Debug and Diagnosis: Mastering the Life Cycle of Nano-Scale Systems on Chip

Hans-Joachim Wunderlich, Melanie Elm, Stefan Holst
Institut für Technische Informatik
Universität Stuttgart
Pfaffenwaldring 47; D-70569 Stuttgart, Germany
email: {wu, elm, holst}@iti.uni-stuttgart.de

Abstract— **Rising design complexity and shrinking structures pose new challenges for debug and diagnosis. Finding bugs and defects quickly during the whole life cycle of a product is crucial for time to market, time to volume and improved product quality. Debug of design errors and diagnosis of defects have many common aspects. In this paper we give an overview of state of the art algorithms, which tackle both tasks, and present an adaptive approach to design debug and logic diagnosis.**

Special design for diagnosis is needed to maintain visibility of internal states and diagnosability of deeply embedded cores. This article discusses current approaches to design for diagnosis to support all debug tasks from first silicon to the system level.

Keywords— **Diagnosis, Debug, Embedded Test**

1 INTRODUCTION

For the economic success of a product three factors are crucial: fast time to market, low cost per unit and high product quality. The relevance of debug and diagnosis to optimize these factors is obvious, in each phase of a product's lifecycle, defects and bugs have to be found quickly and special equipment and automatisms are necessary to achieve this goal. In the development phase of a product, debug is the essential mean to find and locate bugs, in the production and support phase the appropriate mean is diagnosis.

Debug is the process of locating logical and functional flaws in specifications, hardware and software. As logical and functional flaws remain the main cause of today's design respins, verification is turning into a critical bottleneck with increasing complexity of Systems on Chip (SoC) [1], [2]. Despite the efforts spent on advanced verification and validation techniques, the percentage of designs with functional errors has increased between the years 2002 and 2004 [3].

Diagnosis is the process of locating faults in a physical chip at the various levels down to real defects. Numerous parasitic and timing effects may show up in the first silicon [4] and have to be located and eliminated to enable a fast yield ramp up. But even after successful production diagnostic techniques have to be applied to returns to further improve the product quality and learn for future products.

Design verification and diagnosis of microelectronic circuits have long been viewed as separate tasks with individual challenges and techniques. However, in recent years more and more attention is paid to the interaction of individual design steps in verification, diagnosis in production, and field return analysis. Since diagnosis and debug have the common objective of achieving high diagnostic resolution, improving accessability of internal signals and cores helps with all aspects of verification, debug and diagnosis.

Techniques which were formerly employed for test and afterwards discovered for diagnosis - like the scan design method and test point insertion - are now reused for design validation [5], [6]. Additionally it becomes more and more obvious that yield ramping starts with design for manufacturability [7], [8], [9], thus many precautions have to be taken on the designer's side to enable debugging and diagnosis.

Taking a look at the progress of nanometer technology, designs have to be more robust due to increased variations of nano-scaled silicon. Automated maintenance and built in self-repair become strong requirements to achieve high reliability, but employing this, designs get both hard to test and hard to diagnose. Future work will have to overcome these challenges.

In this paper we give an outline of recent research challenges and developments in the area of debug and diagnosis. In the following chapter, debug and diagnosis tasks throughout the life cycle of a SoC are discussed. In chapter 3, algorithms are presented to locate faulty structures in circuits and defects on chips. In chapter 4, methods are discussed to improve accessability to internal states of devices.

2 LIFE CYCLE DEBUG AND DIAGNOSIS

Designing and manufacturing is an error prone process. Some of the errors are due to misconceptions and human mistakes, others are unavoidable due to unknown conditions. Even under the assumption of no human mistakes there is a strong necessity for quality control and improvement during the complete life cycle of the system. This section discusses tasks, challenges and requirements for tackling faults occurring during design, manufacturing and operation.

2.1 Specification

Functional errors caused by the designer are mostly due to an inconsistent or incomplete specification. After specification, we have functional simulation models for all major system components. But with growing system complexity, simulation time for the complete system grows to an extend where simulation covers only a small portion of the design space. Despite all efforts to find functional errors as early as possible, functional errors may then show up later in prototypes and during system-level debug.

Increased accessibility introduced by design for diagnosis helps tracking down functional errors during system-level debug. Though system-level debug itself is beyond the scope of this article, we spent a paragraph on trace buffers in chapter 4.

2.2 Implementation

Implementation is the design of hardware according to the available models or specification. The implementation has to be verified to avoid respectively find design-errors, which are defined as deviations of the implementation from the specification. Estimates today are that more than 70% of the total design time is spent on verification [1], [2]. Standardized debug methods and algorithms have to be developed to decrease verification and thereby design time.

Today, assertion based verification is used and supported by commercial tools. Most often, this task relies on simulation and becomes very expensive. Again, some design errors may only show up during emulation, in prototypes or even during mass production, where then an increased accessability introduced by design for diagnosis during the implemenation phase has to help tracking down design errors.

The variety of design-errors is infinite, nevertheless the most common design flaws can be made out as violated timing constraints or altered logic functions due to manual optimizations. Design debug is the task of finding those flaws in a design. Though design errors are defined relative to the specification, design debug does not depend on a fault-free specification to track down deviations.

