Simulation-based Signal Selection for State Restoration in Silicon Debug

Debapriya Chatterjee, Valeria Bertacco
Department of Computer Science and Engineering, University of Michigan
{dchatt, 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 from pre-silicon verification. The most critical roadblock in post-silicon validation is the limited observability of the 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, designers can afford to trace only a very small fraction of a design's signals. Thus, the pre-silicon phase 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 automatic 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 to produce a trace signal selection. In this work we propose a more accurate restoration capacity metric, based on simulation information, and also present a novel algorithm that overcomes some of the key shortcomings of previous solutions. In our experimental evaluation, we show that our technique provides up to 34 % better state restoration compared to all previous techniques on the more complex benchmark designs while providing much better restoration trend with increasing trace buffer size.

1. INTRODUCTION

Shrinking transistor size with each new generation of digital integrated circuits (IC) has 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 avoid the escalating cost of many re-spins and to meet stringent time-to-market requirements, these design bugs escaped to silicon must be detected and root-caused in the first available silicon. To facilitate the task of detecting and investigating 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 [9] 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 [12]. Though scan chains can capture all or a subset of internal state elements of a design, 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 [1] and have found widespread use in the industry [2, 13, 3]. 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 [13] and data acquisition can be performed concurrently with 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, however, 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. By 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 of the traced data. This approach cannot be used in a general circuit, however, because it leverages architectural knowledge of the design. Indeed, there is a growing need for automated solutions in this domain that can be applied to general circuit structures. Even though the additional inferred information does not guarantee identification of design errors, it still increases internal signal visibility and has the potential of providing valuable debugging information. Indeed, bugs tend to occur in unexpected regions and configurations and 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 in a design from tracing of just a handful of signals, so as to offer pre-silicon quality observability during post-silicon debug.

Recent research addressing these challenges [6] 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 [6] were also 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 for a given time interval. This measure has been adopted by subsequent research in this area to compare the quality of other solutions. Further research [8, 10, 4] 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 restoration ratio. We also propose a novel signal selection method guided by this metric. Previous greedy selection methods suffer from the shortcoming of diminishing returns: when increasing the number of traced signals, the number of additional restored state elements increase sluggishly. Our solution overcomes this shortcoming by offering better restoration trend with increasing number of traced signals.

1.1 Contributions

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

- We show that computing the state restoration ratio by simulation of the design over 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 suggest 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 increases.
- Experiments show that our solution provides up to 34 % better state restoration ratio compared to all previous solutions.

2. 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 [5] considered only the reconstruction of data at the combinational logic nodes of the circuit. Ko and Nicolici [6] 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, 10, 4], 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 [6] 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 in the circuit can be generated. This probabilistic model can leverage 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 [6] lacks theoretical basis and it is then improved on in [8]. In contrast, [4] considers only the restoration probability along the direct paths among 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 direct path probability values accordingly. Another solution described in [10] estimates the visibility of non-traced nodes by non-trivial logic implications of flip-flop values. However [10] 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 uses simulation for estimation instead of a probabilistic metric.

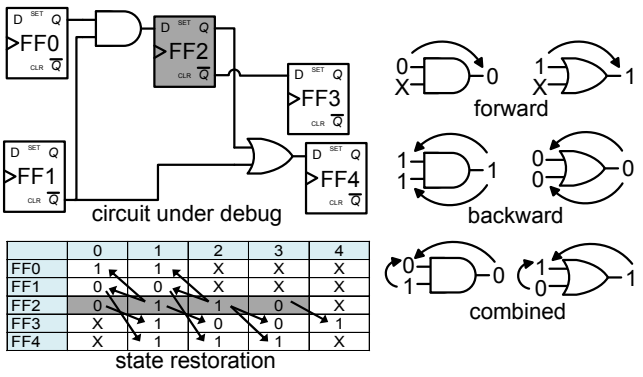
Another line of research [14, 11] suggests that not all state elements or signals are equally relevant for debugging purposes. Hence, instead of striving to maximize the state restoration ratio, they attempt to perform trace signal selection to maximize the restorability of a certain subset of critical flip-flops or signals. In specific [11] describes an algorithm for trace signal selection with the goal of enhancing restorability of a fixed subset of flip-flops while minimizing the observability impact to other flip-flops. The algorithm 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.

