Using Scan-Dump Values to Improve Functional-Diagnosis Methodology

Vishnu C Vimjam[1] M Enamul Amyeen[2] Ruifeng Guo[3] Srikanth Venkataraman[2] Michael Hsiao[1] Kai Yang[4]

[1]ECE Dept, Virginia Tech, Blacksburg, VA – 24061 [2]Intel Corporation, Hillsboro, OR – 97124

[3]Mentor Graphics Corp., Wilsonville, OR – 97070 [4]Novas Software, San Jose, CA – 95110

Abstract

In this paper, we identify two main bottlenecks in the functional diagnosis flow and propose new ways to overcome these. Our approach completely eliminates the "Primary Input (PI) pattern generation and simulation" step and instead employs scan-dump values extracted from the tester. We utilize backward and forward logic implications of the scan-dump values to reconstruct more logic values for the circuit signals. Furthermore, we employ the reset state for the non-scan latches of the design to increase the number of specified signals in the overall circuit. Experimental results on stuck-at faults on industrial designs show that, in most cases, these reconstructed values are sufficient to correctly diagnose a fault, thereby avoiding hours of conventional functional diagnosis runtimes.

1 Introduction

There exist two kinds of techniques for testing an integrated circuit: (i) Functional tests and (ii) ATPG based tests. Functional test patterns are those that are created by designers for design validation purposes or converted from system level tests [1]. Due to the sequential nature and the large number of simulation cycles required to execute these patterns, performing functional analysis can be highly time-intensive, with fault isolation times ranging from weeks to months to identify a failing location. On the other hand, ATPG based tests are usually aimed at generating tests that target specific fault models [2]. Since sequential ATPG approaches are not very efficient for large and complex designs, scan-based design-for-test techniques have been widely adopted to improve testability and reduce test costs for VLSI designs. Nevertheless, functional test still plays an important role in screening defective parts, especially for high performance circuits where full scan design is not practical due to performance or power overhead of scan circuitry [1, 3, 4].

Utilization of scan-dump based techniques for functional failure debug/diagnosis has been well explored in the past [5,6,7,8]. A scan-dump is a snapshot of internal scan cell values at a specific functional test pattern cycle. As such, scan-dump data provides observability of internal states of scan circuitry. Recently, scan-dump data and logic cone analysis techniques were combined together and successfully used for functional failures debug in Ultra-SPARC processors [9,10].

In this paper, we present a new functional failure diagnosis technique using scan-dump data. In addition to simply using the scan as observatory points, we extract scan-dump for a sufficient number of cycles and use logic implications to reconstruct logic values in a circuit, thereby completely avoiding the two time-consuming setup steps needed in conventional diagnosis. We integrated our technique within a logic fault diagnosis tool [11] where the scan-dump data is provided as an input to the tool. Once the logic values of the circuit are reconstructed, fault diagnosis and candidate ranking can be performed as in the conventional way.

The paper is organized as follows: Section 2 reviews traditional functional diagnosis. Section 3 discusses in detail the limitations of the existing flow and then describes the proposed techniques. In section 4, we address issues in our technique and present experimental results in section 5. Section 6 concludes the paper.

2 Functional Pattern Diagnosis Flow

In this section, we briefly review the conventional logic fault diagnosis performed on functional failures. Like scan ATPG based diagnosis techniques, functional-pattern logic fault diagnosis techniques [11] analyze failures observed on scan cells through the scan-dump operation. Logic fault diagnosis is performed in the following steps: (i) Scan dump data from the failing unit is compared with the scan dump data from a good unit, and failing scan cells are identified, (ii) from each failing scan cell, structural backward traversal is performed through the logic circuitry to identify all logic signals that can potentially propagate erroneous values to the failing scan cells, (iii) candidate fault list is created based on the these candidate signals, (iv) diagnostic fault simulation for candidate faults is performed and (v) candidate faults are ranked based on how closely they match the failures observed on the tester. Since functional patterns are usually very long, diagnostic fault simulation has a high time-complexity. Optimization techniques [11] have been proposed to perform logic simulation at the beginning and switch to fault simulation only when it is necessary to excite a logic fault. This significantly reduces the overall diagnosis time. A detailed outline of the overall procedure is shown in Figure 1.

