

LATCH DIVERGENCY IN MICROPROCESSOR FAILURE ANALYSIS

Peter Dahlgren, Paul Dickinson and Ishwar Parulkar

Processor and Network Products

Sun Microsystems, Inc.

430 N. Mary Avenue, Sunnyvale, CA 94085

{peter.dahlgren, paul.dickinson, ishwar.parulkar}@sun.com

Abstract

This paper presents an approach for analysis of system state differences observable through the scan chain for the debug of functional failures. A novel methodology for Latch Divergence Analysis (LDA) is proposed for creating stable failure signatures and reducing system noise. The methodology and processing flow have been integrated into the normal debug flow for the UltraSPARC™ family processors and have been successfully applied in numerous debugs in the bring-up of new products.

1 Introduction

Scan dump based debug has become an increasingly important debug technique for microprocessor bring-ups and electrical characterization as the complexity of processors have increased [6][7][2][9]. A scan dump is a snapshot of the state of the internal scan chain at a particular cycle while executing a functional pattern or running an operating system. Scan dumps can be obtained both at the chip level using ATE and at the system level using dedicated test access buses. Processor debug typically consists of functional verification and electrical characterization [9].

Although structural test methods such as ATPG and BIST are preferable for detecting manufacturing defects, functional tests play an important role in microprocessor testing, in particular for at-speed testing of random logic [1]. Functional tests also play an important role in design verification during first silicon debug for finding potential improper timing marginalities in the design or manufacturing process that limits the operational frequency and voltage range. The test results of UltraSPARC™ processors have shown that the proportion of “functional only” failures is significant [1]. The UltraSPARC™ processor families have a large variety of built-in test and debug features including, IEEE 1149.1 TAP architecture, full scan, clock manipulation, mbist, shadow scan and dedicated observability and access buses for test [3][4][5].

In this paper, we present a failure analysis approach based on analyzing the system state represented by bit strings shifted out from the internal scan chains while executing a functional pattern under different operational conditions. By observing the differences in system state over several adjacent internal cycles it is possible to identify the state elements at which the failure first manifests and at what cycle within the pattern the failure occurs. This information is then used in combination with logic cone analysis for finding candidate signal paths. Figure 1 shows the activities typically involved in scan dump based debug. An important part of the scan dump processing flow is the *Latch Divergence Analysis (LDA)*, shown in step 2 of Figure 1. LDA is the process of analyzing the differences in system state under different operational conditions and infer first failing state elements and cycle of manifestation. This paper focuses on a methodology for creating stable failure signatures and reducing the noise in the LDA process based on multiple scan dump series and a filtering scheme.

The LDA methodology is suitable for debugging electrical design errors or physical implementation errors as well as functional-only manufacturing and field failures.

The big challenge in Failure Analysis (FA) based on functional patterns as compared to ATPG-based FA is the lack of controllability of state elements and the instability in internal electrical failure signature or noise [1]. In ATPG, prior to defect activation, all state elements are deterministically assigned known values by shifting in a known pattern. On the other hand, for functional patterns, all state elements are not deterministic because of several reasons such as uninitialized memories or dependency on external conditions as further described in Section 3.2. This noise, referred to as *Latch Divergence Noise (LDN)*, manifests as unwanted system state deviations that are irrelevant for the failure under investigation. LDN needs to be reduced before the erroneous system state can be efficiently isolated. A method to filter out this noise and to create a stable electrical failure signature is described in Section 3.

The proposed methodology has been integrated into an automated scan dump processing flow, which has been

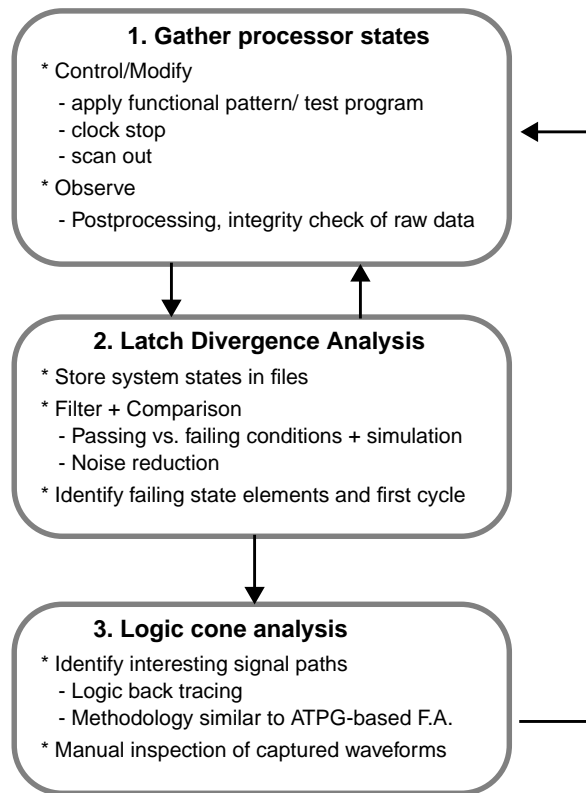


Figure 1 Typical activities in scan dump based debug.

applied to all current UltraSPARC™ processors during first silicon debug, characterization and diagnostics.

