J Electron Test
DOI 10.1007/s10836-009-5109-3

Adaptive Debug and Diagnosis Without Fault Dictionaries

Stefan Holst · Hans-Joachim Wunderlich

Received: 23 May 2008 / Accepted: 15 July 2009

Abstract Diagnosis is essential in modern chip production to increase yield, and debug constitutes a major part in the pre-silicon development process. For recent process technologies, defect mechanisms are increasingly complex, and continuous efforts are made to model these defects by using sophisticated fault models. Traditional static approaches for debug and diagnosis with a simplified fault model are more and more limited. In this paper, a method is presented, which identifies possible faulty regions in a combinational circuit, based on its input/output behavior and independent of a fault model. The new adaptive, statistical approach is named POINTER for 'Partially Overlapping Impact couNTER' and combines a flexible and powerful effect-cause pattern analysis algorithm with high-resolution ATPG. We show the effectiveness of the approach through experiments with benchmark and industrial circuits. In addition, even without additional patterns this analysis method provides good resolution for volume diagnosis, too.

Keywords Diagnosis · Debug · Test · VLSI

Responsible Editor: C. Landrault

S. Holst (✉) · H.-J. Wunderlich
Institut für Technische Informatik, Universität Stuttgart,
Pfaffenwaldring 47, 70569 Stuttgart, Germany
e-mail: holst@informatik.uni-stuttgart.de

H.-J. Wunderlich
e-mail: wu@informatik.uni-stuttgart.de

1 Introduction

1.1 Debug and Diagnosis

Traditionally, design, verification and diagnosis of microelectronic circuits have been viewed as separate tasks with individual challenges and techniques. However, in recent years more and more attention has been paid to the interaction of individual design steps in verification, diagnosis of prototypes, and field return analysis. These are tasks for quality control and improvement during the complete lifecycle of the system by tackling faults occurring during design, manufacturing and operation.

Debug is the time-consuming task of identifying faulty modules and structures within the design. While some methods of formal verification are constructive and able to find the cause of malfunctions, simulation and emulation usually require additional efforts for fault location.

As Systems on Chip (SoC) design complexity increases, verification is turning into a critical bottleneck in the design process. Estimates today are that more than 70% of the total design time is on verification [9, 20]. Despite the efforts spent by the academia and the industry on developing functional verification tools, logical and functional flaws remain the main cause of today's design respins. Between the years 2002 and 2004, the percentage of designs with functional errors has actually increased [14].

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 [28], identifying them is part of silicon debug. With growing circuit complexity and shrinking

Published online: 29 July 2009

geometries, the actual behavior of the silicon is hard to model [16, 21, 22] and cannot always be predicted and simulated [23].

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. The extracted knowledge is used to support yield ramping and yield learning in advanced process technologies by improving design for manufacturability [18].

Precision diagnosis is performed on a small selected set of chips like first silicon or representatives for systematic defects determined by volume diagnosis to find the exact defect mechanisms in the individual chips. The constraints on computing time are reduced but high diagnostic resolution has to be provided to guide the physical inspection accurately.

Diagnosis is more related to defects and debug is closer to design errors, i. e. errors of the designer. However, there is a large overlap in dealing with yield ramping and design for manufacturability [3, 27, 32]. Diagnosis and debug have the common objective of achieving high diagnostic resolution and especially fault model independent approaches like the one discussed in this article are suitable for both of these tasks.

1.2 Effect-cause vs. Cause-effect Analysis

The classic diagnosis algorithms follow two different paradigms: *Effect-cause* analysis looks at the failing outputs and starts reasoning using the logic structure of the circuits [1, 31]. *Cause-effect* analysis is based on a fault model. For each fault of the model, fault simulation is performed, and the behavior is matched with the outcome of the device under diagnosis (DUD).

Standard debug and diagnosis algorithms usually work in two passes: First, a fast effect-cause analysis 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 [2, 12]. The *resolution* of a test set corresponds to the number of faults which cannot be distinguished any further [4, 6, 30].

The main drawback of the cause-effect paradigms is the dependency on a fault model. A general model of design errors is not available, the proposed models so far reflect only a small subset of possible design faults to be found during debug. During diagnosis, we are faced with a plethora of defect mechanisms in nanoscaled CMOS. Many diagnosis approaches focus on special fault models like stuck-at faults or bridges [11, 24], however the main goal of fault diagnosis is rather finding an appropriate fault model than taking a given fault model as an initial assumption [7, 12]. The POINTER approach discussed here will therefore follow the effect-cause paradigm to achieve fault model independence.