There are several types of diagnostic metrics that can be used to quantify the matching between fault signatures and tester failures. The metric *intersection* is used to describe a match between a fault's behavior and the actual defect's behavior [11,12,13]. On the other hand, a fault can cause simulation failures on some patterns even though there are no tester failures on those patterns. Such additional simulation failures are termed as *mispredictions* [11,12,13] and are caused by non-excitation of the silicon defect on those patterns. Fault sorting is usually done according to their intersection count where the ties can be resolved using the misprediction count. Furthermore, faults having identical behaviors, i.e., having same intersections and mispredictions, can be grouped into candidate fault classes. In the sorted fault-lists, class 0 represents the best set of candidate faults, class 1 represents the next best set and so on.

```
Given a functional pattern P; Reset state R
C, C'=starting and ending failure cycles
Initialize T=C-1
While (Improving diagnostic resolution) {
  LogicSim(R: 1, 2, …, T)
  InjectFaults(T+1, T+2, …, C, C+1, …, C')
  FaultSim(T+1, T+2, …, C, C+1, …, C')
  SortFaultsIntoCandidateClasses();
   T = T – 1
}
```

Figure 1: Functional diagnosis procedure

0-7695-2812-0/07 $20.00 © 2007 IEEE

IEEE COMPUTER SOCIETY

3 Applying Scan-dump in Functional Diagnosis

In this section, we first present the main motivation behind our work followed by our techniques for using scan-dump values in improving the functional diagnosis methodology.

There are two main bottlenecks in the conventional functional diagnosis procedure: First, functional patterns are generated by the designers at the full-chip level for validating the design. Depending on the size of the design, each such pattern can be very long (ranging from hundreds of thousands to millions of cycles) in order to exercise various functional states of the design. On the other hand, since full-chip defect analysis is often impractical, diagnosis is limited to individual functional blocks. Consider the empirical figure shown in Figure 2 where a failure is observed in block FB3. In order to diagnose it, first there is a necessity to generate the functional-patterns at the inputs of FB3 (usually called *PI pattern generation*). Generating such a pattern from the full-chip pattern requires simulation at the RTL or gate level and the runtime depends on the size and location of that block. Note that pre-computation of PI patterns for all the blocks is highly infeasible since a design can have thousands of functional patterns and we do not know beforehand which pattern will fail on which block. Current experiences show that, for a given full-chip functional pattern, PI pattern generation itself can range from 8 to 12 hours and is expected to grow for future designs.

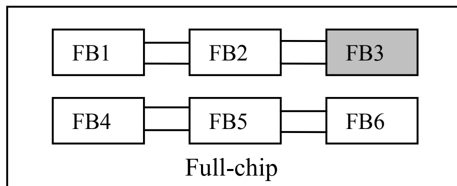


Figure 2: Full-chip view

Second, consider the existing functional diagnosis flow as shown in Figure 3. Let C and C' represent the first and last failure cycles respectively as observed on the tester. These failures are due to the excitation of a defect and its propagation to some observation point(s) on the tester. In the conventional run, logic simulation is performed until the cycle T, after which faults are injected onto the circuit and fault simulated. The diagnosis engine then analyzes the fault simulation results to check if a conclusion can be arrived-at regarding the faulty location. At a given T, note that no diagnosis result can be obtained if none of the faults are detected during fault-simulation or if there is only little confidence that the defect was diagnosed (for example, if the number of intersections is very small or if the misprediction count is very high for the top candidate). In such cases, the number of fault simulation cycles (C-T) is further increased to perform fault simulation for longer times. We denote *fsim-depth* = C-T.

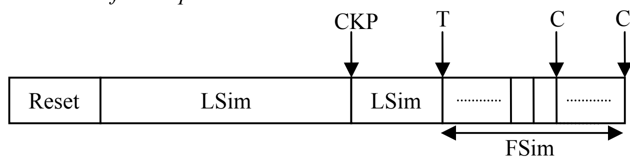


Figure 3: Conventional Diagnosis

Clearly the main challenge in the conventional run is the logic simulation done from the reset cycle up to the cycle T. Since repeating this (within the *while* loop of Figure 1) is prohibitive, in practice, logic simulation is performed until a cycle CKP close to T (refer to Figure 3) and the circuit state at that check-point is stored and used every time to continue the logic simulation from that point. Current experiences show that this one-time logic simulation from R to CKP can range from 6 to 8 hours depending on the size and complexity of the target block.

