University of Cincinnati

Date: 5/2/2016

I, Prabanjan Komari, hereby submit this original work as part of the requirements for the degree of Master of Science in Electrical Engineering.

It is entitled:

A Novel Simulation Based Approach for Trace Signal Selection in Silicon Debug

Student's name: **Prabanjan Komari**

This work and its defense approved by:

Committee chair: Ranganadha Vemuri, Ph.D.

Committee member: Wen-Ben Jone, Ph.D.

Committee member: Carla Purdy, Ph.D.

21024

A Novel Simulation Based Approach for Trace Signal Selection in Silicon Debug

A Thesis submitted to the
Graduate School
of University of Cincinnati
in partial fulfillments of the
requirements for the degree of

Master of Science

In the Department of Electrical Engineering and Computing Systems
in the College of Engineering and Applied Sciences
By

Prabanjan Komari
BE, Visvesvaraya Technological University, Belgaum, India.
July 2013

Thesis Advisor and Committee Chair: Ranga Vemuri, PhD

Abstract

With the fabrication technology fast approaching 7nm, Post-silicon validation has become an integral part of integrated circuit design to capture and eliminate functional bugs that escape pre-silicon validation. The major roadblock in post-silicon functional verification is limited observability of internal signals in a design. A possible solution to address this roadblock is to make use of embedded memories on chip called trace buffers. The amount of debug data that can be acquired from the trace buffer depends on its width and depth. The width of the trace buffer limits the number of signals that can be traced and the depth of the trace buffer limits the number of samples that can be acquired. Using the acquired data from the trace buffer, the values of other nodes in the circuit can be reconstructed. These trace buffers have limited area, hence only a few critical signals can be recorded by it. In this work we used the simulated annealing heuristic to select trace signals. We developed this idea from the fact that trace signal selection can basically be viewed as a bi-partitioning problem, the set of flip-flops being tapped onto the trace buffer is one partition and the other partition is the set of all other flip-flops in the design. Another key contribution of this thesis is that we found and fixed a hole in the established state restoration algorithm. Experimental results demonstrate that our approach can provide significantly better restoration ratio compared to the state-of-the-art techniques.

To my loving parents who never stopped believing in me

Acknowledgement

Firstly, I would like to thank my parents for encouraging and supporting me in each and every endeavor. Their unwavering belief in me, inspired me to work relentlessly towards my goals. Next I would like to thank my thesis advisor Dr. Ranga Vemuri, He is an excellent teacher and an even better research advisor. It was an absolute privilege to work with him and be a part of DDEL. I will always be proud of the fact that I got a chance to work with him.

I would like to thank Dr. Wen Ben Jone and Dr. Carla Purdy for being a part of my committee. I would also like to thank Rob Montjoy for his help with various tools and infrastructure required for completing this thesis. I also thank Xiaobang for reviewing my thesis and providing me valuable suggestions.

I would like to thank my friend Ananthakrishnan for giving me valuable advice on various issues that popped up during my work. His help enabled me to complete my work efficiently, especially with the documentation and thesis defense presentation.

I would like to thank my friends Meera, Ujwal and Sriram for helping me with various issues that cropped up during the work done on this thesis. I would also like to thank Rao Lakamsani from Intel for his help on this thesis, which enabled me to clear a particularly difficult roadblock.

I would also like to thank my uncle M.S Suresh for his invaluable support throughout my student life. Special thanks to my cousin Kiran for helping me with the lengthy process of admission into U.S universities.

Finally, I would like to thank all my friends for always supporting me.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

The trend of scaling in the VLSI field has led to designs with multi million transistors. Due to this trend, functional verification of designs in the pre silicon stage is no longer foolproof and there is a possibility of some bugs escaping into the post silicon stage. The aim of our work is to provide a novel approach to help designers with functional verification in the post silicon stage. To understand our approach, it is first essential to understand the conventional methods of verification. Sections 1.1, 1.2 and 1.3 will describe the verification techniques; pre silicon verification, manufacturing test and post silicon validation.

1.1 Pre Silicon Verification

The objective of pre silicon verification is to verify the correctness and sufficiency of the design. There are two main techniques of pre silicon verification these are simulation and formal verification. The main difference between simulation and formal verification is that the former requires the use of input vectors while the latter does not. In simulation-based verification, the idea is to first generate input vectors and then to obtain reference outputs. However, in the case of formal verification the user specifies the output behavior and then lets the formal checker prove or disprove it. The main advantage of the formal based approach is completeness, as it does not miss any point in the input space, which is a major drawback in the case of simulation-based approach.

Thus, simulation techniques are now using testbenches that drive the design-under-test with constrained-random or coverage-driven input stimuli. These testbenches target to verify a design only up to an acceptable simulation coverage.

Formal verification is the act of proving the correctness of intended algorithms underlying a system with respect to a certain formal specification or property, using formal methods of mathematics [1].

There are two different types of formal verification, formal equivalence checking and formal property checking [2].

In formal equivalence checking, a design is compared to a golden reference and the formal checker concludes if both the designs are functionally equivalent. Here are some examples where formal equivalence checking is used,

1) RTL versus pre routed netlist
2) Pre routed netlist versus post routed netlist
3) Netlist versus ECO (Engineering change order) netlist

Formal property checking is a method by which the correctness of design or the root cause of an error is identified by rigorous mathematical proofs.

Properties are primarily used to validate the behavior of a design and can be checked statically by property checker tool, and proves whether a design meets its specifications.

1.2 Manufacturing Test

The purpose of manufacturing tests is to ensure that the product hardware has no defects caused due to manufacturing that could adversely affect its performance. Usually manufacturing faults are shorts between two conductors or opens in a conductor. This type of behavior is very difficult to model, a simple model has been proposed in literature

called single stuck at fault model, which assumes that all nodes are stuck at 0 or 1 (shorted to GND or VDD). This assumption is not quite true but works well in practice. Ideally, in manufacturing tests it has to be proved that each node in the circuit is not stuck. Ideally, smallest sequence of test vectors has to be applied in order to prove that a node is not stuck. Two factors called controllability and observability are needed to determine the number of test vectors required for manufacturing test. Controllability is the relative difficulty of setting a line to a value. While observability is the relative difficulty of propagating an error from a line to a primary output. Good controllability and observability reduces number of test vectors required for manufacturing test. To increase controllability and observability of circuits scan chains are used. Scan chains provide a simple way to set and observe every flip-flop in an integrated circuit. The flops in the circuit have to modified in order to use scan chains, the D input has to be multiplexed with the scan input and a signal called scan enable is the select line for the multiplexer.

1.3 Post Silicon Validation

Post silicon validation is the process of operating the manufactured chips in their actual environment to find out if they are operating according to their specification.

Post silicon validation has four major steps [3]

- Identifying a problem: By running a sequence of random instructions or by executing the end user application. A bug may appear while doing so.
- Localizing the problem: The problem has to be localized to a small region from the system failure.

- Finding the root cause of the problem: The reason for the occurrence of the problem has to be found out at this stage.
- Finding a solution to the problem: A solution, which would fix the problem, has to be implemented.

The following are reasons as to why post silicon validation is essential even though pre silicon verification and manufacturing test techniques are present

- The actual silicon is several orders of magnitude faster than simulation. Hence, there is a strong possibility of bugs being detected at this stage, which were not detected at the pre silicon stage.
- Accurate modeling of electrical bugs like cross talk and power supply noise is very difficult in the pre silicon stage.
- Unlike manufacturing faults, post silicon bugs may be caused due to subtle interactions between the design and electrical faults. It is very difficult to create an accurate model for such bugs.

The table given in the next page qualitatively compares pre silicon verification, manufacturing testing and post silicon validation [3].

Table. 1.1. Qualitative comparison of Pre silicon verification, Manufacturing testing, Post Silicon Validation [3**].**

Pre Silicon Verification	**Manufacturing testing**	**Post silicon validation**
Full controllability and observability. As any signal can be accessed.	Controllability and observability are primarily due to scan DFT.	Insufficient controllability and observability due to limited access to internal signals.
Complex physical effects are difficult to model.	There are several defect models.	Accounts for signal integrity and process variation.
Simulation for full chip designs extremely slow. Formal verification is not applicable for all cases.	Generally very fast (In the order of few seconds to minutes per chip).	Silicon is several orders of magnitude faster than simulation.
Some established metrics exist (Code coverage, assertion coverage, mutation coverage).	Test metrics like Stuck-at-transition widely used.	Some metrics exist but it is still an open research problem.
Bug fixing inexpensive.	Bug fixing is not the primary objective.	Bug fixing is expensive.

Limited observability of internal signals is a major obstacle in the post silicon validation stage. A number of solutions have been proposed in this area to tackle the problem. The type of error, which one is trying to locate, dictates what information has to be acquired from the design. The following are the different type of errors encountered during the post silicon stage [4],

- Circuit bugs: These are bugs that arise due to circuit mismatch between different levels of abstraction and also effects from the use of deep sub-micron technologies on process variation and signal integrity.
- Logic bugs: The data acquired by the DFD (Design for Debug) hardware can be used to identify logic bugs, which are functional errors that have escaped the pre silicon verification stage. One method for acquiring functional data in silicon is the scan chain based technique. The problem here is that the normal circuit operation is halted and the circuit has to be operated in the scan mode, hence preventing data to be acquired in real time. As functional bugs can span thousands of clock cycles [5] it is essential to keep the circuit working in the normal operation during scan dumps. Hence, to effectively acquire debug data the trace based technique is used which employs on chip memories for at speed data sampling. Details of the trace based technique will be explained later in this thesis.
- System bugs: These type of bugs exist among multiple cores in an SOC. The bugs that occur when a software is running on the system and the different cores are interacting with each other require acquisition of data from the interrelated cores. Hence the Design for Debug hardware has to be different when compared to the hardware required for logic bugs in one core.

1.4 Trace Buffer Technique

As discussed, the scan-based technique is not suited to acquire data for debug in real time as the normal operation of the circuit has to be halted and the circuit has to be operated in the scan mode. To acquire data in real time the trace buffer based technique [6] is used. The debug flow for using this approach is given below [7]

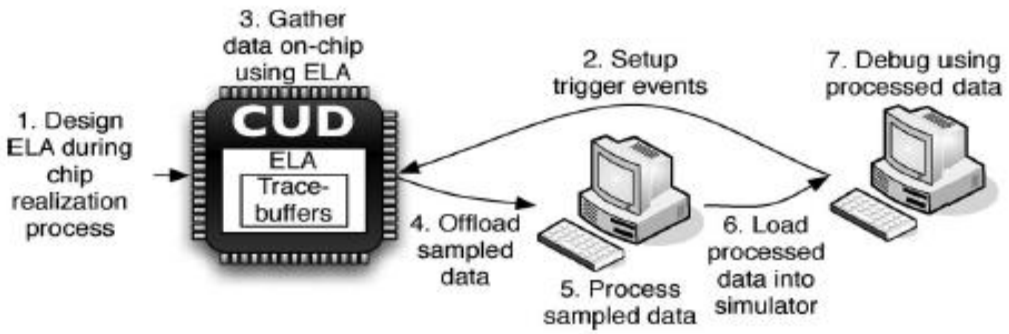
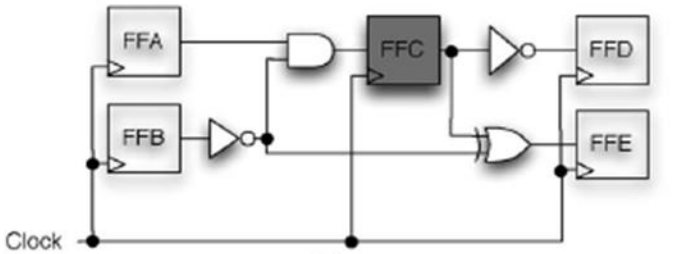


Fig. 1.1. Debug flow for trace buffer based technique [7]**.**

The first step is to design the embedded logic analyzer during the chip realization process. It includes trigger units which determine when data has to be acquired and also the sampling units which records a small set of signals (called trace signals) using trace buffers such as embedded memories on chip. After which the debug engineer controls the trigger events, which determines when the real time debug data is gathered by the embedded logic analyzer. After this, the data is transferred off chip to the post processing stage [8] via a low bandwidth interface. In the next stage, the data is fed to the simulator, where the debug engineer can analyze the data to identify functional bugs. It has been shown in [9] that large designs contain tens to hundreds of bugs. As a result, the debug engineer has to iterate steps 2 through 7 shown in Figure 1.1 to gather all the required data for resolving bugs. The amount of debug data acquired depends on the trace buffer width and depth. The trace buffer width limits the number of signals that can be traced and the trace buffer depth limits the number of samples that can be stored. To reduce the debug time, the number of iterations of steps 2 through 7 has to reduce. In order to achieve this goal, there is a need for better ELA'S, a number of such solutions have been proposed in the literature [10 - 12]. However, as the cost of increased area has to be incurred when the size of the trace buffer is increased, designers are reluctant to increase

the size of trace buffer just for the purpose of silicon debug. Hence a few solutions have been proposed in the literature [13, 14, 15] to compress the debug data present on the chip before it is stored on the trace buffer. Although compression can be used to increase the number of samples stored per trace signal, the number of signals being traced cannot be increased. Hence, there is a need for an automated way to determine the signals being traced, such that maximum data [combinational and sequential nodes] is reconstructed based of the data acquired by the trace buffer. This has to be done in a way that any post-processing algorithm can easily use the enlarged data to identify design bugs in step 7.



(a)

Clock cycles

	0	1	2	3	4
FFA	1	1	X	X	X
FFB	0	0	X	X	X
FFC	0	1	1	0	X
FFD	X	1	0	0	1
FFE	X	1	0	X	X

(b)

Fig 1.2 Example circuit to illustrate the trace buffer technique, (a) Circuit under debug (b) Traced and restored data in sequential elements [7].

In the above Figure we see an example circuit utilizing the trace buffer concept. The trace buffer width is set to 1, implying that only one flip-flop will be connected to the trace buffer. In this example flip-flop C is being traced, we observe that just by recording the values of

flip-flop C at different clock cycles we are able to restore the values of other flip-flops in the circuit. The explanation as to how these values are restored is given in Chapter 2. Another point to be noted here is that the number of signals being restored depends on which flip-flop is being traced, notice that if flip-flop E were to be traced no other signal would be restored. Hence in order to maximize observability at the post silicon stage a clever and automated methodology has to be used to select the trace signals.

1.5 Related Work

Currently the trace signal selection process in the industry is primarily manual. The decision to select signals is guided by the designer's experience and intuition (For example trace signals are selected from hardware blocks that have encountered more bugs during the pre-silicon stage). Due to the lack of techniques for qualifying observability value, the inadequacy of the selected trace signals shows up during silicon debug, usually in the form of observability holes that make it difficult to identify, diagnose and root-cause an observed failure. However, at this stage new trace signals cannot be selected. Inability to observe, validate and debug at the post-silicon stage results in costly escapes or silicon re-spins.

Research in post silicon validation has attempted to address this problem of automating the process of selecting trace signals. The key idea here is to select a set of signals S that maximizes state restorability (the set of states that can be reconstructed by observing S). Most of the work done in this domain [16, 17, 7, and 18] involves defining a metric based of the circuit structure that estimates the state restoration capability of a set of signals and then this estimate is used to converge to a candidate set of trace signals. Chatterjee et al. [19] have designed a simulation based approach that performs better

than the structural based approach, however their approach has drawbacks in restoration quality and also they incur a lot of computational overhead. Li et al. [20] has designed a hybrid approach combining the metric based and the simulation based approaches, however they only make use of simulation for a small set of signals and consequently sacrificing restoration quality. Rahmani et al. [21] has designed an approach based of two components: (1) an iterative approach to signal selection based on mock simulations and (2) a filtering scheme based on Integer-Linear Programming (ILP) to refine the selected set. However, their signal selection algorithm is a greedy algorithm and consequently this affects their restoration quality.

1.6 Thesis Statement

In this work we have developed a novel simulation based approach for selecting trace signals. The popular metric for measuring the quality of a set of trace signals is restoration ratio.

Restoration Ratio = (No. of traced and restored values)/(No. of traced values)

The objective of the work done in this thesis is to maximize the metric restoration ratio. To do this we used the simulated annealing heuristic to select trace signals. We developed this idea from the fact that trace signal selection can basically be viewed as a bi-partitioning problem, the set of flip-flops being tapped onto the trace buffer is one partition and the other partition is the set of all other flip-flops in the design. Another key contribution of this thesis is that we found a hole in the state restoration algorithm developed by the authors of [7]. Their state restoration algorithm is incomplete, the amount of restoration depends on the order of selection of flip-flops in the trace signal list. This will lead to lower estimation of state restoration ratio in some cases.

We have developed an improved state restoration algorithm that solves this problem and using this combined with our novel simulated annealing based trace signal selection method has led to higher quality of restoration. We have also conducted experiments to show how variation in input vector affects the quality of restoration for a fixed set of trace signals.

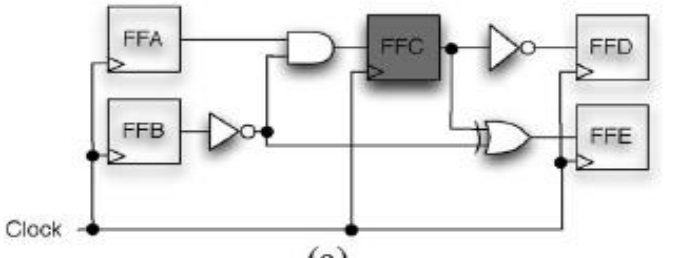
1.7 Thesis Overview

Following this Chapter, there are four more Chapters. In Chapter 2 we explain the design of the logic simulator used for state restoration which exploits bitwise parallelization [7]. We also show how the algorithm for state restoration is incomplete and then we propose an improved version of the state restoration algorithm to solve the issue. In Chapter 3 we propose our novel simulated annealing based technique for selecting trace signals. In Chapter 4 we provide a description of our entire experimental setup. We also provide the description of all experiments performed and the corresponding results and analysis in this chapter. Chapter 6 summarizes our work in this thesis and also provide an insight into potential future research work in this area.

