Pruning-based Trace Signal Selection Algorithm

Kang Zhao and Jinian Bian
EDA Lab, Dept Computer Science & Technology, Tsinghua University, China

Abstract— To improve the observability in the post-silicon validation, how to select the limited trace signals effectively for the data acquisition is the focus. This paper proposes an automated trace signal selection algorithm, which uses the pruning-based strategy to reduce the exploration space. The experiments indicate that the proposed algorithm can bring higher restoration ratios, and it is more effective compared to existing methods.

I. Introduction

Silicon debug is perhaps the most exciting and challenging stage of the integrated circuit (IC) development process [1]. It includes two processes: pre-silicon validation and post-silicon validation. Pre-silicon is mainly related with functional and timing errors, and post-silicon is related with functional and electrical problems [2]. Recently, with the growing complexity of the integrated circuit, many errors may escape the pre-silicon validation. Therefore, it has been leading to a development of the post-silicon validation techniques.

To identify the bugs effectively, designers want to know each state for each signal in the circuit, but it is difficult. Probe can be used to achieve this object [3]; however, due to the high complexity of the circuit, probing tools cannot keep up with the ever-increasing development of the IC design technology. Reusing internal scan chains has been well developed in the industry, but it needs to stop the operation of the circuit under debug (CUD) and only provides postmortem debug ability [4].

To settle this issue and enhance the visibility of the circuit, a new technique is proposed and accepted by the industry, which only monitors and traces some internal signals. Their values will be stored in the trace buffer and can be obtained from the trace port. The transfers will be operated by the interconnection fabric [5]. Based on an analysis on these trace signals, we can estimate the states of other signals. Thus, it need not large buffer to store many signals and can estimate many other signals effectively. Low cost and high efficiency are just its advantages. However, due to the limitation of the trace bandwidth, the number of trace signals is limited. To get a maximum visibility of the circuit, how to select a limited set of trace signals is our focus.

In the industry, designers often select the trace signals based on the working experience [4]. However, it is a boring and time-consuming work, and the efficiency by manual is very low. Sometimes the validation quality cannot be ensured due to the designer's carelessness. Furthermore, many tricky debugs cannot be found easily by manual, and we must make an analysis with the help of the computer. Therefore, the design automation of the trace signal selection is necessary.

This paper will focus on the trace signal selection process for the post-silicon validation, and propose a pruning-based selection algorithm to settle this issue. Our strategy is to compute the restoration range for each signal in the circuit, and then get the optimal result via combination and comparison. During the combination and comparison, pruning-based enumeration is used to reduce the exploration space.

The rest of the paper is organized as follows. In Section II, the motivation and related work are presented. Then, Section III presents the details of the pruning-based selection algorithm. In Section IV, the feasibility of the algorithm is verified based on the experiment. Finally, the conclusion is drawn in Section V.

II. Motivation and Related Work

During the post silicon validation, designers want to view the values for each signal at any time. However, it is high-cost and impossible. To settle this issue, we can select several trace signals to store in the trace buffer. They will help designers to estimate and restore other signals and find the errors. However, the number of trace signals is limited. To view more states in the circuit and identify the bugs effectively, how to select the trace signals is the focus.

To settle this issue, Ko and Nicolici first proposed a probability-based signal selection algorithm [6]. The authors defined two kinds of restorations, forward and backward. For example, if one of the inputs for a two-input AND gate is 0, the output will be estimated as 0 using the forward restoration; if the output is 1, the two inputs will be 1 using the backward restoration. Then the propagation proceeded and the probability of each state was calculated. As shown in Fig. 1, "F0/F1" means the probability that the current signal is 0/1 using forward restoration; "B0/B1" means the probability to be 0/1 using backward restoration. The four restorabilities were calculated using the formulas shown in Fig. 1(c). For each flip-flop (FF), it sums up all restorabilities and selects the biggest one as the trace signal. In the example, FF C is selected as the trace signal and its restorability is 10. The restoration ratio r is used to measure the quality of the final results, which is calculated with $r = \frac{N_{traced}+N_{restored}}{N_{restored}}$. N_{traced} is the number of states for the trace signals, and $N_{restored}$ is the number of states for the estimated signals. As shown in Fig. 1(b),

