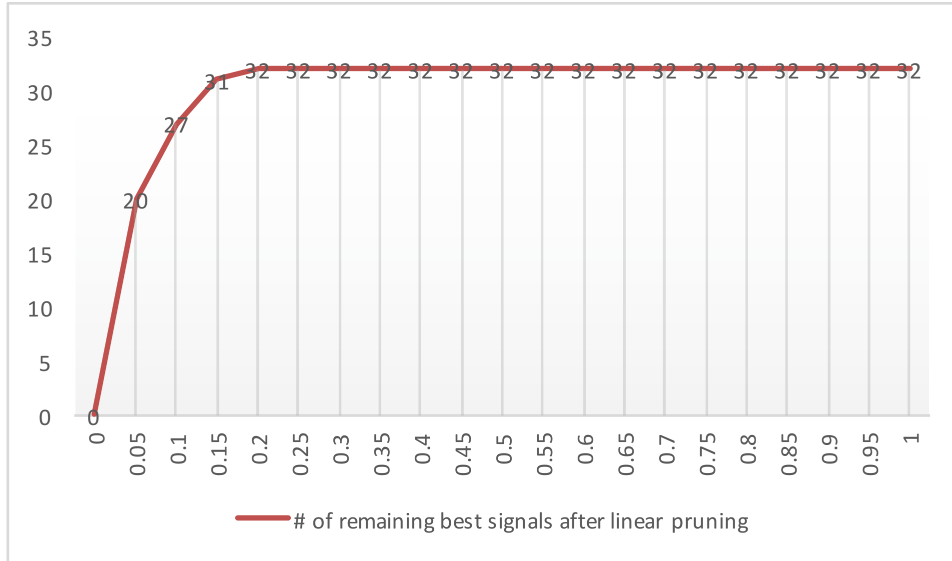


Figure 5-5. Number of profitable signals remaining after linear pruning

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 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.

5.1.3.3 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

```
Data: circuit, m, t
Result: The set of training vectors
Create training vectors set trainingSet ;
Create restoration power set R ;
totalGenerated = 0 ;
foreach 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 i = 2; i <= N; i++ do
    Add a vector to trainingSet in which exactly i random flip-flops are chosen ;
    totalGenerated++ ;
end
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 ∉ trainingSet then
        Add randomVector to trainingSet ;
        totalGenerated++ ;
    end
end
foreach vector trainingVector in trainingSet do
    R(trainingVector) = Restoration power of trainingVector using a mock simulation
    followed by a restoration process over m cycles ;
end
return trainingSet, R ;
```

Algorithm 7: Training Vector Generation

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.

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

Algorithm 8 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.

Data: *circuit*, m, $t_{selection}$, w, *candidateModels*
Result: The best training model
trainVectors = GenerateVectors(*circuit*, m, $t_{selection}$) ;
quickTrainVectors =10% of trainVectors randomly selected ;
quickTestVectors =10% of trainVectors randomly selected, exclusive with quickTrainVectors ;
foreach *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
bestModel = model with minimum MPE *result* = Model $\hat{r}_m(v)$ with pruned features using *bestModel* and *trainVectors* **return** result ;

Algorithm 8: Final Model Selection Algorithm

5.1.3.4 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 9 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 [7]. Because of computational limitation, they use a coarse-grained pruning 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.

```
Data: circuit, r̂_m(v), w
Result: The set of selected signals
Create selected signals set S ;
Create initial vector of v =< 1, 1, ..., 1 >, |v| = N × p ;
remainedSignals = N × p ;
while remainedSignals > w do
    maxRestorability = −∞ ;
    maxIndex = −1 ;
    for i = 1; i <= N × p; i + + do
        if v[i] = 1 then
            v[i] = 0 ;
            if r̂_m(v) > maxRestorability then
                maxRestorability = r̂_m(v) ;
                maxIndex = i ;
            end
            v[i] = 1 ;
        end
    end
    v[maxIndex] = 0 ;
    remainedSignals = remainedSignals − 1 ;
end
for i = 1; i <= N × p; i + + do
    if v[i] = 1 then
        Add i to S ;
    end
end
return S ;
```

Algorithm 9: Elimination-based Signal Selection

5.1.3.5 Augmentation-based signal selection

Algorithm 10 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.* [25]. 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.* [25], as they use simulation only for top 5% of the candidates, which can degrade the selection performance of their approach.

```
Data: circuit, r̂_m(v), w
Result: The set of selected signals
Create selected signals set S ;
Create initial vector of v =< 0, 0, ..., 0 >, |v| = N × p ;
for selected = 1; selected <= w; selected + + do
    maxRestorability = −∞ ;
    maxIndex = −1 ;
    for i = 1; i <= N × p; i + + do
        if v[i] = 0 then
            v[i] = 1 ;
            if r̂_m(v) > maxRestorability then
                maxRestorability = r̂_m(v) ;
                maxIndex = i ;
            end
            v[i] = 0 ;
        end
    end
    v[maxIndex] = 1 ;
end
for i = 1; i <= N × p; i + + do
    if v[i] = 1 then
        Add i to S ;
    end
end
return S ;
```

Algorithm 10: Augmentation-based Signal Selection

5.1.3.6 Signal selection using random initial set

Algorithm 11 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 profitable 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.

Data: $circuit, \hat{r}_m(v), w$
Result: The set of selected signals
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
end
return S ;

Algorithm 11: Signal selection using random initial set

5.2 Feature-based Signal Selection

In previous section, a learning-based signal selection approach [73] has been proposed where it applies a learning technique to the circuit under test to reduce the overhead of $O(N^2)$ simulations, where N is the number of signals. However, it still needs $O(N)$ simulations, typically thousands or millions of simulations of the design, to train the selection model, which limits its applicability on large industry scale circuits. The key contribution of this section is a novel feature-based signal selection technique that retains or improves the restoration quality of our approach outlined in Section 5.1 while achieving significantly faster selection time by drastically reducing training complexity. To the best of our knowledge, our approach is the first attempt in creating an automated signal selection technique that is applicable on billion-gate designs while providing the best possible restoration performance. Our approach is characterized by three key components: (1) simulation-based techniques are applied to a set of few small training circuits; and (2) a proper feature vector is created to apply machine learning techniques to learn the criteria for good trace signals; and (3) apply the model to the

Table 5-1. Time complexity of our approach and existing signal selection techniques

Technique	Mock Simulations	Other Computations
Simulation-based [3]	$O(N^2)$	None
Hybrid [3]	$O(N)$	Fast metric evaluations
Learning-based [73]	$O(N)$	Fast model training and predictions
Feature-based	None	Fast (one time) model training and predictions

larger circuit under test. The basic idea is to run only a small number of simulations to train the machine learning framework with a set of small circuits (one time process). Subsequently, our approach will utilize the prediction capability of the machine learning replacing the need for costly simulation runs in the larger circuits. Table 5-1 summarizes the time complexity of our approach compared to the existing state-of-the-art signal selection techniques for a circuit with N flip-flops.

Figure 5-6 shows the overview of our approach and its relation to existing simulation based approaches [7, 25]. First, we choose a set of small training circuits to build the selection model. For each training circuit, we apply a modified version of both elimination-based (Section 5.1.3.4) and augmentation-based (Section 5.1.3.5) approaches and select the best result. The difference is that in order to run the algorithms in sections 5.1.3.4 and 5.1.3.5, we needed simulation of actual design to generate the model, whereas here, we run simulations on a set of small circuits. We then generate a set of training vectors and add it to the training vectors set. Next, we use this training set to create a selection model using different machine learning regression techniques and pick the one with best accuracy. In this step, we train a model that learns the criteria of a good candidate signal and the relation with its properties. This model can then be used to select trace signals on any design under signal selection without expensive mock simulations involved. It should be noted that the selection model training is a one time process. Once it is done the model can be used to select trace signal on any other circuits.