To gain a deeper understanding of today's debug-techniques, a different view on the problem is helpful: A design fault is the logic function of the smallest circuit part which needs to be rectified. This view bridges debug and diagnosis, where fault-modeling is applied to describe possible misbehaviours of a circuit, caused by design-errors or defects.

One way to describe design errors are conditional stuck-at faults as proposed in [10]. Conditional stuck-at faults are stuck-at faults with an additional activation condition. If multiple conditional stuck-at faults are assigned to a single line or a multiple line, then an arbitrary combinational faulty behavior can be described. Figure 1 amplifies how to model a design error, where an AND gate is exchanged by an OR gate with the help of a conditional stuck-at fault.

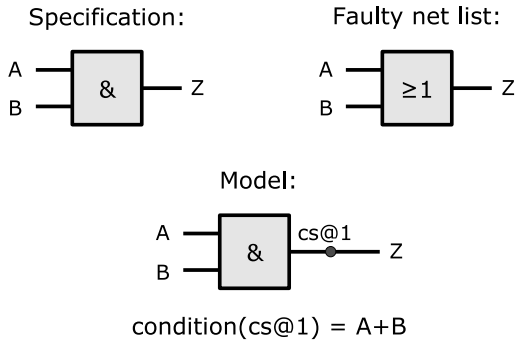

Figure 1: Example of a conditional stuck-at fault

Among all the approaches to describe malfunctioning designs by fault models, there is no fault model matching every possible design fault.

The last step in implementation is the final layout of a design. Layouts are verified by extracting a netlist and debugging the extracted netlist with the same design debug methods used before.

2.3 Prototyping

With the masks from the implementation phase, first chips are produced. Due to various unknown effects, not all of the prototypes produced will work properly. This problem is getting more severe with shrinking structures, as now systematic and random variations are increasing. In consequence, the actual behavior of physical chips gets more and more difficult to predict and simulate [11], and additionally it is harder

to decide whether a device works within a given specification or not [12].

A physical disorder which leads to a behavior different from the implementation is called a defect. Systematic defects must be identified and avoided by altering the design or the process parameters. The new behavior of the internal signals due to a present defect is called a fault. In recent technologies, the complexity and variety of possible faulty behaviors is increasing, and fault models cannot reflect reality any more.

The increasing variations in nano-scale silicon lead to complex defect mechanisms, hence the actual behavior of faulty chips and designs becomes not only difficult to predict but also difficult to model [13], [14], [15]. Rising variations for instance lead to lower signal-to-noise ratios, to complex defect mechanisms and indeterministic behavior.

Stuck-at, delay, bridging and statistical fault models are used today in commercial tools. However there are strong efforts towards a fault model independent diagnosis.

Defects can also be expressed in conditional stuck-at faults as long as they result in a combinational malfunction. Figure 2 shows an example of a bridge or short. However like in design debug, a fault model which can describe all possible defect mechanisms is unknown. Fault models must be determined by the diagnosis algorithm itself to describe the defect mechanisms. As an additional precondition for diagnostic algorithms, a high diagnostic resolution has to be provided to find exact defect mechanisms to guide the physical inspection accurately.

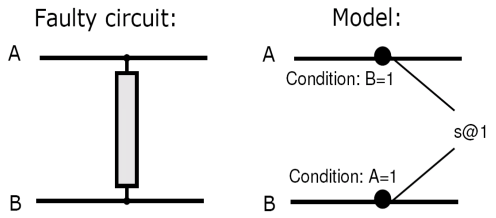

Figure 2: Example of a conditional stuck-at fault modeling a resistive bridge

On a chip, there can be faults in combinational logic, in scan chains or in the clock tree. Finding possible defect locations in random logic based on the observed behavior of the chip is called logic diagnosis. Logic diagnosis together with scan chain diagnosis and interconnect (bus, network) diagnosis forms the precision diagnosis.

2.4 Manufacturing

”Time-to-volume” and ”time-to-market” are essential for the economic success of a product. ”Yield ramping” is a traditional application area of diagnosis as it is used to find yield limiters.

Modern manufacturing processes strongly interact with the design characteristics. This necessitates yield learning for each new design. Adapting process and product requires analysis of root causes for failures and outliers [16], [17]. The extracted knowledge is used to support yield ramping and yield learning in advanced process technologies by improving design for manufacturability [16].

Prior to expensive diagnosis and physical failure analysis, spot defects must be ruled out by volume diagnosis. In volume diagnosis, test data of a large number of failing chips are recorded and analyzed to find yield-limiting systematic defects and design issues. Diagnostic data from a single chip is not sufficient since systematic problems need to be differentiated from sporadic random defects. First attempts to establish standards in volume diagnosis have been made [18].

Research in this area is quite mature, nevertheless with growing design complexity again new problems arise. Complex designs need more patterns to test and testing time is a crucial cost factor. Additionally in modern designs many cores are deeply embedded and test access is a severe problem. The test-solution developed aiming at this issue is built-in self test (BIST). BIST reduces traffic and helps cutting testing time, and many chips can be tested in parallel on one tester. However, classic BIST infrastructures may limit the visibility from outside and gathering diagnostic data may become more difficult. Often, only very limited diagnostic information is available like the number of the first failing pattern.

