Trace Signal Selection Methods for Post Silicon Debugging

Shridhar Chaudhary[1,a)] Amir Masoud Gharehbaghi[1,b)] Takeshi Matsumoto[3,c)]
Masahiro Fujita[2,d)]

Abstract: In post-silicon debugging, only a limited number of states (flip-flops) can be traced, due to the area overhead that is introduced by trace buffers. Therefore, it is important to select the states which can restore most of the other states. There exist researches that try to heuristically select a set of flip-flops (FFs) which maximizes the number of restored FFs. Those existing works are not so robust and the cost functions used for selections do not work well in some cases. In this paper, we introduce a hardware based implementation which tries to improve selection by repeatedly swapping the flip-flops to be traced. As this is faster than software by 3-4 orders of magnitude, we can swap much more times and get consistent results even for large circuits.

1. Introduction

As integrated circuit technology has been closely following Moore's law, the complexity of silicon chips is increasing intensively. While the design cycle is decreasing. Because of shorter design cycle and increased complexity, it has become very hard to analyze the functional bugs during pre-silicon phase of verification. This result in errors being escaped the pre-silicon phase. The bugs which exist deep into design state space are very hard to detect during the manufacturing tests as well. There are also some electrical bugs that are usually hard to detect. The traditional pre-silicon verification techniques such as formal verification and simulation based techniques are not efficient to identify these functional bugs as well as electrical bugs. As a result,post silicon validation methods have come into existence.

Post-silicon debugging tries to identify, locate and correct the bugs that exist in the chip after manufacturing to stop the buggy chip to go in the field.

1.1 Related Work

Mainly the post silicon debugging can be divided into two approaches.

- Bug localization techniques that try to automatically find cause of an observed erroneous state.
- Design for debug techniques that improve observability and controllability of design.

One of the approach to locate errors is that we start from an erroneouse state of the circuit and try to find the traces that causes the corresponding erroneouse state by going back to the first corresponding error free state. This is called Backspace technique[3] which basically records a crashed state, then calculates its error free state, which is the target point. By repeating the previous steps on the new target point it reconstructs error trace backwardly. This method has shown promising results by reconstructing traces for hundreds of cycles, but it has high area overhead for the special on-chip hardware. Obtaining the previous state is represented as a SAT problem. SAT solvers are used to produce trace for such bugs that are detected after thousands of cycles of an execution. Furthermore, this trace based techniques has a key issue that is to improve the utility of trace analysis.

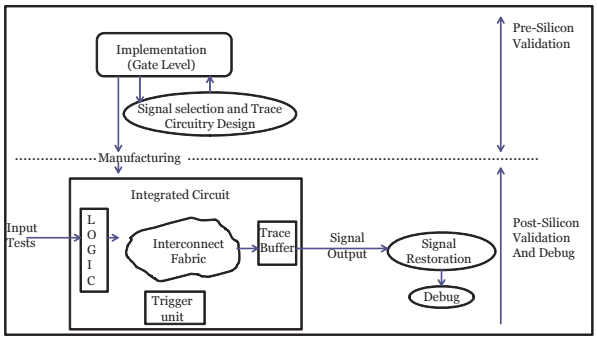

Fig. 1 Overview of system validation [5].

There are techniques that do not require any system level simulation and failure reproduction. For example, a technique called IFRA, Instruction Footprint Recording and Analysis [2] has been developed for localizing bugs and finding the instruction sequence that exposes the bug from a system failure, for instance a crash. A special hardware which records the instruction footprints,which contains special information about the flow of instructions, and what the instructions did as they passed through various micro-architectural blocks of the processor. During normal operation of processor, when a failure is detected, the

[1] Dept. of Electrical Engineering and Information Systems, The University of Tokyo, Tokyo, JAPAN
[2] VLSI Design and Education Center, The University of Tokyo, Tokyo, JAPAN
[3] Ishikawa National College of Technology, Ishikawa, JAPAN
[a)] shridhar@cad.t.u-tokyo.ac.jp
[b)] amir@cad.t.u-tokyo.ac.jp
[c)] matsumoto@ishikawa-nct.ac.jp
[d)] fujita@ee.t.u-tokyo.ac.jp

recorded information is scanned out and analyzed for bug location.Trace based approaches require large amount of internal signal data to be analyzed. However many of this data is irrelevant for reproducing the error and increases the processing time and on-chip memory. Therefore, design for debug techniques are proposed to improve the controllability and observability of internal signals.

