Layout-aware Selection of Trace Signals for Post-Silicon Debug

Prateek Thakyal
ECE, University of Florida, USA
Email: thakyal@ufl.edu

Prabhat Mishra
CISE, University of Florida, USA
Email: prabhat@cise.ufl.edu

***Abstract*—Post-silicon debug is widely acknowledged as a bottleneck in SoC design methodology. A major challenge during post-silicon debug is the 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. We propose a layout-aware signal selection algorithm that takes into account both observability and routing congestion. Our experimental results demonstrate that our proposed approach can select routing friendly trace signals with negligible impact on observability.**

***Keywords*-Post-Silicon Debug, Signal Selection, Layout.**

I. Introduction

Validation and debug are significant contributors in terms of cost (time) in the development cycle of System-on-Chip (SoC) designs. With rapid increase in design complexity, the percentage of time spent on validation is growing with each new generation of products. It is no longer possible to capture all bugs during pre-silicon validation. Thus, post-silicon debug has become extremely critical and indispensable part of the design cycle.

SoC designs are validated using 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 buffer limits the number of signals that can be observed. Thus, limited observability is a major challenge in the post-silicon debug.

As the number of signals which may be observed in post-silicon debug is limited, it is imperative to choose the signals that 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 [1], [5], [9], [10], to simulation-based highly accurate approach [3] and even a hybrid approach [8]. However, all the current approaches overlook the design constraints such as routing congestion. Signal selection 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.

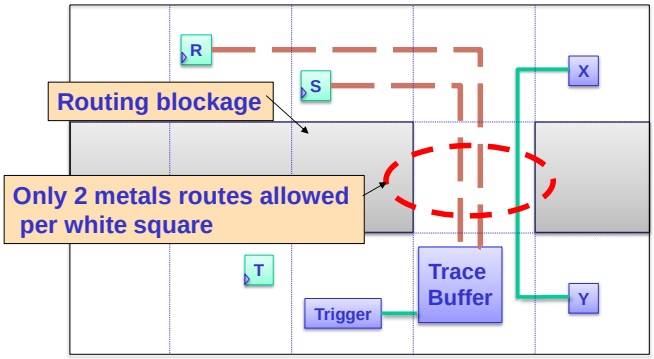
Striking a right balance between the restorability and routability is a major challenge. We propose prioritizing the signals which are near the trace buffer in an effort to reduce the Manhattan distance (wire-length). Our algorithm can be used in tandem with the previously proposed algorithms for layout-aware 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. 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. Of course, it is also important to ensure

that routability improvement does not introduce significant penalty on observability (restorability).

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. Our experimental results demonstrate that the proposed approach can significantly improve the routability with minor impact on the restorability.

Figure 1. Illustration of importance of congestion



The rest of the paper is organized as follows. Section II outlines the related work. Section III describes our layout-aware signal selection algorithm. Section IV presents the experimental results. Finally, Section V concludes the paper.

II. 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:

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

A. Signal selection for post-silicon debug

There are various approaches for metric-based signal selection [1], [5], [9], [10]. Basu and Mishra [1] improved signal selection by providing emphasis on selecting beneficial neighbors. Although metric-based algorithms have an advantage of being extremely fast compared to the simulation-based, their restoration performance is not good. Simulation-based trace signal selection starts with all the signals as observable, and then iteratively eliminates state 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 [8] used a hybrid of the metric- and simulation-based signal selection to come up with 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.

