Combinational Trace Signal Selection with Improved State Restoration for Post-Silicon Debug

Siamack BeigMohammadi and Bijan Alizadeh
School of Electrical and Computer Engineering, College of Engineering
University of Tehran
Tehran, Iran
Email: s.beigmohammadi@ut.ac.ir, b.alizadeh@ut.ac.ir

***Abstract*— Signal selection is the pre-silicon evaluation step which maximizes overall reconstruction rate of state elements. Due to its high complexity, recent efforts on signal selection has focused on sequential nodes in the circuit under debug. In this paper we propose a combinational signal selection algorithm which possibly selects combinational nodes of the circuit to maximize state restoration capability and achieve significant improvements (53% on average) compared to the state-of-the-art signal selection algorithms. To compensate for increase in problem complexity, we also propose a fast state restoration algorithm which offers significant improvement (38% on average) on simulation time over the state-of-the-art state restoration algorithms.**

Keywords—Combinational Trace Signal Selection; State Restoration; Post-Silicon Debug

I. Introduction

Increasing complexity of digital integrated circuits and reduced time to market budget have made pre-silicon verification techniques such as simulation, formal verification, and emulation prone to bugs. Traditionally, each stage of the synthesis flow is combined with a corresponding verification step to ensure error-free synthesis. Anyhow, bugs do escape pre-silicon verifications due to both inaccuracy in modeling and impossibility of full-chip analysis due to limited time to market budget. Besides the synthesis flaws, the initial design may include mistakes itself which may survive to silicon. These trends have forced manufacturers to extend verification beyond pre-silicon steps [7][8]. Due to the high fabrication time and cost, high-quality pre-silicon verification methods remain critical, though.

Despite time-consuming fabrication process, post-silicon prototypes run up to 9 orders of magnitude faster than pre-silicon verification techniques and accept real-world stimulus [9]. The main challenge with post-silicon verification is limited observability into internal circuit nodes. To overcome this limitation, one may use physical probing tools [10] but today's nano-scale feature sizes and high number of metal layers make this approach ineffective due to the design complexity. A better solution is to insert pre-engineered design for debug (DfD) hardware into the design and utilize it in post-silicon verification step to detect and localize functional bugs. One simple implementation would be to reuse internal scan chains (DfT hardware) for post-silicon verification [11][12]. Scan-based techniques are simple and incur minimum hardware overhead. They do not provide real-time data associated with post-silicon verification, though. With the hard-to-activate bugs present in today's complex designs scan-based techniques are inefficient. In fact, debugability requires much more observability than testability DfT hardware offers. In a trace-based post-silicon debug flow, an embedded logic analyzer (ELA) is the DfD hardware that enables real-time data capture [16]. This hardware has found its way into industry as well [13-15]. Basically, an ELA consists of a trigger unit and a trace buffer. The trigger unit signifies misbehaviors or generally programmable events that initiate real-time acquisition of selected signals into an on-chip RAM, i.e. the trace buffer. Trace buffers introduce significant area overhead. This, along with the limited input bandwidth limits the number of traceable signals.

With today's complex circuits manual signal selection is getting far from optimal solution, i.e. a limited set of trace signals that maximize average state restoration capability of the circuit. Automatic trace signal selection problem is a complex problem to which many metric-based [1][2], simulation-based [4], and hybrid [5] algorithms have been proposed. Once the selected state elements are known they can be analyzed offline to reconstruct state elements. State restoration for offline analysis was first defined in [1] accompanied by the first metric-based signal selection algorithm. Further metric-based signal selection algorithms were proposed all incorporating a greedy algorithm that selects state elements one by one based on the proposed metric [2][17]. Also, [3] takes advantage of a circuit graph-based metric with the hypothesis that if the selected signals are in a close proximity, their combination would be suitable to restore more state elements.

