LAYOUT-AWARE SIGNAL SELECTION FOR POST-SILICON DEBUG

By

PRATEEK THAKYAL

A THESIS PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE

UNIVERSITY OF FLORIDA

2014

To His Divine Grace A. C. Bhaktivedanta Swami Prabhupada

ACKNOWLEDGMENTS

I would like to sincerely thank my research advisor, Dr. Prabhat Mishra, without his guidance this thesis would not have been possile. I would also like to thank my commitee members, Dr. Ann Gordon-Ross and Dr. Scott Thompson, for their valuable suggestions.

I convey my deep gratitude to my Krishna House family (H. G. Kalakantha Prabhu and all the wonderful devotees) for all the love, encouragement, enthusiasm, and association with which I could complete my Master's with singing, dancing and feasting literally every single day of my two year stay.

This work was partially supported by National Science Foundation (NSF) grants CNS-0746261 and CCF-1218629.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Abstract of Thesis Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Master of Science

LAYOUT-AWARE SIGNAL SELECTION FOR POST-SILICON DEBUG

By

Prateek Thakyal

May 2014

Chair: Dr. Prabhat Mishra
Major: Electrical and Computer Engineering

Post-silicon debug is widely acknowledged as a bottleneck in System-on-Chip (SoC) design methodology. A major challenge during post-silicon debug is limited observability of internal signals. Existing approaches try to select a small set of beneficial trace signals that can maximize observability. Unfortunately, these techniques do not consider design constraints such as routability of the selected signals or routing congestion. Therefore, in reality, it may not be possible to route the selected signals. In this thesis, I propose a layout-aware signal selection algorithm that takes into account both observability and routing congestion. Experimental results demonstrate that the proposed approach can select routing friendly trace signals with negligible impact on observability. My studies also reveal that the proposed approach is beneficial for both post-silicon debug (in case of SoCs) and pre-silicon prototyping in FPGA.

CHAPTER 1
INTRODUCTION

Validation and debug are significant contributors in terms of cost (time) in the development cycle of System-on-Chip (SoC) designs. With the rapid increase in complexity, the percentage time spent on validation is growing with each new generation of products. With issues like variability – both local and global – in device characteristics, metal interconnects etc. becoming severe in every new technology node, it is no longer possible to sign-off the SoC designs based on just the RTL simulations. Thus, post-silicon debug has become extremely critical and indispensable part of the design cycle.

1.1 Observability - Bottleneck in SoC Post-Silicon Validation

System-on-Chip (SoC) designs are validated widely through netlist simulations at the pre-silicon stage. Although, debug at the simulation stage is difficult due to millions of internal signals, the validation engineers may iteratively choose any set of internal signals besides the inputs and the outputs. In other words, simulation provides complete observability of internal signals for debug. Despite rigorous validation at the pre-silicon stage, some bugs slip into silicon. Post-silicon debug techniques are used to pin point and root cause such bugs.

Post-silicon-debug has a major constraint in terms of number of signals which may be observed. Unlike netlist simulation based debug at pre-silicon stage, in post-silicon debug only the signals either connected to the I/O pins or debug devices can be observed. To improve observability of internal signals, trace buffers are widely used in integrated circuits. The trace buffers store the values of selected signals for a number of cycles at runtime, and the values may be read during debug. The area constraint in trace buffers limits the number of signals which may be observed. Thus, limited observability is a major challenge in the post-silicon debug.

1.2 Design Constraints - A Concern for the Signal Selection

As the number of signals which may be observed in post-silicon debug is limited, it is imperative to choose the signals which maximize restorability. Restorability of a signal is representative of the number of unobserved signals that may be reconstructed, if the given signal is observed.