2.5 Support

Even after successful manufacturing, diagnostic techniques are needed to detect and locate defective modules [19] before repair. As customer satisfaction and warranties are a strong economic factor, the diagnosis-infrastructure of a silicon product facilitating diagnosis in field is also of great importance.

3 LOGIC DEBUG AND DIAGNOSIS

Logic debug and diagnosis is concerned with finding the most reasonable root causes within a random logic network that explain the failing flip-flops of this circuit as good as possible. This circuit can either be a design containing errors or a core on a chip with defects. The

only difference is that the root causes are induced by different defect or error mechanisms. The traditional way to tackle these root causes first to create simple fault and error models to cut on the complexity of the problem, and then to develop special debug and diagnosis algorithms for each of these models.

Due to rising complexity of possible defect mechanisms, new approaches are currently explored which are not restricted to a specific fault or error model. Such algorithms are based on the observation, that simple, local defect mechanisms are more reasonable root causes than complex, distributed ones. An easy metric for resonability is provided by the number of conditional stuck-at faults needed to explain all failures. This observation holds for both, design errors and physical defects. Therefore fault model independent approaches are suitable for both, design debug and logic diagnosis.

Design debug and logic diagnosis have the common goal of not only deriving possible root causes but also to keep the number of suspects as low as possible: The lower the number of returned suspects, the higher the achieved diagnostic resolution. The applied test set itself determines the achievable resolution by the number of faults which cannot be distinguished any further [20], [21], [22]. The effectiveness of pattern response analysis algorithms is evaluated by comparing the achieved diagnostic resolution to the resolution of the test set.

Pattern response analysis algorithms are divided into cause-effect and effect-cause approaches. These two fundamental paradigms will be discussed in the next two subsections. Most debug and diagnosis methods employ at least one of these approaches and some even combine pattern analysis with diagnostic ATPG to provide maximum diagnostic resolution. This concept is called adaptive diagnosis and is covered in the third subsection.

3.1 Cause-Effect Analysis

In cause-effect analysis, a fault model is chosen to enumerate all possible root causes in a circuit. Fault simulation is performed on each fault in the model, and the behavior is matched with the failing responses observed [23], [24].

To cut on simulation time, the erroneous output for each fault and each pattern is stored in a dictionary [25] but depending on the complexity of the chosen fault model and the size of the circuit, such a dictionary may explode. Significant research effort has been spent for reducing the size of fault dictionaries [26], [27]. The size can be reduced by omitting the erroneous output and storing only pass-fail information for each pair or by limiting the diagnostic resolution of the dictionary and performing fault simulation for each case to distinguish the remaining candidates [28].

Dictionary based cause-effect approaches today can handle industrial-sized designs [29] but the main drawback—the dependency on simplistic fault models like stuck-at or bridges—remain. However, some advanced methods still use cause-effect analysis as a final stage in the diagnosis process to improve diagnostic resolution.

3.2 Effect-Cause Analysis

In effect-cause analysis, possible defect locations are derived directly from the observed failing outputs by taking the logic structure of the circuits into account [30], [31]. This approach does not depend on the enumeration of all possible faults, thus it can be used to implement fault model independent diagnosis. As mentioned above, such algorithms assume a certain locality of the root cause.

The most simple effect-cause algorithms rely on the strongest locality possible: the so-called *single fault assumption* or *single fix condition*. This assumption states, that there is a single signal within the circuit which value needs to be altered to explain all failing patterns. Based on this, algorithms were proposed which are based on the intersection of input cones of failing outputs [32] or backtrace critical paths from failing outputs to focus on delay faults [33]. After finding such a signal in an erroneous design, its logic behavior can be extracted and rectified [34].

The 'Single Location At a Time' (SLAT) approach introduced by [35], [36] relaxes the single fault assumption. This approach determines for each pattern single stuck-at faults that can explain the failing response by fault simulation. Those explaining faults can be different for each failing pattern and are used to derive more complex faults. Hence SLAT is a fault model independent approach which merely uses the stuck-at fault model in fault simulation to localize the suspicious region of the circuit.

The main drawback of the SLAT paradigm is the fact that information for fault location is only extracted from patterns which fulfill the single fix condition. All the other patterns are not taken into account, neither failing nor passing ones.

To overcome this limitation, many algorithms work in two passes: First, a fast effect-cause analysis like SLAT is performed to constrain the circuits region where possible culprits may be located. Second, for each of the possible fault sites, a cause-effect simulation is performed for identifying those faults, which match the real observed behavior [23], [24].

3.3 Adaptive Diagnosis

There is no concise test set which provides the best resolution for every possible faulty behavior. Since the maximum achievable diagnostic resolution is determined by the test set, many approaches already employ diagnostic or focused ATPG to distinguish remaining suspects or extracting defective behavior completely [23], [34].

By integrating pattern generation more tightly into the whole diagnosis process, fault location can be even more powerful. This general idea of alternating pattern analysis and pattern generation steps is called adaptive diagnosis [37].

