# Efficient Combination of Trace and Scan Signals for Post Silicon Validation and Debug


Kanad Basu, Prabhat Mishra
Computer and Information Science and Engineering
University of Florida, Gainesville FL 32611-6120, USA
email: {kbasu, prabhat}@cise.ufl.edu

Priyadarsan Patra
Post-Si Validation Architecture
Intel Corporation, USA
email: priyadarsan.patra@intel.com



***Abstract*—Post-silicon validation is as an important aspect of any integrated circuit design methodology. The goal is to capture the bugs that have escaped the pre-silicon validation phase. A major challenge in post-silicon debug is the limited observability of internal signals in the circuit. Recent advances, such as embedded logic analysis, allow to store some signal states in a trace buffer. A promising direction to improve observability is to combine a small set of signals traced every cycle with a large set of scan signals stored across several cycles. The limited size of the trace buffer constrains the number of trace and scan signals that can be stored. In this paper, we propose an efficient algorithm to select a profitable combination of trace and scan signals to maximize the overall signal restoration performance. Our experimental results using ISCAS'89 benchmarks demonstrate that our approach can improve the signal restoration by 44% compared to the existing techniques.**


## I. Introduction

Before a chip can be delivered to a customer, it is essential to verify the device for its functional and structural correctness. Pre-silicon validation is used to check for functional (logical) errors before the design is sent for manufacturing. A combination of simulation-based techniques and formal methods is widely used during pre-silicon validation. However, due to drastic increase in design complexity and decrease in time-to-market window, it is not always possible to detect all the errors during pre-silicon validation. To capture these escaped bugs, efficient approaches are used during post-silicon validation and debug [14]. It is important to note that manufacturing testing and post-silicon validation have different primary objectives. Manufacturing testing is primarily used to detect physical (structural) defects, while post-silicon validation is designed to capture functional errors as well as errors introduced due to electrical faults.

In order to check a circuit for functional correctness, it is necessary to verify the internal signal states of the circuit with some golden reference. However, during post-silicon debug, the circuit is completely manufactured and it is not possible to probe into each and every internal signal. Recent developments such as Embedded Logic Analysis (ELA) have allowed to store some of the signal states. During the debug session, a set of input tests are used, and the selected internal signal states are stored in a trace buffer. Restoration algorithms are used to reconstruct the unknown signal states using the traced signal values. Ko et al. [11] and Liu et al. [12] have proposed efficient signal selection techniques based on partial restorability[1]. Recently Basu et al. [2] have proposed a trace signal selection technique using total restorability[2] that can restore more signals.

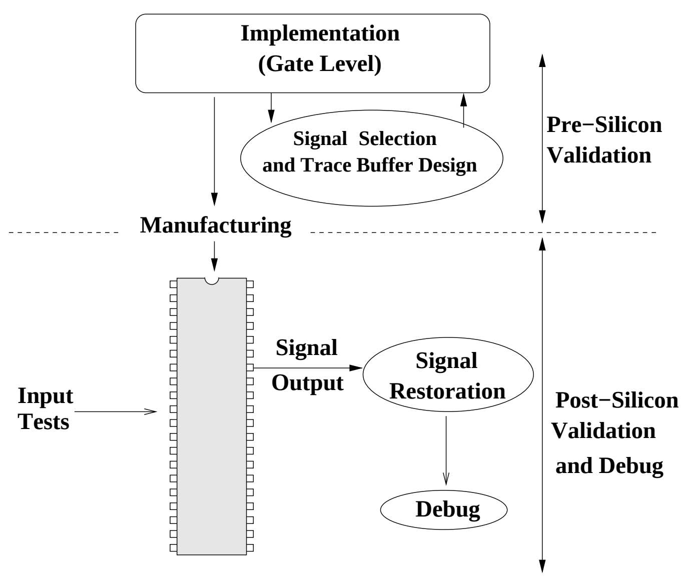


Fig. 1. Overview of post-silicon validation and debug

Figure 1 shows an overview of important activities in post-silicon validation and debug. Decisions regarding selection of efficient trace signals as well as trace buffer design are performed at gate-level model. If an error is

[1]**Partial Restorability** of a signal refers to the probability that the signal value can be reconstructed using known values of some other traced signals.

[2]**Total Restorability** is a metric to compute whether a group of signals can definitely reconstruct a set of signal states.

encountered during post-silicon validation, traced signal values are dumped out. During post-silicon debug both traced and restored signal values are used to pinpoint the error location.

