Simulation-based Signal Selection for State Restoration in Silicon Debug

Debapriya Chatterjee, Calvin McCarter and Valeria Bertacco
Department of Computer Science and Engineering, University of Michigan
Email: {dchatt, cblue, valeria}@umich.edu

***Abstract*—Post-silicon validation has become a crucial part of modern integrated circuit design to capture and eliminate functional bugs that escape pre-silicon verification. The most critical roadblock in post-silicon validation is the limited observability of internal signals of a design, since this aspect hinders the ability to diagnose detected bugs. A solution to address this issue leverage trace buffers: these are register buffers embedded into the design with the goal of recording the value of a small number of state elements, over a time interval, triggered by a user-specified event. Due to the trace buffer's area overhead, only a very small fraction of signals can be traced. Thus, the selection of which signals to trace is of paramount importance in post-silicon debugging and diagnosis. Ideally, we would like to select signals enabling the maximum amount of reconstruction of internal signal values. Several signal selection algorithms for post-silicon debug have been proposed in the literature: they rely on a probability-based state-restoration capacity metric coupled with a greedy algorithm. In this work we propose a more accurate restoration capacity metric, based on simulation information, and present a novel algorithm that overcomes some key shortcomings of previous solutions. We show that our technique provides up to 34% better state restoration compared to all previous techniques while showing a much better trend with increasing trace buffer size.**

I. Introduction

Shrinking transistor sizes with each new generation of digital integrated circuits (IC) have allowed modern IC designs to include more and more logic, thus becoming increasingly complex. Concurrently, the time-to-market for new IC products has been shrinking rapidly. This phenomenon has put enormous burden on the verification flow of digital designs. Traditionally, functional bugs in a design have been identified through the extensive use of simulation and formal verification techniques in the pre-silicon phase. However, with shorter design cycles, and considering the limited speed of simulation and limited capacity of formal tools, these methodologies are often insufficient to detect functional bugs that manifest deep in the design's state space or are very infrequent. As a result, the first silicon prototypes often still contain design bugs, even if they clear manufacturing testing. To facilitate detection and investigation of these bugs, post-silicon debug has emerged in recent years as a crucial technique.

The fundamental challenge in silicon debug lies in the very limited visibility of internal design signals. The capabilities of physical probing tools [1] are very limited, and it is infeasible to observe each and every signal in fabricated silicon. So far, reusing *design for test* (DFT) circuit structures, such as internal scan chains, for silicon debug has been widely adopted in the industry [2]. Though scan chains can capture all or a subset of internal state elements, and thus increase signal observability for silicon debug, it may take several thousand clock cycles to dump out one observed state snapshot and, in most cases, the circuit's execution must be suspended until the completion of this process. The inclusion of shadow flip-flops in the scan chain can maintain normal circuit operation during the scan transfer, but it requires higher area overhead, and can still only produce one snapshot every few thousands cycles, which is too infrequent for being useful in most debugging efforts.

To facilitate silicon debug, *design for debug* (DFD) structures such as embedded logic analyzers (ELAs), have been proposed [3] and have found widespread use in the industry [4], [5], [6]. An ELA consists of a mix of trigger units and sampling units. Programmable trigger units are used to specify an event for triggering the logging of internal signal values. Sampling units are used to log the values of a small set of signals (trace signals) over a specified number of clock cycles into trace buffers. The number of signals traced is known as the *width* of the trace buffer, while the length of the tracing interval is called *depth*. Trace buffers are implemented with on-chip embedded memories [5] and data acquisition can be performed during normal chip operation by setting up the relevant trigger event. Subsequently, the sampled data is transferred off-chip via low bandwidth interfaces for post-processing analysis for debug. Note that DFD structures must maintain a low area overhead profile, since they do not provide added benefits to the design. As a result, only a very small number of signals can be traced in comparison to those available in the design.

For ELAs to be effective, designers must carefully select for tracing those signals that yield the most debug information. Through a judicious choice of trace signals, one can even reconstruct data for state elements that are not traced. As an example, for micro-processor designs, it is common practice to trace pipeline control signals so that the values of other data registers can be inferred during post-analysis. This approach cannot be used for a general circuit, however, because it leverages architectural knowledge of the design. Indeed, the need for generalized solutions in this domain is growing.

Even though the additional inferred information does not guarantee the identification of design errors, it still increases internal signal visibility and has the potential of providing valuable debugging information. Because bugs tend to occur in unexpected regions and configurations, it is not always possible to predict the most important signals to trace. Ideally we would like a mechanism which allows to reconstruct almost all internal signals from the tracing of just a handful of signals, so as to offer pre-silicon quality observability during post-silicon debug.

Recent research addressing these challenges [7] has shown that many un-traced signals and state elements can be inferred from a small number of traced state elements by forward and backward implication, even in arbitrary logic. Ko and Nicolici [7] were first to propose an automated trace signal selection method that attempts to maximize the number of non-traced states restored from a given number of traced state elements. The quality of the trace signal selection was quantified by the state restoration ratio (SRR), that is, the ratio of the number of state values restored over the state values traced, over a given time interval. This measure has been adopted by subsequent research to compare the quality of other solutions. Further research [8], [9], [10] has proposed several automated trace signal selection methods based on different heuristics for estimating the state restoration capabilities of a group of signals. These research solutions share a common structure: (i) a metric to estimate the state restoration capability of a set of state elements and (ii) the use of the metric in a greedy selection process to evaluate candidate set of signals and converge to a final selection.

In this work we show that a more accurate metric for state restoration capability of a set of signals can be obtained by actually simulating the restoration process on the circuit over a small number of cycles, and measuring the corresponding restoration ratio. We also propose a novel signal selection method guided by this metric. Our solution overcomes a key shortcoming of previous greedy approaches to a large degree, namely that of diminishing returns: when the number of traced signals is increased, additional restored state elements increases sub-linearly.