978-1-4244-7516-2/11/$26.00 ©2011 IEEE

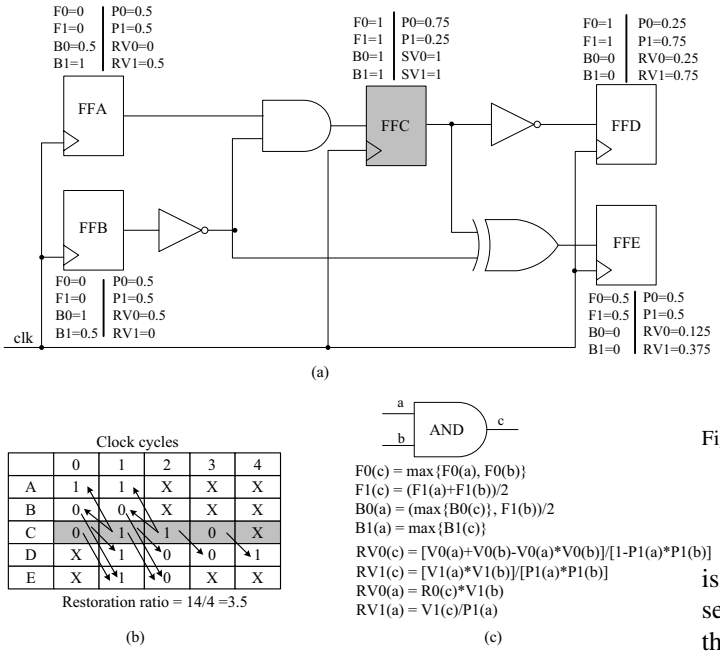

Fig. 1. An example of the state restoration. (a) Circuit under debug; (b) Restored data in the flip-flops; (c) Restoration formula definitions [6, 4].

$r = (4 + 10)/4 = 3.5$. However, the definitions of the circuit-level propagation in [6] is inaccurate.

To improve the method in [6], Xiao and Qiang proposed a new selection algorithm that can restore much more missing states [4]. The authors first presented accurate definitions for the restoration propagation, as shown in Fig. 1. "RV" was used to replace "F" and "B", which was calculated as $RV = P \times (F + B - F \times B)$. Here P means the probability in functional mode. Finally, each restorability was summed up and the biggest one was selected as the trace signal. The advantage of this formulation is that the conditional probability can consider the two propagation directions together. In addition, different initial value of the trace signal can affect the final restoration ratio, so the functional probability P can also improve the accuracy of the results.

Based on [6], [7] proposed another coverage-based trace signal selection method. Its motivation is not to get a highest restoration ratio; instead, the selection speed is the focus. This method also used the definition of F0/F1/B0/B1, and explored the circuit both under forward restoration and backward restoration. To speedup the exploration, it selected the trace signals based on how many other signals will likely to be covered after state restoration. When no signals were found to cover additional signals, this algorithm will gradually decrease the requirement by lowering the restorability for classifying signal coverage. The final results indicated that this method used less restoration time with a lower restoration ratio.

However, [6, 4, 7] all used the greedy trace signal selection algorithm which cannot get the better results. Their strategy

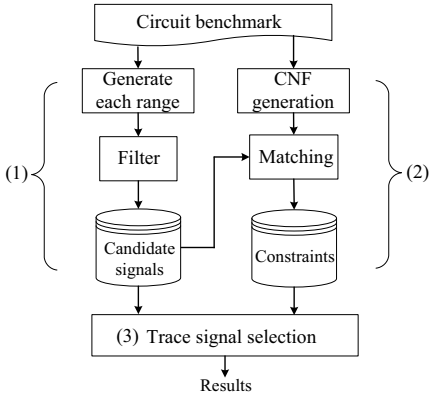

Fig. 2. The flow of the proposed pruning-based selection algorithm.

is to select one trace signal s_1 during the first iteration, then select next trace signal s_2 during the second iteration, and then the third. However, s_1 may be not selected in the better combination, although it is the best choice when selecting only one trace signal. Therefore, this greedy selection algorithm can only get a better result.

To settle the issues above, this paper will propose a pruning-based trace signal selection algorithm. Instead of the probability, we will use the "real" restoration range of each signal. It can match the restoration process accurately. Furthermore, different combinations will be calculated and compared based on each restoration cover range, which can bring a better trace signal result.

III. Pruning-based Trace Signal Selection

Problem: Given the circuit with n FFs, find a subset of trace signals not exceeding k, so that the restoration ratio is maximum.

