Can't See the Forest for the Trees: State Restoration's Limitations in Post-silicon Trace Signal Selection

Sai Ma[1], Debjit Pal[1], Rui Jiang[1], Sandip Ray[2], Shobha Vasudevan[1]
[1]Coordinated Science Laboratory, University of Illinois at Urbana-Champaign.
{saima1, dpal2, rjiang3, shobhav} @illinois.edu
[2] Stategic CAD Labs, Intel Corporation. sandip.ray@intel.com

ABSTRACT

State Restoration Ratio (SRR) has been the de facto standard for evaluating quality of signals selected for post-silicon tracing and debug. Given a set $\mathcal{S}$ of selected signals, SRR measures the fraction of (gate-level) design states that can be inferred from observing signals in $\mathcal{S}$ at each cycle. Unfortunately, in spite of its widespread use, we found that SRR is intrinsically unsuitable as a metric for evaluating trace signal quality, as it captures neither the higher-level functionality of the design nor the constraints and requirements on trace signals imposed by architectural, physical, or security requirements. In this paper, we argue with strong empirical evidence that SRR must be replaced by a metric that closely models high-level behavioral coverage. We propose assertion coverage as a first step in this direction. We also present a new algorithm, based on Pagerank, for post-silicon trace selection. Pagerank is *not* designed to maximize SRR. We found that Pagerank has upto 70% higher behavioral coverage than SRR optimizing methods, and the RTL PageRank has upto 30% higher behavioral coverage than the netlist PageRank algorithm. Assertion coverage of PageRank RTL is upto 50% while SRR based methods have less than 5% assertion coverage.

1. INTRODUCTION

Post-silicon validation entails running tests and software on fabricated, pre-production silicon implementation of a hardware design to ensure that it operates correctly while running actual applications under on-field operating conditions. Since silicon executes at target clock speed, it runs several orders of magnitude faster than pre-silicon validation platforms. This permits exploration of deep design states that cannot be encountered in the pre-silicon environment. Post-silicon validation is a critically important [24], but also expensive activity, accounting for the majority of validation expense in modern SoC designs [28].

A fundamental problem of post-silicon validation is limited observability and control. Due to limitations in the number of output pins, and area and power overheads of internal trace buffers, only a few hundreds among the millions of internal signals can be traced or controlled during silicon execution. These signals must be selected *a priori* through analysis of pre-silicon design collaterals, so that the design can be instrumented with hardware to route them to an observation point (*e.g.*, pins, trace buffers, etc.). If critical signals are missed, their omission can be identified only during post-silicon, as an inability to root-cause an observed failure. At that point, rectifying the omission would require a significant change in the observability architecture together with silicon respin, which is not feasible.

This work was supported in part by National Science Foundation (NSF) under Grant 1-483534-239012-191100.

Given the severity of the impact of missing necessary observability, there has been significant research in the "signal selection problem", *i.e.*, disciplined identification of traceable signals that can maximize the design visibility as necessary for post-silicon debug, under observability restrictions [10–18, 21, 22, 26, 27]. While there are significant differences in the specific approaches proposed, virtually all related work uses the same metric, called *state restoration ratio* for evaluating their approaches. State restoration ratio (SRR) measures the number of design states reconstructed from the signals observed: a set S of signals is considered superior to another set S' if more design states can be inferred from observing S than S' (Section 2.3). Most signal selection algorithms include heuristics to efficiently identify signals that maximize SRRs.

In this paper we show that, in spite of its wide use as a de facto standard in signal selection research, SRR is in fact a poor metric for determining the quality of post-signal trace signals. This casts serious doubts on the practical applicability of all related signal selection algorithms that are based on optimizing SRR. Unsurprisingly, we found no study reporting on the usage of these methods on industrial design: all reported applications are on small benchmarks (*e.g.*, ISCAS89), that are not representative of the complexities of an industrial IC. Based on our observation, we strongly recommend that the post-silicon research community move away from SRR and consider alternative metrics for signal selection based on behavioral coverage.

Why is SRR a poor metric for qualifying post-silicon trace signals? Post-silicon validation targets exercising deep and interesting functional behavior of the design, *e.g.*, booting an oprating system, running target user-level applications, executing different power-management modes, etc. For each such behavior, certain specific design states are critical while others might be irrelevant. A metric for post-silicon trace signals should reward (or favor highly) the selection of signals that facilitate understanding, interpretation, and validation of such high-level functionalities during silicon execution. On the other hand, restorability as a metric takes a "myopic" view of state reconstruction, attempting to maximize restoration of gate-level states without prioritization. Consequently, an attempt to maximize SRR leads to patterns like (i) treating all signals equally, instead of using the natural design structure where some signals are more important than others; (ii) favoring big arrays in the design which are not very useful for debugging and (iii) not considering the context or conditions under which particular signals needs to be monitored.

The paper makes three important contributions.

Our first contribution is to show through empirical evidence and analysis that SRR is severely limiting as a general metric for post-silicon signal selection. We argue that a different metric is necessary that directly correlates with the extent of coverage of the execution flow of the design.

Our second contribution is a new metric, *assertion coverage*, as a step towards the definition of a metric that captures design specification. In particular, assertions correspond closely to invariants and coverage conditions inferred through trace signals during post-silicon debug. We define assertion coverage for an assertion $\mathcal{A}$ as the set of signals sufficient to evaluate $\mathcal{A}$. Note that in many cases, mapping an assertion to the part of the design it covers can be automatic [9]; furthermore, such mapping often exists as part of pre-silicon validation collateral. Nevertheless, assertions can have drawbacks arising from subjectivity and incompleteness, and we treat assertion coverage as only a first step towards a more comprehensive alternative metric for signal selection. We do not argue for assertion coverage as the absolute metric for post-silicon validation, but as a way to initiate a functional coverage based metric that departs from the low return SRR metric.