5.2.1 Selection Model Generation

The core part of our approach is the training model generation and how to best choose the feature vectors. The model should be generic enough so that it can be applied to the

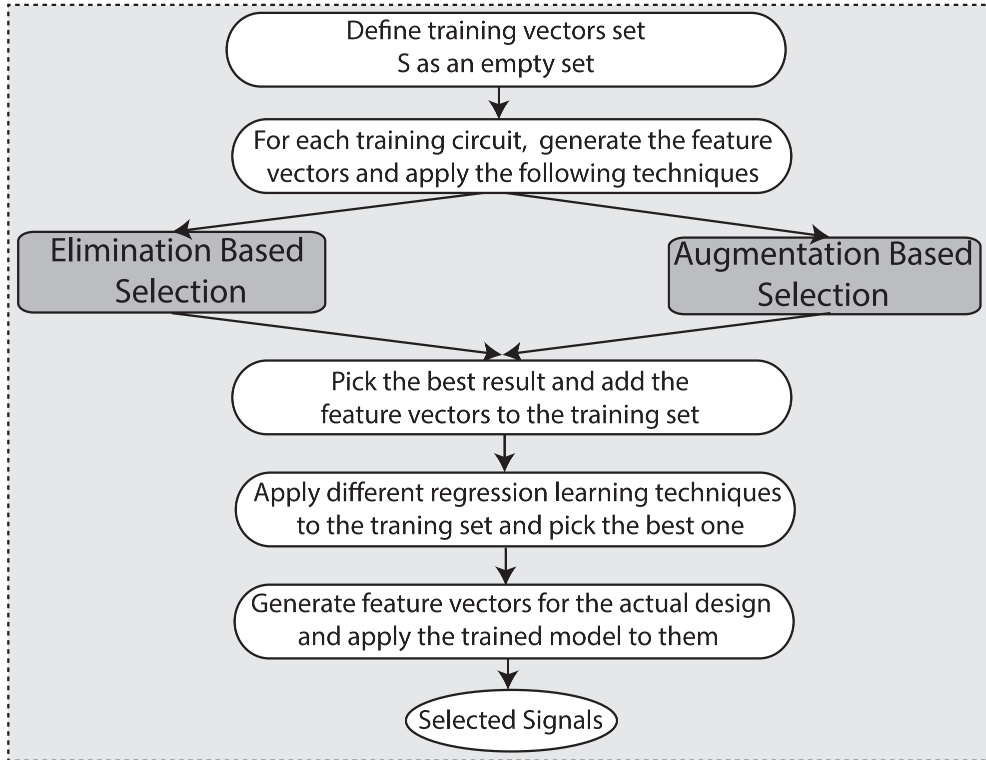


Figure 5-6. Overview of our proposed approach and its relation to existing simulation based approaches

circuit under test and accurate enough to produce hig quality result. Before going into details of our proposed modeling technique, we would like to define few terms and functions for a circuit with N flip-flops and mock simulation window *m*.

ForwardNeighbors$_\mathbf{g}$(x) for flip-flop f is defined as the number of gates of type g connected to its output.

BackwardNeighbors$_\mathbf{g}$(x) for flip-flop f is defined as the number of gates of type g connected to its inputs.

Connectivity(x) for flip-flop f is defined as the number of flip-flops connected to it through other combinational gates in both backward and forward directions.

InputDistance(x) for flip-flop f is defined as the minimum distance of the flip-flop from the input signals (in terms of number of gates).

OutputDistance(x) for flip-flop f is defined as the minimum distance of the flip-flop from the output signals (in terms of number of gates).

ZeroProbability(x) for flip-flop f is defined as the percentage of 0 values for the flip-flop in a mock simulation over m cycles.

SingleRestoration(x) for flip-flop f is defined as the number of restored states in a mock simulation/restoration process over m cycle if f is the only trace signal.

$\mathbf{SelectionOrder_g(x)}$ for flip-flop f is defined as the sequence number of selection when technique g is applied. This number is 1 for the first (best) selected signal and N for the last selected signal.

Rank(g(x)) for flip-flop f is defined as the number of flip-flops with $g(x) \leq g(f)$ divided by N. In other words, it is the normalized relative rank of applying function g to flip-flop f compared to the other flip-flops. This value would be 1 and $1/N$ for the flip-flops with the maximum and minimum value of g(f), respectively.

5.2.1.1 Feature selection

Selecting the right features is the most important part of any machine learning problem as it directly impacts the quality of the model and solution. In our case, the features should be selected such that it can model the true correlation between structural properties of a signal and its performance in state restoration. In addition, it should be independent of the circuit size and structure. This is crucial as we want to train our model using a set of small circuits and apply the learning to the bigger circuit under test. Lastly, the generation of feature vectors should not be computationally expensive so it can easily scale while selecting signals in large industry-scale circuits. In order to address all these requirements, we define feature vector v for flip-flop f in circuit c to have the following components.

$\mathbf{Rank(ForwardNeighbors_g(f))}$ for all gates of type g.

$\mathbf{Rank(BackwardNeighbors_g(f)}$ for all gates of type g.

Rank(Connectivity(f)) for flip-flop f.

Rank(InputDistance(f) for flip-flop f.

Rank(OutputDistance(f)) for flip-flop f.

Rank(ZeroProbability(f)) for flip-flop f.

Rank(SingleRestoration(f)) for flip-flop f.

It can be seen that we have chosen features that are mostly based on the circuit structure and fast to evaluate. In addition, we are applying rank function to all the features to make them relative values instead of using the absolute values. This makes the features independent of the circuit size and number of gates. Intuitively, our feature vector captures the fan-in and fan-out of the signal, its relative position and depth in the circuit, and its impact on restoring its neighbors when selected as a trace signal. Our experiments have shown that there is a high correlation between these features and the restoration performance of a flip-flip.

5.2.1.2 Model selection

In this step, we generate a selection model by applying simulation based techniques on a small set of training circuits. Intuitively, the model learns the criteria of a good trace signal and the relation with its feature vector described before. Algorithm 12 outlines the steps involved in creating our selection model from a set of small training circuits. For each circuit, we apply both augmentation and elimination based techniques and pick the one with better result (the one with better average restoration ratio for trace buffer widths 8, 16, and 32). Next, for each flip-flip f in the circuit we add a pair of feature vector v and selection order rank r to the training vectors set. Choosing selection order rank helps to normalize the training data across all the circuits and makes it independent of number of flip-flops in the circuit. We then apply different techniques in regression models set to the training vectors and return the best one as the result. To measure the quality of a model on a test vector set of size n, we use Mean Prediction Error defined in Section 5.1.3.3. In order to avoid overfitting in our model training, we use 5-fold cross-validation technique where we use 20% of the vectors as test (validation) vectors and 80% as the the training vectors.

5.2.2 Signal Selection

Once we have our selection model trained, it can be used on any circuit for selecting trace signals. To motivate our approach, we have used the smallest circuits in ISCAS'89 benchmarks

Data: trainingCircuits, regressionModels, m
Result: Trained selection model
Create training vectors set trainingVectors ;
for *each circuit c in trainingCircuits* **do**
  n = number of flip-flops in c ;
  apply AugmentationBased(c,n,m) and EliminationBased(c,1,m) to c and pick the best one as g ;
  for *each flip-flop f in c* **do**
    Add $< v, r >$ to trainingVectors where v is the feature vector for the flip-flop and r is Rank(SelectionOrder$_g$(f)) ;
  end
end
apply all the models in regressionModels to trainingVectors ;
return model m with minimum MPE ;

Algorithm 12: Model Generation Algorithm

[2] to train our model using *cubist* regression technique. Figure 5-7 shows the actual versus predicted selection ranks (using the trained model) on a set of flip-flops in s38584 benchmarks (in this example, s38584 is assumed as the actual design under signal selection). It can be seen that there is a high correlation between the real and predicted values. This is significant as it enables us to select high quality trace signals in the circuit under test using very fast predication to generate the selection ranks based on the feature vectors, instead of expensive mock simulations.