Chapter 2
Algorithmic Solution for State Restoration

In Chapter 1 the concept of trace buffer was explained. The amount of debug data that can be acquired from the trace buffer depends on its width and depth. The width of the trace buffer limits the number of signals that can be traced and the depth limits the number of samples that can be acquired. Using the acquired data from the trace buffer, the values of other nodes in the circuit can be reconstructed.

In this Chapter, we discuss an algorithm for state restoration developed by the authors of [7]. The following figure also used in chapter 1 is also used to explain the restoration process



(a)

Clock cycles

	0	1	2	3	4
FFA	1	1	X	X	X
FFB	0	0	X	X	X
FFC	0	1	1	0	X
FFD	X	1	0	0	1
FFE	X	1	0	X	X

(b)

Fig 2.1 Example circuit for state restoration, (a) Circuit under debug (b) Traced and restored data in sequential elements [7].

Ideally, to debug the circuit, the values of all 5 flip-flops are to be recorded, this would imply using a trace buffer of size 5*5, but recording all values of all flip-flops is not a feasible solution. So for debug, the value of one flip-flop is traced and based of this data, the values of other flip-flops are reconstructed as much as possible.

In the above circuit, the trace buffer width is 1 and the trace buffer depth is 5. Which implies that one flip-flop can be traced and its value over 5 clock cycles can be recorded. In this example flip-flop C is traced. The basic idea of state restoration is to forward propagate and backward justify the traced values. This is basically done with Boolean equations. This concept may seem similar to ATPG (Automatic test pattern generation) used in manufacturing tests, however state restoration is different as it does not require any decisions to be made. The algorithm only needs to check if data can be reconstructed at a particular node, if not no backtracking is done and the node is left undefined. In their approach [7], they make an assumption that a gate level netlist is available and also all the trace signals are flip-flops. The methodology defined by [7] cannot be used to debug hard IP's as they require a gate level netlist to be available for state restoration. This approach can only be used to debug circuits that have passed all the manufacturing tests.

2.1 Principal Operations for State Restoration

Any combinational logic can be decomposed into the primitive two input gates (and, or, exor, nand, nor, exnor). The algorithm proposed by [7] involves applying two basic operations to logic gates in the translated circuit (i.e. after decomposing the combinational logic into two input gates).

These are forward propagation and backward propagation. Forward propagation is applied to a gate when the input values are known and the output is computed with the

help of Boolean algebra. This is comparable to what is done in functional simulators. The following figure shows examples of forward propagation and backward propagation

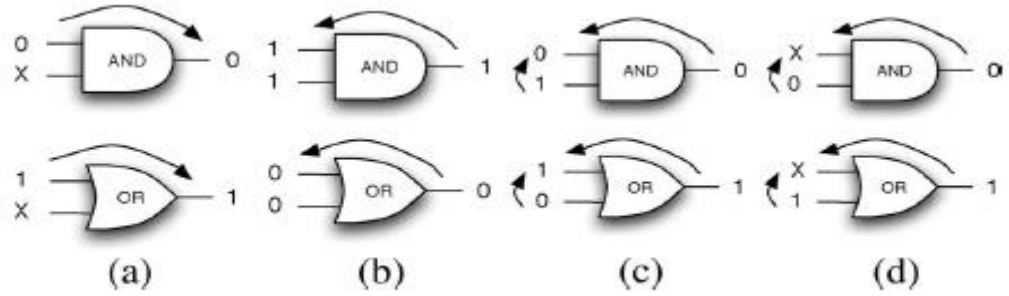


Fig 2.2: Principal operations for state restoration (a) Forward (b) Backward (c) Combined and (d) Undefined [7].

Examples of forward propagation are shown in Fig. 2.2 (a), In the AND gate as one of the inputs is 0, the output can be concluded to be 0 without looking at the other input. Similarly, in the OR gate as one of the inputs is 1, the output can be concluded to be 1 without looking at the other input.

If the output value of a gate is known, the backward operation can be used to find the input values of that gate. For example, in Fig 2.2(b) as the output of the AND gate is 1 both the inputs can be concluded to be 1. Similarly, in the OR gate as the output value is 0, the input values can be concluded to be 0. In the case when forward and backward operations are not sufficient, a combined operation in which the output value and one input value can be used to reconstruct the missing value as shown in Fig. 2.2(c). It is not always possible to reconstruct values as shown in Fig. 3(d), In the case of AND gate as the output is 0 and one input value is 0, the other input value cannot be reconstructed.

Similarly, in the case of OR gate as the output value is 1 and one of the input value is 1, the other input value cannot be reconstructed.

2.2 Exploiting Bitwise Parallelization for State Restoration

The authors of [7] designed an approach such that principal operations can be applied concurrently at a node across multiple clock cycles. They exploited the integer data type in ANSI C on a 32-bit platform to enhance the performance of their algorithm by storing data for 32 consecutive clock cycles in two integers for each node. For the work done in this thesis we have used an unsigned long long data type in C++ on a 64 bit platform to better exploit bitwise parallelization. The following table shows the two-bit code for data representation proposed by the authors of [7]

Logic value	Two bit code
0	00
1	11
undefined	01, 10

Consider for example a flip-flop having the values [1, 1, 0, 0, 1, X] for 6 clock cycles then this data is represented as follows

FFC_bit0 = 1, 1, 0, 0, 1, 0

FFC_bit1 = 1, 1, 0, 0, 1, 1

By working with two unsigned long long variables, the algorithm will be able to restore data for 64 consecutive clock cycles at a time, this is done by using a sequence of logic equations based on bitwise operations for all of the primitive gates. For each principal operation, (Forward propagation and backward propagation) two different equations (For

each unsigned long long variable) will be derived in such a manner so as to reduce the number of bitwise operations. These equations are derived with the help of K-maps. The derivation of forward and backward equations for AND gate is given below

A0A1 \ B0B1	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	0	1	0
10	0	0	0	0

(a)

A0A1 \ B0B1	00	01	11	10
00	0	0	0	0
01	0	1	1	1
11	0	1	1	1
10	0	1	1	1

(b)

Z0Z1 \ B0B1	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	0	0	0	0

(c)

Z0Z1 \ B0B1	00	01	11	10
00	1	1	0	1
01	1	1	1	1
11	1	1	1	1
10	1	1	1	1

(d)

Fig 2.3: Derivation of forward and backward equations for AND gate (a) K-map for Z0 (b) K-map for Z1 (c) K-map for A0 (d) K-map for A1

Here Z0 and Z1 are the two unsigned long long variables corresponding to the output node of the AND gate. A0, A1 and B0,B1 are the variables corresponding to the two inputs of the AND gate.

In the derivation of forward equations, the output of the AND gate Z is computed based of the values of inputs A and B. In the derivation of backward equations, the input of the AND gate A is computed based of the values of the output of the AND gate Z and the other input B, similarly the backward equation for input B can be derived. These are the equations obtained by simplifying the K-maps:

Z0 = A0A1B0B1

Z1= A1B1 + A1B0 +A0B1 + A0B0

$$\mathbf{A0 = Z0Z1}$$

$$\mathbf{A1 = \overline{B0} + \overline{B1} + Z1 + Z0}$$

Similarly, equations for B0 and B1 are obtained,

$$\mathbf{B0 = Z0Z1}$$

$$\mathbf{B1 = \overline{A0} + \overline{A1} + Z1 + Z0}$$

Using these derived equations if the input ports of an AND gate have the following values for 64 clock cycles,

A = [1,1,X,0,0,X,1,1,1,1,1,1 … 1]

B = [1,1,X,0,0,X,1,1,1,1,1,1 … 1]

From the two-bit representation we have,

A0 = [1,1,0,0,0,0,1,1,1,1,1,1 … 1]

A1 = [1,1,1,0,0,1,1,1,1,1,1,1 … 1]

Similarly, for B,

B0 = [1,1,0,0,0,0,1,1,1,1,1,1 … 1]

B1 = [1,1,1,0,0,1,1,1,1,1,1,1 … 1]

Now by applying the forward equations we get,

Z0 = [1,1,0,0,0,0,1,1,1,1,1,1 … 1]

Z1 = [1,1,1,0,0,1,1,1,1,1,1,1 … 1]

From this, we get the value of output Z for 64 clock cycles,

Z = [1,1,X,0,0,X,1,1,1,1,1,1 … 1]

If output is known for specific clock cycles in which the inputs are unknown then applying forward equations would rewrite the values of the output variables. Hence, to preserve

the known values, additional operations are added to the forward and backward equations. The modified equations are,

$$Z0 = \overline{(Z0 \oplus Z1)}\ Z0 + (Z0 \oplus Z1)\ (A0A1B0B1)$$

$$Z1 = \overline{(Z0 \oplus Z1)}\ Z1 + (Z0 \oplus Z1)\ (A1B1 + A1B0 + A0B1 + A0B0)$$

$$A0 = \overline{(A0 \oplus A1)}\ A0 + (A0 \oplus A1)\ Z0Z1$$

$$A1 = \overline{(A0 \oplus A1)}\ A1 + (A0 \oplus A1)\ (\overline{B0} + \overline{B1} + Z1 + Z0)$$

$$B0 = \overline{(B0 \oplus B1)}\ B0 + (A0 \oplus A1)\ Z0Z1$$

$$B1 = \overline{(B0 \oplus B1)}\ B1 + (A0 \oplus A1)\ (\overline{B0} + \overline{B1} + Z1 + Z0)$$

In a similar way forward and backward equations are derived for other basic logic gates. With the help of these equations, existing values will not be overwritten. Digital circuits often involve complex gates with a higher fan-in. These gates have to be decomposed into two input gates in order to use these equations or new equations have to be derived for gates that have a higher fan-in.

2.3 Algorithm for State Restoration

The state restoration algorithm proposed by the authors of [7] is given below.

Input: Circuit Graph, Trace_Signal_List
Output: Circuit Graph with restored data
1 search_list = Trace_Signal_List;
2 while search_list is not empty **do**
3 cur_node = first node in search_list;
4 **for each** (parent_node of cur_node)
5 BackwardOperation (cur_node, parent_node);
6 **if**(new data are restored for parent_node) **then**
7 Put parent_node at end of search_list
8 **for each** (child_node of cur_node) **do**
9 ForwardOperation (cur_node, child_node);
10 **if** (new data are restored for child_node) **then**
11 Put child_node at end of search_list
12 Delete cur_node from search_list

Before applying this algorithm, the circuit netlist is translated into a graph where the nodes represent logic gates, state elements, primary inputs, and outputs, and the directed edges represent signal dependencies.

After the netlist is translated into a graph, principal operations are applied to each node repeatedly until no more data can be reconstructed for all signals from the given subset of signals. The application of principal operations is done as shown in the above algorithm. In Fig 2.1, when comparing the amount of data available before and after state restoration, 14 state values are available after applying the restoration process on only 4 initial state values from FFC. This gives a state restoration ratio of 14/4 = 3.5. It has to be noted here that the amount of data that can be restored depends on the initial data that is sampled. For example, if in Fig 2.1 only FFE was sampled, then no new data would have been reconstructed.

The computation time for the algorithm is directly proportional to the number of nodes in the circuit.

2.4 Significance of the Order of Signal Selection

While doing some experiments for the work done in this thesis, we found that the original algorithm for state restoration is incomplete. The amount of restoration depends on the order of selection of flip-flops in the trace signal list. This can be better explained with the help of the example shown below. In the example, restoration is carried out for a specific window of 64 clock cycles.

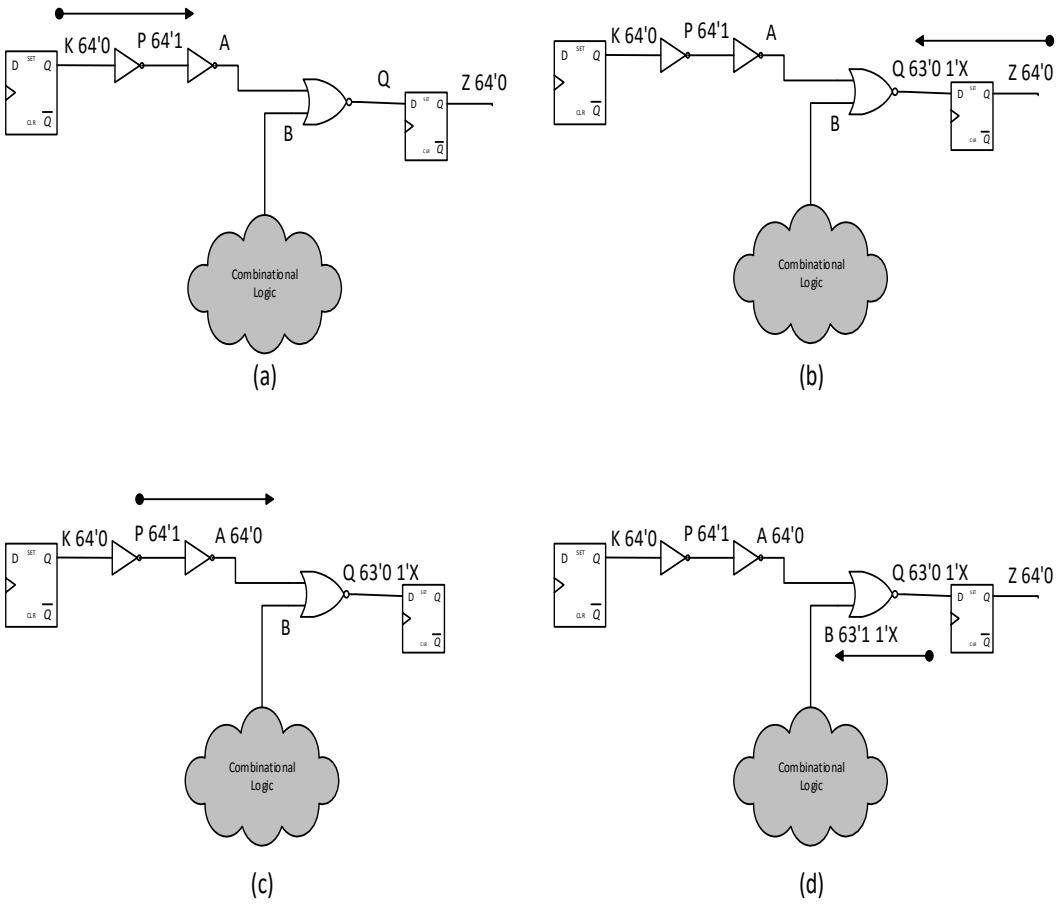


Fig 2.4: Significance of the order of signal selection (Case 1)

In Figure 2.4, both the flip-flops are selected to be part of the trace signal list. The order of selection of these flip-flops in the trace signal list determines the amount of restoration. In Case 1, the flip-flop with output port K is chosen first and the flip-flop with output port Z is chosen next. In Fig 2.4 (a), the restoration process begins and node P is restored by forward propagation. After which the next node in the search_list is Z and it restores node Q by backward propagation as shown in Fig 2.4 (b). The next node in the searchlist is P, it applies the forward propagation operation and restores node A as shown in Fig 2.4 (c) . After which node Q is in the front of the searchlist and it restores node B by backward propagation as shown in Fig 2.4 (d). Next we interchange the order of selection of flip-flops and observe the restoration process.

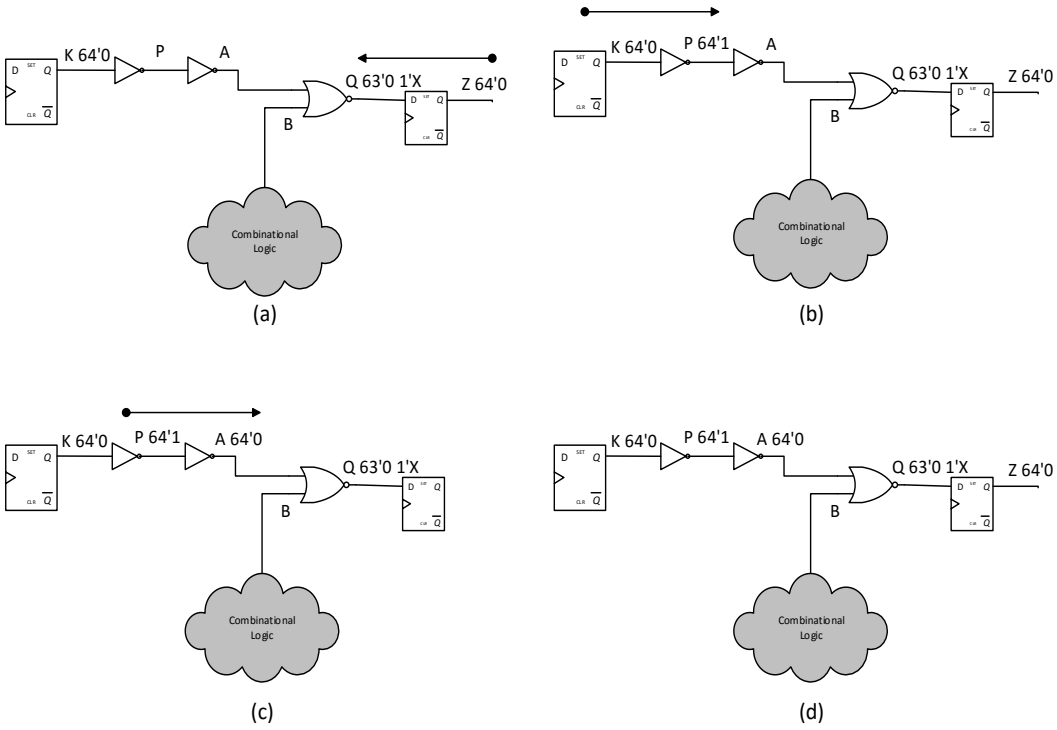


Fig 2.5: Significance of the order of signal selection (Case 2)

In Case 2, the flip-flop with output port Z is chosen first and the flip-flop with output port K is chosen next. In Fig 2.5 (a), the restoration process begins and node Q is restored by backward propagation. After which the next node in the search_list is K and it restores node P by forward propagation as shown in Fig 2.5 (b). The next node in the searchlist is Q, it applies the backward propagation operation and fails to restore anything. After which node P is in the front of the searchlist and it restores node A by forward propagation as shown in Fig 2.5 (c). Finally, node A applies forward propagation and is unable to restore anything and the final state of the circuit is as shown in Fig 2.5 (d). Here Node B is not restored, it should have been restored with the knowledge of nodes A and Q (As seen in case 1). Hence we conclude that the original restoration algorithm is incomplete as the amount of restoration depends on the order of selection of flip-flops. To tackle this problem, we propose a modified algorithm for state restoration in the next section.