Our third contribution is a new signal selection algorithm that performs significantly better than algorithms designed to maximize SRR in achieving functional coverage. Our algorithm is adapted from Google PageRank algorithm [23]. It ranks some signals as more important than the others based on connectivity in the structural netlist or the RTL variable dependency graph. Higher ranked signals are better candidates for tracing. It also avoids inclusion of entire arrays, and selects relevant signals instead. Finally, it typically selects the signals and their operating conditions together due to their high co-occurrence and consequent similar ranking.

We applied the PageRank algorithm at the gate level as well as to the RTL design. At the gate level, we applied PageRank to the structural netlist. For RTL, we applied it to the variable dependency graph. The reason for applying PageRank in these two modes is to study the relative benefits, if any, of signal selection in an RTL data structure over gate level: applying the same algorithm at both levels would prevent the variability in analysis due to algorithmic differences. We compared our two PageRank based methods with respect to SRR and behavioral coverage metrics. Our analysis on an RTL model of an USB controller shows that on the average, overall behaviorial coverage achieved by signals selected by PageRank on netlist have higher coverage than the SRR maximization by up to 42% (average 19.6%). PageRank applied at RTL has a consistently higher behavioral than SRR maximization methods by up to 70% (average 33%) as compared to previous methods. The PageRank RTL method has up to 49% higher coverage than PageRank at netlist level when measuring assertion coverage. The SRR maximization methods generate signals with very low assertion coverage in our experiments. This is primarily due to these signals selecting only a subset of bits of several signals instead of the entire signal.

Interestingly, we found that none of the algorithms (including PageRank at netlist and RTL) are able to cover system level assertions, since they do not trace interface signals. This illustrates that optimizing for a functional coverage metric like assertion coverage will lead to interface signals being emphasized over internal signals. We do not, therefore, intend PageRank to be the final solution for signal selection based on functional coverage. Nevertheless, using it as a case study illustrates how moving away from optimizing SRR can yield techniques with better coverage on interesting observability requirements.

Our experiments are performed on a publicly available USB RTL design [6]. We select this design instead of the ISCAS89 benchmarks used in previous signal selection work for two reasons. First, the design is larger and more complex, reflecting design features of typical hardware IPs used in industrial SoC designs. This complexity helps illustrate the divergence between gate-level state restorability and functional behavior. We speculate that the flaws about SRR pointed to by this paper have not been observed by previous signal selection research because this difference is not pronounced in relatively smaller benchmarks. Second, we perform experiments on a PageRank based signal selection algorithm at both RTL and gate-level designs, to provide a relative comparison of signal quality. Since ISCAS89 and related benchmarks are at gate-level, it would be difficult to use them to analyze the value of an RTL signal selection over a gate-level approach.

2. PRELIMINARIES

2.1 PageRank Algorithm

Google's PageRank algorithm ranks a web page as important if it is hyperlinked from many important web pages. This ensures that all the hyperlinks don't have equal weights. PageRank computes an importance score of each web page based on its incoming hyperlinks. Let p denote a web page. Let *B(p)* denote the set of pages that have an outgoing link to p and let *F(p)* denote the set of pages that p has outgoing links to. Let ϵ be a constant between 0 and 1 and let n be the number of the web pages. The PageRank $PR(p)$ of p is defined as:

$$PR(p) = (1-\epsilon) \sum_{p_i \in B(p)} \frac{PR(p_i)}{|F(p_i)|} + \frac{\epsilon}{n} \quad (1)$$

The first term in the PageRank of Equation 1 represents the probability that a random surfer will navigate to a web page. If the surfer is caught in a cycle of web pages, then it is unlikely that the surfer will continue in the cycle forever. The second term accounts for the surfer eventually coming out of the cycle and navigate to a random webpage.

2.2 Variable Dependency Graph

We define the variable dependency graph based on the semantics of the Verilog Hardware Description Language [8]. An *expression* is a function defined over values, variables and operators. A *left reference* refers to a variable reference that appears in an expression on the left hand side of a Verilog statement. A *right reference* refers to all variable references that are not *left references*. Let v_i and v_j be two Verilog variables. We say that v_i depends on v_j if there exists a Verilog assignment to v_i that will execute only if a right reference to v_j is evaluated. Formally we define a *dependency graph* as a directed graph *G = (V, E)* with vertices V and directed edges E. Let each vertex $v \in V$ denotes a Verilog variable and let each directed edge (v_i, v_j) denote a dependence between variables v_i and v_j. If $(v_i, v_j) \in E$, then v_j depends on v_i.

2.3 Signal Reconstruction and SRR calculation

We show calculation of SRR using the simple circuit of the Figure 1. Let us assume that the trace buffer can record values of 2 signals. The restored values of the other signal states using the method of [10] are shown in the Table 1. The signals that are chosen using total restorability computations are A and C. The selected signals are shown in grey. The state restoration ratio (SRR) is given by the ratio of the number of signal values restored plus the number of signals traced to the number of signal values traced. Since 10 signal values are traced and 22 values are restored, the SRR with this selection is 3.2.

3. INADEQUACY OF SRR AS A METRIC

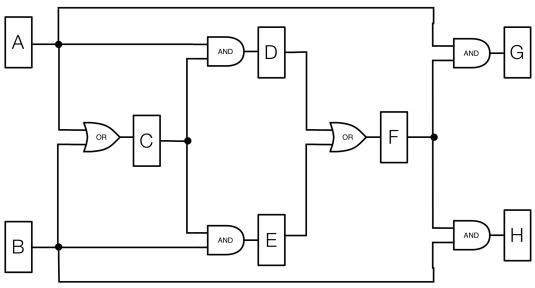

Figure 1: Example circuit

Table 1: State restoration applying [10]

Signal	Cyc1	Cyc2	Cyc3	Cyc4	Cyc5
A	0	0	0	0	1
B	1	0	1	0	X
C	1	1	0	1	0
D	X	0	0	0	0
E	X	1	0	0	0
F	X	X	1	0	0
G	X	0	0	0	0
H	X	X	0	1	0

3.1 A motivating example

In this Section we present a motivating example for the limitations of the SRR metric using an academic processor LC3b [2]. We applied a method that is designed to maximize SRR SigSeT [10] to the LC3b. It selects the complete ISDU FSM state registers, some bits of the PC and some bits of IR at the top of the list. With this set of signals, we can recreate a few control states, but not the rest of the processor state. Without the complete PC and IR, it is not possible to determine which instruction is being processed and fetched from memory next. As a point of reference, assertion coverage of the signals selected by SigSet was 0% for this design, implying that none of the assertions could be evaluated using the signals generated by SRR maximization.