A. Contributions

The main contributions of this work can be summarized as follows:

- We show that computing the SRR by simulation of the design over a small number of cycles (compared to the typical depth of the trace buffer in use) provides an accurate estimate of the SRR obtained from actual trace buffer data over a longer period.
- We propose a novel trace signal selection method based on iterative elimination of state elements. We show experimentally that our solution provides better trends when the number of traced signals is increased.
- Experiments show that our solution provides up to 34% better state restoration ratio compared to all previous solutions.

II. Related Work

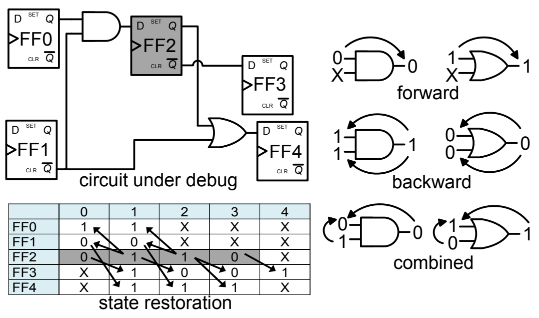
Automatic trace signal selection algorithms for post-silicon debug are a fairly new research area. One of the first solutions in this domain [11] considered only the reconstruction of data at the combinational logic nodes of the circuit. Ko and Nicolici [7] defined the term state restoration and introduced an efficient algorithm to perform state restoration as a post-analysis process on recorded trace-buffer data. They also introduced the first trace signal selection algorithm striving to maximize the amount of restored state. Further research in this area has produced several improved solutions for automatic signal selection [8], [9], [10], all sharing the goal of improving the SRR.

As mentioned earlier, these solutions share a common structure, with a metric to estimate the restoration capacity of a certain set of state elements and a greedy selection algorithm to decide which ones to trace, based on the estimator metric. These previous solutions primarily differ in the way estimation is performed. Both [7] and [8] leverage a probabilistic metric: the steady state probability of the value at flip-flop outputs is estimated assuming uniform random distribution of 0 and 1 logic values at the primary inputs. Given these assumptions and using the knowledge of the traced signal values, a probabilistic model of the *visibility* of 0 and 1 values at the other circuit nodes can be generated. This probabilistic model leverages the circuit topology and logic functionality of individual gates, and the estimation process performs forward and backward propagation of probability values across logic gates. The final state restoration capacity estimate is then expressed as a sum of the predicted visibility of 0 and 1 values at the state elements of the circuit. The probabilistic model presented in [7] lacks theoretical basis and it is then improved on in [8]. In contrast, [10] considers only the restoration probability along paths connecting flip-flops. The probability that a flip-flop output value controls the input value of another flip-flop is computed and called *direct restorability* of the corresponding path. The selection algorithm grows a region of flip-flops in a greedy fashion based on this metric, while an adjustment mechanism accounts for flip-flops that are already selected in the region and updates the path's probabilities accordingly. Another solution presented in [9] estimates the visibility of non-traced nodes by non-trivial logic implications of flip-flop values. However, [9] assumes that in addition to trace signals, all primary input values for every cycle are known to the restoration algorithm. Our proposed solution is fundamentally different from these previous ones as it relies on simulation for estimation instead of a probabilistic metric.

Another line of research [12], [13] suggests that not all state elements or signals are equally relevant for debugging purposes. Hence, instead of striving to maximize the state restoration ratio, the authors of those works focus on maximizing restorability of a specified subset of signals, while minimizing the impact to other flip-flops. In particular, the algorithm in [13] uses a probabilistic estimation metric analogous to [8], and follows a pareto optimal selection process. We show that our solution can be adapted to solve this problem variant as well, by simply assigning larger weight coefficients to the set of critical flip-flops.

III. Background and Motivation

An ideal post-silicon debugging solution would enable a pre-silicon quality observability, *i.e.*, every signal value is observable at each cycle, with little design effort and area overhead. A more realistic goal is to attain partial observability by tracing a small set of signals and use them to find the root cause of the bug. Several previous solutions have suggested automatic signal selection algorithms to determine which state elements allow maximum restoration if traced. An intuitive measure for evaluating restoration quality is the state restoration ratio, defined as $SRR = \frac{N_{traced}+N_{restored}}{N_{traced}}$, where N_{traced} is the number of traced state elements and $N_{restored}$ is the number of restored ones during the time window dictated by the trace buffer's depth. Automated signal selection strives to maximize SRR.



	0	1	2	3	4
FF0	1	1	X	X	X
FF1	0	0	X	X	X
FF2	0	1	1	0	X
FF3	X	1	0	0	1
FF4	X	1	1	1	X

Fig. 1. **Example of state restoration process**. The circuit shown at the top left is the circuit under debug, with flip-flop FF2 traced for 4 clock cycles (shown in grey). The table below lists the values of all flip-flops, whether traced, restored or unknown(X). Forward inference and backward justification through the logic gates (shown with forward and backward arrows in the table) allows to restore several flip-flop values that were not traced. The elementary rules of forward inference, backward justification and combined inference are shown for two types of logic gates on the right side of the figure.

A. State Restoration Process

The state restoration process relies on the property that if a controlling value is known for at least one input of a logic gate, the output can be inferred without the knowledge of other inputs. This property is used for forward inference of signal values in case of partial knowledge. Similarly, if a non-controlled value is observed at the output of a gate, all inputs can be inferred to hold the non-controlling value for that type of gate, enabling backward justification. Combined inferences leveraging knowledge of both inputs and output are also possible (see Figure 1. Repeated application of these simple operations for all gates of a circuit till no new value can be generated, leads to value reconstruction for state elements beside those traced. This process is used in the post-analysis of the data obtained from trace-buffers to restore non-traced signals. Figure 1 illustrates this process with an example inspired by [7]. In this example, the flip-flop FF2 is traced over four clock cycles; additional values at other flip-flops can be inferred as shown in the table in the lower part of the figure. In this particular example, the state restoration ratio, $SRR = 15/4 = 3.75$ ($N_{traced} = 4, N_{restored} = 11$). An efficient bit-parallel algorithm to perform this restoration process is introduced in [7], and it is extensively used in our implementation. It is important to note that the forward inference and backward justification operations are correct only if the logic functions of the gates in the circuit conform to the structural netlist, with no stuck-at-faults or other such faults (this is assured since the IC has cleared manufacturing tests). Timing errors must also be avoided for correct restoration, a goal that can be attained by reducing the clock frequency during debug operations. Hence, this technique is only effective for investigating functional bugs. The key challenge of this process is how to select which state elements to trace among the thousands of a typical design to achieve the best possible restoration of internal signals and other state elements.