Scan based debugging has been popular in manufacturing test domain. They are primarily used to identify fabrication defects. It would be beneficial to use scan dump in post-silicon debug. However, data can only be dumped in scan or debug mode, during which the design stops its normal execution, thus preventing real time observability of internal states of the circuit. Enhanced scan chains are used to address this issue where shadow flip-flops to form a shadow scan chain [10]. During scan dump, the internal states are propagated through the shadow scan chains without interrupting the normal execution of the circuit.

Ko et al. [10] have shown the importance of combining scan chains and trace signals. They use a part of the trace buffer input bandwidth to store selected trace signals every cycle. The remaining input bandwidth is used to dump the scan signals at a certain frequency. Although this approach produced promising results, there are several challenges to make it useful in practice. One major issue is that it used exhaustive exploration to determine the profitable combination of trace and scan signals. Such an exhaustive exploration can be infeasible for real designs. Another major concern is that the selected scan signals include almost all the flip-flops. Such an approach is neither practical nor profitable in many real scenarios.

In this paper, we have proposed an efficient technique to determine the profitable combination of trace and scan signals. Our approach uses a graph based representation to select i) efficient trace signals to be stored every cycle, ii) the most profitable scan signals to be included in the shadow scan chain, and iii) the scan dump frequency based on the trace buffer width constraints. It is important to note that trace signal states are stored every cycle whereas scan signal states of a specific clock cycle are stored based on dump frequency[3]. A major challenge is that these three aspects are inter-dependent. For example, selecting more trace signals implies less space for scan signals, and vice versa. Even when the space for the scan signals is reserved, choosing a large scan chain (too many scan signals) implies longer scan dump frequency. In other words, there is a critical balance between how many signals to observe versus how many signal states can be obtained for a specific clock cycle. Our proposed approach addresses these challenges. Our experimental results show that our method can significantly improve restoration ratio compared to existing methods.

The rest of the paper is organized as follows. Section II presents related works in signal selection. Section III describes combined signal selection using illustrative examples. Section IV describes our signal selection algorithms. Section V presents the experimental results. Finally, Section VI concludes the paper.

## II. Related Work

Limited observability of internal signal states is one of the biggest challenges for post-silicon debug. Knowledge of the internal signal states helps us to debug the circuit using algorithms like failure propagation tracing [3]. Similar to pre-silicon debug techniques, formal methods for post-silicon debug have been proposed by De Paula [5]. However, these formal techniques are only applicable for circuits with small number of gates. Physical probing techniques were proposed by Nataraj et al. [13]. These techniques are not useful in practice because of the high complexity of modern IC designs as well as a sharp decrease in feature size due to introduction of nanoscale technologies. DeOrio et al. [6] proposed an approach to verify memory subsystems in CMPs. However, their method is only limited to the memory and does not take into account other functional parts of the chip. Double buffering [9] of scan elements are useful for debug; but they come with an additional area penalty. Observability of internal signal states can be increased using Design-for-Debug (DfD) techniques. This is achieved by sampling the data which is stored in on-chip trace buffers. Although various DfD techniques like embedded logic analyzer [1], and shadow flip flops [9] have been proposed over the years, none of them were effective.

Trace buffer based debugging is popular these days. Some selected signal states are stored in the trace buffer, from which, the rest of the signal states are obtained. An important problem in this domain is which of the signals need to be selected for tracing. Ko et al. [11] and Liu et al. [12] have proposed efficient trace signal selection algorithms based on partial restorability. Basu et al. [2] improved their methods by proposing an algorithm based on total restorability. A logic implication based trace signal selection method was proposed by Prabhakar et al. [15]. They used the primary inputs, in addition to the traced signals for restoration purposes.

The use of scan chains for improving signal observability during post-silicon debug has been extensively

[3]For example, if the trace buffer width is 32, and 8 trace signals are used, we have space left for only 24 scan signals. If we choose a scan chain of 48 flip-flops, the scan dump frequency is in every two ($48 \div 24 = 2$) cycles. In other words, in clock cycle 0, states of 8 trace signals are stored, whereas only the states of first 24 scan signals are stored. Similarly, in the next cycle, 8 trace signal states are stored, whereas the last 24 scan signals (with states of cycle 0) are stored.

studied [4], [8], [17]. A combination of scan and trace signals for post silicon debug was first proposed by [7]. In their approach, the trace buffer is used to determine the time window over which the bug might have occurred. The experiment is then re-run with the scan data concentrating on that particular time window. Combination of trace and scan data were also used by [18] for silicon debug. They used multiple runs of the same experiment to obtain the trace data that helps in debugging. The scan data were used to select a known state. Both of these approaches suffered from the fact that they are only applicable to repeatable experiments. Hence, they are not useful when the circuit response is not uniform for multiple debug runs. [10] first proposed an approach of combining scan and trace data that works well even in non-repeatable experiments. However, their method used an exhaustive exploration of all possible trace-scan combinations to determine the one which gives the best result for a particular circuit. As a result, this approach is not useful in practice. In this paper, we have proposed an efficient algorithm to determine the profitable combination of trace and scan signals as well as the optimum scan dump frequency without explicitly exploring all possible alternatives.