As a point of contrast with the above results, consider the performance of our PageRank Algorithm that does not seek to maximize SRR (cf. Section 4) on the same example. PageRank selects all of the FSM state registers of ISDU module, all 16 bits of PC and IR as complete words and NZP branching registers. This is sufficient to check the sequence of states in the design, the opcode and operands fetched, all transitions in the control state machine and branching behavior. PageRank ranks all of the control signals with high priority while ranking eight 16-bit data registers lower. PageRank achieved an assertion coverage of 78.5% for this design.

3.2 Deconstructing SRR Inadequacies

The example above suggests a key problem with the utility of SRR as a metric: it treats all gate-level design states as "equals". Reconstructing any specific design state is not considered more valuable than reconstructing any other state. However, practical debugging experience suggests that some signals are inherently more valuable for validation and debug than others. Also, some signals can only provide useful state information in the presence of some other signals as well. For example, reconstructing only the lower-order bit of a program counter (PC) provides little information on program behavior or execution flow while reconstructing all bits of the PC can provide significant insight. Consequently, signals selected to optimize SRR do not necessarily facilitate debug. In particular, SRR is not useful for signal selection for designs with the following features.

Large Arrays. In such designs, individual array elements are typically less valuable for debug than control signals that affect reads and writes to the arrays. Methods optimizing SRR, on the other hand, would tend to reconstruct individual array values.

On-chip Instrumentations. Modern IC designs include a significant amount of on-chip hardware instrumentations not contributing to functionality, including Design-for-Test (DFT) features, instrumentations for security, and indeed, hardware to enable post-silicon debug and control. Since SRR is agnostic to design intent, selection based on SRR typically includes a sampling of signals for different functionality as well as different instrumentation features. The result is that the traced signals are inadequate for functional debug while not providing sufficient design visibility for validating the instrumentations as well.

Complex Protocols. Most multi-core systems and SoC designs include design blocks (referred to as "Intellectual Property" or "IP") that coordinate through complex protocols. One of the critical applications of hardware trace is to validate these protocol implementations during post-silicon debug. This implies that the traced signals include the messages communicated across the IPs during system execution. However, SRR does not account for the relative importance of these signals. Indeed, algorithms optimizing SRR would tend to favor signals in larger IPs with more design states while missing smaller IPs; thus, routers in communication fabrics through which protocol messages are communicated would be typically ignored. fla

3.3 Asssertion coverage as a trace signal selection metric

In this section, we propose a new metric, *assertion coverage*, for evaluation of post-silicon trace signals. Informally, assertion coverage captures the notion that a key usage of signal traces in post-silicon debug is to check if the system satisfies expected invariants. A set of signals considered valuable for post-silicon debug must permit the validator to detect such properties.

Conventions. For simplicity of formalization, we assume that the value of each signal s ranges over a Boolean; this can be easily extended over other datatypes. For this paper, an assertion is a *bounded temporal* expression over design signals. A bounded temporal expression [25] over a set of variables is a temporal expression includes Boolean operators and next-time (`X`) operator. Given an assertion $\mathcal{A}$ and set $\mathcal{S}$ of signal values over a sequence of clock cycles, we use standard temporal logic semantics to define the value of $\mathcal{A}$ at a cycle t. Note that because of temporal operators, the value of an assertion $\mathcal{A}$ at cycle t may require values of signals at cycles other than t.

Definition 1 (Assertion Coverage). *Let $\mathcal{S}$ be a set of signals and $\mathcal{A}$ be an assertion. We say $\mathcal{S}$* covers *$\mathcal{A}$ if all the signals mentioned in $\mathcal{A}$ are members of $\mathcal{S}$. Let $\kappa = \{\mathcal{A}_0, \mathcal{A}_1, \ldots, \mathcal{A}_k\}$ be a set of assertions. Then the coverage of κ with $\mathcal{S}$ is the proportion of assertions in κ covered by $\mathcal{S}$.*

To illustrate assertion coverage, consider the following two System-Verilog assertions from LC3B design

A1: *(cpu/isdu/state_reg[4:0] == 5'd18) → ##1 (cpu/PC/Dout_reg[15:0] == $past(cpu/PC/Dout_reg[15:0]) + 16'd1)*

A2: *(cpu/isdu/state_reg[4:0] == 5'd18) → ##1 (cpu/isdu/state_reg[4:0] == 5'd33).*

If *cpu/isdu/state_reg[4:0]* is selected as trace signal, then **A2** is covered, but **A1** is not covered (since the signal *Dout_reg[15:0]* in the consequent of **A1** is not selected).

The use of assertion coverage as an optimization target connects trace selection requirements to some of the properties of interest to silicon validators. Nevertheless, we do not propose assertion coverage as a final end-all metric for trace selection. For instance, it does not account for some of the other usages of signal tracing, *e.g.*, protocol interactions, design behavior at specific corner cases, etc. Furthermore, note that our definition only measures the assertions covered through the set of traced signals $\mathcal{S}$, without accounting for coverage from signal values that may be reconstructed from $\mathcal{S}$. This makes sense for checking design invariants, since checking such conditions require observation of covered signals at each cycle. Recall that signal restoration can only restore values at some of the traced cycles (cf. Table 1). However, for more general coverage condition we may need to account for restored signal values as well. Finally, The quality of the selected signals based on optimizing assertion coverage depends on the quality of the set of assertions used. Traditionally, one would need to manually develop them during design verification. Of course, techniques for automated assertion generation as well as techniques for (independently) evaluating assertions [1, 4, 9] can assist in developing high-quality assertions. Notwithstanding these limitations, assertion coverage points towards a promising direction of developing metrics that target functional behavior rather than blindly reconstructing design states.