2.5 Improved Algorithm for State Restoration

As discussed in the previous sub section the original state restoration algorithm is incomplete. There is a difference in the amount of restoration obtained depending on the order in which flip-flops are pushed into the trace signal list. The algorithm below solves this problem

Input: Circuit Graph, Trace_Signal_List
Output: Circuit Graph with restored data

```
1 search_list = Trace_Signal_List;
2 while search_list is not empty do
3    cur_node = first node in search_list;
4    for each (parent_node of cur_node)
5       BackwardOperation (cur_node, parent_node);
6       if (new data are restored for parent_node) then
7          Put parent_node at end of search_list
8    for each (child_node of cur_node) do
9       ForwardOperation (cur_node, child_node);
10      if (new data are restored for child_node) then
11         Put child_node at end of search_list
12      else if(child_node is not unknown) then
13         Backward operation (child node, sister_node of cur_node)
14         if(new data are restored for sister_node)
15           put sister_node of cur_node at end of search_list
16  Delete cur_node from search_list
```

In the modified version of the algorithm during forward propagation if the output node is defined, then a backward operation is carried out, where there is an attempt to restore sister node of the current node. Note that this only has to be done for two input gates, as there is no need for this in a NOT gate. We now present the example discussed in the previous section and show how we have removed the dependence on the order of selection of flip-flops.

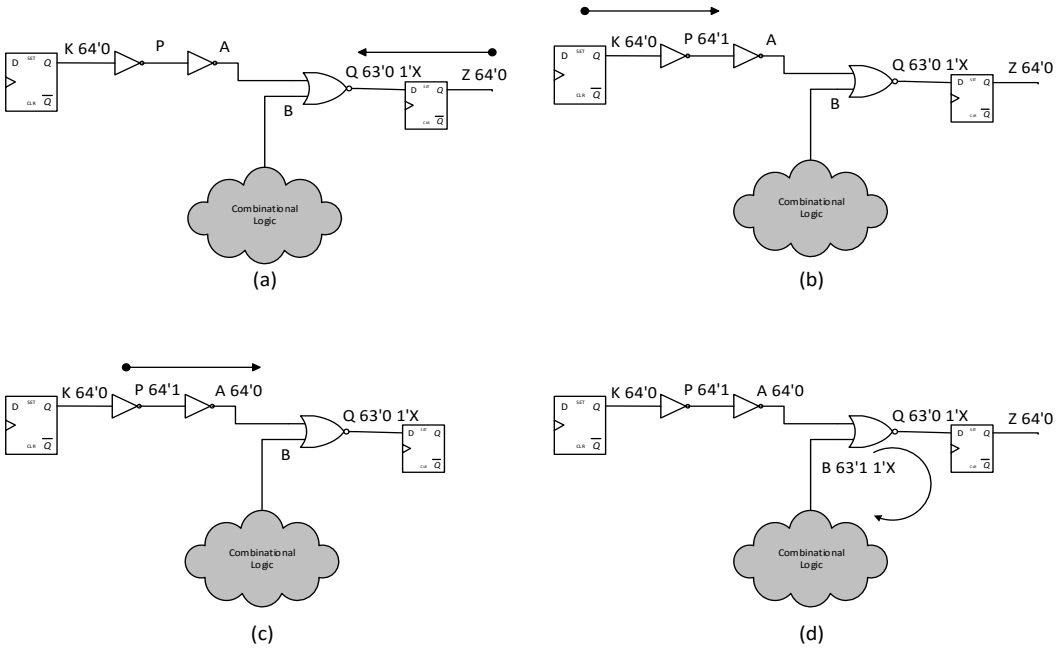


Fig 2.6: Eliminating dependence on order of selection of flip-flops

As discussed previously if the flip-flop with output port Z is chosen first, then node B would never be restored in the original restoration algorithm. Using the modified restoration when node A is in the front of the search_list it would detect that the output node of the gate (Node Q) is not unknown, after which a backward propagation operation is applied to restore the sister node of the current node (Node B). This operation would successfully restore the value of Node B for 63 clock cycles. In this way the dependence on order of selection of flip-flops is eliminated.

2.6 Summary of Chapter 2

In Chapter 2 we discussed the design of the logic simulation tool used for state restoration. We explained the principal operations for state restoration and also how the equations for these operations are implemented with the help of bitwise logic operations. We also presented an example to show how the original restoration algorithm is incomplete, as the amount of restoration depends on the order of selection of flip-flops in

the trace signal list. We also proposed an improved restoration algorithm which would tackle this problem. In the next chapter we present our novel simulated annealing based technique to select trace signals.

Chapter 3
Automated Approach to Select Trace Signals

The popular metric for measuring the quality of a set of trace signals is restoration ratio.

Restoration Ratio = (No. of traced and restored values)/(No. of traced values)

The objective of the work done in this thesis is to maximize the metric restoration ratio. Existing trace signal selection approaches can be classified into two categories, structural and simulation based. The structural based approach involves using a greedy heuristic to select trace signals optimizing a metric which is dependent on the structure of the circuit [16 7 17 18]. These set of approaches are computationally efficient but suffer from the drawback of poor restoration quality. The other set of approaches is the simulation based approach which are computationally inefficient but offer a higher quality of restoration. Simulation based approaches work on the intuition that if a set of signals work well for a particular input vector they should work well for other input vectors as well. Chatterjee et al [19] were the first to provide a simulation based approach to select trace signals. They designed a greedy elimination approach with a time complexity of the order of n^2. They start of by selecting all flip-flops in the design. In each step, they remove one flip-flop which is least important (based on the results of simulation). This continues until the number of flip-flops in the list becomes equal to the width of trace buffer. The main problem with this approach is removing any flip-flop initially may still lead to 100% restoration, hence there is a possibility of eliminating beneficial signals. Another simulation based approach was designed by Rahmani et al. [21] Their approach is based

of two components: (1) an iterative approach to signal selection based on mock simulations and (2) a filtering scheme based on Integer-Linear Programming (ILP) to refine the selected set. However, their signal selection algorithm is a greedy algorithm and consequently this affects their restoration quality.

3.1 Simulation Based Approach with Short Trace Buffer Depth

A common way to reduce effort in simulation-based estimations is to perform several short simulations and average their outcomes. Chatterjee et al [19] proposed the use of a shorter trace buffer depth. They showed that the state restoration ratio variation is negligible beyond a trace buffer size of 64.

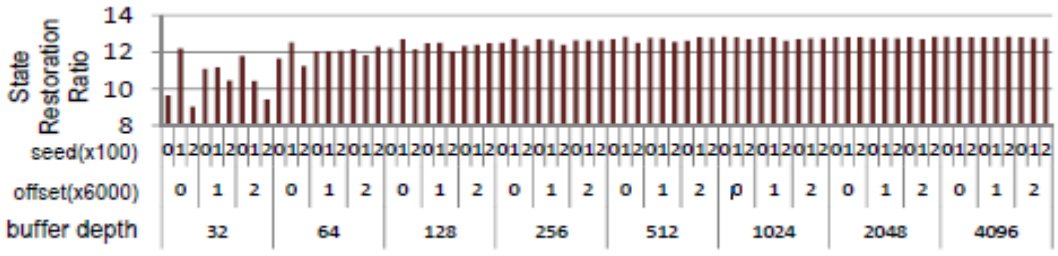


Fig. 3.1. Impact of trace buffer size on SRR. Analysis on ISCAS 89 benchmark s35932 over 3 random starting points of tracing and 3 random sets of input values per starting point indicates that SRR for a fixed set of signals is fairly insensitive to trace buffer sizes beyond 64 [19].

To further validate their hypothesis that short trace buffer sizes are accurate enough for State Restoration Ratio estimation, they performed a correlation study of their simulation based restoration capacity metric with observed SRR (Trace buffer depth of 4096 clock cycles). Their SRR estimate is computed for 1000 random sets of 8 flip-flops each using a fast mock simulation with a trace buffer size of 64 and only one random set of inputs and starting time for tracing.

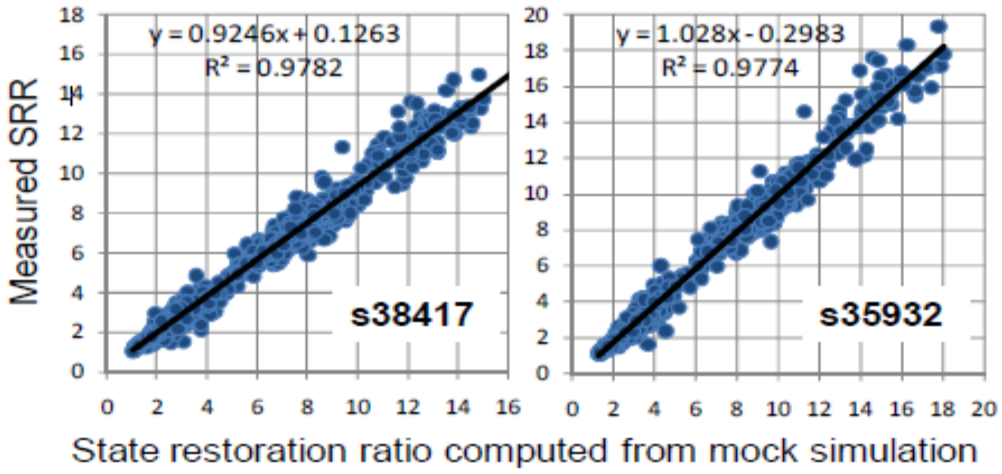


Fig. 3.2. Correlation of simulation-based restoration capacity metric of [19] with observed SRR using a mock simulation trace buffer depth of 64 clock cycles for ISCAS 89 benchmarks s38417 and s35932 [19].

The plots shown above clearly indicate a very high correlation between the estimation metric proposed by [19] and the observed state restoration ratio.

For these reasons, this metric is used to select trace signals for the work done in this thesis.

3.2 Simulated Annealing Based Signal Selection

Basically the trace signal selection problem can be viewed as a Bi-partitioning problem. The first partition here is the set of flip-flops which will be recorded by the trace buffer and the other partition is the set of all other flip-flops in the design. This insight gave us the idea to use the simulated annealing heuristic for this problem.

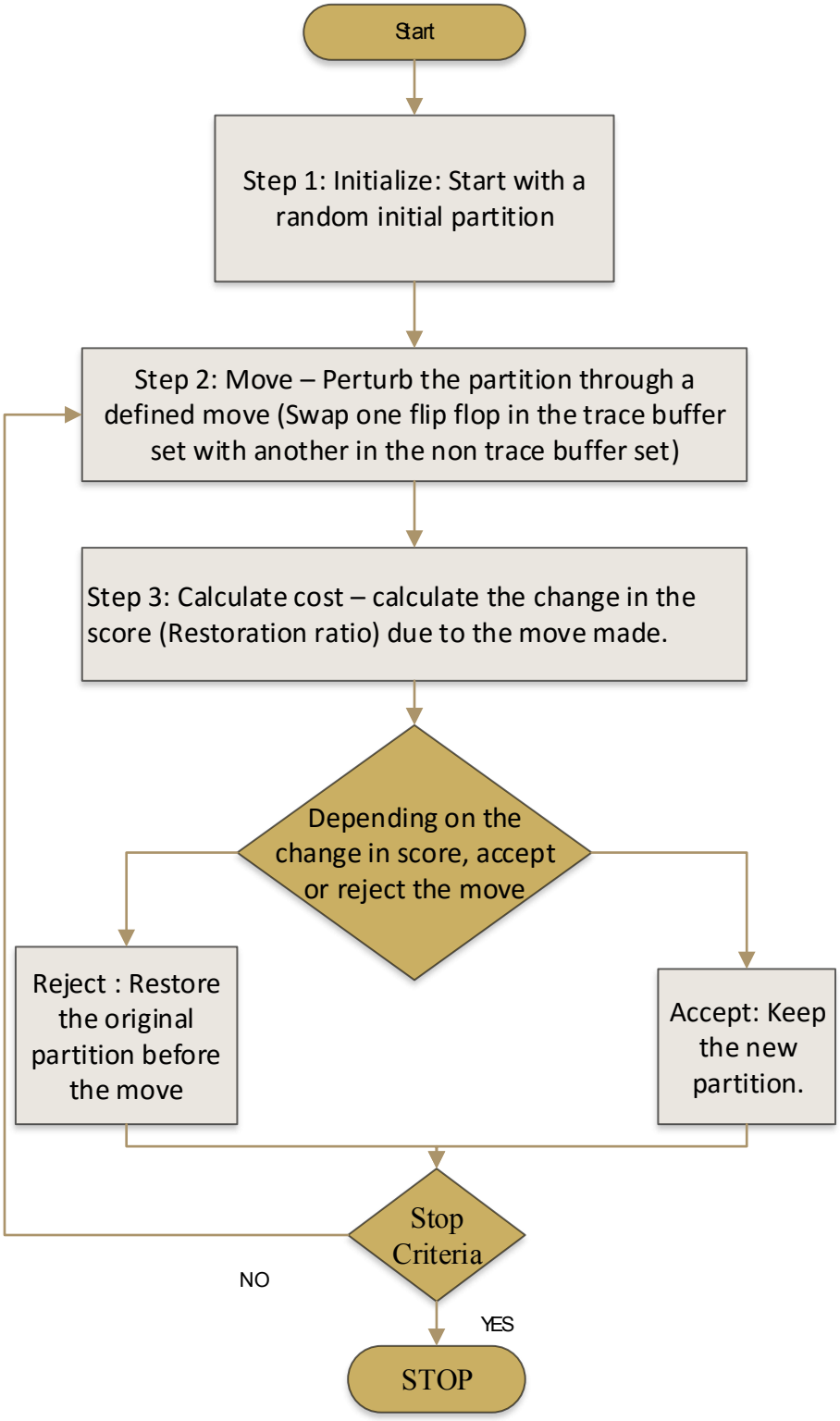


Fig. 3.3.: Flowchart to show the steps involved in the simulated annealing heuristic to select trace signals.

.

3.2.1 Initialization Step: Random Initial Partition

There are basically two partitions, the first partition has all the flip-flops which will be tapped onto the trace buffer. The second partition has all other flip-flops in the design. The size of the first partition, depends on the trace buffer width (Usually in the industry width = 8 or 16 or 32). The size of the second partition depends on the number of flip-flops in the design. In the initialization step a certain number of flip-flops (Number = trace buffer width) will be randomly selected to be a part of the first partition, all other flip-flops will be in the second partition. Evaluate the state restoration ratio metric for the set of flip-flops in the first partition. This metric serves as the cost function for the simulated annealing heuristic.

3.2.2 Move Function

In the move function the partition is perturbed through a defined move. One flip-flop in the trace buffer set is moved to the non-trace buffer set. Some other flip-flop in the non-trace buffer set is moved to the trace buffer set. The selection of these flip-flops is done randomly.

3.2.3 Cost Function

The new trace buffer set may have a different score (State Restoration Ratio). The difference in the score between the new trace buffer set and the old trace buffer set, will dictate if the move is accepted or not. If there is an improvement in the score the move will be accepted. If there is a degradation in the score, the move may or may not be accepted. Initially a lot of inferior moves are accepted, but as the number of iterations

keeps increasing the probability of an inferior move being accepted decreases. In the end, no degrading moves are accepted.

3.2.4 Stop Criteria

For the work done in this thesis, Will Naylor's simulated annealing package [22] has been used. There are 3 user inputs that have to be given to the package which control the stop criteria. They are

- Problem size
- Stop run length
- Epochs to run

Epochs are "problem size acceptances". At each acceptance, the temperature is decreased by a fixed amount, the amount is chosen to make the temperature 0 after "Epochs to run" epochs. Temperature is not decreased at rejections.

Problem size is a parameter which specifies the number of variables in the problem to be optimized.

Stop run length specifies the unaccepted mutations to terminate the anneal.

All mutations which give improvement are immediately accepted. To avoid the algorithm getting stuck in local minima too soon, degradations are sometimes accepted with probability equal to

$$Prob = \exp(-delta/temp)$$

Where delta is the change in objective function produced by the mutation. The temperature decreases by a fixed amount each time a mutation is accepted. Temperature starts at some medium to large value and falls throughout the run toward 0. At the end the temperature is equal to 0. At temp=0, no degrading mutations are accepted.

3.3 Summary of Chapter 3

In this Chapter we discussed about conventional simulation based approaches for selecting trace signals. Then we presented a flowchart that illustrated our simulated annealing based approach for selecting trace signals. We explained the various steps and functions involved in simulated annealing. We also explained how the stop criteria is controlled by a few parameters. The tuning of these parameters is discussed in Chapter 4. In the next chapter we present our entire experimental setup, list of experiments and the corresponding analysis for it.

Chapter 4

Experimentation and Results

In this Chapter we discuss the experimental setup that is used to evaluate the proposed technique discussed in Chapter 3.

4.1 Experimental Setup

The benchmarks that we have used to evaluate our proposed technique are the ISCAS 89 benchmarks [24]. The reasoning behind using these benchmarks is that they are publically available and most papers in this line of research have used these benchmarks.

4.1.1 Benchmarks

The ISCAS 89 benchmarks are publically available gate level netlists. The required information of the benchmarks we have used is provided in the table below.