Algorithm 13 outlines our proposed selection technique. We first generate feature vectors for all the flip-flops in the circuit. We then use these vectors to predict the selection sequence rank for the flip-flops using the given selection model *m*. We return the top *w* (trace buffer width) flip-flops with the highest value of predicted selection sequence rank as the result.

5.3 Experiments

In order to investigate the effectiveness of our proposed approaches, 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

[2] s1494, s1488, s713, s1238, s1196, and s838.

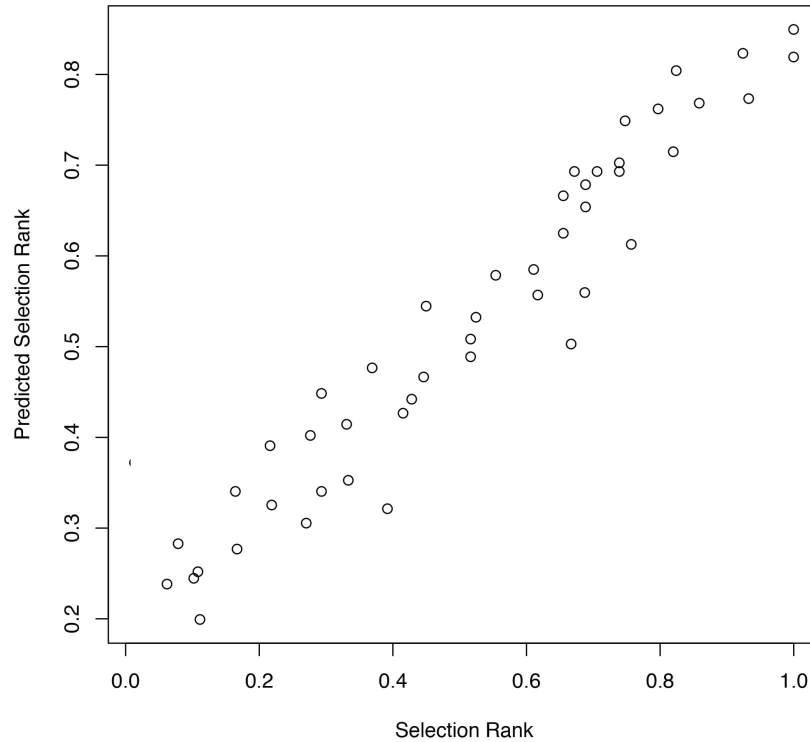


Figure 5-7. Actual versus predicted values of selection ranks in s38584 benchmark

Data: circuit, m, w
Result: Set of selected signals
Initialize predictionMap as an empty map ;
for *each flip-flop f in circuit* **do**
  v = feature vector of f ;
  r = m(v), the predicted value of selection sequence rank of f using model m ;
  Add $< f, r >$ to predictionMap ;
end
Sort predictionMap based on r values ;
return top w flip-flops with the highest values of r ;
Algorithm 13: Signal Selection Algorithm

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* [70].

In our experiments, we did not use the reported numbers of Li *et al.* [25] and Chatterjee *et al.* [7], since they used modified versions of ISCAS'89 benchmarks (with some specific

optimizations). To perform a fair comparison, we tried to obtain the executables of [25] and [7]. Li et al. [25] 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 $m = 32$ as simulation window. For reporting the restoration ratio, 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. In this section, we present the experimental result of both learning-based (5.1) and feature-based (5.2) signal selection techniques.

5.3.1 Learning-based Signal Selection

5.3.1.1 Experimental setup

In order to investigate the effectiveness of our proposed approach, we used the set of largest circuits in ISCAS'89 as has been studied by previous works. We used *caret* package in *R* [72] as the modeling/prediction tool. In addition, we used *10-fold cross validation* and *normalization and scaling* while training our models.

5.3.1.2 Model selection

In order to choose the best non-linear after pruning model for the benchmarks, we explored several models available in *caret* package [72]. Fig. 5-9 illustrates Mean Prediction Errors of different models on the set of our benchmarks caluculated using Equation 5–5. 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. 5-8 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

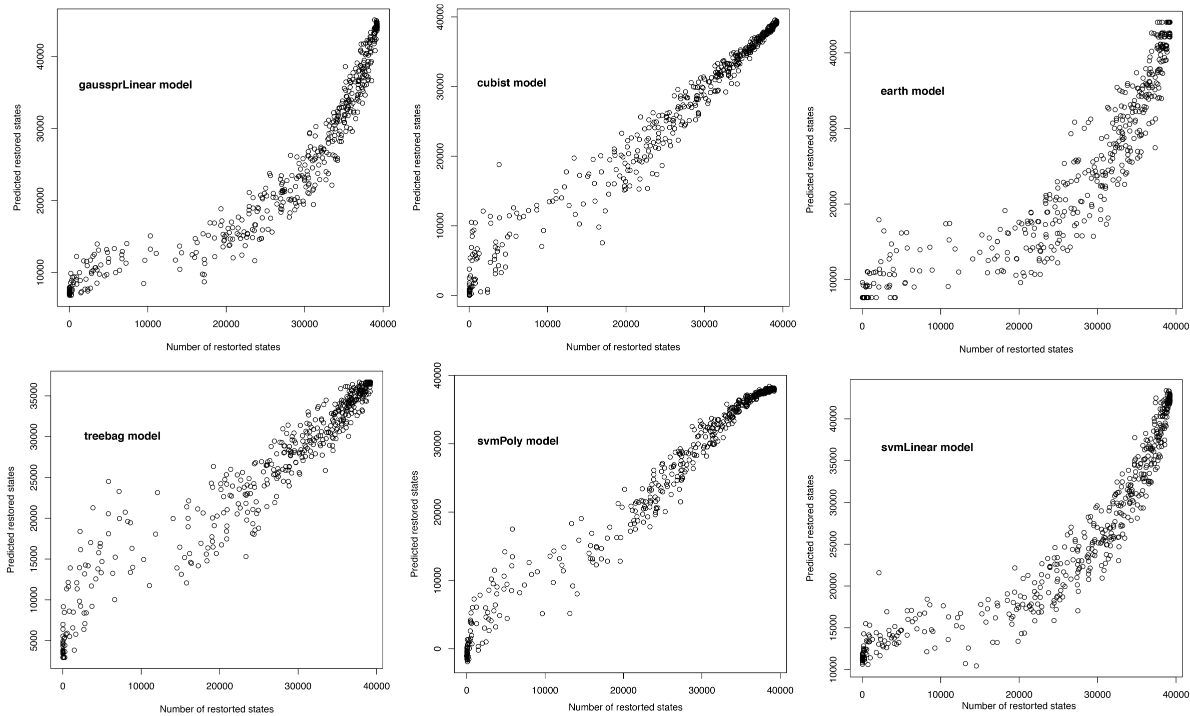
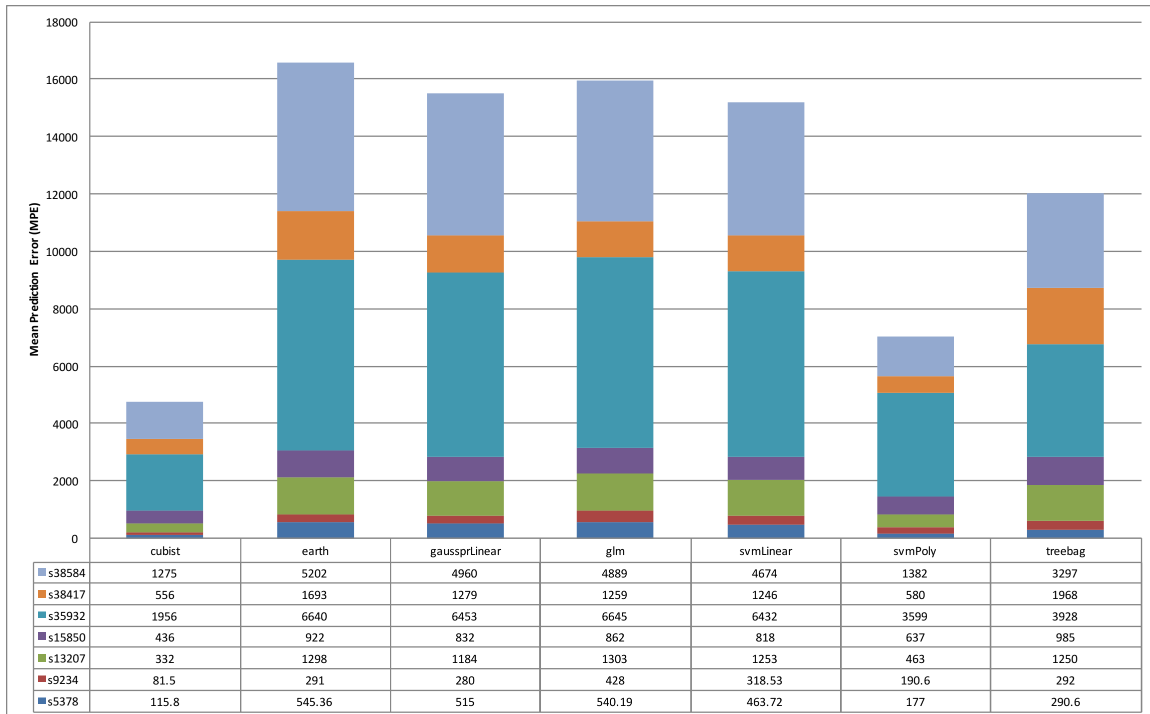


