Efficient Selection of Trace and Scan Signals for Post-Silicon Debug

Kamran Rahmani, *Student Member, IEEE*, Sudhi Proch, and Prabhat Mishra, *Senior Member, IEEE*

***Abstract*— Post-silicon validation is a critical part of integrated circuit design methodology. The primary objective is to detect and eliminate the bugs that have escaped pre-silicon validation phase. One of the key challenges in post-silicon validation is the limited observability of internal signals in manufactured chips. A promising direction to improve observability is to combine trace and scan signals—a small set of trace signals are stored in every cycle, whereas a large set of scan signals are dumped across multiple cycles. Existing techniques are not very effective, since they explore a coarse-grained combination of trace and scan signals. In this paper, we propose a fine-grained architecture that addresses this issue using various scan chains with different dumping periods. We also propose efficient algorithms to select beneficial signals based on this architecture. Our experimental results demonstrate that our approach can improve restoration ratio up to 127% (36% on average) compared with existing trace-only techniques. Our approach also shows up to 125% improvement (61.7% on average) compared with techniques that allow a combination of trace and scan signals with minor (<1%) area and power overhead.**

***Index Terms*— Debug, post-silicon, scan, trace.**

I. Introduction

THE goal of post-silicon validation of an integrated circuit is to ensure that the fabricated, preproduction silicon operates correctly under actual operating conditions with real application. It is a complex activity performed under aggressive schedules, representing $>50\%$ of the overall validation cost [1]. A fundamental challenge in post-silicon validation is limited observability and controllability. Due to the limitations in the number of output pins and area and power overheads of internal trace buffer, only a small percentage of internal signals in the design can be observed during silicon execution. Furthermore, in order for a signal to be observed, the design must be instrumented *a priori* with appropriate control hardware that routes a signal to an observation point. It is, therefore, crucial to identify trace signals that maximize design visibility and debug information under the observability constraints.

Fig. 1 provides an overview of post-silicon validation and debug process. Signal selection and trace buffer design are done in pre-silicon phase. If an error occurs during post-silicon validation phase, the traced values of internal signals are dumped. During the debug process, both dumped signals and restored signals are used to locate the error. The number of signals that can be traced per cycle is limited to trace buffer width. In addition, the maximum number of values recorded per signal is limited to the trace buffer depth. Therefore, the primary objective is to select a small set of profitable signals that can maximize restoration performance. A major challenge in efficient signal selection is that the exploration space (number of potential alternatives) can be prohibitively large even for small circuits. For example, s35932 circuit of ISCAS'89 benchmarks suite has 1728 flip-flops. If the trace buffer width is 32, we need to choose 32 signals out of the total 1728 flip-flops. It is easy to observe that there are $>10^{69}$ such combinations. This makes exhaustive exploration infeasible.[1]

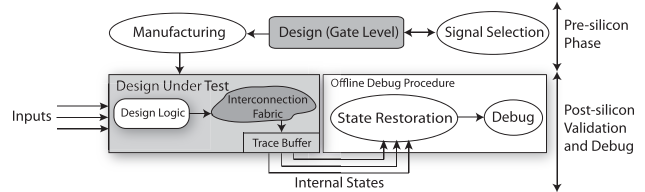

Fig. 1. Simplified overview of post-silicon validation flow and role of trace signal selection. Signals selected through pre-silicon analysis are funneled to trace buffer from which silicon states are restored offline to assist in debug.

Several trace signal selection techniques were proposed over the years [2]–[8]. These algorithms attempt to select a set of promising trace signals to maximize the number of states that can be restored. To improve the observability further, various approaches [9], [10] explored a profitable combination of trace and scan signals. The idea is to divide the trace buffer (width) into two parts. The first part stores the trace signals and the second part stores the scan signals. There is a very small set of important control signals that would be traced in every cycle. The remaining slots of the trace buffer will be filled with a portion of a large set of scan signals that would be dumped across several cycles. Existing approaches divide signals into two extreme categories: 1) very important and 2) less important. They lose opportunity from scenarios where some other partitioning is useful, such as very important, important, less important, and so on.

It would be beneficial if we divide the signals in a large number of categories in terms of dumping period. This enables

Manuscript received June 13, 2013; revised March 30, 2014 and October 16, 2014; accepted January 14, 2015. Date of publication February 16, 2015; date of current version December 24, 2015. This work was supported in part by National Science Foundation under Grant CNS-1441667 and Grant CCF-1218629 and in part by the Semiconductor Research Corporation under Grant 2014-TS-2554.

The authors are with the Department of Computer and Information Science and Engineering, University of Florida, Gainesville FL 32611 USA (e-mail: kamran@cise.ufl.edu; sproch@ufl.edu; prabhat@cise.ufl.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2015.2396083

[1] If each simulation for evaluating one combination takes only 1 s, $>10^{60}$ years is needed in order to find the best 32 trace signals in s35932 circuit.

1063-8210 © 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

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

The rest of this paper is organized as follows. Section II describes related works in post-silicon debug and signal selection. Section III presents the background and motivates the need for fine-grained signal selection. Section IV describes our debug architecture and proposes signal selection algorithms. Section V presents our experimental results. Finally, the conclusion is drawn in Section VI.

II. Related Work

Trace buffers have been widely studied in post-silicon debug [14]–[18]. Trace buffers are used to store the state of some selected internal signals. The rest of the signals are obtained using restoration algorithms for traced signals. The primary problem is which of the signals need to be traced to maximize the number of states that can be restored. Liu and Xu [2] and Ko and Nicolici [4] proposed efficient signal selection algorithms based on restorability metric. Basu and Mishra [3] improved their methods by proposing an efficient algorithm that considers already selected signals (using region growth) while selecting new signals. Chatterjee *et al.* [5] proposed a simulation-based signal selection algorithm to further improve the restoration performance. They showed that simulation-based signal selection is more promising compared with metric-based signal selection approaches. Prabhakar and Hsiao [19] proposed a logic implication-based trace signals selection method. Recently, Li and Davoodi [7] presented a hybrid signal selection approach combining the advantages of both simulation- and metric-based signal selection techniques.

