

# RTL Test Point Insertion to Reduce Delay Test Volume

Kedarnath J. Balakrishnan  
NEC Laboratories America  
4 Independence Way, Princeton, NJ  
bala@nec-labs.com

Lei Fang  
Dept. of Electrical and Computer Engg.  
Virginia Tech, Blacksburg, VA  
leifang@vt.edu

## Abstract

*In this paper, a novel test point insertion methodology is presented for RTL designs that aims to reduce the data volume of scan-based transition delay tests. Test points are identified based on functional information of RTL primitives using a satisfiability based algorithm. A subset of scan flip-flops is identified for conversion to enhanced-scan, i.e., the values are stored in two flip-flops thereby removing the circuit dependency of the second pattern in broadside transition tests. Using the proposed methodology, the number of specified bits required to test transition faults is reduced thus improving test set compaction. The advantage of test point insertion at RTL is that the extra delay due to multiplexers can be absorbed during logic synthesis. Experimental results show that the proposed methodology can reduce transition test data volume by more than 30% with 1% area overhead and without violating timing constraints.*

## 1 Introduction

Manufacturing test of current generation complex integrated chips is a critical part of the design process. Deep sub-micron technology nodes require new fault models to explain the defects that occur during the manufacturing process. To maintain good test quality and low test escapes, test generation is required for many types of faults. Scan-based delay testing (esp. transition fault testing) has become an integral part of the structural Design-for-Test (DFT) flow. However, the test data volume required for delay testing is usually much higher than the test data volume required for stuck-at fault testing. Moreover, the memory present in external automated test equipment (ATE) to store test data has not kept pace with the increase in test data volume.

Detecting a transition delay fault requires application of two test patterns [8]; the first one to initialize the desired value in the target circuit line and the second pattern to launch the transition at the circuit line and propagate it to one or more primary outputs or scan flip-flops. There are two popular approaches to apply transition tests in full scan design environments. The difference between the two approaches is in the way the second pattern is derived from the first pattern. The first pattern in both the approaches

is shifted in through the scan chains, similar to stuck-at pattern testing. In the launch-on-shift (LOS) or *skewed-load* approach, the second pattern is obtained by shifting the first pattern one more clock cycle. In the launch-on-capture (LOC) or *broad-side* approach, the second pattern is the circuit response to the first pattern. Thus, in both the approaches mentioned above, because of the dependency of the second pattern, the possible combinations of two patterns that can be applied to the circuit is restricted. Hence transition fault coverage is significantly lower than stuck-at fault coverage. One way to remove the dependency is to use two separate flip-flops in the scan cell to store the values of the first and second time-frames. However, this “enhanced scan” approach requires much higher hardware overhead as compared to normal scan. A disadvantage of using the LOS approach is that it requires the scan enable signal to be shifted at system clock (to get the second pattern) which involves more design and routing overhead. In this work, we assume that a LOC (*broad-side*) approach is used for applying transition delay test patterns to the circuit.

Test point insertion [1] has been traditionally used to improve the fault coverage of designs with low fault coverage due to their structure. By inserting test points at the input (output) path of the target fault, controllability (observability) is improved and test generation for the fault is made easier. The parts of the gate level netlist that are difficult to control or observe, *i.e.*, where test points need to be inserted, can be identified using either the structural information or a test generation based technique or a combination of the two. The disadvantage of adding test points is that they modify the functional path of the design resulting in area as well as timing/delay overhead. However, if test points can be identified and inserted at the Register-Transfer Level (RTL) design, the area and delay overheads can be absorbed during logic synthesis to the gate level.

Identifying test points at RTL is difficult since structural information is absent. Depending on the synthesis constraints, the same RTL design can map into several gate level structures. Previous work on RTL test point insertion has focused on improving the stuck-at fault coverage during Built-In Self-Test (BIST) [2, 3, 7]. In these techniques, test points are inserted in the functional path after performing

testability analysis.