DSC (Dynamic Slicing Circuit)[13]is an on-chip circuitry that identifies input signal values influencing the buggy output value in a particular execution of a chip by exploiting dependencies between different signals. As these input signal values are smaller subset of entire input sequence, error can be reproduced by simulation of these input patterns. But this approach assumes that the sequence of control states is stored in an on-chip buffer, that may not be sufficient to store the required number of sequences to get the input values to reproduce the error.

To improve the internal signal observability, special ad hoc DFD hardwares are proposed in the literature that improve data acquisition[8][9][10][11]. They are either trace buffer based or scan based approaches. Fig 1 shows the flow of design in case of using trace buffers inside the chip. Most recently special hardware called ELA (Embedded Logic Analyzer)[7] are used on-chip which samples the data into trace buffers. An ELA has many trigger units which determines when the data should be sampled into trace buffers. These DFD hardware utilizes the sampled data by sending it through low bandwidth device pins and finally the errors can be identified off-chip using some special post processing algorithms.

1.2 Motivation

All the above techniques are trying to utilize the on-chip stored data. Similarly among DFD techniques like DSC or ELA the trace buffers area overhead is a problem. To tackle this issue efficient use of the on-chip stored data has become a key in post-silicon debugging. The on-chip stored data which is traced should be efficiently used to reconstruct the circuit state as much as possible. In our previous work, for a specific simulation vector, we introduced a way to select the optimum number of traced states using a PBO (Pseudo Boolean Optimization) based algorithm which can optimize the on-chip memory by restoring the unknown states from the fewer number of traced states stored in the on-chip buffer. Basically, we try to select some particular signals for tracing instead of tracing all the values, following some algorithms trying to reproduce other states of the design. Logical bugs can be detected using restorability based techniques. We can get optimum FF selections for small circuits with a single simulation vector by utilizing PBO techniques.

From the experiments of PBO we observed that there are cases where existing simulation based methods do not work well. That is because although the selection of FFs done by cost functions of previous X-smiluations based methods is consistently correct if FFs are fixed. However, the cost function using incremental or decremental selection of FFs may not necessarily be a good criteria. In this paper, we show that the correct cost functions cannot be compensated neither by a method with two FF additions nor a method with a cost function including interval signal

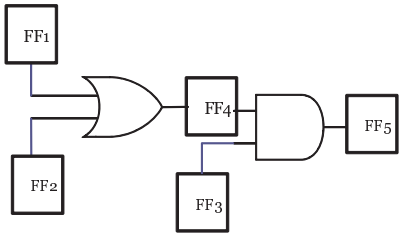

Fig. 2 Example circuit.

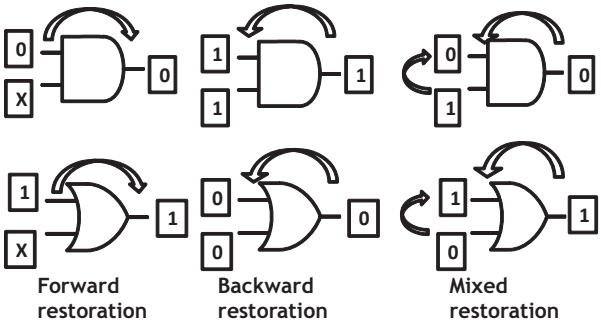

Fig. 3 Restoration examples.

values. Therefore, we need to fix the FF selection for all slots in order for the cost function to be correct or effective. We introduce a swap based signal selection process using FPGA based hardware implementiation and compare it with an software implementation. We have shown that swapping can improve signal selection from existing simulation based methods but many swappings are needed for large circuits and software take excessive time for these large circuits. Therefore, we use FPGA to perform around 1000 times more swaps than software. We also show a functional dependency based heuristics which we tried for the analysis. Finally, we will conclude this paper showing our estimated improvement in number of swaps using hardware implementation which is still under progress.