The use of scan chains in post-silicon debug has been extensively studied in [20] and [21]. Ko and Nicolici [9] proposed an architecture that divides trace buffer bandwidth into two parts: 1) for the trace signals and 2) for the scan signals. In order to find the most beneficial partitioning, they proposed an exhaustive exploration. However, exhaustive exploration is not practical in real designs with a large number of flip-flops. Basu *et al.* [10] proposed an efficient algorithm that chooses trace and scan signals based on the connectivity graph of flip-flops. They reduce the scan chain length by pruning the graph in each iteration. However, both of these techniques divide the signals into two extreme categories. One set of signals are traced in every cycle. The other signals are dumped in a relatively large period. They do not consider other profitable fine-grained scenarios. In this paper, we propose a promising fine-grained architecture that shares the trace buffer bandwidth between several scan chains with different lengths to significantly improve the restoration performance.

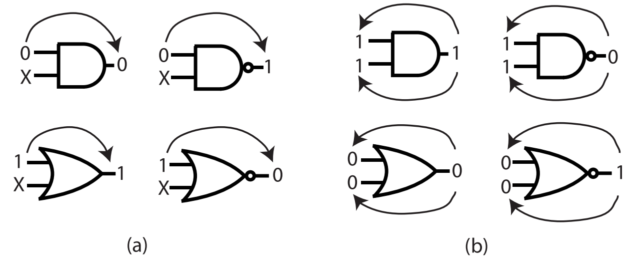

Fig. 2. Basic restoration rules for common logic gates in (a) forward restoration—knowledge of inputs can reconstruct the output—and (b) backward restoration—knowing the output can restore the unknown inputs.

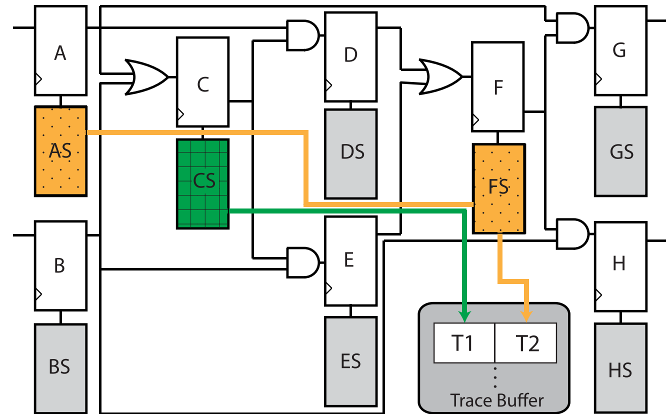

Fig. 3. Simple circuit to illustrate restorability used in [10]. Shades: shadow flip-flops. In addition, shadow flip-flops of same scan chain are shown in same pattern. Dotted scan chain: AS and FS. The first entry of the trace buffer is fed from CS, and can be viewed as a scan chain of length 1. The second entry of the trace buffer is fed from a scan chain of length 2 consisting of AS and FS.

III. Background and Motivation

In post-silicon debug, unknown signals can be restored from the traced signal states using a forward and backward restoration. Fig. 2 shows forward and backward restoration for common logical gates. Forward restoration deals with reconstructing the output from the input. For example, if one of the inputs of the AND gate is 0, the output value would be 0. If all the inputs are known, the unknown output can be definitely determined. On the other hand, backward restoration deals with restoring the inputs from the output. For example, if the output of the AND gate is 1, both of the inputs would be 1. Backward restoration might fail in certain scenarios. For example, if the output of a two-input AND gate is 0 and one of the input has known value of 0, the other input still cannot be reconstructed. During signal value reconstruction, forward and backward restorations are repeated for all the gates in the circuit until no more states can be restored.

Fig. 3 shows a simple circuit with eight flip-flops [10] with associated shadow flip-flops. Shadow flip-flops are shown in shades. For example, DS is the shadow flip-flop for *D*. In addition, the proposed debug architecture is shown using two scan chains. Shadow flip-flops of an identical scan chain are shown in same pattern. For example, AS and FS create a scan chain of length 2, which is shown in dotted pattern.

TABLE I
RESTORED SIGNALS FOR THE CIRCUIT SHOWN IN FIG. 3. SIGNAL C IS TRACED IN EVERY CYCLE AND SIGNALS A AND F ARE DUMPED IN EVERY OTHER CYCLE. AN X INDICATES THAT THE SIGNAL VALUE CANNOT BE RESTORED USING THE KNOWN SIGNAL STATES

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

We use this example to show the benefit of using different fine-grained scan chains. Restoration ratio (RR), which is a popular metric for measuring restorability, is defined as follows:

$$\text{Restoration Ratio} = \frac{\text{No. of traced and restored states}}{\text{No. of traced states}}.$$

Let us assume that the trace buffer width is 2, which means state of only two signals can be stored in each clock cycle. Table I shows the signal states that can be restored using the selected signals by [10]. Traced signals are shown in shades/bold. Signal C is traced in every cycle, whereas A and F are dumped in alternate cycles. Although scan signals are dumped in alternate cycles, the table shows states for both A and F in cycle 1, cycle 3, and so on. This is because in cycle 1, the state of signal A is dumped, whereas in cycle 2, the state of signal F is dumped. However, the scan chain (i.e., A and F shadow flip-flops) holds the state for the same cycle, although different parts were dumped in different cycles. In other words, the signal state of F captured at cycle 1 is dumped in cycle 2. Forward and backward restorations are used to reconstruct the values for the signals that were not traced. For example, the entry corresponding to D in cycle 2 will be 0, because C was 0 in cycle 1 (forward restoration). The symbol X represents the state that cannot be restored using known signal states. It can be observed that in this case, a total number of 36 states can be restored and a total number of 16 states are traced. Therefore, the RR is 2.25.