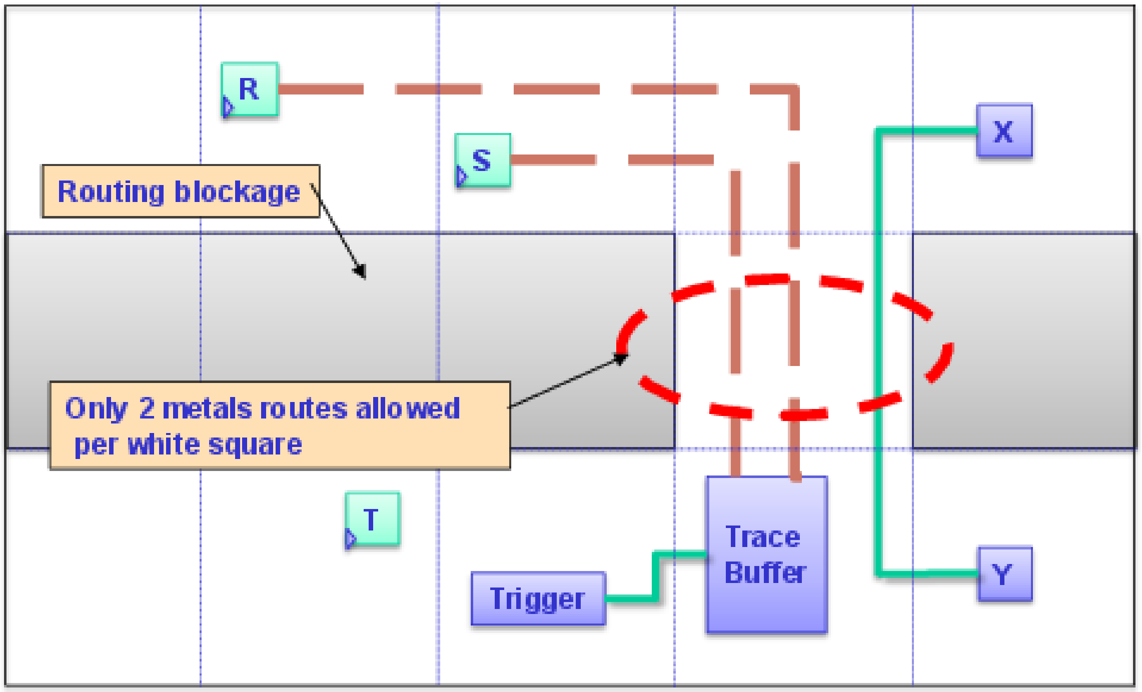
A wide variety of contemporary solutions exist ranging from extremely fast metric based restorability evaluation [7] [1] [11] [12], to simulation based highly accurate approach [3] and even a hybrid approach [10]. However, all the current approaches overlook the design constraints such as routing congestion. Signal selection for the debug is typically done in the final stages of the design cycle when most of the routing is frozen. Hence, although a designer may get the best possible set of signals from restorability perspective, it may not be possible to route them. One may argue that selection of the signals at an earlier stage may resolve the routability issue, but due to the dynamics in netlist changes over the design cycle it may not be possible to select the signals at an earlier stage.

Striking a right balance between the restorability and routability is a major challenge. I propose prioritizing the signals which are near the trace buffer in an effort to reduce the Manhattan distance (wire-length). Our algorithm can be used in tandem with the previously proposed algorithms for layout-aware signal selection towards post-silicon debug. For example, two different signals S1 and S2 may provide same restorability. However, it may be easier to connect to the signal which is near the trace buffer.

Consider an illustrative example in Figure 1-1. Assume that a designer may trace at most 2 signals. For simplicity, assume that only one level of metal is allowed and each white space block may accommodate a maximum of 2 vertical routes due to routing constraints. Assume that signals X and Y are pre-routed and signal selection is to be done at this stage among signals R, S and T. The trigger block triggers the trace buffer to start tracing the signals. In the figure, gray represents an area which is blocked for any further routing. In this case, the existing signal selection algorithm may select R and S, if they provide higher restorability compared to (R,T) and (S,T). However, it is not possible to route both the signals due to the congestion constraint. Thus the signal T, which is not as good as R or S in terms of the restorability needs to be chosen. It may be noted that as the distance between two connected cells (signals) increases, the probability of hitting a blockage gets higher and higher. Hence, it is better to give priority to the signals which are near the trace buffer.

A specific scenario would be to choose between two signals with equal restorability. Existing signal selection algorithms randomly select one of them. However, it is better to choose the signal closer to the trace buffer in such a scenario.

Figure 1-1. Illustration of importance of congestion



1.3 Debug at Different Abstraction Levels

Validation and debug is done at multiple levels. System validation entails that the system be subjected to all possible states. It may seem like that, a relatively small design netlist with just 10 inputs may be validated with about 2^{10} (1024) test vectors.

However, this may not necessarily be true. If the given circuit is with only combinational logic, certainly 1024 test vectors should suffice. But, in case of sequential logic to do an exhaustive analysis just for a sequence of two states would take ${}^{1024}C_2$ (half a million) test vectors. Scaling it up to more number of inputs and states would lead to state space explosion.

Netlist simulation based debug is widely used during the RTL development stage. Designers write testbenches to catch any bugs. The process of updating the netlist and reruning the simulation helps weed out a lot of first level errors. Debugging at this stage is relatively easy. The designers can trace a large number of signals for a large simulation time window. However, simulation at this level has a problem - it runs extremely slow compared to actual hardware. Thus, the validation in simulation is limited by the simulation duration. A lot of bugs may not occur in the intial stages of execution but may get triggerd later.

The run-time bottlneck in simulation based debug is a motivation for FPGA based validation. FPGAs, though slower than actual hardware (SoC), run at a much higher speed than the software simulations and hence, can potentially help in hunting down some intermittent bugs which are missed during the software simulation stage. Though, FPGAs have a lower visibility into internal signals compared to simulations, the designer can reconfigure the FPGAs to select a different set of signals for tracing. The cost of validation and debug at this stage is still relatively low and only overhead is the synthesis time. Table 1-1 highlights the pros and cons of debugging at each level.

1.4 Thesis Contributions and Organization

The thesis has made two major contributions : i) layout-aware signal selection, and ii) exploration of layout-aware signal selection for FPGAs. Layout-aware signal selection strives for layout congestion reduction while selecting the most benefitial signals, by giving precedence to the signals near the trace buffer. The framework is implemented using Cadence Encounter Design Integration (EDI) and Xilinx ISE.