1.3 Fault Dictionaries vs. Adaptive Diagnosis

Cause-effect diagnosis can be speeded up, if for each fault and each failing pattern the erroneous output is determined by simulation and then stored in a dictionary [25]. Even after an effect-cause pass, the size of such a dictionary may explode, and significant research effort has been spent on reducing the size of fault dictionaries [8, 10].

During debug and during diagnosis of first silicon, there exists an efficient alternative to precomputed fault dictionaries in so-called adaptive diagnosis [15]. Here, we use faulty and fault free responses of the device under diagnosis (DUD) 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. Such a diagnostic ATPG does not rely on a precomputed fault dictionary, and significant memory savings are obtained. The loop ends, when an acceptable diagnostic resolution is reached (Fig. 1). The definition of the exact abort criterion depends on the number and confidence levels of fault candidates.

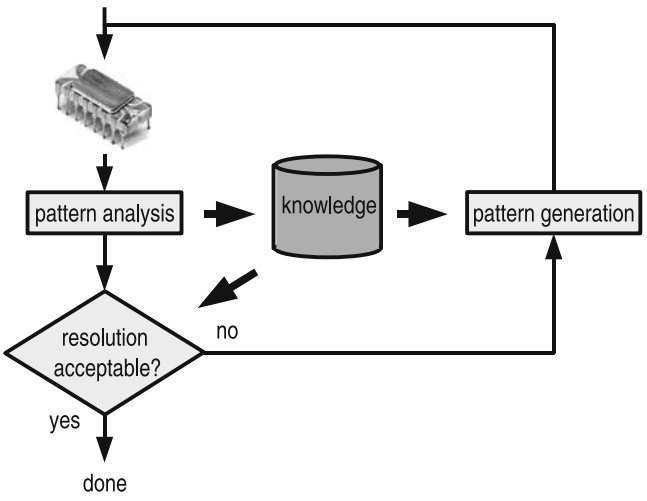

Fig. 1 Adaptive diagnosis flow

In the subsequent sections we present the POINTER approach [17]. Based on the input/output behavior of the DUD, a small region of the design is identified which behaves in a faulty way. Only very few approaches are known with the same goals, e.g. [5–7, 19, 29]. POINTER can be applied to adaptive precision diagnosis as well as to volume diagnosis.

The next section of this paper introduces a calculus to model the faulty behavior of a circuit without preassuming a fault model. An analysis algorithm which identifies possible faulty or defect regions based on a given test set is presented in Section 3. Section 4 adds a pattern generation algorithm for increasing the resolution, and Section 5 validates the approach experimentally by locating faults of different surrogate models, for both adaptive and volume diagnosis.

2 Fault Model Independence

POINTER is an extension of the 'Single Location At a Time' (SLAT) technique introduced by [5, 19]. A diagnostic test pattern has the SLAT property, if there is at least one stuck-at fault which produces a response on that pattern which is identical with the response of the DUD. In general, not all patterns will have the SLAT property, and not all the SLAT patterns will point to the same stuck-at faults. Hence, SLAT is not based on the stuck-at fault model, but uses a set of identified stuck-at lines for describing the suspicious region of the DUD.

Traditionally, fault models have been introduced to reduce the complexity of pattern generation and pattern analysis algorithms. This reduction has mainly been achieved due to both the limited amount of faults which have to be considered and their simple behavior like stuck-at or transition faults. Over the years, numerous specialized fault models were introduced to account for more and more complex defect mechanisms [21, 22]. During diagnosis however, defect mechanisms are not known in advance but target of the investigation, and specialized models are of limited use. State-of-the-art diagnosis is mainly concerned with the extraction of topological information about errors and defects based on functional failure information in order to direct the more detailed and complex physical failure analysis (PFA). The quality of a diagnostic result corresponds to the number of PFA attempts until the physical defect is finally discovered.

There is a clear distinction between the topological and the functional part of a defect description. The topological information can be expressed directly in terms of signals, gates and pins within a circuit, independently of any functional behavior of the defect. The functional part is the faulty behavior of the suspected internal signals.

