Creating Value Through Test

Erik Jan Marinissen[1] Bart Vermeulen[1] Robert Madge[2] Michael Kessler[3] Michael Müller[3]

[1] Philips Research Laboratories
IC Design – Digital Design & Test
Prof. Holstlaan 4 – WAY-41
5656 AA Eindhoven
The Netherlands
erik.jan.marinissen@philips.com
bart.vermeulen@philips.com

[2] LSI Logic Corp.
Product Engineering
23400 N.E. Glisan Street
Gresham, OR
United States of America
rmadge@lsil.com

[3] IBM Deutschland Entwicklung GmbH
Hardware Development
Schönaicherstrasse 220
71032 Böblingen
Germany
mkessler@de.ibm.com
mulm@de.ibm.com

Abstract

Test is often seen as a necessary evil; it is a fact of life that ICs have manufacturing defects and those need to be filtered out by testing before the ICs are shipped to the customer. In this paper, we show that techniques and tools used in the testing field can also be (re-)used to create value to (1) designers, (2) manufacturers, and (3) customers alike. First, we show how the test infrastructure can be used to detect, diagnose, and correct design errors in prototype silicon. Secondly, we discuss how test results are used to improve the manufacturing process and hence production yield. Finally, we present test technologies that enable systems of high reliability for safety-critical applications.

1 Value for IC Designers: Silicon Debug

Bart Vermeulen – Philips Research Laboratories

For today's system chip designs, no designer can guarantee that all design errors are found before first tape-out, despite meticulous analysis and verification [1, 2]. These errors can go undetected because the verification methods are only applied to an abstract model of the IC and not to the actual silicon. When more detail is added to the IC model (to better match it to the reality), most verification methods can no longer be applied exhaustively because of the computational cost involved. Given that design teams are under continued time-to-market pressure, it is important to find design errors quickly. Apart from improving the pre-silicon verification methods, it is worthwhile to improve existing silicon debug methods and complement them with other approaches.

Debugging silicon is difficult because the engineer's ability to observe the chip's internal processes (e.g., states and data flow) is limited. We can identify two traditional approaches to diagnose faulty behavior of a chip in the application. First there are diagnosis methods such as visual inspection (e.g., electron-beam probing) and direct physical contact techniques (e.g., using probe needles). A drawback of these methods is that it is often difficult to pin-point the exact location of an error in the chip without additional information. To examine the entire layout of a design for a possible error cause using only visual inspection is clearly not feasible. In addition, with decreasing feature sizes and an increasing number of metal layers in modern process technologies, the usage of these debugging techniques is becoming more difficult. A second method involves manual and ad-hoc debug techniques, such as trial-and-error programming of the chip. Although these techniques can help to provide information on the type of design error and its location, they are often unpredictable in both success rate and time required to sufficiently localize a design error.

One other method is to provide electrical observability of on-chip signals through the device pins. The most popular application of this method involves reusing the scan chains already inserted in the design for manufacturing test. These scan chains are used to provide state dumps of flip flop and memory content while the chip is in the application. This method is so popular because the scan chains allow observability of the chip's full state, while minimizing the amount of additional hardware required to implement this functionality, provided that design-for-test has already been implemented.

Below, a generic debug methodology is described that utilizes scan chains for silicon debug [3]. This debug methodology consists of two steps. First, certain Design-for-Debug (DfD) structures are added during the design phase of a chip. Second, debugger software that executes on a computer connected to the application board, is used to debug the chip. The debugger software is used to control the chip and its debug hardware in the application.

1.1 Scan-Based Silicon Debug

A scan-based silicon debug methodology is based on re-using scan chains, inserted for manufacturing test, to analyze design errors. To allow access to the scan chains in the application, certain modifications need to be made to the design. Figure 1 gives an overview of an architecture that allows this access. Please note that only two cores are shown for clarity. This architecture can be easily extended to cover any number of cores.

All debug operations are controlled from the IEEE Std. 1149.1 Test Acces Port (TAP) controller. The advantage of using this controller for debug is that the TAP controller and its associated pins (1) have often already been included in the design to allow for board-level manufacturing test, (2) are easy to access when the chip is put on its application board, and (3) are themselves not used by the application.