Figure 5-8. Real versus predicted restoration states for different models in *S38584* benchmark

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. *Cubist* is a non-linear model, simpler that neural network, and designed to anaylyze millions of records which makes it a good fit for our large scale applciation [74]. 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.

5.3.1.3 Restoration quality

Table 5-2 presents the restoration ratios of our approach compared with previous techniques [7, 25] 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 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



	cubist	earth	gaussprLinear	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

Figure 5-9. Mean prediction errors (MPE) of different models on the set of our benchmarks

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.* [25] 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

Table 5-2. Restoration ratios using our approach compared with existing selection approaches

Circuit	#Flip-flops	Buffer Width	Simulation-based [7]	Hybrid [25]	Learning-based	Imp. over the best
		8	13.41	**14.35**	14.20 (E)	-1.0%
s5378	179	16	7.35	**8.36**	8.40 (E)	0.5%
		32	4.47	**4.99**	4.93 (R)	-1.2%
		8	**13.98**	9.25	15.33 (E)	9.7%
s9234	228	16	**8.30**	6.13	8.76 (R)	5.5%
		32	**4.46**	4.38	4.84 (R)	8.5%
		8	**26.33**	21.90	44.03 (E)	67.2%
s15850	597	16	**19.89**	14.78	23.13 (E)	16.3%
		32	**13.19**	10.88	13.92 (A)	5.5%
		8	**35.52**	33.60	47.18 (E)	32.8%
s13207	669	16	20.13	**23.22**	29.00 (A)	24.9%
		32	11.25	**13.64**	15.42 (R)	13.0%
		8	19.73	**27.00**	54.25 (A)	100.1%
s38584	1452	16	**28.39**	13.97	69.03 (R)	143.1%
		32	**32.45**	7.50	43.66 (R)	34.5%
		8	29.23	**37.71**	52.33 (E)	38.8%
s38417	1636	16	17.02	**23.80**	27.12 (R)	13.94%
		32	**15.14**	11.83	16.73 (R)	10.5%
		8	132.00	**144.00**	186.8 (E)	29.7%
s35932	1728	16	67.45	**72.00**	93.60 (E)	30.0%
		32	34.63	**36.00**	46.98 (A)	30.5%

our newly introduced random initial set selection technique yielded the best result in several benchmarks. The improvement in restoration performance 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 [25].

5.3.1.4 Selection time, 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 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 experiments, 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 [25], 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

Table 5-3. Runtime comparison of our approach compared with existing selection approaches

Circuit	#Flip-flops	Buffer Width	Simulation-based [7]	Hybrid [25]	Learning-based
		8	00:01:53	00:00:08	00:01:46
s5378	179	16	00:01:52	00:00:10	00:01:52
		32	00:01:48	00:00:16	00:02:09
		8	00:08:52	00:00:32	00:00:10
s9234	228	16	00:08:43	00:00:40	00:00:10
		32	00:08:10	00:00:50	00:00:10
		8	03:44:12	00:05:20	00:04:20
s15850	597	16	03:44:04	00:06:00	00:04:35
		32	03:43:39	00:06:36	00:05:04
		8	01:21:41	00:01:36	00:03:45
s13207	669	16	01:21:35	00:02:00	00:04:01
		32	01:21:13	00:02:40	00:04:12
		8	28:43:02	00:05:28	00:16:52
s38584	1452	16	28:42:16	00:06:06	00:17:09
		32	28:38:59	00:09:02	00:17:35
		8	196:51:50	00:22:42	00:20:23
s38417	1636	16	196:50:44	00:33:04	00:21:07
		32	196:48:27	00:34:28	00:23:55
		8	11:39:36	00:04:28	00:16:49
s35932	1728	16	11:39:09	00:05:56	00:17:33
		32	11:38:01	00:08:38	00:18:21

complexity of our approach is $\theta(N)$ and $\theta(wN)$ for [25], with potentially different constant coefficients.

To compare the runtime 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 5-3 presents the runtime of our approach compared with previous techniques [7, 25] using different ISCAS'89 benchmarks. 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 [25], 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 [25] 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).

5.3.2 Feature-based Signal Selection

5.3.2.1 Experimental setup

In order to investigate the effectiveness of our proposed approach, we used the set of largest circuits in ISCAS'89 as has been studied by previous works. We also used the largest circuits in ITC'99 benchmarks. We used a set of regression techniques (cubist, earth, gaussprLinear, glm, svmLinear, svmPoly, and treebag) in *caret* package in *R* [72] as the modeling/prediction tool. In addition, we used *5-fold cross validation* while training our models. We used a set of smallest ISCAS'89 benchmarks (s1494, s1488, s713, s1238, s1196, and s838) to train our selection model.

5.3.2.2 Model Selection

In order to choose the best regression model for our signal selection application, we explored several models available in *caret* package [72] in *R*. Figure 5-10 illustrates real versus estimated values of selection rank on S38584 benchmark using training circuits set of s1494, s1488, s713, s1238, s1196, and s838. It can be observed that *cubist* is the best model in our experiments with minimum prediction error and highest correlation between the real and estimated value. In fact, *cubist* outperformed other models consistently for other benchmarks as well. 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 regression model for the rest of our experiments.