Circuit	Number of flip-flops	Number of primary inputs	Number of primary outputs
S5378	179	35	49
S9234	211	36	39
S15850	534	77	150
S38584	1426	38	304
S38417	1636	28	106
S35932	1728	35	320

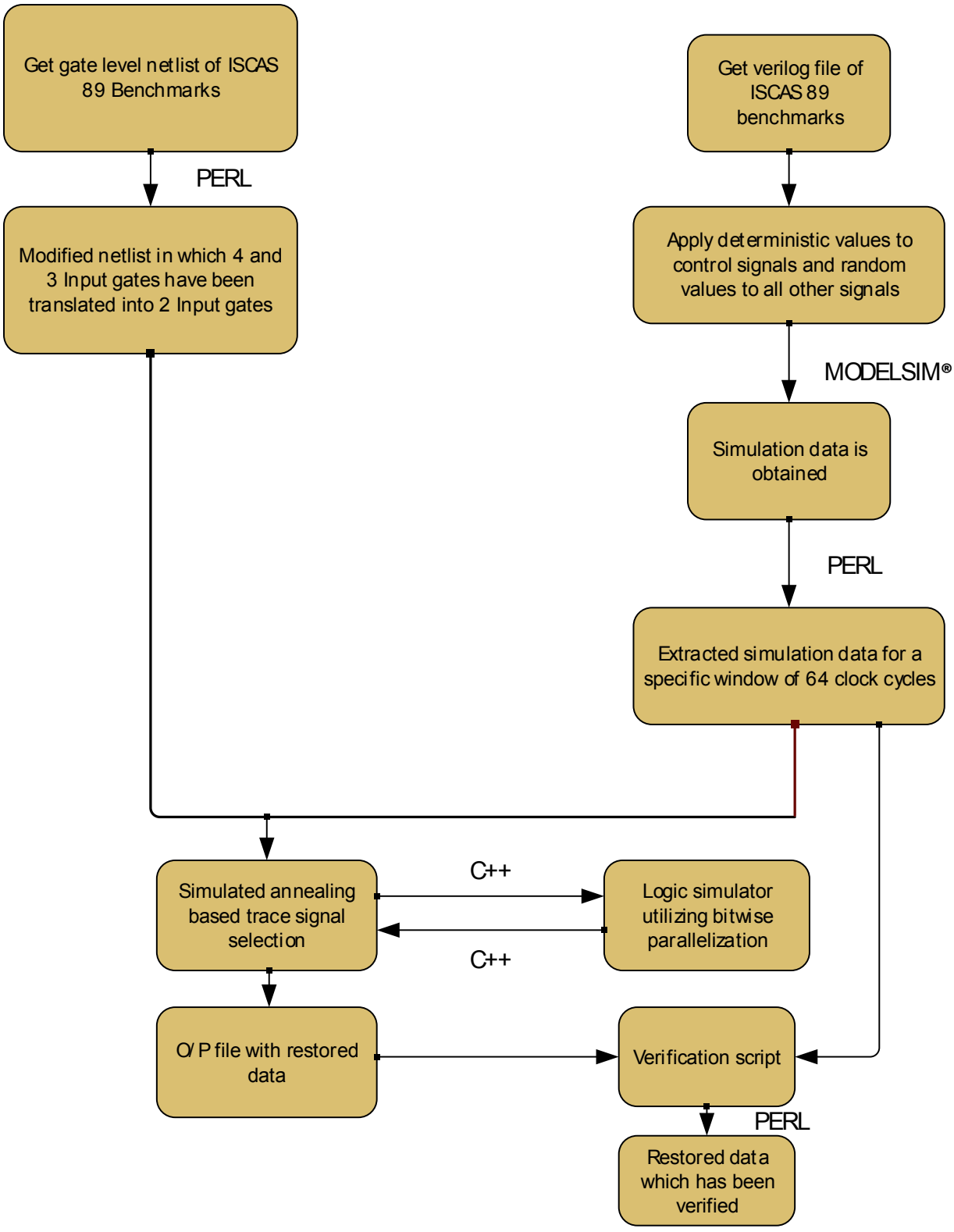


Fig. 4.1. Flow diagram for our entire experimental setup

4.1.2 Netlist Translation Script (PERL)

As discussed in section 2 of this thesis, the logic simulator we have used requires all the combinational logic to be translated into 2 input gates. In the benchmarks that we have used, initially the netlist is made up of 3 input and 4 input gates. We have developed a PERL script that automates the combinational logic translation into 2 input gates.

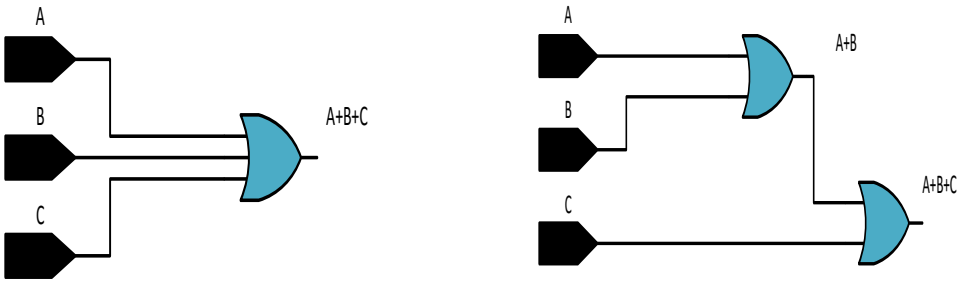


Fig. 4.2. 3 input OR gate translated into 2 input OR gates

4.1.3 Simulation Data from Modelsim®

To replicate data stored in trace buffer at the post silicon stage, we obtain the simulation data from the Mentor Graphics logic simulation tool Modelsim ® [23]. To get this data we apply deterministic values to control signals and random values to all other signals. To find the control signals in a design we use the same Design compiler and Tetramax® (Synopsys tools) flow proposed by [4]. After which we have a script in PERL which parses the log files [From Modelsim®] and gives an output file which has the simulation data for a specific window of 64 clock cycles. For example, the output file has data as shown below,

G0 1010101010101010101010101010101010101010101010101010101010101010

The node G0 has the values shown above for a specific window of 64 clock cycles. This file is fed to the trace signal selection tool.

4.1.4 Trace Signal Selection Tool, Logic Simulation Tool (C++)

The design of these two tools have been discussed in detail in Chapters 3 and 2 respectively. Basically the simulated annealing tool selects a set of flip-flops (Number of flip-flops being equal to trace buffer width) in every iteration and the logic simulation tool evaluates the restoration ratio for this set of flip-flops. The inputs to these tools are the outputs of the netlist translation script and the Modelsim® parser script.

4.1.5 Verification Script (PERL)

The trace signal selection tool generates a log file, which has the restored data for the set of flip-flops which achieves the best restoration ratio. We use this log file and the output of the Modelsim® parser script to verify if the restored data is completely correct.

4.2 Comparison Between Original and Improved Restoration Algorithm

As discussed in Chapter 2, the original restoration algorithm is incomplete. When the output and one input of a gate are known, the other input of the gate can only be restored by backward propagation (this is because of the way the forward and backward equations are derived, the forward equations can only restore the output and not the other side input). To fix this problem we proposed an improved restoration algorithm in Chapter 2. We observed that the problem has been solved and we achieve better final restoration ratio for the same circuit, same input vector and same flip-flops being traced. To evaluate the improvement obtained, we selected 10 random sets of flip-flops (Trace buffer width = 8) for all the benchmarks and calculated the restoration ratio with the original restoration algorithm and the improved restoration algorithm. The idea behind selecting just 10 random sets was to check if the improvement is obtained on a regular basis (that is the

order of selection of flip-flops matters on a regular basis). The comparison tables are given below.

Table. 4.1. Evaluating difference in restoration algorithms for S5378

Original Restoration	Improved Restoration	Percentage of Improvement
10.24	10.24	0%
9.41	9.41	0%
3.28	11.08	238%
4.67	8.78	88%
11.51	11.51	0%
10.01	10.01	0%
9.11	9.11	0%
1.48	1.48	0%
4.47	4.47	0%
3.27	3.27	0%

Table. 4.2. Evaluating difference in restoration algorithms for S9234

Original Restoration	Improved Restoration	Percentage of Improvement
2.6	2.63	1%
2.51	2.51	0%
4.22	4.22	0%
1.61	1.61	0%
5.15	5.25	2%
3.92	3.92	0%
2.36	2.36	0%
3.63	3.63	0%
5.24	5.23	0%
2.05	2.16	5%

Table. 4.3. Evaluating difference in restoration algorithms for S15850

Original Restoration	Improved Restoration	Percentage of Improvement
2.53	2.53	0%
4.43	4.43	0%
1.57	5.68	262%
1.56	1.56	0%
3.93	3.93	0%
9.79	14.125	44%
4.66	4.7	1%
1.03	1.03	0%
3.21	3.33	4%
4.47	5.39	21%

Table. 4.4. Evaluating difference in restoration algorithms for S38417

Original Restoration	Improved Restoration	Percentage of Improvement
11.52	12	4%
10.44	10.93	5%
2.98	2.99	0%
3.95	4	1%
3.25	3.32	2%
8.6	10.29	20%
1.73	1.73	0%
2.05	4.62	125%
1.24	1.24	0%
12.46	13.05	5%

Table. 4.5. Evaluating difference in restoration algorithms for S38584

Original Restoration	Improved Restoration	Percentage of Improvement
2.28	2.28	0%
1.29	1.29	0%
1.29	1.29	0%
1.36	1.36	0%
1	1	0%
2.56	2.56	0%
2.24	2.24	0%
1.49	1.49	0%
1	1	0%
3.06	3.26	7%

Table. 4.6. Evaluating difference in restoration algorithms for S35932

Original Restoration	Improved Restoration	Percentage of Improvement
13.62	14.45	6%
3.77	3.93	4%
7.02	7.07	1%
9.82	10.21	4%
12.83	13.71	7%
5.73	6.34	11%
6.07	6.34	4%
10.01	10.58	6%
6.78	6.95	3%
6.58	6.71	2%

Conclusion from these tables: Clearly the improved restoration algorithm is able to restore more or equal values for the same input vector and same flip-flops being traced when compared to the original restoration algorithm. We can also see clearly that the order of selection of flip-flops matters on a regular basis. In some cases, the improvement obtained is extremely large. For this particular experiment we observed an improvement in Restoration Ratio up to 262%.

Next we performed short simulated annealing trace signal selection runs with the same random seed to show the difference between the two restoration algorithms. For this particular experiment we have set the trace buffer width equal to 8. Given next are convergence plots for each benchmark with both the original and improved restoration algorithms.

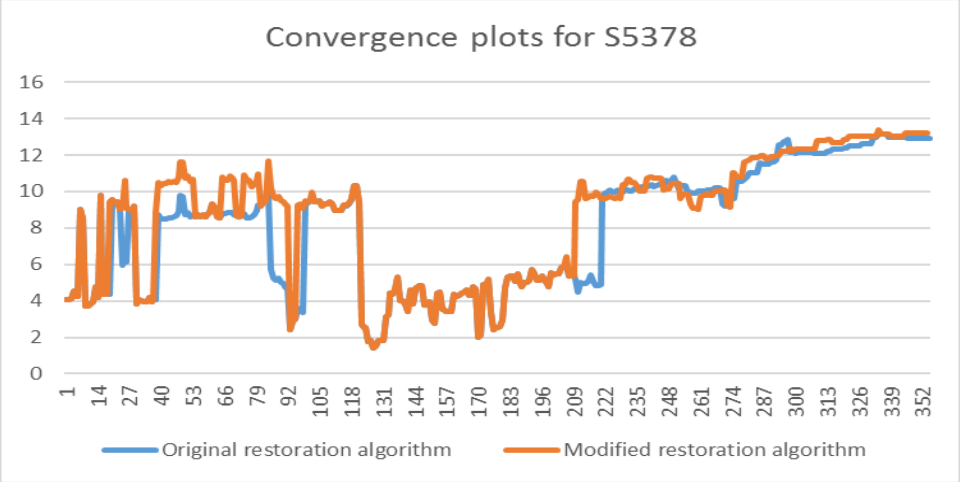


Fig. 4.3. Convergence plots for s5378 comparing the restoration algorithms

The difference is not clearly visible in this benchmark, the restoration ratio (cost function) ends up being 12.89 in the original approach and 13.24 with the improved approach.

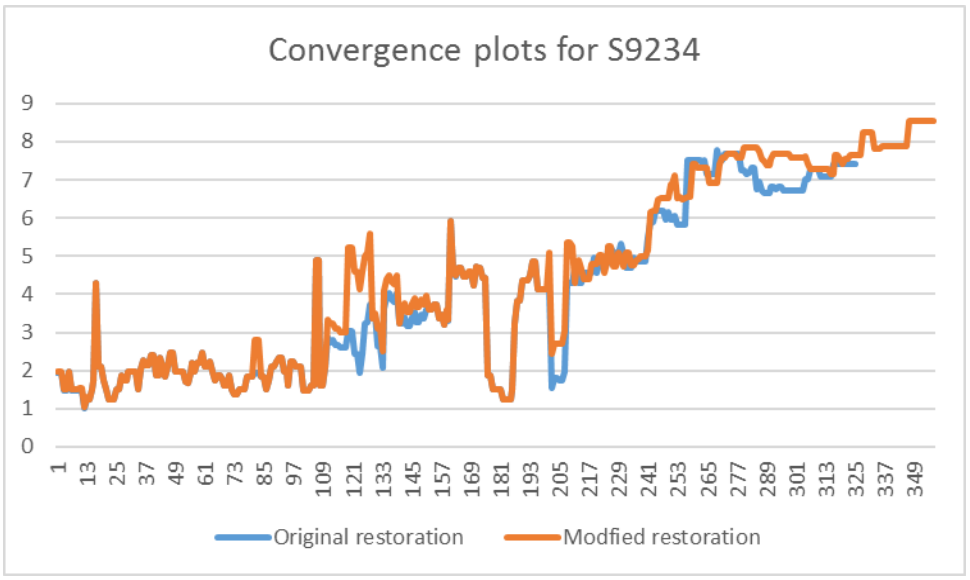


Fig. 4.4. Convergence plots for s9234 comparing the restoration algorithm

The difference is more clearly visible in this benchmark with the restoration ratio reaching 7.42 with the original approach and 8.54 with our approach.

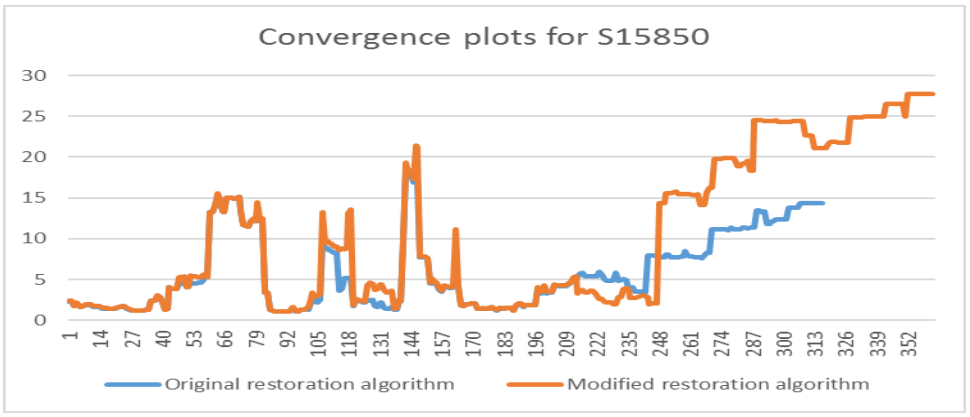


Fig. 4.5. Convergence plots for s15850 comparing the restoration algorithms

There is a huge difference between the two plots for this benchmark with the restoration ratio reached being double in our approach.

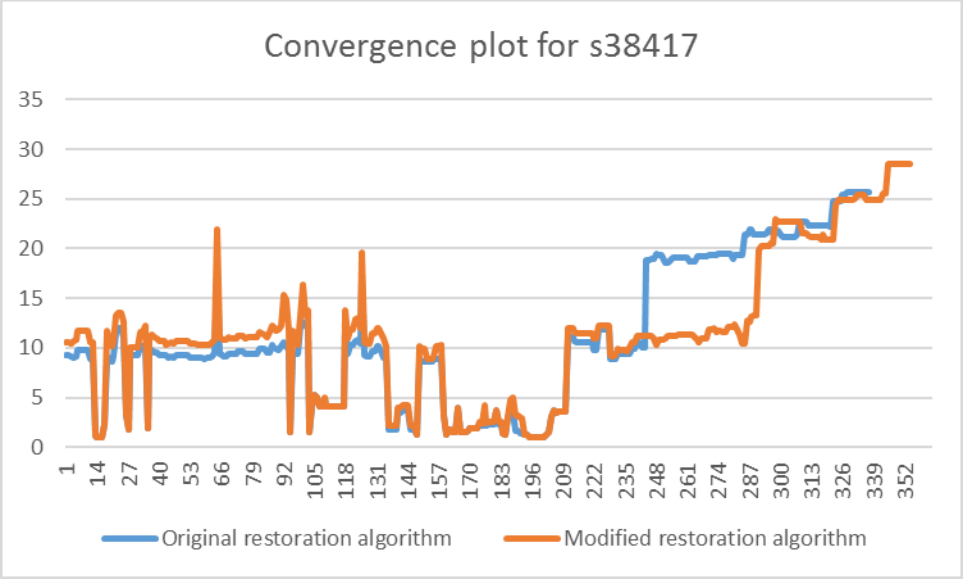


Fig. 4.6. Convergence plots for s38417 comparing the restoration algorithms

The restoration ratio ends up being 25.67 in the original restoration algorithm and 28.57 with the improved algorithm.

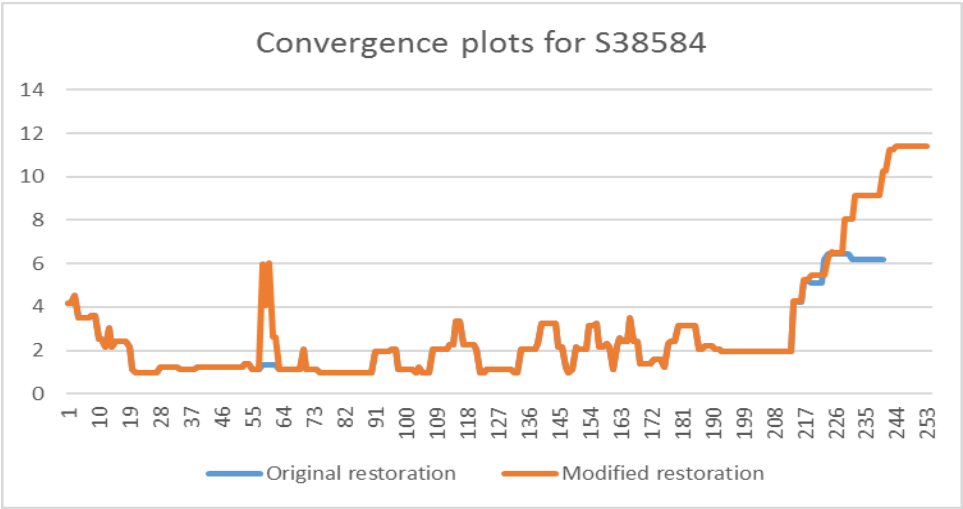


Fig. 4.7. Convergence plots for s38584 comparing the restoration algorithms

There is a considerable difference between the two plots for this benchmark with the restoration ratio reaching 6.2 with the original approach and 11.43 with our approach.