4. PAGERANK BASED TRACE SIGNAL SELECTION ALGORITHM

4.1 Pagerank for Netlist

Algorithm 1 [5] is the PageRank algorithm for netlist level signal selection. We apply it on the Example circuit of Figure 1 in this section.

4.1.1 Network Construction

We parse the synthesized netlist of an RTL design to construct a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ representing the connectivity between different logic elements, where every $v \in \mathcal{V}$ represents a logic element and every directed edge $(v_i, v_j) \in \mathcal{E}$ represents a connection between the logic element v_i and v_j. Figure 2 shows the directed graph for the example circuit in Figure 1.

4.1.2 PageRank Value Calculation

After constructing the directed graph for the circuit, we apply PageRank to compute the importance of each node. The directed graph in Figure 2 has 14 logic elements (8 sequential elements and 6 logic gates). Each node transfers its importance equally to the nodes that it links to. For example, node A has 3 out-links, so it will transfer 1/3 of its importance to each of the node OR1, AND1, and AND3. In general, if a node has n out-links, it will pass on $1/n$ of its importance to each of the node that it is linked. Following this importance transition rule, we annotate every edge of the Figure 2 with the corresponding importance value.

Initially we assume an equal rank for each of the nodes i.e. if there are n nodes in the network, every node will have a rank of $\frac{1}{n}$. In Figure 2 each node has a rank of 1/14. As each incoming link increases the rank of a node, we update the rank of each node by adding the importance of the incoming links. We continue this until the rank of all of nodes stabilizes. We use a standard error tolerance value in PageRank algorithm, which is 1e-6, to check convergence in the power iteration process. If the rank across two iterations is within this error tolerance, the rank of that node is assumed to be stabilized and returned. In the example network, node G and H do not have any outgoing links and PageRank refers them to as dangling nodes.

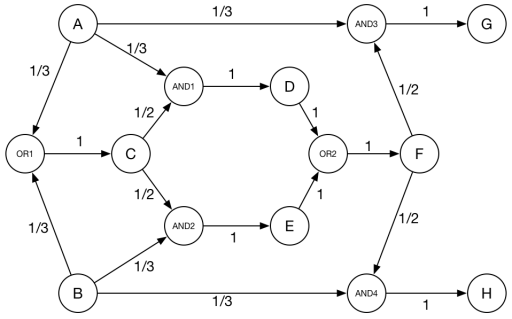

Figure 2: Circuit network of Figure 1 annotated with importance contribution

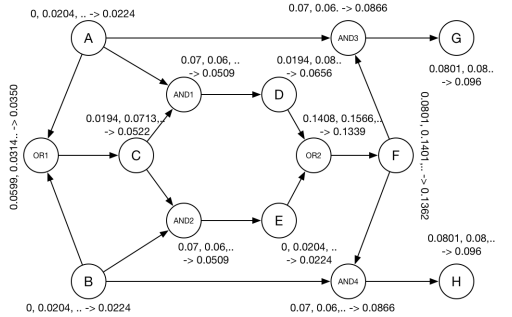

Figure 3: Circuit nodes annotated with importance values in successive iteration and the final importance value

Dangling nodes would cause the final rank of each node to converge to 0 and the importance of these nodes cannot be propagated further. Since dangling nodes and disconnected components are quite common in the Internet as well as in common circuits, a positive constant between 0 and 1 (typically 0.15) is introduced, which is the damping factor ϵ [23]. We add a virtual directed edge from G and H to every other node in the network and assign ϵ to every outgoing edge from G and H.

After the dangling nodes adjustment, we recalculate the rank of each of the node in the graph until the PageRank value stabilizes. For our example, the final value of the PageRank of each node is shown in Figure 3. According to this PageRank value, we will select flip-flop F and G or H as the trace buffer signals, since the trace buffer width is 2.

Let $0 < \epsilon < 1$ be a constant source of importance. Let $\mathbf{r}$ denote an importance score vector over variables and let $\mathbf{r}^k$ denote $\mathbf{r}$ in the k-th iteration of the importance computation. Let $\mathbf{r}_i^0 = \frac{1}{n}$. We compute the importance score of each of the variable as follows:

$$\mathbf{r}^{k+1} = (1 - \epsilon)\mathbf{A}\mathbf{r}^k + \frac{\epsilon}{n} \tag{2}$$

4.2 PageRank for RTL

To compute the importance of each variable in an RTL design, we adapt the idea of PageRank algorithm [23] using the idea of *variable dependency graph* described in Section 2.2.

PageRank computation requires a dependency graph. We present a variable dependency graph using an adjacency matrix. Let a_{ij} denote the number of right references to variable i in all assignments to variable j. Let a_i denote the number of right references to variable i in all assignments. Let $\mathbf{A}$ be an $n \times n$ square matrix with rowa and columns corresponding to variables. Let $\mathbf{A}_{ij} = \frac{a_{ij}}{a_i}$ if $a_i > 0$ and let $\mathbf{A}_{ij} = \frac{1}{n}$ otherwise. Intuitively, $\mathbf{A}_{ij}$ is equal to the fraction of right references to variable i that exist in all assignments