The IEEE Std. 1149.1 TAP is essentially a serial port, with one serial input (TDI) and one serial output (TDO). In Figure 1, all debug functionality is therefore controlled from one or more serial debug control blocks (DCB). These DCBs are under control of the test hardware, to

1530-1591/03 $17.00 © 2003 IEEE

allow them to be fully tested during manufacturing test and to control them from the TAP controller.

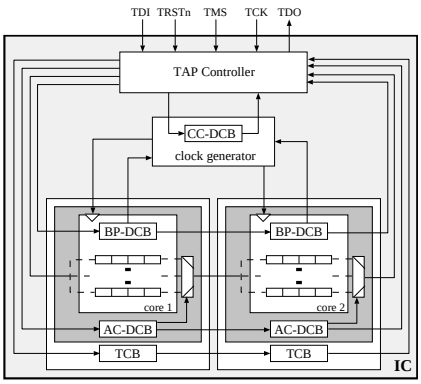

Figure 1: Scan-based silicon debug architecture.

The architecture in Figure 1 implements the "ABC" of scan-based silicon debug; Access to the scan chains, Breakpoints to detect one or more internal events, and Clock control. These are explained below.

Access
The scan chains are controlled via dedicated TAP data registers. One clock domain is scanned out at a time, as the TAP has only one serial output. In debug mode, the scannable flip flops are concatenated into debug scan probes, one in each clock domain. This concatenation is performed at core-level using a debug shell, shown in gray in Figure 1. The debug shell hides all core-specific details, such as number of scan chains and clock domains inside the core, and provides a single, uniform hardware interface for debug at the integration level. Each core provides one serial input and one serial output to its debug scan probes, and a standardized debug interface to an Access Control Debug Control Block (AC-DCB), that controls the scan probe multiplexing. At the integration level, all serial inputs and outputs are daisy-chained.

During a silicon debug session, the AC-DCB is used to select each core-level scan probe in turn, while its input and output are connected to the chip's TDI and TDO pins. To the user this complexity in accessing the various scan probes is hidden. The debugger software takes care of issuing the proper TAP commands to select each probe in turn, and translating the bit-streams received on the chip's TDO output to individual flip flop, multi-bit register, and memory content.

Breakpoints
To examine the behavior of the chip in detail using state dumps, it is required to first determine at which point during the chip's execution a state dump has to be made. An on-chip breakpoint mechanism is added to the design to allow the chip to be stopped at regular intervals during its execution. This regularity is important, as it allows the state dumps to provide a clear insight in the data and control processing going on inside the chip over time. If the breakpoint mechanism does not provide enough temporal resolution, the 'blind spots' in between state dumps, where no information can be obtained, might seriously complicate and lengthen the debug process.

During a silicon debug session, the breakpoint mechanism is programmed via a Breakpoint DCB to stop the chip at a certain point in time. After stopping, a scan dump via the TAP is made and compared to golden reference data. Based on this comparison, the end user can use the debugger software to re-program the breakpoint mechanism to stop the chip at a different point in time to obtain more debug information. Ultimately this should help the user to diagnose the design error. Figure 2 shows the debug flow in which state dumps are used to analyze a design error.

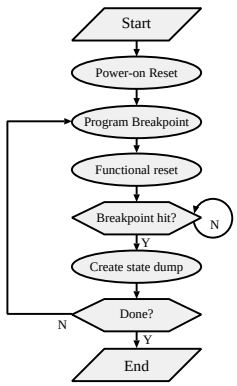

Figure 2: Flow used for scan-based silicon debug.

Clock Control
The chip is stopped by gating the on-chip clocks. Stopping the clocks effectively freezes the content of all flip flops and embedded memories. After the clocks have stopped, the circuit can be safely switched to debug scan mode. In debug scan mode, each internal clock is selected in turn to allow the content of the corresponding scan probe to be scanned out. The stop and re-activation functions for the internal clocks for scan chain shifting are added to the on-chip clock generation unit and controlled from a Clock Control DCB.