## III. Background and Motivation

In post-silicon debug, unknown signal states can be reconstructed from the traced states in 2 ways - forward and backward restoration [11]. Forward restoration deals with the restoration of signals from input to output, that is, knowledge of input values is used to reconstruct the output. On the other hand, backward restoration deals with reconstructing the input from the output. Details on forward and backward restoration is available in [11].

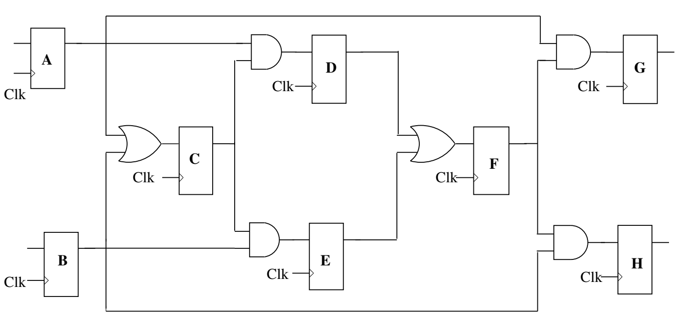


Fig. 2. Example circuit with 8 flip-flops

Figure 2 shows a simple circuit 8 flip-flops (based on [2]) to illustrate how signal restoration works for both scan chain and trace buffer based techniques. Let us assume that the trace buffer width is 2, that is, value of only two signals can be recorded in a clock cycle. Table I shows the signal states that can be restored using selected signals $A$ and $C$ (shown in shades) based on [2]. The 'X's represent those states that cannot be determined. **Restoration ratio**, which is a popular metric for calculation of signal restorability is defined as follows.

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

It can be seen that the restoration ratio of **3.2** is obtained in this case and a total of 32 states are obtained including 10 traced ones and 22 newly reconstructed ones.

TABLE I
RESTORED SIGNALS USING [2]

| Signal | Cycle 1 | Cycle 2 | Cycle 3 | Cycle 4 | Cycle 5 |
|---|---|---|---|---|---|
| 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 | 0 | 0 |

We now show how combination of trace and scan signals can help in signal reconstruction using the same circuit. In the previous example, the trace buffer stored a total of 10 states (width 2 and depth 5). In this case, we use a trace buffer that can store 11 states. Signal $C$ is selected for tracing every cycle. The other two important signals, $A$ and $F$ are used as scan signals. The scan dump is performed in alternate cycles. The eventual circuit is shown in Figure 3.

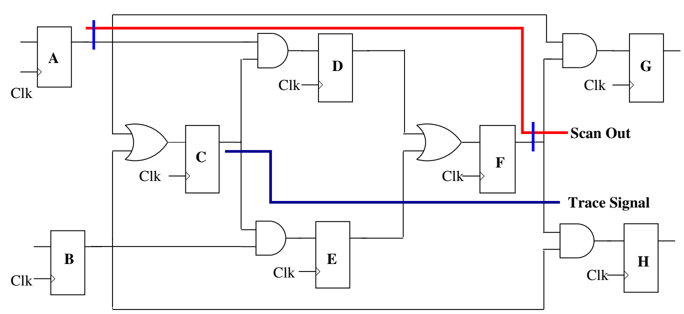


Fig. 3. Example circuit with both scan and trace signals

Table II shows the traced, scanned and restored signals. The scanned values are shown in bold. Although scan signals are dumped in alternate cycles, the table shows values 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 $F$ using shadow flip-flops) holds the value 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. As described by [10], the scan chains need not consist of flip-flops that are physically connected. For example, the scan chain here consists of flip-flops $A$ and $F$ that are connected via flip-flop $D$, which, in turn, is not part of the scan chain. In other words, a virtual scan chain can be developed only comprising of the two flip-flops $A$ and $F$. Although the restoration ratio obtained here is **3.1** (less than the no-scan method), the number of states restored is **34** which is higher than obtained earlier **32**. Thus, more signal states give a more detailed view of the internal state of the circuit.