Our strategy is to compute the restoration range of each signal first, then compare the effects based on different combinations, and finally find the better combination. As shown in Fig. 2, the workflow includes three parts: candidate signal generation, constraints generation, and trace signal selection. They will be presented respectively in the following sections.

A. Candidate Signal Generation

To get the maximum restoration ratio, we first present the definition of the restoration range for each FF.

Definition 1 *For each FF x in the circuit, the set R(x) is the restoration range: $\{y \mid y$ can be restored by $x\}$.*

The method of computing each $R(x)$ relies on the rules of the circuit gates, such as AND, OR, XOR and NOT. As shown in Fig. 3, the restoration process includes two directions: forward and backward. For the AND gate, if the trace signal is

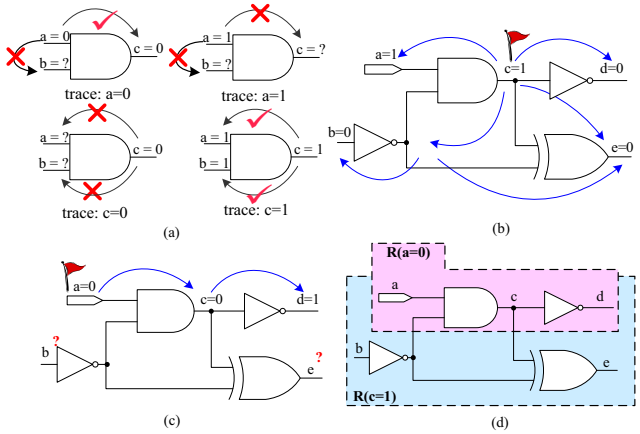

Fig. 3. (a)Restoration principles for the AND gate; (b)(c)(d) the restoration ranges of $c = 1$ and $a = 0$ respectively.

R_half(Node x, Direction dir)
1. $S \Leftarrow \{x\}$;
2. **for**(each node i which connects with x under dir)
3. **if**(i can be traced from x)
4. $S \Leftarrow S \cup$ R_half(i, dir);
5. **return** S;

R(Node x)
1. $S \Leftarrow \emptyset$;
2. **for**(each gate i in the circuit)
3. $D[i] \Leftarrow$the total port number of i; //max degree
4. $S \Leftarrow S \cup$ Add(S, R_half(x, Back), D);
5. $S \Leftarrow S \cup$ Add(S, R_half(x, Forward), D);
6. **return** S;

Fig. 4. Algorithms of the restoration range generation, where R_half generates the range under one direction, and R is the whole range.

AND	OR	XOR
a = 1, b = 0, c = 0 trace: a=1, c=0	a = 0, b = 1, c = 1 trace: a=0, c=1	a = X, b, c = X trace: a=X, c=X
a = 1, b = 1, c = 1 trace: a=1, b=1	a = 0, b = 0, c = 0 trace: a=0, b=0	a = X, b = X, c trace: a=X, b=X
Input=1 Output=0	Input=0 Output=1	Input=X Output=X

Fig. 5. The conditions that the signal is potential. If yes, D will be changed.

Add(Set S_0, Set S_1, Degree $\&D$)
1. $A \Leftarrow \emptyset$; $Queue \Leftarrow S_1 - S_0$;
2. **while**($Queue$ is not empty){
3. $x \Leftarrow$ the top element of $Queue$;
4. Remove the top element of $Queue$;
5. **for**(each gate i connecting with the node x){
6. **if**(i is not dead and x is potential for i){
7. $D[i] \Leftarrow D[i] - 1$;
8. **if**($D[i] = 1$){
9. $A \Leftarrow A \cup$R(y); //y is the last port of i
10. Add R(y) to the end of $Queue$;
11. Sign the gate i to be dead;
12. }//end if
13. }//end if
14. }//end for
15. }//end while
16. **return** $A \cup S_1 - S_0$;

Fig. 6. The generation of the additional range during combination.

$a = 0$, only c can be estimated; if the trace signal is $c = 1$, both a and b can be visible. Therefore, the restoration range is calculated under two directions respectively, as the algorithm R_half shown in Fig. 4. It uses the recursion idea. If the neighboring signal can be restored by the current signal, the propagation will continue. However, there may be conflict or additional new propagation when combining two ranges, as shown in Theorem 1.