The scan-based silicon debug architecture presented above uses scalable modules to implement the required silicon debug functionality. These modules are added to each of the submodules in the design, allowing a core-based design, test, and debug methodology to be followed. The advantage of this is that the debug architecture can be completely tailored to suit a particular design. As an example, a design may contain cores with a different number of scan chains and/or clock domains. The debug shell hides these test-details by providing one core-level debug interface. Each of these cores has the same debug interface, allowing a design-for-debug tool to automatically perform the core interconnect at integration level. This reduces the time required for a designer to add design-for-debug hardware to a design. In addition, because of the scalable nature of the debug architecture, an estimation tool can be used to make an educated trade-off between for example breakpoint granularity and hardware cost.

1.2 Silicon Debug Successes

This debug methodology has been successfully applied to a number of large digital system chips within Philips.

The debug facilities on the CPA chip [4] proved essential in verifying its video-processing capabilities. The first silicon exhibited several problems during initialization, causing the chip to malfunction after 50 to 75 video frames. The source of the problems was detected by examining scan dumps taken at each cycle during the initialization sequence. Using the debug controllability, we replaced the faulty ROM-based initialization code by loading corrected code into an external SDRAM and instructing the chip to fetch its code from that SDRAM. After this fix, the designers could successfully verify all image processing functions without further silicon spins.

For the PNX8525/Viper chip [5], and more recently for the PNX7100/Chrysalis chip, the state dumping functionality allowed designers to correctly diagnose the faulty behavior of subcomponents. In one case, the flow shown in Figure 2 was repeatedly used to back-track mismatches between simulation states and silicon states back to the output of a single gate. Under specific circumstances, the output of this gate was not able to drive an internal signal to the correct value in time, which ultimately resulted in erroneous behavior. Once this was discovered, the fix was easy to implement and verify.

As a result of these successes, a standardization activity is currently ongoing to make this debug methodology available to all digital designs within Philips.

1.3 Conclusion

The presented debug methodology successfully relies on the existing scan chain access to debug prototype silicon. With only little extra hardware to create debug access, breakpoints, and clock control, it becomes possible to obtain state dumps while the chip is in its application. These state dumps provide the debug engineer with essential information to locate design errors still left in the chip, and overall help to reduce the number of silicon spins and time-to-market of the chip.

---

2 Value for Manufacturing: Understanding Defects and Improving Yield

Robert Madge – LSI Logic Corporation

The increasing complexity of fabrication processes and the proliferation of foundry fabs has resulted in significant challenges to achieve competitive yield at the introduction of new technologies and for fast ramp to volume manufacturing. Time-to-market and quality goals are increasingly hard to meet due to the growing product complexity and gate count. This section highlights the immense added value of test data in attempts to overcome these challenges and meet the market and profitability requirements.

2.1 Process Yield Improvement

Process and yield engineers have traditionally relied on the pass/fail 'bin' data from wafer sort and package testing to monitor the capability of their process technology and to improve the defect density and parametric performance. Bit fail signature data has long been a yield improvement method for discrete and embedded memory manufacturers [6, 7, 8], but for logic products, raw test data has recently become a critical part of the yield engineer's tool set. This has significantly increased the added value of testing to the manufacturing world.

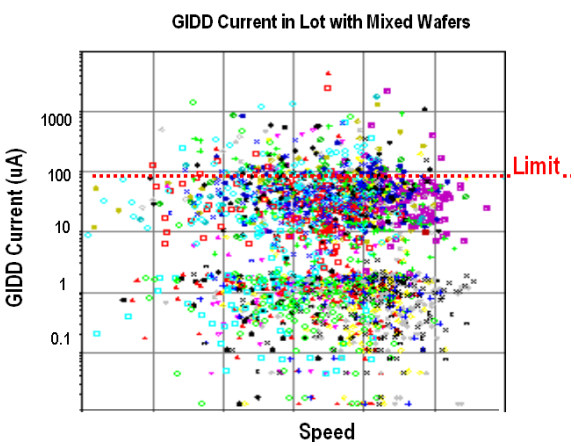

Figure 3: A bi-modal I_{DDQ} distribution due to silicon ingot defects.