TABLE II
RESTORED SIGNALS USING [10]

| Signal | Cycle 1 | Cycle 2 | Cycle 3 | Cycle 4 | Cycle 5 |
|---|---|---|---|---|---|
| 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 | **1** | X | **1** | 0 | **0** |
| G | X | 0 | 0 | 0 | 0 |
| H | X | 1 | 0 | 0 | 0 |

The primary problem of combining scan and trace together is to determine what signals to select for tracing, and which ones to be incorporated in the scan chain. Trace signals should be chosen such that they comprise of the important signals in the circuit that can control significant parts of the circuit. Scan chains on the other hand should be distributed around the circuit so that the total snapshot at a particular clock cycle can be obtained during debug. Since the trace buffer is getting divided between the trace signals and the scan chains, it is also important to know how this division is done. Clearly, an equal division is not beneficial since it would mean much more importance to the trace signals and decreasing the scan dump frequency. [10] tried all combinations of number of trace signals and scan dump frequency to obtain the final outcome. In this paper, we have developed an algorithm, to select an efficient combination of trace and scan signals.

## IV. Trace and Scan Signal Selection

We have proposed an efficient signal selection technique based on a profitable combination of both scan and trace data. Both the trace and scan signals are chosen during the design phase of a particular circuit. The states of the trace signals are monitored every cycle, while the scan signals are dumped at certain time intervals in a repeated fashion. We first introduce our debug architecture. Next, we describe our trace and scan signal selection algorithms.

### A. Trace+Scan Debug Architecture

Our trace-scan combined architecture is motivated by the implementation of Ko et al. [10]. The entire space of the trace buffer is divided into two parts - one for the trace data and the other for the scan dump. The states of trace signals are offloaded into the trace buffer at every clock cycle. The trace buffer width determines the number of scan signals dumped as well as the scan dump frequency. However, since the trace buffer size is constant, the total amount of data that can be stored remains fixed. With an increase in number of flip-flops in the scan chain, the amount of data produced in each dump increases. As a result, the number of scan dumps has to be decreased in order to maintain the trace buffer constraints. The scan chain is divided into small sub chains to allow complete utilization of the total trace buffer width in the same way as [16]. These are represented as $n$ sub chains in Figure 4. The partitions are shown to be numbered from 1 to $n$. Each of these $n$ sub chains utilize the trace buffer inputs for dumping. To facilitate the tradeoff between scan and trace data, [10] have proposed introduction of multiplexers in front of the trace buffer inputs. This helps in dynamically reconfiguring the inputs for the trace or the scan signals. In our case, the inputs to the trace buffer are predetermined for a particular circuit. This reduces the hardware overhead as well as delay associated with dynamic reconfiguration mechanism. The trace buffer has a width $w$ and depth $d$. Therefore, the total number of bits that can be stored in the trace buffer are $w \times d$. Here, $m$ of the inputs are dedicated for trace signals, while $n$ sub-scan chains dump their values in the trace buffer. Clearly, $w = n + m$.

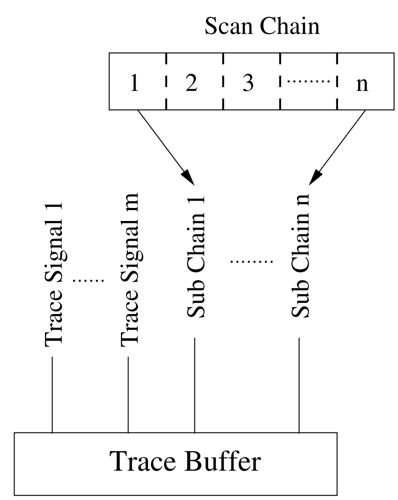


Fig. 4. Proposed Architecture: The width $w$ of the trace buffer is shared by $m$ trace signals and $n$ subchains of the scan chain

We now describe how the trace buffer based technique is used to differentiate between scan and trace data. We consider the same example circuit in Figure 3. Let the trace buffer be of width 2 and depth 5. Now, one of the two trace buffer inputs is dedicated for the trace signal, that is, $C$, which is traced every clock cycle. The other input, is dedicated to the scan chain, comprising of flip-flops $A$ and $F$. It has to be noted that since the scan chain is of length 2, the scan dump will take two cycles. The amount of data offloaded using trace based debugging is 10 bits (as seen in Table I), while the amount of data obtained using the scan-trace dual is 11 bits (as seen in Table II), which is slightly higher. A small increase in trace buffer size can help in accommodating all the scan signals. Since the scan data comes from a larger number of scan signals, the observability is enhanced compared to the original approach where only trace signals were selected for debugging.

Our proposed algorithm comprises of two parts. First, we determine which trace signals are beneficial. Next, we determine profitable set of scan signals and signal dump frequency.