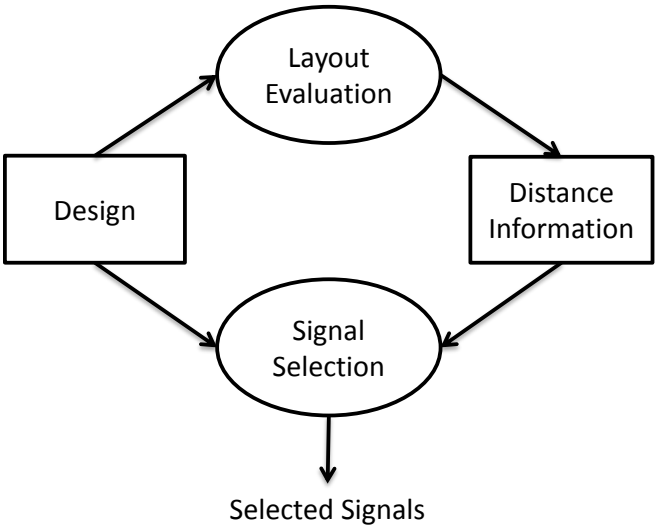
B. Layout-aware Approaches

Due to ever-increasing complexity of SoC designs, layout friendliness has been looked at by various researchers. Layout-awareness has been used as a key criteria in scan-chain reordering [4], fault pattern generation [7] and memory BIST synthesis [6]. However, layout has not been considered in the context of signal selection for integrated circuits.

III. Layout-Aware Signal Selection

Figure 2 provides an overview of our proposed layout-aware signal selection.

Figure 2. Layout-aware signal selection flow



The basic idea of our algorithm is that the normalized Manhattan distance values from the flip-flops (signals) to the trace buffer are computed, and then used with signal selection parameters like restorability (or visibility) to either eliminate or add a flip-flop into the set of flip-flops selected for the trace buffer. It is important to note that our algorithm can be used on top of any existing signal selection procedure.

In the remainder of this section, we describe two important components of our framework: distance calculation and layout-aware signal selection. First, we describe how to compute normalized Manhattan distance of flip-flops from the trace buffer. Next, we describe how to introduce layout awareness in two different signal selection techniques: metric- and simulation-based approaches.

Algorithm 1: Layout-aware Metric-based Selection

Inputs : normalized Manhattan distance $ndist[]$, design netlist, trace buffer width w
Output: selected signals

Load the Circuit; Let $finalRR$ be 0;
Let $\{T\}$: Set of all flip-flops in the design;
Let $\{SS\}$: Set of selected signals = ϕ;
Let rw be the routing weight of the signal under consideration;
for $rw \rightarrow 0$ **to** $1; rw+ = 0.1$ **do**
    Let $\{S\}$: Set of all selected flip-flops = ϕ;
    while *w greater than number of elements in* $\{S\}$ **do**
        Max Restoration Ratio: $maxRR = 0$;
        Let j: Index to the element to be added in the current iteration $= 0$;
        for $i \leftarrow 1$ **to** *number of elements in* $\{T\}$ **do**
            Let $\{V\} \leftarrow \{S\} \bigcup ith$ FF in {T} ;
            Let $rrv = srr(V)$;
            /* Evaluate selection */
            $rr = rw * (1 - ndist_i) + (1 - rw) * rrv$;
            if *rr* $>$ *maxRR* **then**
                $j = i; maxRR = rr$;
            end
        end
        {T} = {T} - jth *element*;
        {S} = {S} $\bigcup$ jth *element*;/* Add signal as selected */
    end
    if $finalRR < srr(S)$ **then**
        $SS \leftarrow S; finalRR \leftarrow srr(S)$;
    end
end
Return: {SS} is the set of selected signals

Algorithm 2: Layout-aware Simulation-based Selection

Inputs : normalized Manhattan distance $ndist[]$, trace buffer width w, simulation function to get state restoration ratio (SRR) $fsrr()$
Output: selected signals