2. Signal Restorability

In post-silicon debugging, ideally we want to observe every signal value in each cycle, while utilizing little chip area and consuming lesser time. With increasing logic designs in limited area, it is unrealistic to observe each and every state of the signal at every cycle. While we use trace buffers to store these signals we want to decrease the amount of on-chip memory used by them. Therefore, it is better to trace only limited number of signal values and by using some techniques, trying to restore other unknown values. Restoration process utilizes the controlling value of a logic gate[14]. A controlling value at one of the input of a logic gate can be used to infer the value of some or all inputs of the corresponding gate. For example,value 1 on an input of OR gate directly conveys that the value of the output is 1. Similarly, if we know one of the inputs and the output value of a 2 input gate, we can infer the other input value. Using similar technique for other logic gates we can restore unknown input states and output states for different cycles. The restoration can be divided into three types. All these three kinds of restoration can be seen in Fig.3 which represent each type separately.

Table 1 Restoration performed for the example circuit.

	1	2	3	4	5
FF1	0	x	0	x	x
FF2	0	x	0	x	x
FF3	x	x	x	x	x
FF4	0	0	1	0	x
FF5	x	0	0	x	0

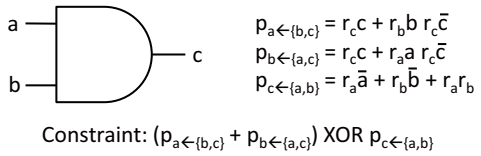

Fig. 4 Formulation example for AND gate.

- *Forward restoration: When one of the inputs of a gate has controlling value, the output can be inferred without knowing other input values*
- *Backward restoration: When the output of a logic gate has the non-controlling value, we can infer that all the inputs have that non-controlling value.*
- *Combined restoration: When we know one input as well as the output of a 2 input gate, we can infer the other input from these values.*

As shown in Table 1, restoration process performed on the example circuit shown in Fig.2, restoration depends on the the signals that are traced. Therefore, the trace signals should be carefully selected to get the optimum result. Which signals and how many signals to be traced has been a research topic and many algorithms to select the best signals are proposed. For evaluating the quality of restoration, State Restoration Ratio (SRR) is defined as: $\frac{sum\,of\,number\,of\,traced + number\,of\,restored\,states}{number\,of\,traced\,states}$. In the example circuit Fig.2, we can see that FF2 is traced for 4 cycles and we were able to restore 7 states. So SRR for this circuit equals to (7+4)/4 = 2.75.

3. PBO Formulation

Based on the restoration technique mentioned above, we have presented a PBO-based formulation for the selecting optimum number of traced signals for a given circuit. The pseudo-Boolean optimization problem is the task of finding a satisfying assignment to a set of PB-constraints that minimizes a given objective function. We introduce four Boolean variables $r_s(c), tr_s, p_{s\leftarrow t}(c)$ to have logical representation of restorability[16]. $r_s(c)$ is 1 iff signal s is known (i.e. restored or traced) at c-th clock cycle. tr_s is defined only for state variables and indicates s is traced. $p_{s\leftarrow t}(c)$ is 1 iff signal s is restored using the value of signal set $t = t1, t2, ...$ at c-th cycle. Variables $p_{s\leftarrow t}$ are defined to represent restoration inside a logic gate, where each of s and $t1, t2, ... \in t$ are an input or output of a gate. An example of how p variables are defined is illustrated in Fig. Then, r_s is defined as OR of all p variables to restore s and tr_s (if s is a state variable). Note that p variables to restore s can exist in different gates, since s can be restored both forwardly in a fanin gate of s and backwardly in fanout gates of s. The logic formulas generated in the way described above are translated into PBO formulas. We made a PBO problem to find the optimum number of flip-flops which can be traced for maximizing the number of state restored in the circuit.

Algorithm 1 Simulation Based Selection Process

```
1:procedure SelectSignals(circuit,w,c)
2:      Create list of selected signals S => Initial selection
3:      while |S| < w do
4:         Gererate a random input vector I
5:         for every pair of FFs those are not in S do
6:                  Calculate restoration difference
7:         end for
8:         Find FFs with maximum restoration difference
9:         Add those 2 FFs to the list S
10:     end while
11:     return S
12: end procedure
```

Fig. 5 Simulation based selection process.

4. Signal Selection Heuristics

We used PBO-based method's result to analyze X-simulation based methods and it showed that cost function for selection in X-simulation based methods doesn't work well. The state of the art simulation-based method is an augmentation based method which selects flip-flops incrementally based on which flip-flops restores maximum number of state[20]. Following, we present our heuristics for signal selection.