Thus, the main goal behind our work is to avoid these two high setup times that were needed in order to perform diagnosis for functional failures. We propose techniques using scan-dump values and reset state of the design to avoid these setup times in their entirety thereby speeding up the diagnosis procedure manifold. The following subsections detail our work.

3.1 Extracting scan-dump values

Observation points in a design include its primary outputs and scan-cells. Since we perform diagnosis at the functional block level, all the outputs for a given block might not be observed directly at the tester and hence we consider only the scan-cells of the block as its observation points. The scan-dump extraction process for good and faulty values are described next.

Failures from functional patterns are first observed at the primary outputs of the chip. In order to obtain the first scan mismatch cycle that occurred before the first bus failure, the following steps are performed: The scan signature is compacted by feeding scan cell values into a linear feedback shift register (LFSR) to minimize the amount of data to be compared. A snapshot of the compressed scan signature is then obtained several hundred cycles before the first bus cycle failure. The compressed scan signature is then compared with the corresponding scan signature from a good unit. If the two signatures match, i.e., the signature obtained from the failing unit passes, then most likely the first scan mismatch has occurred somewhere between the current cycle and the bus failing cycle. On the other hand, if the two signatures mismatch, then the first scan mismatch would likely have occurred before the current scan comparison cycle. We take additional snapshots of compressed signatures and compare signatures in a binary search fashion to arrive at the first scan mismatch cycle. Once the first scan mismatch cycle is identified, the process is continued to identify more mismatch cycles (if any). Our preliminary experiments showed that, typically 10-20 observed failures after the first failure is sufficient to correctly diagnose a defect in the conventional functional diagnosis flow. Note that this might not be the case for ATPG-based (partial-scan) patterns.

We stop the failure identification process once we reach a pre-defined goal on the number of failures or reach a cycle sufficiently larger than the first failing cycle. We then utilize a good unit and obtain uncompressed scan dumps for several consecutive cycles before the first scan mismatch cycle and up to the last failing cycle (In a two-phase design, scan-dump values are extracted at the HIGH phase of the clock in a cycle). Since we do not exactly know how many fault simulation cycles are needed to obtain the final diagnosis result, we extract scan-dump values for sufficiently long number of cycles before the first failing cycle (say, between 50 to 200 depending on the size and complexity of the functional block). These golden scan-dumps from a good unit are used for deriving good simulation values while the scan mismatches from the failing unit are used for diagnosis.

3.2 Reconstructing values in the circuit

Since our aim is to reconstruct circuit signal values without logic simulation, we try to employ as much auxiliary information available for the design as possible. In this regard, we employ

IEEE COMPUTER SOCIETY

the following: (i) VSS and VDD values are set in all phases and all cycles, (ii) the grid clock CLK value is set to logic 0 in LOW phase and logic 1 in HIGH phase in all cycles, (iii) Any other signals that are known to have constant values during the functional mode of operation. These include signals such as the scan-clock signal which is always reset (unless observing scan-out values through the scan-chain on the tester), or any other domain-specific input signal that is guaranteed to take a certain value for that functional pattern, (iv) the scan-dump values that are extracted from the tester and (v) the reset state of that functional block which can be derived from the overall reset state of the complete design.

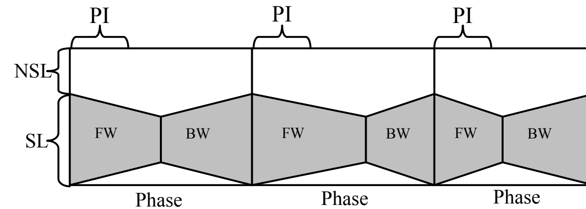
```
Given reset state R
C, C'=starting and ending failure cycles
Initialize T=C-1
While (Improving diagnostic resolution) {
  ApplyAuxInfo(T+1, T+2, …, C, C+1, …, C')
  ApplyScanDump(T+1, T+2, …, C, C+1, …, C')
  ApplyResetState(T) & ResolveConflicts();
  InjectFaults(T+1, T+2, …, C, C+1, …, C')
  FaultSim(T+1, T+2, …, C, C+1, …, C')
  SortFaultsIntoCandidateClasses();
  T = T – 1
}
```