Chatterjee, et al. [4], introduced a new insight into automatic signal selection: using simulation instead of a metric to estimate restoration capability. Despite the inevitable simulation time overhead, much accurate results were obtained due to the better correlation of simulated restoration capability to actual restoration capability. Hybrid signal selection algorithms are a mixture of metric and simulation based algorithms which use the metrics to narrow down the candidate state elements for inclusion in simulation step. These state-of-the-art algorithms have demonstrated promising results [5]. Due to its random nature and limited simulation window, [6] proposes to select multiple sets of signals and refine the selection set based on a metric. Despite the time consuming process, this approach reveals further improvements.

A silicon prototype must be verified for both physical and functional bugs. It is noted that the algorithms proposed in this work only apply to functional bugs, assuming that manufacturing test has already been passed and the behavior of the prototype matches the behavior of the circuit netlist. It is also noted that this work focuses on generalized circuits without prior knowledge of the circuit architecture. With this knowledge at hand, it is possible to take advantage of the hierarchy and regularities of the circuit under debug to accommodate designer-aware and faster debug process [18][19]. Besides this architectural knowledge-based signal selection, the need for general automatic signal selection algorithms is growing. Due to its iterative nature, our signal selection algorithm may start with designer-selected signals and add other interesting signals to the selection set to increase design visibility further more.

In this work, the basic idea is to consider combinational gates rather than only state elements in signal selection process. We believe that selection of only the state elements in current signal selection algorithms is a side-effect of the problem complexity and show that with proper manipulations in the current hybrid algorithms, it is feasible to include combinational nodes in the gate-level signal selection process and considerably increase design visibility. We propose a fast combinational hybrid signal selection algorithm that takes advantage of the insights from [5] to speed up combinational simulation-based signal selection. We also propose a fast state restoration algorithm to speed up our hybrid signal selection algorithm.

The remainder of the paper is organized as follows. Section II provides the relevant background and motivation. In Section III our signal selection algorithm is explained. A fast state restoration framework is proposed in Section IV. The experimental results and conclusion are given in Sections V and VI, respectively.

II. Background and Motivation

The major challenge with post-silicon debug is limited visibility of internal states of the circuit under debug, the best available solution of which is design for debug hardware insertion. Fig. 1 shows the post-silicon validation and debug flow. Note that this flow includes both pre-silicon and post-silicon phases. At pre-silicon phase, an ELA is embedded into the initial user design. Signals to be traced by ELA are selected at pre-silicon phase. The more signals get restorable knowing only these traced signals, the more post-silicon visibility is achieved which in turn increases debugability of the circuit.

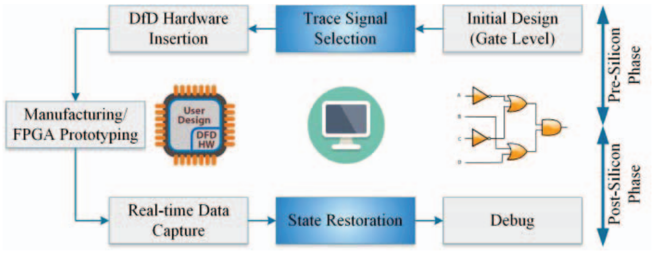

Fig. 1. Post-silicon validation and debug flow.

Trace signal selection is accompanied by a post-silicon signal restoration step which is used to expand trace data and restore unknown states of the circuit under debug. In this paper, we focus on these two compute-intensive steps of post-silicon debug flow. We will introduce both an efficient signal restoration algorithm and an automatic trace signal selection algorithm which possibly selects combinational signals to increase post-silicon visibility.

A. *Trace Signal Selection*

To ensure a high state restoration ratio (SRR) after the chip's trace data are collected, one must wisely select a set of signals to be traced at pre-silicon phase such that the probability of having a high SRR be maximized. This pre-silicon step is referred to as trace signal selection. Theoretically, with a buffer bandwidth of w and depth of d, the best solution to the signal selection problem is to select all w combinations of state elements (or gates) and perform d cycle simulations in several random iterations to estimate restoration capability of each set. The set with maximum capability will be the optimum solution. With today's complex circuits, this approach is definitely infeasible, even without considering combinational logic gates in the process.