5.3.2.3 Restoration quality

Table 5-4 presents the restoration ratios of our approach compared with previous state-of-the-art techniques [7, 25, 73] using different ISCAS'89 and ITC'99 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 ones shown as 'N/A' for

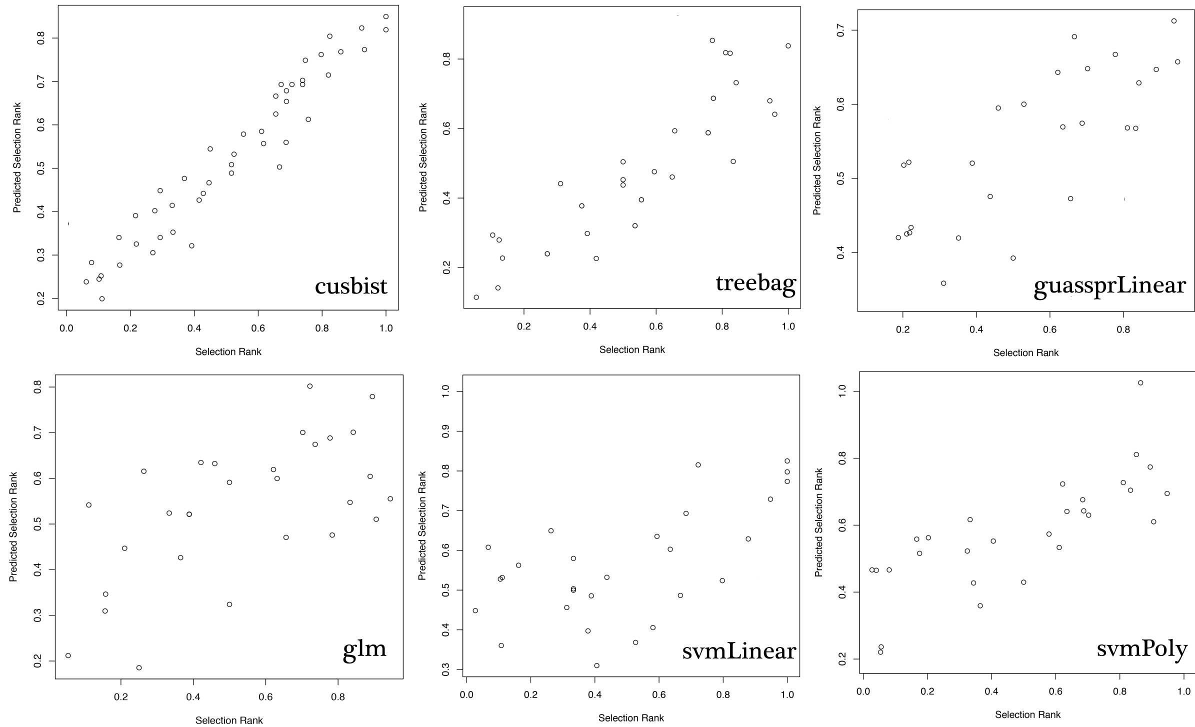


Figure 5-10. Real versus estimated values of selection rank on S38584 benchmark

[7] and [73] means that their technique was not able to finish within 24 hour of runtime. In addition, unfortunately we were not able to get the result of [25] for ITC'99 benchmarks as their binary failed to parse the benchmarks. The last column indicates the percentage of improvement using our approach compared with the best (shown in bold) result provided by the existing approaches. The results indicate that our approach consistently performs comparable or better compared to existing approaches. The improvement in restoration performance is up to 135.4% in *s38584* and 8.8% on average. Compared to Chatterjee *et al.* [7], we run the elimination-based technique on training circuits without any pruning which reduces the chance of removing effective flip-flops prior to selection itself. Similarly, Li *et al.* [25] incorporated simulations for only top 5% of the candidate flip-flops, which sacrifices the precision of the selection process. In addition, building the selection model using small training circuits, allows us to run both elimination-based and augmentation-based techniques at the same time and pick the best one for each circuit. Compared to Rahmani *et al.* [73], we train the selection model based on the training circuits structure and the best result of

Table 5-4. Restoration ratios using our approach compared with existing selection approaches

Circuit	#Flip-flops	Buffer Width	Simulation-based [7]	Hybrid [25]	Learning-based [73]	Our Approach	Imp. over the best
		8	13.41	**14.35**	14.20	14.13	-1.5%
s5378	179	16	7.35	8.36	**8.40**	8.92	6.1%
		32	4.47	**4.99**	4.93	5.12	2.6%
		8.0	13.98	9.25	**15.33**	15.82	3.2%
s9234	228	16	8.30	6.13	**8.76**	9.10	3.9%
		32	4.46	4.38	**4.84**	5.11	5.6%
		8	26.33	21.90	**44.03**	45.12	2.5%
s15850	597	16	19.89	14.78	**23.13**	24.37	5.4%
		32	13.19	10.88	**13.92**	13.82	-0.7%
		8	35.52	33.60	**47.18**	49.30	4.5%
s13207	669	16	20.13	23.22	**29.00**	31.21	7.6%
		32	11.25	13.64	**15.42**	16.13	4.6%
		8	N/A	27.00	**54.25**	127.72	135.4%
s38584	1452	16	N/A	13.97	**69.03**	79.09	14.6%
		32	N/A	7.50	**43.66**	44.02	0.8%
		8	N/A	37.71	**52.33**	53.27	1.8%
s38417	1636	16	N/A	23.80	**27.12**	26.97	-0.5%
		32	N/A	11.83	**16.73**	17.10	2.2%
		8	132.00	144.00	**186.80**	186.90	0.1%
s35932	1728	16	67.45	72.00	**93.60**	93.42	-0.1%
		32	34.63	36.00	**46.98**	47.15	0.4%
		8	5.99	N/A	**6.15**	7.18	16.7%
b15	449	16	3.56	N/A	**4.83**	4.98	3.1%
		32	34.63	N/A	**3.31**	3.46	4.53%
		8	N/A	N/A	**14.12**	14.43	2.1%
b17	1415	16	N/A	N/A	**13.19**	13.31	0.9%
		32	N/A	N/A	**7.93**	8.77	10.6%
		8	N/A	N/A	N/A	25.12	N/A
b18	3320	16	N/A	N/A	N/A	21.60	N/A
		32	N/A	N/A	N/A	12.49	N/A
		8	N/A	N/A	N/A	32.00	N/A
b19	6642	16	N/A	N/A	N/A	24.64	N/A
		32	N/A	N/A	N/A	18.11	N/A

simulation-based techniques, whereas they use machine-learning to just reduce the number of mock simulation on the circuit under test. In addition, our model is trained using the best result of simulation-based techniques on a set of training circuits (instead of just one), which provides a more globally optimized selection model. Finally, although our model is trained using small circuits in ISCAS'89 benchmarks, it still outperforms [73] in ITC'99 benchmarks (b15 and b17). This shows that the proposed feature vector and selection model is generic enough that can be applied to the designs that are not even related to the training circuits.

5.3.2.4 Selection time and scalability

To compare the runtime of different approaches, we used an Octa-Core AMD Opteron 6378 (1400 MHz) machine with 188GB of memory for all the experiments. The runtime for our approach is calculated as the summation of required time for generating training vectors

on the training circuits, modeling, generating the feature vectors for the circuit under test, and the signal selection process itself. Table 5-5 presents the runtime of our approach compared with the previous techniques [7, 25, 73] using different ISCAS'89 and ITC'99 benchmarks. The reported runtime format is 'hour:minute:second'. The ones shown as 'N/A' for [7] and [73] means that their technique was not able to finish within 24 hour of runtime. In addition, unfortunately we were not able to get the result of [25] for ITC'99 benchmarks as their binary failed to parse the benchmarks. As expected, our approach is significantly faster than the existing approaches. The speed-up is up to 37X in *s38417* and *b17* for buffer width of 32 and 17.6X on average. This is because in our approach we need mock simulations only on a set of small training circuits. Once the model is created, there is no need for any simulations on the circuit under test as the selection process just uses fast predictions instead of actual simulations. Moreover, model creation is a one time pre-process in our approach. Once it is created using the training circuits, it can be used to select trace signals in any industry-scale large circuits. This makes our approach significantly more scalable and practical compared to the existing ones. In summary, our approach not only produces comparable or better restoration quality, but it is also significantly faster than the existing approaches.

5.4 Summary

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 signal selection techniques using machine learning which provides comparable or better restorability while providing drastic reduction in selection selection time, making it applicable for industry-scale circuits.