3. BACKGROUND AND MOTIVATION

An ideal post-silicon debugging solution would allow pre-silicon level 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.

3.1 State Restoration Process

The state restoration process relies on the special Boolean 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 the case of partial knowledge. Similarly, if a non-controlled value is observed on the output of a gate, all input values can be inferred to be the non-controlling value for that type of gate, enabling backward justification. Combined inferences leveraging knowledge of both inputs and output are also possible. Repeated application of these simple operations for all gates of a circuit till no new value can be generated at any signal leads to value reconstruction for state elements beside those traced. This process is used in post-analysis of the data obtained from trace-buffers to restore other non-traced signals. Figure 1 illustrates this process with an example inspired by [6]. In this example flip-flop FF2 is traced over four clock cycles; additional values at



	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

Figure 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.

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) is $SRR = 15/4 = 3.75$ ($N_{traced} = 4, N_{restored} = 11$). [6] introduces an efficient bit-parallel algorithm to perform this restoration process, which we extensively use 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.

3.2 Structure of Signal Selection Algorithms

The signal selection algorithms presented in the literature so far [7, 8, 10, 4] focus on delivering maximal restoration ratio and share a common structure. First, a metric is devised to estimate the capacity of state restoration of a given set of signals; second, a greedy selection process guided by the metric converges 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 ∪ {s}
    Visibility V = f_C(T)
    T = T − {s}
    if(V > maxV){
      selected = s
      maxV = V
    } }
  T = T ∪ {selected}
}
```

Figure 2: Pseudo-code for the general structure of greedy automatic signal 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 with one more signal. In Section 4.1 we will explore how a better capacity metric can be obtained by simulated restoration, while a critical shortcoming of the greedy selection process itself is detailed in next section.

3.3 The Problem of Diminishing Return with Greedy Selection

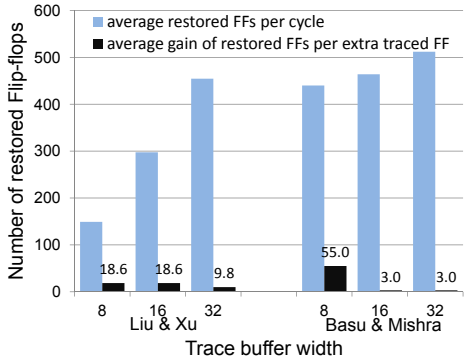


Figure 3: Diminishing return of number of restored flip-flops with increasing trace buffer size is observed for two previous solutions. The plots are corresponding to circuit s38417.

The greedy selection process adopted in the previous solutions suffer from another critical problem with regards to the quality of the final set of signals chosen. Figure 3 plots the average number of restored flip-flops per cycle for 3 different width of the trace buffer (8,16,32) for the ISCAS89 benchmark circuit s38417. Alongside the average number of restored flip-flops gained by addition of each new traced flip-flop is plotted as well. The plots correspond to the data reported by Liu and Xu [8] and by Basu and Mishra [4]. Note that in the result obtained by Liu and Xu, growing the number of observed flip-flops from 8 to 16 increases the average number of restored flip-flops per cycle, from 149 to 298, which is a good rate ($(298 - 149)/(16 - 8) = 18.62$) of gain of information per added new trace signal, shown in the adjacent dark bar. However when the number of traced signals is increased from 16 to 32, the rate of gain is much lower. This effect is more pronounced in the results obtained by Basu and Mishra [4], where a much better initial set of signals is obtained but as the number of trace signals are doubled, the gain in the average number of restored flip-flops is very minute. This behavior results from inaccuracy in the estimation metric and due to the very nature of the greedy selection. The greedy selection algorithm starts off with the flip-flop promising maximum restoration and attempts to grow the set by one flip-flop at a time, and the average number of restored flip-flops plateaus off when a larger number of flip-flops are traced. When choosing $2n$ flip-flops, the choice is already constrained by previously chosen n flip-flops: We have to keep the n chosen flip-flops in the set and find additional flip-flops which when added with the existing set provides maximum restoration possible under this constraint. However the best possible set of $2n$ flip-flops might not have all the n flip-flops, since there might be other $n + 1$ or more flip-flops which when taken together are able to restore more missing signals, but would not be able to enter the