In this paper, we describe a technique to identify and insert test points at RTL to reduce the volume of scan-based transition test patterns. Test points are inserted in the scan-paths and do not modify the functionality of the circuit. The dependency of the second pattern in transition test is reduced using the proposed test points which significantly decrease the number of inputs (and flip-flops) that need to be specified to detect a particular fault. Reducing the specified bits for each fault results in higher test pattern compaction as well as higher test compression. The proposed technique uses a very fast satisfiability (SAT) based algorithm with rules depending on the function of the RTL component to identify scan flip-flops that are difficult to control. By inserting the test points identified using the proposed paper, the data volume of transition test patterns can be reduced significantly while the impact on area and delay are negligible.

## 2 Background

### 2.1 Test Generation for Transition Faults

Test generation for transition faults using LOC (broad-side) approach is similar to two time-frame sequential test generation. It is illustrated below using the example circuit in Fig. 1 which has been expanded into two time-frames. Without any loss of generality, assume that primary inputs and outputs are also scanned. In the first time-frame, a pattern is generated to initialize the transition at the faulty gate. For a slow-to-rise (slow-to-fall) transition fault, this means justifying the faulty line to 0 (1). In the second time-frame, a stuck-at-1 (stuck-at-0) test pattern is required to provoke the transition and sensitize a path to the circuit output. Since the values of the scan flip-flops required for the second time-frame are captured from the circuit responses, more inputs and scan flip-flops need to be specified in the first time-frame to detect transition faults as compared to stuck-at faults. This results in fewer chances of test pattern compaction and thus leads to higher test pattern volume.

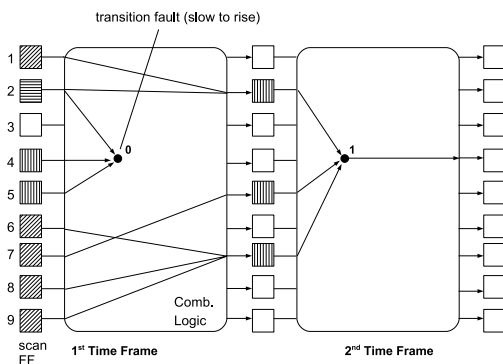


Figure 1. Detecting a Transition Fault

Consider the example in Fig. 1. To generate a transition test pattern for the slow-to-rise fault shown in Fig. 1, the circuit line has to be justified to 0 in the first time-frame and a stuck-at-1 test is required in the second time-frame. For each of these steps, several scan flip-flops need to be specified. In Fig. 1, scan flip-flops 2, 4 & 5 are required to initialize the faulty node to 0 in the first time-frame. In the second time-frame, scan flip-flops 2, 5 & 7 are required to set the faulty node to 1 and propagate the fault to outputs. To capture the required value of scan flip-flop 2 in the second time-frame, scan flip-flops 1 & 2 need to be set in the first time-frame. Similarly to capture required values in scan flip-flops 5 & 7, scan flip-flops 7 and scan flip-flops 6, 8 & 9 need to be specified respectively in the first time-frame. Hence a total of 8 scan flip-flops need to be specified to detect the given fault.

### 2.2 Boolean Satisfiability (SAT)

Boolean Satisfiability refers to the problem of determining a satisfying variable assignment for a Boolean function, or a proof that no such variable assignment exists, which means the function is unsatisfiable. A SAT problem takes as input a propositional formula that is usually represented in the conjunctive normal form (CNF). The CNF is a conjunction of several clauses, each of which is a disjunction of literals. A literal is a variable in its positive or negative polarity.

Generally any circuit representing a Boolean function can be transformed into a propositional formula. For instance, a two input AND gate with input  $a, b$  and output  $o$  can be represented as  $(a + \bar{o})(b + \bar{o})(\bar{a} + \bar{b} + o)$ . All circuit elements can be transformed in a similar manner. Test generation for a circuit can also be transformed into a Boolean propositional formula [4]. Each circuit element is represented in one or more clauses by using input/output propagation rules. The justification and propagation events for a fault are added as Boolean implications. SAT solvers like ZCHAFF [6] can then be used to either find a satisfying variable assignment (if a test exists) for the propositional formula or prove that the fault is untestable if the formula is unsatisfiable. For transition delay faults using LOC approach, test pattern generation is given by two excitation conditions in the two time-frames and a set of propagation conditions in the second time-frame. If we transform these conditions along with the circuit into a SAT formula, the satisfiability of this formula indicates whether the corresponding transition fault is testable or not.