A one-to-one correspondence between a defect mechanism and a representative is not obtainable, and a calculus must therefore allow for a metric of plausibility. This metric provides a ranking of defect candidates from the most reasonable ones down to uncommon yet possible ones. One metric proven very effective is based on the locality of a defect. The fewer faulty signals have to be assumed at the logic level, the more reasonable is the explanation of the real defect behavior.

Such a metric can be established by the *conditional stuck-at line* calculus already used for many years and in many variants. A conditional stuck-at line consists of a signal line (the topological part) and an activation condition (the functional part). Figure 2 shows a design error where an AND gate is exchanged by an OR gate expressed by using the conditional stuck-at calculus. An arrow represents a conditional stuck-at, its orientation indicates the polarity, and its annotation describes the activation condition.

Some more general cases are shown in Fig. 3. For the crosstalk bridge, the activation condition also depends on previous signal values as the victim line B will show a logic 1, if there was a rising transition on aggressor line A. Multiple conditional stuck-at lines are necessary for modeling more complex defect behavior.

Conditional stuck-at lines are behaving like traditional stuck-at faults but they are active only for a subset of the test patterns. This subset is defined by the activation condition which may be expressed as a boolean function, may depend on timing or involve environmental conditions. In many approaches the conditions are assumed to be deterministic [6] or expressable as a boolean function [29]. In recent technologies, however, such assumptions become more and more restrictive since defect behavior is often indeterministic or timing related. If indeterminism is taken into account

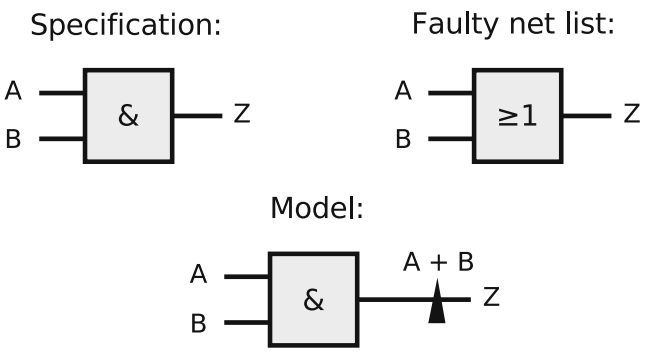

Fig. 2 Example of a conditional stuck-at line

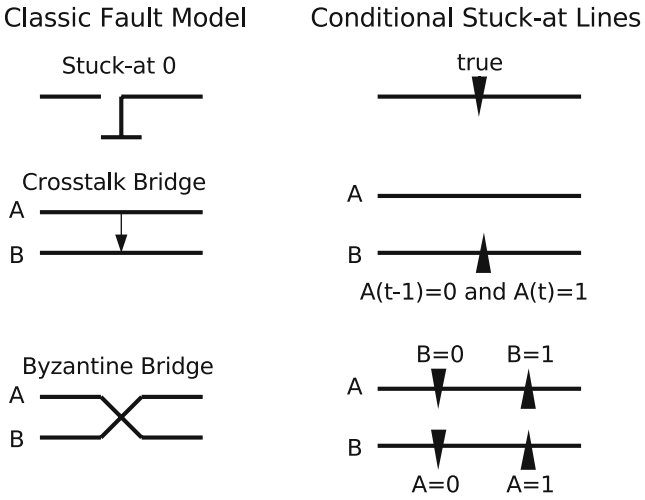

Fig. 3 Traditional fault models and conditional stuck-at lines

for the conditions, all technology related faults and design related errors can be expressed at least by multiple conditional stuck-at lines.

At the first glance, the explanations with the minimum number of conditional stuck-at lines are the most reasonable ones, but there is the risk of aliasing as demonstrated in Fig. 4.

There are two stuck-at-0 faults at the inputs of an OR-gate. Given an exhaustive test set, an explanation with a minimal number of conditional stuck-at lines would diagnose the output of the OR-gate as the only candidate. However, the correct diagnosis consists of two faults at the inputs of this gate. Note, that the conditional stuck-at line at the output is only active if $C = 0$. The conditional stuck-at lines in the correct, but more complex, explanation however are always active. The correct diagnostic result can only be achieved by considering passing patterns and by using the frequency of activation as an additional criterion. An efficient way to do so is described below.