Table 1-1. Advantages and disadvantages of validation at different abstraction levels

Stage	Visibility	Window size testable	Window size traceable	Cost
RTL	Complete visibility of internal signals	Very small	Large	Minimal
FPGA	Partial visibility of internal signals	Very long	Medium	Low
SoC	Limited to signals selected at design	Very long	Small	Extremely high

The rest of the thesis is organized as follows. Chapter 2 surveys existing works related to signal selection for SoCs and FPGAs. Chapter 3 describes our layout-aware signal selection algorithms in two scenarios, and presents the experimental results. Chapter 4 investigates the application of layout-aware signal selection in FPGA. Finally, Chapter 5 concludes the thesis.

CHAPTER 2
RELATED WORK

Signal selection algorithms can be classified into three categories: (i) metric based, (ii) simulation based and (iii) hybrid of metric- and simulation-based techniques. The metric based algorithms compute restorability of untraced signals for a given signal and try to maximize the restorability by adding signals to the trace selection. Restorability or restoration ratio (RR) is defined as the ratio of the total number of signal states that can be restored over the total number of selected signal states. Following equation defines restoration ratio:

$$RR = \frac{No.\ of\ Restored\ Signal\ States + No.\ of\ Selected\ Signal\ States}{No.\ of\ Selected\ Signal\ States}$$

2.1 Signal Selection for Post-silicon Debug

Ko and Nicolici [7] defined “forward” and “backward” restorability for signal selection. Basu and Mishra [1] improved signal selection by providing emphasis on selecting beneficial neighbors. Metric-based methods select signals by iterative additions of beneficial signals till the trace buffer is full. Although metic-based algorithms have an advantage of being extremely fast compared to the simulation-based approach, their restoration performance is not good. Simulation-based trace signal selection starts with all the signals as observable, and then iteratively eliminates signals which have minimum impact on the restoration ratio on removal [3]. Simulation based methods provide higher state restoration ratio, but have a longer runtime. Li and Davoodi [10] developed a hybrid of the metric and simulation-based signal selection to select trace signals. Hybrid approach first identifies top candidates using metric evaluation and then uses simulation to accurately evaluate the state restoration ratio for each candidate.

2.2 Layout-aware Approaches

Due to ever-increasing complexity of SoC designs, layout friendliness has been investigated by various researchers. Layout-awareness has been used as a key criteria in scan-chain reordering [4], fault pattern generation [9] and memory BIST synthesis [8].

However, layout has not been considered in the context of signal selection in integrated circuits.

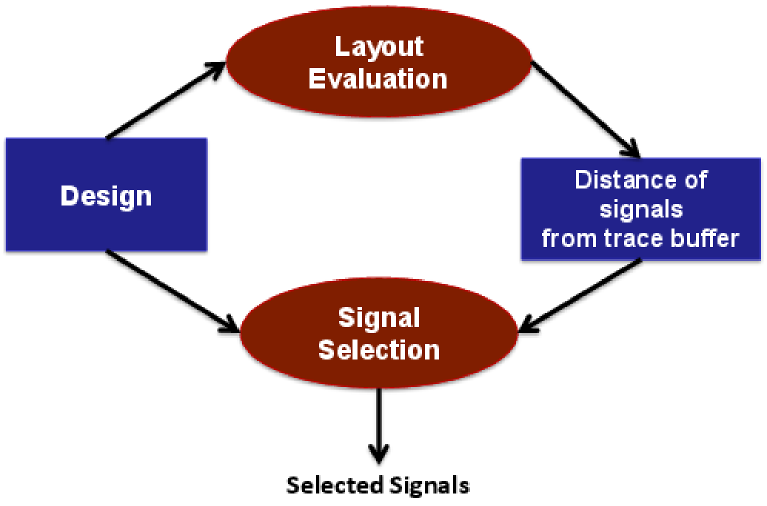
2.3 Signal Selection for FPGAs

Hung and Wilton [5] suggested a new metric, “post-silicon debug difficulty” for signal selection in FPGA. For a set of selected signals, the metric indicates the number of circuit states that can be activated. The reasoning used here is that the designers do not go bit-by-bit to hunt down a bug. Rather, if the designer knows the state closely, then he may be able to brainstorm on the possible reasons of the bug. Hung and Wilton[6] also presented graph centrality as one of the metrics for faster selection of the signals. However, quantitative comparison between “post-silicon debug difficulty” and restorability based algorithms has not been studied. The restorability based algorithms have not been applied to the FPGA. My work is a first attempt on applying the restorability based algorithms on the FPGAs.

CHAPTER 3
LAYOUT-AWARE SIGNAL SELECTION

Figure 3-1 provides an overview of our proposed layout-aware signal selection. Layout of design is first evaluated to get distances of signals from the trace buffer. Signal selection module takes two inputs – the design (netlist), and signal distance values – and produces the selected signals. The following sections describe these two important steps: Manhattan distance calculation using the layout, and layout-aware signal selection.