This paper is organized as follows. Section 2 describes the global scan dump processing flow and Section 3 concentrates on the proposed latch divergence noise reduction scheme. Section 4 presents the results from some real debug cases and, finally, Section 5 concludes the paper.

2 Processing flow for scan dump based debug

A scan dump is a non-intrusive image of the state of the internal scan chain of a functional pattern at a particular cycle. Scan dumps typically provide the highest possible observability of internal states of all debug techniques. With almost 100% of the internal state elements scannable, the UltraSPARC™ processor families are very suitable for scan dump based debug.

2.1 Design features to support tester and system scan dumps

To support scan dump based debug capability several dedicated design features are required.

- i) Deterministic clock stop at a predefined cycle number
There are several built-in features in the UltraSPARC™ processors that can manipulate the internal clock including

clock stop and clock stretch [3][6]. The internal CPU clock is typically generated by a PLL and runs at a frequency of a multiple of the external system clock frequency. In order not to cause any corruption the clock has to be stopped in a controlled manner without any glitches and without changing the pulse width for the last clock pulse. Thus, certain synchronization and control logic is needed in the PLL.

The clock can be stopped by asserting a control pin; by loading a dedicated instruction into the TAP controller; or by internal events controlled by software. The first method is used at the ATE whereas the second and third methods are usually applied in system debug as the clock stop control pins may not be easily accessible from a system platform. These methods select a certain system cycle to stop at. Next, the internal cycle within a system cycle is selected by having the cycle number loaded through the TAP controller into a dedicated register in the clock controller prior to initiating the clock stop sequence.

- ii) Shift out the scan chain

Shifting out the contents of the scan chain is accomplished by regular TAP full scan instructions already available for ATPG, but without the capture sequence since the state of any state element must not be overwritten between the time of clock stop and scan out. The various modes available for ATPG such as single chain, multiple chain or single/dual core modes, [5], are all available for scan dumps as well to reduce the amount of data to collect or in case of partially working devices.

2.2 Principles of multiple cycle dumps

Frequently, a failure causes an error internally in state elements many cycles before the tester observes any pin failure. Thus, in order to find the failing cycle, a backward cycle search from the failing cycle in the tester data log needs to be conducted. Instead of dumping, processing and analyzing cycles individually, an efficient approach is to dump a sufficiently wide range of adjacent cycles prior to the pin failure and analyze the raw data as waveforms of all state elements in the search for the failure.

The steps involved in taking scan dumps of functional patterns in this manner are as follows:

- (1) Reset and run the functional pattern up to CPU cycle N ,
- (2) Execute the clock stop control sequence and switch to test mode.
- (3) Scan out the contents of the scan chain.
- (4) Reset and restart the pattern, run up to CPU cycle $N + 1$.
- (5) Repeat steps 2-4 for the number of cycles desired.

The data collected gives the complete set of system state transition over time for the cycle range considered. Capa-