We now show how different scan chains can help in signal restoration using the same circuit. Fig. 4 shows an illustrative example of our proposed partitioning of trace buffer of width 2 for the same circuit. It can be observed that the trace buffer width is shared between two different scan chains of length 2 and length 3 shown in dotted and grid patterns. In this case, there are no trace signals and the buffer width is partitioned between two scan chains. We apply our method to select efficient signals of the sample circuit for this debug architecture. Consequently, we assign signals A and C to the first scan chain, whereas signals B, D, and E to the second scan chain. Scan chains consist of corresponding shadow flip-flops of selected signals. Table II shows the values of trace buffer and shadow flip-flops in each cycle. The subscript indicates the value in that clock cycle. For example, A_3 implies the value of flip-flop A in cycle 3. It can be observed that

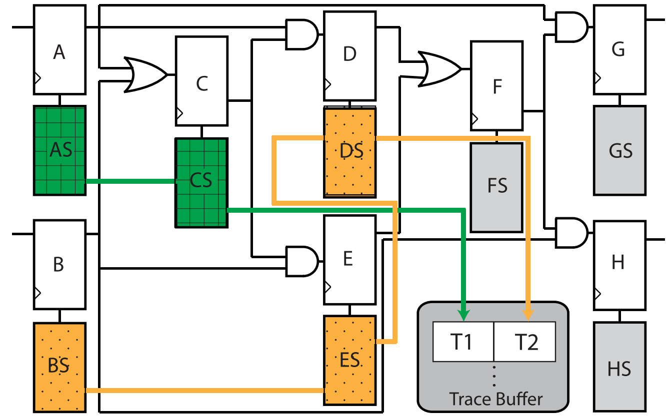

Fig. 4. Proposed debug architecture, for example, circuit in Fig. 3. Trace buffer of width 2 is shared between two scan chains of length 2 and length 3. Scan chains consist of corresponding shadow flip-flops of selected signals, shown in same color/pattern (AS → CS and BS → ES → DS).

TABLE II
TRACE BUFFER SLOTS AND SHADOW FLIP-FLOPS VALUES IN OUR PROPOSED DEBUG ARCHITECTURE IN FIG. 4. SIGNALS A AND C ARE STORED IN TRACE BUFFER IN ALTERNATE CYCLES, WHEREAS B, D, AND E ARE DUMPED IN EVERY THIRD CYCLE

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

signals A and C are stored in trace buffer in alternate cycles, whereas B, D, and E are dumped in every third cycle. In other words, signals A and C are dumped with period (T) equals to 2, whereas dumping period for signals B, D, and E is equal to 3.

Table III shows the signal states that can be restored using the signals chosen by our method (described in Section IV). It can be observed that a total number of 55 states can be restored and a total number of 17 states are traced. The RR is 3.24, which is >2.25 [10]. Thus, more states give a more detailed view of the internal state of the circuit.

The primary problem of using different scan chains is to determine the length of scan chains and signals to select for each scan chain. Signals should be chosen such that more important signals are assigned to smaller scan chains with small dumping period. Less important signals, on the other hand, should be distributed among the scan chains with larger dumping periods. In addition, minimizing the overlaps between the states that can be restored by different scan chains should be considered in selection algorithm in order to maximize the RR in a debug scenario. In this paper, we have developed two algorithms to select profitable signals for each scan chain.

TABLE III
RESTORED SIGNALS USING OUR PROPOSED DEBUG ARCHITECTURE IN FIG. 4. SIGNALS C AND A ARE DUMPED IN ALTERNATE CYCLE, WHEREAS E, D, AND B ARE RECORDED IN EVERY THIRD CYCLE

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

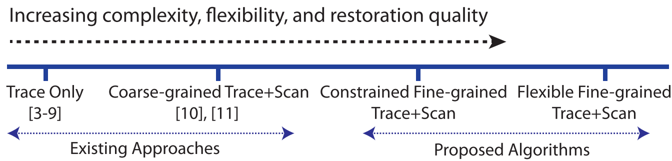

Fig. 5. Spectrum of existing and proposed debug architectures. Trace-only approach is one extreme with simple architecture but less restoration efficiency. On the other hand, flexible hardware architecture is another extreme with additional hardware controller and superior restoration performance.

IV. FINE-GRAINED COMBINATIONS

In this section, we first propose our fine-grained debug architecture. Next, we present signal selection algorithms for constrained and flexible debug architectures. In constrained debug architecture, the length of each scan chain is determined prior to the signal selection process. This approach is used when the designer fixes the scan chains lengths or there are hardware constraints in the system. The goal of constrained hardware signal selection algorithm is to assign the best possible signals to each scan chain. On the other hand, in flexible hardware architecture, there are no constraints on the and types (length) of scan chains. Therefore, the primary objective of flexible selection algorithm is to maximize the restorability, regardless of scan chains lengths. Fig. 5 shows the spectrum of existing and proposed architectures. It can be observed that trace-only and flexible hardware architectures are two extremes of this spectrum. Trace-only architecture is the simplest with less complexity, whereas flexible hardware architecture is more efficient in terms of observability but may introduce minor hardware overhead.

A. Debug Architecture

Our fine-grained architecture is motivated by the coarse-grained design of [9] and [10]. They proposed an architecture that divides the trace buffer into two parts: 1) for trace signals and 2) for scan signals. However, this partitioning is coarse grained. Very important signals are traced in every cycle, whereas the less important ones are assigned to a scan chain. The dumping period for scan signals depends on the trace buffer width and number of signals in the scan chain. Putting more signals in scan chain increases the dumping period for signals. On the other hand, putting fewer signals may not be desirable as it decreases the coverage of the circuit.

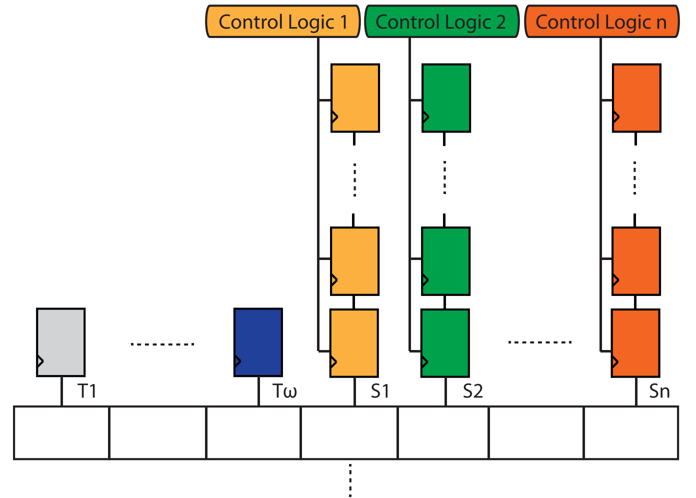

Fig. 6. Proposed debug architecture: width bw of the trace buffer is shared by ω trace signals and n scan chains of different lengths. Each scan chain consists of shadow flip-flops of same color. Dumping period for each scan chain is determined by its length and is controlled by control logic.

As discussed earlier, the limitation of the existing approaches is to consider only two extremes and losing the opportunity for not considering in-between scenarios. We consider a fine-grained approach by allowing multiple partitions of trace buffer width.