B. Structure of Signal Selection Algorithms

Most signal selection algorithms presented in the literature so far [14], [8], [9], [10] share a common structure. First, a metric is devised to estimate the state restoration capacity of a given set of signals; second, a greedy selection process guided by the metric is used to converge to a locally-optimal selection. Figure 2 summarizes this general structure.

```
Input: circuit, width of trace buffer $w$,
restoration capacity metric $f_C(...)$
Output: selected flip-flop set $T$

while ($|T|$ < $w$) {
  maximum visibility $maxV = 0$
  for (each unselected flip-flop $s$ in circuit){
    $T = T \cup \{s\}$
    visibility $V = f_C(T)$
    $T = T \setminus \{s\}$
    if($V > maxV$){
      $selected = s$
      $maxV = V$ }
  }
  $T = T \cup \{selected\}$
}
```

Fig. 2. **General structure of greedy selection algorithms.**

For the algorithm to be successful, the capacity metric should have the following properties: (i) it should be proportional to the actual average SRR that can be obtained with the given set of signals over many runs, (ii) it should be as computationally inexpensive as possible, since several such computations will be needed in the final selection process. The first criterion is especially important for the greedy selection process to be successful, since it guides the successive greedy choices towards the optimal subset. The greedy selection process starts off with the signal which promises the maximum capacity, and then enlarges the set, one signal at a time, by evaluating the restoration capacity of all possible candidate sets including one more signal. In Section IV-A, a better capacity metric obtained by simulated restoration is explored, while a critical shortcoming of the greedy selection process itself is detailed in the next section.

C. The Problem of Diminishing Returns with Greedy Selection

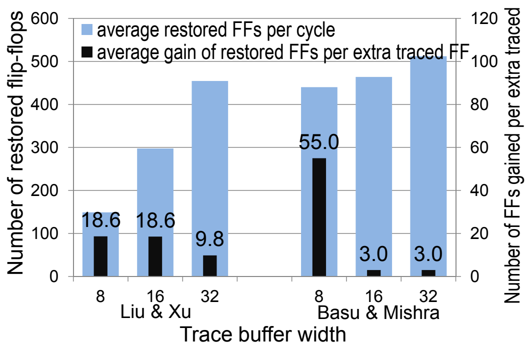


Fig. 3. **Diminishing returns in restored flip-flops when increasing trace buffer size** is observed for two previous solutions. The plots correspond to circuit s38417.

The greedy selection process suffer from another critical problem with regards to the quality of the chosen signals. Figure 3 plots the average number of restored flip-flops per cycle for 3 different trace buffer widths (8, 16 and 32) for the ISCAS89 benchmark circuit s38417. Alongside, we also plot the average number of restored flip-flops attained when adding each new traced flip-flop (FF). The plots correspond to the data reported by Liu & Xu [8] and by Basu & Mishra [10]. Note that in the result obtained by Liu & Xu, an increase of the observed FFs from 8 to 16 corresponds to an increase in the number of restored FFs from 149 to 298, leading to a $(298 - 149)/(16 - 8) = 18.62$ gain per added new trace signal, as shown by the inner dark bar corresponding to width 16. However, when the traced signals increase from 16 to 32, the rate of gain is much lower (9.8). This effect is even more pronounced in the results by Basu & Mishra [10], where a much better initial restoration is obtained, but as the number of trace signals are doubled, the improvement is minute. This behavior results from the inaccuracy of the estimation metric, as well as the very nature of the greedy selection. Indeed, the restoration obtained by greedy selection algorithm plateaus when a large number of flip-flops are traced. This is because the selection of $2n$ flip-flops is constrained by the previous selection of the first n flip-flops. In contrast, the best possible set of $2n$ flip-flops might not even include some of the first n flip-flops. Hence, we propose an alternative approach that applies greedy selections backward: *i.e.*, we start off with the set of all FFs, and then we iteratively reduce this set until we obtain a set of the desired cardinality. In the following section we outline an algorithm based on this approach.

IV. Signal Selection Algorithm

We first derive a more accurate restoration capacity metric, and then we use this metric in our proposed algorithm.

A. Improving the Restoration Capacity Metric

A good restoration capacity metric should have a high degree of correlation with the actual SRR in the post-silicon post-analysis, since the more accurate the metric, the more likely it is to obtain an optimal subset of signals in the selection process. To evaluate the quality of a restoration capacity metric, we devise the following experiment: we choose 1,000 random sets of 8 flip-flops each and measure the average SRR in each set, using a trace buffer depth of 4,096, obtained with 100 simulation runs on the same design (we used 10 sets of random seeds and 10 different starting points for tracing per seed). We also asserted the appropriate control signals to ensure that the circuit would operate in its normal functional mode during the simulation. Figure 4 plots the average SRR vs. the estimated one obtained with the Liu & Xu's restoration capacity estimation metric. Data is shown using a scatter plot to highlight the correlation of the metric with the actual measured SRR.