Recent advances in the area of satisfiability [9] make it possible to identify the set of clauses that cause the SAT instance to be unsatisfiable. These clauses form the unsatisfiable segment. From the unsatisfiable segment, a minimal set of clauses (called unsatisfiable core) can be extracted. Extracting the unsatisfiable (UNSAT) core is explained in

detail in [5], where it is utilized for inserting non-scan DFT. The UNSAT core is useful in identifying the regions of the circuit which are difficult to solve. Other information about the circuit can be obtained from conflict analysis learned clauses which are generated by the SAT engine while solving a SAT instance during test generation. In this work, we use both the UNSAT core and the learned clauses to identify specific characteristics of the circuit for test point insertion.

### 3 Test Point Insertion at RTL

#### 3.1 Key Idea

The key idea of the proposed technique is to insert test points such that the number of specified bits required in transition fault testing is reduced. This is done by reducing the dependency of the second time-frame pattern on the first time-frame pattern. The scan flip-flops in the second time-frame which require a large number of scan flip-flops to be specified in the first time-frame are controlled directly, thereby reducing the number of specified bits to detect transition faults. This results not only in higher static and dynamic compaction but also higher test compression since the effectiveness of most compression schemes is directly related to the number of specified bits in test patterns. In the proposed technique, we insert test points such that these flip-flops are directly loaded with the second time-frame value using a multiplexer. The test point is an extra flip-flop that stores the required second time-frame value.

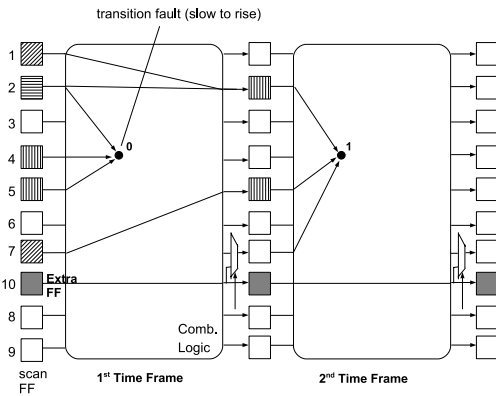


Figure 2. Example Test Point Insertion

Consider again the example in Fig. 1. Figure 2 shows an example of the proposed test point insertion where the second time-frame value required for scan flip-flop 7 is directly stored in scan flip-flop 10 in the first time-frame. Scan flip-flop 10 (fully shaded in Fig. 2) is the inserted test point. Flip-flops 6, 8 and 9 which needed to be specified earlier to detect the fault need not be specified after adding this test point. Hence the number of specified bits required to detect the same fault reduces to 6.

#### 3.2 Test Point Design

As mentioned above, the proposed test point is an additional flip-flop which is added to the scan chain so that it stores the required second time-frame value. As seen from Fig. 2, an additional multiplexer is required which selects between the combinational circuit output and the test point. The select input of the multiplexer need to be generated such that it has opposite values in the two time-frames. For the test point of Fig. 2, the select input should be 1 in the first time-frame so that the value stored in the test point flip-flop is captured and 0 in the second time-frame so that the circuit response is captured. If multiple test points are inserted in the design, the select inputs of the multiplexers for all the inserted test points can be shared. Ideally, the select input should be generated such that any ATPG can understand it. Figure 3 shows a way of generating the select signal by using an additional scan flip-flop. In Fig. 3,  $FF_i$  and  $FF_j$  are selected to be inserted with test points which are  $FF_{TP1}$  and  $FF_{TP2}$  respectively. An additional flip-flop  $FF_s$  is used to generate the select signal. With this method, to insert  $k$  test points in the design,  $k + 1$  flip-flops and  $k$  multiplexers are required. The advantage of performing insertion at RTL is that the extra area required for the multiplexers can be assimilated during logic synthesis.