Algorithm 1 pseudo-code of PageRank algorithm

procedure PageRank($\mathcal{G}$, iteration)
$\epsilon = 0.15$ {outlink count hash from $\mathcal{G}$}
oh $\leftarrow \mathcal{G}$ {inlink count hash from $\mathcal{G}$}
ih $\leftarrow \mathcal{G}$ {number of nodes from $\mathcal{G}$}
n $\leftarrow \mathcal{G}$ {initialize PageRank}
for p in $\mathcal{G}$ **do**
  opg[p] $\leftarrow \frac{1}{n}$
end for
while *iteration* > 0 **do**
  dp $\leftarrow 0$ {PageRank values from nodes with out-links}
  for p has no out-links **do**
    dp $\leftarrow$ dp+ $(1-\epsilon)\times \frac{opg[p]}{n}$
  end for
  for p in $\mathcal{G}$ **do**
    npg[p] $\leftarrow$ dp+ $\frac{\epsilon}{n}$ {PageRank values from random jumps}
    for p in $\mathcal{G}$ **do**
      npg[p] $\leftarrow$ npg[p]+ $\frac{(1-\epsilon)*opg[ip]}{oh[ip]}$ {PageRank values from in-links}
    end for
  end for
  opg $\leftarrow$ npg
  iteration=iteration-1
end while
end procedure

```
1   module arb2(clk, rst, req1, req2, gnt1, gnt2);
2   input clk, rst, req1, re2;
3   output gnt1, gnt2;
4   reg gnt_, gnt1, gnt2;
5   always @(posedge clk or posedge rst)
6     if(rst)
7       gnt_ <= 0;
8     else
9       gnt_ <= gnt1;
10  always @(*)
11    if (gnt_)
12    begin
13      gnt1 = req1 & ~req2;
14      gnt2 = req2;
15    end
16    else
17    begin
18      gnt1 = req1;
19      gnt2 = req2 & ~req1;
20    end
21  endmodule
```

Figure 4: Verilog Code of a 2-port Arbiter

to variable j. If no references to variable i exist in the design, then we assume a right reference to variable i exists in an assignment to each other variable. Hence $\mathbf{A}_{ij} = \frac{1}{n}$ when $a_i = 0$.

The global importance computation iteratively computes the importance score of each variable in the design until the score is stabilized. We have found through experimentation that when $\epsilon = 0.5$, the global importance score distribution of the variable agrees with the designer intuition. The equation for computing the rank of variables in the variable dependency graph is the same as Equation 2.

Figure 5 shows the variable dependency graph of the arbiter of Figure 4. Each node in the graph is labeled with its respective variable and the PageRank score. Edge weights denote the number of dependencies between the variables. For example, since `gnt1` depends on `req1` in both lines 13 and 18 of the Verilog, the weight of the edge (`req1, gnt1`) is equal to 2. Any edge without a specified weight has a a weight equal to 1. From the final ranks after convergence, we find that gnt_, which is the arbitration signal is ranked highest, after which gnt1 and gnt2, the two signals receiving

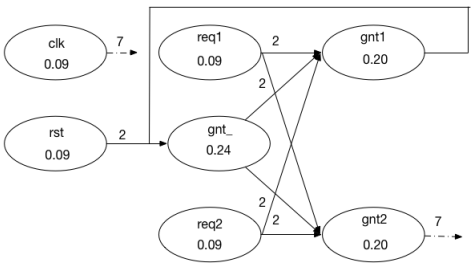

Figure 5: The variable dependency graph for the 2 port arbiter. Each node annotated with its final PageRank importance score

the grant are ranked. Other signals are equally (less) important. We select the top 20% of the signals rank sorted by the PageRank algorithm.

5. EXPERIMENTAL RESULTS

In our experiments we compare PageRank on RTL, PageRank on Netlist, SigSeT and HybrSel[1] We primarily use the publicly available USB 2.0 [6] design to demonstrate our result. Since SigSeT and HybrSel accept deisgns in ISCAS89 format, we use the three biggest benchmarks of ISCAS89 namely s35932, s38417, s38584 and the LC3B for comparison with these algorithms. We convert LC3B and USB into ISCAS89 netlist format for comparison with these algorithms. We synthesize LC3B and USB using Synopsys Design Compiler with NanGate 45 nm library [3] and constrain the library such that the synthesized DC netlist contains only basic logic gates like AND, OR, NAND etc. and DFF. We then convert the DC netlist into the ISCAS89 format. However, we could not obtain results on the USB using HybrSel as we ran into implementation issues.

All experiments were run on an AMD Opteron 8 cores 22xx processor with 15GB of RAM. In most of our experiments, we use simulation based coverage metrics for behavioral coverage. This includes line coverage, FSM coverage, condition coverage and branch coverage. We also compare the algorithms with respect to our assertion coverage metric.

5.1 Comparative analysis of algorithms with respect to SRR

In this experiment, we compare the SRR values of the signals selected by four different algorithms. Since all the ISCAS benchmarks are in netlist format, we could not run PageRank on RTL to select signals from them. We could not run HybrSel on the USB design. To calculate SRR values, we used the top 20% of the signals selected by each method for each benchmark and restored signals for 5000 cycles. Table 2 shows a comparative analysis of the runtime of the different tool during signal selection phase. To record the maximum memory usage, we used the Massif tool in Valgrind [7]. Table 3 shows the different SRR values calculated on different benchmarks.

We note a considerable difference in the runtime between HybrSel and PageRank on netlist while selecting signals from ISCAS benchmark. In PageRank method, we iterate until the values of the ranking matrix is stabilized. In case of HybrSel it iteratively updates the restorability rate of each state element based on the current signal selection which is computationally intensive and takes more time.

On the ISCAS89 benchmarks, none of the methods perform consistently better. Since PageRank on netlist is not aimed to optimize SRR, its value is not higher than SigSeT

[1]We were only able to gain access to SigSeT and HybrSel from prior art (SRR maximizing algorithms). The others were not accessible for research purposes.