Figures 3 and 4 show examples of how raw ATE data is analyzed for yield learning and process improvement. In Figure 3, I_{DDQ} data is shown to have a bi-model distribution, even though only a small percentage fail the test limit. The cause of the bi-modal behavior was gate oxide defects due to incoming silicon stacking faults from one silicon vendor causing 5–10% yield loss on logic ASICs and 50–60% yield loss on ASICs with embedded memory, and was resolved by reducing the oxygen content of the incoming silicon. This problem could not have been identified without the raw I_{DDQ} data. Figure 4 shows an across-wafer variation of minimum working voltage for a 1.8 V functional core. Even though the core is not failing at the test voltage of 1.8 V, the raw data identified a marginality caused by a process defect. Early identification and fix of this defect was critical to the yield learning for this product.

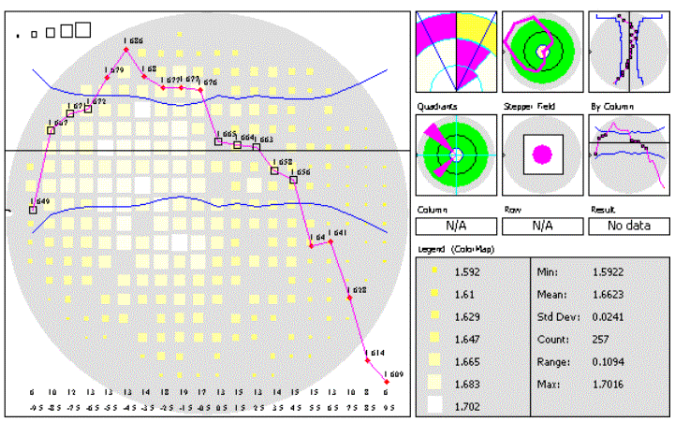

Figure 4: Intrinsic minimum voltage variation across the wafer.

2.2 Quality and Reliability Improvement

Quality and reliability engineers are also utilizing raw test signatures, particularly I_{DDQ} (Figure 5) and Min-V_{DD} (Figure 6), to understand the latent defectivity of the silicon process and the potential for quality and reliability improvements by screening die with abnormal signatures (or statistical outliers) [9, 10]. The more recent trend is to utilize the raw test results to predict the intrinsic (defect-free) behavior based on neighborhood or deltas and reduce the variance such that defective outliers can be clearly identified and eliminated for quality improvement [11]. Statistical post-processing methods have been developed which move the pass/fail decision making step from on-tester to off-tester (see Figure 7). This allows much improved identification of the outlier die and has been shown to result in 40–60% improvements in Early Fail Rate (EFR) and customer-return Defects-Per-Million (DPM) [12, 13].

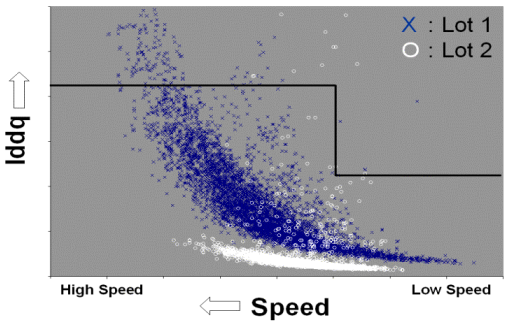

Figure 5: I_{DDQ} data from two silicon lots from the same process. Note outlier die from both lots which cannot be efficiently screened with the on-tester limit.

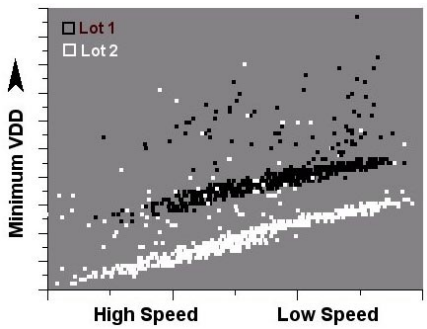

Figure 6: Min-V_{DD} vs. device speed for two different lots, clearly showing Min-V_{DD} outliers and lot-to-lot intrinsic variation.