Fig. 6 shows our proposed fine-grained architecture. It can be observed that width bw of the trace buffer is partitioned for ω trace signals and n different scan chains, i.e., $bw = \omega + n$. Each of these scan chains comprises of a different number of signals denoted by an identical color that determines the dumping period for those signals. In each cycle, the shadow flip-flops of a particular scan chain capture the value of their corresponding flip-flops. It has to be noted that if a particular scan chain contains only one signal, it is essentially a trace signal that is traced in every cycle. These different signal chains provide more fine-grained dumping periods. Thus, each signal can be assigned to the appropriate scan chain based on its importance. These fine-grained scan chains enable us to dump a larger number of signals that improve the observability in the circuit compared with the coarse-grained scan signals. Sections IV-B and IV-C describe two variations (constrained and flexible) of our proposed algorithms, which try to select the best signal in each iteration considering all the signals that have been selected before.

B. Constrained Signal Selection

In this section, we propose a greedy heuristics for constrained selection algorithm that selects profitable signals for each scan chain of determined length in order to maximize the observability. We define $P_0(f)$ and $P_1(f)$ for flip-flop f in the circuit that define the probability of its value being 0 and 1, respectively. These values can be calculated by feeding the simulator with circuit graph and random input vectors and running it numerous times. We also use connectivity information similar to [10]. The connectivity of a flip-flop is the number of flip-flops connected with it through other combinational gates in both backward and forward directions.

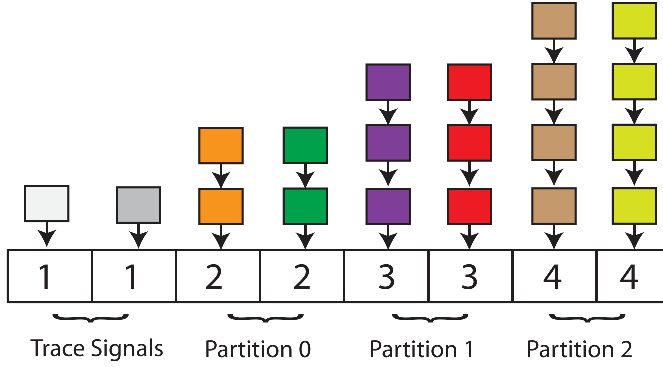

Fig. 7. Example of trace buffer partitioning in a debug architecture with $bw = 8, \omega = 2, \alpha = 3$, and $\phi(i) = \phi(i-1) + 1$.

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

Fig. 7 shows an example of trace buffer partitioning in a debug architecture with $bw = 8, \omega = 2, \alpha = 3$, and $\phi(i) = \phi(i-1) + 1$. It can be observed that there are two trace signals that are dumped in every cycle. The rest of the trace buffer is shared between the fine-grained scan chains. For example, first partition consists of two scan chains each of them with identical length of 2. In other words, two signals that are assigned to scan chain 1 will be dumped in alternate cycles. We also define dumping period (T) for each scan chain. Clearly, dumping period for a particular scan chain is equal to its length. For example, four signals are assigned to last scan chain in Fig. 7. Each of these signals will be dumped for every four cycles. Fine-grained partitioning enables us to assign signals to different dumping periods based on their importance. For example, in Fig. 7, important (control) signals are assigned to trace slots ($T = 1$). On the other hand, less important signals are assigned to scan chains with larger lengths ($T = 2$, $T = 3$, and $T = 4$), based on their impact on the restoration performance.

We use restoration power (RP) as a selection metric in our algorithm. Assume S is the current set of assigned flip-flops to scan chains. In addition, f_T implies that flip-flop f is dumped in every T cycles. We show the dumped values of f using v that can be either 0 or 1. We define $\delta(S \cup \{f_T\}, v)$ as the number of additional states that can be restored using $S \cup \{f_T\}$ (compared with restored stated using only S) over a window of c cycles when we dump f each T cycles with the assumption that the value of f is fixed to v throughput the dumping cycles. $\delta(S \cup \{f_T\}, v)$ is calculated for both $v = 0$ and $v = 1$. These values are then weighted averaged based on the probabilities of being 0 or 1 in flip-flop f and are used in the selection metric of our algorithm. Clearly, larger c is more desirable as it yields more precise result. However, large c in real scenarios is not practical as there are numerous number of flip-flops that make the restoration process computationally expensive. Our experimental results demonstrate that $c = \text{LCM}(l_0, l_1, \ldots, l_{n-1})$ is large enough where l_i is the length of scan chain i and LCM is least common multiple. The reason is that restoration pattern in whole circuit is repeated over each $c = \text{LCM}(l_0, l_1, \ldots, l_{n-1})$ cycles. For example, in Fig. 7, $c = \text{LCM}(1, 2, 3, 4) = 12$ is used in our algorithm. For a particular flip-flop f with a dumping period of T (f_T), RP is defined as follows:

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

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

Algorithm 1 outlines the major steps in our constrained signal selection (CSS) algorithm. First, we create a graph of flip-flops using the same methodology described in [3]. This graph is used to compute the connectivity[2] of each flip-flop. Next, we calculate P_0 and P_1 for each flip-flop, and partition the trace buffer using input parameters. We create an empty list S to hold the list of selected flip-flops. In each iteration, RP is calculated for all the remaining flip-flops (flip-flops that are not in S). If two or more flip-flops have equal RP, we choose the one with higher connectivity. This increases the chance of restoring more states during the real debug scenario as it is connected to more flip-flops. We continue assigning one flip-flop in each iteration until the entire scan chains get full.