final selection, since the algorithm only makes greedy choices in the forward direction trying to grow a pre-decided set of n flip-flops. Hence for choosing a larger number of traced signals an alternative approach of making greedy decisions from the backward direction, i.e. starting off with the set of all flip-flops and then constraining the set slowly to the required width, can be more successful. We outline an algorithm to perform this elimination process.

4. SIGNAL SELECTION ALGORITHM

First we derive a more accurate restoration capacity metric and then use this metric in our proposed alternative algorithm. The key factor in deciding whether a metric is good, lies in the correlation it possesses with actual measured SRR.

4.1 Improving Restoration Capacity Metric

As mentioned earlier, a good restoration capacity metric should possess high degree of correlation with the actual observed SRR obtained with a set of signals. Since, the more accurate the metric, the more likely it is to arrive at the optimal subset of signals at the end of selection process. To evaluate the quality of a restoration capacity metric, we devise the following experiment. For a design we choose 1000 random sets of 8 flip-flops each and measure the average SRR per group, for a trace buffer depth of 4096, obtained with 100 simulation runs (using 10 sets of random seeds and 10 different starting point of tracing i.e. offset from the initial circuit reset state, per seed). It is ensured that the circuit remains in functional mode during the entire tracing process, by asserting appropriate value at reset and other control signals. We can now plot the average SRR versus the estimated state visibility obtained with a restoration capacity estimation metric in a scatter plot to measure the correlation of the metric with actual measured SRR.

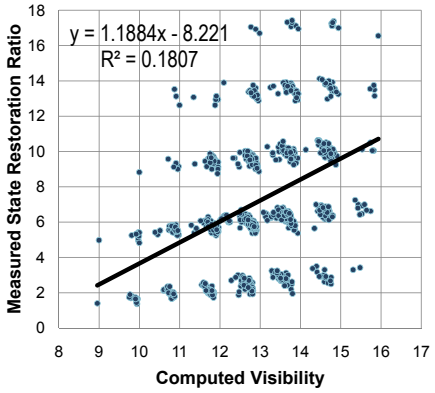


Figure 4: Correlation of restoration capacity metric described by Liu and Xu with measured SRR for circuit s35932. The metric has poor yet positive correlation with measured SRR. Note that data points in the bottom right corner represents selection of flip-flops that have a high estimated value of state visibility but rather poor measured SRR. This behavior can drive the greedy selection algorithm to sub-optimal selections. A linear regression fit of the data is shown in the plot, along with square of the correlation coefficient.

We implemented the restoration capacity metric called visibility V, described by Liu and Xu [8]. Figure 4 shows the correlation of this metric with observed SRR. As seen in the figure, though this metric has positive correlation with measured SRR, the extent of correlation is poor; as indicated by a low value of the correlation coefficient(R). Also this metric can over-estimate as well as under-estimate the SRR of certain selections leading to a sub-optimal final selection. The fundamental reason behind this behavior is lossy information compaction in probability based restorability estimates. Consider the two input AND gate in Figure 5, where the restoration probability of value 1 at the both inputs are known to be 0.5 and no other knowledge is present. 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 value in the two signals over 6 successive clock cycles are $1X1X1X$ and $X1X1X1$, both in accordance with the estimated restoration probability, though we can not restore the output for any of the cycles. This flaw is common to all probability based estimates and the inaccuracy results from compaction of information that is spread across several cycles into a single number, and could be avoided if we had a conditional probability distribution of each signal's restorability given the value of other signals. However such detailed probabilistic treatment is infeasible. This example shows that the restoration probability estimates are not reliable, and often do not correlate well with actual restoration behavior.