Theorem 1 *Let $R(a)$ and $R(b)$ be two different ranges. If there is no conflict, $R(a, b) \geq R(a) \cup R(b)$.*

Proof: Let $x \in$R(a) and $y \in$R(b). If x and y are connected with the same gate, the other signal z on the same gate may be restored. For example, if $x = 1 \wedge y = 1$ and they are both the inputs of the AND gate, the output signal z must be 1.Therefore, there may be an additional set U which can be restored, and $\forall u(u \in U \wedge u \notin$ R(a) $\cup$ R(b)).

For each gate, if the values of other signals are visible, the last signal will be restored. Then we call this gate is "dead". As shown in Fig. 5, if $a = 1$ and $c = 0$ for the AND gate, the input b will be restored. Based on the analysis, if the signal satisfies the conditions in Fig. 5, it is said to be potential. Let the gate g has n ports ($n - 1$ inputs and 1 output). If $n - 1$ ports are potential, the last port will be restored. Based on this idea, we present an array D and each element in D represents the number of invisible ports. Once a port is potential, $D(g) \Leftarrow D(g) - 1$. This process will stop until $D(g) = 1$. The details are presented in Fig. 6. Therefore, $Range(S_0 + S_1) = S_0 +$Add(S_0, S_1, D). Based on the function "Add", we present the implementation of R(x) (Fig. 4), which is used to compute the restoration range.

However, not all signals can be used as the trace signals. To reduce the exploration space, the redundant signals will be filtered based on the following theorem:

Theorem 2 *Let a and b be two different signals and there is*

Filter( )
1. $C \Leftarrow$all signals in the circuit;
2. **for**(each two signals i, j in the circuit)
3. **if**(i, j are connected with a NOT gate)
4. $C \Leftarrow C - \{\mathrm{R}(j)\}$;
5. **else if**($j \in \mathrm{R}(i)$)
6. $C \Leftarrow C - \{\mathrm{R}(j)\}$;
7. **end if**
8. **end for**
9. **return** C;

Fig. 7. The implementation for the filter.

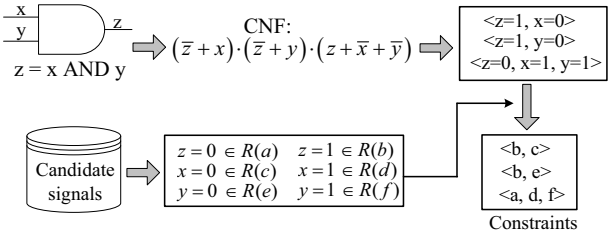

Fig. 8. The flow of the constraint generation in detail.

no NOT gate between them. Then we can get: (1) $b \in R(a) \rightarrow a \notin R(b)$; (2) $b \in R(a) \rightarrow R(b) \subset R(a)$.

Proof: (1) Let a and b be connected with a multi-input gate. By contradiction, let $b \in \mathrm{R}(a) \wedge a \in \mathrm{R}(b)$. Based on the analysis on the NOT, AND, NAND, OR, NOR and XOR gates, only the NOT gate satisfies this condition. This conflicts the precondition (no NOT gate). Therefore, $a \notin \mathrm{R}(b)$. If a and b are not connected directly, there must be a path between them, so it can be proved step by step. (2) Since $b \in \mathrm{R}(a) \rightarrow a \notin \mathrm{R}(b)$, the propagation will be under only one direction. Therefore, $\forall y(y \in \mathrm{R}(b) \wedge b \in \mathrm{R}(a) \rightarrow y \in \mathrm{R}(a))$. Then, $\mathrm{R}(b) \subset \mathrm{R}(a)$.

Based on Theorem 2, the restoration range satisfies the partial order. If $b \in \mathrm{R}(a)$, $\mathrm{R}(a)$ must be larger than (or equal to) $\mathrm{R}(b)$ and b can be replaced by a. The implementation is presented in Fig. 7, which generates the candidate trace signals. The function Filter() first filters the signals related with NOT gate, and then removes the signals which belong to other restoration ranges. Finally, the candidate trace signals can be obtained.