Table 2: Runtime (in seconds) and maximum memory (in MB) usage of SigSeT, HybrSel, PageRank on Netlist and PageRank on RTL during signal selection phase

Bench	SiG-Set		Hybr-Sel		PR on Netlist		PR on RTL	
	T	Mem	T	Mem	T	Mem	T	Mem
s35932	9.52	498	17.6K	389	7.74	275.8	-	-
s38417	208	702	19.4K	359	9.54	326.9	-	-
s38584	150	285	8.94K	287	7.86	298.81	-	-
USB2.0	181	385	-	-	-	-	624	166.4

Table 3: Comparative Analysis of Restoration Ratio using SigSeT [10], HybrSel [19] and Algorithm 1

Benchmark	SiGSeT	HybrSel	PR on Netlist	PR on bf RTL
s35932	4.7	4.7	4.7	-
s38417	4.0	3.8	3.9	-
s38584	4.7	4.6	4.7	-
USB2.0	3.7	-	3.5	3.8

and HybrSel. Interestingly, although PageRank on RTL is not designed to optimize SRR, it produces the highest SRR value on the USB design, whereas PageRank on netlist produces least SRR value.

We note that the SRRs of the ISCAS benchmarks in Table 3 are significantly less than the values reported in previous papers [10, 18, 20]. SRR is a ratio and defined as (total number of signals restored + total number of signals traced) / (total number of signals traced). Previous work [10, 18, 20] used a fixed length trace buffer of size 8/16/32 and therefore the denominator is 8/16/32. If the average number of signals restored is 1000, the RR value will be 126/63/32. We select approx 350 signals in each design, making our denominator very large. So even with 1200 signals restored, SRR value is small.

5.2 Comparing behavioral coverage of signals selected

5.2.1 *Behavioral coverage of LC3B*

In this experiment, we compare trace signals of LC3B selected by PageRank and assess each one's simulation metric based behavioral coverage. We use the values of the trace signals from both the methods and simulate the design for 250ns. The behavioral coverage values for LC3B are shown in Table 4. Signals selected by PageRank achieve upto 30% more behavioral coverage than the signals selected by SigSeT.

5.2.2 *Behavioral coverage of USB*

In the this experiment, we compare the behavioral coverage achieved with the trace signals selected by SigSeT, PageRank on NetList and PageRank on RTL on the USB design.

We trace values of 355 flip-flops for a simulation duration of 175ms. Such a long trace was needed because atleast 100ms is needed to activate different important states (such as high speed state mode of USB) of the USB line control module. We use the traced value of the selected signals along with 5 important input control signals as the stimulus in RTL and measure the behavioral coverage using Synopsys VCS. The behavioral coverage consists of four components namely *branch coverage*, *line coverage*, *condition coverage* and *FSM coverage*. Table 5 shows the behavioral coverage values reported by VCS. In each of the methods, we do not report FSM coverage for `u4 (usbf_rf)` since it does not contain any state machines. Also, we do not report FSM and conditional coverage for `u2 (usbf_mem_arb)` as it is a combinational design module.

Table 4: Comparative analysis of behavioral coverage of the trace signals selected by applying PageRank at Netlist level and SigSeT
L: Line Coverage, C: Condition Coverage, F: FSM Coverage, B: Branch Coverage

Module	PR on Netlist				SigSeT			
	L	C	F	B	L	C	F	B
_CPU	86.68	-	-	69.41	56.68	-	-	39.98

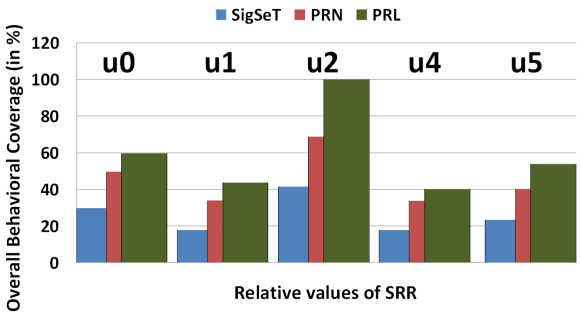

Figure 6: Graphs showing lack of correlation between SRR and overall behavioral coverage on different USB modules u_0, ... u_5. Width of each bar $\propto$ SRR. **PRN**: PageRank Netlist, **PRL**: PageRank RTL

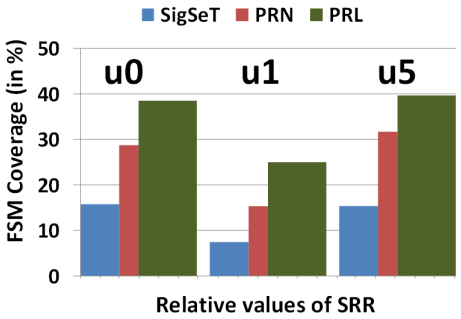

Figure 7: Graphs showing lack of correlation between SRR and FSM coverage on different USB modules u_0, u_2, u_5. Width of each bar $\propto$ SRR. **PRN**: PageRank Netlist, **PRL**: PageRank RTL

The behavioral coverage of signals selected by PageRank on netlist is upto 42% (average of 19.6%) more than the signals selected by SigSeT. The signals from PageRank on RTL achieves behavioral coverage upto 70% (average of 30%) more than the signals from SigSeT. The result of this experiment shows that compared to SigSeT, PageRank on Netlist and PageRank on RTL select functionally relevant signals from the USB design.

In Figure 6, 7, 8 and 9 we analyze the correlation between SRR and the different components of behavioral coverage. In this plot, the width of each bar is proportional to the SRR value shown in Table 3 and the height of the each bar is equal to the coverage achieved. Clearly, there is no correlation between the SRR value and the coverage. Although SRR value of PageRank on Netlist is less than that of SigSeT, the behavioral coverage of PageRank on Netlist is higher than SigSeT. This underscores the point that high SRR has low to no correlation with relevance to functional behavior.

5.3 Comparison of assertion coverage

In this experiment we compare the signals of SigSeT, PageRank on Netlist and PageRank on RTL with respect to assertion coverage using USB design.