Figure 4: Overall algorithm

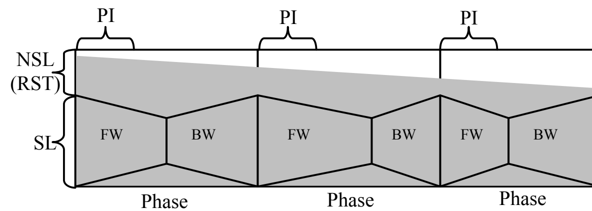
Our overall algorithm is shown in Figure 4. Initially, all the auxiliary values known from (i), (ii) and (iii) above are applied onto the circuit and forward simulated to see if they can imply any values further. The scan-dump values that are extracted from the tester are then applied in the HIGH phase of each cycle. For the sake of discussion, let Φ1 represent the HIGH phase and Φ2 the LOW phase. In order to maximize the number of specified signals, we use the following rules: If a scan-latch L1 is a Φ1-latch (i.e., it is transparent in Φ1), it will be holding in Φ2 and hence is guaranteed to have the same value as that of Φ1 in Φ2. Hence, the scan-dump value in Φ1 can be replicated in Φ2 within the same cycle. Similarly, if another scan-latch L2 is a Φ2-latch, it will be holding in Φ1 of the next cycle. In other words, the scan-dump value for L2 in Φ1 of cycle C can be replicated in Φ2 of cycle C-1. Note that we cannot replicate values in the other two cases since it doesn't guarantee correctness.

After all the scan-latches are given values in all the phases in all the cycles between T+1 and C', we use logic implications to propagate these further around the circuit. The main intention here is to specify more signals in the circuit by just using the scan-dump values. An empirical propagation procedure is shown in Figure 5(a) where each unrolled frame represents one phase of the clock-cycle.

With all the existing values in a given phase, first backward implications (BW) are performed to propagate logic values backwards. Whereas the backward implication procedure is straightforward for combinational logic gates, care should be taken when processing latches. We use the two basic rules for deriving values around latches: (i) If the signal driving the CLK-port of the latch has transparent value for that latch, then the value at Q-port can be propagated to the D-port of that latch and (ii) If the Q-port of a latch has differing logic values in two consecutive phases, then that latch is guaranteed to have its transparent value on its CLK-port in the later phase. Note that case (ii) is the only scenario where we can imply logic values on the internal clock signals (apart from the already known CLK grid value). However, once an internal clock value is known, it can imply more values in its driving circuitry or can help in implying logic values on D-ports of other latches.



(a) Scan-dump propagation



(b) Scan-dump + Reset propagation

Figure 5: Propagation of logic values

Once the backward implications are saturated, all the newly implied values are forward simulated (FW) together with the existing values to learn more signal values in the circuit. Note that the main efficiency of these implication procedures depend on the scan-dump values applied in those phases. For example, in the first phase of Figure 5(a), both forward and backward implications were equally useful whereas in the second phase forward propagation specified more signals while in the third phase backward propagation was more productive. This process is performed within each phase and values are transferred across phases as more internal clock values become known. The entire process is repeated until no more signals can be specified using backward or forward implications in the circuit. The main flow behind using the scan-dump values is shown in Figure 6(a), where the logic simulation part until cycle T is skipped, while the scan-dump values are processed between T and C' by using an all unknown logic state (all-X) at T.

3.3 Applying Reset state for Non-scan elements

In the above section, we have seen how scan-dump values are used to reconstruct circuit signal values within a range of cycles. However, the main limitation here is that the parts of the circuit that cannot be reached from scan cells remain unspecified. In this regard, we explore the use of the reset state for the non-scan elements for a given functional block which can be derived from the full-chip reset state. Note that the reset state is independent of the specific functional pattern being applied.