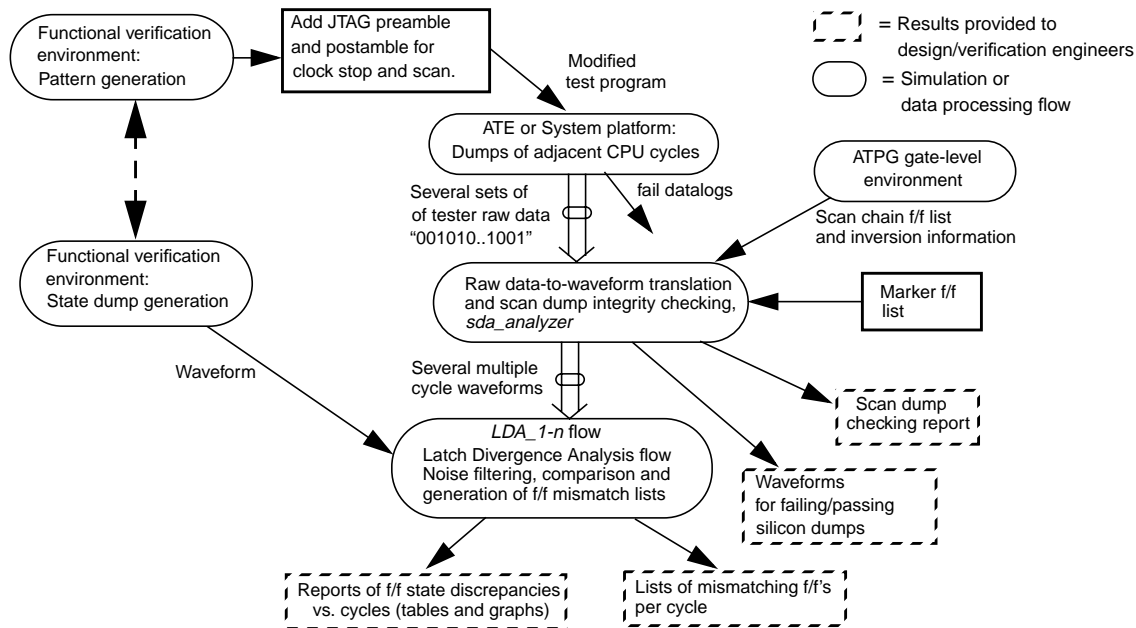


Figure 2 Global scan dump flow.

bility of resuming normal operation of the functional pattern after clock stop and scan out operations was never intended as this would require significant design overhead and affect the overall performance of the chip so the functional pattern needs to be restarted for each cycle dumped. There is usually no power down between the individual dumps and the scan chain is loaded with a known state prior to executing the regular functional reset sequence to ensure that all state elements not affected by this sequence are assigned the same value at the start of each dump (1). This will not in any way affect the functional behavior of the test since a normal restart after power-down corresponds to forcing the state elements into unknown states.

2.3 Scan dump processing flow

The scan dump data processing flow involves the interaction between several levels of design abstractions, such as functional verification environment; ATPG gate-level environment and the ATE environment. In addition, the actual debug activities involve several groups including: design/verification engineers; DFT engineers and product/test engineers as need dictates on a case by case basis. Figure 2 shows the scan dump processing environment used for the debug of UltraSPARC™ processors.

2.3.1 Data acquisition and translation

A particular functional test is failing for certain electrical conditions (frequency or voltage) and it has been determined that the failure signature is sufficiently stable for

scan dumps, i.e. the failing operational region is wide enough not to cause the device to drift across the boundary between passing and failing regions during the application of the scan dump procedure.

Referring to the upper box of Figure 2, first, the failing functional test is translated into a tester program with JTAG preambles and postambles added for initiating clock stop and scan out sequences for the proper cycle range. Next, in the ATE session, the device is subjected to the scan dump pattern and the raw data shifted out is recorded and stored in files. The cycle range is typically from several hundred cycles before the pin failure up to a few cycles after the pin failure. Several series of dumps are taken under both passing and failing electrical conditions for the cycle range of interest. The raw data collected by the tester is then processed through a translation flow which maps the bits onto the proper hierarchical names of the state elements in the design hierarchy. This step also involves the generation of waveform representations of the cycle range observed, which is useful in multiple cycle dumps and can be used for manual inspection of suspicious internal signals. The translation requires information about the order of the scan cells in the scan chain as well as information on global inversion from the cell to the scan out pin at which the data is being observed, which is obtained from the ATPG environment.

2.3.2 Raw data integrity checking

An important part of the processing is the raw data integrity

check which makes use of spare flip flops in the design with known fixed capture values. All spare elements are checked for correct expected values before any further processing takes place.

In addition, there are built-in registers, which act as cycle counters and increments for every internal cycle regardless of type of functional tests. These registers are checked between adjacent cycles for proper delta values. The integrity check ensures with high likelihood that the clock stop and scan operations executed normally without any data corruption and verifies the translation flow, which is particularly important in cases where the flow inputs and configurations come from many different environments and organizations. These registers also serve as a mean of synchronizing the silicon dumps with simulation dumps on a cycle by cycle basis. Furthermore, as observed in [8], in some cases the number of cycles consumed for reset on the physical device may be non-deterministic or different from the number of cycles in the simulation environment.

2.3.3 Comparison and filtering

Finally, the generated waveforms are applied to the latch divergence analysis flow, called LDA_1-n in Figure 2, which is described in detail in Section 3.3. The objectives in that procedure are to generate information on discrepancies in state elements between failing and passing conditions versus internal cycles. The results are presented in summary reports, graphs and lists of deviating state element names.

In addition to identifying failing state elements, the scan dump waveforms generated are useful for confirming or rejecting a failure hypothesis to decide whether further debug activities such as FIB (focused ion beam) edit or back-side probing should be conducted.

In parallel with the ATE activity, logic simulation of the functional pattern is launched, that can dump all scannable state elements. These simulation are typically done at the functional gate-level environment as access is needed to each scannable state elements. Simulation data provides a higher efficiency in the LDN reduction as it provides information on uninitialized state elements represented by X.