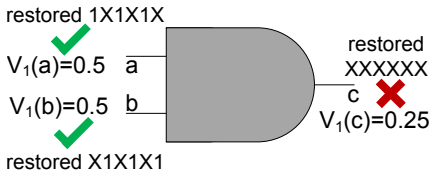


Figure 5: Restoration probability estimates can be misleading, as seen in this example.

Keeping the ideal characteristics of a restoration capacity metric in mind, we investigated whether a metric of restoration capacity can be constructed out of simulation of restoration itself. The best estimate of SRR for a group of traced signals and trace depth in a circuit can be obtained by performing a large number of simulations with different random seeds (for generating inputs) and starting tracing at several random offsets from the initial reset state, then performing the restoration process for the circuit, finally taking the average of the SRR values from each individual simulation. This is effectively analogous to performing Monte-Carlo simulations for obtaining an estimate of SRR for a group of traced signals. However, even though this estimate would be extremely accurate, each of the individual simulations (also includes the restoration process per simulation) takes up a considerable amount of execution time when performed for typical trace buffer depth ($\sim$4K clock cycles) and also several such simulations will be needed to establish a single estimate. This violates the second criterion of an ideal capacity estimation metric. A selection algorithm will need a large number of such estimates to converge on to the final set of signals, hence if each of the individual estimations are computationally intensive the overall selection process would demand an inordinate amount of time for any realistic circuit size.

A key insight to solve this problem is the fact that the estimate of state restoration capacity does not need to exactly match observed SRR, it only has to be highly correlated with the actual SRR that can be obtained with the same group of traced signals. A common method of reducing effort in simulation based estimation is to perform several short simulations and average their results. In this particular case which amounts to performing the state restoration process but for a smaller length of the trace buffer. This observation lead us to carry out a study about sensitivity of SRR on varying depth of the trace buffer. The results for a certain selection of 8 flip-flops in s35932 circuit is shown in Figure 6. For purpose of legible representation only 9 random samples per trace buffer depth are displayed: 3 different random offsets and 3 random seeds per offset. The main observation from this study is that the value of the SRR obtained from a certain group of traced signals is fairly insensitive to depth of the trace buffer. In fact, there is very little variation beyond the depth of 64 cycles. Similar behavior is observed for all other circuits, as well as when more random samples are obtained. This observation suggests that measured SRR from simulated restoration for small depths

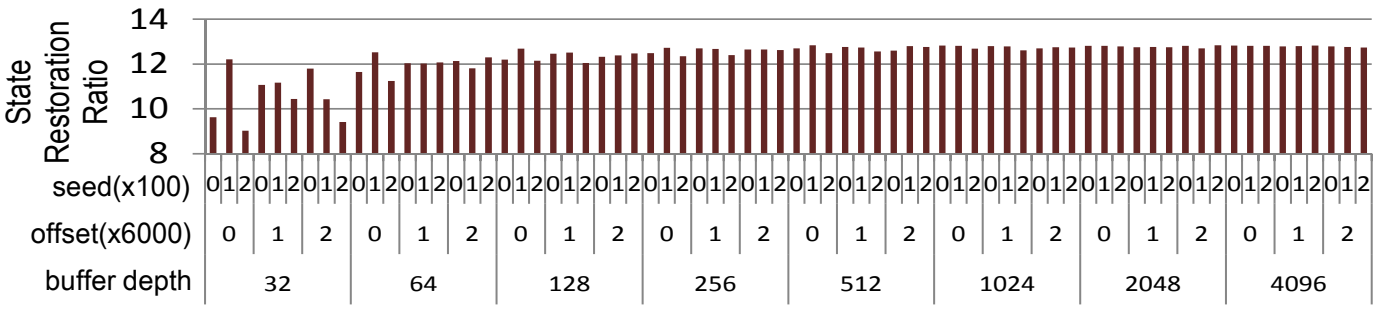


Figure 6: Variation of SRR with trace buffer depth (3 random offsets per case, 3 random simulation seeds per offset for the s35932 circuit). The value of the observed SRR for a group of signals is fairly insensitive to buffer depth beyond 64.

(∼64) can serve as an estimation metric of restoration capacity.

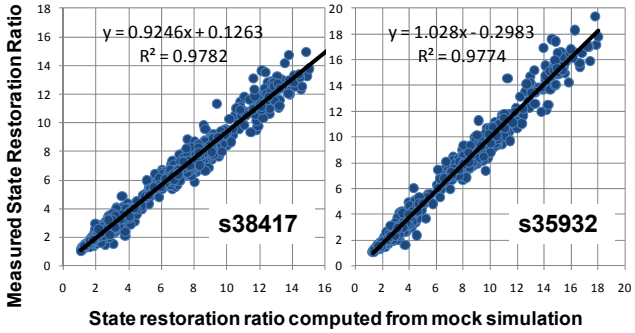


Figure 7: Correlation of observed SRR with our proposed restoration capacity metric namely, SRR obtained from mock simulation with 64 cycle of buffer depth. Correlation is shown for two circuits: s38417 and s35932. The proposed metric bears strong positive correlation with the observed SRR indicated by the value of the correlation coefficient.