4.1 Selecting 2 FFs Incrementally At Once

Selecting FFs incrementally using X-simulations[20] is proved to be better than decremental selection[14]. In the incremental approach, initial pool of selected FFs is empty. From there on, at every step, one FF is being traced and the number of restored states are checked based on the traced FFs. Repeating this procedure for all FFs, the FF with highest number of restored states is selected. The above step is repeated until the trace buffer width, which is the number of FFs to be traced, is full. We implemented the same algorithm and compared the results with the optimum selection from PBO-based method. We observed that in some cases during the incremental selection of FFs at the second step or further steps, a wrong FF was selected in the sense that final pool of selection was not optimum.

Therefore, as shown in the Fig.5, we changed the selection process by selecting 2 FFs at a step instead of one in the X-simulations based method presented in the previous works[20]. Experimental results showed that even selecting 2 FFs at a single step finally selects wrong selection because the number of traced FFs didn't change too much even for the new selection. That lead us to the conclusion that every position of FFs during the selection is needed to be fixed for the best selection of FFs.

4.2 Selection Based On Cost Function With Intermediate Variables

So far in the existing works based on simulation, we just count the number of restored FFs. As in every circuit there are intermediate signals between different FFs, those intermediate signals may affect the restoration of connected FFs. Therefore, we considered intermediate variables as well, while counting the total number of restored states to improve selection of FFs.

Basically cost function in previous works just uses number of

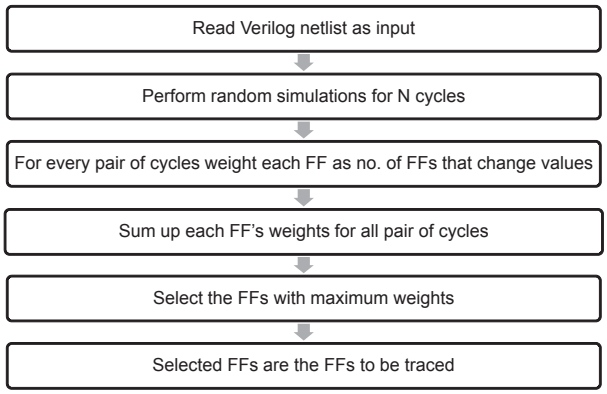

Fig. 6 Signal selection process.

restored FFs for evaluation but we included restored intermediate signals as well, however, the results after the inclusion of intermediate variables in the cost function evaluation didn't change the final selection as expected.

4.3 Functional-Dependency Based Selection

From the results of previous tried heuristics, we concluded that the heuristics, that are based on X-simulation and use incremental and decremental selection are not working well to get optimum selection of FFs, because the cost function might not be a good criteria for some cases and every position of FF selected needs to be fixed. Therefore, we came up with a new cost function that is completely different from the previous works. In our proposed heuristic, selection of flip-flops is based on functional dependency among the FF signal values. As the value of a FF changes in a given pair of cycles, it might affect other FFs by changing their values in the same pair because ultimately they are connected with each other with some logic gates in between. Therefore, this functional dependency among different FF values can be a key to a new cost function. Function-dependency based cost function can be defined as :

- *When a signal's values in two cycles(time frames) are different, what are the other signals that their values are different.*
- *For each two time frames, count the number of signals whose values are different.*
- *The proposed cost function is the summation of the above for all combinations of two time frames*

We select FFs with the highest value of the above described cost function to be traced. The signal selection process can be seen in the Fig.6.

5. Swapping Based Selection Improvement

As can be seen in the experimental results section, all the heuristics shown above couldn't improve the final FF selection too much. However, based on PBO formulation we know that there is a scope for improvement. Therefore, we proposed to start with an initial selection of best FFs as per existing simulation based methods and try to improve from there. For improving the FF selection,we introduce a swapping based heuristic. Basically we start with an initial list of selected FFs for restoration and a list of remaining FFs. Then we randomly swap FFs between the two lists and check wether the number of restored states are improved or not. We accept the swapping if the number of restored FFs is increased and continue swapping until we can not improve the restoration amount further.

Algorithm 2 Swap Based Signal Selection Process

```
1:procedure SelectFFs(circuit,w(TraceBufferWidth),c(cycle))
2:      Create list of selected FFs S => Initial selection
3:      Create  list of remaining FFs  R
4:      Calculate # of restored FFs for S => N
5:      for (10000 times) do
6:          Swap 1 FF between S and R
7:          Calculate # of restored FFs for S => N_new
8:          if (N_new > N) accept the swap
9:          else  re-swap the selection
10:     end for
11:     return S
12: end procedure
```

Fig. 7 Swap based signal selection process.