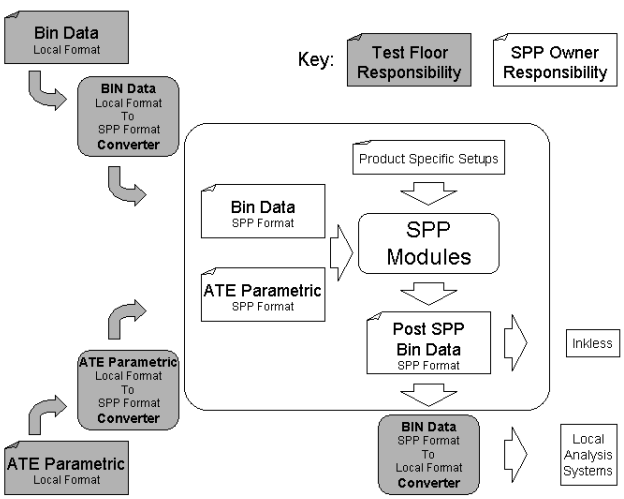

Figure 7: Statistical Post-Processing (SPP) data flow showing the use of raw parametric ATE data in off-tester pass/fail decision making and inkless re-binning.

2.3 Failure Analysis

Failure analysis continues to provide immense value to the yield ramp and quality and reliability improvement. Test data is contributing significantly to this continued success through mapping of defects to structural test datalogs such as scan or I_{DDQ} to in-line defects [14, 15, 16]. Figure 8 shows how the ATE failure datalog is combined with the design files to identify the failing nets and isolate the defect causing the failure. The defect cause in this case was a metal bridge, which would be almost impossible to identify using traditional failure analysis techniques.

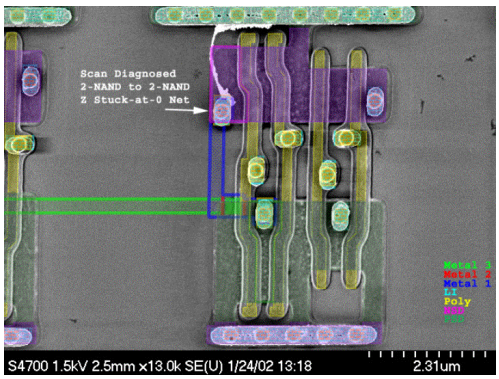

Figure 8: A bridging defect identified by the scan datalog diagnosis method.

2.4 Test Cost Reduction/Adaptive Testing

Test engineers have always utilized raw test data to isolate correlation or repeatability issues in the test program or hardware. Recent trends, however, have been towards collection and analysis of large volumes of test data over longer periods of time. Test times can be reduced through elimination of unnecessary or redundant tests or vectors and adaptive test methods can be introduced where the results of certain tests can determine the need for more extensive testing based on probability of failure. More recent trends towards foundry wafer manufacturing has led to the importance of adaptive testing due to the potentially variable quality of silicon coming from the fabs. Test time improvements of 26% have been reported with adaptive test methods while also improving product quality [17].

2.5 Conclusion

ATE testing and the raw data results provide ever-increasing value in the manufacturing of complex integrated circuits. No longer are the raw data results ignored in favor of pass/fail 'bin' results, rather the results are a critical part of the yield learning, quality and reliability improvement and cost reduction process in all areas of manufacturing and process or product development.

3 Value for End Customers: Highly Reliable Systems

Michael Kessler & Michael Müller – IBM Deutschland

The heart of a zSeries 900 system (S/390) consists of a Multi-Chip Module (MCM) and a number of surrounding support chips and units. The 2000 design generation MCM, operating at 1 GHz, contains 20 Central Processors (CP), 2 Cache Controllers (SCC), 8 Cache Chips (SCD), 4 Memory Bus Adapters (MBA), and a Clock Chip (CLK) for clock distribution. The test techniques were developed and improved from generation to generation to fulfill the quality expectations of the customers. Business customers with mission-critical applications are highly dependent on the premium quality and reliability of the machine, 365 days a year, 24 hours a day.

The goal is to minimize all possibilities for malfunction and to improve reliability by removing all early life problems by stressing the components and the complete system. You also want to be cost effective overall (from wafer to system) by generating highest quality. This is *not* a contradiction, but a necessary prerequisite.

Many components are necessary, to guarantee the reliability targets for such a machine. First of all, perfection is needed during testing and stressing before shipment. From the system view a consequent system design for Reliability, Availability and Serviceability (RAS) is necessary, which includes redundancy, no Single Point of Failure (SPOF), fault-tolerance, recoverability, traceability, and diagnosability after shipment.