The hypothesis that SRR obtained from mock simulated restoration for small depths has good correlation with the observed SRR is further validated by repeating the earlier correlation study except for plotting the simulation based metric on the X axis in this case for two benchmark circuits. The mock simulation uses depth of only 64 cycles with one random seed and one random offset, a convention used for all estimation purposes described in the rest of the paper. The resultant scatter plot for circuits s38417 and s35932 is shown in Figure 7. The simulation based capacity estimation evidently shows an extremely high degree of linear correlation with the observed SRR. Similar strong correlation was found for other circuits as well. This observation confirms the viability of using SRR obtained from mock simulation of restoration for a small depth as an accurate estimate of restorability of state elements. Note that, a larger depth and averaging over more random seeds and offset values will make the estimate even more accurate and should be deployed if more compute resources are available.

4.2 Algorithm Details

The problem of selecting the optimal set of flip-flops can be viewed as a problem of retaining the maximum amount of information in the unrolled circuit graph. We start off with all flip-flops in the circuit (which will restore almost all signals and states), and then we try to constrain this set by removing flip-flops. This will ensure that we do not get constrained by our sub-selections when selecting a larger set of trace signals as pointed out in Section 3.3. The flip-flops whose knowledge contribute least to restoring others should get eliminated earlier. When all but the desired number of flip-flops are eliminated, this process terminates. We use the previously proposed simulation based metric, as an estimate of the information retained by the remaining set of flip-flops. If elimination of two or more candidate flip-flops result in same amount of state restoration in mock simulation, we break the tie by comparing total number of signals restored. If a tie still exists then it is broken by considering the number of other flip-flops, that the candidate flip-flop is connected with via a forward or backward path in the circuit graph. The flip-flop with less connections will get eliminated, if a tie still remains it will be broken by random choice.

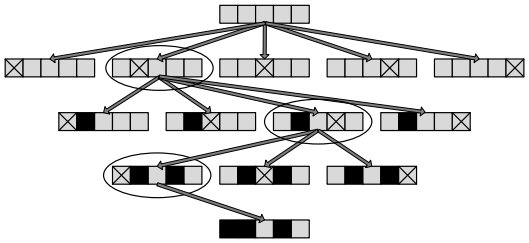


Figure 8: The flip-flop selection process . The flip-flop whose elimination leads to maximum retention of restored states according to the estimation metric is decided to be removed in next round. The blackened out flip-flops has been already eliminated, while we have to try out all elimination possibilities(shown by crossed) before deciding upon the next elimination. In this example the trace buffer width is 2, so 2 flip-flops are selected out of 5.

This method is shown in Figure 8. Note that if we start with N flip-flops, it takes $O(N^2)$ steps to converge at the final set. Hence, for large circuits the procedure might become very computationally demanding. We noticed that in typical circuits some flip-flops are always restorable from the knowledge of other flip-flops and hence they do not carry any information. We take advantage of this by performing a fast pruning on a large number of flip-flops, to reduce this size of the set to an extent that application of an $O(N^2)$ algorithm will be feasible. To perform this pruning, we consider the SRR estimate of each possible set by removal of one flip-flop, however instead of only removing the flip-flop whose elimination leads to maximum estimated SRR and repeating the process, we remove a set of flip-flops which have poor information content, in one step. We consider all possible eliminations in sorted order of SRR estimate values (as $RCW[]$ in 9). The flip-flops whose elimination lead to the top few SRR estimate values are the candidates to be in the final elimination set. The size of the set is a parameter called step-size d. For our experiments this parameter was set as 50. To limit the extent to which this coarse grain pruning is done on a circuit, we can specify a pruning termination parameter PT such that if the average number of restored flip-flops in the mock simulation drops below that value, the coarse grain pruning will stop and the actual elimination algorithm will work on the residual set. This parameter can create a trade-off