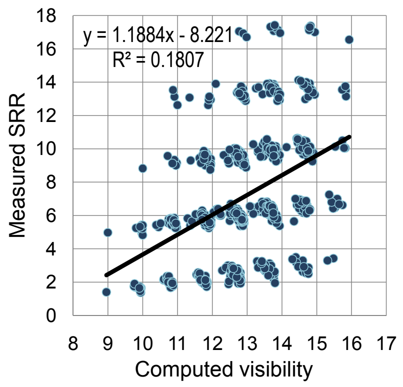


Fig. 4. **Correlation of the Liu & Xu restoration capacity metric with measured SRR for s35932.** The metric has a positive but poor correlation with measured SRR. We also report a linear regression fit of the data and the square of the correlation coefficient. Data points in the lower right corner represents selection of flip-flops that have a high estimated value of state visibility but rather poor measured SRR. This can drive the greedy selection algorithm to sub-optimal selections.

As can be noted from the figure, although the metric has positive correlation with measured SRR, the extent of correlation is poor, as indicated by the small correlation coefficient (R). The fundamental reason behind this pattern lies in the lossy information compaction of probability-based restorability estimates. For example, consider the two input AND gate of Figure 5, where the only knowledge available is that the restoration probability of value 1 (V_1) at the inputs is 0.5. A probability-based estimation scheme will infer the restoration probability of value 1 at the output to be $0.5 \times 0.5 = 0.25$. However, if the actual restored values for the two inputs over 6 successive clock cycles are $1X1X1X$ and $X1X1X1$, compatible with the estimated restoration probability, we can not restore the output for any cycle. This type of flaw is common to all probability based estimates and it results from the compaction of information over several cycles into a single measure. It could be avoided if we had a conditional probability distribution of each signal's restorability given the value of other signals, an infeasible level of accuracy in practice. In conclusion, the example shows that restoration probability estimates are not reliable, and often do not correlate well with actual restoration.

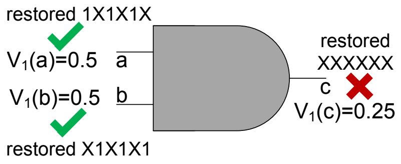


Fig. 5. **Example of a misleading Restoration Probability Estimate**.

Keeping the ideal characteristics of a restoration capacity metric in mind, we investigated whether a new metric could be constructed from the simulation of restoration itself. Indeed, a better estimate of SRR for a given group of signals and trace depth can be obtained by performing a large number of simulations while randomizing input values and the starting point for tracing; then performing the restoration process for the circuit; and finally averaging the SRR values from each individual simulation. This corresponds to estimating the SRR for the group of trace signals by Monte Carlo simulation, and unfortunately it is a very compute intensive process for typical trace buffer sizes and depths. In contrast, as we indicated earlier, individual restoration capacity estimations should be kept fairly simple, due to the large number of estimations required for a selection to converge to a final set.

A key insight in our search for an accurate SRR estimator is that the estimate of state restoration capacity metric does not need to match exactly the SRR, but only be highly correlated with it, so that it guides us to the same group of traced signals. A common method of reducing effort in simulation-based estimations is to perform several short simulations and average their outcomes. Specifically, we could use a shorter trace buffer depth. This observation led us to a study of SRR sensitivity to trace buffer depth. The results for one selection of 8 flip-flops for circuit s35932 circuit are shown in Figure 6. In the figure, we plot the SRR estimate computed over several trace buffer depths, three different random starting points of tracing and three different random input value selections per starting point. The main conclusion that can be derived from the study is that the SRR obtained from a certain group of traced signals is fairly insensitive to the trace buffer depth: indeed it can be noticed from the figure that the SRR variation is negligible beyond a trace buffer size of 64. We observed a similar behavior for all other ISCAS circuits, as well as when using a larger set of random samples. Intuitive reasoning suggests SRR is relatively insensitive to trace buffer depth beyond a certain size, since most circuits tend to stay in a small fraction of possible states, and each occurrence of such states has similar restoration behavior. We conclude then that SRR measurements over simulated restorations on small trace buffer sizes (~64) provide an accurate estimation of restoration capacity.

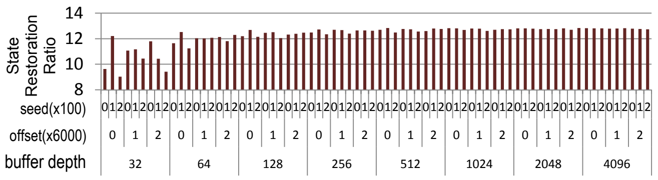


Fig. 6. **Impact of trace buffer size on SRR.** Analysis on s35932 over 3 random starting points of tracing and 3 random sets of input values per starting point indicates that SRR for a fixed set of signals is fairly insensitive to trace buffer sizes beyond 64.

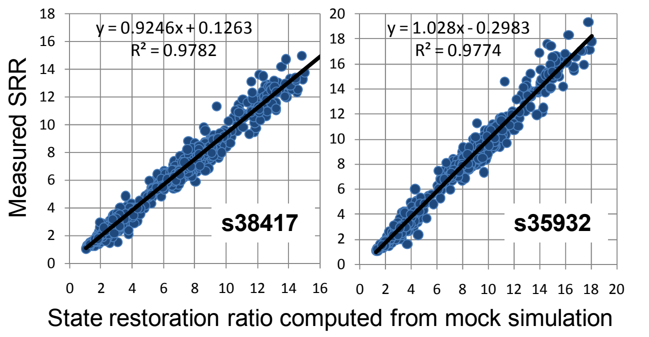


Fig. 7. **Correlation of our simulation-based restoration capacity metric with observed SRR** using a mock simulation trace depth of 64 for s38417 and s35932. The proposed metric bears strong positive correlation with the observed SRR as indicated by the high value of the correlation coefficient.