3.1 Testing and Stressing

Design-for-Test
The base for test is a structural test approach of the silicon with highest possible DC and AC fault coverage. Logic Built-In Self Test (LBIST) and Array BIST (ABIST) are perfectly suitable for the following reasons. The tester resource requirements are minimal. Only initialization and measurement of the pre-calculated signature are necessary. The LBIST can be applied at-speed and beyond (for margin testing) through on-chip clock generators. The LBIST is capable of applying pseudo-random and programmable weighted pseudo-random patterns. The design is made random testable for highest possible test coverage, e.g., "99% AC test coverage using LBIST only" [18]. The test time is kept low, by means of the STUMPS architecture [19] with many short STUMPS channels.

Test
The LBIST/ABIST are applied through several packaging levels from wafer, to single chip, to MCM, to the various system configurations and during power-on at the customer site. It is also used for BurnIn and RunIn at chip-level and system-level. The wafer test is applied through a Reduced-Pin-Count tester interface and consists of parametric tests, Flush/Scan, LBIST/ABIST, and supplemental stored patterns. The single-chip test uses the same tests again plus the external I/O tests. MCM test reuses the LBIST/ABISTs and at-speed interconnect tests.

Above tests are applied to guarantee functionality at all voltage, temperature, pattern, and cycle time corners. LBIST/ABISTs are used as sorting criteria together with other speed indicators.

Due to the nature of semiconductor chips, the difference between worst case (slow) and best case (fast) is fairly high. During wafer and single-chip test, the Flush-Delay through a Shift-Register-Latch (SRL) chain is used as speed indicator. A Performance-Screening-Ring-Oscilator (PSRO) could be used as well. A given chip must not only fulfill its raw flat cycle time limit (e.g., 1 GHz), but must perform according to its predicted cycle time calculated from its own Flush-Delay [20]. Chips that perform outside of a narrow performance distribution are *not* put into a slower sort bucket, but get discarded. Any *outliers* get removed, they are suspect to fail in the future. Tests are applied at very low and at very high voltages (outside of the functional window) to accelerate and detect certain fault behaviors.

BurnIn, RunIn
LBIST/ABIST is used to operate the chips at elevated temperatures and voltages to accelerate any early-life failures that do not cause failures initially, but later (possibly in the customer's application) so-called 'reliability failures'. RunIn is used to stress the chips at and beyond the target cycle time, again to improve the reliability. Later, after assembly of the machine, extended stress tests use again LBIST/ABISTs.

3.2 Usage of Built-in Test Equipment in the Field

The zSeries z900 is an example of a system which uses instantaneous error detection on each CPU and on each I/O operation while the system is executing the customer's workload. Instantaneous detection is the detection of an error in the ongoing operation prior to committing the result to any other functional unit [21]. All arrays (L1-I-Cache, L1-D-Cache, TLB, BHT, L2-Cache, L3-Memory) and all buses use error-correcting codes to detect and correct errors. The state machines are implemented with redundancy in the state encoding, invalid state-detection, and sequence checking. The checking in the PU chips for example is implemented by duplicating the Execution Units (Instruction Unit, Floating-Point Unit, Fixed-Point Unit) and performing a result compare before committing the result to the self-checked Recovery Unit (R-Unit) [22].

Usage of LBIST/ABIST for Maintenance
At system power-on or whenever new hardware, either for upgrade or repair, is added to the system, the LBIST and ABIST are executed to ensure the new hardware including the instantaneous error detection circuitry is working before the component joins the configuration. The server executes ABIST to find failures in the large arrays and repairs the failure using an extension to the fuse-programmable array-line-relocate method used in manufacturing to increase the yield. In the rare case of larger damage, when the failure cannot be self-healed, single array lines, quadrants of the large arrays, and up to complete chips can be de-configured to allow for an emergency operation in degraded mode, until the scheduled repair can be performed at the customer's convenience.

Error Reporting, Containment, Recovery
There are two major error-reporting-methods in the z900 system. For 'clock-running' errors, used for less severe errors where the unit continues to function through the error, the reporting is done in-band. For 'clock-stop' errors, used for severe errors, where the unit is no longer functioning, the reporting is done out-of-band to the service subsystem by scanning out the SRL chain. In both cases the error information is collected to determine the amount of damage, to trigger the appropriate recovery, and to perform fault isolation. The error information together with the result of the recovery is stored as First-Failure-Data-Capture (FFDC). The information collected from the detection circuitry identifies the offending unit and the scope of the error.