```
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

The set of all flip-flops in the circuit S
Current visibility V = f_SRR(S) × |S|
Start with all flip-flops T = S

while (V > PT) {
  for(each flip-flop s in T){
    T = T − {s}
    Visibility V = f_SRR(T) × |T|
    Restoration capacity without s RCW[s] = V
    T = T ∪ {s}
  }
  T = T−{s | RCW[s] is within top d values }
  V = f_SRR(T) × |T|
} //end of pruning

while (|T| > w) {
  Maximum Visibility maxV = 0
  for (each s in T){
    T = T − {s}
    Visibility V = f_SRR(T) × |T|
    T = T ∪ {s}
    if(V > maxV){
      selected = s
      maxV = V
    } }
  T = T − {selected}
}
```

Figure 9: Pseudo-code for the final algorithm.

between quality of selection and the execution performance of the algorithm. It was chosen as 95 percent of the total number of flip-flops in the circuit to assure good quality of signal selection for our experiments. The final algorithm is illustrated in Figure 9.

5. EXPERIMENTAL RESULTS

We evaluate the quality of the trace signals selected by the proposed algorithm by comparing SRR obtained on six ISCAS89 benchmark circuits, which were used in previous works that strive to maximize restoration [7, 8, 10, 4]. The number of flip-flops in the circuits and other circuit characteristics are presented in Table 1. The benchmarks are re-synthesized using Synopsys Design Compiler targeting the GTECH gate library, to conform with the quality of optimization performed on netlists used in industry currently (re-synthesis is performed in[11] as well). Note that, some redundant flip-flops in these designs are removed by the synthesis tool.

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

Table 1: Benchmark circuits used to evaluate proposed signal selection algorithm

The X-simulator which restores the value of non-traced signals and states forms an integral part of our solution since it is used to compute the estimation metric through mock simulations, as well as for measuring SRR attained by the algorithm. The 3-input or larger gates are internally de-composed into elementary 2-input gates in the X-simulator for efficient computation, a transformation that has no other consequence since the trace signals are only flip-flop values. We implemented our X-simulator using the efficient event-driven bit-parallel forward and backward propagation technique described in [7]. 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 the 32 bits described in the original, to utilize the 64 bit word size of the processor, which greatly increases the performance of individual mock simulations, performed for a depth of 64 cycles.

During the tracing operation each circuit was kept in the functional mode, by keeping global reset signals de-asserted and forcing fixed values at other control inputs while feeding random values at other primary inputs. This input restriction is referred as "deterministic random" in several previous works [7, 4]. This restriction at the inputs is very important to evaluate the quality of trace signal selection. If control inputs are allowed to toggle, the circuit might intermittently enter the reset state and the reset signal itself might be traced, leading to a large amount of state restoration. However, during debug this scenario is unlikely to happen and the circuit will remain in the functional mode most of the time, so the state restoration ratio obtained when control signals are allowed to toggle is not representative of actual restoration capacity of the trace signals. This issue has been pointed out in [7, 8]. All our experimental results correspond to the circuit operation in functional mode, and all the mock simulation estimates are also obtained under this constraint.

5.1 Restoration Quality

Circuit	trace width	Ko & Nicolici [7]	Liu & Xu [8]	Basu & Mishra [4]	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 2: State restoration ratio without input knowledge for ISCAS89 circuits. Only traced state elements are used for restoration. SRR obtained by previous solutions which only use the knowledge of traced signals are presented for comparison. The last column represents percentage change over the best reported in literature.