To further validate our hypothesis that short trace buffer sizes are sufficient for accurate SRR estimation we performed the previous correlation study using our new estimation metric, as shown in Figure 7. The SRR estimate is computed using a fast mock simulation with a trace buffer size of 64 and only one random set of inputs and starting time for tracing. This is the setup for estimations that we used in the rest of the paper. We conclude that the SRR measurements over simulated restorations on small trace buffer sizes (~64) provide a reliable estimate of restoration capacity. The plots of Figure 7, obtained for s38417 and s35932, clearly indicate a very high correlation between our estimation metric and the observed SRR. The simulation based capacity estimation evidently shows an extremely high degree of linear correlation with the observed SRR. Similarly strong correlations were observed for other ISCAS circuits as well. These results confirm the viability of an SRR estimator based on mock simulation of restoration over a small trace buffer size.. We expect that greater buffer sizes and averaging over more simulation with different random input values and starting points would further improve the accuracy of the estimate, although with smaller returns.

B. *Algorithm Design*

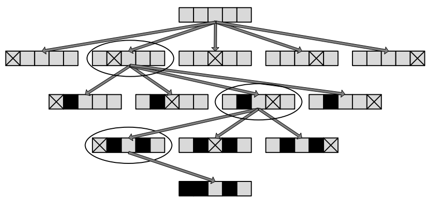


Fig. 8. **The signal selection process**. Each row corresponds to one round of the algorithm; the flip-flop (FF) whose elimination leads to maximum retention of restored states according to the estimation metric is removed in next round. The black squares correspond to FFs previously eliminated, while crosses indicate the FF being evaluated for elimination. In this example there are a total of 5 FFs and a trace buffer width of 2, so 3 FFs must be eliminated.

The problem of selecting an optimal set of flip-flops can be thought of as the problem of retaining the maximum amount of information in the unrolled circuit graph. In our algorithm we start off including all flip-flops in the circuit, which will restore almost all signals and states, and then we try to reduce this set by removing flip-flops incrementally. This process will ensure that early selections do not limit the quality of the final pool, as discussed in Section III-C. The Flip-flops that contribute least to restoration of others should be eliminated first. The process terminates when we are left with a set of flip-flops of the desired cardinality. During each step of the algorithm, we use the proposed simulation-based estimator to evaluate the restoration capacity of the candidate set of flip-flops. If elimination of two or more candidate flip-flops results in the same restoration estimate, we break the tie by comparing the total number of signals restored. If a tie still exists, then we consider the number of connected flip-flops via a forward or backward path in the circuit graph: flip-flops with fewer connections will get eliminated, if a tie still remains it will be broken by random choice.

Our algorithm is illustrated in Figure 8: the schematic represents each elimination step of the algorithm when operating on a circuit with 5 flip-flops and a target trace buffer width of 2 state elements. Note that if the initial candidate pool includes N flip-flops, $O(N^2)$ steps are required to converge to the final set. Hence, for large circuits this might be too computationally demanding. To this end, we noticed that it is common that some flip-flops are always restorable from others; hence they do not carry any additional information. We take advantage of this fact by using a fast pruning phase on a large number of flip-flops at the beginning of the algorithm, so as to reduce this size of the initial set to make the application of an $O(N^2)$ algorithm feasible. For the pruning phase, we consider the SRR estimate of each candidate set obtained by removal of one flip-flop, and then we remove multiple flip-flops in one single step, all characterized by a small contribution to restoration capacity. As shown in the pseudo-code of Figure 9, we consider all possible eliminations in sorted order of SRR estimate values (stored in $RCW[]$ vector). The flip-flops whose elimination lead to the top SRR estimate values are selected to be in the elimination set. The size of the set is a parameter called step-size d, set to 50 in our experiments. To limit the extent to which this coarse grain pruning is applied, we specify a pruning termination parameter PT such that, if the average number of restored flip-flops in the mock simulation drops below PT, the coarse grain pruning phase ends. This parameter establishes a trade-off between quality of selection and computational cost of the algorithm. In our experiments we set $PT = 95\%$.

```
Input: circuit, width of trace buffer $w$,
mock simulation based SRR estimator $f_{SRR}(...)$
Output: selected flip-flop set $T$
Parameter: step-size $d$, pruning termination parameter $PT$

while ($V >$ $PT$) {
  for(each flip-flop $s$ in $T$){
    $T = T \ \setminus \ \{s\}$
    visibility $V = f_{SRR}(T) \ \times \ |T|$
    restoration capacity without $s$ $RCW[s] = V$
    $T = T \ \cup \ \{s\}$   }
  $T = T - \{s$ | $RCW[s]$ is within top $d$ values }
  $V = f_{SRR}(T) \ \times \ |T|$ } //end of pruning

while ($|T|$ > $w$) {
  maximum visibility $maxV = 0$
  for each $s \ \in \ T${
    $T = T \ \setminus \ \{s\}$
    visibility $V = f_{SRR}(T) \ \times \ |T|$
    $T = T \ \cup \ \{s\}$
    if($V$ > $maxV$){
      $selected = s$
      $maxV = V$ } }
  $T = T \ \setminus \ \{selected\}$  }
```

Fig. 9. **Pseudo-code for our proposed algorithm.**

Adapting to Biased Selection: An important variation of the problem of state restoration was addressed in [13]. A set of critical flip-flops (S_C) are identified among the entire set of flip-flops (S_{all}) based on a user-defined criteria. For example in [13], the objective is to estimate power droop in the circuit, hence the netlist is partitioned into a coarse placement grid, and one flip-flop is selected from each grid block as a critical flip-flop. The trace selection algorithm biases the selection process so to restore the maximum number of critical flip-flops, while sacrificing restoration for non-critical flip-flops the least. Our proposed method can be adapted to tackle this problem variation by assigning appropriate weights to the critical flip-flops. We define the critical visibility V_C as the average number of critical flip-flops restored by mock simulation, the non-critical visibility V_{NC} is defined similarly. We combine these to form the total weighted visibility as $V_w = V_{NC} + (|S_{all}| - |S_C| + 1) \times V_C$. Here V_w replaces the usual visibility in the algorithm of Figure 9. The choice of weight ensures that restoration of a single critical flip-flop is a more rewarding choice than restoring all non-critical ones. However, since in each step of elimination the set providing maximum weighted visibility is retained, the algorithm will reach the optimal solution w.r.t. to our estimation metric. Results obtained with this biased selection process are discussed in Section V-D.