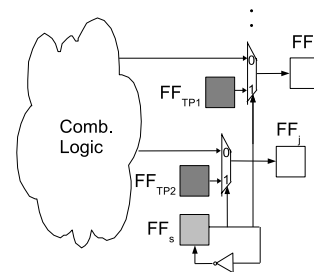


Figure 3. Generating the select signal

#### 3.3 Selecting Test Points

In this work, test points are identified based on the functional information of RTL elements. If behavioral RTL is given, we analyze and elaborate the design into a temporary structural RTL netlist which gives a basic sense of circuit topology. A fast search is then done on this temporary structural RTL netlist to identify flip-flops that are (i) specified most in test patterns for transition faults and (ii) difficult to control in the second time-frame. Since adequate structural information is not available, the traditional controllability/observability analysis techniques cannot capture all the information. A functional approach is required and since RTL primitives like adders, multiplexers etc. can be easily transformed into CNF formula, a SAT based heuristic is used.

In the proposed scheme, a fast (and approximate) test generation for transition faults is done using SAT on the temporary RTL netlist which is derived from a behavioral description using a verilog/VHDL parser. The UNSAT core and learned clauses are utilized to identify potential flip-flops where test points can be inserted. A subset of transition faults is sampled for test generation to reduce the computation overhead. A score is maintained for each flip-flop in the design that indicates the number of times it has been utilized for test generation. For a sample of the transition faults in the temporary RTL netlist, CNF instances for test generation are formed and sent to the SAT solver. If the instances are satisfiable, the score of flip-flops appearing in the learned clauses during conflict analysis is increased. If the SAT instances are unsatisfiable, the score of flip-flops appearing in the UNSAT core are increased. The scan flip-flop with the highest score after the sampling is identified for test point insertion. Note that weighted scores can be used to differentiate between UNSAT and learned clauses and the weights can be adjusted depending on the type of circuit, test generation ease etc. After each test point insertion, the clause used to represent constraints on the selected flip-flop needs to be modified in the CNF representation of the circuit.

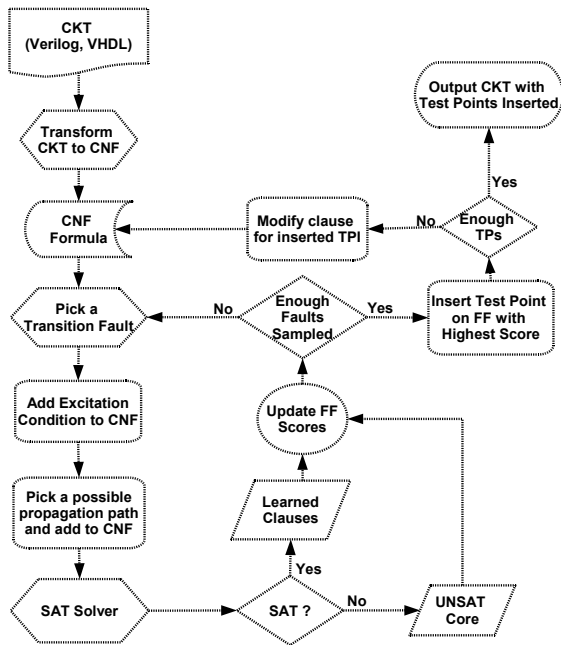


Figure 4. Test Point Insertion Algorithm

The test point insertion algorithm is shown in Fig. 4. The behavioral design (VHDL or Verilog) is read using a HDL parser and elaborated to a temporary RTL netlist that instantiates RTL elements. For LOC transition fault testing, the circuit is unrolled for two time-frames and transformed into a CNF formula  $F_c$ , in which each circuit element is rep-

resented by a variable. A transition fault at one of the lines of this RTL netlist,  $f$ , is chosen for test generation. The clause corresponding to its excitation condition is added to the CNF. A propagation path,  $p_f$ , for the fault effect is heuristically chosen and the constraints added to obtain a new CNF,  $F_{t,f}$  corresponding to test generation for  $f$ .

The heuristic used for fault sampling and propagation path selection is explained below. A two phase sampling is used for selecting the faults. Initially, faults are selected randomly. In the second phase, faults are selected on lines with lowest scores. A score  $S_g$  is maintained for each line (variable)  $g$ , that represents the number of times it has appeared in UNSAT cores and learned clauses. In this work, the score is defined as  $S_g = U_g \times w + L_g$  where  $U_g$  represents the number of times the line  $g$  occurs in UNSAT cores and  $L_g$  represents the corresponding number in learned clauses. The weight,  $w$ , depends on the circuit but usually a higher weight is given for the UNSAT core. The propagation path for a sampled fault is selected using a greedy heuristic. The next node in the path is selected as the node in the fanout of the current node that has minimum  $S_g$  and this continues till the path reaches a primary or pseudo-primary output.