Here, faulty and fault free responses are used in order to guide the automatic generation of new patterns for increasing the resolution. A pattern analysis step extracts information from responses of the DUD and accumulates them in a knowledge base. This knowledge in turn guides an automatic test pattern generator (ATPG) to generate relevant patterns for achieving high diagnostic resolution. The loop ends, when an acceptable diagnostic resolution is reached (Fig. 3). The definition of the exact abort criterion depends on the number and confidence levels of fault candidates.

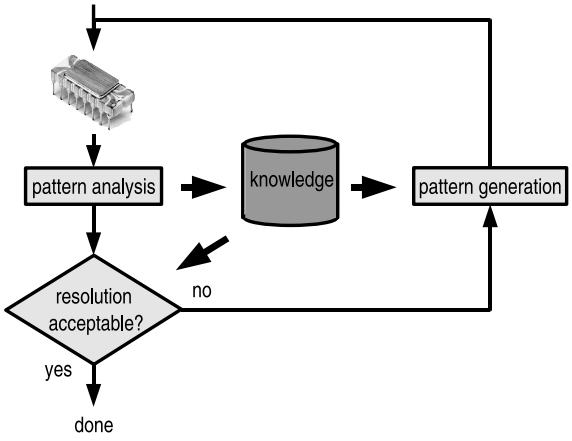

Figure 3: Adaptive diagnosis flow

One way to implement such an adaptive diagnosis flow is by the generalization of the SLAT paradigm. Where SLAT only considers perfect matches for each pattern, a measure can be defined to quantify how well a stuck-at signal explains a response of the circuit under diagnosis [38]. Let FM(f) be a fault machine, i.e. the circuit with stuck-at fault f injected. For each test pattern $t \in T$, the evidence

$$e(f,t) = (\Delta\sigma_t, \Delta\iota_t, \Delta\tau_t, \Delta\gamma_t)$$

is defined as as tuple of numbers where $\Delta\sigma_t$ is the number of failing outputs f can explain, $\Delta\iota_t$ is the number of additional failures f induces, $\Delta\tau_t$ is the number of failing outputs not explained by f (see fig. 4), and $\Delta\gamma_t$ is the minimum of $\Delta\sigma_t$ and $\Delta\iota_t$.

A failing pattern t which is completely explained by a stuck-at signal f will lead to evidence $e(f,t) = (\Delta\sigma_t > 0, 0, 0, 0)$. So the SLAT approach is only a special case in this notation. Note, that $\Delta\sigma_t$ will be maximum for all evidences of t.

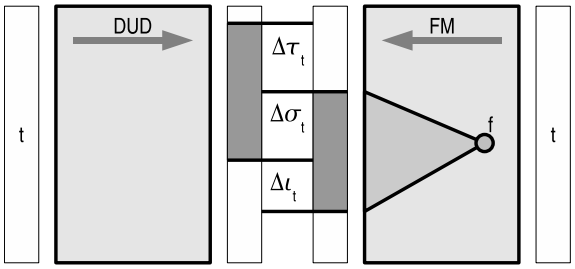

Figure 4: Definition of evidence $e(f,t) = (\Delta\sigma_t, \Delta\iota_t, \Delta\tau_t, \Delta\gamma_t)$

The evidence of a fault f and a test set T is simply the sum of all evidences for f:

$$e(f,T) = (\sigma_T, \iota_T, \tau_T, \gamma_T), \text{ with}$$
$$\sigma_T = \sum_{t\in T} \Delta\sigma_t,\ \iota_T = \sum_{t\in T} \Delta\iota_t,$$
$$\tau_T = \sum_{t\in T} \Delta\tau_t \text{ and } \gamma_T = \sum_{t\in T} \Delta\gamma_t.$$

If $\Delta\sigma_t$ was maximum for a stuck-at signal f and each $t \in T$, σ_t is also maximum. In addition, a candidate is more suspicious if it causes less additional failures in places where the observed response shows the correct values. So the ranking is derived by sorting evidences first by σ_T and then by ι_T. Table 1 provides an example of such a ranking.

This ranking shows, that f_1 is the only stuck-at signal, which can explain every observed failure and induces no additional ones. Hence, all pattern responses analyzed so far can be explained by this single stuck-at fault in the circuit under diagnosis. The failures of stuck-at signal f_2 is a proper superset of the observed failures because σ_T is maximum and ι_T is positive. Moreover since γ_T is 0 for this stuck-at signal, f_2 explains all failing pattern $t \in T_f \subset T$ completely ($\Delta\sigma_t$ maximum, $\Delta\iota_t = 0$) but not every passing pattern ($\Delta\sigma_t = \Delta\tau_t = 0$ and $\Delta\iota_t > 0$ for some $t \in T_p \subset T$). This leads to the only conclusion, that f_2 can explain all the responses as a *conditional stuck-at fault*.

Table 2 shows suspect evidences for some classic models. If ι_T, τ_T and γ_T are all zero, a single stuck-at fault explains the DUD behavior completely. With

stuck-at sig.	σ_T	ι_T	τ_T	γ_T
f_1	42	0	0	0
f_2	42	35	0	0
f_3	42	35	0	0
f_4	42	35	0	0
f_5	42	38	0	0
f_6	23	22	19	0
f_7	23	23	19	0

Table 1: A ranking with f_1 as the best candidate.