To reduce complexity, current automatic signal selection algorithms come up with two solutions: (1) using a greedy algorithm inside which (2) a computationally inexpensive metric for estimating state restoration capability is utilized. The best metric is SRR which requires computationally intensive state restoration process. Thus, the first implementations used a probabilistic metric to estimate state restoration capability of a set of signals [1-3], but simulation-based and hybrid methods gained significance because of their precise SRR estimations which is a key for the greedy algorithm to be successful [4-6].

The current signal selection algorithms employ a greedy procedure in either forward or backward fashion (not to be confused with forward or backward restoration). In forward fashion, at each iteration, signals are evaluated for their combined restoration capability and the best signal is added to selection set. Backward procedure proposed in [4] is based on elimination of signals and was later criticized for both its complexity and accuracy [5][6].

B. *State Restoration*

State restoration is the process of analyzing the currently known signal values in N clock cycles to restore possible values of unknown state elements in that N cycles. Since the number of signals to be traced in a post-silicon prototype is limited, this process is particularly important in post-silicon debug flow to address the limited visibility problem. Fig. 2 illustrates forward and backward restoration rules for two logic gates. The restored values are in a shaded box. The value 'x' indicates that signal is unknown. For example, if one of the inputs of an AND gate is logic '0' by forward restoration we assign logic '0' to its output.

The quality of trace signal selection is popularly quantified by the state restoration ratio (SRR) which is defined as:

$$SRR = \frac{No.of\ known\ FFs\ (state\ elements)}{No.of\ traced\ FFs\ (state\ elements)}. \quad (1)$$

As an example, let us consider the simple gate-level circuit in Fig. 3. Suppose only flip-flop B is traced. By iteratively applying the forward and backward restoration at every clock cycle, state elements will be restored as shown in Table I. Flip-flops D, E, and F are restorable when B is logic 1. After restoration a total of 12 extra flip-flop states are known (flip-flops D, E, and F are restored in 2 clock cycles). This yields SRR of (10+12)/10=2.2.

We tend to include combinational signals in the selection set and thus fairly define combinational state restoration ratio (CSRR) as:

$$CSRR = \frac{No.of\ known\ FFs\ (state\ elements)}{No.of\ traced\ signals\ (FF\ or\ logic\ gates)}\ . \qquad (2)$$

For example if we traced the NOR gate (G1) in Fig. 3, restoration could be as shown in Table II. This gives CSRR of 29/10=2.9. It should be noted that despite combinational gate G1 is not a flip-flop itself, it can restore much more state elements (29) than flip-flop B (12). This signifies possible gains of considering combinational gates in signal selection process. In the rest of this paper, to preserve consistency with previous works, we use the abbreviation SRR whether the selected signals are all state elements or not.

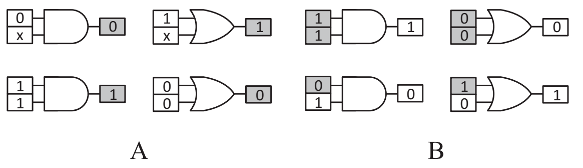

Fig. 2. (a) Forward restoration, and (b) backward restoration.

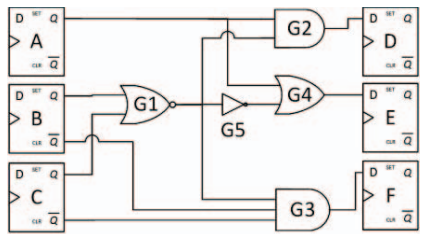

Fig. 3. Example circuit for state restoration.

TABLE I. SAMPLE RESTORED STATES FOR FIG. 3.

FF	Clock Cycle									
	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
A	x	x	x	x	x	x	x	x	x	x
B	1	1	1	0	0	0	1	0	0	0
C	x	x	x	x	x	x	x	x	x	x
D	x	0	0	0	x	x	x	0	x	x
E	x	1	1	1	x	x	x	1	x	x
F	x	0	0	0	x	x	x	0	x	x

TABLE II. SAMPLE RESTORED VALUES BY TRACING G1.