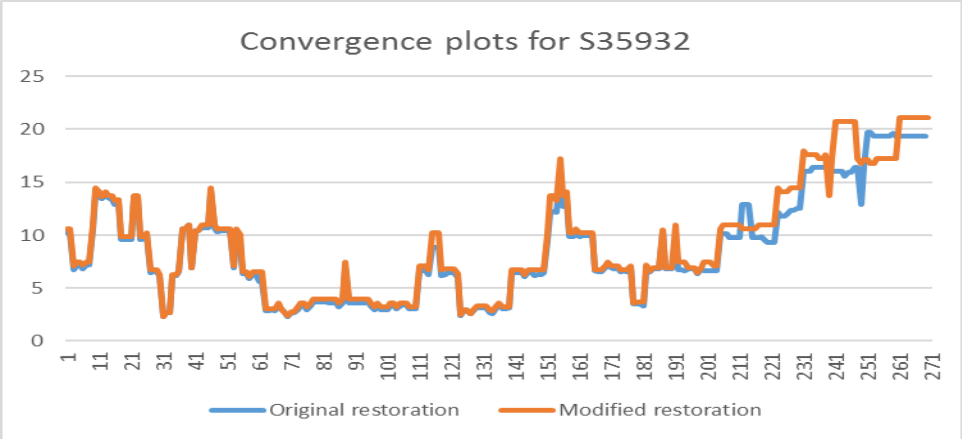


Fig. 4.8. Convergence plots for s35932 comparing the restoration algorithms

The restoration ratio ends up being 19.31 in the original approach and 21.10 with our approach.

Conclusion from these plots: Clearly our restoration algorithm is able to restore more values for the same input vector and same flip-flops when compared to the original restoration algorithm. Due to the difference in the cost function (Restoration ratio) between the two runs, there is a difference in moves being accepted or rejected, consequently the convergence plot differs for the two runs. Hence even though we applied the same cooling schedule and stop criteria for both approaches they need not run for the same number of epochs and there is also a difference in the final cost reached.

4.3 Tuning for Simulated Annealing and Convergence Plots

The trace buffer widths used in our experiments are 8,16 and 32, we have selected these widths as these are the widths selected by all of the papers in this research area. As explained in Chapter 3, there are three user given inputs to the simulated annealing package

- Problem size

- Stop run length
- Epochs to run

Since we are optimizing one particular variable, “Restoration Ratio”, we are setting the problem size to be equal to 1.

We set the stop run length to be equal to 500, this is a large number as it means that the anneal will only be terminated if there are 500 consecutive rejected moves and it was chosen as our goal was maximizing the restoration ratio regardless of the run time.

We wanted the epochs to run to be a function of the number of flip-flops in a design and the trace buffer width chosen for that design. Also since our goal was maximizing the restoration ratio, we set this variable to be equal to (number_of_flip_flops*trace_buffer_width)*100.

We used the simulated annealing heuristic to select trace signals for 6 different ISCAS 89 benchmarks and 3 different trace buffer widths. For a specific ISCAS 89 benchmark and trace buffer width, we launched six different runs. These six runs correspond to 3 different random seeds for obtaining simulation data from Modelsim® and 2 different windows of 64 cycles.

We present the convergence plots for each benchmark and trace buffer width for one particular window of 64 clock cycles. These plots show how the cost function (Restoration ratio) moves towards the global optimum value.

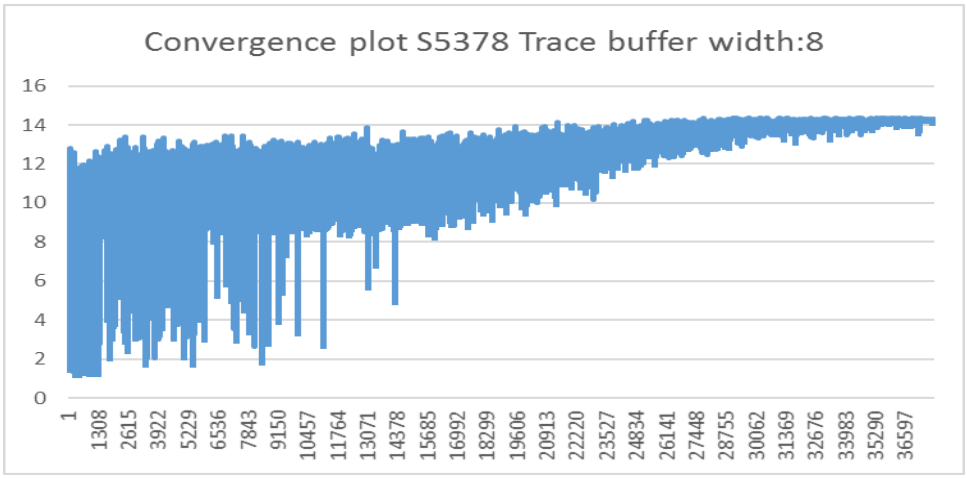


Fig. 4.9. Convergence plot for s5378 and trace buffer width 8 for the actual trace signal selection run

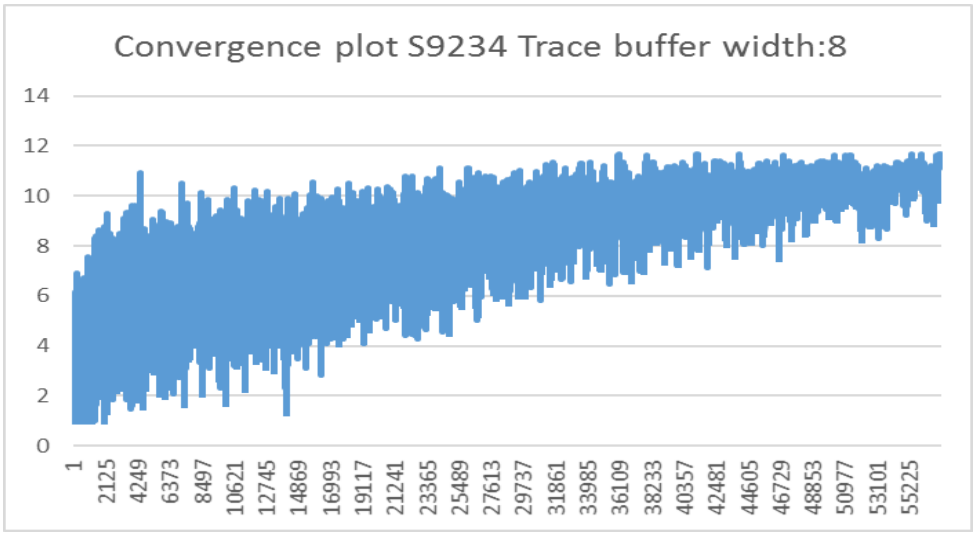


Fig. 4.10. Convergence plot for s9234 and trace buffer width 8 for the actual trace signal selection run

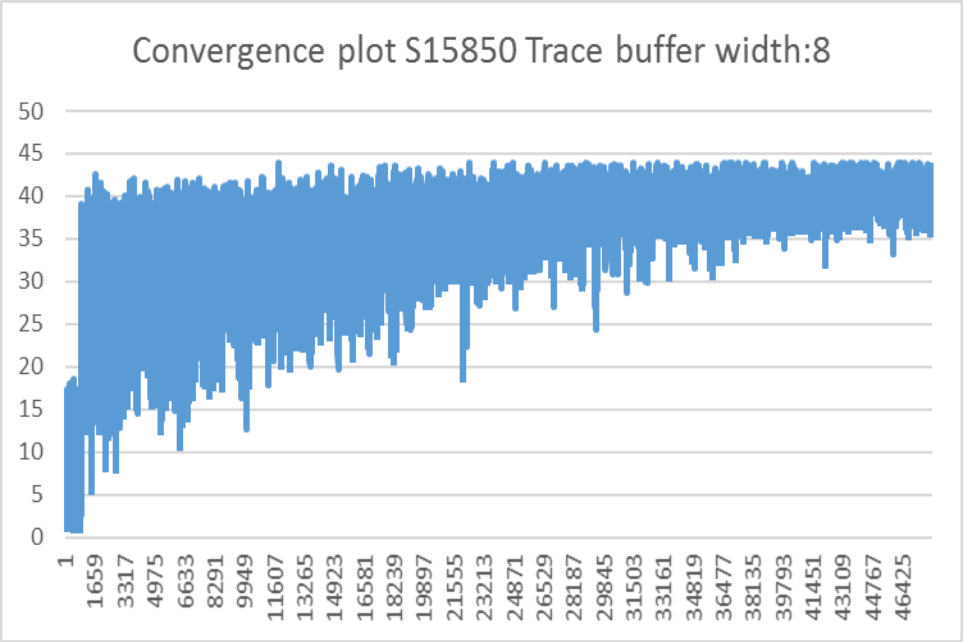


Fig. 4.11. Convergence plot for s15850 and trace buffer width 8 for the actual trace signal selection run

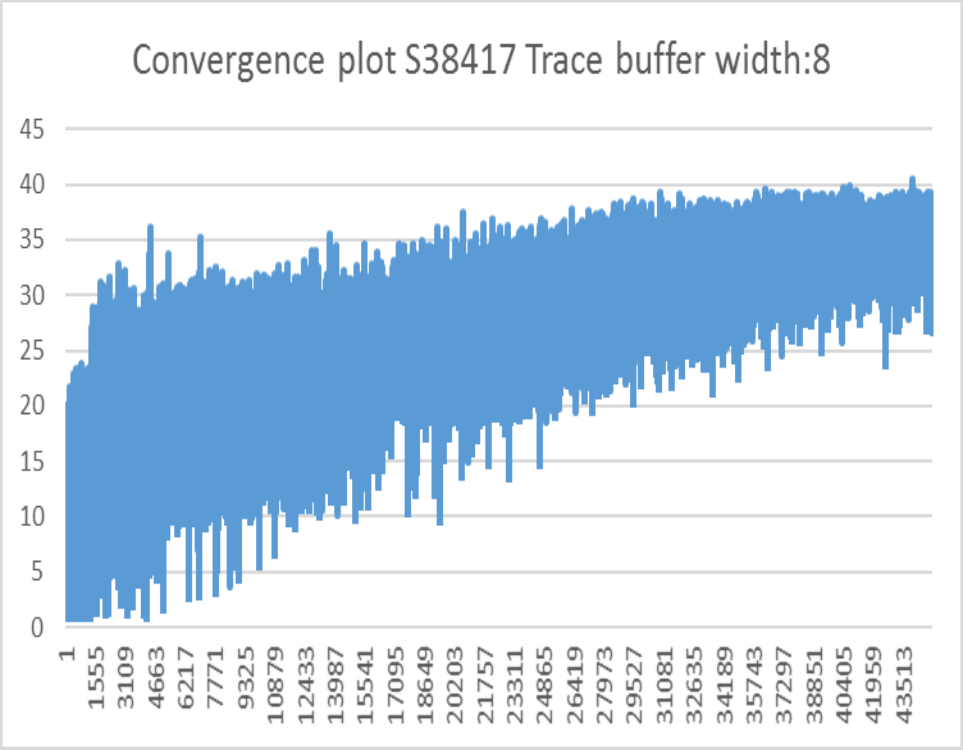


Fig. 4.12. Convergence plot for s38417 and trace buffer width 8 for the actual trace signal selection run

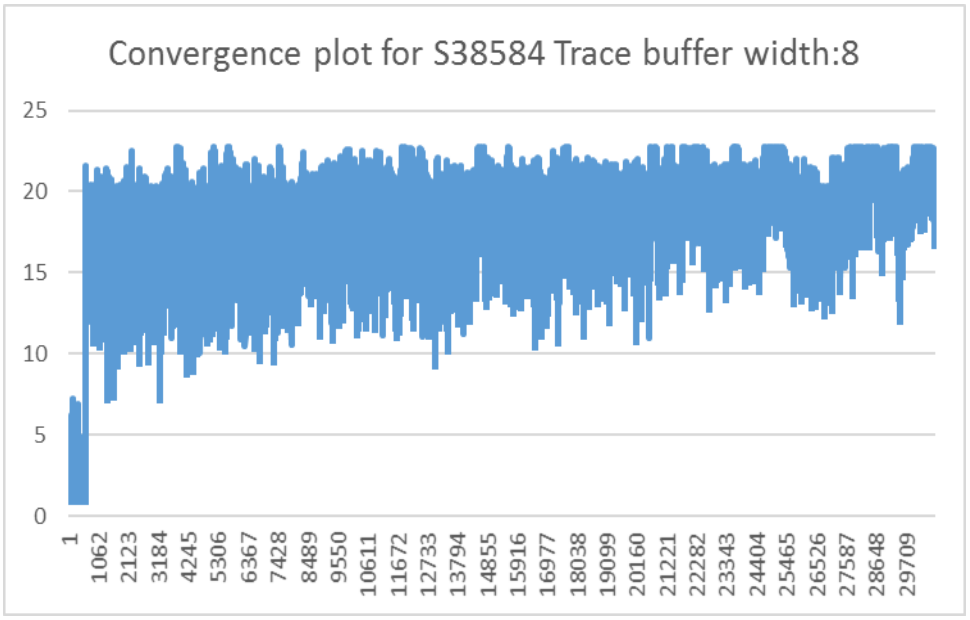


Fig. 4.13. Convergence plot for s38584 and trace buffer width 8 for the actual trace signal selection run

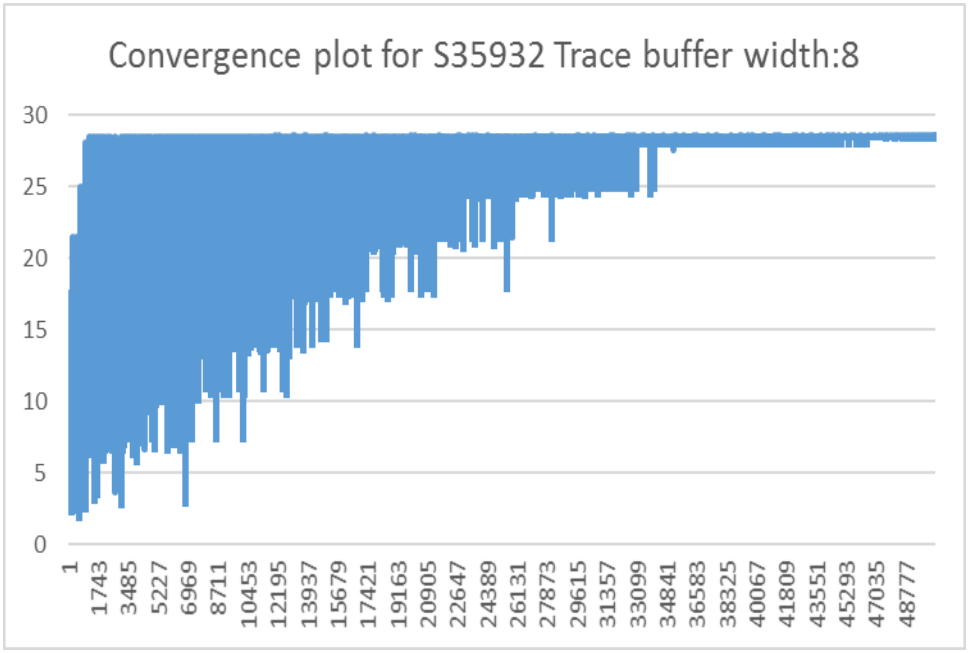


Fig. 4.14. Convergence plot for s35932 and trace buffer width 8 for the actual trace signal selection run

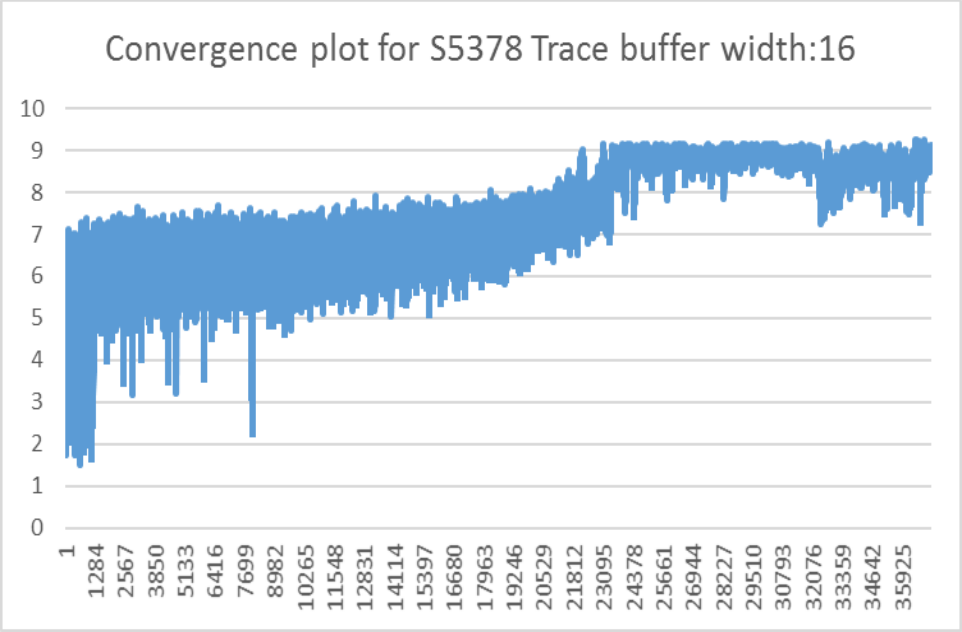


Fig. 4.15. Convergence plot for s5378 and trace buffer width 16 for the actual trace signal selection run