B. Constraints

During the combination of those restoration ranges, there may be conflicts. Therefore, how to get the constraints is the emphasis in this step. Every Boolean circuit can be expressed by the conjunctive normal form (CNF). For the gate "$z = x$ AND y" as an example, its CNF can be expressed as: $(\overline{z} + x) \cdot (\overline{z} + y) \cdot (z + \overline{x} + \overline{y})$. The method of the CNF generation is referred to [8]. With the CNF expression, we can get the conflict constraints easily. For example as shown in Fig. 8, three constraints are generated via negative operation on each item of the CNF: $< z = 1, x = 0 >$, $< z = 1, y = 0 >$ and $< z = 0, x = 1, y = 1 >$. Since the current signal may be not in the set of candidate trace signals, there must be a matching process. As shown in Fig. 8, if the current signal $z = 0$ belongs to the restoration range of a, it will be replaced by a in the constraints. The implementation details of the matching process are omitted here.

C. Trace Signal Selection

The proposed trace signal selection method is to enumerate different combinations and find the better one. However, the exploration space is exponential. To reduce the huge space, a pruning strategy is proposed, which has three characters.

First, the data structure is fit for a fast pruning. As the binary tree shown in Fig.9(a), each level represents whether the current signal is selected or not (1/0). Since the enumeration process uses the lexicographic order (Fig.9.b), once the current combination violates the constraints, there will be no necessary to explore its subtree. Furthermore, it is obvious that the initial order has a distinct impact on the enumeration speed. If the invalid combination appears earlier, the pruning will be faster. Therefore, the invalid combinations will be scheduled first based on the constraints, so that we can eliminate them quickly. In addition, the maximum range is the final target, so the initial order will also be scheduled with a descending order.

Second, the algorithm efficiency can be increased based on the stack structure. From Fig.9(b) it is clear that the neighboring candidates have the same segment. To speedup the exploration efficiency, the intermediate results should be reused in the following computing. During the implementation, we use the stack structure to settle this issue. As shown in Fig. 10, three stacks are presented. Stack A is used to store the candidate signals, stack X stores the restoration range of the current signal, and stack Y records the additive length in X for each change. Based on this structure, the previous range set can be reused directly.

Fig. 11 presents the implementations of the trace signal selection. After the initialization (line 1-7), different combinations are enumerated with the lexicographic order. Once the pointer exceeds the scope, the exploration will stop and pop the top of the stacks (line 13-22); otherwise, next signal will be push into the stacks (line 23-29). If the current combination violates the constraints, this iteration will be omitted (line 25-27). When the stack A is full, the size of its restoration range can be obtained by the stack X. Once a better combination appears, it will be recorded. Here R_pop and R_push represent the pop and push functions for the three stacks. Finally, the better result is found.

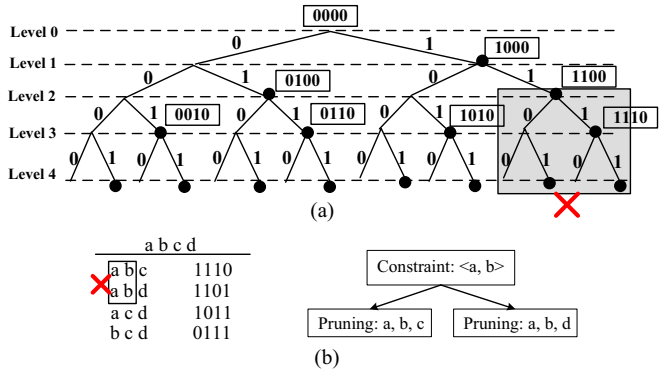

Fig. 9. (a)The binary tree model to illustrate the pruning-based strategy; (b)enumeration in the lexicographic order.

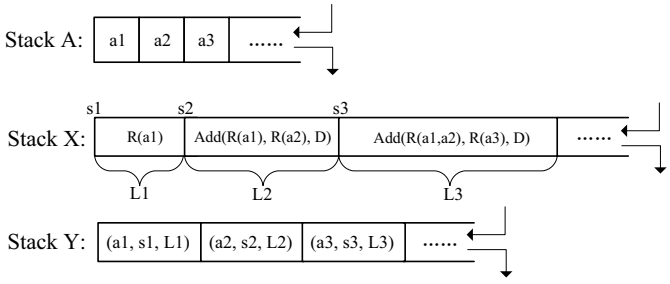

Fig. 10. Enhance the algorithm efficiency via stack structure.