GATE	Clock Cycle									
	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
A	x	x	x	x	x	x	x	x	x	x
B	x	x	x	x	0	0	x	x	0	0
C	x	x	x	x	0	0	x	x	0	0
D	x	0	0	0	0	x	x	0	0	x
E	x	1	1	1	1	x	x	1	1	x
F	x	0	0	0	0	1	1	0	0	1
G1	0	0	0	0	1	1	0	0	1	1

III. Proposed Signal Selection Algorithm

Combinational signal selection is the extension of signal selection problem into a much bigger solution space which takes combinational gates as well as state elements into account. The goal is the same being maximization of restoration of state elements. We believe that inside this bigger solution space there exist more optimal/sub-optimal solutions which we aim to capture. As discussed in subsection II-B, automatic signal selection is inherently a computationally intensive problem and by expanding solution space its complexity grows significantly. Thus we need a fast signal selection algorithm. To do so, we have modified the hybrid signal selection algorithm in [5] such that combinational signals can also be selected. Our generalized signal selection algorithm is explained in the following subsections.

A. Metrics

Authors in [5] implement a forward greedy selection algorithm and propose a good set of metrics which are used to identify top candidates for selection at each iteration of the algorithm. These top candidates are then evaluated by a few number of simulations and the best one is chosen at each iteration. First, we'll give a brief introduction to our interpretation of the metrics for combinational signal selection. A more comprehensive definition of these metrics can be found in [5].

L_f^v: Reachability List of Signal f Taking Value v: The set of restorable state elements if signal f takes value 0 or 1 while all other signals except control signals are unknown. Those state elements that are restorable only by control signals are excluded from the set. Note that in our definition, f can be either a state element or a gate but the contents of the reachability list are only state elements. We use limited X-simulation similar to [5] to extract reachability lists. This process is done once at an initialization step explained in subsection III-B.

r_i: Restorability Rate of State Element i: The probability that unselected state element i is restorable using the trace signals identified so far. We use three simulations at non-overlapping 64-cycle intervals and calculate r_i at each interval and finally assign the average of these three probabilities to the corresponding unselected state element. Note that here i is a state element.

$d_{i,f}^v$: Demand of State Element i From Signal f Taking Value v: A measure of how much signal f with value v can further contribute to the restoration of state element i, given that i is already restored by a rate of r_i. Signal f can be any signal but we limit it to be a signal with state element i in its reachability list, for either value 0 or 1. Demand is defined as follows:

$$d_{i,f}^v = \min\left(1 - r_i, a_f^v\right), \forall i \in L_f^0\ or\ i \in L_f^1 \qquad (3)$$

where a_f^v is the probability that signal f takes value v. This probability is accurately computed just once in the initialization step. Note that i is a state element while f can be either a state element or a combinational gate.

w_f: Impact Weight of Signal f: A measure of the amount of restoration if f is selected as the next trace signal. The impact weight of signal f is defined as follows:

$$w_f = \sum_{v=0,1} \sum_{\forall i \in L_f^v} d_{i,f}^v. \quad (4)$$

Higher values of impact weight indicates that more state elements can be restored if signal f is selected as the next trace signal.

B. Selection Algorithm

We extend the algorithm in [5] for combinational signal selection. Fig. 4 shows overview of our signal selection algorithm. We eliminate the island signal selection step in [5] due to its limited gains. Island signals are those that have both their reachability lists empty (i.e. $L_f^0 = L_f^1 = \emptyset$).

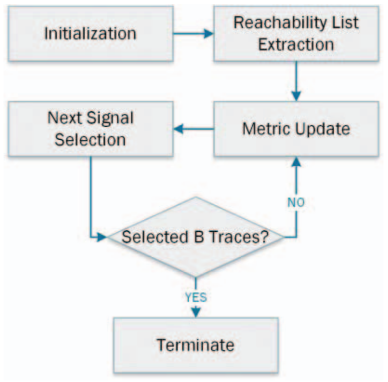

Fig. 4. Overview of the proposed trace signal selection algorithm.