3 Latch divergence analysis and noise reduction

3.1 Latch divergence noise

Shmoo plots [9] are informative representations of multiple data logs to visualize the sensitivity of electrical failure types to various parameters such as supply voltage, frequency and temperature. In this section we consider failure types of functional patterns that manifest as an abnormal failing region in a voltage/frequency shmoo plot. Possible

causes of such failures are insufficient timing marginality or coupling effects or permanent physical defects such as resistive shorts. It is also assumed that the failure manifestation is repeatable. This is an important assumption for scan dumps as the cycles of interests are typically far before the failure manifests at the external pins so there is no way of verifying that the pattern indeed would fail had the test been left running without interruption till the end. Proving that the failure manifestation is electrically stable usually requires some ATE experimentation before the scan dump tester session can start.

A *passing dump* is defined as a scan dump taken from the functional pattern when it was run under passing voltage/frequency conditions and a *failing dump* is a dump taken when the same pattern was run under its failing conditions.

The basic problem in LDA of functional patterns is that comparing system states of repeated dumps taken under same cycles and same conditions (passing or failing) results in a significant number of state elements that are in different state as also observed in [1][6][8]. In the final comparison between failing and passing conditions, this electrical instability in latched system states manifests as a large number of unwanted state discrepancies spread all over the chip, which tend to hide the real cause of the failure. The occurrence of such deviations in the system state, which are irrelevant for the failure targeted is referred to as *Latch Divergence Noise* (LDN). Section 3.3 presents an efficient methodology for minimizing the amount of LDN based on filtering among repeated scan dump series.

3.2 Reasons for LDN in processors

The main contribution to the LDN is the lack of consistent initialization of internal memory arrays and state elements. Since there is no direct controllability of scannable state elements immediately prior to the scan dump cycle as in the case of ATPG, data from uninitialized blocks frequently propagates back and forth between memory arrays and scannable registers in a fairly random manner as the functional pattern is being applied. Although most of these affected registers usually get overwritten before any data get propagated to the pins, this inconsistency in internal state contributes to a large proportion of the internal LDN observed. A solution to this problem could be to write a dedicated functional program that initializes all memories and apply this program before running the target functional pattern for each scan dump. However, as indicated in [8], relying on an initialization pattern in the scan dump based debug may not be an option in many cases, for example if the initialization pattern itself fails, which may be the case in early silicon debug. In addition, in narrow failing regions one important goal is to minimize the external impact of the measurement procedure on the target device.

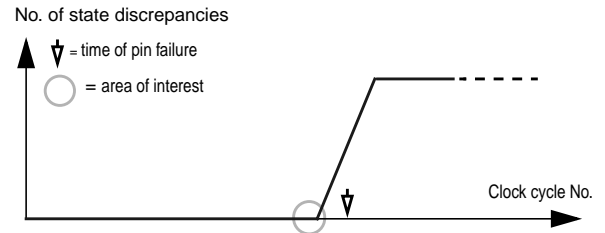
In some cases there are simulation dumps available from the functional pattern. Information about X state flip flops can then identify the majority of the effects of uninitialized state elements. However, logic simulators frequently fails to accurately model the spread of X's and may therefore not reliably identify all cases and the results are sometimes pessimistic. There may also be potential differences in the initialization procedure between the ATE and verification environments.

Since scan dump based debug operates on internal (CPU) clock cycles, another reason for LDN is various situations of non-determinism in internal cycles but determinism in external test cycle. Examples of such cases are: multiple cycle paths; use of asynchronous circuits; ATE input timing; and multiple clock domains. Another reason for LDN is the existence of state elements that depend on analogue properties, such as impedance and temperature controlled state machines and state elements within PLLs, which are not modelled at the functional level.

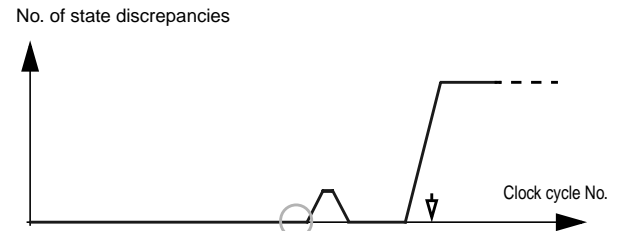
Ideally, if all reasons listed above were avoided the latch divergence between passing and failing dumps would follow the characteristics of the graph types shown in Figure 3a-b. These graphs show the number of state elements whose latched value is different between passing and failing conditions versus internal clock cycle number. The pin failure observed by the tester occurs later when the internal state difference caused by the failure propagates to the I/O. Figure 3b illustrates the case that the state deviation caused by a failure gets written into non-scannable memories and thus gets hidden for a number of cycles until, eventually, the error is read from the memory and propagated to the pins. An example of the latch divergence graph for a real failure is shown in Figure 3c, which shows a high average noise level of about 1% of all state elements prior to failure manifestation. It would not be possible to identify first failing cycle and failing state elements in this case.