Load the Circuit;
Let $\{T\}$: Set of all flip-flops in the design;
while *w< number of elements in* $\{T\}$ **do**
    Let j: Index to the element to be eliminated in this iteration $= 0$;
    /* Calculate simulation-based restoration ratio (SRR) of {T} */
    Let $\vartheta = fsrr(T)$;
    Let ν be the change in SRR$= 1000000$;
    for $i \leftarrow 1$ **to** *number of elements in* T **do**
        /* Create a new set by removing ith Signal. */
        Let $\{V\} \leftarrow \{T\} - \{i^{th}$ FF$\}$;
        /* Evaluate δSRR & scale with routing weight */
        Let $\nu_i = (1 - ndist_i) * (\vartheta - fsrr(V))$;
        /* Keep note of element with least reduction in overall SRR */
        if $\nu_i < \nu$ **then**
            $j = i$
        end
    end
    /* Remove element with least impact on SRR */
    {T} = {T} - jth *element*;
end
Return: {T}

A. 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. Euclidean 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. Therefore, wire-length estimation techniques such as half-perimeter wire-length, squared-Euclidean distance, minimum Steiner-tree wire-length, minimum spanning tree wire-length or complete-graph wire-length may be used [2]. We 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 (MD) is defined as the sum of the absolute values of differences in the X and Y coordinate values of any two points. $MD = (|x_{tb} - x_i| + |y_{tb} - y_i|)$.

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

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

We use normalized Manhattan distance as a layout-awareness metric for different signals. Manhattan distance (from the trace buffer) to all the prospective signals is calcu-

lated 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:

$$ndist_i = MD_i/MD_{max}$$

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

B. Metric-based approach

Algorithm 1 shows the key steps of applying our approach on top of metric-based signal selection. It 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 step size of 0.1.

The algorithm has two nested loops. Within the $while$ loop in step 7, a temporary set of signals (V) is created by adding i^{th} element of set $\{T\}$ to set of selected signals. This set is created to evaluate the effectiveness of the i^{th} signal. Each time $while$ loop in step 7 is invoked it generates a set of selected signals for the given routing weight (rw). The for loop in step 5 evaluates the sets of selected signals, for different values of rw, generated by the internal $while$ loop. Step 13 and Step 14 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 both restoration ratio and Manhattan distance perspective. Steps 5 and 13 are the the additional steps, over default metric-based signal selection, required for layout-aware signal selection.

C. Simulation-based approach

Simulation-based signal selection uses iterative elimination of less beneficial signals. In every iteration, simulation is used to determine the impact of eliminating a signal from the remaining signals. Signal with minimal impact on the restoration ratio is eliminated every cycle. Elimination continues until 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-flops, based on their normalized Manhattan distance from the trace buffer. Signal with the minimum impact on visibility, and minimum reduction in Manhattan distance is eliminated.

Algorithm 2 shows the key steps of applying our approach on top of simulation-based signal selection. Steps 1 and 2 in Algorithm 2 are initialization steps. Step 2 marks all the flip-flops as selected and puts them in set $\{\mathrm{T}\}$. The while loop in step 3 eliminates the elements in $\{\mathrm{T}\}$ until the number of elements remaining are equal to the width of trace buffer. Step 5 calculates the visibility based on the signals in $\{\mathrm{T}\}$. Step 7 iterates over the whole set in $\{\mathrm{T}\}$, and creates a new set $\{\mathrm{V}\}$ with just one element removed from $\{\mathrm{T}\}$. The visibility is calculated for each set $\{\mathrm{V}\}$ and scaled with the routing values. The signal to be eliminated is the one for which set $\{\{\mathrm{T}\}$ - "element"$\}$ has minimal impact on the visibility compared to set $\{\mathrm{T}\}$ alone. Set $\{\mathrm{T}\}$ is updated by eliminating the signal. At the end of the while loop in step 3, set $\{\mathrm{T}\}$ has the required selected signals, and is returned by the algorithm. Step 9 is the step for layout-awareness over the default simulation-based signal selection.

IV. Experiments

The section describes the experimental setup and presents the experimental results.

A. 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. We added a trace buffer of a given width and connected it to random internal nets of the design. Random internal nets were chosen to avoid biasing the layout for any signals. We used Cadence Encounter Digital Implementation tool (EDI) to synthesize and generate layout for the modified Verilog design. A design exchange format (DEF) file was dumped from the design synthesis tool. Manhattan distance for 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 the trace buffer. Final restoration ratio was then computed for the selected signals.