### *B. Trace Signal Selection Algorithm*

In this section, we determine the signals that are needed to be traced during debug. The main problem that we face here are twofold. First of all, the trace signals need to be chosen efficiently in order to incorporate the advantages of using the scan signals during debug. Also, unlike [2] and [11], the number of signals to be traced is not fixed. Although, the maximum number of signals to be traced is equal to the trace buffer width, the actual number of traced signals can be different to accommodate the scan signals.

We use two terms *connectivity* and *threshold* in our algorithm. The *connectivity* of a flip-flop is defined as the number of flip-flops connected with it through other combinational gates in both forward and backward directions, as explained in Section III. The *threshold* is a minimum limit on the *independent connectivity* of a flip-flop, so that the flip-flop is selected for tracing. The *independent connectivity* of a flip-flop is defined as the connectivity of a flip-flop after all the flip-flops with higher or equal *connectivity* and their adjoining flip-flops are removed from the circuit. We now explain these two terms using the example in Figure 2.

The *connectivity* of the flip-flops can be determined using the circuit diagram. For example, in Figure 2, the *connectivity* of $C$ is 4, since flip-flops $A$, $B$, $D$ and $E$ are connected to it. Similarly, *connectivity* of flip-flop $A$ is 2 since only $C$ and $G$ are connected to it. To show how *independent connectivity* is computed, let's consider the flip-flop $F$. The *connectivity* of $F$ is 4, since it is connected to $D$, $E$, $G$ and $H$. In this case, the *connectivity* of $C$ is 4, equal to $F$. According to the definition of *independent connectivity*, once we remove $C$ and its adjoining flip-flops from the circuit, $D$ and $E$ get removed. As a result, the only nodes connected with $F$ are $G$ and $H$. Thus, the *independent connectivity* of $F$ is 2.

Algorithm 1 outlines the major steps in our trace signal selection algorithm. First we create a graph from the circuit, with each node representing a flip-flop. The edges between the nodes represent the path taken to reach from one flip-flop to the other. This graph construction follows the same methodology described in [2]. The graph for the example circuit (in Figure 2) is shown in Figure 5.

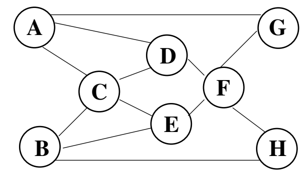


Fig. 5. Graphical representation of example circuit

As can be seen, each of the 8 flip-flops are represented by 8 nodes in the graph. The connectivity between the nodes in the graph corresponds to the flip-flop *connectivity* in the original circuit. It should be noted that no separate provision is kept for forward or backward edges. Once the graph is constructed, the node with the highest *connectivity* is selected as the most profitable trace signal. All the adjacent nodes and itself are deleted from the graph. The next node with highest *independent connectivity* is chosen. If the *independent connectivity* of the node is less than the *threshold*, the computations stop, otherwise the signal selection procedure goes on until the trace buffer width is reached.

**Algorithm 1**: Trace signal selection algorithm

**Input**: Circuit, threshold
**Output**: List of trace signals *S* (initially empty)
**1:** Create a graph *GP* from the circuit.
**while** *trace buffer is not full* **do**
    **2:** Find the node with the highest *independent connectivity* in *GP*.
    **3:** If *independent connectivity* is less than *threshold* return *S*.
    **4:** Add it to the list *S*.
    **4:** Delete the node and all its adjoining nodes from *GP*.
**end**
return *S*

For example, in Figure 5, the number of nodes is 8. Let *GP* define the entire set of nodes in the graph. Let the *threshold* be 40% of the total number of flip-flops in the circuit (i.e., 3.2). The node with the highest *connectivity* is $C$, 4, which is more than the *threshold*. Therefore, it is selected for tracing. Let $R\{C\}$, defined as relations of $C$, be the set of nodes connected with $C$, including $C$. It is obvious that $R\{C\} = \{A,B,C,D,E\}$. Step 4 of

Algorithm 1 recalculates $GP$ as

$$GP = GP - R\{C\}$$

The node with the next highest *independent connectivity* in $GP$ is $F$, with a value of 2. Since this is less than 40%, $F$ is not considered as part of the trace signal. Thus, signal $C$ is the only one selected for tracing in this case.

### C. Scan Signal Selection Algorithm

In this section, we describe our proposed algorithm for selection of both scan chain and scan dump frequency. It should be noted that in our approach, the length of the scan chain is fixed for a specific circuit, and hence, depending on the trace buffer size, the scan dump frequency is also constant.