3.3 LDN reduction

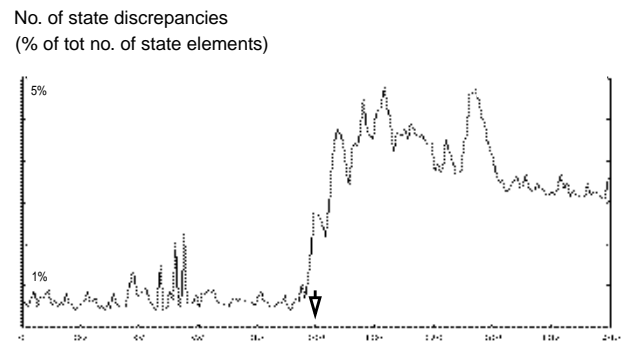
An efficient way of filtering out LDN is to use multiple dump series taken under identical conditions and exclude those state elements that differ under identical conditions from the final comparison between passing and failing conditions. The whole idea behind this filtering scheme is that exercising the device in multiple dump series as outlined in Section 2.2 is likely to cause some degree of randomization in the latch divergence and thus improve the identification of state elements that are not deterministic. This filtering process needs to be dynamic, which means that discrepancies are considered on a cycle-by-cycle basis. If logic simulation data is available, an additional level of filtering is provided by state elements assigned the X state. Figure 4 shows the details of this filtering process.



a) Example of expected ideal latch divergence graphs



b) Example of expected ideal waveforms with failure effects hidden in internal memory



(c) Example of latch divergence in a real microprocessor

Figure 3 Latch divergence under passing vs. failing condition

- (1) Simulation dump:
From a gate-level logic simulation dump of all state elements, X-state f/f's and associated cycle numbers are extracted and stored in a list, SX . This step is optional as simulation dumps may not always be available for certain specially written debug patterns.
- (2) Comparison among passing dumps: P_1 vs. P_2 , P_1 vs. P_3, \dots, P_1 vs. P_N
Since comparisons in the 0,1 Boolean domain represent an equivalence relation these $N - 1$ comparisons will identify all possible differences per cycle among all dumps taken in passing conditions. Each comparison generates a list, P_1P_j , which contains pairs of cycle number and f/f ID for each state element that differs.

- (3) Comparison among failing dumps:
 F_1 vs. F_2 , F_1 vs. F_3, \dots, F_1 vs. F_M
 The same comparison scheme is applied to all failing dump series.
- (4) Generation of exclusion list:
 All the state discrepancy lists P_1P_j and F_1F_k generated in (2) and (3) and simulation list SX are concatenated to form a global exclusion list which is used as input to the final comparison operation (5). This list consists of entries of type cycle number and a list of f/f IDs. The interpretation of this list is that, for each cycle, all f/f ID's that are present should be excluded from comparison for that cycle.
- (5) Final analysis:
 The results from the final comparison is a table of number of discrepancies vs. cycle number, which indicates the true discrepancies that are caused by the failure with all irrelevant noise removed. This LDA report, which is represented as a graph, shows the time at which the failure starts manifesting. If the filtering mechanism is efficient the shape of the graph will be of the types shown in Figure 3a-b. This procedure also generates a list of the hierarchical names of the state elements that differ, which can be used for further circuit analysis and/or logic cone analysis to identify the

fault site in logical gates or signal paths. The analysis can be performed in two ways depending on whether or not simulation data is available. It's irrelevant which of the passing and failing dump series are compared. The results will always be the same.

The number of series needed in (2) and (3) may vary with functional pattern and device as the noise level is closely related to the way the functional pattern accesses the embedded memories. Empirically, $N=M=3$ has shown to be a reasonable compromise between LDN reduction efficiency and tester resource requirement.

Notice that the exclusion list generated by comparing failing dumps (3) will not cause the elimination of any real discrepancies caused by the failure as the failure manifestation is assumed to be repeatable. Consequently, any discrepancies obtained in the steps (3) and (2) are entirely caused by LDN. The gain in (3) is that, as the electrical conditions are usually significantly different compared to passing conditions, the LDN is more likely to manifest in a different manner and therefore enable the identification of more LDN, which can be excluded in the final analysis (5). In general, the higher level of randomization the device is subjected to without crossing the passing/failing boundaries the more efficient becomes the LDN reduction.