3 Pattern Analysis

In this section, we define a measure to quantify how well a stuck-at fault reflects the behavior of the DUD

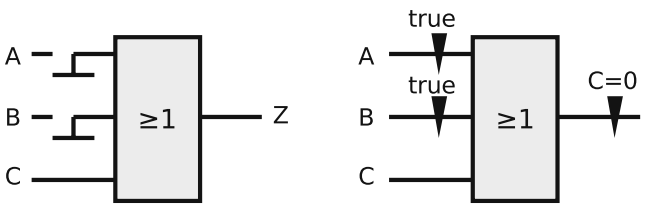

Fig. 4 Aliasing between possible explanations at the inputs and the output of an OR-gate

for a given test set. The SLAT paradigm will be just the special case of a perfect match for one pattern. Let FM(f) be a fault machine, i.e. the circuit with stuck-at fault f injected. For each test pattern $t \in T$, we define the evidence

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

as tuple of natural numbers $\Delta\sigma_t, \Delta\iota_t, \Delta\tau_t, \Delta\gamma_t \in \mathbb{N}$ (see Fig. 5) where:

- $\Delta\sigma_t$ is the number of failing outputs where both the DUD and the fault machine FM match.
- $\Delta\iota_t$ is the number of outputs which fail in FM but are correct in DUD.
- $\Delta\tau_t$ is the number of outputs which fail in DUD but are correct in FM.
- $\Delta\gamma_t$ is the minimum of $\Delta\sigma_t$ and $\Delta\iota_t$: $\Delta\gamma_t = \min\{\Delta\sigma_t, \Delta\iota_t\}$.

For a SLAT test pattern t, the evidence will provide maximum $\Delta\sigma_t$ and $\Delta\iota_t = \Delta\tau_t = \Delta\gamma_t = 0$.

The evidence of a fault f and a test set T is

$e(f, T) = (\sigma_T, \iota_T, \tau_T, \gamma_T)$, with

$$\sigma_T = \sum_{t\in T} \Delta\sigma_t,\ \iota_T = \sum_{t\in T} \Delta\iota_t,$$

$$\tau_T = \sum_{t\in T} \Delta\tau_t \text{ and } \gamma_T = \sum_{t\in T} \Delta\gamma_t.$$

Again, if the real culprit is the stuck-at fault f indeed, we get $\iota_T = \tau_T = \gamma_T = 0$ and σ_T will be maximum.

While processing pattern after pattern $t_1, ..., t_i$, the knowledge base is constructed by the evidences $e(f, T_i)$, $T_i = \{t_1, ..., t_i\}$ of all the stuck-at faults f. If a fault is not observable under a certain pattern, no value change takes place and this fault is handled neutrally within this iteration. If the DUD gives the correct output under a pattern t, only ι_t is increased for faults which are observable under this pattern. In this way, candidates can be excluded using passing patterns, too. The maximum

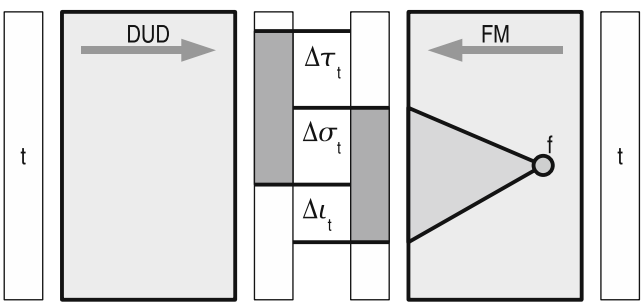

Fig. 5 Definition of evidence $e(f, t) = (\Delta\sigma_t, \Delta\iota_t, \Delta\tau_t, \Delta\gamma_t)$

achievable diagnostic resolution is bound by the size of the equivalence classes of the faults in the knowledge base.