The procedure to determine the scan chain is shown in Algorithm 2. First, we create a graph from the circuit, in the same way described in Section IV-B. Once the graph is constructed, all the nodes that are part of the trace signals or are connected to those trace signals are removed from the graph. Then, a minimal node set is obtained from the graph. A minimal node set has two requirements. First, it is a group of nodes such that each and every other node in the graph is connected to at least one node in the set. Also, it should be minimal i.e., the set should have the least number of nodes. The procedure to obtain the minimal node set is shown in Algorithm 3. The flip-flops corresponding to the nodes in the node set constitute the scan chain. We first describe the two important steps in this algorithm: graph construction and creation of minimal node set. Next, we use an illustrative example to describe how the algorithm works.

---

**Algorithm 2**: Scan signal selection algorithm

---

**Input**: Circuit
**Output**: List of scan signals $S$ (initially empty)
**1:** Create a graph from the circuit. by
Remove the trace buffer related nodes.
**2:** Find the node values for all the nodes in the circuit.
**3:** Find the minimal node set from the graph.
**4:** Put all the flip-flops in the node set in $S$.
return $S$

---

*1) Graph Construction:* The first step of the algorithm is to construct a graph from the circuit. The graph for the example circuit (in Figure 2) is shown in Figure 6. Once the graph is constructed, we proceed to compute the values of each node. The value of each node is the number of nodes connected with it ( *connectivity*, as was defined in Section IV-B). The values are shown next to each node in Figure 6.

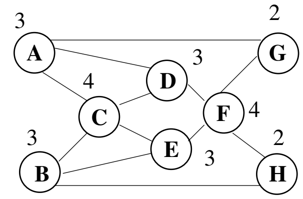


Fig. 6. Graphical representation of example circuit

*2) Creation of Minimal Node Set:* The algorithm for minimal node set construction is shown in Algorithm 3.

---

**Algorithm 3**: Minimal signal set creation

---

**Input**: Circuit as graph, Node Values
**Output**: Minimal Node Set $S$ (initially empty)
**1:** Put all the nodes in a list $GPS$.
**while** *G is not empty* **do**
    **2:** Find the node in $GPS$ with the highest value.
    **3:** Remove the node from $GPS$.
    **4:** Remove all nodes associated with that node from $GPS$ along with their associated edges.
    **5:** Recompute node values based on current nodes in $GPS$.
**end**
return $S$

---

We now explain each of the steps in the algorithm using the graph in Figure 6. It should be noted that since the circuit in Figure 2 is small, we have not taken into consideration the effect of trace buffer nodes; that is, we have shown the scan signal approach independently of the algorithm described in Section IV-B. As can be seen from Figure 6, the node $C$ and $F$ have highest value. We choose node $C$ as the initial node. The nodes associated with it, that are, $A$, $B$, $D$ and $E$ are also removed along with their corresponding edges. The node values are then recomputed based on the present *connectivity*. The graph $GPS$ obtained along with the revised node values is shown in Figure 7.

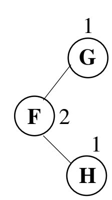


Fig. 7. Graphical representation of example circuit after $C$ and associated nodes are removed

The node with the next highest value is found to be $F$. Once $F$ is selected, it can be seen that no nodes remain in the graph. Therefore, the computation stops. The scan chain obtained comprises of the two flip-flops $C$ and $F$. The circuit graph along with the scan chain is shown in Figure 8. The nodes that are part of the scan chain, $C$ and $F$ are shown in shades.

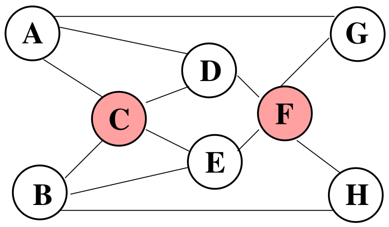


Fig. 8. Graphical representation of example circuit with scan chain

The basic idea behind this form of scan cell selection is that each node in the entire circuit is either in the minimal set or connected to at least one node in the set. Therefore, when the scan dumps (of flip-flops in the minimal set) are performed, the nodes that are not in the minimal set get their states reconstructed based on scan dumps. For example, in Figure 8, if the state of $C$ is dumped in cycle $i$, the states of $A$ and $B$ can be obtained in cycle $i-1$, while the state of $D$ and $E$ can be obtained in cycle $i+1$.

It should be noted that since the scan chain length is constant, for a particular trace buffer size and number of inputs dedicated to the trace signals, the scan dump frequency is also constant. Let the trace buffer depth and width be $d$ and $w$ respectively. Let $n$ be the number of inputs dedicated to the trace signals. Therefore, the number of inputs of the trace buffer dedicated to the scan chain per cycle are $w-n$. Let the scan chain length be $l$. Therefore, number of cycles it takes to dump the entire scan chain into the trace buffer is $\frac{l}{w-n}$. This determines the scan dump frequency, since the scan chain will be dumped after each $\frac{l}{w-n}$ cycle.