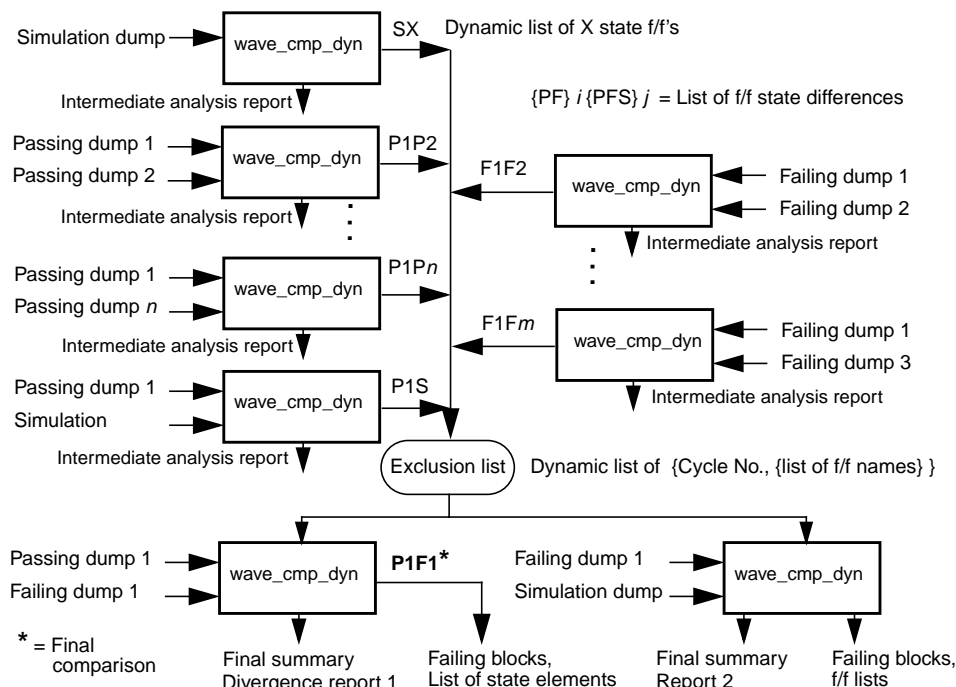


Figure 4 Procedure for latch divergence analysis with noise filtering

It should be mentioned as a limitation that the LDN filtering scheme cannot be applied in cases of unstable failure signature as the LDN filtering scheme may eliminated state deviations caused by the real failure. Furthermore, since each cycle dumped is generated from a different application of the same functional pattern the resultant waveforms may become contaminated by containing values originating from an erroneous state for some cycles and an error-free state for other cycles.

As indicated in Figure 4, latch divergence reports are generated for all intermediate comparison steps. These results are mainly to be used for inspection of the LDN manifestation and for sanity checks. Certain types of functional patterns usually generate similar LDN signatures. This information can also be used as a guidance in the determination of the number of dump series required.

In case of hard failures of functional patterns, without any passing region a simulation dump may act as a golden (passing) dump.

4 Case studies

Scan dump based debug normally uses the information provided by the shmoo plot representation [9] of tester data logs over failing/passing areas to determine at what operational conditions to conduct scan dumps as shown in Figure 5. These plots show the frequency vs. V_{DD} roll-off characteristics expected from a typical speed path related design problem.

In processor debug, the types of failures for which the proposed LDA is suitable are functional-only failures caused by electrical design errors or physical implementation errors, which cause an abnormal operational passing regions.

Initially, without any root cause hypothesis a window of 500 internal core cycles or more prior to pin failure is dumped. The total time required in generating and processing tester scan dumps for functional failure debugs is in the order of 1-3 hours per functional pattern, assuming 6 dump series of 500 internal cycles each. The major contribution is the tester time required to verify stability of the failure signature. In addition, during the tester scan dump session, a significant amount of time is consumed for the data transfer of sampled raw data between ATE and local file servers.

The LDN filtering processing time is in the order of 15 minutes. As the debug activity proceeds, a wider or different window may be needed. In addition, each debug case typically involves several different functional patterns under various failure signatures.

4.1 Case study 1

Figures 6-8 show the latch divergence graphs of a silicon

bring-up debug case for an UltraSPARC™ processor under various conditions of filtering. This failure was caused by a speed path in the floating point unit, which caused an instruction code to change. Figure 6 shows the proportion of state elements in different states when comparing raw data between passing and failing dumps without any noise filtering applied and Figure 7 shows the comparison between two passing dumps taken under identical conditions. It can be seen that even for the passing vs. passing case there is a significant noise level, which means that usually no isolation of the failure manifestation can be done.

In general, studying the signature of these graphs gives some global insight into the way a particular functional test operates in terms of usage and feeding uninitialized values into the state elements over the life of the test. High values indicate a high degree of state elements that are not in use or will not propagate data to the pins. Certain peaks tend to occur periodically and correlate with deterministic sequences taking place internally such as instruction pipe lining, which periodically causes more or less unknown data to be written into registers or with the triggering events of other clock domains running at lower frequency.