We use an automatic assertion generation tool to generate module level assertions from the different modules of USB RTL designs. We use the default setting of the assertion generator tool to generate module level assertions. Table 6 shows the assertion coverage achieved by the signals from different methods on different modules. Based on the defi-

Table 5: Comparative analysis of trace signals from PageRank in RTL and Netlist Level with respect to simulation based coverage metrics u0: usbf_utmi_if, u1: usbf_pl, u2: usbf_mem_arb, u4: usbf_rf, u5: usbf_wb L: Line Coverage, C: Condition Coverage, F: FSM Coverage, B: Branch Coverage, O: Overall Coverage $\frac{L+C+F+B}{4}$

Module	SigSeT					PR on Netlist					PR on RTL				
	O	L	C	F	B	O	L	C	F	B	O	L	C	F	B
u0	29.73	35.81	28.89	15.75	38.35	49.61	71.81	38.89	28.75	58.97	59.67	76.92	57.78	38.5	65.48
u1	17.71	22.94	15.02	7.44	25.43	33.94	52.94	22.02	15.33	45.46	43.67	58.91	35.41	24.98	52.97
u2	41.44	33.45	-	-	49.43	68.75	75.00	-	-	62.50	100.0	100.0	-	-	100.0
u4	17.71	30.98	1.67	-	20.48	33.70	60.98	2.45	-	37.68	40.12	69.34	4.91	-	46.12
u5	23.27	31.90	13.49	15.37	32.33	40.14	56.34	19.23	31.67	53.33	53.81	82.35	23.23	39.67	70.00

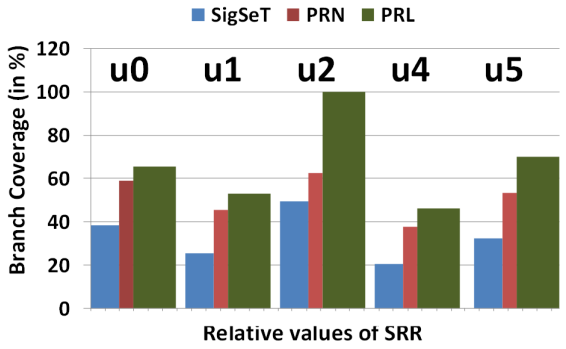

Figure 8: Graphs showing lack of correlation between SRR and branch coverage on different USB modules u_0, ... u_5. Width of each bar $\propto$ SRR. **PRN**: PageRank Netlist, **PRL**: PageRank RTL

Table 6: Comparison of signals from SigSeT, PageRank on Netlist and PageRank on RTL with respect to assertion coverage. The 0% coverage is for system level assertions.

Module	No. of Assertions	SigSeT	PR on Netlist	PR on RTL
usbf_pe	57	1.75%	50.87%	26.31%
usbf_rf	65	0%	7.69%	12.30%
usbf_pd	1402	0%	0%	0%
usbf_pa	44	0%	0%	0%
usbf_idma	24	4.1%	8.3%	12.5%
usbf_utmi_ls	55	0%	0%	0%

nition of assertion coverage given in Section 3.3, we say an assertion in the module level is covered if all the signals in its antecedent and consequent are present in the set of trace signals selected from the whole USB design

In this experiment, SigSeT selects partial bits of bus, whereas PageRank on RTL and PageRank on netlist selects complete bits of the bus signals. PageRank on RTL and netlist have a higher assertion coverage upto 49% on usbf_pe and usbf_rf than SigSeT. Table 6 reveals that overall, none of the methods perform very well in terms of assertion coverage. This is because the selected trace signals are only 20% of the total signals available in USB design and are spread across different modules. Hence, either the signals of antecedent or consequent or both are not present in the selected trace buffer signals resulting in low assertion coverage. If functional coverage metrics like assertion coverage are researched further, the signal selection can optimize this metric to avoid such a scattering across modules.

To find coverage of the system level behavior of USB design, we wrote 10 system level (inter modular) assertions. Surprisingly, the trace buffer signals of any of the methods do not cover any of the system level assertions. Careful examination of the selected trace signals shows that 95% of the selected signals are internal module level signals and the remaining 5% are interface signals. These interface signals did not cover any of the 10 system level assertions that we wrote. Future research in post-silicon trace signal selection

Table 7: High level functionality covered by PageRank and SigSet selected signals on USB Netlist. **P:** Partial bit selected

Signal Name	Module Name	Signal Functionality	Sig SeT	PR Net
no_bufs0	usbf_pe	**A.** Indicates available buffer size is less than payload size to switch to other buffer, **B.** BUF0 is full in DMA mode (Only BUF0 is used in DMA mode), **C.** Indicates if the BUF1 needs to be selected for next operation by the functional controller	✗	✓
token_pid_sel	usbf_pe	Handshaking signals indicating the packet accepting capacity of the buffer	✗	✓
dma_out_buf_avail	usbf_ep_rf	Indicates that there is a space for at least one MAX_PL_SZ packet in the buffer	✗	✓
inta, intb	usbf_rf	A fully programmable interrupt to provide full flexibility to software, the interrupts may be endpoint dependent or independent, indicating an error condition or overall events that have global meaning	✗	✓
state	usbf_pe	Indicates the states of operation of the USB protocol engine	✓	✓
state	usbf_utmi_ls	Indicates the states of operation of the USB protocol engine	P	✓
abort	usbf_pe	Indicates to abort an ongoing data transfer if the following conditions happen **A.** Buffer overflows (Received data packet size is too big and Rx_Data_Valid is asserted), **B.** Register end points matched and protocol engine is not in IDLE mode, **C.** Received packet size is more than MAX_PL_SZ	✗	✓
chirp_count	usbf_utmi_ls	A counter to initiate USB high speed mode	✗	✓
pid_seq_err	usbf_pe	An interrupt notifying USB function controller a loss of sync due to bad packets resulting in CRCs	✗	✓

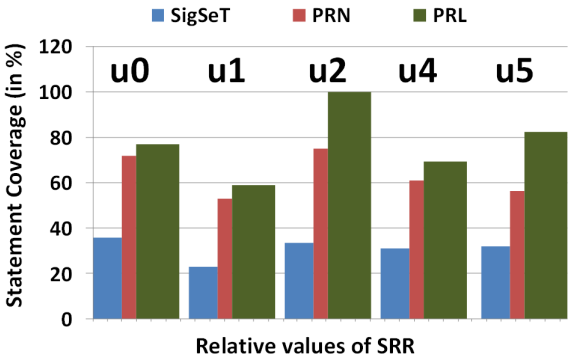

Figure 9: Graphs showing lack of correlation between SRR and statement coverage on different USB modules u_0, ... u_5. Width of each bar $\propto$ SRR. **PRN**: PageRank Netlist, **PRL**: PageRank RTL

should try to select more interface signals so that the system level assertions can be verified during debugging.