Recovery is attempted and will be successful in case of a transient fault. If the error is caused by a frequently occurring intermittent fault and thus exceeds a certain threshold, it is considered permanent [23]. For a permanent fault recovery will activate an alternate path, a spare element, or, if none available, inform the operating system about the exact point of interruption and the precise amount of damage to the interrupted operation. This allows the operating system to associate the failure with the impacted application and preserve the unaffected applications.

Fault Isolation and Repair
The captured failure data allow effective automatic isolation down to the Field Replaceable Unit (FRU). The server generates a call-home to the maintenance provider that includes the failed FRU, the current system status, FFDC, and the scope of the repair action, such that the service personnel can schedule the repair at a convenient point in time with the customer. The service personnel does not need to run any diagnostics to reproduce the failure, but has the spare part right at hand to replace the defective part in a concurrent repair on-line. This reduces

the repair time dramatically.

DRAM Sparing and Cache Line Relocate
Accumulation of soft-errors in seldom accessed storage can be avoided by continuously scrubbing the complete storage to correct single bit errors. Scrubbing uses the error syndrome to count the errors in each DRAM module. When the count of errors exceeds a specified threshold, a spare DRAM module is activated. The content of the faulty module is copied into the spare module. Any store operation stores the data in both modules. When copying is completed, the faulty module is replaced by the spare module. The self-repair using a spare DRAM avoids downtime for memory card replacement [21]. Similar monitoring is applied to the large caches. When a cache line exceeds a certain threshold for single-bit error-correction-events, the cache line is marked unusable and scheduled for relocation to a spare cache line at the next power-on [24].

CPU Instruction Recovery and Sparing
The failing current instruction is retried when the R-Unit detects a mismatch, using the correct committed results of the previous instructions contained in the R-Unit. The PU chip is fenced and its clocks are stopped in case retry was not successful. The clock-stop event triggers the Service Subsystem of the z900 server to scan out the R-Unit SRL chain. This last valid checkpoint is sent to the remaining host PUs which determine the target spare. The R-Unit contents is passed to the designated spare, which begins a "self-initiated brain transplant". The spare has then the identity of the clock-stopped PU and resumes execution where the failed PU left off by retrying the same instruction [21]. Dynamic CPU sparing permits the system to be restored to full capacity in less than a second as opposed to hours.

3.3 Conclusion

Built-in test equipment identifies defect chips. In combination with BurnIn, RunIn, and stress tests with higher guard-band conditions, it sorts out even potentially defect chips. Removing these chips as early as possible in the production cycle minimizes overall costs from production to warranty and service and furthermore protects the customer from outages. Since the BISTs clearly separate technology and manufacturing failures from logic design flaws it speeds up bring-up, ties less capital to bring-up hardware and improves time-to-market. Checking logic protects integrity of customer data, identifies the fault of a unit and is thus the base for error containment, transparent recovery, activation of alternate paths and spare parts to maximize system availability, to completely avoid repair or at least defer to scheduled repair. The data collected at the first occurance of a failure lay the foundation for automatic fault isolation down to the field replaceable unit, instead of relying on failure reproduction through diagnostics, and to call-home for the correct spare part in order to reduce the repair time and cost.

References

[1] E.J. Aas et al. Quantifying Design Quality Through Design Experiments. *IEEE Design & Test of Computers*, Vol. 11(No. 1):27–38, 1994.

[2] K. Holdbrook, S. Joshi, S. Mitra, J. Petolino, R. Raman, and M. Wong. microSPARC: A case-study of scan based debug. In *Proceedings IEEE International Test Conference (ITC)*, pages 70–75, 1994.

[3] B. Vermeulen, T. Waayers, and S.K. Goel. Core-Based Scan Architecture for Silicon Debug. In *Proceedings IEEE International Test Conference (ITC)*, pages 638–647, October 2002.

[4] B. Vermeulen and G.J. Rootselaar. Silicon Debug of a Co-Processor Array for Video Application. In *Digest of IEEE Intnl. High-Level Design Verification and Test Workshop (HLDVT)*, pages 47–52, November 2000.