Figure 3-1. Layout-aware signal selection flow



3.1 Manhattan Distance Calculation

One of the major challenges in placing and routing a design is the routing congestion. Routing congestion is defined as the percentage of tracks blocked of the total tracks available for routing. Many metrics provide an evaluation criteria for layout congestion. Elucidian distance, Manhattan distance, and total wire-length may be used as a representative of the congestion in the design. Collection and interpretation of congestion information is non-trivial with the present tools. Using exact wire-length may be compute intensive.

Thus wire-length estimation techniques such as half-perimeter wire-length, squared-Elucidian distance, minimum Steiner-tree wire-length, minimum spanning tree wire-length or complete-graph wire-length may be used [2]. I use wire-length estimate as the congestion criteria. All prospective selected flip-flops need to connect to the trace buffer in a star fashion, with the trace buffer at the center. Hence, half-perimeter wire-length is best suited for this purpose. Half-perimeter wire-length of any two connected nodes is equal to the Manhattan distance between them.

Manhattan distance is defined as the sum of the absolute values of differences in the X and Y coordinate values of any two points.

$ManhattanDistance = (|x_{tb} - x_i| + |y_{tb} - y_i|)$.

For layout-awareness, the total Manhattan distances of all the selected flip-flops to the trace buffer needs to be minimized. Total Manhattan Distance(TMD) is given by following equation:

$TMD = \sum_{i=1}^{TraceBufferSize} (|x_{tb} - x_i| + |y_{tb} - y_i|)$

I use normalized-Manhattan distance as a layout-awareness metric for different signals. Manhattan-distance (from the trace buffer) to all the prospective signals is calculated and normalized with respect to the farthest prospective signal.

Maximum Manhattan distance, among all prospective flip-flops from the trace buffer can be computed as:

$MD_{max} = max(MD_i)$

Similarly, normalized Manhattan distance is computed as:

$MD_I = MD_i / MD_{max}$

Normalized Manhattan distance is used in signal selection algorithms to prioritize signals based on the proximity to the the trace buffer.

3.2 Layout-aware Signal Selection

The basic idea of our algorithm is that the normalized Manhattan distance values (computed in the previous section) are used with signal selection parameters like restorability and visibility to either eliminate or add a flip-flop into the set of flip-flops selected for the trace buffer. Algorithm 1 shows the major steps during layout-aware signal selection. It is important to note that our algorithm can be used on top of any existing signal selection procedure by invoking the specific procedure in step 6. In other words, the *signalSelection*() subroutine in the algorithm can be replaced by any of the existing signal selection algorithms.

In the algorithm, the first step is to identify the trace buffer in the layout and get its coordinates. In step 2, coordinates of all the flip-flops are queried and their Manhattan distance from the trace buffer is calculated. Step 3 computes the maximum value among the Manhattan distances. In step 4, the Manhattan distance of all the signals is normalized with respect to the maximum Manhattan distance calculated in step 3. Step 5 identifies the restoration ratio offered by all the signals. Restoration and Manhattan distance values of all the signals are used to select the signals in step 6. Finally, the algorithm returns the selected signals.

In the remainder of this section, I describe how step 6 can invoke two specific signal selection algorithms - simulation-based and metric-based signal selection.

3.2.1 Simulation-based Approach

Simulation-based signal selection uses iterative elimination of less beneficial signals. In every iteration, simulations are used to determine the impact of eliminating a signal among the remaining signals. Signal with minimal impact on the restoration ratio is eliminated every cycle. Elimination continues till the number of elements in the observable set is equal to the trace buffer capacity. Layout awareness is added by scaling the visibility of the flip-flop, based on the normalized-Manhattan-distance. Signal with the minimum impact on visibility, and minimum reduction in Manhattan distance is eliminated.

Steps 1, 2, and 3 in Algorithm 2 are initialization steps. Step 2 marks all the flip-flops as selected and puts them in set $\{T\}$. The while loop in step 4 eliminates the elements in $\{T\}$ till the number of elements remaining are equal to the width of the trace buffer. Step 7 calculates the visibility which the set $\{T\}$ has into the design. Step 9 iterates over the whole set $\{T\}$, and creates a new set $\{V\}$ with just one element removed from $\{T\}$. The visibility is calculated for each set $\{V\}$. The signal to be eliminated is the one for which set $\{T\}$ - "element" has minimal impact on the visibility compared to set $\{T\}$ alone. The algorithm returns selected signals. In addition to the default elimination, two other approaches may be used for layout-aware signal elimination. One approach is based on a user-specified maximum Manhattan distance threshold, in which the signals beyond the threshold are eliminated without visibility evaluation. In the second approach, the visbility itself is scaled by the normalized Manhattan distance of the signal. The approaches are captured in the *siganlToBeEliminated* function. This ensures, that the signals which do not have much impact on visibility, and are far from the trace buffer are eliminated first.