If the fault in the DUD is not always active due to indeterministic behavior or some unknown activation mechanism, the measure still provides consistent evidences. For instance, let f' be a slow to rise transition fault. For some patterns t, f' will appear as a stuck-at 0 fault f, for others it is not observable. Then

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

provides $\Delta\sigma_t \geq \Delta\tilde{\sigma}_t$ for all other evidences

$$e(\tilde{f},t) = (\Delta\tilde{\sigma}_t, \Delta\tilde{\iota}_t, \Delta\tilde{\tau}_t, \Delta\tilde{\gamma}_t).$$

As a consequence, we have $\sigma_T \geq \tilde{\sigma}_T$ for all evidences $e(\tilde{f},T)$ and the evidence $e(f,T)$ is still useful for locating the fault. However, the value ι_T will not be zero any more and can be used for ranking fault candidates. This assumption is confirmed in the experimental results.

Let f be a conditional stuck-at fault which models at least a part of the DUD behavior for some patterns. Under each test pattern $t \in T$, the failing outputs of FM(f) and DUD are either disjoint ($\Delta\sigma_t = 0$) because the condition of f is not satisfied in the DUD or the set of failing outputs of FM(f) is a subset of the fails of DUD ($\Delta\iota_t = 0$). Hence, all $\Delta\gamma_t$ and also γ_T are zero for fault f. If there is a pattern t with $\Delta\gamma_t > 0$ like in Fig. 5, the corresponding conditional stuck-at is not a candidate.

This fault model independent pattern analysis approach is able to identify circuit parts containing arbitrary faulty behavior. However, if the behavior of the DUD can be explained using some classic fault models, certain evidence forms are observed. Table 1 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 $\iota_T = \gamma_T = 0$, 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.

Table 1 Fault models and evidence forms for $e(f,T)$ with $\sigma_T > 0$

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

If all suspects show positive values in all components ι_T, τ_T, γ_T, all simplistic fault models would fail to explain the DUD behavior.

For further analysis, the evidences in the knowledge base are ordered as follows to create a ranking with the most suspicious fault sites at the beginning (lowest rank). Firstly, evidences are sorted by increasing γ_T, i.e.

$$\gamma_T^a > \gamma_T^b \Rightarrow \text{rank}(e(f^a,T)) > \text{rank}(e(f^b,T))$$

moving single conditional stuck-at faults in front. Evidences with equal γ_T are then sorted by decreasing σ_T moving candidates in front, which explain most failures:

$$\sigma_T^a > \sigma_T^b \Rightarrow \text{rank}(e(f^a,T)) < \text{rank}(e(f^b,T)).$$

Finally evidences with equal γ_T and σ_T are ordered by increasing ι_T:

$$\iota_T^a > \iota_T^b \Rightarrow \text{rank}(e(f^a,T)) > \text{rank}(e(f^b,T)).$$

4 Volume Diagnosis and Pattern Generation

In volume diagnosis, the pattern set is fixed, and we have to extract as much diagnostic information as possible from rather limited information. Usually, only the first i failing patterns are recorded, and in addition, all the passing patterns up to this point can be used for diagnosis.

The number of suspected nodes reported by logic diagnosis must be limited in order to be used for low level failure analysis. If the number of suspects exceeds a parameter k, further manual or physical analysis is too expensive and logic diagnosis fails. If diagnosis successfully identified the culprit, the *rank* describes the position of the corresponding evidence within the ordered list. The average *rank* reported in the section below is just the average number of trials until the real culprit is identified.

During adaptive diagnosis and during design debug, we have more options if the resolution provided by the evidences of a test pattern set T is not sufficient, since the evidences may be used to guide further diagnostic ATPG.

For each fault f with $e(f,T) = (\sigma_T, \iota_T, \tau_T, \gamma_T)$ we have $\sigma_T + \iota_T > 0$, if T detects f. Otherwise, f may be undetected due to redundancy, or T must be improved to detect f.

Even if there are no suspects with $\sigma_T > 0$, the possible fault sites are ranked by ι_T. This way, multiple faults on redundant lines can be pointed out. For the special

case of $\iota_T = 0$, at least a subset of DUD failures can be explained with an unconditional stuck-at fault.

The faults with $e(f, T) = (\sigma_T, \iota_T, \tau_T, \gamma_T)$ and $\sigma_T > 0$ are the *suspects*, and by simple iteration over the ranking, pairs of suspects f^a, f^b are identified with equal evidences $e(f^a, T) = e(f^b, T)$. To improve the ranking, fault distinguishing patterns are generated [4, 30] and applied to the DUD.

To reduce the number of suspects and the region under consideration further, diagnostic pattern generation algorithms have to be employed which exploit layout data [12].