[5] B. Vermeulen, S. Oostdijk, and F. Bouwman. Test and Debug Strategy of the PNX8525 Nexperia Digital Video Platform System Chip. In *Proceedings IEEE International Test Conference (ITC)*, pages 121–130, 2001.

[6] W. Maly. Yield Diagnostics Through Interpretation of Test Data. In *Proceedings IEEE International Test Conference (ITC)*, pages 10–20, October 1987.

[7] M.A. Merino. SmartBit : Bitmap to Defect Correlation Software for Yield Improvement. In *Proceedings IEEE/SEMI Advanced Semiconductor Manufacturing Conference*, pages 194–198, 2000.

[8] C. Gloor. Embedded Memory Analysis for Standard Cell ASIC Yield Enhancement. In *Proceedings International Symposium for Testing and Failure Analysis (ISTFA)*, pages 69–76, 2000.

[9] A. Gattiker and W. Maly. Current Signatures. In *Proceedings IEEE VLSI Test Symposium (VTS)*, pages 112–117, April 1996.

[10] T.J. Powell, J. Pair, M. St.John, and D. Counce. Delta IDDQ for Testing Reliability. In *Proceedings IEEE VLSI Test Symposium (VTS)*, pages 439–443, April 2000.

[11] W.R. Daasch, R. Madge, K. Cota, and J. McNames. Neighbor Selection for Variance Reduction in I_{DDQ} and Other Parametric Data. In *Proceedings IEEE International Test Conference (ITC)*, pages 92–100, October 2001.

[12] R. Madge, M. Rehani, K. Cota, and W.R. Daasch. Statistical Post-Processing at Wafersort – An Alternative to Burn-in and a Manufacturable Solution to Test Limit Setting for Sub-micron Technologies. In *Proceedings IEEE VLSI Test Symposium (VTS)*, pages 69–74, April 2002.

[13] R. Madge et al. Screening Min-V_{DD} Outliers using Feed-Forward Voltage Testing. In *Proceedings IEEE International Test Conference (ITC)*, pages 673–682, October 2002.

[14] A. Kinra, H. Balachandran, R. Thomas, and J. Carulli. Logic Mapping on a Microprocessor. In *Proceedings IEEE International Test Conference (ITC)*, pages 701–710, September 2000.

[15] C. Hora, R. Segers, S. Eichenberger, and M. Lousberg. An Effective Diagnosis Method to Support Yield Improvement. In *Proceedings IEEE International Test Conference (ITC)*, pages 260–269, October 2002.

[16] B. Benware. Logic Mapping on ASIC Products. In *Proceedings International Symposium for Testing and Failure Analysis (ISTFA)*, pages 579–586, 2002.

[17] S. Benner and O. Boroffice. Optimal Production Test Times Through Adaptive Test Programming. In *Proceedings IEEE International Test Conference (ITC)*, pages 908–915, October 2001.

[18] M. Kusko, B. Robbins, T. Koprowski, and W. Huott. 99%- AC Test Coverage Using only LBIST on the 1-GHz IBM S/390 zseries 900 Microprocessor. In *Proceedings IEEE International Test Conference (ITC)*, pages 586–592, October 2001.

[19] H. Bardell, W.H. McAnney, and J. Savir. *Built-In Test for VLSI: Pseudorandom Techniques*. John Wiley & Sons, Chichester, 1987.

[20] R.F. Rizzolo et al. System performance management for the S/390 Parallel Enterprise Server G5 1999. *IBM Journal of Research and Development*, Vol. 43(No. 5/6):651–660, 1999.

[21] M. Müller et al. RAS Strategy for IBM S/390 G5 and G6. *IBM Journal of Research and Development*, Vol. 43(No. 5/6):875–888, 1999.

[22] B.W. Curran et al. IBM eServer z900 High Frequency Microprocessor Technology, Circuits, and Design Methodology. *IBM Journal of Research and Development*, Vol. 46(No. 4/5):631–644, 2002.

[23] L. Spainhower and T. Gregg. IBM S/390 Parallel Enterprise Server G5 Fault Tolerance: A Historical Perspective. *IBM Journal of Research and Development*, Vol. 43(No. 5/6):863–873, 1999.

[24] L.C. Alves et al. RAS Design for the IBM eServer z900. *IBM Journal of Research and Development*, Vol. 46(No. 4/5):503–521, 2002.