Algorithm 1: Layout-aware Signal Selection

```
Inputs : Design(netlist), layout
Output: Selected Signals
/* Find (x,y) coordinates of the trace buffer.                        */
1 $(x_{tb}, y_{tb}) = getLocation(traceBuffer, Layout)$;
/* Calculate Manhattan distance of all the signals.                   */
2 $\bigvee_{i=1}^{Signals} dist_i = |x_{tb} - x_i| + |y_{tb} - y_i|$ ;
/* Get the maximum Manhattan Distance.                                */
3 $max_{dis} = max(\bigvee dist_i)$ ;
/* Compute normalized-Manhattan distance for all the signals.         */
4 $\bigvee_{i=1}^{Signals} ndist_i = dist_i / max_{dist}$ ;
/* Get Restoration Ratio for all the signals.                         */
5 $\bigvee_{i=1}^{Signals} restorability_i = getRestorability(signal, netlist)$;
/* Invoke signal selection with normalized-Manhattan distance and
   restorability.                                                     */
6 $SelectedSignals = signalSelection(ndist, restorability)$ ;
Return: Selected Signals
```

Algorithm 2: Layout-aware Simulation-based Signal Selection

```
Inputs : trace buffer width $w$, simulation function to get restoration ratio SRR
         $fsrr()$, layout priority offset constant: $\alpha$, maximum Manhattan distance
         threshold: $\beta$
Output: Selected Signals
1  Load the Circuit;
2  Let {T} : Set of all flip-flops in the design;
3  Associate values with element in {T} ;
4  while $w <$ Number of Elements in {T} do
5      Set Minimum Restoration Ratio: $minRR = 1000000$;
6      Let $j$: Index to the Element to be Eliminated in this iteration $= 0$;
       /* Calculate visibility of the signals in set {T}                */
7      Let $\vartheta = Visibility(T)$;
8      Let $\nu$ be the change in visibility $= 1000000$;
9      for $i \leftarrow 1$ to Number of Elements in $T$ do
10         Let $\{V\} \leftarrow \{T\} - \{ith$ FF$\}$ ;
           /* Evaluate $\delta$visibility & scale with routing weight   */
11         $bool = signalToBeEliminated(\vartheta, Visibility(V), \beta, ndist_i)$;
12         if $bool$ then
13             $j = i$
14         end
15     end
       /* Remove element based on signalToBeEliminated function        */
16     {T} = {T} - jth element;
17 end
Return: {T}
```

3.2.2 Metric-based Approach

The algorithm in this approach tries to maximize restoration ratio, while adding new signals to trace buffer, until the trace buffer gets full. The algorithm is modified to evaluate the combined impact of restoration ratio and normalized Manhattan distance. Separate weight is used for the restoration ratio and normalized Manhattan distance (ndist). The weight is then varied from 0 to 1 at a step size of 0.1. Algorithm 3 provides the details of the approach. The algorithm uses *selectBestSignal* function which may be implemented in two different ways. One way may be to skip all the signals which are not within certain Manhattan distance β. Other way is to use the following equation for evaluation of goodness of a signal from both restorability and Manhattan distance perspective.

$$SelectionCriteria = r * (\alpha - ndist_i) + (1 - r) * Restorability_i \qquad (3–1)$$

The algorithm has two nested loops. Within the *while* loop in step 6, a temporary set of signals (V) is created by adding *ith* element of set $\{T\}$ to set of selected signals. This set is created to evaluate the effectiveness of the *ith* signal. Each time *while* loop in step 6 is invoked it generates a set of selected signals for the given routing weight (rw). The *for* loop in step 4 evaluates the sets of selected signals, for different values of rw, generated by the internal *while* loop. Steps 12 and 13 decide if a signal is to be added to a set of selected signals or not. Finally, the algorithm returns the set of signals which would be best from the restoration ratio and Manhattan distance perspective.

Algorithm 3: Layout-aware Metric-based Signal Selection

```
Inputs : netlist, trace buffer width $w$, maximum Manhattan distance threshold: $\beta$
Output: Selected Signals
1  Load the Circuit; Let $finalRR$ be 0;
2  Let $\{T\}$ : Set of all signals in the design; Let $\{SS\}$ : Set of Selected Signals = $\phi$;
3  Let $rw$ be the routing weight of the signal under consideration;
4  for $rw \to 0$ to $1; rw+ = 0.1$ do
5      Let $\{S\}$ : Set of all selected signal Flip-Flops = $\phi$;
6      while $w >$ Number of elements in $\{S\}$ do
7          Max Restoration Ratio: $maxRR = 0$;
8          Let $j$: Index to the element to be added in the current iteration $= 0$;
9          for $i \leftarrow 1$ to Number of Elements in $\{T\}$ do
10             Let $\{V\} \leftarrow \{S\} \bigcup ith$ FF in {T} ;
11             Let $rrv = stateRestorationRatio(V)$;
12             $bool = selectBestSignal(i, ndist_i, rrv, maxRR)$;
13             if $bool$ then
14                 $j = i; maxRR = rrv$;
15             end
16         end
17         {T} = {T} - jth $element$;
18         {S} = {S} $\bigcup$ jth $element$;/* Add signal as selected */
19     end
20     if $finalRR < stateRestorationRatio(S)$ then
21         $SS \leftarrow S; finalRR \leftarrow stateRestorationRatio(S)$;
22     end
23 end
Return: {SS} is the set of selected signals
```

3.3 Experiments

The section describes the experimental setup and presents the results.