Table 5-5. Runtime comparison of our approach compared with existing selection approaches

Circuit	Buffer Width	Simulation-based [7]	Hybrid [25]	Learning-based [73]	Our Approach	Speedup
	8	00:01:53	00:00:08	00:01:46	00:00:11	0.7X
s5378	16	00:01:52	00:00:10	00:01:52	00:00:11	0.9X
	32	00:01:48	00:00:16	00:02:09	00:00:11	1.5X
	8	00:08:52	00:00:32	00:00:10	00:00:11	1.0X
s9234	16	00:08:43	00:00:40	00:00:10	00:00:11	1.0X
	32	00:08:10	00:00:50	00:00:10	00:00:11	1.0X
	8	03:44:12	00:05:20	00:04:20	00:00:1	20.1X
s15850	16	03:44:04	00:06:00	00:04:35	00:00:13	21.2X
	32	03:43:39	00:06:36	00:05:04	00:00:13	23.4X
	8	01:21:41	00:01:36	00:03:45	00:00:13	7.4X
s13207	16	01:21:35	00:02:00	00:04:01	00:00:13	9.2X
	32	01:21:13	00:02:40	00:04:12	00:00:13	12.3X
	8	N/A	00:05:28	00:16:52	00:00:36	9.1X
s38584	16	N/A	00:06:06	00:17:09	00:00:36	10.2X
	32	N/A	00:09:02	00:17:35	00:00:36	15.1X
	8	N/A	00:22:42	00:20:23	00:00:39	31.4X
s38417	16	N/A	00:33:04	00:21:07	00:00:39	32.5X
	32	N/A	00:34:28	00:23:55	00:00:39	36.8X
	8	11:39:36	00:04:28	00:16:49	00:00:37	7.2X
s35932	16	11:39:09	00:05:56	00:17:33	00:00:37	9.6X
	32	11:38:01	00:08:38	00:18:21	00:00:37	14.0X
	8	06:12:09	N/A	00:06:49	00:00:12	34.1X
b15	16	06:09:55	N/A	00:07:03	00:00:12	35.3X
	32	06:06:40	N/A	00:07:11	00:00:12	35.9X
	8	N/A	N/A	00:19:10	00:00:35	32.9X
b17	16	N/A	N/A	00:20:30	00:00:35	35.1X
	32	N/A	N/A	00:21:40	00:00:35	37.1X
	8	N/A	N/A	N/A	00:06:11	N/A
b18	16	N/A	N/A	N/A	00:06:11	N/A
	32	N/A	N/A	N/A	00:06:11	N/A
	8	N/A	N/A	N/A	00:21:09	N/A
b19	16	N/A	N/A	N/A	00:21:09	N/A
	32	N/A	N/A	N/A	00:21:09	N/A

CHAPTER 6
CONCLUSIONS AND FUTURE WORK

Post-silicon validation and debug is a challenging step in chip design methodology. A fundamental challenge in post-silicon validation is limited observability. This dissertation proposed efficient and scalable techniques to enhance the observability during post-silicon debug to reduce overall validation effort. This chapter concludes the dissertation and outlines future research directions.

6.1 Conclusions

Due to limitations in the number of output pins as well as area and power overhead of internal trace buffer, only a small percentage of internal signals in the design can be observed during silicon execution. Furthermore, in order for a signal to be observed, the design must be instrumented a priori with appropriate control hardware that routes a signal to an observation point. It is therefore crucial to identify trace signals to maximize design visibility and debug information under the observability constraints. To improve observability, this dissertation made several important contributions as summarized below.

In chapter 3, we presented a simulation-based signal selection technique that yields signals with higher restorability than current approaches while still being computationally efficient. Our key contribution is the observation that simulation-based signal selection can be significantly improved by augmentation through ILP-based refinement, together with the insights to smoothly integrate the augmentation phase into the selection framework resulting in a unified scalable infrastructure.

To improve the observability in post-silicon debug further, various approaches explored a profitable combination of trace and scan signals where they divided signals into two coarse-grained categories, trace and scan signals. In chapter 4, we presented a novel debug architecture consisting of fine-grained combination of trace and scan signals. We also developed efficient algorithms to select most profitable signals based on the proposed

architecture. Our experimental results show significantly higher restoration quality compared to existing approaches.

Existing trace signal selection are either metric-based or simulation-based. 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. To address this, we presented two novel signal selection techniques based on machine learning in Chapter 5 which retains or improves the restoration quality of simulation-based signal selection while achieving significantly faster selection time, making it practical for billion-gate industry-scale circuits.

6.2 Future Research Directions

Post-silicon validation and debug has been and will continue to be a critical step in silicon design cycle. The research presented in this dissertation can be extended in the following possible directions:

Existing trace buffers mostly store the value of the signals in every cycle. Recent works [34, 35] are based on the idea that it is not necessary to store all the cycles, specially for those signals that can be restored using simulation or those that are most likely from a non-buggy cycle. In fact, during the debug session we may not know which cycles are the erroneous ones to store and we have to rely on a multiple-step off-line software analysis to find those based on the input. This thesis can be further extended by exploring an anomaly detection technique. Figure 6-1 illustrates the steps involved in this approach. First, we need to build a time series anomaly detection model where the feature vector consists of the cycle number and the values of trace signals. This model then can be mapped to a hardware similar to [75]. This hardware then can be integrated into the chip and works as our trigger logic. In each cycle, we store or skip the trace signal values based on the anomaly detector output. This means, we only store the signals if there is a high chance that they are buggy. Depending on the anomaly detector accuracy, this helps us to compress the data and increase the effective visibility window in trace

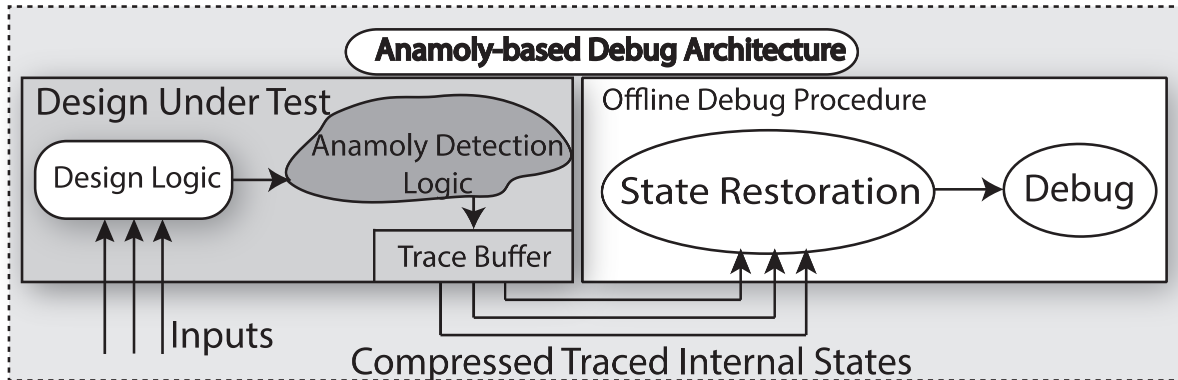


Figure 6-1. Overview of our proposed anomaly-based debug architecture

buffer. In addition, this can help in early detection of erroneous state where an error occurs, which can take several hundred cycles to manifest in output signals.

There are existing works on compression-enabled trace buffers which try to increase the effective size of trace buffer in the circuit. In order to achieve the maximum observability, our signal selection techniques can be further extended by modifying them such that the compression is incorporated as a selection metric. In addition, our proposed techniques can be further extended to incorporate multiple clock domain systems such as SoCs where each subsystem might operate on a different frequency. Furthermore, a generic selection technique needs to be developed that can take into account the fact that each subsystem in SoCs can have their own local trace buffer that is used for global system-level state restoration.

To summarize, in this dissertation I have developed several signal selection techniques to improve the observability of the signals during post-silicon validation. Our proposed selection techniques maintain a high restoration quality while scalable for large industry scale circuits. Our proposed techniques can be further extended by applying them to other domains such as trace buffer compression, multi clock systems, and distributed trace buffers in SoCs.

APPENDIX: LIST OF PUBLICATIONS

Journal Articles

1. **Kamran Rahmani**, Sandip Ray, and Prabhat Mishra *Post-silicon Trace Signal Selection Using Machine Learning Techniques*, IEEE Transactions on VLSI (TVLSI), 25(2), 570-580, 2017.