Note that  $F_{t,f}$  actually represents a possible scenario to detect fault  $f$ . If  $F_{t,f}$  is satisfiable then  $f$  is testable. The opposite is not true because there may be other sensitizable propagation paths. The key idea is is not about checking whether  $F_{t,f}$  is satisfiable but to locate the appropriate parts of the circuit that hinder test generation. If  $F_{t,f}$  is unsatisfiable, an UNSAT core is extracted. On the other hand, if  $F_{t,f}$  is satisfiable, a set of conflict LEARNED clauses are obtained. Both the UNSAT core and conflict LEARNED clauses represent the difficulty of test pattern generation and the scores for each line are updated appropriately. After enough faults have been sampled, the flip-flop which has the highest score is selected as the candidate for test point insertion. The fault sampling ratio is a parameter that can be decided based on the circuit characteristics and the required execution time of the algorithm. Note that inserting a test point will change the testability of the circuit. If multiple test points are inserted, the test points need to be added iteratively. In each iteration, only one flip-flop is selected to add the test point. The CNF for the circuit  $F_c$  is updated with this test point for the new iteration.

The proposed test point insertion algorithm is explained using an example. Figure 5 shows the ISCAS'89 benchmark circuit  $s27$  unrolled into two time-frames for LOC transition testing. The benchmark circuit is assumed to be fully scanned and has 4 primary inputs (labelled 1 – 4 in Fig. 5), 3 flip-flops (labelled 5 – 7), 10 gates (labelled 8 – 17) and one output (18) for a total of 18 primitives. All the primitives are labeled with different numbers in the two time-frames for transforming the unrolled circuit into CNF. The CNF corresponding to the unrolled circuit has over 80

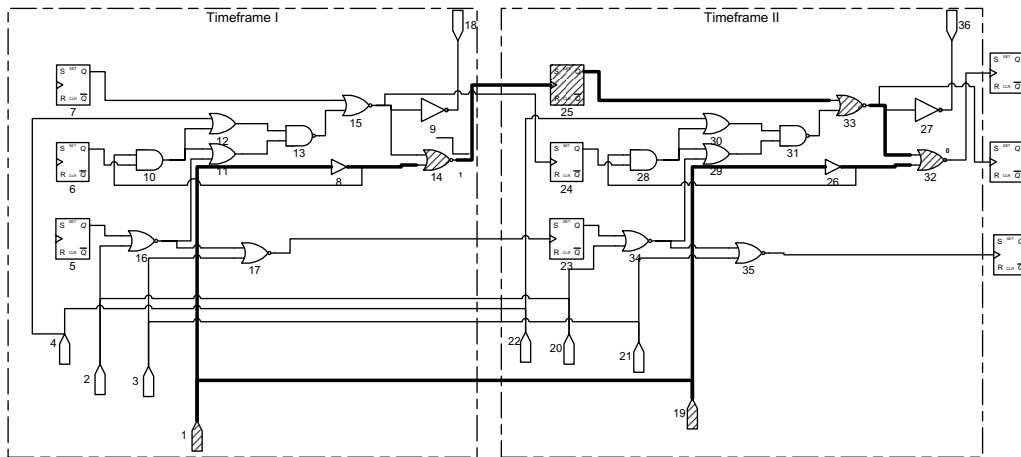


Figure 5. Test Point Insertion Algorithm on ISCAS'89 Benchmark s27

clauses, assume that it is denoted by  $F_c$ . If the transition fault currently selected is primitive 14 slow-to-fall, the excitation condition is primitive 14 equals 1 in first time-frame and primitive 32 equals 0 in the second time-frame. Because primitive 14 directly connects to a flip-flop, the propagation condition is empty. The excitation condition can be encoded as  $(14)(\overline{32})$ . When the SAT solver is called with  $F_c \cap ((14)(\overline{32}))$ , it returns that this formula is UNSAT. After analyzing the UNSAT core of this formula, the primitives responsible are identified. These primitives are shown shaded in Fig. 5, and the connections between them shown in bold lines. Since primitive 25 is in the UNSAT core, its score ( $S_{25}$ ) is increased by 2 (assuming  $w = 2$ ). The sampling of faults is carried on till a certain limit is reached. For the circuit in Fig. 5, after 50% faults are sampled, the scores of the three flip-flops are  $S_{23} = 16, S_{24} = 8, \text{ and } S_{25} = 12$ . For this circuit, a test point will be added at flip-flop corresponding to primitives 5 and 23.