V. Experimental Results

We evaluated the quality of our proposed algorithm by comparing the SRR obtained on six ISCAS89 benchmark circuits, against that obtained by previous works [14], [8], [9], [10]. In addition, we present results for three control path blocks taken from the OpenSparc processor core design[15], synthesized from their RTL description. Important circuit characteristics are presented in Table I. The benchmarks are re-synthesized using Synopsys Design Compiler targeting the GTECH gate library to conform with the quality of optimization performed on industrial netlists. Note that, the synthesis tool automatically removes some redundant flip-flops in the designs under evaluation.

Circuit	# Flip-flops before synthesis	# Flip-flops after synthesis	# Gates after synthesis
s5378	179	164	1,058
s9234	211	145	920
s15850	534	524	3,619
s38584	1,426	1,426	12,560
s38417	1,636	1,564	10,564
s35932	1,728	1,728	4,981
Sparc MMU	-	262	1,977
Sparc EXU	-	327	2,168
Sparc IFU	-	2,755	19,912

TABLE I
Benchmark circuits used to evaluate our signal selection algorithm

We used an X-simulator that we developed in house to compute the simulation-based estimation metric and to measure the final SRR obtained by applying our proposed algorithm. The X-simulator takes a design along with traced values, and it restores all possible values of non-traced signals and states. We implemented our X-simulator using the efficient event-driven bit-parallel propagation technique described in [14]. All the experiments were run on a quad core Intel processor running at 2.4 GHz. The width of the bit-parallel operations in the restoration process was extended to 64 bits from 32 bits described in [14], to better utilize the 64 bit word size capabilities of the processor. This led to much better performance in the estimation phases, since the trace buffer depth was also 64 cycles.

We forced each design to operate in its normal functional mode during tracing by forcing fixed values at the relevant control inputs, including reset, while assigning random values to all other inputs. This setup is referred as "deterministic random" in several previous works [14], [10].

A. Restoration Quality

Table II compares the state restoration ratio obtained by several previous solutions against our proposed technique. As in [8], [10], the trace buffer widths used in the experiments are 8, 16 and 32, while its depth is kept at 4,096 cycles. The corresponding SRR for each solution (wherever known) is reported. The percentage improvement of SRR obtained by our proposed algorithm over the best reported value is indicated in last column. Each restoration ratio is averaged over 100 simulations, using 10 different random seeds (to generate random values at non-control primary inputs), and 10 different starting points past from the initial reset state, per seed. For certain buffer sizes, especially in smaller circuits, the SRR obtained by our solution is not better than some of the previous solutions. This is primarily due to the fact that our optimized ISCAS89 circuits have fewer flip-flops. Hence, even though our technique actually restores a higher fraction of the flip-flops, the reported SRR of

Circuit	trace width	Ko & Nicolici [14]	Liu & Xu [8]	Basu & Mishra [10]	Proposed Solution	Improv.(%) over best
s5378	8	-	*14.67*	-	*13.24*	*-9.75*
	16	-	*8.99*	-	*7.83*	*-12.93*
	32	-	*4.72*	-	*4.89*	**+3.60**
s9234	8	-	*4.76*	-	*10.68*	**+24.36**
	16	-	*7.18*	-	*7.16*	*-0.27*
	32	-	*4.67*	-	*4.18*	*-10.49*
s15850	8	-	19.93	-	39.54	**+98.39**
	16	-	24.22	-	24.85	**+2.60**
	32	-	13.30	-	13.60	**+2.25**
s38584	8	19.00	19.23	78.00	84.10	**+7.82**
	16	10.56	13.96	40.00	47.04	**+17.60**
	32	6.32	8.68	20.00	26.97	**+34.85**
s38417	8	19.62	18.63	55.00	45.21	-17.80
	16	11.22	18.62	29.00	30.77	**+6.10**
	32	6.73	14.20	16.00	20.25	**+26.56**
s35932	8	41.45	64.00	95.00	96.12	**+1.17**
	16	39.31	38.13	60.00	67.45	**+12.41**
	32	24.76	21.06	35.00	43.23	**+23.51**

TABLE II
Comparioson of state restoration ratio with no input knowledge. The table compares our solution against previous ones, computing restoration only based on traced state elements. The last column reports change over the best reported in literature.

previous solutions has the advantage of including the restoration of redundant flip-flops. For example, for s9234 at a buffer size of 32, our algorithm restores 4.18x32 = 134 (approx.) flip-flops on average, per cycle, out of total 145, which is 92% of all flip-flops, whereas the best reported solution only restores 4.67x32=149(approx.) out of 211 flip-flops, corresponding to 70%. For larger circuits, which better represent practical post-silicon debug situations, our solution achieves an improvement of up to 34.85% (for s38584) in the SRR.

Circuit	trace width		
	8	16	32
Sparc MMU	12.22	8.03	4.67
Sparc EXU	4.53	3.46	4.02
Sparc IFU	99.10	62.01	35.67

TABLE III
SRR for OpenSparc blocks, using only traced state elements.

We report the SRR obtained for the OpenSparc blocks in Table III. The primary inputs were driven by the trace recorded during the execution of a functional test from the OpenSparc regression suite. The trace buffer depth is kept at 4,096 cycles for these designs as well.

Circuit	trace width	Prabhakar & Hsiao [9]	Basu & Mishra [10]	Proposed Solution	Improv.(%) over best
s5378	8	19.30	19.00	20.25	**+6.58**
	16	9.70	9.90	10.21	**+3.13**
	32	4.84	5.00	5.12	**+2.40**
s9234	8	*20.30*	*23.30*	*14.34*	*-38.45*
	16	*10.30*	*11.80*	*7.80*	*-33.89*
	32	*5.20*	*6.00*	*4.21*	*-29.83*
s15850	8	55.60	55.10	55.89	**+1.43**
	16	27.80	29.80	31.01	**+4.06**
	32	13.90	15.80	16.36	**+3.54**
s38584	8	130.10	151.20	176.84	**+16.95**
	16	66.02	78.40	88.47	**+12.84**
	32	34.80	40.50	44.32	**+9.43**
s35932	8	209.60	209.40	215.94	**+3.12**
	16	104.80	105.80	107.97	**+2.05**
	32	52.40	53.30	53.98	**+1.27**

TABLE IV
SRR leveraging input knowledge. Comparison of SRR computed using both state tracing and input knowledge. The last column represents percentage change over the best reported in literature.