We now show how our algorithm works for the example circuit in Fig. 3. Assume that trace buffer width is 2. We partition the trace buffer using $bw = 2, \omega = 0, \alpha = 2$, and $\phi(i) = 1 + \phi(i-1)$. In other words, we have two scan chains of length 2 and 3, respectively. Hence, from our algorithm, we would have $c = \text{LCM}(2, 3) = 6$, which is used in RP calculation. Table IV summarizes intermediate results of our algorithm in each iteration. First column is the candidate flip-flops from the example circuit. Second and third columns are P_0 and P_1 of each flip-flop, which are calculated by feeding our simulator with 100 different random inputs. The rest of the columns are the RP of flip-flops in each iteration. Each cell in these columns contains two rows, which are the RP values if we assign the flip-flop to the scan chain of length 2 ($T = 2$) or length 3 ($T = 3$), respectively. In the first iteration, signal C has the highest RP for $T = 2$ and is assigned to the first scan chain of length 2 (the RP value for the selected signal in each iteration is shown in bold). In the second iteration,

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

Algorithm 1 CSS

```
1: procedure CSS(circuit, bw, ω, α, φ)
2:     Create a graph model of the circuit
3:     Calculate P_0 and P_1 for all the flip-flops
4:     Create list of selected signals S //initially empty
5:     Partition trace buffer using (bw, ω, α, φ)
6:     while there is an available scan chain do
7:         for each flip-flop f that is not in the S do
8:             for each available scan chain of length T do
9:                 Calculate RP for f_T using S ∪ {f_T}
10:            end for
11:        end for
12:        Find flip-flop f with dump period T (f_T) that has
13:        the maximum RP. If two or more flip-flops have
14:        same RP, find the one with higher connectivity
15:        Assign f to a scan chain with length T
16:        Add f_T to the list S
17:    end while
18:    return S
19: end procedure
```

TABLE IV
DIFFERENT RP VALUES OF OUR ALGORITHM IN EXAMPLE CIRCUIT OF FIG. 3. IN EACH ITERATION, A NEW SIGNAL IS ASSIGNED TO ONE OF THE SCAN CHAINS. THIS PROCESS CONTINUES UNTIL ALL THE SCAN CHAINS ARE FULL

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

both signals A and B yield the maximum RP when they are assigned to scan chain of length 2. In addition, since both of them have same connectivity (3), our algorithm selects one of them (signal A) randomly. Until now, signals A and C are assigned to first scan chain of length 2. Using the same procedure, signals B, D, and E are assigned to second scan chain of length 3 in the remaining iterations.

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

Algorithm 2 FSS

```
1: procedure FSS(circuit, bw, c)
2:     Create a graph model of the circuit
3:     Create initial configuration S        ▷ empty scan chains
4:     continue = true
5:     while continue do
6:         Generate a random input vector I
7:         currentImpact = RI of (S, I, c)
8:         for each flip-flop f that is not in the S do
9:             for k_th scan chain; 1 ≤ k ≤ w do
10:                Calculate RI of (S ∪ {f_k}, I, c)
11:            end for
12:        end for
13:        Find f_k with maximum RI. If two or more
14:        flip-flops have same RI, find the one with
15:        higher connectivity
16:        if RI of S ∪ {f_k} > currentImpact then
17:            Add f to k_th scan chain of S
18:            continue = true
19:        else
20:            continue = false
21:        end if
22:    end while
23:    return S
24: end procedure
```

C. Flexible Signal Selection

In this section, we describe our flexible signal selection (FSS) algorithm that is used when there are no predefined architectural constraints on scan chains. FSS starts with initially empty scan chains. In each iteration, a flip-flop is added to one of the scan chains. This process stops once adding more flip-flops is not profitable anymore. Algorithm 2 outlines the major steps of proposed flexible selection algorithm. The inputs of the algorithm are the circuit, trace buffer width (w), and the number of cycles in mock simulations (c). To understand the workings of the algorithm, we need a key concept: restoration impact.

Given a scan chain configuration s, an input vector I, and the number of mock simulation cycles c, we define the restoration impact RI(s, I, c), as the total number of states that can be restored if we do a mock simulation over c cycles using input vector I and the scan chain configuration s. For a given scan chain configuration s and flip-flop f, $s \cup \{f_k\}$ is a new configuration, which is same as s except that flip-flop f is added to kth scan chain of s.

Informally, for a given c-cycle mock simulation on I, the restoration impact shows how scan chain configuration s is profitable in terms of observability performance. In particular, if $s_2 = s_1 \cup \{f_k\}$ for some design signal f, then $\mathrm{RI}(s_2, I, c) - \mathrm{RI}(s_1, I, c)$ measures the observability improvement achieved by augmenting kth scan chain of s_1 with f. Algorithm 2 is a greedy heuristics that uses this metric to iteratively grow the set S of current scan chain configuration. At each iteration, it: 1) performs a new simulation for c cycles using a random input vector I and computes the

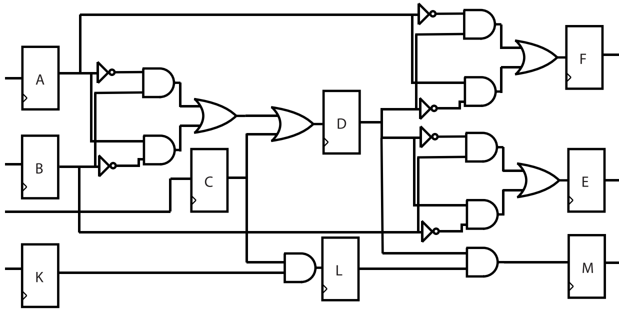

Fig. 8. Sample circuit to illustrate our flexible selection algorithm.

restoration impact of current configuration S; 2) computes the restoration impact of $S \cup f_k$ for each design signal f, which is not selected before when f is added to kth scan chain; and 3) augments kth scan chain of S with the signal that maximizes the restoration impact, if it increases the restoration impact of current configuration. If two or more signals have identical restoration impact, then the tie is broken in favor of the signal that has the highest connectivity. The process is continued until augmenting scan chains is not profitable anymore.

We now show how our algorithm works for the example circuit in Fig. 8. Assume that the trace buffer width is 2 and $c = 32$ cycles is used in mock simulations of the algorithm. Table V summarizes the intermediate results of our algorithm in each iteration. Second row shows the current configuration S in each iteration and third row demonstrates the corresponding restoration impact. The remaining rows are the restoration impact values of the circuit flip-flops in each iteration. Each cell in these rows contains two numbers, which are the RI values if we add the flip-flop to the first or second scan chain of current configuration, respectively. In the first iteration, signal L has the highest RI and is added to the first scan chain (the RI value for the selected signal in each iteration is shown in bold). Using the same procedure, signals A, D, B, and C are added to the second scan chain in the following iterations. It can be observed that in the last iteration adding a new flip-flop to any of the scan chains is not profitable anymore. In other words, none of the RI values is greater than the RI of current configuration S. The algorithm stops with L assigned to the first scan chain and A, D, B, and C assigned to the second scan chain.