There are two main motivations behind using the reset state: (i) the non-scan latches together with the scan-latches form a sufficiently large cut in the circuit. Consider the example shown in Figure 5(b) where the reset state is applied to the non-scan latches in the first phase. Using the reset state values for the non-scan latches and the scan-dump values for the scan-latches would thus specify this large cut in the circuit which when propagated forward can lead to a huge set of signals being specified (shown by the grey region). Note however that since we do not have PI values, some portion of the circuit still remains unspecified. This unspecified region clearly increases as we move forward from phase to phase. We have noticed however that the percentage of increase is usually very small

(~2-7%) since the number of PIs is much smaller compared to the total number of latches. (ii) The second reason behind using the reset state is the observation that the rate at which the state bits change is very small when functional patterns are applied. In other words, the hamming distance between two states appearing at two distant cycles is small. Hence the reset state represents an approximation of the *actual* state appearing at the cycle T during functional simulation. The main flow behind using the reset state is shown in Figure 6(b), where the logic simulation till cycle T is skipped, while the scan-dump values are processed between T and C' by setting the reset-state at T.

3.3.1 Resolving the conflicts

Let the reset state be denoted by R and the actual state at the cycle T be denoted by S. Since R is only used as an approximation of S, there can be differences in the logic values of the non-scan latches in R and S. Since the state S and scan-dump values are arising from the same functional pattern, they will be consistent in the circuit. On the other hand, when R is applied at cycle T and forward simulated, it can lead to inconsistencies among the already existing values (those resulting from implications of scan-dump). For example, consider the 3-input AND gate shown in Figure 7, where values at fanin C=1 and output D=0 are already specified due to scan-dump propagation from scan-latches L3 and L4. When the reset values are applied to non-scan latches, let L1 and L2 imply A=1 and B=1 respectively.

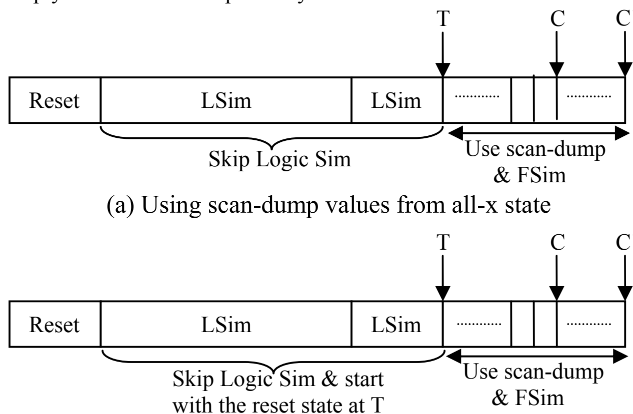


Figure 6: Proposed Diagnosis Runs

This causes a conflict at D during forward simulation. Whenever such a conflict is seen, we use critical-path tracing [2,14] to trace the error back to the non-scan latches causing it (L1 and L2 in this case). Note that any conflict causing path will arise at a non-scan latch (due to the reset state value applied) and can pass through several other latches and finally conflict at a combinational gate or a scan-latch. As a result, we have observed that such paths can be very long and many times extend for several cycles. Hence care should be taken in the implementation to correctly identify the conflict causing non-scan latch(es).

Once we find all the conflict-causing non-scan latches, we simply assign them to unknown (X) values and re-compute the forward simulation to clear the inconsistencies. Instead of doing this, one can also try to assign them to the opposite logic value (to that of the conflict-causing one from the reset state). However, since there can be more than one non-scan latches causing conflicts, it does not guarantee the removal of all inconsistencies. Another way of resolving this is by doing a branch-and-bound search on the conflict causing non-scan latches to get to a state that is consistent with the scan-dump values. However, we have observed that usually a very small percentage of non-scan latches are conflict causing and hence leaving them with X values did not significantly affect the overall specified region.

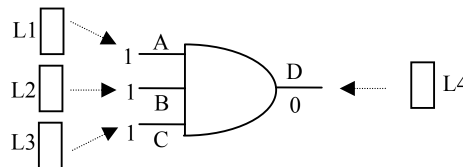


Figure 7: Example conflict while using Reset

4 Complexity Issues

For each functional block, there will be input control signals that disable certain clock circuitries depending on the functional pattern applied. Due to clock gating of scan clocks from these primary input control signals, some of the scan clock values remain unchanged in some cycles irrespective of the clock-phase. As a result, we cannot assume free-flowing clocks for all scan-latches. A large fraction of un-initialized signal values are due to unknown clock values and since our technique completely avoids the use of input patterns, the only way that we can specify values on such input control signals is via backward implications as explained in the previous section. On the other hand, if it is guaranteed that there will be no clock-gating for a given functional pattern, then we can assume free-flowing internal clock values which will clearly increase the efficiency of our approach.