Initialization: The first step is to generate a valid trace of circuit states. This trace is calculated using random values for non-control inputs and a forward-only event-driven 10K cycle simulation. The result is stored and used for the remaining steps of the algorithm. The probability a_f^v is calculated for each state element and combinational gate and remains constant throughout the algorithm. The demands and impact weights are also initialized in this step.

Reachability List Extraction: Initially, control signals are assigned and the circuit is simulated. Signals are then selected one-by-one and assigned 0 or 1 while all other signals are set to be unknown. For each signal, state restoration process is done for both values of 0 or 1 and its reachability lists are extracted and saved for further processing.

Next Signal Selection: In a pure simulation based approach all unselected signals must be simulated in conjunction with currently selected signal and in a limited number of cycles (64 cycles as proposed in [4]) to calculate the restoration gain considering already selected signals. As proposed in [5], we select a number of signals with highest impact weight equal to 5% of the number of state elements. By simulating these top candidates, the best one is chosen to be added to currently selected signals. Note that to reduce selection time, we do not use island signals in our simulation as the offered gains are minimal [5].

Metric Update: At each iteration of the algorithm, we update the impact weight to be used for next signal selection step. This requires recalculation of the three metrics r_i, $d_{i,f}^v$, and w_f. Note that these calculations are very simple compared to the state restoration process.

It is noted that in order to reduce the selection time we exclude inverter and buffer gates from the selection process without losing solution quality.

IV. Proposed State Restoration Algorithm

The authors of [1], accelerate state restoration process by exploiting the inherent parallelism of the problem. In addition to state restoration step in post-silicon verification, this event-driven bit-parallel implementation of gate-level state restoration problem has been extensively used in the emerging simulation-based and hybrid trace signal selection algorithms [4-6]. In this work, an even faster state restoration algorithm is proposed to further accelerate state restoration and trace signal selection processes in post-silicon verification. This implementation can directly speedup state-of-the-art simulation-based and hybrid trace signal selection algorithms [4-6].

Algorithm 1 shows our proposed event-driven state restoration algorithm which uses forward and backward restorations to restore as many signal states as possible. The basic idea is the fact that if a node value is restored by forward restoration, backward restoration will not restore any of the node's unknown input values. Similarly, if backward restoration restores a node's input value, then forward restoration on the node will not restore any unknown output value. Thus we use two event queues to reduce the number of forward and backward restorations. It should be noted that with simple bit-parallel formulations for state restoration, the simulation time is proportional to the number of events.

Algorithm 1: Faster Algorithm for State Restoration
Input: *Circuit, Known_Signal_Values*
Output: *Restored Signal Values*

```
1 fwd_search_list = Known_Signal_Values;
2 bwd_search_list = Known_Signal_Values;
3 while both search lists are not empty do
4   while fwd_search_list is not empty do
5     cur_node = first node in fwd_search_list;
6     for each (child_node of cur_node) do
7       ForwardOperation (cur_node, child_node);
8       if (new data are restored for child_node) then
9         Put child_node at end of fwd_search_list
10        if child_node has more than one input then
11          Put child_node at end of bwd_search_list
12  while bwd_search_list is not empty do
13    cur_node = first node in bwd_search_list;
14    for each (parent_node of cur_node) do
15      BackwardOperation (cur_node, parent_node);
16      if (new data are restored for parent_node) then
17        Put parent_node at end of bwd_search_list
18        if parent_node is a fanout then
19          Put parent_node at end of fwd_search_list
```

Note that if the restored node after a forward restoration drives a multi-input node, a combined backward restoration is needed on that node. Also, if backward restoration restores a fanout node, forward restoration on that node is also required. Also note that in order to further reduce the number of events, in the while loop of Algorithm 1, we evaluate combinational events prior to events on sequential nodes.

V. Experimental Results

We've implemented both our combinational hybrid signal selection (referred to as CHYBR) and state restoration (referred to as FXSim) algorithms in C++. We have tested FXSim against ModelSim and verified it for correct operation.