2. **Kamran Rahmani**, Sudhi Proch, and Prabhat Mishra, *Efficient Selection of Trace and Scan Signals for Post-Silicon Debug*, IEEE Transactions on VLSI (TVLSI), 24(1), 313-323, 2016.

3. Hadi Hajimiri, **Kamran Rahmani**, and Prabhat Mishra, *Compression-Aware Dynamic Cache Reconfiguration for Embedded Systems*, Elsevier Sustainable Computing: Informatics and Systems (SUSCOM), 2(2), 71-80, 2012.

Conference Papers

1. **Kamran Rahmani**, and Prabhat Mishra, *Feature-based Signal Selection for Post-silicon Debug Using Machine Learning*, Under Review in International Test Conference (ITC), 2017.

2. Hadi Hajimiri, **Kamran Rahmani**, and Prabhat Mishra, *Efficient Peak Power Estimation using Probabilistic Cost-Benefit Analysis*, 28th International Conference on VLSI Design, 369-374, 2015.

3. **Kamran Rahmani**, Prabhat Mishra, and Sandip Ray, *Efficient Trace Signal Selection using Augmentation and ILP Techniques*, 15th ISQED, 148-155, 2014.

4. **Kamran Rahmani**, Prabhat Mishra, and Sandip Ray, *Scalable Trace Signal Selection using Machine Learning*, IEEE 31st International Conference on Computer Design (ICCD), 384-389, 2013.

5. **Kamran Rahmani**, and Prabhat Mishra, *Efficient Signal Selection using Fine-grained Combination of Scan and Trace Buffers*, 26th International Conference on VLSI Design, 308-313, 2013.

6. **Kamran Rahmani**, Prabhat Mishra, and Swarup Bhunia, *Memory-based Computing for Performance and Energy Improvement in Multicore Architectures*, ACM Great Lakes Symposium on VLSI (GLSVLSI), 287-290, 2012.

7. **Kamran Rahmani**, Hadi Hajimiri, Kartik Shrivastava, and Prabhat Mishra, *Synergistic Integration of Code Encryption and Compression in Embedded Systems*, ACM Great Lakes Symposium on VLSI (GLSVLSI), 363-368, 2012.

8. Hadi Hajimiri, **Kamran Rahmani**, and Prabhat Mishra, *Synergistic Integration of Dynamic Cache Reconfiguration and Code Compression in Embedded Systems*, IEEE International Green Computing Conference (IGCC), 1-8, 2011.

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] K. Basu and P. Mishra, "Rats: Restoration-aware trace signal selection for post-silicon validation," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 21, no. 4, pp. 605–613, 2013.

[4] 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.

[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] 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.

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

[9] 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.

[10] 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.

[11] 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.

[12] O. Caty, P. Dahlgren, and I. Bayraktaroglu, "Microprocessor silicon debug based on failure propagation tracing," in *IEEE International Conference on Test, 2005.*, Nov 2005, pp. 10 pp.–293.

[13] F. Koushanfar, D. Kirovski, and M. Potkonjak, "Symbolic debugging scheme for optimized hardware and software," in *IEEE/ACM International Conference on Computer Aided*

Design. ICCAD - 2000. IEEE/ACM Digest of Technical Papers (Cat. No.00CH37140), Nov 2000, pp. 40–43.

[14] F. M. D. Paula, M. Gort, A. J. Hu, S. J. E. Wilton, and J. Yang, "Backspace: Formal analysis for post-silicon debug," in *2008 Formal Methods in Computer-Aided Design*, Nov 2008, pp. 1–10.

[15] N. Nataraj, T. Lundquist, and K. Shah, "Fault localization using time resolved photon emission and stil waveforms," in *International Test Conference, 2003. Proceedings. ITC 2003.*, vol. 1, Sept 2003, pp. 254–263.

[16] A. DeOrio, I. Wagner, and V. Bertacco, "Dacota: Post-silicon validation of the memory subsystem in multi-core designs," in *2009 IEEE 15th International Symposium on High Performance Computer Architecture*, Feb 2009, pp. 405–416.

[17] G. J. V. Rootselaar and B. Vermeulen, "Silicon debug: scan chains alone are not enough," in *International Test Conference 1999. Proceedings (IEEE Cat. No.99CH37034)*, 1999, pp. 892–902.

[18] D. Josephson and B. Gottlieb, "The crazy mixed up world of silicon debug [ic validation]," in *Proceedings of the IEEE 2004 Custom Integrated Circuits Conference (IEEE Cat. No.04CH37571)*, Oct 2004, pp. 665–670.

[19] M. Abramovici, P. Bradley, K. Dwarakanath, P. Levin, G. Memmi, and D. Miller, "A reconfigurable design-for-debug infrastructure for socs," in *Design Automation Conference, 2006 43rd ACM/IEEE*, 0-0 2006, pp. 7 –12.

[20] 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.

[21] 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.

[22] K. Rahmani, P. Mishra, and S. Ray, "Efficient trace signal selection using augmentation and ilp techniques," in *Quality Electronic Design (ISQED), 2014 15th International Symposium on*, March 2014, pp. 148–155.

[23] K. Basu and P. Mishra, "Test data compression using efficient bitmask and dictionary selection methods," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 18, no. 9, pp. 1277 –1286, sept. 2010.

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

[25] 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.

[26] S. Ma, D. Pal, R. Jiang, S. Ray, and S. Vasudevan, "Can't see the forest for the trees: State restoration's limitations in post-silicon trace signal selection," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, ser. ICCAD '15. Piscataway, NJ, USA: IEEE Press, 2015, pp. 1–8.

[27] C. S. Zhu, G. Weissenbacher, and S. Malik, "Coverage-based trace signal selection for fault localisation in post-silicon validation," in *Proceedings of the 8th International Conference on Hardware and Software: Verification and Testing*, ser. HVC'12. Berlin, Heidelberg: Springer-Verlag, 2013, pp. 132–147.

[28] S. BeigMohammadi and B. Alizadeh, "Combinational trace signal selection with improved state restoration for post-silicon debug," in *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2016, pp. 1369–1374.

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

[30] X. Liu and Q. Xu, "On multiplexed signal tracing for post-silicon debug," in *2011 Design, Automation Test in Europe*, March 2011, pp. 1–6.

[31] K. Basu, P. Mishra, P. Patra, A. Nahir, and A. Adir, "Dynamic selection of trace signals for post-silicon debug," in *2013 14th International Workshop on Microprocessor Test and Verification*, Dec 2013, pp. 62–67.

[32] K. Han, J.-S. Yang, and J. Abraham, "Dynamic trace signal selection for post-silicon validation," in *26th International Conference on VLSI Design and 12th International Conference on Embedded Systems (VLSID)*, 2013, pp. 302–307.

[33] C. S. Zhu and S. Malik, "Optimizing dynamic trace signal selection using machine learning and linear programming," in *2015 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2015, pp. 1289–1292.

[34] J. S. Yang and N. A. Touba, "Expanding trace buffer observation window for in-system silicon debug through selective capture," in *VLSI Test Symposium, 2008. VTS 2008. 26th IEEE*, April 2008, pp. 345–351.

[35] E. Anis and N. Nicolici, "Low cost debug architecture using lossy compression for silicon debug," in *Design, Automation Test in Europe Conference Exhibition, 2007. DATE '07*, April 2007, pp. 1–6.

[36] E. Anis and N. Nicolici, "On using lossless compression of debug data in embedded logic analysis," in *2007 IEEE International Test Conference*, Oct 2007, pp. 1–10.

[37] J. S. Yang and N. A. Touba, "Improved trace buffer observation via selective data capture using 2-d compaction for post-silicon debug," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 2, pp. 320–328, Feb 2013.

[38] S. S. Muthyala and N. A. Touba, "Improving test compression with scan feedforward techniques," in *2014 International Test Conference*, Oct 2014, pp. 1–10.