The complexity of the scan-dump implication procedure is O(T.N.I) where T is the number of time-frames unrolled, N is the number of circuit gates and I is the number of iterations needed to reach the saturation point (typically ranges from 3-10). When the reset state is applied, the additional work done is of the order O(T.N) since just one forward simulation is sufficient to propagate all the values.

5 Experimental Results

We have written a prototype tool using the proposed techniques which is integrated within the main diagnosis tool Poirot [11]. Functional diagnosis experiments for large functional blocks from the Pentium 4 microprocessor design are conducted on a 2.8GHz Xeon machine with 2GB memory and running Linux OS. All the functional patterns applied had clock-gating signals and hence we could not use free-flowing clocks.

The experimental setup is as follows: For each functional block, we ran non-dropping *stuck-at* fault simulation using a functional pattern starting from the reset state R. Among all the faults that were detected, let F be the set of stuck-at faults that were detected after 100000 vectors from the reset state. Among these faults, let F1 represent a set of sampled faults whose first detection was as late as possible. Since there are some easy-to-detect faults in F1 that are be detected at these high vectors, our aim is to check if our technique can isolate the faults in F1 from all those in F. To do this, we inject a fault *f* in F1 onto the circuit, perform fault simulation and capture the fault-effects. These were in-turn given as failures to the diagnosis engine with the target fault-set F to see if *f* can be correctly diagnosed.

Table 1 shows the statistics of the four functional blocks that we tested our techniques on, where the columns *PIs*, *Gates*, *Tot Lat, ScanLat%* show the number of primary inputs, total gates and

IEEE COMPUTER SOCIETY

total latches and scan-latch percentages respectively in each of these designs. Table 2 shows the actual diagnosis results, where the column *Flts Taken* shows the number of faults taken in F1 for each block. The columns *Cls 0*, *Cls 1* and *Cls 2* report the number of faults that were diagnosed into Class 0, 1 and 2 respectively. SD denotes our Scan-Dump technique, while SD+R is used for SD with reset state.

Table 1: Functional block statistics

Block	PIs	Gates	Tot Lat	%ScanLat
D1	491	89.8K	8K	70.6%
D2	493	220K	16.2K	87.6%
D3	708	549K	36.7K	73.0%
D4	1014	1.01M	77.7K	34.0%

The column *NoResult* reports the number of faults that could not be diagnosed into any class (i.e., those faults could not be detected using that technique). The number shown in pareentheses under *NoResult* column gives the number of those faults that are either in the clock-tree or memory-related faults. The following key observations can be made from the table: (i) the conventional run distinguishes the real faults among the target set and reports them in the top class; (ii) The two new runs SD and SD+R are limited in their capability to diagnose clock-faults and memory-related faults; (iii) if considering only logic faults, those two runs were able to achieve almost 100% perfect diagnosis and more than 90% of those faults are reported in class 0; (iv) the run SD+R either performs similar or better than the SD run in classifying more faults into the top classes. Note that the values derived on the clock/memory segments depend mainly on the *specific* scan-dump values applied and so does the efficiency of our techniques in diagnosing those faults. On the other hand, most of the logic driven by the latches will be specified due to the scan-dump values and hence logic faults could be diagnosed efficiently.

Table 2: Functional diagnosis results

Block	Type	#Flts Taken	Cls 0	Cls 1	Cls 2	No Result
D1	Conv	50	50	-	-	0
	SD	50	43	3	-	4 (4)
	SD+R	50	45	1	-	4 (4)
D2	Conv	50	50	-	-	0
	SD	50	41	2	4	3 (3)
	SD+R	50	42	1	4	3 (3)
D3	Conv	50	50	-	-	0
	SD	50	33	-	-	17 (17)
	SD+R	50	35	-	-	15 (15)
D4	Conv	40	40	-	-	0
	SD	40	40	-	-	0
	SD+R	40	40	-	-	0

Table 3 shows the percentages (averaged per clock-phase) of the signals specified during the scan-dump runs, where the columns *Lat*, *ScanLat*, *NonScanLat*, *PIs*, *TotGates* show respectively the percentage of latches, scan latches, non-scan latches, primary inputs and total gates. As seen, when only the scan-dump values were used, the total percentage of gates specified ranged from 47% to 77%, whereas these increased to 66% to 85% when the reset values are also used. (Note that the % of scan-latches in these runs are not 100% since there can be more than one latch inside a scan-cell and we obtain only the outer most scan-latch value in the scan-dump). Clearly, due to the use of the reset state, the amount of signals specified is increased significantly which explains why more faults are diagnosed into top classes in Table 2.