A. Comparisons for CHYBR

We compare our combinational signal selection algorithm with the following algorithms:

- SIM-F: simulation-based with forward greedy selection.
- CSIM-F: combinational simulation-based with forward greedy selection.
- HYBR: Hybrid signal selection [5].

Note that as proposed in [5][6], in general, SIM-F performs better than its backward greedy selection algorithm [4] and that is why we've excluded the backward selection from the comparisons. CSIM-F is our extension of SIM-F which we use to demonstrate achievable improvements when considering combinational gates in signal selection. Note that in order to make the comparisons fair, all the algorithms (including HYBR [5]) are implemented as single-threaded programs in Ubuntu OS which run on an Intel Core i7 Sandy Bridge architecture with maximum clock frequency of 2.2 GHz. We've used our state restoration engine (Algorithm 1) to implement these signal selection algorithms.

In this experiment, we use seven benchmarks from ISCAS89 suite [20] and three benchmarks from IWLS05 [21]. The two benchmarks S38584 and S35932 include control signals. S38554 has an active low reset signal (g35) which is set to 1 throughout the process. S35932 includes an active low reset signal (RESET) which is set to 1 and two other control signals TM0 and TM1 which define four modes of operation. The reported results are the average over these modes of operation. Three buffer widths of 8, 16, and 32 bits and a buffer depth of 4K is assumed. Table III shows the SRR and Table IV shows the runtime of each algorithm. The number of gates reported in Table III are the number of non-inverter gates which we include in the selection process. Note that we apply all the signal selection algorithms to the original ISCAS89 benchmarks without a re-synthesis. The optimized version of IWLS05 benchmarks are used, since they do not include any signal.

From Table III, we see that our proposed simulation-based method (CSIM-F) outperforms the state-of-the-art signal selection algorithms (i.e., SIM-F and HYBR) for three buffer bandwidths of 8, 16, and 32 and buffer depth of 4096. For ISCAS89 benchmarks, this implementation is 18% better in terms of SRR compared to typical simulation-based algorithm. For IWLS05 benchmarks, CSIM-F performs 122% better on average.

Our proposed hybrid algorithm (CHYBR) achieves 53% higher SRR compared to the current state-of-the-art hybrid signal selection algorithm (HYBR) and 34% higher SRR compared to the time-consuming SIM-F algorithm. The ISCAS89 benchmarks have lower logic depth compared to the IWLS benchmarks. This is why for two benchmarks B17 and B18 our proposed methods offer significant improvements. The average improvement of CSIM-F over SIM-F and average improvement of CHYBR over HYBR for each buffer bandwidth is given in Table III.

On the other hand, our combinational simulation-based method is extremely time-consuming compared to the sequential-only simulation-based approach. For ISCAS89 benchmarks, the combinational approach is 5.9 times slower than its sequential counterpart, reducing its feasibility in large digital circuits. For IWLS05 benchmarks this problem becomes more dominant with 18 times slower speed on average. Note that these slowdowns are proportional to the increase in solution space by considering combinational gates.

As it can be seen for CHYBR, we have successfully reduced the signal selection time while increasing solution quality (SRR). Compared to the reference sequential simulation based approach (SIM-F) our hybrid implementation is 6.4 times faster. Compared to the simulation-based counterpart of our implementation (CSIM-F), CHYBR is 66 times faster while the attained SRR is just 12% lower. Our hybrid algorithm has even lower runtime compared to sequential-only hybrid implementation due to the elimination of island signal selection. CHYBR offers 36% lower runtime while giving 53% improvement in solution quality. Note that we have implemented all four signal selection algorithms using our fast state restoration engine and the runtimes are solely algorithm-oriented.