$\iota_T = \gamma_T = 0$, such a stuck-at fault explains a subset of all fails, but some other faulty behavior is present in the DUD. If τ_T and γ_T are zero, a faulty value on a single signal line under some patterns $T' \subset T$ provides complete explanation. With only $\gamma_T = 0$, a faulty value on the corresponding single signal line explains only a part of DUD behavior. If only τ_T is zero, the suspect fails are a superset of DUD fails. If all suspects show positive values in all components $\iota_T, \tau_T, \gamma_T$, all simplistic fault models would fail to explain the DUD behavior.

classic model	ι_T	τ_T	γ_T
single stuck-at	0	0	0
stuck-at, multiple fault sites	0	> 0	0
single conditional stuck-at	> 0	0	0
cond. stuck-at, multiple fault sites	> 0	> 0	0
delay fault, i.e. long paths fail	> 0	0	> 0

Table 2: Fault models and evidence forms for $e(f, T)$ with $\sigma_T > 0$

By simple iteration over the ranking, pairs of suspects f^a, f^b are identified with equal evidences $e(f^a, T) = e(f^b, T)$. In table 1, the stuck-at signals f_2, f_3 and f_4 are not distinguished yet. To improve the ranking, fault distinguishing patterns are generated [20], [21] and applied to the circuit. During analysis of these responses, different values will be added to the evidences under consideration and the ranking will improve. However, this may also introduce other sets of equal evidences, the approach iterates until all remaining pairs of equal evidences can not be distinguised by diagnostic ATPG. To reduce the number of suspects and the region under consideration further, diagnostic pattern generation algorithms have to be employed which exploit layout data [23].

This generalization of SLAT provides a consistent, single-pass adaptive diagnosis algorithm which extracts evidence from every pattern and the diagnostic results are very encouraging [38]. The consideration of every pattern is important especially if there is only limited failure information available.

4 DESIGN FOR DEBUG AND DIAGNOSIS

Diagnostic capabilities are needed during the whole life cycle of a system [9]. Besides the techniques and algorithms to find the actual locations of faults or defects there is the need to provide access to the internal states of a device and to record and evaluate diagnostic data. The fulfillment of this tasks is done by Design for Debug and Diagnosis (DDD).

Two major problems have to be overcome by DDD:

1) Diagnostic data may reach an enormous bandwidth, due to the requirement of high diagnostic resolution.
2) The diagnosis or debug of so-called hard-to detect faults requires long observation periods.

Generally the problem complexity can be broken down by the same means as applied for design-for-testability: scan-design to provide access to internal states, compression and compaction to reduce the data-volume. To guarantee the correctness of a scanned out diagnosis response, the diagnostic equipement has to be fault-free. The diagnosis of shift-registers respectively scan-chains is nowadays quite mature [39], [40], [41], [42].

Figure 5 shows the embedded test equipement reused for diagnosis and a schematic of volume diagnosis reusing the multi-site test structure.

Compaction of diagnosis responses on the other hand is and will be a major research topic.

4.1 Compaction Techniques

Special precautions are necessary to gain more valuable information while keeping the traffic as low as possible. As detailed knowledge on the diagnosis responses is not available, compaction techniques have to be applied to reduce the amount of necessary tester channels on the outputs of the circuit.

Compactors can be classified due to different properties. The simplest classification separates time- and space compactors. Space compactors reduce the amount of output channels of a circuit by employing parity trees. Thus space compactors preserve the length of a test response. Time compactors reduce the length of a response vector by compaction of several shift-out cycles and employing memory cells. Combinations of time- and space compactors consisting of a space compaction stage and attached to this a time compaction stage can also be employed to reduce both, response length and width.

Compaction may discard valuable information for diagnosis and may reduce diagnostic resolution. Unknown values in the response vectors, caused by buses

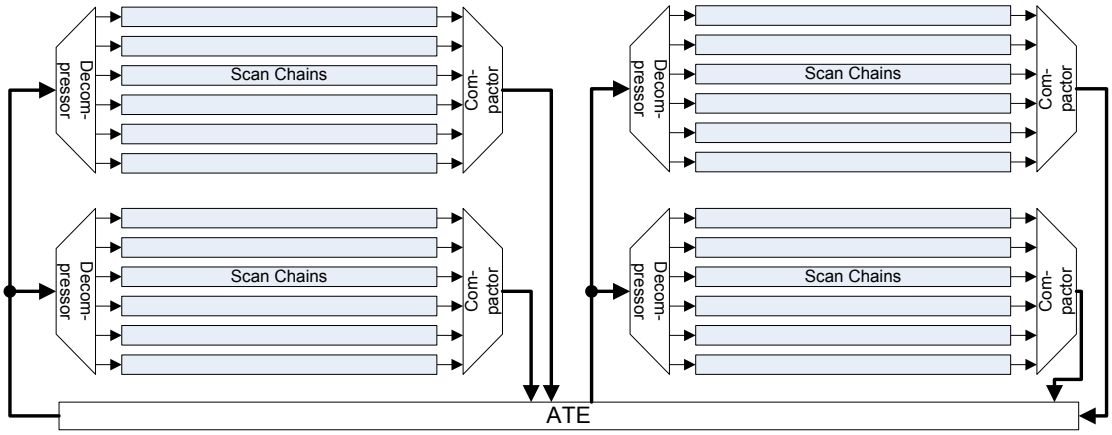

Figure 5: Multi-Site Test

or uninitialized logic, may additionally cause fault-cancellation and -aliasing after compaction.