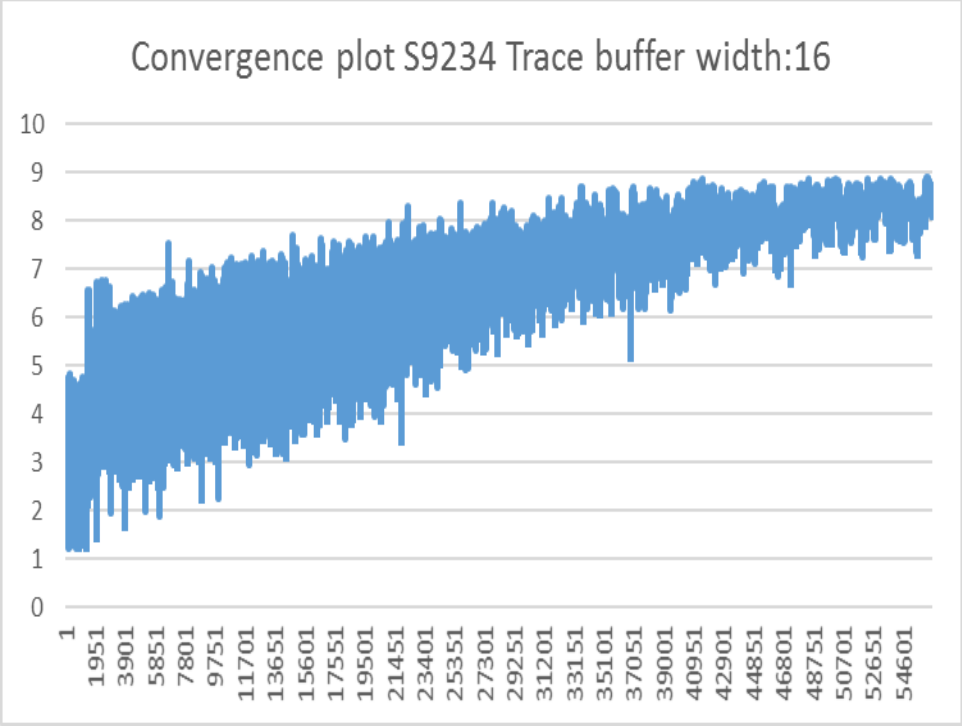


Fig. 4.16. Convergence plot for s9234 and trace buffer width 16 for the actual trace signal selection run

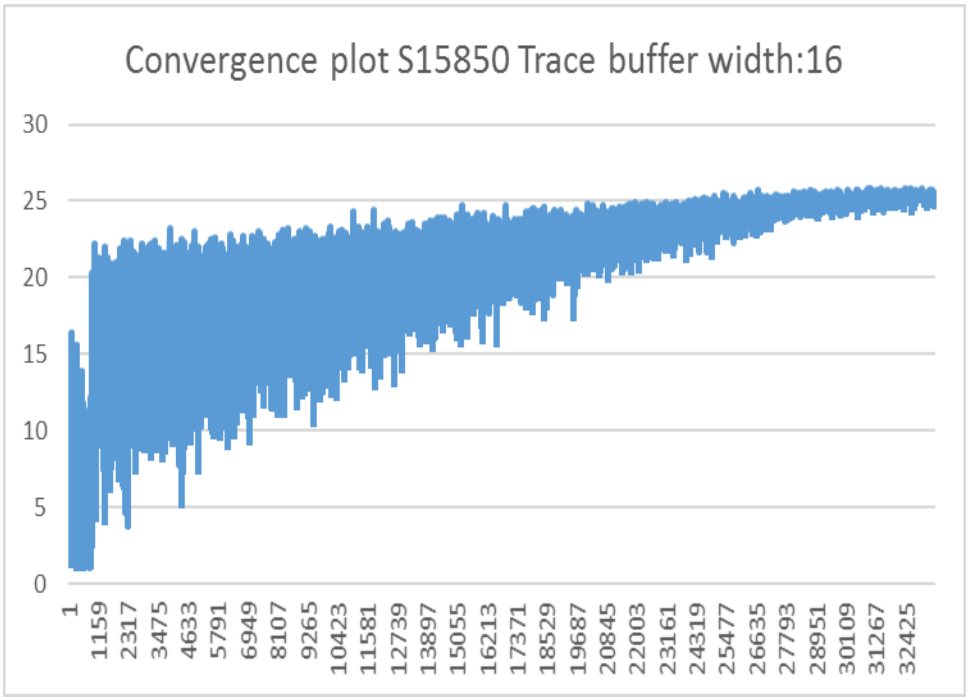


Fig. 4.17. Convergence plot for s15850 and trace buffer width 16 for the actual trace signal selection run

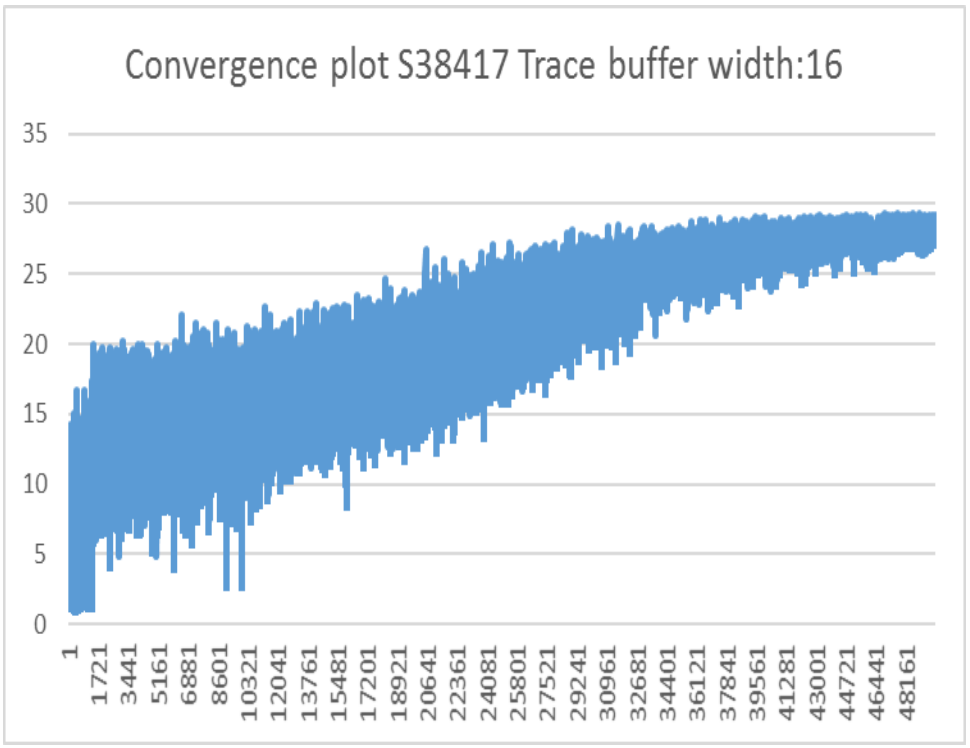


Fig. 4.18. Convergence plot for s38417 and trace buffer width 16 for the actual trace signal selection run

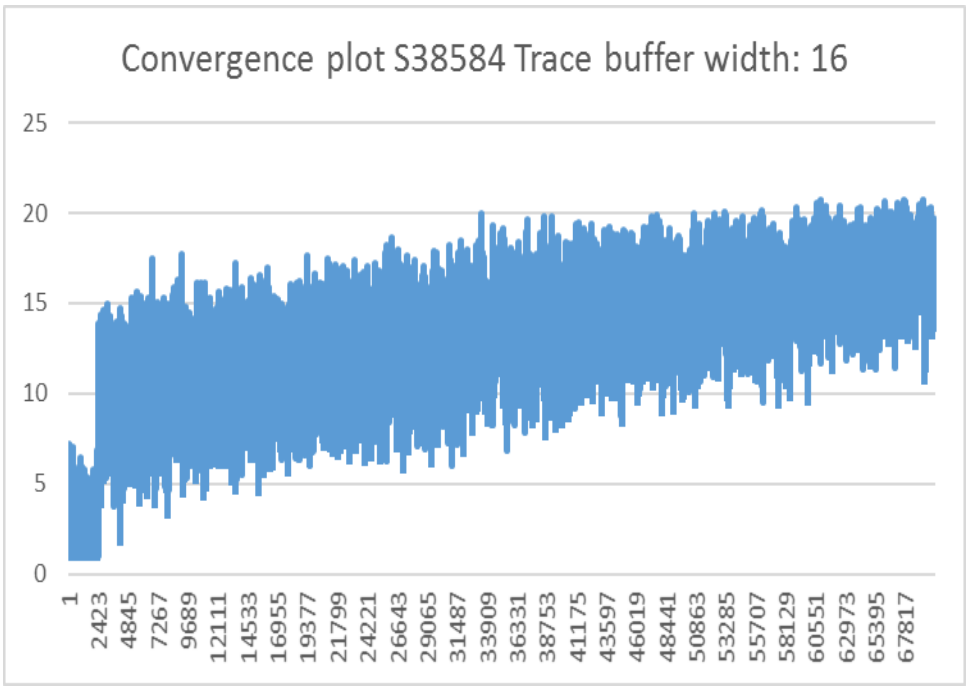


Fig. 4.19. Convergence plot for s38584 and trace buffer width 16 for the actual trace signal selection run

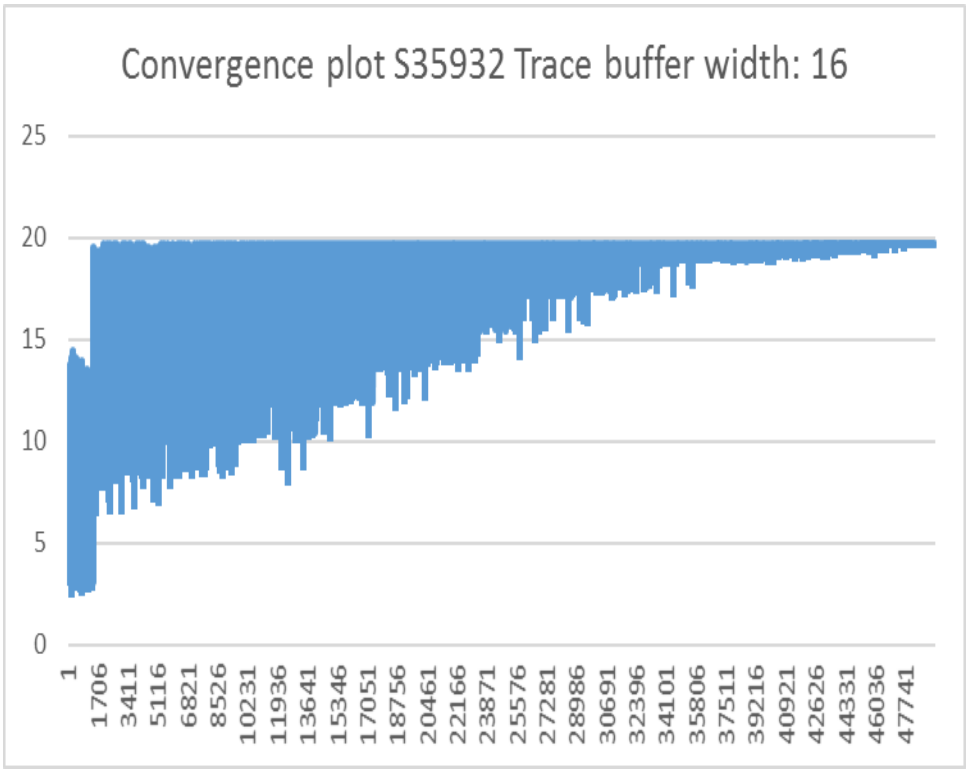


Fig. 4.20. Convergence plot for s35932 and trace buffer width 16 for the actual trace signal selection run

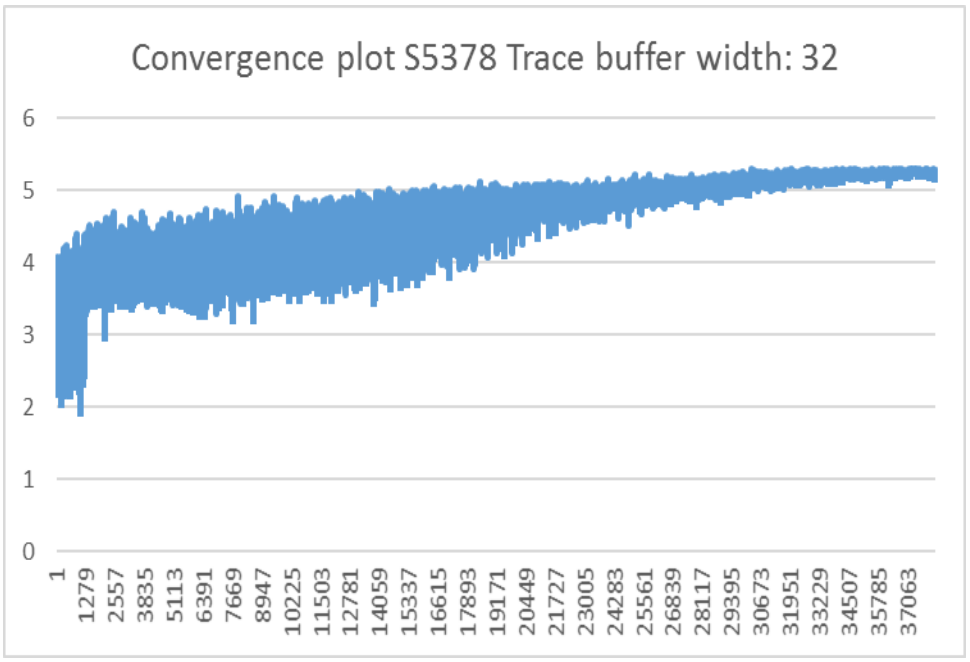


Fig. 4.21. Convergence plot for s5378 and trace buffer width 32 for the actual trace signal selection run

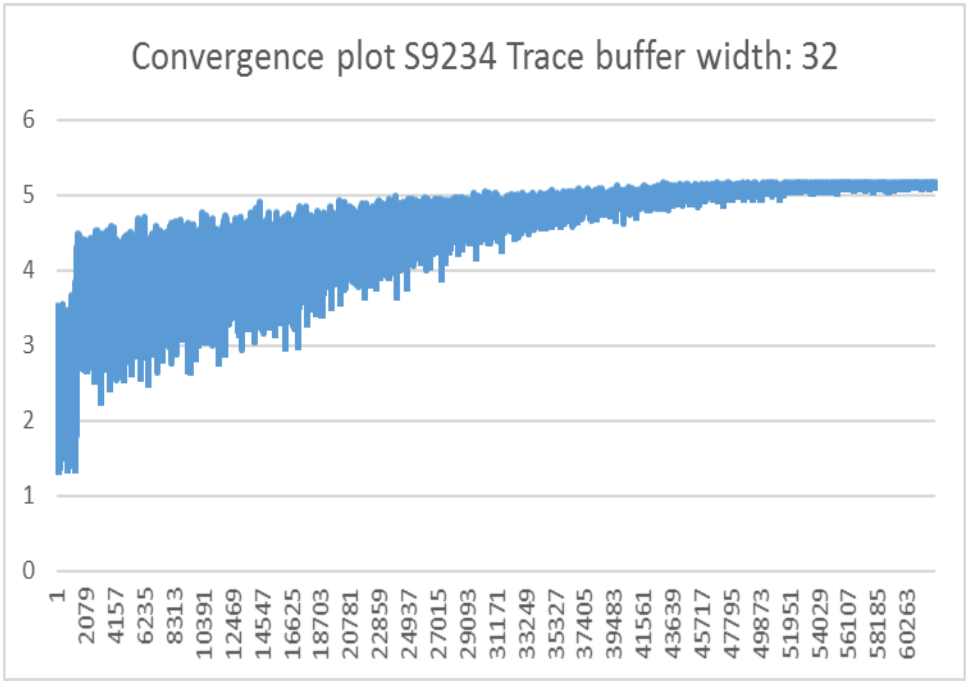


Fig. 4.22. Convergence plot for s9234 and trace buffer width 32 for the actual trace signal selection run

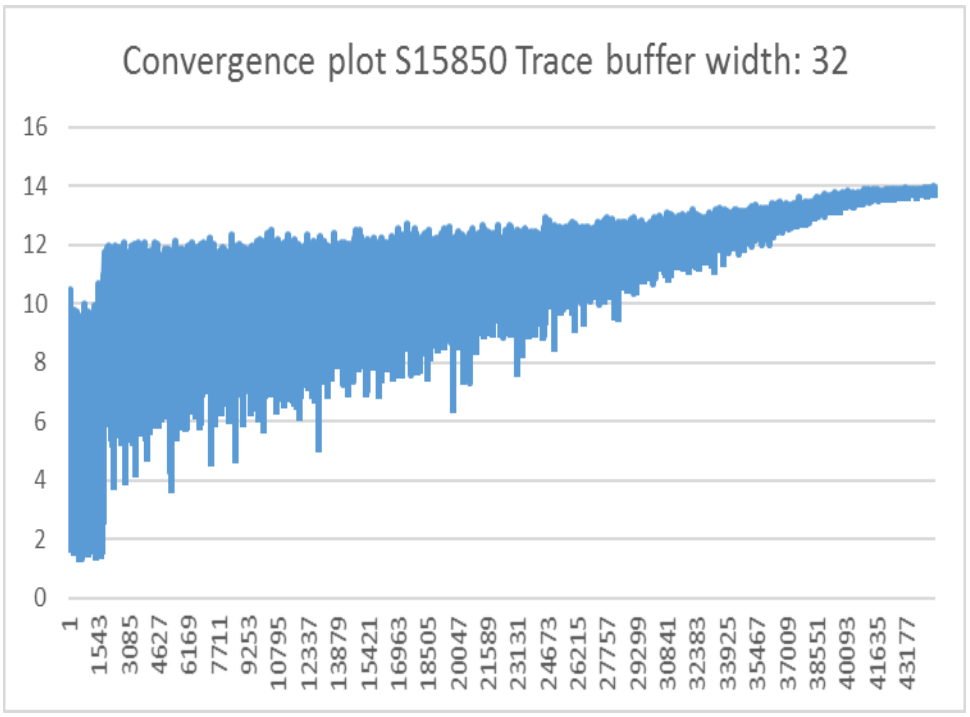


Fig. 4.23. Convergence plot for s15850 and trace buffer width 32 for the actual trace signal selection run