We also compare the restoration quality of our approach to [9], when all the primary input values are known at every clock cycle during the traced interval. Though this assumption is not as realistic, since in a real IC design the circuit blocks under study will probably be embedded within a larger design, still, for sake of completeness, we compare the performance of our algorithm versus [9] and [10]. Note that in this case almost 100% flip-flops are restored by previous algorithms, so the scope of improvement is very limited. The results are presented in Table IV, the reported SRR for the proposed algorithm is averaged over 100 simulations as before. We observe better restoration ratio than both previous solutions for all circuits, except s9234. This anomaly is due to a smaller number of flip-flops in our circuits due to synthesis optimization. Indeed our solution restores an even higher fraction of the state elements than [9].

B. *Effect of Pruning*

We studied the effect of the pruning optimization (discussed in Section IV-B) in our elimination-based algorithm. The effect of pruning is shown in Figure 10. This data corresponds to execution of the proposed algorithm for circuit s15850, when the $f_{SRR}()$ metric is based on a simulation with a trace buffer depth of 32 (instead of the usual 64, for purposes of visible fine granularity), and a trace buffer width also of 32. Hence, the algorithm terminates when the traced set reaches 32. A total of 524x32=16,768 flip-flop values (s15850 has 524 flip-flops, refer Table I) are present in the simulation window for the estimator metric. The y-axis plots the value of $f_{SRR}(T) \times |T| \times 32$ during each iteration in the execution of our signal selection algorithm. Note that the no-pruning line is smooth as only one flip-flop is removed per iteration, and the total number of restored flip-flops in the mock simulation gradually decreases. On the other hand, pruning uses a step-size(d) of 50 flip-flops; hence, during the pruning phase, the total number of restored flip-flops drops as a step function. In this example pruning termination (PT) was set at 93% *i.e.* 16,768x0.93=15,594, a value by which the set is reduced to a size of approximately 200. Note that the quality of pruning is only slightly worse than the exact version (the with-pruning line ends slightly lower than the no-pruning line). Thus, pruning trades-off some accuracy for faster execution.

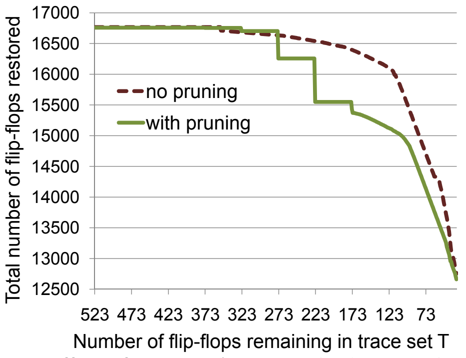


Fig. 10. **The effect of the pruning phase** in the trace signal selection algorithm for s15850.

C. *Return on additional traced signals*

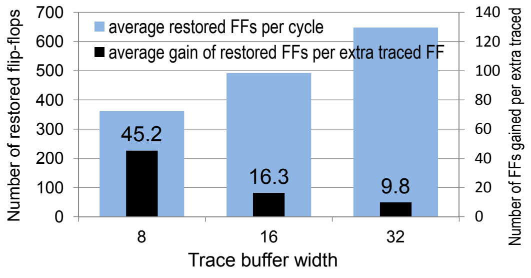


Fig. 11. **Restored flip-flops vs. trace buffer size for circuit s38417**. A moderately steady rate of increase of the number of restored flip-flops with increasing trace-buffer size is observed for our proposed solution.

Diminishing gain with additional traced flip-flops was pointed out as a shortcoming of the greedy algorithms in Section III-C. Our proposed algorithm alleviates this issue to a large extent. Figure 11 plots the same information as Figure 3, but using our algorithm. It can be noticed that we restore on average more flip-flops than previous solutions for buffer sizes of 16 and 32. Moreover, far more steady gain in the number of restored flip-flops per additional traced signal is observed, compared to Basu & Mishra [10], the best previous solution so far in terms of total restoration. Similar trends are observed for other benchmarks as well.

D. Restoration Quality for Biased Selection

We compare the restoration quality with biased selection against that of the pareto-optimal biased selection described in [13] and using the same experiment. The circuit is partitioned into a coarse 4x4 grid and one flip-flop per partition is chosen as a critical flip-flop, leading to a critical set of 16. The critical flip-flop of a partition is defined as the flip-flop that, when traced alone, leads to the maximum restoration for the partition. Two trace buffer widths (16 and 32) are used for the evaluation. We also use the same quality metric as in [13], namely the number of flip-flops fully or partially restored in total and from the critical subset. Table V reports the obtained results. Note that our technique achieves restoration of more non-critical flip-flops while restoring the same number of critical flip-flops in almost all cases. Also there is a sharper increase in the number of restored non-critical flip-flops compared to [13], when the trace-buffer size is doubled. It is important to note that this is not a violation of pareto-optimality of the selection in [13], since pareto-optimality is maintained assuming perfect linear correlation between the estimation metric and the actual SRR, an assumption that does not hold, as we have shown.