Special compactors were proposed to preserve diagnostic resolution and capability of X-tolerance. Parity check matrices of error correcting codes were employed to construct space compactors able to tolerate a certain amount of X-states and able to detect and locate a certain amount of errors. The first approach implementing this was proposed in [43]. Nowadays a large variety of extensions to this approach and similar approaches is available. A popular representative is the X-Compact compactor proposed in [44].

Besides the pure employment of coding theory, the interaction of compactor design with the ATPG was proposed. I-Compact for instance [45] first employs coding techniques to gain X-tolerance and further enhances X-tolerance by storing all possible X-positions in addition to the fault-free response vectors calculated during ATPG. By reusing the parity check matrix this can be done very space-efficiently.

A different approach uses the ATPG to determine scan-chains, which have to be switched off by a selection logic on scan-out. This can be used on the one hand to enable error propagation for any error [46] and in more recent work to allow for X-tolerant compaction [47], [48].

The different compactor designs are already integrated in commercial tools. Despite all the enhancements in compaction the problem of error-masking is not solved completely and will increase with growing circuits. Application of coding theory and the interaction with ATPG will reach its limits with growing cicuits and the demand of fault-model independent or adaptive diagnosis.

4.2 Trace Buffers

Contrary to the scan-design method in prototypes, for system-level debug and silicon debug (fault location before destructive probing) a different approach is often applied. Trace buffers are an on-chip instrumentation supporting at-speed sampling and a low bandwidth connection to external debug software which for instance uses a JTAG interface [49].

This approach was influenced by software debugging used in embedded systems [50]. Trace buffers can be classified in special purpose trace buffers designed for a special architecture—e.g. [51]—or generic trace buffers applicable to any SoC [52].

In contrast to scan-chains trace buffers monitor only a subset of internal signals. They are implemented on chip using the available memory to store failing pattern responses. This is area efficient on the one hand, but affects diagnostic respectively debug capabilities on the other hand as the buffer's size limits the observation window. In consequence the window might be too small to locate faults manifesting themselves only after a long execution time.

One of the most recent approaches to overcome this problem was proposed in [53]. In cases where the debug experiment can be repeated a cyclic debugging used to zoom into the interesting intervals is employed.

5 CONCLUSION

Today's challenges in diagnosis and debug can be seen in two different areas. First shrinking structures may cause unpredictable circuit behavior. This fact requires diagnosis algorithms and test pattern generation independent of an underlying fault model to enable reliable test and diagnosis. In this paper an overview of the existing methods meeting these requirements was presented.

Second the growing complexity of circuits makes access to and transfer of internal states for debug and diagnosis more complicated. Basically, test equipment like scan chains and compactors can be reused to overcome this problem, but special care has to be taken to keep a high diagnostic resolution. Here we gave an overview of compaction approaches and debug facilities aiming at this issue.

6 ACKNOWLEDGEMENT

This work has been funded by the DFG under contract WU 245/4-1.

References

[1] K. C. Chen, "Assertion-based verification for SoC designs," in *Proceedings 5th International Conference on ASIC, Vol. 1*, 2003, pp. 12–15.

[2] R. Klein and T. Piekarz, "Accelerating functional simulation for processor based designs," *Mentor Graphics Corporation, white paper*, 2005.

[3] T. Fitzpatrick, "Realizing advanced functional verification with questa," *Mentor Graphics Corporation, white paper*, 2005.

[4] K. Roy, T. M. Mak, and K.-T. T. Cheng, "Test consideration for nanometer-scale cmos circuits." *IEEE Design & Test of Computers*, vol. 23, no. 2, pp. 128–136, 2006.

[5] K.-T. Cheng, S.-Y. Huang, and W.-J. Dai, "Fault emulation: A new methodology for fault grading." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 18, no. 10, pp. 1487–1495, 1999.

[6] R. Ludewig, T. Hollstein, F. Schütz, and M. Glesner, "Architecture for testing and debugging of system-on-chip components." in *IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems (DDECS) 2004, Stará Lesná, Slovakia, April 18-21, 2004*, 2004, pp. 91–98.

[7] M. Riley, N. Chelstrom, M. Genden, and S. Sawamura, "Debug of the CELL processor: Moving the lab into silicon," in *Proceedings IEEE International Test Conference 2006, Santa Clara, CA, USA, October 24-26, 2006*, 2006, p. 26.1.

[8] T. Arnaout, G. Bartsch, and H.-J. Wunderlich, "Some common aspects of design validation, debug and diagnosis," in *Third IEEE International Workshop on Electronic Design, Test and Applications (DELTA 2006), 17-19 January 2006, Kuala Lumpur, Malaysia*, 2006, pp. 3–10.

[9] H.-J. Wunderlich, "From embedded test to embedded diagnosis," in *Proceedings European Test Symposium, Tallin, Estonia*, 2005, pp. 216–221.

[10] O. E. Cornelia and V. K. Agarwal, *Conditional Stuck-at Fault Model for PLA Test Generation.* VLSI Design Laboratory, McGill University, 1989.