Table 3: Specified signal statistics

Block	Type	Lat %	Scan Lat %	Non Scan Lat %	PIs %	Tot Gates %
D1	SD	60%	84%	3%	1%	64%
	SD+R	93%	90%	99%	1%	85%
D2	SD	87%	96%	22%	6%	77%
	SD+R	94%	96%	85%	6%	82%
D3	SD	42%	87%	10%	10%	50%
	SD+R	82%	96%	70%	10%	69%
D4	SD	32%	92%	3%	14%	47%
	SD+R	64%	97%	48%	14%	66%

Table 4 shows the analysis according to *fsim-depth,* which denotes the number of cycles between the first fault simulation cycle and the first failure observation cycle. The column *Flts With Result* is the sum of columns *Cls 0*, *Cls 1* and *Cls 2* from Table 2. The columns *FSD 1*, *FSD 5* and *FSD 20* show the corresponding number of faults that were correctly diagnosed at those fsim-depths. The two main observations to be made from this table are: (i) In the conventional functional diagnosis runs, a small fsim-depth (~20) is sufficient to correctly diagnose each fault. This in turn implies that the fault-effects in functional tests are only short-lived; (ii) the scan-dump based runs usually require a higher fsim-depth than the conventional run. As mentioned earlier, the number of signals specified increase as more scan-dump values are employed and as such some faults might only be detected at higher fsim-depths.

Finally, we show the resource usage for the 3 blocks D1, D2 and D3 with respect to the fsim-depth in Table 5, where the columns *LogicSim*, *Impl* and *Total* show the runtimes for logic simulation (in conventional run), for implication processing (in scan-dump runs) and total diagnosis runtimes respectively. The column Memory shows the memory consumption in mega bytes. The column *Conflicts* shows the number of conflicts that were seen while using the reset state, as a result of resolving which the *Impl* time under SD+R is higher than that of SD alone.

Table 4: Analysis with respect to fsim-depth

Block	Type	#Flts With Result	FSD 1	FSD 5	FSD 20
D1	Conv	50	48	2	-
	SD	46	37	9	-
	SD+R	46	21	25	-
D2	Conv	50	45	5	-
	SD	47	47	-	-
	SD+R	47	47	-	-
D3	Conv	50	26	8	16
	SD	33	2	28	3
	SD+R	35	1	21	13
D4	Conv	11	11	-	-
	SD	11	11	-	-
	SD+R	11	11	-	-

Clearly, it can be seen that the logic simulation runtime in conventional run is the dominating factor and the scan-dump based runs can outperform this by 3-9 times. Note that these logic simulation times represent logic simulation performed for just 100000 vectors (as used in our experimental setup) and can exceedingly increase as longer patterns are used or the design is

IEEE COMPUTER SOCIETY

larger and complex. Also, by just using the scan-dump values and the reset state, our framework avoids the PI pattern generation which consumes significantly more runtime. When the memory consumption is compared, clearly the scan-dump based runs require more memory due to the storage of circuit state values while processing implications. However, the increase over the conventional run is only nominal and stayed within the current days' systems limits.

6 Conclusions

In this paper, we have presented a new methodology based on scan-dump values to avoid excessive functional-diagnosis run-times. Our technique uses backward/forward implications and reset state of the design to effectively achieve close-diagnosis coverage to that of the conventional run. Experimental results on large industrial circuits showed that almost all the stuck-at faults on the logic regions can be diagnosed and more than 90% of these are perfectly diagnosed into class 0. For the faults that were diagnosed into lower classes, a range of top faults can be selected for analysis by the failure-analysis team. For faults within regions such as clock-trees and memory structures, although the proposed technique poses a limitation, they can be addressed using special techniques for clock-fault detections or using array-dumps. For future work, we want to explore advanced techniques such as recursive learning [15] to enhance our implication processing.