5 Experimental Results

We discuss the diagnostic resolution of POINTER for known benchmark circuits and large industrial designs and apply the algorithm to surrogate fault models. The circuits used are the combinational parts of the largest ISCAS89 and ITC99 benchmarks and industrial circuits provided by NXP. The characteristics of the circuits are given in Table 2. Column 2 denotes the number of two-input gates in the circuit. The number of evidences in the knowledge base equals the number of stucturally collapsed stuck-at faults shown in column 3.

Table 2 Circuit characteristics

Circuit	Gates	Faults
s38417	24079	32527
s38584	22092	38945
s35932	16353	39094
b20	22557	47964
b21	23100	48812
b22_1	24385	52931
p35k	46435	70382
b22	33569	71365
p45k	43190	72164
b17	37446	84737
b17_1	44544	96092
p77k	72370	124572
p89k	88726	155900
p78k	74243	163310
p100k	96685	168102
p81k	108991	224424
b18	130949	285210
p141k	172686	291546
p267k	271538	375958
p269k	272630	378142
p239k	259241	456982
p330k	312666	558163
b19	263547	575193
p286k	332726	672496
p418k	382633	715945
p388k	433331	865000
p951k	816072	1618672

Two classes of experiments were performed. For volume diagnosis, success rates for a fixed pattern set are compared with the success rate of the SLAT algorithm. For adaptive debug and diagnosis, pattern analysis and generation are performed until the culprits are identified.

5.1 Volume Diagnosis

In this section, we discuss the diagnostic success rates and resolution of POINTER for known benchmark circuits and large industrial designs provided by NXP. We diagnose defects of three different types and compare the results to the outcome of the SLAT algorithm.

In the experiments reported below, first we apply 5000 random patterns and add deterministic patterns for maximum stuck-at fault coverage. The maximum number of suspects is $k = 10$, and if the real culprit not within the first 10, we classify that logic diagnosis failed. The second parameter which is important for any diagnosis approach is the average number of low level investigations we have to invoke. If diagnosis is successful, this number is the rank. If diagnosis fails, this number is 11. The third important parameter for multi-site testing, embedded diagnosis or even BIST is the number of failing patterns which have to be stored and evaluated. Results reported below are obtained for evaluating i failing patterns, $i = 1, 4$, and 8.

The outcome of the SLAT approach however is not a ranked list of suspects but an unordered set of multiplets. Each multiplet is a minimal set of evidences which can explain all the SLAT patterns. For the SLAT algorithm, the rank is defined as the expected number of drawings from these multiplets until the real culprit is found, however the maximum is set to 11 as for POINTER.

We are reporting the average success rate in percent and the average ranks after encountering i failing patterns during analysis. These averages are calculated from at least 1000 test cases per design. The best values in these tables are set in bold.

5.1.1 Single Stuck-at Fault Diagnosis

Success rates and average ranks for diagnosis of randomly injected single stuck-at faults are shown in Table 3. SLAT is always able to produce a set of multiplets with the victim line included, but the resulting rank often exceeds the threshold k and leads to a