## V. Experimental Results

We applied our approach using the ISCAS'89 benchmarks. These benchmarks were also used by existing approaches [10]. The trace buffer is chosen with a width of 32 and a depth of 1024. Table III shows the results of comparison with [10]. Column 2 reports the number of trace signals selected using Algorithm 1. To enable fair comparison, we report the best results of [10] using the same number of trace signals. The next two columns represent the number of states restored using the method outlined in [10] and our approach. Finally, the last column gives the improvement, which is the ratio of the number of extra states restored by our approach compared to the states restored using [10][4].

We have defined a parameter known as *threshold* in Section IV-B. Table IV shows how the restoration

[4]It should be noted that the numbers we presented for [10] is different from the ones reported in [10]. We generated their numbers using the same setup including same set of trace signals. Our experiments produced different results probably due to different random inputs. We have reported this disparity to the primary author of [10].

TABLE III
Comparison with [10]

| Circuit | Trace Signal | Restored States [10] | Restored States Our Approach | % Improvement |
|---|---|---|---|---|
| s38584 | 2 | 283792 | 332854 | 17.29% |
| s38417 | 4 | 540603 | 601878 | 11.33% |
| s9234 | 2 | 52080 | 75220 | 44.43% |
| s15850 | 2 | 83569 | 89041 | 6.54% |

ratio varies with the change in *threshold* using the 7 largest ISCAS'89 benchmarks. As before, a trace buffer of width 32 and depth 1024 is used. The first column gives the circuit name. The next two columns signify the *threshold* as a percentage of the total number of signals in the circuit and the scan chain length. The fourth column gives the number of scan dumps for the particular configuration. The fifth column shows the number of trace signals for that particular *threshold*. The last two columns signify the number of states restored and the restoration ratio respectively. The variation of restoration ratio is not significant in this case.

We now compare our proposed approach with the existing trace only approaches in [2] and [11]. Figure 9 shows the comparison using 3 largest ISCAS benchmarks. As expected, our proposed approach outperforms for $s38417$ benchmark. Our approach also outperforms [11] for $s38584$ benchmark. However, trace only approaches perform better for $s35932$ because this benchmark is very well connected. As a result, a small set of selected trace signals can restore a large set of unknown signals. Therefore, when the trace signals are sacrificed for scan signals, some of these profitable signals are not traced every cycle and restoration diminishes if both trace and scan signals are selected. In summary, it is beneficial to select a large set of trace signals (or employ trace only approach) if a design is well connected (very hign fan-in/fan-out), whereas selection of both trace and scan signals is beneficial in all other scenarios.

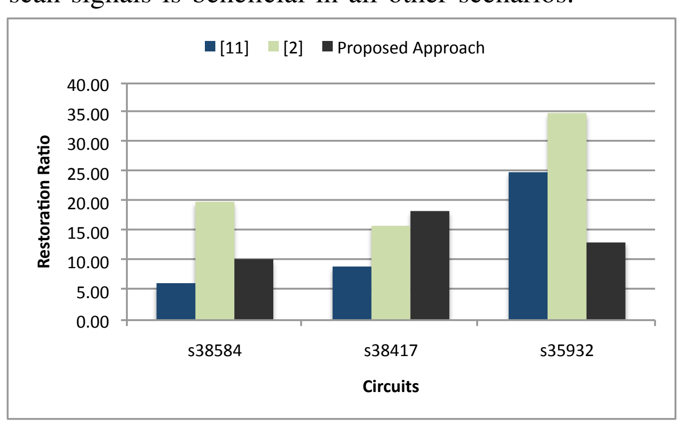


Fig. 9. Comparison with [11] and [2]

TABLE IV
EFFECT OF *threshold* ON TRACE SIGNAL SELECTION

| Circuit | Threshold | Scan Chain Length | Number of Scan Dumps | Number of Trace Signals | Number of States Restored | Restoration Ratio |
|---|---|---|---|---|---|---|
| s38584 | 10% | 182 | 170 | 2 | 332457 | 10.15% |
| | 5% | | 161 | 4 | 332854 | 10.16% |
| s38417 | 10% | 236 | 135 | 1 | 601878 | 18.37% |
| | 5% | | 127 | 3 | 556503 | 16.98% |
| s35932 | 10% | 300 | 106 | 1 | 338090 | 10.32% |
| | 5% | | 106 | 1 | 338090 | 10.32% |
| s15850 | 10% | 133 | 240 | 1 | 89041 | 2.71% |
| | 5% | | 216 | 5 | 85261 | 2.6% |
| s13207 | 10% | 195 | 163 | 1 | 171613 | 5.24% |
| | 5% | | 154 | 3 | 158366 | 4.83% |
| s9234 | 10% | 53 | 593 | 3 | 74678 | 2.28% |
| | 5% | | 585 | 4 | 75220 | 2.3% |
| s5378 | 10% | 61 | 529 | 1 | 96538 | 2.95% |
| | 5% | | 493 | 5 | 108965 | 3.33% |