#### 4 Experimental Results

Experiments were performed to verify the effectiveness of the proposed algorithm. The test point selection algorithm was implemented in C++ and integrated with the Boolean SAT solver ZCHAFF [9]. The behavior level design model is read using a HDL parser and elaborated into a temporary RTL netlist for test point insertion. Both the original and modified design (before and after test point insertion) are synthesized using Synopsys Design Compiler with the original delay (clock period) constraints. Transition fault test patterns are generated using a proprietary ATPG.

Experiments were performed on four ITC'99 benchmark circuits and four industrial SoC cores. In each circuit, 2% of the flip-flops are selected to insert test points. The results

are reported in Table 1 which compares the transition fault test generation and synthesis characteristics of the original circuit and the modified circuit (*i.e.*, after test point insertion). The first three columns under *Circuit* show the circuit name, number of test points inserted and the number of flip-flops in the circuit. The top row corresponds to the original circuit while the bottom row corresponds to the modified circuit. The next four columns, under *Test Generation* show the transition fault test generation results using a proprietary ATPG. Fault coverage (*FC*), fault efficiency (*FE*), number of test patterns (*#V*) and ATPG execution time (*ATPG RT*) are reported. For the original circuit, the maximum fault coverage that can be obtained are shown. For the modified circuit, fault coverage was limited to the fault coverage of the original circuit. The next two columns show the maximum delay of the circuit which corresponds to minimum clock period and the area reported by *report\_area* command in Design Compiler. The same timing constraint was used during synthesis of both the original and the modified circuit.

The last four columns in Table 1 show the effectiveness of the proposed scheme. The column *Vol. Red.* shows the percentage reduction in the transition test data volume while *ATPG Red.* reports the reduction in ATPG execution time to achieve the same fault coverage. Test data volume is calculated as the product of the number of test patterns and number of bits that need to be stored for each pattern. Hence, it takes into account the storage required for the extra test point flip-flops. The column *Area Over.* reports the area overhead of inserting the test points and the last column (*Run Time*) reports the execution time of the proposed scheme. As can be observed from Table 1, the test set size can be significantly reduced by inserting test points using the proposed scheme. Even though the number of scan elements in the circuit increases with the addition of the proposed test points, the test pattern generator is now

**Table 1. Results after RTL Test Point Insertion**

Circuit	Test Generation						Synthesis		Vol. Red.	ATPG Red.	Area Over.	Run Time (s)
	TP	#FF	FC(%)	FE(%)	#V	ATPG RT(s)	CLK (ns)	Area (units)				
b14	-	250	66.98	92.68	622	459.2	12.68	36157	<b>42.87%</b>	<b>83.91%</b>	<b>1.13%</b>	248.9
	5	256	67.04	89.67	347	73.9	12.64	36565				
b15	-	460	58.35	85.61	935	1899.1	9.9	56271	<b>56.84%</b>	<b>84.29%</b>	<b>1.40%</b>	182.9
	9	470	58.41	85.64	395	298.3	9.8	57513				
b20	-	501	69.96	93.51	891	947.0	12.77	76952	<b>27.05%</b>	<b>74.17%</b>	<b>1.01%</b>	245.1
	10	512	70.00	92.63	636	244.6	12.53	77728				
b21	-	497	70.64	93.88	907	984.8	12.83	76866	<b>41.40%</b>	<b>42.52%</b>	<b>1.65%</b>	484.2
	10	508	70.72	86.92	520	566.1	12.61	78137				
D1	-	337	85.48	98.94	1703	364.5	7.51	33548	<b>12.65%</b>	<b>19.09%</b>	<b>-0.04%</b>	157.8
	7	345	85.50	98.45	1453	294.9	7.84	33535				
D2	-	876	69.16	93.56	1166	906.9	9.81	60252	<b>39.45%</b>	<b>67.93%</b>	<b>2.28%</b>	422.9
	18	895	69.17	92.03	691	290.8	9.77	61623				
D3	-	1858	58.66	90.87	2347	3216.9	9.84	105174	<b>14.39%</b>	<b>48.51%</b>	<b>0.25%</b>	378.0
	38	1897	58.66	92.45	1968	1656.4	9.68	105440				
D4	-	3738	34.76	92.32	1688	5422.2	10.94	250049	<b>21.85%</b>	<b>50.17%</b>	<b>0.63%</b>	2720.5
	75	3863	34.76	92.67	1286	2702.1	10.94	251619				
Average									<b>32.06%</b>	<b>58.82%</b>	<b>1.03%</b>	