TABLE III. Comparison of SRR of Different Algorithms

Benchmark	**#FFs**	**#Gates**	**SIM-F**			**CSIM-F**			**HYBR [5]**			**CHYBR**		
			8	***16***	***32***	***8***	***16***	***32***	***8***	***16***	***32***	***8***	***16***	***32***
S5378	179	1004	14.8	8.6	5.3	14.7	9.2	5.5	14.8	8.2	4.9	14.9	8.1	5.1
S9234	211	2027	12.7	8.7	5.1	16.3	11.2	6.1	10.1	6.4	4.0	14.5	8.7	5.6
S13207	638	2573	37.3	27.5	16.3	49.8	29.9	16.5	37.0	24.7	14.8	44.1	29.1	16.0
S15850	534	3448	42.5	24.9	14.0	56.1	29.2	15.4	29.2	23.6	12.9	49.3	26.3	14.8
S35932	1728	12204	43.2	31.2	20.8	51.8	37.2	28.1	25.8	19.7	13.8	42.5	28.5	18.0
S38417	1636	8709	15.9	18.8	19.6	24.9	19.7	21.2	10.1	14.9	13.3	15.2	15.0	12.8
S38584	1426	11448	14.8	8.6	5.3	14.7	9.2	5.5	14.8	8.2	4.9	14.9	8.1	5.1
B17	1412	21215	9.5	9.8	7.6	26.0	20.8	16.2	9.1	7.7	4.3	25.6	15.3	10.4
B18	3270	62978	18.3	10.4	6.1	38.3	39.1	25.2	15.1	10.7	11.0	37.2	37.2	24.5
B22	703	13674	13.3	6.1	5.1	13.3	7.3	4.4	9.3	7.2	5.2	12.6	7.3	5.5
Avg. Improvement (%)						50.1	54.4	54.6				67.5	50.9	40.5

TABLE IV. RUNTIME COMPARISON OF DIFFERENT ALGORITHMS (IN SECONDS)

Benchmark	SIM-F			CSIM-F			HYBR [5]			CHYBR		
	8	*16*	*32*	*8*	*16*	*32*	*8*	*16*	*32*	*8*	*16*	*32*
S5378	0.1	0.2	0.3	0.6	1.1	2.2	0.0	0.1	0.1	0.0	0.0	0.1
S9234	0.3	0.6	0.9	2.5	4.7	8.7	0.1	0.1	0.3	0.0	0.1	0.1
S13207	1.1	2.2	4.1	5.5	11.0	19.7	0.3	0.6	1.1	0.2	0.3	0.6
S15850	1.7	2.7	4.7	8.9	16.0	28.9	0.5	0.8	1.3	0.3	0.5	0.8
S35932	6.8	13.6	25.2	39.6	80.0	166.3	2.3	3.8	7.2	1.5	2.3	4.1
S38417	5.7	14.2	25.0	55.3	102.9	213.2	1.0	2.7	5.7	0.9	1.7	3.3
S38584	4.4	7.7	16.6	58.6	130.7	333.1	1.0	2.0	4.1	0.8	1.7	3.3
B17	37.0	74.2	146.8	768.1	1430.5	3156.1	10.4	19.3	38.1	6.3	12.7	25.8
B18	3.2	5.2	7.1	41.8	64.8	145.5	0.4	0.8	1.8	0.3	0.5	1.1
B22	0.1	0.2	0.3	0.6	1.1	2.2	0.0	0.1	0.1	0.0	0.0	0.1

B. *Comparisons for FXSim*

We've implemented both Algorithm 1 (referred to as FXSim) and the algorithm in [1] (referred to as XSim). Table V shows the achievable improvements of our algorithm over [1] while restoring 10K cycles of three largest benchmarks using selected gates from this work. Our algorithm reduces both the number of forward and backward operations and memory accesses. Third column shows speedup of our algorithm. On average, it is 38% faster compared to the state-of-the-art state restoration engine [1]. Note that this improvement directly improves runtime for current simulation-based and hybrid signal selection algorithms as the state restoration process is the computationally intensive part of these algorithms.