References

[1] Parvathala, P.; Maneparambil, K.; Lindsay, W., "FRITS - a microprocessor functional BIST method", *International Test Conference*, 2002, pp.590 – 598.

[2] M. Abramovici, M. A. Breuer, and A. D. Friedman, "Digital System Testing and Testable Design", AT&T Bell Laboratories and W. H. Freeman and Company, 1990.

[3] P. Maxwell, I. Hartanto, L. Bentz, "Comparing Functional and Structural Test", *International Test Conference,* 2000, pp. 400-407.

[4] P. Nigh, W. Needham, K Butler, P. Maxwell and R. Aitken, "An Experimental Study Comparing the Relative Effectiveness of Functional, Scan, IDDQ and Delay-fault Testing", *VLSI Test Symposium*, 1997, pp. 459-464.

[5] Levitt, M.E., "Designing UltraSparc for testability", *Design & Test of Computers, IEEE* On page(s): 10-17, Volume: 14, Issue: 1, Jan-Mar 1997.

[6] Goel, S.K., and Vermeulen, B, "Hierarchical Data Invalidation analysis for Scan-Based Debug on Multiple-Clock system Chips", *International Test Conference,* 2002, pp. 1103-1110.

[7] D. Josephon, S. Poehlman, and V. Govan,, "Debug Methodology for the McKinley Processor", *International Test Conference*, 2001, pp. 451-460.

[8] Anjali Kinra, "Towards Reducing "Functional Only" Fails for the UltraSPARCTM Microprocessors," *International Test Conference*, 1999, p.147,

[9] Dahlgren, P.; Dickinson, P.; Parulkar, I., "Latch divergency in microprocessor failure analysis", *International Test Conference*, 2003, pp.755 – 763.

[10] O. Caty, P. Dahlgren, I. Bayraktaroglu, "Microprocessor silicon debug based on failure propagation tracing", *International Test Conference*, 2005.

[11] S. Venkataraman, S. Drummonds, "Poirot: Applications of a Logic Fault Diagnosis Tool", *IEEE Design & Test of Computers*, Jan.-Feb. 2001 pp. 19-31.

[12] B. Chess, D. B. Lavo, F. J. Ferguson, and T. Larrabee, "Diagnosis of Realistic Bridging Faults with Single Stuck-at Information*,"* *IEEE/ACM International Conference on CAD*, pp. 185-192, Nov. 95.

[13] D. B. Lavo, T. Larrabee, and B. Chess, "Beyond Byzantine Generals: Unexpected Behavior and Bridging Faults Diagnosis," *IEEE International Test Conference*, pp. 611-619, Oct. 96.

[14] M. Abramovici, P. R. Menon and D. T. Miller, "Critical Path Tracing: An Alternative to Fault Simulation", in IEEE Design & Test of Computers, vol. 1, no. 1, Feb 1984.

[15] W.Kunz and D.K.Pradhan, "Recursive Learning: a new implication technique for efficient solutions to CAD problems-test, verification and optimization", *IEEE Trans. on CAD*, Sept 1994, pp. 1149-1158.

Table 5: Resource usage for each functional block

Block	fsim-depth	Conventional (Conv)			Using Scan-dump (SD)			Using Scan-dump + Reset (SD+R)			
		Logic Sim (s)	Total (s)	Memory (MB)	Impl (s)	Total (s)	Memory (MB)	#Conflicts	Impl (s)	Total (s)	Memory (MB)
D1	1	200	211	69.1	2	**7**	82.3	2	2	**7**	82.4
	5	200	214	69.1	3	**12**	84.7	4	3	**12**	84.6
	20	200	222	69.3	7	30	94.3	2	7	**27**	94.4
D2	1	200	237	92.4	7	**32**	128	1	9	35	128
	5	200	254	93.2	9	**49**	135	3	12	54	136
	20	200	324	94.0	15	**112**	154	5	21	126	155
D3	1	900	988	271	10	92	339	38	23	**89**	336
	5	900	1037	272	15	148	352	46	35	**146**	351
	20	900	1243	274	28	**349**	402	58	62	399	403