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

TABLE V

RESTORATION IMPACT VALUES OF OUR FLEXIBLE SELECTION ALGORITHM IN EXAMPLE CIRCUIT OF FIG. 8. IN EACH ITERATION, A NEW FLIP-FLOP IS ADDED TO ONE OF THE SCAN CHAINS. THIS PROCESS CONTINUES UNTIL ADDING A NEW FLIP-FLOP IS NOT PROFITABLE ANYMORE

Variable/Iteration	I1	I2	I3	I4	I5	I6
S	{} {}	{L} {}	{L} {A}	{L} {A,D}	{L} {A,D,B}	{L} {A,D,B,C}
currentImpact	0	84	123	130	142	144
A	32 32	64 **127**	- -	- -	- -	- -
B	32 32	60 122	117 129	128 **142**	- -	- -
C	81 81	78 121	119 123	136 137	142 **149**	- -
D	60 60	61 118	114 **141**	- -	- -	- -
E	32 32	66 122	92 119	105 133	125 133	130 134
F	32 32	60 116	127 122	103 123	116 146	123 128
K	66 66	84 113	115 114	125 131	148 140	144 133
L	**96** 96	- -	- -	- -	- -	- -
M	80 80	77 88	97 98	130 131	149 140	144 131

V. EXPERIMENTS

A. Experimental Setup

To investigate the effectiveness of our proposed approach, we have developed a cycle-accurate simulator for ISCAS'89 benchmarks using C++. Our simulator conducts restoration in both forward and backward directions following the mechanism outlined in [2]. The simulator iterates on the unknown signals queue and tries to restore them using both forward and backward directions. This process terminates when it is not possible to restore any more states. In addition, we checked the correctness of our simulator by comparing its output with the output of Verilog simulation of the same circuits using Icarus Verilog [22].

We fed the simulator with 100 sets of random values and noted the average RRs. However, we forced the circuits to operate in their normal mode by fixing the relevant control (reset) signals, while assigning random values to all the other inputs. The control signals include active low reset signals RESET in s35932 and $g35$ in s38584 that was set to 1 in our experiments. Table VI shows the set of parameters that we used for each benchmark and different trace buffer widths for CSS algorithm.

In addition, we used $c = 64$ cycles in mock simulations of FSS algorithm. Our experiments show that using $c = 64$, relative restoration performance is consistent between a different set of trace signals and input vectors. In other words, $c = 64$ is enough to remove the random behaviors from our selection process. Important signals always perform better compared with other signals (for almost all random

TABLE VI
DIFFERENT PARAMETERS USED IN CSS

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

input vectors). Therefore, these signals are always selected in final result. We use different random signals to make sure that different states of the circuits are covered (different areas enabled). These random signals may affect borderline signals in the final result. However, the effect on restoration performance in borderline signals is negligible.

We used the original reported restoration quality numbers in [3] and [4]. We did not use the reported numbers of [5] and [7] as they used modified (performed some optimizations) version of ISCAS'89 benchmarks. To perform a fair comparison, we tried to obtain the executables of [5] and [7]. Li and Davoodi [7] provided us with their signal selection framework and we used it for the selection process in this revision. Unfortunately, we were not able to get the implementation of [5] and we used our implementation of their approach in this revision. It should be noted that we used our framework for the simulations and restoration quality calculations.

B. Restoration Quality

Table VII compares the RRs of our approach with several previous trace-only techniques [3]–[5], [7] using different ISCAS'89 benchmarks. The trace buffer used in our experiment are $8 \times 4k$, $16 \times 4k$, and $32 \times 4k$. The corresponding RR for each technique (if available) is reported. Seventh and ninth columns indicate the percentage improvement using CSS and FSS techniques, respectively, compared with the best (shown in bold) result provided by existing approaches. The improvement in restoration performance is up to 91% in s9234 (28% on average) for constrained selection algorithm. Likewise, this improvement is up to 127% in s9234 (36% on average) for flexible selection algorithm. It can be observed that our approach performs significantly better because existing trace-only approaches only take advantage of temporal observability of a small set of signals while miss the opportunity of both spatial and temporal observability of a large set of signals. The last column shows the percentage improvement by relaxing the scan chain lengths (FSS) over-constrained hardware approach (CSS). As expected, FSS outperforms CSS approach as there is no constraint on scan chains hardware. This improvement is up to 19% in s9234 and 5% on average.

We also compared our approach with the existing trace + scan approach proposed in [10] (Table VIII). It is important to note that we did not compare with other trace + scan approaches (such as [9]), since [10] has shown to perform better than other approaches. It can be observed that our approach outperforms [10] consistently. The improvement in restoration performance is up to 116% in s38584 (54.7% on average) for CSS, and up to 125% in s38584 (61.7% on average) for FSS. The reason for significant improvement is that their approach is limited by coarse-grained partitioning (two fixed partitions) of signals.

C. Selection Time

Table IX presents the runtime of our approach compared with the previous simulation-based/hybrid techniques [5], [7] using different ISCAS'89 benchmarks. The reported runtime format is hour:minute:second. For this comparison, we used an Ubuntu 12.04.5 machine with a 64-Core AMD Opteron 6378 (1400 MHz) processor and 189 GB of physical memory. We measured the runtime of [7] using their provided multithread binary file. In addition, to make the comparison fair, we used a multithread implementation of both [5] and our approach. It can be observed that compared with [5], our approach reduces the selection runtime significantly. Compared with [7], our approach demonstrated significant improvement in restoration quality, the runtime (of CSS, more specifically) is still comparable. In addition, since CSS enforces constraints on scan chain lengths, as expected, it demonstrates consistent speedup compared with FSS approach. In short, although FSS outperforms CSS in terms of restoration performance, it needs more time to run the signal selection. CSS is beneficial when there are constraints on hardware architecture or selection time. On the other hand, FSS is beneficial when the restoration quality is the primary objective. In addition, in each step of our approach, all the evaluations (RP in CSS and RI in FSS) can be done simultaneously. This makes our approach scalable for large industrial designs using MapReduce or similar programming models.

D. Hardware Overhead

To investigate the hardware overhead of our approach, we developed Verilog Register-Transfer Level (RTL) design for each of the profitable scan-trace configurations (shown in Table X) and performed logic synthesis. For each benchmark design, we selected the most profitable scan-trace combinations for different trace buffer widths. We fixed the trace buffer depth to 128 for all the experiments. The RTL design has been developed as a standalone module that accepts the scan-trace configuration as a parameter. In each scan chain, shadow flip-flops are connected through a chain and are stored in the trace buffer.