Table 2 compares the state restoration ratio obtained by several previous solutions with our proposed technique on the ISCAS89 benchmarks. As in [8, 4], the trace buffer widths used in the experiments are 8,16 and 32, while the depth is kept at 4096 cycles and corresponding SRR for each solution (wherever known) is reported. The percentage improvement of SRR obtained by the proposed algorithm over the best reported value is reported in last column. Each reported restoration ratio for the proposed algorithm is the average over 100 simulations, with 10 different seeds (to generate random values at non-control primary inputs), and 10 different cycle offsets from the initial reset state, per seed. For certain buffer sizes, especially in the case of smaller sized ISCAS89 circuits the SRR obtained by our solution is less than that of the best reported. This anomalous behavior is primarily caused by the fact that the optimized ISCAS89 circuits have a reduced number of flip-flops. Hence, even though our technique actually restores higher percentage of flip-flops on average per cycle the reported SRR of previous solutions is often boosted by restoration of the redundant flip-flops. As an example, for buffer

size of 32 in the case of s9234 circuit, our algorithm restores 4.18x32 = 134(approx.) flip-flops on average per cycle out of 145, which is 92 percent of all flip-flops, where as the best reported solution only restores 4.67x32=149(approx.) out of 211, which is only about 70 percent. For the larger circuits, which are better representative of the cases encountered in post-silicon debug, our solution achieves up to 34.85 percent (for s38584) better state restoration ratio.

Circuit	trace width	Prabhakar & Hsiao [10]	Basu & Mishra [4]	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 3: State restoration ratio with input knowledge for ISCAS89 circuits. SRR obtained by previous solutions which also use inputs are presented for comparison. The last column represents percentage change over the best reported in literature.

We also compare restoration quality of our approach to [10], where it is assumed that all the primary input values are known at every clock cycle along with the traced flip-flops. Though this assumption is not realistic, since in a real IC design the circuit blocks under study will be embedded inside a larger design, hence the inputs to the circuit block will also need a trace buffer. However for the sake of completeness, we compare the performance of our algorithm versus [10], as it was done by [4] as well. Note that in this case almost 100 percent flip-flops are restored by previous algorithms, so the scope of improvement is very limited. The results are presented in Table 3, the reported SRR for the proposed algorithm is averaged over 100 simulations as described before. We observe better restoration ratio than both previous solutions for all circuits except s9234. This anomalous behavior is again caused by the same reason of different number of flip-flops, as our algorithm actually restores even closer to 100 percent flip-flops.

5.2 Effect of Pruning

We studied the effect of the pruning optimization (discussed in Section 4.2) on top of 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 using a mock simulation of depth 32 (instead of usual 64, for purposes of visible fine granularity), and the trace buffer width is set at 32. Hence the algorithm terminates at trace set size of 32. A total of 524x32=16768 flip-flop values are present in the window of mock simulation (s15850 has 524 flip-flops refer Table 1). The y-axis effectively 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 total number of restored flip-flops drop as a step function at each 50 interval. In this example pruning termination(PT) was set at 93 percent of all flip-flop values i.e. 16768x0.93=15594, by which point the whole set of 524 has already been reduced to around 200. Note that the pruning produces only slightly lesser quality signal selection than exact version, as the with-pruning line ends slightly lower than the no-pruning line. Thus pruning sacrifices accuracy to a small degree for faster execution of signal selection algorithm.

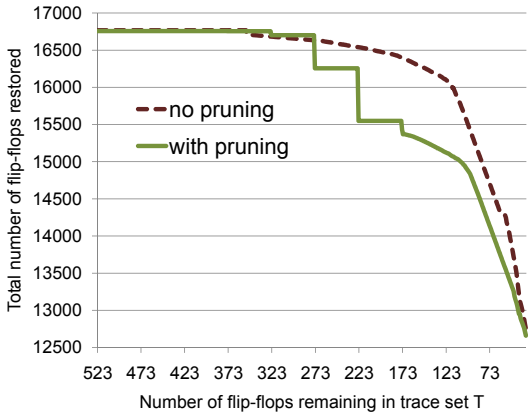


Figure 10: The effect of pruning during execution of trace signal selection algorithm is shown for circuit s15850.

5.3 Gain per Additional Trace Signal

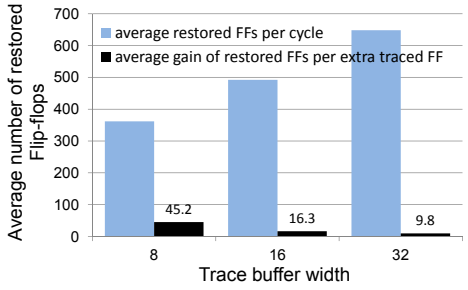