[11] J. W. McPherson, "Reliability challenges for 45nm and beyond." in *Proceedings of the 43rd Design Automation Conference, DAC 2006, San Francisco, CA, USA, July 24-28, 2006*, 2006, pp. 176–181.

[12] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V. De, "Parameter variations and impact on circuits and microarchitecture." in *Proceedings of Design Automation Conference 2003, Anaheim, USA, June 2-6 2003*, pp. 338–342.

[13] A. Krstic, L.-C. Wang, K.-T. Cheng, J.-J. Liou, and M. S. Abadir, "Delay defect diagnosis based upon statistical timing models - the first step." in *2003 Design, Automation and Test in Europe Conference and Exposition (DATE 2003), 3-7 March 2003, Munich, Germany*, 2003, pp. 10 328–10 335.

[14] C. L. Henderson and J. M. Soden, "Signature analysis for ic diagnosis and failure analysis." in *Proceedings IEEE International Test Conference 1997, Washington, DC, USA, November 3-5, 1997*, 1997, pp. 310–318.

[15] D. B. Lavo, B. Chess, T. Larrabee, and I. Hartanto, "Probabilistic mixed-model fault diagnosis." in *Proceedings IEEE International Test Conference 1998, Washington, DC, USA, October 18-22, 1998*, 1998, pp. 1084–1093.

[16] C. Hora, R. Segers, S. Eichenberger, and M. Lousberg, "An effective diagnosis method to support yield improvement." in *Proceedings IEEE International Test Conference 2002, Baltimore, MD, USA, October 7-10, 2002*, 2002, pp. 260–269.

[17] ——, "On a statistical fault diagnosis approach enabling fast yield ramp-up." in *IEEE European Test Workshop 2002, Corfu, Greece, May 26-29, 2002*, 2002.

[18] A. Leininger, A. Khoche, M. Fischer, N. Tamarapalli, W.-T. Cheng, R. Klingenberg, and W. Yang, "The next step in volume scan diagnosis: Standard fail data format." in *Asian Test Symposium 2006, Nov. 2006, Fukuoka, Japan*, pp. 360–368.

[19] M. Abramovici, J. Emmert, and C. Stroud, "Roving stars: an integrated approach to on-line testing, diagnosis, and fault tolerance for fpgas in adaptive computing systems." in *Proceedings. The Third NASA/DoD Workshop on Evolvable Hardware, July 12-14 2001, Long Beach, CA, USA*, 2001, pp. 73–92.

[20] A. G. Veneris, R. Chang, M. S. Abadir, and M. Amiri, "Fault equivalence and diagnostic test generation using atpg." in *Proceedings IEEE International Symposium on Circuits and Systems, 2004*, 2004, pp. 221–224.

[21] T. Bartenstein, "Fault distinguishing pattern generation." in *Proceedings IEEE International Test Conference 2000, Atlantic City, NJ, USA, October 2000*, 2000, pp. 820–828.

[22] N. K. Bhatti and R. S. Blanton, "Diagnostic test generation for arbitrary faults." in *Proceedings IEEE International Test Conference 2006, Santa Clara, CA, USA, October 24-26, 2006*, 2006, p. 19.2.

[23] R. Desineni, O. Poku, and R. D. S. Blanton, "A logic diagnosis methodology for improved localization and extraction of accurate defect behavior," in *Proceedings IEEE International Test Conference 2006, Santa Clara, CA, USA, October 24-26, 2006*, 2006, p. 12.3.

[24] M. E. Amyeen, D. Nayak, and S. Venkataraman, "Improving precision using mixed-level fault diagnosis," in *Proceedings IEEE International Test Conference 2006, Santa Clara, CA, USA, October 24-26, 2006*, 2006, p. 22.3.

[25] I. Pomeranz and S. M. Reddy, "On the generation of small dictionaries for fault location," in *IEEE/ACM International Conference on Computer-Aided Design, ICCAD92, November 8-12, 1992, Santa Clara, CA, USA*, 1992, pp. 272–279.

[26] V. Boppana, I. Hartanto, and W. K. Fuchs, "Full fault dictionary storage based on labeled tree encoding," in *14th IEEE VLSI Test Symposium (VTS'96), April 28 - May 1, 1996, Princeton, NJ, USA*, 1996, pp. 174–179.

[27] B. Chess and T. Larrabee, "Creating small fault dictionaries,"

IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 18, no. 3, pp. 346–356, Mar 1999.

[28] P. G. Ryan, S. Rawat, and W. K. Fuchs, "Two-stage fault location." in *Proceedings IEEE International Test Conference 1991, Test: Faster, Better, Sooner, Nashville, TN, USA, October 26-30, 1991*, 1991, pp. 963–968.

[29] "Faloc reference manual 3.9," *NXP Semiconductors*, 2004.

[30] M. Abramovici and M. A. Breuer, "Fault diagnosis based on effect-cause analysis: An introduction," in *17th Conference on Design Automation, June 1980*, 1980, pp. 69–76.

[31] J. A. Waicukauski and E. Lindbloom, "Failure diagnosis of structured VLSI," *IEEE Design & Test of Computers*, vol. 6, no. 4, pp. 49–60, Aug 1989.