Circuit	trace width	Shojaei et al.[13]		proposed		non-critical
		total	critical	total	critical	gain (%)
s5378	16	128	16	125	16	-2.6
	32	146	16	156	16	**+7.7**
s9234	16	64	11	82	11	**+33.9**
	32	89	15	105	15	**+21.6**
s15850	16	137	16	172	16	**+28.9**
	32	137	16	230	16	**+76.8**
s38584	16	211	16	254	16	**+22.0**
	32	210	16	361	16	**+77.8**
s38417	16	210	15	278	16	**+34.9**
	32	312	16	356	16	**+17.9**
s35932	16	313	16	389	16	**+25.5**
	32	377	16	464	16	**+24.1**

TABLE V
Results for biased selection indicates restoration of more non-critical flip-flops and a much sharper increase of restoration when trace width is doubled.

E. Algorithm Execution Performance

Circuit	trace width	Ko & Nicolici [14]	Liu & Xu [8]	Basu & Mishra [10]	Proposed Solution
s5378	8	-	14	-	656
	16	-	36	-	634
	32	-	75	-	600
s9234	8	-	26	-	456
	16	-	75	-	441
	32	-	148	-	433
s15850	8	-	298	-	3,877
	16	-	764	-	3,823
	32	-	1,656	-	3,781
s38584	8	34,440	388	1,200	18,143
	16	73,500	802	2,600	18,091
	32	149,580	2,826	5,500	18,003
s38417	8	28,200	2,319	2,200	24,943
	16	69,060	5,285	4,500	24,819
	32	149,940	11,732	9,100	24,734
s35932	8	31,440	1,407	2,200	19,857
	16	68,700	5,251	4,400	19,832
	32	142,800	10,496	8,900	19,801

TABLE VI
Comparison of execution performance for the algorithms considered. All execution times are reported in seconds.

Trace signal selection is performed only once during the design phase of the circuit blocks to be included in the signal list for the ELA. Hence, the run-time of the selection algorithms is less important than the quality of the selected signals. However, if an inordinate amount of time is needed for even moderately sized circuit blocks, performance would be an issue. In our algorithm, the pruning phase was designed specifically for this reason. A comparison of the execution time of previous solutions and our solution is presented in Table VI. Note that, the execution performance of the proposed algorithm is often worse for small designs, this is due to the large number of simulations needed in our algorithm. However, these simulations are for computation of the estimation metric, and they are independent of each other during each iteration of the selection algorithm. A possible way to improve the algorithm's performance, if necessary, is to leverage the pattern parallelism of GPU platforms, where the same execution is applied on different data sets.

Acceleration of the Selection Algorithm: We implemented a parallel version of the X-simulation kernel on a GPU platform. The parallel version which performs the $|T|$ independent simulations, required for every step of the elimination algorithm, concurrently. We use an NVIDIA GTX 480 GPU as the execution platform. Each distinct thread-block performs the X-simulation using a different traced flip-flop set. The main restoration algorithm is also modified in order to fit single instruction multiple thread execution paradigm used by GPUs. Execution times corresponding to trace buffer width of 32 were improved by a factor of 2.69, 3.08 and 3.05 times for s38584, s38417 and s35932, respectively. This implementation leads to an overall performance comparable to that of previous solutions, even in light of a much more accurate estimation metric.

VI. CONCLUSION

In this work, we have presented a trace signal selection algorithm that strives to maximize state restoration ratio. Our algorithm is guided by a more accurate simulation based restoration capacity metric and achieves better state restoration ratio than previous solutions. It also achieves better restoration trends per additional traced signal while restoring a higher number of states on average.

ACKNOWLEDGMENTS

This work was developed with partial support from the Gigascale Systems Research Center.

REFERENCES

[1] N. Nataraj, T. Lundquist, and K. Shah, "Fault localization using time resolved photon emission and STIL waveforms," in *Proc. ITC*, 2003, pp. 254 – 263.
[2] B. Vermeulen, T. Waayers, and S. Bakker, "IEEE 1149.1-compliant access architecture for multiple core debug on digital system chips," in *Proc. ITC*, 2002, pp. 55 – 63.
[3] M. Abramovici, P. Bradley, K. Dwarakanath, P. Levin, G. Memmi, and D. Miller, "A reconfigurable design-for-debug infrastructure for SoCs," in *Proc. DAC*, 2006, pp. 7–12.
[4] *SignalTap II Embedded Logic Analyzer*, Altera Verification Tool, 2006, http://www.altera.com/products/software/products/quartus2/verification/signaltap2/sig-index.html.
[5] *ChipScope Pro*, Xilinx Verification Tool, 2006, http://www.xilinx.com/ise/optional_prod/cspro.html.
[6] *Embedded Trace Macrocells*, ARM limited, 2007, http://www.arm.com/products/solutions/ETM.html.
[7] H. F. Ko and N. Nicolici, "Automated trace signals identification and state restoration for improving observability in post-silicon validation," in *Proc. DATE*, 2008, pp. 1298–1303.
[8] X. Liu and Q. Xu, "Trace signal selection for visibility enhancement in post-silicon validation," in *Proc. DATE*, 2009, pp. 1338–1343.
[9] S. Prabhakar and M. Hsiao, "Using non-trivial logic implications for trace buffer-based silicon debug," in *Proc. ATS*, 2009, pp. 131–136.
[10] K. Basu and P. Mishra, "Efficient trace signal selection for post silicon validation and debug," in *Proc. VLSI design*, 2011, pp. 352–357.
[11] Y.-C. Hsu, F. Tsai, W. Jong, and Y.-T. Chang, "Visibility enhancement for silicon debug," in *Proc. DAC*, 2006, pp. 13–18.
[12] J.-S. Yang and N. A. Touba, "Automated selection of signals to observe for efficient silicon debug," in *Proc. VTS*, 2009, pp. 79–84.
[13] H. Shojaei and A. Davoodi, "Trace signal selection to enhance timing and logic visibility in post-silicon validation," in *Proc. ICCAD*, 2010, pp. 168–172.
[14] H. F. Ko and N. Nicolici, "Algorithms for state restoration and trace-signal selection for data acquisition in silicon debug," *IEEE Trans. on CAD*, vol. 28, no. 2, pp. 285–297, 2009.
[15] "Sun Microsystems OpenSPARC," http://opensparc.net/.