TABLE VII
RRs OF DIFFERENT TRACE-ONLY APPROACHES COMPARED WITH OUR PROPOSED ARCHITECTURE

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

TABLE VIII
RRs USING OUR APPROACH COMPARED WITH [10]

Circuit	Buffer Width	Basu et al. [10]	CSS	CSS Imprv.(%)	FSS	FSS Imprv.(%)
s38584	32	20.00	43.22	116	45.00	125
s38417	32	18.00	21.33	19	22.11	23
s35932	32	35.00	45.01	29	47.85	37

For example, Fig. 9 shows a sample configuration for a trace buffer of width 4. In this example, each entry of the trace buffer is connected to a specific scan chain. The first entry is connected to a scan chain of length 1 consisting of only one flip-flop (A). Similarly, the second entry is connected to a scan chain consisting of a flip-flop (B). In other words, A and B are essentially trace signals that are traced in every cycle. The third entry is connected to a scan chain of length 2 consisting of flop-flops C and D that are traced in alternate cycles. The last entry is connected to a scan chain of length 3 consisting of flip-flops E, F, and G that are traced in every third cycles. This configuration can be termed as 2T-1S2-1S3 to indicate that it consists of two trace signals (2T), one scan chain of two flip-flops (1S2) and one scan chain of three flip-flops (1S3).

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

We present both area and power overhead for performing fine-grained signal selection compared with existing trace-only approaches in Tables XI and XII, respectively. The area overhead computation includes the area of the entire required scan registers (shadow flip-flops that capture the state of the

TABLE IX
RUNTIME COMPARISON OF DIFFERENT APPROACHES

Circuit	Buffer Width	Chatterjee et al. [5]	Li et al. [7]	CSS	FSS
s5378	8	00:00:14	00:00:04	00:00:01	00:00:04
	16	00:00:14	00:00:08	00:00:19	00:00:33
	32	00:00:13	00:00:11	00:00:17	00:00:55
s9234	8	00:01:06	00:00:07	00:00:15	00:00:28
	16	00:01:05	00:00:19	00:00:29	00:00:51
	32	00:01:01	00:00:22	00:00:44	00:01:06
s15850	8	00:28:01	00:01:40	00:00:45	00:01:17
	16	00:28:00	00:01:58	00:04:11	00:07:02
	32	00:27:57	00:02:07	00:05:32	00:06:00
s38584	8	03:20:22	00:01:20	00:02:57	00:06:45
	16	03:20:17	00:05:40	00:06:29	00:15:08
	32	03:19:52	00:08:30	00:13:47	00:31:59
s38417	8	02:49:00	00:07:10	00:01:42	00:24:03
	16	02:49:00	00:33:30	00:09:25	00:21:18
	32	02:49:00	00:35:30	00:11:55	00:13:22
s35932	8	01:27:27	00:01:21	00:01:28	00:09:04
	16	01:27:23	00:03:36	00:03:14	00:10:31
	32	01:27:15	00:04:47	00:05:59	00:15:48

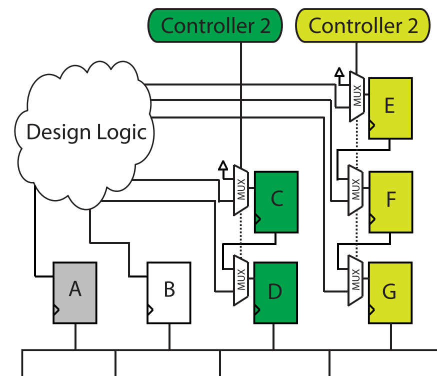

Fig. 9. Sample scan-trace configuration.

signals) as well as the necessary logic to control the timing of the signal storage for each scan chain. Scan control logic,

TABLE X
CONFIGURATIONS FOR FINE-GRAINED ARCHITECTURES

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

TABLE XI
AREA OVERHEAD OF FINE-GRAINED SIGNAL SELECTION COMPARED WITH TRACE-ONLY APPROACH

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

in this case, is essentially a counter for dumping the values of each scan chain by generating suitable control signals to enable shift operation in each cycle. In order to synthesize the designs, we used Synopsys Design Compiler version F-2011.09 and FreePDK45 target library (45-nm technology) [23].

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

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

TABLE XII
POWER OVERHEAD OF FINE-GRAINED SIGNAL SELECTION COMPARED WITH TRACE-ONLY APPROACH

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

Table XII presents the power overhead of our approach for different benchmarks and trace buffer widths. Power numbers include both dynamic and leakage power. Dynamic power is computed assuming frequency of 500 MHz. The first and the second columns indicate the benchmarks and trace buffer widths, respectively. The third column presents the power (in milliwatt) for the circuit, including the trace buffer. The fourth column presents the power (in milliwatt) of the circuit with trace buffer, including our fine-grained trace controller. The last column indicates the percentage of power overhead, which is calculated as $100 \times$ (column 4 $-$ column 5)/column 4. It can be observed that in most of the cases the power overhead of our approach is negligible (1% or less). It can also be observed that the power overhead is more noticeable in smaller circuits (s5378, s9234, and s15850). The reason is that the size of the trace buffer is dominant in these scenarios. However, the power overhead is much smaller for larger benchmarks where the circuit area is dominant compared with the trace buffer area, which is the case in real industrial designs. In conclusion, our promising fine-grained architecture proposes significant improvement in restoration quality with minor area and power overhead.

VI. CONCLUSION

Signal selection is an important part of post-silicon debug. Existing techniques mainly focused on trace-only signal selection. Recent techniques employed coarse-grained combination

of trace and scan signals and showed limited effectiveness. In this paper, we presented a debug architecture consisting of fine-grained combination of trace and scan signals. We developed efficient algorithms to select the most profitable signals based on the proposed architecture. Our experimental results using ISCAS'89 benchmarks demonstrated that our approach shows up to 127% (36% on average) higher restoration compared with existing trace-only approaches. Our approach produces up to 125% improvement (61.7% on average) compared with the state-of-the-art approaches that consider a combination of trace and scan signals. We have also demonstrated that our approach introduces minor (<1%) area and power overhead compared with existing trace-only approaches.

ACKNOWLEDGMENT

The authors would like to thank Dr. Min Li and Prof. Azadeh Davoodi from the University of Wisconsin at Madison for providing their signal selection and restoration framework.