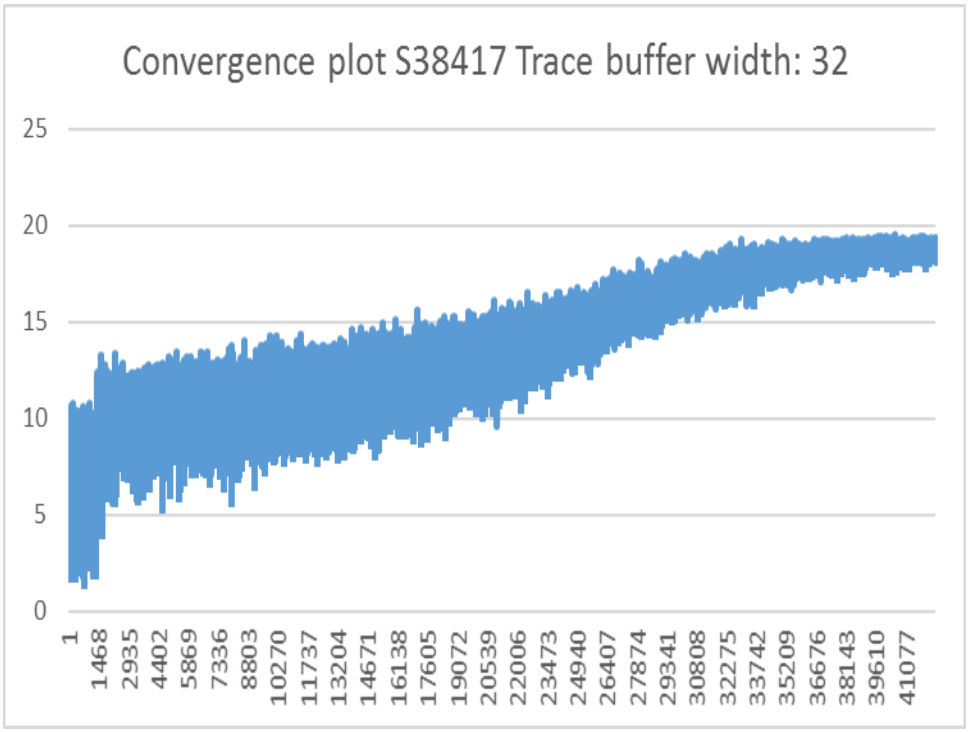


Fig. 4.24. Convergence plot for s38417 and trace buffer width 32 for the actual trace signal selection run

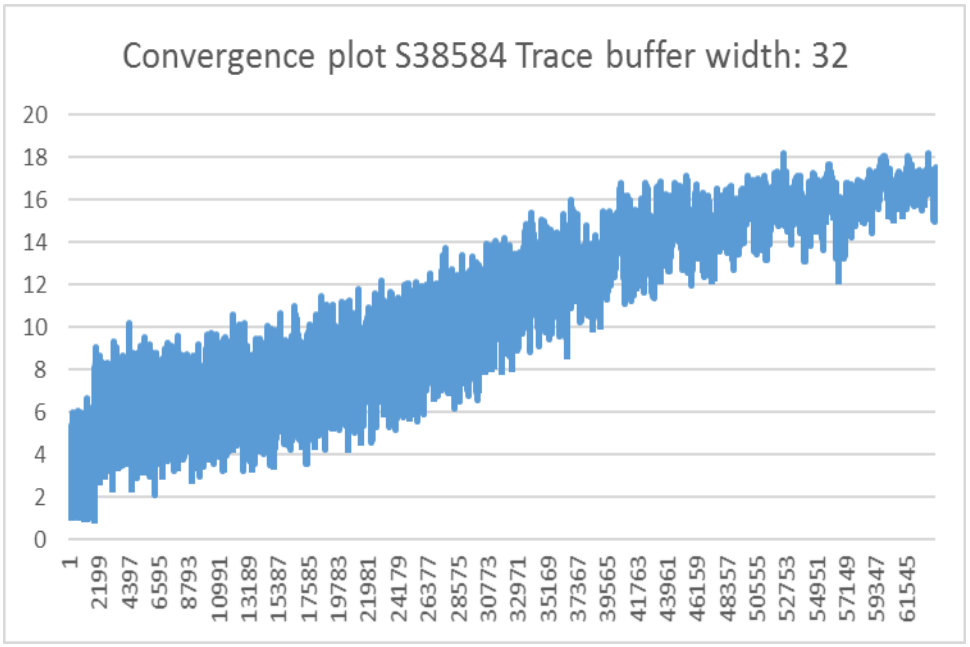


Fig. 4.25. Convergence plot for s38584 and trace buffer width 32 for the actual trace signal selection run

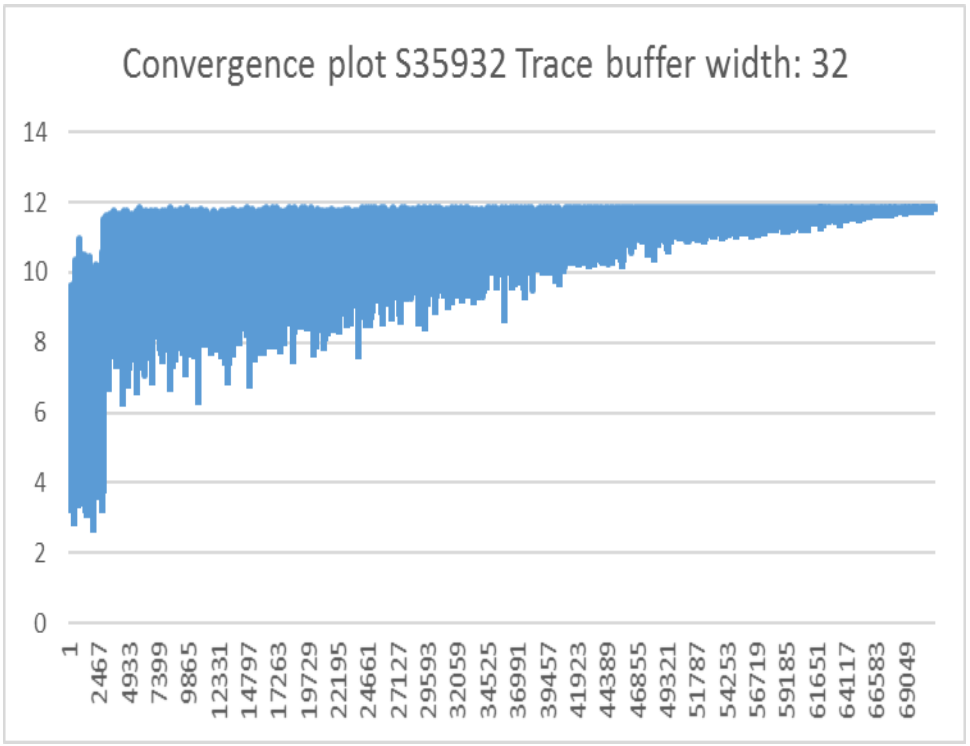


Fig. 4.26. Convergence plot for s35932 and trace buffer width 32 for the actual trace signal selection run

4.4 Evaluating Dependence on Input Vector

As stated earlier for a specific ISCAS 89 benchmark and trace buffer width, we launched 6 different runs. These 6 runs correspond to three different random seeds for obtaining simulation data from Modelsim® and two different windows of 64 cycles. From these 6 runs we obtain 6 different sets of flip-flops and we choose the set which has the highest score which corresponds to the average restoration ratio for 6 sets of input vectors (Each corresponding to 64 clock cycles). The future work of this thesis would be to feed these 6 sets of flip-flops into an ILP optimizer [21] which would then select the best signal set such that the total number of lost states in all runs is minimized. For now, we use the best average to select the trace signals. Next we present the table for each benchmark and a particular trace buffer width, showing how the restoration ratio for each set varies with the input vector.

Table. 4.7. Table to evaluate the best set of trace signals for S5378 trace buffer width 8

Set	RR1	RR2	RR3	RR4	RR5	RR6	Average	Standard deviation
Set 1	14.34	14.19	14.23	14.07	14.18	14.03	14.17	0.10
Set 2	14.33	14.33	14.33	14.16	14.34	14.14	14.27	0.09
Set 3	14.33	14.30	14.37	14.14	14.33	14.15	14.27	0.09
Set 4	14.32	14.33	14.35	14.17	14.35	14.14	14.28	0.09
Set 5	**14.33**	**14.33**	**14.35**	**14.16**	**14.35**	**14.15**	**14.28**	**0.09**
Set 6	13.70	14.02	14.00	13.74	13.92	14.23	13.93	0.18

Table. 4.8. Table to evaluate the best set of trace signals for S9234 trace buffer width 8

Set	RR1	RR2	RR3	RR4	RR5	RR6	Average	Standard deviation
Set 1	**11.65**	**12.00**	**12.46**	**11.78**	**12.22**	**11.53**	**11.94**	**0.33**
Set 2	11.28	12.51	11.28	12.43	11.47	11.98	11.82	0.51
Set 3	**11.65**	**12.00**	**12.46**	**11.78**	**12.22**	**11.53**	**11.94**	**0.33**
Set 4	10.97	11.71	10.60	12.63	10.23	11.43	11.26	0.78
Set 5	11.20	12.04	11.90	11.84	11.59	11.54	11.68	0.28
Set 6	10.12	12.25	10.05	11.59	10.38	12.20	11.10	0.94

Table. 4.9. Table to evaluate the best set of trace signals for S15850 trace buffer width 8

Set	RR1	RR2	RR3	RR4	RR5	RR6	Average	Standard deviation
Set 1	43.98	42.72	41.12	38.49	37.24	41.16	40.78	2.31
Set 2	32.87	43.98	34.04	36.08	33.92	34.83	35.95	3.72
Set 3	39.88	38.33	42.14	40.16	36.54	40.57	39.60	1.77
Set 4	31.86	39.14	33.25	44.58	34.79	39.65	37.21	4.36
Set 5	**41.93**	**41.58**	**40.21**	**38.69**	**42.04**	**40.75**	**40.87**	**1.17**
Set 6	36.96	41.56	35.55	39.70	35.17	44.05	38.83	3.24

Table. 4.10. Table to evaluate the best set of trace signals for S38417 trace buffer width 8

Set	RR1	RR2	RR3	RR4	RR5	RR6	Average	Standard deviation
Set 1	40.46	36.14	36.20	40.36	39.23	29.11	36.92	3.91
Set 2	**39.51**	**39.59**	**39.17**	**39.35**	**39.01**	**32.34**	**38.16**	**2.61**
Set 3	31.49	30.88	39.71	39.63	36.97	31.12	34.97	3.91
Set 4	36.04	35.76	40.08	40.31	40.03	32.00	37.37	3.06
Set 5	28.06	27.45	39.63	39.60	40.55	32.17	34.58	5.56
Set 6	32.39	35.29	27.96	31.45	40.23	40.38	34.62	4.55

Table. 4.11. Table to evaluate the best set of trace signals for S38584 trace buffer width 8

Set	RR1	RR2	RR3	RR4	RR5	RR6	Average	Standard deviation
Set 1	22.79	14.87	17.38	15.52	18.67	14.43	17.27	2.87
Set 2	20.77	22.39	18.62	19.35	22.53	22.36	21.00	1.56
Set 3	12.10	9.12	23.10	13.35	14.87	8.65	13.53	4.81
Set 4	9.67	10.96	8.62	22.54	13.50	13.49	13.13	4.58
Set 5	**21.67**	**22.39**	**18.63**	**19.36**	**22.53**	**22.36**	**21.16**	**1.57**
Set 6	3.02	9.00	11.95	3.50	4.99	22.93	9.23	6.89

Table. 4.12. Table to evaluate the best set of trace signals for S35932 trace buffer width 8

Set	RR1	RR2	RR3	RR4	RR5	RR6	Average	Standard deviation
Set 1	28.56	32.00	28.56	32.00	28.56	32.00	30.28	1.72
Set 2	28.56	32.00	28.56	32.00	28.56	32.00	30.28	1.72
Set 3	28.56	32.00	28.56	32.00	28.56	32.00	30.28	1.72
Set 4	28.56	32.00	28.56	32.00	28.56	32.00	30.28	1.72
Set 5	28.56	32.00	28.56	32.00	28.56	32.00	30.28	1.72
Set 6	28.56	32.00	28.56	32.00	28.56	32.00	30.28	1.72

Table. 4.13. Table to evaluate the best set of trace signals for S5378 trace buffer width 16

Set	RR1	RR2	RR3	RR4	RR5	RR6	Average	Standard deviation
Set 1	**9.16**	**8.55**	**8.43**	**8.59**	**8.67**	**8.55**	**8.66**	**0.24**
Set 2	8.58	8.72	8.50	8.51	8.52	8.48	8.55	0.08
Set 3	8.61	8.55	8.61	8.46	8.61	8.51	8.56	0.06
Set 4	8.42	8.45	8.08	8.75	8.73	8.38	8.47	0.23
Set 5	8.14	8.48	8.11	8.59	8.78	8.25	8.39	0.24
Set 6	7.84	8.26	7.81	8.36	8.56	8.68	8.25	0.33

Table. 4.14. Table to evaluate the best set of trace signals for S9234 trace buffer width 16

Set	RR1	RR2	RR3	RR4	RR5	RR6	Average	Standard deviation
Set 1	**8.91**	**7.84**	**8.30**	**8.36**	**8.20**	**8.17**	**8.30**	**0.32**
Set 2	8.10	8.47	7.80	8.29	8.09	8.15	8.15	0.20
Set 3	8.13	6.79	8.90	8.14	7.35	7.19	7.75	0.71
Set 4	7.95	7.58	8.74	8.67	8.27	7.74	8.16	0.44
Set 5	7.60	7.75	7.91	7.51	8.64	8.43	7.97	0.42
Set 6	7.67	7.93	8.04	7.72	8.63	8.50	8.08	0.36

Table. 4.15. Table to evaluate the best set of trace signals for S15850 trace buffer width 16

Set	RR1	RR2	RR3	RR4	RR5	RR6	Average	Standard deviation
Set 1	25.85	23.88	24.45	23.05	21.88	24.45	23.92	1.24
Set 2	20.11	26.29	21.93	20.39	21.96	21.94	22.10	2.02
Set 3	21.27	24.41	26.30	22.96	23.56	24.52	23.84	1.54
Set 4	22.11	23.14	23.26	26.17	23.27	25.61	23.93	1.45
Set 5	**21.51**	**23.70**	**23.31**	**24.93**	**25.57**	**24.99**	**24.00**	**1.36**
Set 6	21.30	20.29	21.34	23.56	21.42	25.84	22.29	1.86

Table. 4.16. Table to evaluate the best set of trace signals for S38417 trace buffer width 16

Set	RR1	RR2	RR3	RR4	RR5	RR6	Average	Standard deviation
Set 1	29.30	28.75	24.62	24.73	28.94	29.01	27.56	2.04
Set 2	29.05	28.92	24.32	24.34	28.80	26.35	26.96	2.07
Set 3	**26.98**	**26.31**	**29.20**	**29.09**	**29.51**	**24.29**	**27.56**	**1.89**
Set 4	24.88	24.67	29.27	29.36	28.92	22.70	26.63	2.65
Set 5	18.83	18.27	26.84	26.71	29.66	28.88	24.86	4.59
Set 6	11.20	10.56	10.92	10.98	18.45	29.89	15.33	7.07

Table. 4.17. Table to evaluate the best set of trace signals for S38584 trace buffer width 16

Set	RR 1	RR2	RR3	RR4	RR5	RR6	Average	Standard deviation
Set 1	20.71	17.00	20.24	18.82	17.27	19.40	18.91	1.39
Set 2	**18.22**	**20.11**	**17.88**	**20.39**	**19.13**	**19.63**	**19.23**	**0.93**
Set 3	20.49	12.21	20.59	17.76	17.43	7.17	15.94	4.81
Set 4	19.83	18.56	20.10	20.40	14.07	5.34	16.38	5.38
Set 5	19.18	16.68	19.99	16.81	20.63	16.47	18.29	1.70
Set 6	2.17	17.24	19.05	3.29	9.49	20.30	11.92	7.36

Table. 4.18. Table to evaluate the best set of trace signals for S35932 trace buffer width 16

Set	RR1	RR2	RR3	RR4	RR5	RR6	Average	Standard deviation
Set 1	19.72	19.72	19.72	19.72	19.72	19.72	19.72	0.00
Set 2	19.72	19.72	19.72	19.72	19.72	19.72	19.72	0.00
Set 3	19.72	19.72	19.72	19.72	19.72	19.72	19.72	0.00
Set 4	19.72	19.72	19.72	19.72	19.72	19.72	19.72	0.00
Set 5	19.72	19.72	19.72	19.72	19.72	19.72	19.72	0.00
Set 6	19.72	19.72	19.72	19.72	19.72	19.72	19.72	0.00

Table. 4.19. Table to evaluate the best set of trace signals for S5378 trace buffer width 32

Set	RR1	RR2	RR3	RR4	RR5	RR6	Average	Standard deviation
Set 1	5.30	4.85	4.73	4.78	4.98	4.85	4.92	0.19
Set 2	5.19	5.22	5.03	5.16	5.15	5.01	5.13	0.08
Set 3	**5.15**	**5.18**	**5.21**	**5.17**	**5.18**	**5.16**	**5.17**	**0.02**
Set 4	5.17	5.14	5.08	5.20	5.17	5.11	5.15	0.04
Set 5	5.00	5.10	4.90	5.15	5.22	5.10	5.08	0.10
Set 6	5.16	5.11	5.12	5.14	5.14	5.19	5.14	0.02

Table. 4.20. Table to evaluate the best set of trace signals for S9234 trace buffer width 32

Set	RR1	RR2	RR3	RR4	RR5	RR6	Average	Standard deviation
Set 1	**5.17**	**4.84**	**5.03**	**4.98**	**4.95**	**4.89**	**4.98**	**0.11**
Set 2	4.30	5.03	4.16	4.07	4.04	4.03	4.27	0.35
Set 3	4.80	4.67	5.40	5.01	4.98	4.92	4.96	0.23
Set 4	4.99	4.62	5.08	5.14	4.89	4.93	4.94	0.17
Set 5	5.05	4.35	5.35	4.88	5.11	4.95	4.95	0.31
Set 6	4.84	4.80	5.03	4.88	5.00	5.03	4.93	0.09

Table. 4.21. Table to evaluate the best set of trace signals for S15850 trace buffer width 32