able to detect more faults per generated pattern and perform higher static compaction. The running time of the ATPG is also dramatically reduced because the improved testability makes test pattern generation easier. The area overhead for most circuits is around 1%. The execution time of the proposed algorithm for most circuits is in the order of few minutes. From Table 1 it is clear that the extra delay due to the multiplexers inserted for test points has been taken care of during logic synthesis. For some of the circuits, the critical path delay has even decreased after test points are inserted. This confirms the advantage of inserting test points at RTL. On average, by using the proposed RTL test point insertion technique, transition test data volume is reduced by over 32% while the ATPG execution time is reduced by over 58% with around 1% area overhead.

## 5 Conclusions

In this paper, a new RTL test point insertion methodology is proposed to reduce the data volume of scan-based broadside transition delay tests. A subset of scan cells is identified and the flip-flops in this subset are converted to enhanced-scan. These scan elements are replaced by two flip-flops so that the values required in both the patterns for a two pattern test can be stored, thereby removing the dependency of the second pattern in broadside transition tests. The proposed methodology utilizes recent advances in SAT that identify clauses responsible for unsatisfiability of SAT instances to locate parts of the circuit that hinder test generation for transition faults using the broadside (LOC) approach. Functional information of RTL primitives can be used in a SAT based method and hence the unavailability of structural information does not hinder the proposed test point identification algorithm. Using the proposed method-

ology, the number of specified bits required to test transition faults is reduced thereby improving test set compaction. By inserting test points in RTL, extra delay due to multiplexers are absorbed during logic synthesis. Experimental results show the proposed methodology can reduce transition test data volume by more than 30% without violating timing constraints while maintaining less than 1% area overhead.

## References

- [1] M. Abramovici, M. Breuer, and A. Friedman. *Digital Systems Testing and Testable Design*. IEEE Press, 1990.
- [2] S. Boubezari and et. al. Testability Analysis and Test-Point Insertion in RTL VHDL Specifications for Scan-Based BIST. *IEEE Transactions on CAD*, 18(9):1327–1340, Sept. 1999.
- [3] J. Carletta and C. Papachristou. Testability Analysis and Insertion for RTL Circuits Based on Pseudorandom BIST. In *Proc. IEEE International Conference on Computer Design*, pages 162–167, 1995.
- [4] T. Larrabee. Test Pattern Generation Using Boolean Satisfiability. *IEEE Transactions on CAD*, 11:4–15, 1992.
- [5] L. Lingappan and N. K. Jha. Unsatisfiability based Efficient Design for Testability Solution for Register-Transfer Level Circuits. In *Proc. VLSI Test Symposium*, pages 418–423, 2005.
- [6] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik. CHAFF: Engineering an Efficient SAT Solver. In *Proc. of Design Automation Conference*, pages 530–535, 2001.
- [7] S. Roy, G. Guner, and K.-T. Cheng. Efficient Test Mode Selection and Insertion for RTL-BIST. In *Proc. of International Test Conference*, pages 263–272, 2000.
- [8] J. Savir and S. Patil. Scan-Based Transition Test. *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, 12(8):1232–1241, 1993.
- [9] L. Zhang and S. Malik. Validating SAT Solvers Using an Independent Resolution Based Checker - Practical Implementations and Other Applications. In *Proc. of Design, Automation and Test in Europe Conference*, pages 10880–10885, 2003.