Figure 8 shows the divergence graphs from the same case under two conditions of filtering. Graph g_1 is the case in which noise reduction is done solely based on excluding

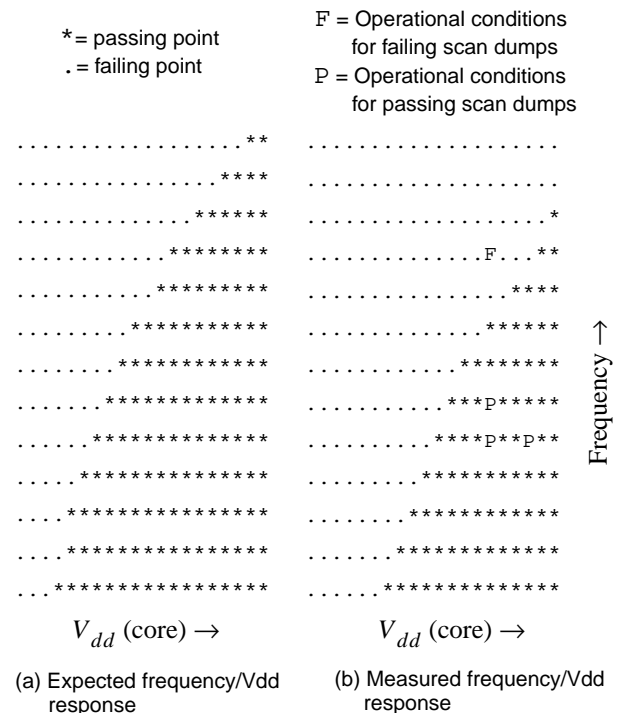


Figure 5 Shmoo plots for scan dumps

simulation X state flip flops. It can be seen that the noise level goes down to a more manageable level, although still too high for a direct identification of first failing state element. Finally, graph g_2 shows the case in which the full LDN reduction of Section 3.3 was applied and the noise level prior to failure falls to zero for most of the cycle range, which is close to the optimal results. (The optimal result is an LD graph of the type shown in Figure 3a.) The few number of spurious state differences obtained in this case are manageable and this case will provide useful debug information to the design and verification engineers.

It should be noted that the efficiency of the noise reduction filtering for a certain case is somewhat dependent on the type of functional pattern since different patterns may cause more or less activation and writing of unknown values so the number of dump series may need to be increased if the noise level is too high. The noise filtering $N=M=3$ was used for all the debug cases presented in this section. It is not always possible to get all the way down to zero LDN level, but even with a small non-zero noise level it is usually possible to provide useful debug information with some knowledge about in what major blocks of the design the failure is expected to occur.

4.2 Case study 2

Figure 9 shows the LD graphs from another debug case for an UltraSPARC™ processor. These dumps are from a functional pattern that failed only in one out of all possible clock ratio modes. Graph a_1 shows the typical characteristics of the LDN with no filtering conducted and, again, as graph a_2 shows, the improvement of the complete LDN reduction scheme in this case is close to the optimal results.