Set	RR1	RR2	RR3	RR4	RR5	RR6	Average	Standard deviation
Set 1	13.97	13.37	13.62	12.85	13.32	13.53	13.44	0.34
Set 2	13.25	14.11	13.86	13.22	12.75	13.55	13.46	0.45
Set 3	**13.50**	**13.95**	**14.33**	**13.23**	**13.70**	**13.54**	**13.71**	**0.35**
Set 4	13.28	12.74	13.11	12.95	14.19	13.06	13.22	0.46
Set 5	12.42	13.79	13.04	14.27	12.58	12.08	13.03	0.77
Set 6	12.46	11.44	11.83	13.52	12.18	13.85	12.55	0.87

Table. 4.22. Table to evaluate the best set of trace signals for S38417 trace buffer width 32

Set	RR1	RR2	RR3	RR4	RR5	RR6	Average	Standard deviation
Set 1	19.54	19.22	18.23	18.25	19.47	19.29	19.00	0.55
Set 2	**19.22**	**19.42**	**19.27**	**19.32**	**19.06**	**18.00**	**19.05**	**0.48**
Set 3	17.96	18.14	19.46	19.08	19.07	14.15	17.98	1.79
Set 4	16.74	16.67	19.19	19.45	19.13	15.47	17.77	1.54
Set 5	19.36	18.63	17.10	17.08	19.80	16.56	18.09	1.24
Set 6	19.41	18.95	16.74	16.69	19.81	19.94	18.59	1.36

Table. 4.23. Table to evaluate the best set of trace signals for S38584 trace buffer width 32

Set	RR 1	RR2	RR3	RR4	RR5	RR6	Average	Standard deviation
Set 1	18.17	17.12	17.77	17.15	16.98	17.68	17.48	0.43
Set 2	**17.24**	**18.19**	**17.54**	**16.93**	**17.42**	**17.65**	**17.49**	**0.39**
Set 3	18.11	17.24	18.42	10.34	17.25	17.65	16.50	2.79
Set 4	15.94	17.62	17.39	18.49	17.66	17.87	17.49	0.78
Set 5	11.20	9.88	16.98	15.34	18.03	15.54	14.49	2.96
Set 6	9.57	16.85	17.67	16.79	17.33	18.20	16.07	2.95

Table. 4.24. Table to evaluate the best set of trace signals for S35932 trace buffer width 32

Set	RR 1	RR2	RR3	RR4	RR5	RR6	Average	Standard deviation
Set 1	11.85	11.85	11.85	11.85	11.85	11.85	11.85	0.00
Set 2	12.30	12.30	12.29	12.30	12.29	12.30	12.30	0.00
Set 3	**12.29**	**12.31**	**12.30**	**12.30**	**12.30**	**12.30**	**12.30**	**0.01**
Set 4	12.30	12.30	12.30	12.30	12.30	12.30	12.30	0.00
Set 5	12.30	12.30	12.30	12.31	12.30	12.31	12.30	0.00
Set 6	12.32	12.31	12.32	12.32	12.30	12.33	12.32	0.01

We also evaluated the dependence on Input vector for sets of flip-flops which would give poor restoration quality for a specific input vector. We present tables for a few benchmarks with trace buffer width set to 8, showing how restoration ratio varies W.R.T input vector for a set of flip-flops which have poor restoration quality for one input vector.

Table. 4.25. Benchmark: S5378 Trace Buffer Width: 8, Evaluating dependence on input vector for sets of flip-flops having poor restoration quality

Set	RR 1	RR2	RR3	RR4	RR5	RR6	Average	Standard deviation
Set 1	1.49	1.49	2.06	1.49	1.79	1.78	1.68	0.21
Set 2	1.44	1.42	1.42	1.42	1.43	1.43	1.43	0.01
Set 3	1.39	1.37	1.37	1.37	1.37	1.37	1.37	0.01
Set 4	1.86	1.86	1.86	1.86	1.86	1.86	1.86	0.00
Set 5	1.74	1.74	1.75	1.74	1.74	1.74	1.74	0.01
Set 6	1.25	1.25	1.25	1.25	1.25	1.25	1.25	0.00

Table. 4.26. Benchmark: S9234 Trace Buffer Width: 8, Evaluating dependence on input vector for sets of flip-flops having poor restoration quality

Set	RR 1	RR2	RR3	RR4	RR5	RR6	Average	Standard deviation
Set 1	1.04	1.04	1.03	1.01	1.03	1.04	1.03	0.01
Set 2	1.25	1.25	1.25	1.25	1.25	1.25	1.25	0.00
Set 3	1.37	1.37	1.37	1.37	1.37	1.37	1.37	0.00
Set 4	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00
Set 5	1.49	1.22	1.12	1.48	1.12	1.48	1.32	0.17
Set 6	1.74	1.74	1.74	1.74	1.74	1.74	1.74	0.00

Table. 4.27. Benchmark: S15850 Trace Buffer Width: 8, Evaluating dependence on input vector for sets of flip-flops having poor restoration quality

Set	RR 1	RR2	RR3	RR4	RR5	RR6	Average	Standard deviation
Set 1	1.79	1.78	1.88	1.92	1.88	1.77	1.84	0.06
Set 2	1.21	1.25	1.23	1.24	1.22	1.27	1.24	0.02
Set 3	1.62	1.70	1.75	1.68	1.74	1.71	1.70	0.04
Set 4	1.25	1.27	1.26	1.28	1.24	1.30	1.27	0.02
Set 5	1.40	1.43	1.50	1.46	1.38	1.57	1.46	0.06
Set 6	1.87	1.92	1.93	1.95	1.90	1.94	1.92	0.03

Conclusion from these tables: Restoration ratio for both the best case and worst case sets remain fairly consistent W.R.T the input vector barring a few exceptions. So averaging over 6 sets of input vectors should provide a good estimate of restoration ratio.

4.5 Comparison with Conventional Methods

Table. 4.28. Restoration quality of existing trace signal selection approaches [21]

Circuit	#Flip-flops	Buffer Width	Simulation based [19]	Hybrid [20]	ILP [21]
S5378	179	8	13.41	13.32	14.63
		16	7.35	7.26	9.26
		32	4.47	4.27	5.11
S9234	211	8	13.98	14.58	15.97
		16	8.3	8.55	9.32
		32	4.46	4.46	5.53
S15850	534	8	26.33	27.38	45.89
		16	19.89	20.65	25.82
		32	13.19	13.19	13.97
S38584	1426	8	19.73	25.87	159.1
		16	28.39	29.01	48.39
		32	32.45	34.62	44.46
S38417	1636	8	29.23	51.01	53.47
		16	17.02	19.22	26.87
		32	15.14	13.25	17.22
S35932	1728	8	132	139.52	185.1
		16	67.45	71.36	93.2
		32	34.63	35.08	47.13

The above table shows the restoration quality of existing trace signal approaches. Our goal was to get an improvement over the conventional simulation based [19] approach and also the hybrid approach [20]. The ILP method has merit and can be applied to our approach as well.

Table. 4.29. Comparison of Simulation based approach with our method

Circuit	#Flip-flops	Trace Buffer Width	Simulation based [19]	Our method	Improvement over [19]
S5378	179	8	13.41	14.28	6.49%
		16	7.35	8.66	17.82%
		32	4.47	5.17	15.66%
S9234	211	8	13.98	11.94	-14.59%
		16	8.3	8.3	0.00%
		32	4.46	4.98	11.66%
S15850	534	8	26.33	40.87	55.22%
		16	19.89	24	20.66%
		32	13.19	13.71	3.94%
S38584	1426	8	19.73	21	6.44%
		16	28.39	19.23	-32.26%
		32	32.45	17.49	-46.10%
S38417	1636	8	29.23	38.16	30.55%
		16	17.02	27.56	61.93%
		32	15.14	19.05	25.83%
S35932	1728	8	132	30.28	-77.06%
		16	67.45	19.72	-70.76%
		32	34.63	12.3	-64.48%

Barring one benchmark (S35932) where there seems to be a mismatch in the simulation data used by other conventional methods and our method, our approach does quite well in comparison to [19]. We obtain an improvement in restoration ratio up to 61.93%. We do note however that in a few cases our approach yields inferior results when compared to [19].

Table. 4.30. Comparison of Hybrid based approach with our method

Circuit	#Flip-flops	Buffer Width	Hybrid [20]	Our method	Improvement over [20]
S5378	179	8	13.32	14.28	7.21%
		16	7.26	8.66	19.28%
		32	4.27	5.17	21.08%
S9234	211	8	14.58	11.94	-18.11%
		16	8.55	8.3	-2.92%
		32	4.46	4.98	11.66%
S15850	534	8	27.38	40.87	49.27%
		16	20.65	24	16.22%
		32	13.19	13.71	3.94%
S38584	1426	8	25.87	21	-18.82%
		16	29.01	19.23	-33.71%
		32	34.62	17.49	-49.48%
S38417	1636	8	51.01	38.16	-25.19%
		16	19.22	27.56	43.39%
		32	13.25	19.05	43.77%
S35932	1728	8	139.52	30.28	-78.30%
		16	71.36	19.72	-72.37%
		32	35.08	12.3	-64.94%

Our method performs up to 49.27% better than the hybrid approach. Barring the benchmark S35932, for which as stated earlier there seems to be a mismatch in the simulation data, our approach again yields better results when compared to the hybrid approach. It is to be noted that there are a few cases in which the hybrid approach performs better than our approach.

Table. 4.31. Comparison of ILP based approach with our method

Circuit	#Flip-flops	Buffer Width	ILP [21]	Our method
S5378	179	8	14.63	14.28
		16	9.26	8.66
		32	5.11	5.17
S9234	211	8	15.97	11.94
		16	9.32	8.3
		32	5.53	4.98
S15850	534	8	45.89	40.87
		16	25.82	24
		32	13.97	13.71
S38584	1426	8	159.1	21
		16	48.39	19.23
		32	44.46	17.49
S38417	1636	8	53.47	38.16
		16	26.87	27.56
		32	17.22	19.05
S35932	1728	8	185.1	30.28
		16	93.2	19.72
		32	47.13	12.3

In any simulation based approach, trace signals may be different in different runs depending on the generated random input vector seed and also the window of tracing. The goal of the ILP refinement is to eliminate the influence of randomness and also to cover more states of a given circuit through selected signals. To do so, the authors of [21] used multiple runs of the signal selection algorithm which is then processed by ILP to select the best signal set among all outcomes. The same methodology can be applied to our approach, as stated before we launch six different runs for a given benchmark and trace buffer width. Corresponding to these six runs we get six sets of trace signals. We take each of these six sets of trace signals and calculate its restoration ratio W.R.T each

of the 6 input vectors. After which we select the set which has the best average restoration ratio. We could replace this step with the ILP refinement approach, we feed the six sets of signals into the ILP optimizer which would return a set of signals (equal to trace buffer width) such that minimum number of states are lost over all the runs. This would greatly enhance the restoration quality, as the base signal selection algorithm used by the authors of [21] is a greedy approach which limits the quality of restoration obtained. We note that even without the ILP optimization step, our approach performs better than the ILP approach for a few cases.

Conclusion from these comparisons: Between the simulation based approach, the hybrid approach and our approach there is no method which gives better results for all the benchmarks. The ILP method has merit and if its initial greedy signal selection approach is replaced by the simulated annealing method, it would yield great results. Even without the ILP optimization step in our methodology we have got better results than the original ILP methodology in a few cases. Hence there is no clear winner among all the approaches, and it is ideal for designers to launch all methods and pick the one which gives the best restoration ratio for that circuit.

4.6 Summary of Chapter 4

In this Chapter we first described our entire experimental setup. After which we compared the original restoration algorithm to the improved restoration algorithm by performing short simulated annealing trace signal selection runs. This was followed by presenting the simulated annealing convergence plots for the actual trace signal selection runs. Then we presented our results for the experiment to find the dependence of restoration ratio on input vector. Finally, we compared our trace signal selection approach to the existing

trace signal selection approaches. In the next chapter we summarize the entire thesis and provide insight into potential future work.

Chapter 5
Conclusion and Future Work

5.1 Conclusion

In this work we have developed a novel simulation based approach to select trace signals which takes a gate level netlist as input and gives a list of flip-flops which should be tapped onto the trace buffer. The selection of flip-flops is done in a manner so as to maximize the amount of signals that can be restored by these flip-flops. We viewed this as a partitioning problem, which led us to using the simulated annealing heuristic for this problem. We also found and fixed a hole in the original state restoration algorithm. Our methodology works well for most ISCAS 89 benchmarks, it yields up to 61.93% improvement in restoration ratio over the simulation based approach [19], up to 49.27% improvement over the hybrid approach and up to 10.62% over the ILP [21] approach. It has been explained in Chapter 4 how the ILP method can be integrated into our methodology, this would further improve the restoration ratio. We also conducted experiments to show the correlation between restoration ratio and input vector. We observed that restoration ratio remains fairly consistent W.R.T input vector, barring a few exceptions. The runtime for our approach is fairly high, as our primary goal was to maximize restoration ratio regardless of the run time. The advantage of using our approach is that runtime can be controlled as per the user's requirement, by changing the stop criteria. We have observed that reasonably good results can be obtained with a much shorter runtime using different stop criteria.

5.2 Future Work

In this section we discuss potential future work of this thesis.

5.2.1 Integration of ILP Filtering

We have discussed this extensively in Chapter 4. Once we get different sets of trace signals corresponding to different input vectors, we can feed these sets into the ILP optimizer which would return a set of trace signals such that minimum number of states are lost over all runs.

5.2.2 Incremental Restoration Method

Whenever our simulated annealing tool makes a call to the logic simulation tool, restoration is recomputed entirely, even though the only difference between two consecutive calls to the logic simulation is 1 flip-flop (Because of the move function, which swaps one flip-flop in the trace buffer list with some other flip-flop not in the trace buffer list). This work was attempted as a part of this thesis, but could not be executed successfully because of the way the logic simulation tool is designed (Restoring each node 64 clock cycles at a time). However, if the design of the logic simulation tool is changed, incremental restoration must be possible. This would lead to a huge reduction in the run time and lead to simulated annealing to search for many more possible states.

5.2.3 Identifying Critical Unreachable Flip-flops

It is possible that some flip-flops cannot be restored by the principal operations forward propagation and backward propagation.

If this knowledge is used to guide the selection of trace signals (The unreachable flip-flops should be a part of the trace signal list), it should lead to better restoration ratios.

6. References

[1] Wikipedia, the free encyclopedia. Formal verification https://en.wikipedia.org/wiki/Formal_verification .

[2] Sini Balakrishnan, Formal Verification – An Overview. http://vlsi.pro/formal-verification-an-overview/ .

[3] S. Mitra ; Dept. of EE and Dept. of CS, Stanford University, Stanford, CA, USA ; S. A. Seshia ; N. Nicolici. Post-silicon validation opportunities, challenges and recent advances.

[4] HO FAI KO, B.Eng. & Mgt., M.A.Sc. New Algorithms and Architectures for Post-Silicon Validation.

[5] D. D. Josephson ; Hewlett-Packard Co., Fort Collins, CO, USA. The manic depression of microprocessor debug.

[6] C. MacNamee and D. Heffernan, "Emerging on-ship debugging techniques for real-time embedded systems," in Computing & Control Engineering Journal, vol. 11, no. 6, pp. 295-303, Dec. 2000.
doi: 10.1049/cce:20000608.

[7] H. F. Ko and N. Nicolici, "Algorithms for State Restoration and Trace-Signal Selection for Data Acquisition in Silicon Debug," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 28, no. 2, pp. 285-297, Feb. 2009.
doi: 10.1109/TCAD.2008.2009158

[8] Morris, K. "On-Chip Debugging—Built-in Logic Analyzers on your FPGA." (2004).

[9] Sarangi, Smruti, et al. "Patching processor design errors with programmable hardware." Micro, IEEE 27.1 (2007): 12-25.

[10] Abrmovici, M. A reconfigurable design-for-debug infrastructure for SoCs,(2006) Proceedings. In Design Automation Conference (pp. 7-12).

[11] Riley, Mack W., and Mike Genden. "Cell broadband engine debugging for unknown events." IEEE Design & Test of Computers 5 (2007): 486-493.

[12] Mayer, Albrecht, Harry Siebert, and Klaus D. McDonald-Maier. "Boosting debugging support for complex systems on chip." *Computer* 4 (2007): 76-81.

[13] Burtscher, Martin, et al. "The VPC trace-compression algorithms." Computers, IEEE Transactions on 54.11 (2005): 1329-1344.

[14] Anis, Ehab, and Nicola Nicolici. "On using lossless compression of debug data in embedded logic analysis." Test Conference, 2007. ITC 2007. IEEE International. IEEE, 2007.

[15] Anis, Ehab, and Nicola Nicolici. "Interactive presentation: Low cost debug architecture using lossy compression for silicon debug." Proceedings of the conference on Design, automation and test in Europe. EDA Consortium, 2007.

[16] Liu, Xiao, and Qiang Xu. "Trace signal selection for visibility enhancement in post-silicon validation." Proceedings of the Conference on Design, Automation and Test in Europe. European Design and Automation Association, 2009.

[17] Basu, Kaustav, and Prabhat Mishra. "RATS: restoration-aware trace signal selection for post-silicon validation." Very Large Scale Integration (VLSI) Systems, IEEE Transactions on 21.4 (2013): 605-613.

[18] Shojaei, Hamid, and Azadeh Davoodi. "Trace signal selection to enhance timing and logic visibility in post-silicon validation." Proceedings of the International Conference on Computer-Aided Design. IEEE Press, 2010.

[19] Chatterjee, Debapriya, Calvin McCarter, and Valeria Bertacco. "Simulation-based signal selection for state restoration in silicon debug." Computer-Aided Design (ICCAD), 2011 IEEE/ACM International Conference on. IEEE, 2011.

[20] Li, Min, and Azadeh Davoodi. "A hybrid approach for fast and accurate trace signal selection for post-silicon debug." Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on 33.7 (2014): 1081-1094.

[21] Rahmani, Kamran, Prabhat Mishra, and Sambaran Ray. "Efficient trace signal selection using augmentation and ILP techniques." Quality Electronic Design (ISQED), 2014 15th International Symposium on. IEEE, 2014.

[22] http://www.willnaylor.com/mantext/wnanl.txt

[23]https://www.mentor.com/company/higher_ed/modelsim-student-edition

[24] https://filebox.ece.vt.edu/~mhsiao/iscas89.html