[39] T. C. May and M. H. Woods, "Alpha-particle-induced soft errors in dynamic memories," *IEEE Transactions on Electron Devices*, vol. 26, no. 1, pp. 2–9, Jan 1979.

[40] P. Shivakumar, M. Kistler, S. W. Keckler, D. Burger, and L. Alvisi, "Modeling the effect of technology trends on the soft error rate of combinational logic," in *Proceedings International Conference on Dependable Systems and Networks*, 2002, pp. 389–398.

[41] S. Mitra, N. Seifert, M. Zhang, Q. Shi, and K. S. Kim, "Robust system design with built-in soft-error resilience," *Computer*, vol. 38, no. 2, pp. 43–52, Feb 2005.

[42] M. Nicolaidis, "Time redundancy based soft-error tolerance to rescue nanometer technologies," in *Proceedings of the 1999 17TH IEEE VLSI Test Symposium*, ser. VTS '99. Washington, DC, USA: IEEE Computer Society, 1999, pp. 86–.

[43] A. Sanyal, S. M. Alam, and S. Kundu, "A built-in self-test scheme for soft error rate characterization," in *2008 14th IEEE International On-Line Testing Symposium*, July 2008, pp. 65–70.

[44] A. Sanyal, K. Ganeshpure, and S. Kundu, "On accelerating soft-error detection by targeted pattern generation," in *8th International Symposium on Quality Electronic Design (ISQED'07)*, March 2007, pp. 723–728.

[45] A. Rubio, J. Pons, and R. Anglada, "A crosstalk tolerant latch circuit design," in *Proceedings of the 33rd Midwest Symposium on Circuits and Systems*, Aug 1990, pp. 653–656 vol.2.

[46] S. Kundu, S. T. Zachariah, Y.-S. Chang, and C. Tirumurti, "On modeling crosstalk faults," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 12, pp. 1909–1915, Dec 2005.

[47] H. Takahashi, K. J. Keller, K. T. Le, K. K. Saluja, and Y. Takamatsu, "A method for reducing the target fault list of crosstalk faults in synchronous sequential circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 2, pp. 252–263, Feb 2005.

[48] W.-Y. Chen, S. K. Gupta, and M. A. Breuer, "Test generation for crosstalk-induced faults: framework and computational results," in *Proceedings of the Ninth Asian Test Symposium*, 2000, pp. 305–310.

[49] A. Sanyal, K. Ganeshpure, and S. Kundu, "Test pattern generation for multiple aggressor crosstalk effects considering gate leakage loading in presence of gate delays," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 3, pp. 424–436, March 2012.

[50] S. Chun, T. Kim, and S. Kang, "Atpg-xp: Test generation for maximal crosstalk-induced faults," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 9, pp. 1401–1413, Sept 2009.

[51] X. Shi and N. Nicolici, "On-chip generation of uniformly distributed constrained-random stimuli for post-silicon validation," in *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov 2015, pp. 808–815.

[52] X. Shi and N. Nicolici, "On-chip cube-based constrained-random stimuli generation for post-silicon validation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 6, pp. 1012–1025, June 2016.

[53] A. Adir, E. Almog, L. Fournier, E. Marcus, M. Rimon, M. Vinov, and A. Ziv, "Genesys-pro: innovations in test program generation for functional processor verification," *IEEE Design Test of Computers*, vol. 21, no. 2, pp. 84–93, Mar 2004.

[54] L. Liu and S. Vasudevan, "Efficient validation input generation in rtl by hybridized source code analysis," in *2011 Design, Automation Test in Europe*, March 2011, pp. 1–6.

[55] M. Chen and P. Mishra, "Functional test generation using efficient property clustering and learning techniques," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 3, pp. 396–404, March 2010.

[56] F. Farahmandi and P. Mishra, "Automated test generation for debugging arithmetic circuits," in *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2016, pp. 1351–1356.

[57] H. Mangassarian, A. Veneris, S. Safarpour, M. Benedetti, and D. Smith, "A performance-driven qbf-based iterative logic array representation with applications to verification, debug and test," in *2007 IEEE/ACM International Conference on Computer-Aided Design*, Nov 2007, pp. 240–245.

[58] B. Le, H. Mangassarian, B. Keng, and A. Veneris, "Non-solution implications using reverse domination in a modern sat-based debugging environment," in *2012 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2012, pp. 629–634.

[59] J. Lv, P. Kalla, and F. Enescu, "Efficient groebner basis reductions for formal verification of galois field arithmetic circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 9, pp. 1409–1420, Sept 2013.

[60] F. Farahmandi, Y. Huang, and P. Mishra, "Trojan localization using symbolic algebra," in *22nd Asia and South Pacific Design Automation Conference, ASP-DAC 2017, Chiba, Japan, January 16-19, 2017*, 2017, pp. 591–597.

[61] D. Lin, S. Eswaran, S. Kumar, E. Rentschler, and S. Mitra, "Quick error detection tests with fast runtimes for effective post-silicon validation and debug," in *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*, ser. DATE '15. San Jose, CA, USA: EDA Consortium, 2015, pp. 1168–1173.

[62] D. Lin, E. Singh, C. Barrett, and S. Mitra, "A structured approach to post-silicon validation and debug using symbolic quick error detection," in *2015 IEEE International Test Conference (ITC)*, Oct 2015, pp. 1–10.

[63] A. Vali and N. Nicolici, "Satisfiability-based analysis of failing traces during post-silicon debug," in *2015 IEEE 24th North Atlantic Test Workshop*, May 2015, pp. 17–22.

[64] P. Taatizadeh and N. Nicolici, "Automated selection of assertions for bit-flip detection during post-silicon validation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 12, pp. 2118–2130, 2016.

[65] A. Vali and N. Nicolici, "Bit-flip detection-driven selection of trace signals," in *2016 21th IEEE European Test Symposium (ETS)*, May 2016, pp. 1–6.

[66] K. Goossens, B. Vermeulen, R. v. Steeden, and M. Bennebroek, "Transaction-based communication-centric debug," in *First International Symposium on Networks-on-Chip (NOCS'07)*, May 2007, pp. 95–106.

[67] A. M. Gharehbaghi and M. Fujita, "Transaction-based post-silicon debug of many-core system-on-chips," in *Thirteenth International Symposium on Quality Electronic Design (ISQED)*, March 2012, pp. 702–708.

[68] M. Dehbashi and G. Fey, "Transaction-based online debug for noc-based multiprocessor socs," in *2014 22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, Feb 2014, pp. 400–404.

[69] H. Zheng, Y. Cao, S. Ray, and J. Yang, "Protocol-guided analysis of post-silicon traces under limited observability," in *2016 17th International Symposium on Quality Electronic Design (ISQED)*, March 2016, pp. 301–306.

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

[71] lp_solver. Http://lpsolve.sourceforge.net/5.5.

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

[73] K. Rahmani, S. Ray, and P. Mishra, "Postsilicon trace signal selection using machine learning techniques," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. PP, no. 99, pp. 1–11, 2016.

[74] "Cubist Regression Model," https://www.rulequest.com/cubist-info.html.

[75] T. Zhang, X. Zhuang, S. Pande, and W. Lee, "Anomalous path detection with hardware support," in *Proceedings of the 2005 International Conference on Compilers, Architectures and Synthesis for Embedded Systems*, ser. CASES '05. New York, NY, USA: ACM, 2005, pp. 43–54.

BIOGRAPHICAL SKETCH

Kamran Rahmani received his B.Sc. degree in computer engineering from Sharif University of Technology, Tehran, Iran and the M.S. and Ph.D. degrees from the Department of Computer and Information Science and Engineering, University of Florida, Gainesville, FL, USA. He is a Staff Software Engineer at Medallia Inc., Palo Alto, CA, USA. His current research interests include post-silicon debug and validation, and reliable embedded systems. He received multiple travel grant awards from CISE department (2012, 2014) and University of Florida Office of Research (2012). He has served as a reviewer of several premier international conferences and journals.