## VI. CONCLUSIONS

Post-silicon validation is extremely complex and time consuming in overall design methodology. Trace buffer based debug is widely used. Combining trace and scan signals is a promising approach that helps in enhanced signal reconstruction. We developed efficient algorithms to select the profitable trace signals and scan chains that can maximize restoration ratio. Our experimental results using ISCAS'89 benchmarks demonstrated that our method provided up to 44% higher restoration compared to existing approaches.

## REFERENCES


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

[2] K. Basu and P. Mishra. Efficient Trace Signal Selection for Post Silicon Validation and Debug. In *Proceedings of the International Conference on VLSI Design*, 2011.

[3] O. Caty, P. Dahlgren, and I. Bayraktaroglu. Microprocessor silicon debug based on failure propagation tracing. In *Proc. IEEE International Test Conference ITC 2005*, pages 10pp.–293, 8–8 Nov. 2005.

[4] R. Datta, A. Sebastine, and J. Abraham. Delay fault testing and silicon debug using scan chains. In *European Test Symposium, Proceedings.* Published by the IEEE Computer Society, 2004.

[5] F. M. De Paula, M. Gort, A. J. Hu, S. Wilton, and J. Yang. Backspace: Formal analysis for post-silicon debug. In *Proc. FMCAD '08. Formal Methods in Computer-Aided Design*, pages 1–10, 17–20 Nov. 2008.

[6] A. DeOrio, I. Wagner, and V. Bertacco. Dacota: Post-silicon validation of the memory subsystem in multi-core designs. In *Proc. IEEE 15th International Symposium on High Performance Computer Architecture HPCA 2009*, pages 405–416, 14–18 Feb. 2009.

[7] J. Gao, Y. Han, and X. Li. A New Post-Silicon Debug Approach Based on Suspect Window. In *VLSI Test Symposium, 2009. VTS'09. 27th IEEE*, pages 85–90. IEEE, 2009.

[8] X. Gu, W. Wang, K. Li, H. Kim, and S. Chung. Re-using DFT logic for functional and silicon debugging test. In *Test Conference, 2002. Proceedings. International*, pages 648–656. IEEE, 2002.

[9] D. Josephson and B. Gottlieb. The crazy mixed up world of silicon debug [ic validation]. In *Proc. Custom Integrated Circuits Conference the IEEE 2004*, pages 665–670, 3–6 Oct. 2004.

[10] H. Ko and N. Nicolici. Combining scan and trace buffers for enhancing real-time observability in post-silicon debugging. In *Test Symposium (ETS), 2010 15th IEEE European*, pages 62–67. IEEE, 2010.

[11] H. F. Ko and N. Nicolici. Algorithms for state restoration and trace-signal selection for data acquisition in silicon debug. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 28(2):285–297, Feb. 2009.

[12] X. Liu and Q. Xu. Trace signal selection for visibility enhancement in post-silicon validation. In *Proceedings of the Design Automation and Test in Europe*, 2009.

[13] N. Nataraj, T. Lundquist, and K. Shah. Fault localization using time resolved photon emission and stil waveforms. In *Proc. International Test Conference ITC 2003*, volume 1, pages 254–263, Sept. 30–Oct. 2, 2003.

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

[15] S. Prabhakar and M. Hsiao. Using non-trivial logic implications for trace buffer-based silicon debug. In *Proc. Asian Test Symposium*, pages 131–136, Dec. 2009.

[16] J. Saxena, K. Butler, and L. Whetsel. An analysis of power reduction techniques in scan testing. In *Test Conference, 2001. Proceedings. International*, pages 670–677. IEEE, 2002.

[17] G. Van Rootselaar and B. Vermeulen. Silicon debug: Scan chains alone are not enough. In *Test Conference, 1999. Proceedings. International*, pages 892–902. IEEE, 2002.

[18] Y. Yang, N. Nicolici, and A. Veneris. Automated data analysis solutions to silicon debug. In *Design, Automation & Test in Europe Conference & Exhibition, 2009. DATE'09.*, pages 982–987. IEEE, 2009.