3.3.1 Experimental Setup

ISCAS'89 benchmarks were used for the evaluation. To emulate the layout availability, the layout was first generated by modifying the Verilog description. I added a trace buffer of a given width and connected to random internal nets of the design. Random internal nets were chosen to avoid biasing the layout for any signals. I used Cadence Encounter Digital Implementation tool (EDI) to synthesize the modified Verilog design. A design exchange format (DEF) file was dumped from the design synthesis tool. Manhattan distance of each of the flip-flops to the trace buffer was tabulated using the coordinates in the DEF. Signals were selected giving more weight to the flip-flops near to the trace buffer. Final restoration ratio was then computed for the selected signals.

I compared the total Manhattan distance of all the selected signals from the trace buffer between existing algorithms and our proposed layout-aware signal selection.

3.3.2 Comparison with Metric-based Signal Selection

Table 3-1 compares our approach with existing metric-based signal selection [1]. The first column in the table specifies the benchmark used for evaluation. I selected 32 trace signals. Restoration ratio from the existing approach and our layout-aware approach are specified in second and third columns, respectively. Degradation in the restoration ratio is provided in fourth column. The fifth and sixth columns give the total Manhattan distance of all the selected signals with the existing approach and our approach, respectively. It is important to note that the existing approaches did not consider layout and therefore I computed the Manhattan distance numbers for the signals selected by existing approaches.

The results show that the Manhattan distance can be significantly reduced with minor impact on restoration ratio. For a trace buffer width of 32, Table 3-1 shows an

Table 3-1. Comparison with metric-based restoration ratio

Benchmark	Restoration Ratio			Manhattan Distance		
	Basu & Mishra [1]	Layout-aware	% change	Basu & Mishra [1]	Layout-aware	% change
s9234*	2.66	1.63	-38.72	12324.6	8346.6	-32.28
s13207*	8.30	6.18	-25.54	18366.6	8347.8	-54.55
s35932	35.00	24.92	-28.80	27594.0	19600.8	-28.97
s38584	20.00	23.81	+19.05	21869.4	13146.6	-39.89
Average	16.49	14.14	-14.28	20038.7	13629.0	-31.99

*Restoration ratio not provided in [1] and had to be generated

average 32% (peak 55%) improvement in the Manhattan distance across different benchmarks, with an average 14% penalty on the restoration ratio.

Two important scenarios can be observed in the results: i) Manhattan distance improves at the cost of restoration ratio, and ii) both Manhattan distance and restoration ratio improve. The first scenario is expected as the algorithm deliberately chooses signals near the trace buffer, although they may not provide the best restorability.

The second case is counter-intuitive and is seen for s38584. Our analysis revealed that in the existing approach, while selecting signals, forward and backward restoration is done just once. However, for calculating the final restoration ratio, forward and backward restoration is done multiple times till all the values saturate. During iterative addition of signals to the trace buffer, a signal may indicate higher restoration ratio. However, with multiple forward and backward restoration cycles in the final restoration ratio calculation, a signal which was not so good during the selection may show a higher restoration ratio value. In other words, the probability calculation during signal selection is not identical to the restoration calculation. Therefore, the existing approach does not select the best possible signals in each iteration. For these benchmarks, the perturbation caused by layout awareness, in fact, enabled the selection of better signals.

Another case was observed for trace buffer width 8 (not shown in the table) where Manhattan distance improves, but the restoration ratio remains constant. Although, by choosing a different set of signals the restoration ratio does not change, but the

Manhattan distance significantly comes down. This is due to the fact that two signals can be equally beneficial from restoration perspective but one has less Manhattan distance from the trace buffer compared to the other. Our approach has chosen the one with least distance whereas the existing approach has randomly chosen one of them, which happens to have higher Manhattan distance.

It may not be desirable to have a good Manhattan distance at the cost of a very high degradation in the restoration ratio. Designers may choose a value better suited for the restoration ratio in such a case. The priority for Manhattan distance can be reduced in such a case, by choosing low values of routing weight. For example, Figure 3-2 provides detailed data points for s9234 benchmark. Figure 3-2 presents restoration ratio values corresponding to the routing weights. It shows that the restoration ratio comes down while the Manhattan distance is reduced through signal selection. In this case, for buffer width of 32, designers may choose a lower routing weight (0.1) to get a minimal impact on the restoration ratio.