Selection(Set C_{trace}, Size n, Constraints)

1. $V \Leftarrow$ Schedule(C_{trace}, Constraints);
2. Generate a new array A, and its length is n;
3. A.push_back(V.begin); $p \Leftarrow$1; //p: a pointer to V
4. Generate two empty arrays X and Y;
5. **for**(each gate i in the circuit)
6. $D[i] \Leftarrow$the total port number of i; //max degree
7. **for**(each $i \in A$) R_push($A[i], X, Y, D$);
8. **while**(A is not empty **And** V is not full){
9. **if**(A is full and its range is larger than ever){
10. Record the combination in $result$;
11. }
12. **if**(A is full){A.pop_back(); R_pop(X, Y, D);}
13. **else if**(V is full){
14. A.pop_back();
15. **if**(no violation during previous iteration){
16. A.pop_back(); R_pop(X, Y, D);
17. }
18. **if**(A is not empty){
19. $p \Leftarrow$ the next position in V for A.end;
20. A.pop_back(); R_pop(X, Y, D);
21. }//if
22. }//else if
23. **else**{
24. A.push_back($V[p]$);
25. **if**(A violates the constraints){
26. A.pop_back(); ++p; **continue**;
27. }
28. **else**{R_push($V[p], X, Y, D$); ++p;}
29. }//else
30. }//end while
31. **return** the better $result$;

Fig. 11. The algorithm of the pruning-based trace signal selection.

IV. Experiment

To verify the proposed algorithm, we have implemented it in C++ on a Linux machine. This machine has an Intel Xeon 3-GHz CPU and 4-GB memory, and the OS is Red Hat Enterprise Linux AS release 3. The programs were compiled using GCC 3.4.6 with option -03. As the same with [6, 4, 7], the trace buffer is defined as $8 \times 4k$ and $16 \times 4k$. For comparison, we used some ISCAS'89 benchmark circuits. Here we do not use the same benchmarks; instead, some smaller circuits in ISCAS'89 are selected. The reason is that our algorithm is exhaustive and the executing time will rise faster than previous methods. Its advantage is that it can bring higher restoration results. Therefore, we implemented the methods in [6, 4, 7] and compare them.

The front-end should be a random 0/1 generator. We have implemented it with the C++ function $rand()$. For the $8 \times 4k$ trace buffer as an example, the generator will generates 8 different 0/1 vectors, and each vector has 0/1 values with a probability. Here the 0/1 probability is defined as 0.5. The back-end is a simulator which computes the restoration ratio r. As the example shown in Fig. 1(b), the values of the trace signals are obtained from the 0/1 generator and other signals were restored. Finally the restoration ratio was calculated. It should be noted that the number of cycles is not more than $4k$ in our simulation, which is different with [4]. Since the vectors with fixed length $4k$ were used in the program, the restored signals beyond $4k$ were not considered. The trace selection time was measured using the C++ function $clock()$ in "time.h".

To compare the efficiency of the trace selection algorithms, the results will be presented for the gradual approach in [6] (referred to as "$gradual$"), the greedy selection approach in [4] (referred to as "$greedy$") and the coverage-based method in [7] (referred to as "$cover$"). The proposed selection algorithm in this paper is referred as "$pruning$".

Table I and Table II presents the results when 8 nodes and 16 nodes are selected to trace respectively. Column 2 presents the number of DFFs in each circuit. Then the experimental results are presented in Column 3 to Column 5 (for [6]), Column 6 to Column 8 (for [4]), Column 9 to Column 10 (for [7]), and Column 11 to Column 12 (for the proposed algorithm). "#TN" means the number of selected traced signals, "#RS" is the number of the restoration states, r is the restoration ratio, and $time$ is the execution time for the trace selection process.

TABLE I
EXPERIMENTAL RESULTS AND RESTORATION RATIO COMPARISON (8 NODES).

		gradual[6]			**greedy**[4]			**cover**[7]			**pruning**(proposed)			$\triangle r$ ($\times$)		
name	**#DFF**	**#RS**	r	**time(s)**	**#RS**	r	**time(s)**	**#RS**	r	**time(s)**	**#RS**	r	**time(s)**	**[6]**	**[4]**	**[7]**
s382	21	51987	2.625	2.90	55977	2.99	2.64	59986	3.49	0.42	40356	11.09	1.51	4.22	3.71	3.18
s641	19	43997	2.38	12.01	56107	4.51	7.96	50995	7.37	3.73	36321	10.08	1.71	4.24	2.24	1.37
s1196	18	39664	2.24	6.18	39713	2.24	6.78	52164	5.34	1.24	33692	9.42	8.50	4.21	4.21	1.76
s1238	18	39664	2.24	5.88	39713	2.24	4.15	52164	5.34	1.10	33692	9.42	9.13	4.21	4.21	1.76
s1494	6	0	1.00	5.21	0	1.00	7.75	12000	2.00	3.01	10090	3.52	12.66	3.52	3.52	1.76
s1423	74	262087	9.19	226.22	266530	14.33	152.20	262081	9.19	29.07	144415	37.10	105.32	4.04	2.59	4.04