Figure 11: Restored flip-flops vs. trace buffer size for circuit s38417. A moderately steady rate of increase of the number of restored flip-flops with increasing trace-buffer size is observed for the proposed solution. While it is also able to restore more flip-flops on average than all previous solutions for buffer sizes of 16 and 32.

Diminishing gain in the number of restored flip-flops per additional traced flip-flop was pointed out as a shortcoming of the greedy algorithms in Section 3.3. The proposed algorithm alleviates this issue to a large extent. The s38417 circuit is again used as a representative case in Figure 11, as it was used in Figure 3. The selection produced by our solution is able to restore more flip-flops on average compared to that of all 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 the solution by Basu and Mishra [4] (the best previous solution so far in terms of total restoration). Similar trends are observed for other benchmarks as well.

5.4 Algorithm Execution Performance

The trace signal selection is done only once during design phase of the circuit blocks, to be included in signal list for the ELA. Hence the run-time of the selection algorithms is of lesser importance than the quality of selected signals. However if an inordinate amount of time is needed for moderately sized circuit blocks, it might become a bottleneck. In our algorithm the pruning phase was devised especially for this reason. A comparison of the execution time of previous solutions and our solution is presented in Table 4. Note that, especially for small designs the execution performance of the proposed algorithm is often worse, this is due to the number of simulations needed

Circuit	trace width	Ko & Nicolici [7]	Liu & Xu [8]	Basu & Mishra [4]	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 4: Comparison of execution performance for the algorithms considered. All execution times are reported in seconds. Unreported results create the blank cells.

in our selection process. However, these simulations are for restoration estimation, and they are independent of each other during each iteration of the selection algorithm. A possible way to speed this up is to use pattern parallelism often used in GPU platforms, where the same execution is applied on different data sets.

6. 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 trends of restoration per additional traced signal while restoring higher average number of states.

7. 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. DAC*, pages 7–12, 2006.

[2] Altera Verification Tool. *SignalTap II Embedded Logic Analyzer*, 2006. `http://www.altera.com/products/software/products/quartus2/verification/signaltap2/sig-index.html`.

[3] ARM limited. *Embedded Trace Macrocells*, 2007. `http://www.arm.com/products/solutions/ETM.html`.

[4] K. Basu and P. Mishra. Efficient trace signal selection for post silicon validation and debug. In *Proc. VLSI design*, pages 352–357, 2011.

[5] Y.-C. Hsu, F. Tsai, W. Jong, and Y.-T. Chang. Visibility enhancement for silicon debug. In *Proc. DAC*, pages 13–18, 2006.

[6] H. F. Ko and N. Nicolici. Automated trace signals identification and state restoration for improving observability in post-silicon validation. In *Proc. DATE*, pages 1298–1303, 2008.

[7] H. F. Ko and N. Nicolici. Algorithms for state restoration and trace-signal selection for data acquisition in silicon debug. *IEEE Trans. on CAD*, 28(2):285–297, 2009.

[8] X. Liu and Q. Xu. Trace signal selection for visibility enhancement in post-silicon validation. In *Proc. DATE*, pages 1338–1343, 2009.

[9] N. Nataraj, T. Lundquist, and K. Shah. Fault localization using time resolved photon emission and STIL waveforms. In *Proc. ITC*, pages 254 – 263, 2003.

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

[11] H. Shojaei and A. Davoodi. Trace signal selection to enhance timing and logic visibility in post-silicon validation. In *Proc. ICCAD*, pages 168–172, 2010.

[12] B. Vermeulen, T. Waayers, and S. Bakker. IEEE 1149.1-compliant access architecture for multiple core debug on digital system chips. In *Proc. ITC*, pages 55 – 63, 2002.

[13] Xilinx Verification Tool. *ChipScope Pro*, 2006. `http://www.xilinx.com/ise/optional_prod/cspro.html`.

[14] J.-S. Yang and N. A. Touba. Automated selection of signals to observe for efficient silicon debug. In *Proc. VTS*, pages 79–84, 2009.