3.3.3 Comparison with Simulation-based Signal Selection

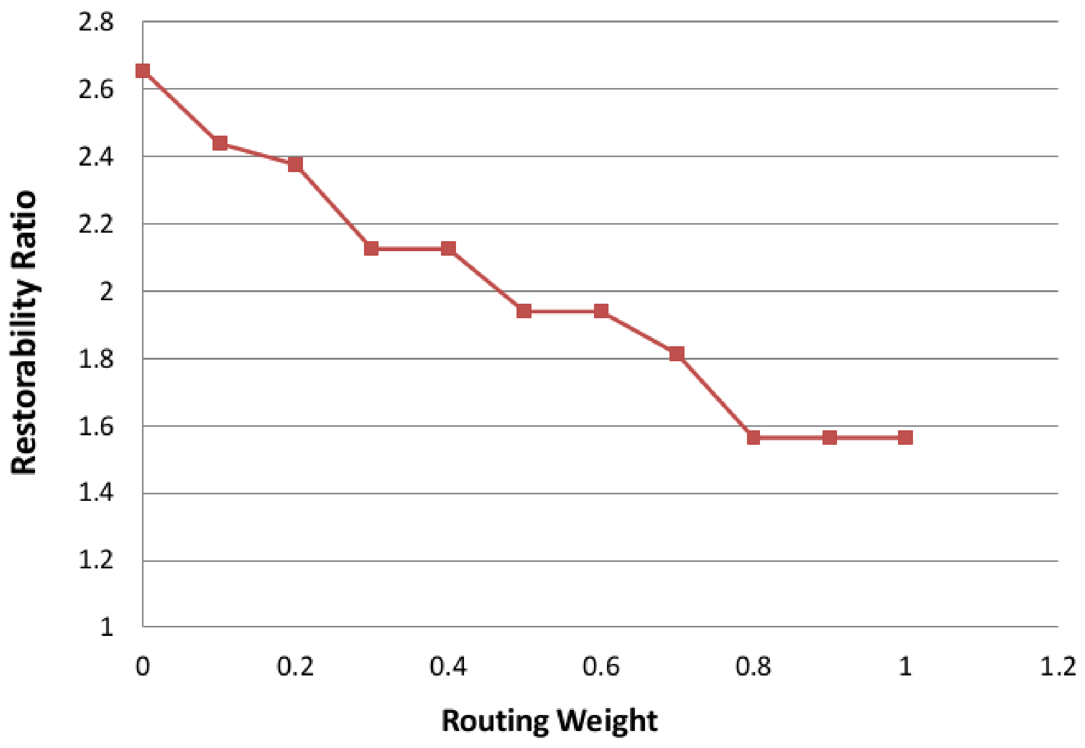
Table 3-2 provides data for the analysis of benchmarks using simulation based signal selection [3]. The columns used in Table 3-2 are same as that used in Table 3-1. For a trace buffer width of 32, the algorithm shows an average 31.9% improvement in the Manhattan distance, with an average 18.95% penalty on the restoration ratio.

Table 3-2. Comparison with simulation-based restoration ratio

Benchmark	Restoration Ratio			Manhattan Distance		
	Chatterjee et al. [3]	Layout-aware	% change	Chatterjee et al. [3]	Layout-aware	% change
s9234	4.18	2.56	-38.76	13860.6	13449.6	-2.97
s13207*	9.47	8.56	-9.61	22164.6	13764.0	-37.90
s35932	43.13	34.39	-20.26	14109.0	12153.6	-13.86
s38584	27.00	22.39	-17.07	38292.0	20853.0	-45.54
Average	20.95	16.98	-18.95	22106.6	15055.1	-31.90

*Restoration ratio not provided in [3] and had to be generated

Figure 3-2. Restoration ratio for s9234



CHAPTER 4
SIGNAL SELECTION IN FPGA

4.1 Introduction

Signal selection differs in the FPGA domain compared to the SoC domain in terms of usage. While for SoCs the selected signals cannot be altered once fabricated (unless a larger set is used with dynamic selection), in FPGAs the validation engineer may change the signals for observation through reconfiguration based on debug requirements.

Software based netlist debug and SoC post-silicon debug are different in terms of the ability to observe internal signals. While software-based debug provides ability to observe all the internal variables or signals, the post-silicon debug offers very little visibility of the internal signals and that too with constraints in terms of time snapshot duration and the number of signals to be observed. FPGAs bridge this gap.

Although the FPGAs may not supply the freedom to choose any number of variables of IDE (integrated development environments), the power to alter the trace signals through reconfiguration, without any area or delay penalty, is a huge advantage.

One can argue that since FPGAs can be reconfigured, there is little room for signal selection algorithms. However, it is not the case. Manual signal selection is based on trial-and-error and hence can result in large number of iterations. Moreover, in the manual selection it is difficult to prevent closely co-related signals. Thus, signal selection algorithms can play a significant role in efficient debug and reduction in iterations. The initial runs may be able to partition the susceptible regions. Signal selection may then be further applied on the susceptible region, thereby narrowing down the hunt.

Routing congestion challenge is also important in case of FPGAs . Hence, it is imperative to select signals considering layout constraints. I use Manhattan distance as input to select the signals. The signal selection criteria and algorithms used for the FPGA are exactly same as described in the previous chapter.

4.2 Experiments

4.2.1 Experimental Setup

ISCAS'89 benchmarks were used for the evaluation. I generated a BRAM using the *coregen* application of ISE and used it as the trace buffer. The design was then synthesized in the Xilinx framework and user constraints file (.ucf) was dumped from the floorplan editor. The constraints file has the locations of all the flip-flops and the BRAM cells used in the design. Using the coordinates in the constraints file, Manhattan distances of each and every flip-flop to the trace buffer was calculated and normalized. The signal selection was then run giving more precedence to the signals near the trace buffer.