REFERENCES

[1] A. Nahir *et al.*, "Bridging pre-silicon verification and post-silicon validation," in *Proc. 47th ACM/IEEE DAC*, Jun. 2010, pp. 94–95.

[2] X. Liu and Q. Xu, "Trace signal selection for visibility enhancement in post-silicon validation," in *Proc. Design, Autom., Test Eur. Conf. Exhibit. (DATE)*, Apr. 2009, pp. 1338–1343.

[3] K. Basu and P. Mishra, "RATS: Restoration-aware trace signal selection for post-silicon validation," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 4, pp. 605–613, Apr. 2013.

[4] H. F. Ko and N. Nicolici, "Algorithms for state restoration and trace-signal selection for data acquisition in silicon debug," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 28, no. 2, pp. 285–297, Feb. 2009.

[5] D. Chatterjee, C. McCarter, and V. Bertacco, "Simulation-based signal selection for state restoration in silicon debug," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2011, pp. 595–601.

[6] H. Shojaei and A. Davoodi, "Trace signal selection to enhance timing and logic visibility in post-silicon validation," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2010, pp. 168–172.

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

[8] K. Han, J.-S. Yang, and J. A. Abraham, "Dynamic trace signal selection for post-silicon validation," in *Proc. 26th Int. Conf. VLSI Design, 12th Int. Conf. Embedded Syst. (VLSID)*, Jan. 2013, pp. 302–307.

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

[10] K. Basu, P. Mishra, and P. Patra, "Efficient combination of trace and scan signals for post silicon validation and debug," in *Proc. IEEE Int. Test Conf. (ITC)*, Sep. 2011, pp. 1–8.

[11] F. M. De Paula, M. Gort, A. J. Hu, S. J. E. Wilton, and J. Yang, "BackSpace: Formal analysis for post-silicon debug," in *Proc. Formal Methods Comput.-Aided Design (FMCAD)*, Nov. 2008, pp. 1–10.

[12] N. Nataraj, T. Lundquist, and K. Shah, "Fault localization using time resolved photon emission and stil waveforms," in *Proc. Int. Test Conf. (ITC)*, vol. 1. Sep./Oct. 2003, pp. 254–263.

[13] D. Josephson and B. Gottlieb, "The crazy mixed up world of silicon debug [IC validation]," in *Proc. IEEE Custom Integr. Circuits Conf.*, Oct. 2004, pp. 665–670.

[14] K. Basu and P. Mishra, "Efficient trace signal selection for post silicon validation and debug," in *Proc. 24th Int. Conf. VLSI Design (VLSI Design)*, Jan. 2011, pp. 352–357.

[15] K. Rahmani, P. Mishra, and S. Ray, "Scalable trace signal selection using machine learning," in *Proc. IEEE 31st Int. Conf. Comput. Design (ICCD)*, Oct. 2013, pp. 384–389.

[16] K. Rahmani, P. Mishra, and S. Ray, "Efficient trace signal selection using augmentation and ILP techniques," in *Proc. 15th Int. Symp. Quality Electron. Design (ISQED)*, Mar. 2014, pp. 148–155.

[17] K. Basu and P. Mishra, "Test data compression using efficient bitmask and dictionary selection methods," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 9, pp. 1277–1286, Sep. 2010.

[18] K. Rahmani and P. Mishra, "Efficient signal selection using fine-grained combination of scan and trace buffers," in *Proc. 26th Int. Conf. VLSI Design, 12th Int. Conf. Embedded Syst. (VLSID)*, Jan. 2013, pp. 308–313.

[19] S. Prabhakar and M. Hsiao, "Using non-trivial logic implications for trace buffer-based silicon debug," in *Proc. Asian Test Symp. (ATS)*, Nov. 2009, pp. 131–136.

[20] G. J. van Rootselaar and B. Vermeulen, "Silicon debug: Scan chains alone are not enough," in *Proc. Int. Test Conf. (ITC)*, 1999, pp. 892–902.

[21] R. Datta, A. Sebastine, and J. A. Abraham, "Delay fault testing and silicon debug using scan chains," in *Proc. 9th IEEE Eur. Test Symp. (ETS)*, May 2004, pp. 46–51.

[22] S. Williams. *Icarus Verilog*. [Online]. Available: http://iverilog.icarus.com/, accessed Feb. 9, 2015.

[23] A. Nahir *et al.*, "Bridging pre-silicon verification and post-silicon validation," in *Proc. 47th DAC*, 2010, pp. 94–95.

Kamran Rahmani (S'12) received the B.Sc. degree from the Department of Computer Engineering, Sharif University of Technology, Tehran, Iran, and the M.S. degree from the Department of Computer and Information Science and Engineering, University of Florida, Gainesville, FL, USA, where he is currently pursuing the Ph.D. Degree with the Embedded Systems Laboratory.

He is a Senior Software Engineer with Medallia Inc., Palo Alto, CA, USA. His current research interests include post-silicon validation and debug and reliable embedded systems.

Sudhi Proch received the B.E. degree from the Motilal Nehru National Institute of Technology, Allahabad, India, and the M.S. degree from the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL, USA.

His current research interests include validation of analog and mixed-signal systems, post-silicon debug, and embedded systems.

Prabhat Mishra (S'00–M'04–SM'08) received the B.E. degree from Jadavpur University, Kolkata, India, the M.Tech. degree from IIT Kharagpur, Kharagpur, India, and the Ph.D. degree from the University of California at Irvine, Irvine, CA, USA, all in computer science.

He is currently an Associate Professor with the Department of Computer and Information Science and Engineering, University of Florida, Gainesville, FL, USA. He has authored 4 books, 10 book chapters, and over 100 research articles in premier international journals and conferences. His current research interests include the design automation of embedded systems, energy-aware computing, security and reliability, hardware/software verification, and post-silicon debug.

Dr. Mishra is a Senior Member of the Association for Computing Machinery. His research was recognized by several awards, including the NSF CAREER Award from the National Science Foundation, two best paper awards (VLSI Design 2011 and CODES + ISSS 2003), and the EDAA Outstanding Dissertation Award from the European Design Automation Association in 2004. He serves as the Deputy Editor-in-Chief of *IET Computers and Digital Techniques*. He also serves as an Associate Editor of several journals, including *ACM Transactions on Design Automation of Electronic Systems*, the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION SYSTEMS, the *IEEE Design and Test Magazine*, and the *Journal of Electronic Testing*.