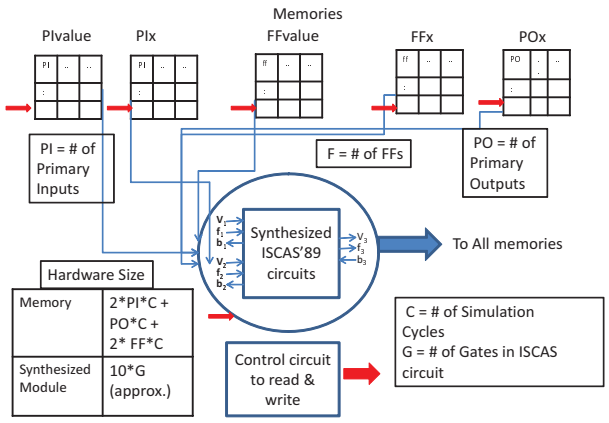

Fig. 8 Swap based selection hardware.

5.1 Software Implementation

We performed a software implementation of the swapping based selection. As shown in Fig.7, algorithm for swapping accepts only improved selection. To reach the the best selection for large circuits with thousands of FFs, too many swaps are required to reach to the best selection, hence not feasible with software method.

5.2 Hardware Implementation

As we need as many as possible swaps for larger circuits, we introduce a hardware implementation of swapping based signal selection. As shown in Fig.8, we introduce v, f, b variables for simulation value, forward restoration and backward restoration, respectively. Synthesized circuit module contains all the formulas for v, f and b variables similar to our PBO formulas as shown in Fig.4. These formulas are used for x-simulation of restoration through the circuit. f and b variables are basically used as x values for simulation. We introduce 5 memories each for primary input value, x value as PI and PIx respectively, primary output x value as POx, FF value and FF x value as FFv and FFx respectively. Width of these memories are the number of variable they represent. For example, FFv memory width is the number of FFs in the circuit. Depth of these memories is same as number of

cycles. In Fig.8 size of the hardware is represented. These memories are connected with the synthesized module through a control circuitry. Current synthesized ISCAS'89 circuit size is more than 10 times the original size. That is because, we have introduced three kinds of variables which in turn are synthesized into more number of gates.

At the start of the process, we give an initial selection vector to FFx memory while whole PI memory and FFv's 1st address is randomly initialized. Then x-simulation is performed in hardware for restoration by repeating forward and backward restoration until we reach to the maximum number of possible restored FFs. After restoration is finished, FFx memory is read to count the number of FFs restored and a pre-processor will decide that swapping should be performed similar to software implemented algorithm shown in Fig.7.

6. Experimental Results

Table 2 Software swapping improvement.

circuit	# at start	# of effective Swaps	# of restored
s298	802	3	850
s344	396	8	576
s349	396	7	590
s386	381	0	381
s444	929	6	1254
s510	384	0	384
s526	1269	1	1278
s713	754	2	821

We applied our PBO formulation and all the three heuristics explained in section 4 to ISCAS'89 circuits and obtained the trace signal to be traced. PBO based method worked only for smaller circuits because of scalability issues of SAT solver, we compared it with our own implementation of existing simulation based method, which selects FFs incrementally. In this experiment for small ISCAS'89 circuits trace buffer size was fixed as 4 because the total number of flip-flops in these benchmarks were very small. We could get optimum FF selections for small circuits with a single simulation vector by utilizing PBO techniques. For further experiments, we implemented one of the algorithms shown in existing simulation based method with incremental selection. In Table 3, Sim-exist shows the SRR for the implementation. We used 100 sets of random input vectors for the evaluation of restoration experiments and comparison of our results with the existing simulation based approach which was implemented with the same set of 100 random input vectors. The results shows that PBO gives better restoration in some of the cases excluding s400,s510 and s832 where it was almost equal but not less. For comparison of our proposed heuristics we implemented simulation based approach[20]. Simulation based approach provides better SRR compared to all the existing solutions. However, in some cases specially for small buffer width of 4, SRR was reduced because of fewer number of FFs after optimization using synthesis tools. When random inputs are used, according to[14], restoration capability can be obtained by actually simulating the restoration process on the circuit over a small number of cycles 64, and measuring the corresponding SRR. As trace signal selection is done only once during the design flow of circuit blocks of ELA, the run time of selection algorithm is less important than the quality of the selected signals[14].