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

B. Comparison with Metric-based Approach

Table I compares our approach with existing metric-based signal selection [1]. The first column in the table specifies the benchmark used for evaluation. We 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 provide 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 we 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 I shows an average 32% (peak 55%) improvement in the Manhattan distance across

Table I
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

different benchmarks, with an average 14% penalty on the restoration ratio.

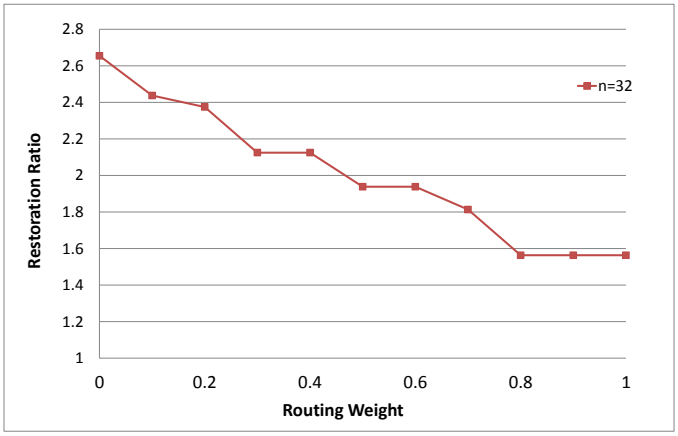
Two important scenarios can be observed in the results: i) Manhattan distance improves at the cost of restoration ratio, and ii) both Manhattan distance and restoration ratio improve. The first scenario is expected as the algorithm deliberately chooses signals near the trace buffer, although they may 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 provides detailed data points for s9234 benchmark. Figure 3 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.

Figure 3. Restoration Ratio for s9234



C. Comparison with Simulation-based Approach

Table II provides data for the analysis of benchmarks using simulation-based signal selection [3]. The columns used in Table II are same as that used in Table I. 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.

V. CONCLUSION

Post-silicon validation and debug are critical components of the SoC design methodology. It is important 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,

Table II
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

thereby selecting some signals which may not be routable. We developed techniques to incorporate layout-awareness in the existing approaches towards identification of signals which are not only beneficial from the debug perspective but also from a routing perspective. Our technique further gives designer freedom to customize the weight for layout-awareness to suit the design needs. Our 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 our approach can select layout-friendly signals with minor impact on restorability.

ACKNOWLEDGMENT

This work was partially supported by National Science Foundation (NSF) grant CCF-1218629.

REFERENCES

[1] K. Basu and P. Mishra. Rats: Restoration-aware trace signal selection for post-silicon validation. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 21(4):605–613, April 2013.

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

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

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

[5] H.F. Ko and N. Nicolici. Automated trace signals identification and state restoration for improving observability in post-silicon validation. In *Design, Automation and Test in Europe, 2008. DATE '08*, pages 1298–1303, March 2008.

[6] A. Kokrady, C. P. Ravikumar, and N. Chandrachoodan. Layout-aware and programmable memory bist synthesis for nanoscale system-on-chip designs. In *Asian Test Symposium, 2008. ATS '08. 17th*, pages 351–356, Nov 2008.

[7] J. Lee, S. Narayan, M. Kapralos, and M Tehranipoor. Layout-aware, ir-drop tolerant transition fault pattern generation. In *Design, Automation and Test in Europe, 2008. DATE '08*, pages 1172–1177, March 2008.

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

[9] Xiao Liu and Qiang Xu. Trace signal selection for visibility enhancement in post-silicon validation. In *Design, Automation Test in Europe Conference Exhibition, 2009. DATE '09.*, pages 1338–1343, April 2009.

[10] S. Prabhakar and M. Hsiao. Using non-trivial logic implications for trace buffer-based silicon debug. In *Asian Test Symposium, 2009. ATS '09.*, pages 131–136, Nov 2009.