I compared the total Manhattan distance of all the selected signals from the trace buffer using existing algorithms and our proposed layout-aware signal selection.

4.2.2 Results

Signal selection evaluation was done using both, metric-based and simulation-based approaches. Table 4-1 compares with metric-based approach. Results show on an average 49% reduction in the Manhattan distance with a restoration ratio penalty of 18.89%. Added routing-constraints caused perturbations in the evaluation resulting in a higher restoration ratios.

Table 4-1. Comparison with metric-based signal selection for FPGAs

	Restoration Ratio			Manhattan Distance		
Benchmark	Basu & Mishra [1]	Layout-aware	% change	Basu & Mishra [1]	Layout-aware	% change
s9234*	2.66	2.97	11.65	6062	2063	-65.97
s13207*	8.30	9.58	15.42	6528	3133	-52.01
s35932	35.00	24.73	-29.34	8980	5778	-35.66
Average	15.32	12.43	-18.89	7190	3658	-49.12

*Restoration ratio not provided in [1] and had to be generated

Table 4-2 presents data for simulation-based approach. The results show on an average 23% improvement in the Manhattan distance and a 17.6% degradation in restoration ratio.

Table 4-2. Comparison with simulation-based signal selection for FPGAs

Benchmark	Restoration Ratio			Manhattan Distance		
	Chatterjee et al. [3]	Layout-aware	% change	Chatterjee et al. [3]	Layout-aware	% change
s13207*	9.47	8.54	-9.82	5217	2320	-55.53
s35932	43.13	34.39	-20.26	7327	6261	-14.55
s9234	4.18	3.87	-7.42	6080	5744	-5.53
Average	18.93	15.60	-17.58	6208	4775	-23.08

*Restoration ratio not provided in [3] and had to be generated

CHAPTER 5
CONCLUSION

Post-silicon validation and debug are critical components of the SoC design methodology. The challenge is to select beneficial signals for debug while considering the design constraints like routability. Existing approaches, though efficient at identifying good signals, overlook the design constraints, thereby selecting some signals which may not be routable. I developed techniques to incorporate layout-awareness in the existing set of algorithms towards identification of signals which are not only beneficial from the debug perspective but also from a routing perspective. My technique further gives designer freedom to customize the weight for layout-awareness to suit the design needs. My approach can be applied on top of the existing signal selection techniques such as metric based [1] and simulation based approach [3]. Experimental results demonstrated that my approach can select layout-friendly signals with negligible impact on restorability.

REFERENCES

[1] K. Basu and P. Mishra "RATS: Restoration-Aware Trace Signal Selection for Post-Silicon Validation." *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21(4),2013, pp. 605–613.

[2] Yao-Wen Chang "Placement." 2013.

URL http://cc.ee.ntu.edu.tw/ywchang/Courses/PD/unit5.pdf

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

[4] P. Gupta, A.B. Kahng, I.I.Mandoiut and P. Sharma "Layout-aware scan chain synthesis for improved path delay fault coverage." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol 24(7), 2005, pp. 1104–1114.

[5] E. Hung and S.J.E. Wilton "On evaluating signal selection algorithms for post-silicon debug." *International Symposium on Quality Electronic Design (ISQED)*, 2011, pp. 1–7.

[6] E. Hung and S.J.E. Wilton "Scalable Signal Selection for Post-Silicon Debug." *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. 21(6), 2013, pp. 1103–1115.

[7] H.F. Ko and N. Nicolici "Automated Trace Signals Identification and State Restoration for Improving Observability in Post-Silicon Validation." *Design, Automation and Test in Europe (DATE),*2008, pp. 1298–1303.

[8] A. Kokrady, C. P. Ravikumar and N. Chandrachoodan "Layout-Aware and Programmable Memory BIST Synthesis for Nanoscale System-on-Chip Designs." *Asian Test Symposium (ATS),* 2008, pp. 351–356.

[9] J. Lee and S. Narayan and M. Kapralos and M. Tehranipoor "Layout-Aware, IR-Drop Tolerant Transition Fault Pattern Generation." *Design, Automation and Test in Europe (DATE),* 2008, pp. 1172–1177.

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

[11] Xiao Liu and Qiang Xu "Trace signal selection for visibility enhancement in post-silicon validation." *Design, Automation Test in Europe Conference Exhibition (DATE),* 2009, pp. 1338–1343.

[12] S. Prabhakar and M. Hsiao "Using Non-trivial Logic Implications for Trace Buffer-Based Silicon Debug." *Asian Test Symposium (ATS),* 2009, pp. 131–136.

BIOGRAPHICAL SKETCH

Prateek Thakyal received his Bachelor of Technology in Electronics and Instrumentation Engineering from National Institute of Technology, Rourkela, India in 2006. He worked in Texas Instruments on reliability, and power integrity from 2006 to 2012. He completed his Master of Science in Department of Electrical and Computer Engineering from University of Florida in 2014.