Table 3 Experiment for initial heuristics.

circuit	# of FFs	Sim-exist	Sim1	Sim2	Fun-Dep	PBO
s298	14	4.23	4.23	3.91	3.16	4.4
s344	15	3.24	3.31	3.13	2.73	4.14
s386	6	2.48	2.48	2.48	2.25	2.5
s444	21	4.81	5.78	3.76	3.81	6.05
s510	6	2.49	2.48	2.49	2.38	2.5
s526	21	5.95	5.96	3.74	3.71	6.04
s713	19	3.97	3.97	3.97	2.5	4.01
s832	5	2.15	2.15	2.06	2.14	2.17
s1423	74	7.8	T/O	6.8	5.56	T/O
s5378	179	15.3	T/O	15.1	12.1	T/O
s9234	211	10	T/O	9.59	5.18	T/O
s15850	638	27.8	T/O	T/O	23.8	T/O

Table 3 column 4 Sim1 shows the SRR for 2FFs at once heuristic with random vector for selection and 100 sets of random vectors for the evaluation of restoration. The same sets of 100 random vectors were applied for comparison with the selection which is the same as done by the existing simulation based approach. The number of cycles and the trace buffer width was fixed to 4 for the smaller ISCAS'89 circuits. Comparison was done based on the average of the 100 sets of restored FFs. As the experiments could not finish for large circuits comparison can be seen in the Table 4 for smaller circuits only. We can see there is not much improvement or even the restoration decreases in some cases. This shows that every position while selecting FFs should be fixed for optimum selection.Similarly we compared the intermediate variable included heuristic's results, shown as Sim2 in Table 3, with the existing method's selection using 100 random vectors for evaluation of restoration process. For functional dependency based heuristic we performed different selections using 5 sets of random vectors and chose the best for comparison. Fun-Dep shown in Table 3 represents the SRR for function dependency based heuristic's result. Comparison of our heuristic was done with the simulation based method for 100 sets of different random vectors used for restoration based evaluation.

6.1 Experimental Calculations For Swap Based Selection Improvement

Table 4 Experiment comparing hardware and software swaps

circuit	# of Gates	# of FFs	# of Swaps Software (per hour)	# of Swaps Hardware (per hour)	Ratio
s5378	2779	179	10000	9309662	931
s9234	5597	211	6428	6209048	966
s15850	7951	638	3461	1821386	526
s38417	22179	1636	538	1136754	2113

As shown in the Table 4,comparing column 4 with column 2 shows that, software based swapping could improve the initial selection of FFs in smaller ISCAS'89 circuits. The effective number of swaps shown in the table implies to the swaps which improve the selection. All the results are shown after repeating the swapping algorithm for 1000 times which means a large number of swaps are required to get effective swaps. It can be seen that as

we try swapping for large ISCAS'89 circuits the number of effective swaps decreases. As software could take forever for larger circuits, we show estimated improvement in the number of swaps using our FPGA based hardware in Table 4. Results shown are based on frequencies of our synthesized modules as discussed in Fig. 8. The time hardware takes for one swap can be calculated for a particular circuit based on following formulas. If C shows the number of simulation cycles and Cres shows the number of clock cycles required for one complete process of restortaion,

Clock cycles for one swap(CC) = (FF number X C) + Cres

Time for one swap = $\frac{1}{FPGA clock frequency for the circuit}$ X CC

Using above mentioned formulas Tabel 4 shows the comparison between the estimated number of swaps that can be performed in one hour using software and hardware implementations. From these results, we can say that the larger the circuits gets more the number of swaps performed by hardware increases compared to the software. Hardware as expected can perform more than 1000 times more swaps than software for largest ISCAS'89 circuit which in turn is expected to get better selection. So using hardware swapping we expect to achieve improvement in signal selection even for circuits bigger than ISCAS'89.

7. Conclusions and Future Work

Post silicon validation has become an essential step in the design flow of integrated circuits. In this paper we presented different heuristics, that use trace buffers and restoration methods to select effective signals which can reconstruct other signals of the circuit while consuming less hardware memory. While the approaches like simulation based are very effective than other previous probabilistic based techniques all the methods seen so far are not exact. There is a need for more optimal and less time consuming methods. Although our PBO formulation can give better results for 1 random simulation but it can work only for smaller circuits due to exponential runtime during simulation. Through experiments after modifying existing simulation based approach we came to know that signal selection can be improved by fixing FFs selected during each step of simulation based method. We did so by repeated local optimization on an early selection using swapping.