5.4 High level functionality selected by PageRank on USB netlist

To give a flavor of the type of high level functionality captured by the signals selected, we provide a qualitative analysis of two gate level algorithms, PageRank for netlist and the SRR optimizing SigSet. Table 7 for each signal, we list the corresponding RTL module and its high level functionality. PageRank selects all the FSM state registers of the USB protocol engine (`usbf_pe`) and USB line state module (`usbf_utmi_ls`) and other important signals. SigSet, the SRR maximization algorithm selects one signal completely, and the other partially.

6. RELATED WORK AND CONCLUSION

Automatic selection of trace buffer signals to maximize state restoration ratio (SRR) has been studied and many effective methods are proposed. Some techniques advocate signal selection based on partial restoration [15, 16, 21, 22]. Since partial restoration techniques are insufficient for signal reconstruction, Basu *et al.* [10–12] proposed signal selection based on total restorability. In this method, the group of signals selected can completely restore a certain amount of untraced signals; that is, it is a special case of partial SRR with SRR value of 100%. Chatterjee *et al.* [13] proposed Simulation based signal restoration, which departs from the probabilistic analysis of signal behavior, but has deterministic simulations instead. Hybrid signal selection techniques have also been developed [18], combining the probability based methods and the simulation based signal restoration method. Machine learning techniques have also been studied for signal selection [27].

In conclusion, we show that state restoration ratio as a metric does not accomplish the behavioral coverage of the design that is desirable in practical post-silicon debug. Trace signal selection methods that are independent of SRR maximization, like the PageRank, have higher behavioral coverage. We also conclude that RTL signal selection methods are superior to gate level signal selection methods in terms of behavioral coverage. Going ahead, we need to invent algorithms and methods that can optimize for relevant behavioral coverage.

7. REFERENCES

[1] Jasper. `http://www.jasper-da.com/products/jaspergold_apps`.

[2] LC3B Processor. `https://courses.engr.illinois.edu/ece411/mp/LC3b_ISA.pdf`.

[3] Nangate FreePDK. `http://www.nangate.com/?page_id=2325`.

[4] NextOp. `http://www.nextopsoftware.com/te_technicalpapers.html`.

[5] PageRank Algorithm. `http://www.ccs.northeastern.edu/home/daikeshi/notes/PageRank.pdf`.

[6] USB 2.0. `http://opencores.org/project,usb`.

[7] Valgrind Massif Tool. `http://valgrind.org/docs/manual/ms-manual.html`.

[8] Verilog Hardware Description Language. `http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1620780`.

[9] V. Athavale, S. Ma, S. Hertz, and S. Vasudevan. Code coverage of assertions using RTL source code analysis. In *The 51st Annual Design Automation Conference 2014, DAC '14, San Francisco, CA, USA, June 1-5, 2014*, pages 1–6, 2014.

[10] 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*, pages 352–357. IEEE, 2011.

[11] 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, 2013.

[12] K. Basu, P. Mishra, and P. Patra. Constrained signal selection for post-silicon validation. In *2012 IEEE International High Level Design Validation and Test Workshop (HLDVT)*, pages 71–75. IEEE, 2012.

[13] 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. IEEE, 2011.

[14] K. Han, J.-S. Yang, and J. A. Abraham. Enhanced algorithm of combining trace and scan signals in post-silicon validation. In *VLSI Test Symposium (VTS), 2013 IEEE 31st*, pages 1–6. IEEE, 2013.

[15] 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*, 28(2):285–297, 2009.

[16] H. F. Ko and N. Nicolici. Automated trace signals selection using the rtl descriptions. In *Test Conference (ITC), 2010 IEEE International*, pages 1–10. IEEE, 2010.

[17] H. F. Ko and N. Nicolici. Combining scan and trace buffers for enhancing real-time observability in post-silicon debugging. In *European Test Symposium*, pages 62–67, 2010.

[18] M. Li and A. Davoodi. A hybrid approach for fast and accurate trace signal selection for post-silicon debug. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 485–490. EDA Consortium, 2013.

[19] M. Li and A. Davoodi. A hybrid approach for fast and accurate trace signal selection for post-silicon debug. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 33(7):1081–1094, 2014.

[20] M. Li and A. Davoodi. Multi-mode trace signal selection for post-silicon debug. In *ASP-DAC*, pages 640–645, 2014.

[21] X. Liu and Q. Xu. Trace signal selection for visibility enhancement in post-silicon validation. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 1338–1343. European Design and Automation Association, 2009.

[22] X. Liu and Q. Xu. On signal selection for visibility enhancement in trace-based post-silicon validation. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 31(8):1263–1274, 2012.

[23] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford University, 1998.

[24] P. Patra. On the cusp of a validation wall. *Design & Test of Computers, IEEE*, 24(2):193–196, 2007.

[25] A. Pnueli. The Temporal Logic of Programs. In *18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA*, pages 46–57. IEEE Computer Society, , 31 October - 1 November 1977.

[26] K. Rahmani and P. Mishra. Efficient signal selection using fine-grained combination of scan and trace buffers. In *VLSI Design and 2013 12th International Conference on Embedded Systems (VLSID), 2013 26th International Conference on*, pages 308–313. IEEE, 2013.

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

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