[32] S. Venkataraman and S. B. Drummonds, "Poirot: a logic fault diagnosis tool and its applications." in *Proceedings IEEE International Test Conference 2000, Atlantic City, NJ, USA, October 2000*, 2000, pp. 253–262.

[33] A. Rousset, A. Bosio, P. Girard, C. Landrault, S. Pravossoudovitch, and A. Virazel, "Derric: A tool for unified logic diagnosis." in *12th European Test Symposium (ETS 2007), 20 May 2007, Freiburg, Germany*, 2007, pp. 13–20.

[34] R. Ubar, "Design error diagnosis with resynthesis in combinational circuits," *Journal of Electronic Testing: Theory and Applications*, vol. 19, pp. 73–82, 2003.

[35] T. Bartenstein, D. Heaberlin, L. M. Huisman, and D. Sliwinski, "Diagnosing combinational logic designs using the single location at-a-time (SLAT) paradigm." in *Proceedings IEEE International Test Conference 2001, Baltimore, MD, USA, 30 October - 1 November 2001*, 2001, pp. 287–296.

[36] L. M. Huisman, "Diagnosing arbitrary defects in logic designs using single location at a time (SLAT)," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, no. 1, pp. 91–101, January 2004.

[37] Y. Gong and S. Chakravarty, "On adaptive diagnostic test generation," in *Proceedings IEEE International Conference on Computer-Aided Design, November 1995*, 1995, p. 181.

[38] S. Holst and H.-J. Wunderlich, "Adaptive debug and diagnosis without fault dictionaries." in *12th European Test Symposium (ETS 2007), 20 May 2007, Freiburg, Germany*, 2007, pp. 7–12.

[39] J. L. Schafer, F. A. Policastri, and R. J. McNulty, "Partner srls for improved shift register diagnostics." in *Digest of Papers. 10th Anniversary IEEE VLSI Test Symposium, 7-9 April 1992, Atlantic City, NJ, USA*, 1992, pp. 198–201.

[40] S. Kundu, "On diagnosis of faults in a scan-chain." in *Digest of Papers., Eleventh Annual IEEE VLSI Test Symposium, 6-8 April 1993, Atlantic City, NJ, USA*, 1993, pp. 303–308.

[41] J. C.-M. Li, "Diagnosis of multiple hold-time and setup-time faults in scan chains." *IEEE Transactions on Computers*, vol. 54, no. 11, pp. 1467–1472, 2005.

[42] J.-S. Yang and S.-Y. Huang, "Quick scan chain diagnosis using signal profiling." in *Proceedings IEEE International Conference on Computer Design: VLSI in Computers and Processors, ICCD 2005, 2-5 Oct. 2005*, 2005, pp. 157–160.

[43] K. K. Saluja and M. Karpovsky, "Testing computer hard-ware through data compression in space and time." in *Proceedings IEEE International Test Conference (ITC), 1983*, 1983, p. 8389.

[44] S. Mitra and K. S. Kim, "X-compact: an efficient response compaction technique for test cost reduction." in *Proceedings on International Test Conference, 7-10 Oct. 2002*, 2002, pp. 311–320.

[45] J. Patel, S. Lumetta, and S. Reddy, "Application of saluja-karpovsky compactors to test responses with many unknowns." in *Proceedings on 21st VLSI Test Symposium, 27 April-1 May 2003*, 2003, pp. 107–112.

[46] A. Morosov, K. Chakrabarty, M. Gossel, and B. Bhattacharya, "Design of parameterizable error-propagating space compactors for response observation." in *Proceedings on 19th IEEE VLSI Test Symposium, VTS 2001, 29 April-3 May 2001, Marina Del Rey, CA, USA*, 2001, pp. 48–53.

[47] T. Clouqueur, H. Fujiwara, and K. Saluja, "A class of linear space compactors for enhanced diagnostic." 2005, pp. 260–265.

[48] W.-T. Cheng, M. Kassab, G. Mrugalski, N. Mukherjee, J. Rajski, and J. Tyszer, "X-press compactor for 1000x reduction of test data." in *IEEE International Test Conference, ITC '06, Oct. 2006, Santa Clara, CA, USA*, 2006, pp. 1–10.

[49] "1149.1-1990, IEEE standard test access port and boundary - scan architecture." IEEE Computer Society, 1990.

[50] C. MacNamee and D. Heffernan, "Emerging on-ship debugging techniques for real-time embedded systems." *Computing & Control Engineering Journal*, vol. 11, no. 6, pp. 295–303, 2000.

[51] H. Vranken, "Debug facilities in the trimedia cpu64 architecture." in *Proceedings of European Test Workshop 1999, 25-28 May 1999, Constance, Germany*, 1999, pp. 76–81.

[52] A. Hopkins and K. McDonald-Maier, "Debug support strategy for systems-on-chips with multiple processor cores." *IEEE Transactions on Computers*, vol. 55, no. 2, pp. 174–184, 2006.

[53] E. Anis and N. Nicolici, "Low cost debug architecture using lossy compression for silicon debug." in *Proceedings 2007 Design, Automation and Test in Europe (DATE '07), 16-20 April 2007, Nice, France*, 2007, pp. 225–230.