We have done experiments using swapping on software but it cost many simulation cycles and excessive runtime. Therefore we introduced a FPGA based swapping which can be thousand times faster than software and give us optimum results even for very large circuits because of high resources available in FPGA. We have formulated the restoration into FPGA based hardware and plan to run the experiments of swapping using a software controller with FPGA hardware in the future research.

References

[1] N. Nataraj, T. Lundquist, and K. Shah, "Fault localization using time resolved photon emission and STIL waveforms," *Proc. International Test Conference, 2003*, pp. 254–263.

[2] S.-B. Park and S. Mitra, "IFRA: Instruction Footprint Recording and Analysis for Post-silicon Bug Localization in Processors," *Proc. of Design Automation Conference*, pp. 373–378, 2008.

[3] F. M. De Paula, M. Gort, A. J. Hu, S. Wilton, and Y. Jin, "BackSpace: Formal Analysis for Post-Silicon Debug," *Proc. of Formal Methods in Computer-Aided Design*, pp. 1–10, 2008.

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

[5] K. Basu and P. Mishra, "Efficient trace signal selection for post silicon validation and debug," *Proc. VLSI Design*, 2011, pp. 352–357.

[6] H. F. Ko and N. Nicolici, "Automated trace signals identification and state restoration for improving observability in post-silicon validation," *Proc. Design Automation and Test in Europe*, 2008, pp. 1298–1303.

[7] M. Abramovici, P. Bradley, K. Dwarakanath, P. Levin, G. Memmi, and D. Miller, "A reconfigurable design-for-debug infrastructure for SoCs," *Proc. Design Automation Conference (DAC)*, 2006, pp. 7–12.

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

[9] ChipScope Pro, Xilinx Verification Tool, 2006, http://www.xilinx.com/ise/optional prod/cspro.html.

[10] Sun Microsystems OpenSPARC, http://opensparc.net/.

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

[12] B. Vermeulen and S. K. Goel, "Design for Debug: Catching Design Errors in Digital Chips," *IEEE Design and Test of Computers*, vol. 19, no. 3, pp. 35–43, May 2002.

[13] Yeonbok Lee, Takeshi Matsumoto, Masahiro Fujita, "Generation of I/O Sequences for a High-level Design from Those in Post-silicon for Efficient Post-silicon Debugging," *Proc. of 28th IEEE International Conference on Computer Design*, pp. 402–408, October 2010.

[14] Debapriya Chatterjee, Calvin MacCarter, Valeria Bertacco, "Simulation-based signal selection for state restoration in silicon debug," *Proc. of the International Conference on Computer-Aided Design*, pp. 595–601, Nov. 2011.

[15] H. F. Ko and N. Nicolici, "Algorithms for state restoration and tracesignal selection for data acquisition in silicon debug," *IEEE Trans. on CAD*, vol. 28, no. 2, pp. 285?297, 2009.

[16] Shridhar Choudhary, Kousuke Oshima, Amir Masoud Gharehbaghi, Takeshi Matsumoto and Masahiro Fujita, "Exact Solution for Trace Signal Selection with Pseudo Boolean Optimization (PBO)," Design Automation for Understanding Hardware Designs(DUHDe), 2014.

[17] Xiao Liu and Qiang Xu,"On Signal Selection for Visibility Enhancement in Trace-Based Post-Silicon Validation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol.31, no.8, pp.1263–1274, Aug. 2012.

[18] Kang ZHAO and Jinian BIAN, "Pruning-Based Trace Signal Selection Algorithm for Data Acquisition in Post-Silicon Validation," *IEICE TRANS. FUNDAMENTALS*, VOL.E95-A, NO.6 pp. 1030–1040, Jun. 2012.

[19] N. Eén and N Sörensson, "Translating Pseudo-Boolean Constraints into SAT," *Journal on Satisfiability, Boolean Modeling and Computation*, vol. 2, pp.1–26, 2006.

[20] Rahmani,K.Mishra, P.Ray, S., "Efficient trace signal selection using augmentation and ILP techniques," *Quality Electronic Design (ISQED), 2014 15th International Symposium on* , vol., no., pp.148,155, 3-5 March 2014.