TABLE II
EXPERIMENTAL RESULTS AND RESTORATION RATIO COMPARISON (16 NODES).

		gradual[6]			**greedy**[4]			**cover**[7]			**pruning**(proposed)			$\triangle r$ ($\times$)		
name	**#DFF**	**#RS**	r	**time(s)**	**#RS**	r	**time(s)**	**#RS**	r	**time(s)**	**#RS**	r	**time(s)**	**[6]**	**[4]**	**[7]**
s382	21	19997	1.31	4.31	55977	2.99	2.37	59986	3.49	0.39	40356	11.09	1.01	8.47	3.71	3.18
s641	19	11999	1.19	15.99	56107	4.51	7.04	50995	7.37	2.25	36321	10.08	1.13	8.47	2.24	1.37
s1196	18	7997	1.12	7.22	7999	1.12	8.39	52164	5.35	0.96	33692	9.42	7.08	8.41	8.41	1.76
s1238	18	7997	1.12	9.72	7996	1.12	7.45	52164	5.34	1.10	33692	9.42	13.73	8.41	8.41	1.76
s1423	74	230318	4.59	357.53	246605	21.55	90.88	258111	8.17	27.29	144415	37.10	80.39	8.08	1.72	4.54

Finally, the last three columns presents the ratio improvements over the previous algorithms.

Table I and Table II show that the proposed algorithm can bring a higher restoration ratio than the previous methods, about $2 \sim 4$ times. Since the method in [6] and [4] used the greedy selection strategy, their restoration ratio will be lower. Furthermore, when the number of the trace nodes increases, our improvement on the ratio may be higher. However, [7] proposed a novel strategy which focused on the execution time instead of the restoration ratio, so different number of the trace nodes has a lower effect on the results of [7].

V. CONCLUSION

This paper proposes a pruning-based trace signal selection algorithm to improve the restoration ratio for the data acquisition in the post-silicon validation. The final experimental results indicate that the proposed algorithm can get a better restoration ratio than previous methods and it can be used to restore other signal states effectively for the debug.

ACKNOWLEDGEMENTS

This work was supported by National Natural Science Foundation of China under grant NSFC-90207017, NSFC-90607001 and NSFC-60876030; National Basic Research Program of China (973) under grant 2005CB321605; National Postdoctoral Sustentation Fund under grant 023250010.

REFERENCES

[1] D. Gizopoulos. “Advances in Electronic Testing: Challenges and Methodologies”. *Publisher: Springer*, ISBN-10: 0387294082, ISBN-13: 978-0387294087, Jan. 2006.

[2] K. H. Chang, I. L. Markov, V. Bertacco. “Current Landscape in Design and Verification”. *Lecture Notes in Electrical Engineering*, 10.1007/978-1-4020-9365-4-2, 2008.

[3] D. D. Josephson. “The Manic Depression of Microprocessor Debug”. *Proc. IEEE International Test Conference (ITC)*, pp. 657-663, 2002.

[4] X. Liu and Q. Xu. “Trace Signal Selection for Visibility Enhancement in Post-Silicon Validation”. *Proc. DATE*, pp. 1338-1343, 2009.

[5] Q. Xu and X. Liu. “On Signal Tracing in Post-Silicon Validation”. *Proc. ASP-DAC*, pp. 262-267, 2010.

[6] H. F. Ko, N. Nicolici. “Automated Trace Signals Identification and State Restoration for Improving Observability in Post-Silicon Validation”. *Proc. DATE’08*, 2008.

[7] H. F. Ko, N. Nicolici. “Algorithms for state restoration and trace-signal selection for data acquisition in silicon debug”. *IEEE Trans. CAD*, vol.28, no.2, 2009.

[8] T. Larrabee. “Test pattern generation using Boolean satisfiability”. *IEEE Trans. CAD*, pp. 4-15, 1992.

[9] F. Brglez and et al. “Combinational profiles of sequential benchmark circuits”. *IEEE ISCAS*, pp. 1929-1934, 1989.