TABLE V. RUNTIME COMPARISON OF PROPOSED STATE RESTORATION PROCESS (IN SECONDS)

Benchmark	FXSim	XSim [1]	% Speedup
B18	7.62	10.00	23.8
S38417	3.24	5.96	45.6
S38584	3.12	5.71	45.4

VI. CONCLUSION

We presented an efficient combinational hybrid signal selection algorithm that can be used for post-silicon validation and debug. In terms of speed, our algorithm have even less runtime compared to current non-combinational hybrid signal selection algorithm. In terms of solution quality, we achieve an average of 53% improvement over non-combinational hybrid signal selection algorithm. We do achieve 34% higher SRR compared to the non-combinational time-consuming simulation-based approach. Our key contribution is expanding solution space into combinational gates as well. Our experiments demonstrate that combined with our proposed state restoration engine with 38% speedup, this approach provides aforementioned 53% improvement in solution quality.

REFERENCES

[1] H. F. Ko and N. Nicolici, "Algorithms for state restoration and trace-signal selection for . acquisition in silicon debug," IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, vol. 28, no. 2, pp. 285-297, Feb. 2009.

[2] X. Liu and Q. Xu, "Trace signal selection for visibility enhancement in post-silicon validation," in Proc. DATE, pp. 1338-1343, 2009.

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

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

[5] M. Li and A. Davoodi, "A hybrid approach for fast and accurate trace signal selection for post-silicon debug," IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, vol. 33, no. 7, pp. 1081,1094, July 2014.

[6] K. Rahmani, P. Mishra, and S. Ray, "Efficient trace signal selection using augmentation and ILP techniques." International Symposium on Quality Electronic Design (ISQED), pp. 148-155, March 2014.

[7] J. Keshava, N. Hakim, and C. Prudvi, "Post-silicon validation challenges: How EDA and academia can help," Design Automation Conference (DAC), ACM/IEEE, pp. 3-7, June 2010.

[8] A. Nahir, et al. "Post-silicon validation of the IBM POWER8 processor," Design Automation Conference (DAC), ACM/EDAC/IEEE, June 2014.

[9] R. Kaivola, et al. "Replacing testing with formal verification in Intel® Core™ i7 processor execution engine validation," in CAV'09: Proc. of the 21st International Conference on Computer Aided Verification, pp. 414-429, Jan. 2009.

[10] C. Boit, et al. "Physical IC debug-backside approach and nanoscale challenge," Advances in Radio Science 6.10, pp. 265-272, 2008.

[11] B. Vermeulen, T. Waayers, and S. Bakker, "IEEE 1149.1-compliant access architecture for multiple core debug on digital system chips," in Proc. ITC, pp. 55-65, 2002.

[12] F. M. De Paula, M. Gort, A. J. Hu, S. J. E. Wilton, and J. Yang, "BackSpace: Formal analysis for post-silicon debug," Proceedings of the International Conference on Formal Methods in Computer-Aided Design (FMCAD), pp.1-10, Nov. 2008.

[13] Xilinx Inc., "Vivado design suite user guide: programming and debugging," ver. 2015.2. http://www.xilinx.com.

[14] Altera Inc., "Quartus II Handbook Volume 3: Verification", May 2015. http://www.altera.com.

[15] ARM Inc., "Better Trace for Better Software: Introducing the new ARM CoreSight System Trace Macrocell and Trace Memory Controller," Oct. 2010. http://www.arm.com.

[16] M. Abramovici, P. Bradley, K. Dwarakanath, P. Levin, G. Memmi, and D. Miller, "A reconfigurable design-for-debug infrastructure for SoCs," in Proc. IEEE/ACM Des. Autom. Conf. (DAC), pp. 7-12, July 2006.

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

[18] I. Wagner and V. Bertacco, "Post-Silicon and Runtime Verification for Modern Processors," Springer Science & Business Media, 2010.

[19] S. B. Park and S. Mitra, "IFRA: Instruction footprint recording and analysis for post-silicon bug localization in processors," in 45th Design Automation Conference, ACM/IEEE, pp. 373-378, 2008.

[20] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits," in Proc. IEEE Int. Symp. Circuits Syst., pp. 1929–1934, May 1989.

[21] C. Albrecht. IWLS 2005 benchmarks. In *IWLS*, 2005.