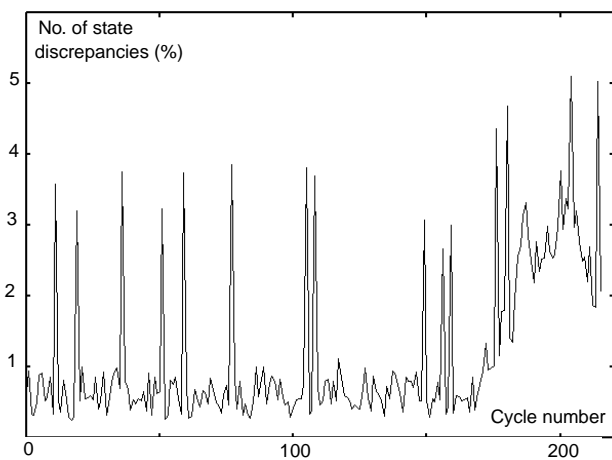


Figure 6 LD: Failing vs. Passing dumps

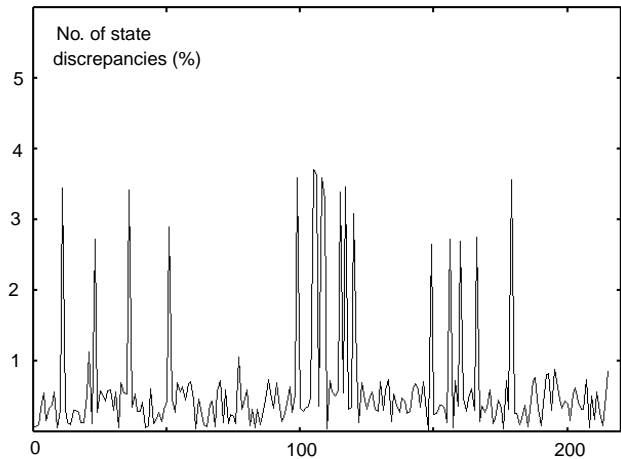


Figure 7 LD: Passing1 vs. Passing2 dumps

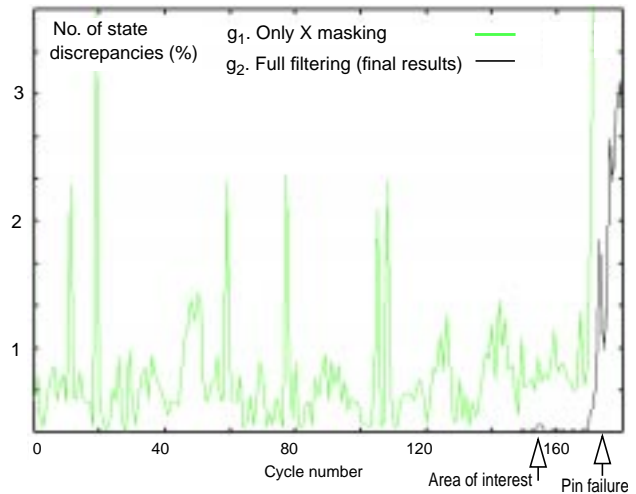


Figure 8 LD: Failing vs. Passing dumps

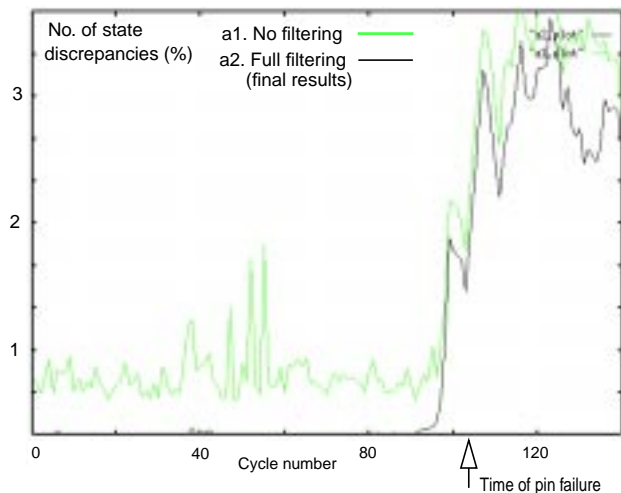


Figure 9 LD: Failing vs. Passing dumps

5 Conclusions

A methodology for managing and processing scan dump data and for reducing latch divergence noise has been proposed. Using the proposed scan dump processing flow, scan dumps have become an important part of the test and debug flow of the UltraSPARC™ products and have been successfully applied to numerous debug cases both for identifying state deviations and for confirming root cause hypotheses. Several real silicon bring-up debug cases have shown the efficiency of the latch divergence noise reduction scheme in failure isolation of functional patterns for speed path types of failures.

6 Future work

Several improvements related to the diagnostics capability of the processing flow are currently under development including automated logic cone analysis similar to the methods used in ATPG failure analysis, which spans over both temporal and spatial LDA. In addition, there are significant on-going design efforts, which aim at reducing LDN per design by adding special control logic.

7 Acknowledgements

Microprocessor debug is a joint effort between many different engineering groups. The authors would like to thank the hard work and dedicated support from the design, validation, test engineering and silicon bring-up teams of the UltraSPARC™ projects.

Thanks also go to the ITC reviewers for their valuable comments and suggestions.

References

- [1] Kinra, A., "Towards Reducing Functional Only Fails for the UltraSPARC Microprocessors", *Proc. International Test Conference*, 1999, pp. 147-154.
- [2] Josephson D. et al., "Debug Methodology for the McKinley Processor", *Proc. International Test Conference*, 2001, pp. 451-460
- [3] Levitt, M. E., "Testability, Debuggability, and Manufacturability Features of the UltraSPARC™-I Microprocessor", *Proc. International Test Conference*, 1995, pp. 157-166.
- [4] Golshan, F., "Test and On-line Debug Capabilities of IEEE Std 1149.1 in UltraSPARC™-III Microprocessor", *Proc. International Test Conference*, 2000, pp. 141-150.
- [5] Parulkar, I., et al., "A Scalable, Low Cost Design-for-Test Architecture for UltraSPARC™ Chip Multi-Processors", *Proc. International Test Conference*, 2002, pp. 726-734.
- [6] Hao, H. and Avra, R., "Structured Design-for-Debug - The SuperSPARC II Methodology and Implementation", *Proc. International Test Conference*, 1995, pp. 175-183.
- [7] Katz, J., "A Case-Study in the use of Scan in microSPARC™ testing and debug", *Proc. International Test Conference*, 1994, pp. 456-450.
- [8] Holdbrook, K. et al., "microSPARC™: A Case-Study of Scan Based Debug", *Proc. International Test Conference*, 1994, pp. 70-75.
- [9] Josephson D., "The Manic Depression of Microprocessor Debug", *Proc. International Test Conference*, 2002, pp. 657-663.