Table 3 Stuck-at fault diagnosis with limited failure information

Design	1st fail				4th fail				8th fail			
	Slat		Pointer		Slat		Pointer		Slat		Pointer	
	%s	r	%s	r	%s	r	%s	r	%s	r	%s	r
s35932	50	6.2	**86**	**3.9**	100	2.7	100	**1.5**	100	2.3	100	**1.4**
s38584	97	3.6	**99**	**2.2**	100	2.3	100	**1.2**	100	2.1	100	**1.2**
b20	66	7.0	**93**	**3.2**	95	3.2	**100**	**1.4**	97	2.6	**100**	**1.2**
b21	64	7.1	**93**	**3.2**	94	3.2	**99**	**1.4**	96	2.7	**100**	**1.2**
s38417	86	4.3	**95**	**2.7**	99	2.3	**100**	**1.4**	100	2.0	100	**1.2**
b22	69	6.9	**94**	**3.1**	97	3.0	**99**	**1.3**	98	2.5	**100**	**1.2**
b17	87	4.9	**96**	**2.6**	98	2.9	**100**	**1.5**	99	2.6	**100**	**1.4**
p45k	71	5.4	**90**	**3.2**	99	2.8	**100**	**1.3**	99	2.2	**100**	**1.2**
p35k	41	6.9	**75**	**4.5**	86	3.9	**99**	**1.7**	96	2.9	**100**	**1.4**
p77k	75	5.0	**88**	**3.1**	92	3.2	**95**	**1.9**	95	2.7	**96**	**1.7**
p78k	80	5.8	**92**	**3.7**	100	2.1	100	**1.2**	100	1.8	100	**1.1**
p89k	73	5.5	**93**	**2.7**	97	3.1	**100**	**1.4**	99	2.6	**100**	**1.2**
p100k	65	5.6	**87**	**3.4**	97	2.9	**100**	**1.4**	100	2.2	100	**1.2**
p81k	54	7.2	**89**	**3.5**	100	2.6	100	**1.4**	100	2.3	100	**1.3**
b18	73	5.8	**89**	**3.3**	90	3.6	**98**	**1.7**	96	3.0	**99**	**1.5**
p141k	81	4.8	**95**	**2.7**	100	2.4	100	**1.3**	100	2.1	100	**1.2**
p239k	72	5.1	**89**	**3.3**	99	2.5	**100**	**1.3**	100	2.0	100	**1.2**
b19	77	5.5	**89**	**3.2**	89	3.6	**97**	**1.8**	93	3.1	**99**	**1.5**
p267k	90	4.2	**96**	**2.6**	99	2.3	**100**	**1.4**	100	2.0	100	**1.3**
p269k	89	4.1	**96**	**2.5**	100	2.2	100	**1.3**	100	2.0	100	**1.3**

fail even if the number i of analyzed failing patterns is increased. The average ranks and success rates of POINTER mark the maximum achievable diagnostic resolution because all remaining candidates are equivalent under the pattern set applied. The diagnostic result of SLAT can be refined towards this resolution by performing a passing pattern validation [6].

5.1.2 Crosstalk Fault Diagnosis

During the experiments a crosstalk fault is described by the change of a victim line, if there is a transition on the aggressor line. Since this is also a single line fault, similar results are obtained as in single stuck-at fault diagnosis (see Table 4). Again, POINTER provides

Table 4 Diagnosis of crosstalk faults with limited failure information

Design	1st fail				4th fail				8th fail			
	Slat		Pointer		Slat		Pointer		Slat		Pointer	
	%s	r	%s	r	%s	r	%s	r	%s	r	%s	r
s35932	52	6.1	**84**	**4.2**	100	2.7	100	**1.6**	100	2.3	100	**1.4**
s38584	97	3.5	**98**	**3.0**	100	2.0	100	**1.4**	100	1.8	100	**1.3**
b20	74	6.5	**91**	**3.7**	99	2.5	99	**1.5**	100	2.0	100	**1.3**
b21	70	6.7	**91**	**3.8**	99	2.5	**100**	**1.4**	100	2.0	100	**1.2**
s38417	90	3.9	**93**	**3.2**	99	2.1	**100**	**1.5**	100	1.8	100	**1.3**
b22	75	6.4	**93**	**3.7**	99	2.4	**100**	**1.4**	100	2.0	100	**1.3**
b17	89	4.4	**94**	**3.0**	98	2.4	**100**	**1.6**	99	2.1	**100**	**1.4**
p45k	72	5.1	**82**	**4.0**	98	2.5	**99**	**1.6**	100	1.9	100	**1.3**
p35k	63	5.1	**74**	**4.4**	97	2.3	**99**	**1.8**	100	1.8	100	**1.5**
p77k	82	4.2	**87**	**3.6**	98	2.3	98	**1.7**	99	2.0	99	**1.5**
p78k	77	6.0	**81**	**5.0**	100	1.9	100	**1.3**	100	1.6	100	**1.1**
p89k	75	5.0	**90**	**3.6**	100	2.3	100	**1.6**	100	1.9	100	**1.4**
p100k	67	5.4	**79**	**4.4**	98	2.6	**99**	**1.6**	100	2.0	100	**1.3**
p81k	55	6.9	**77**	**5.1**	100	2.3	100	**1.6**	100	2.0	100	**1.4**
b18	73	5.6	**82**	**4.1**	91	3.2	**95**	**2.1**	96	2.6	**98**	**1.7**
p141k	85	4.4	**94**	**3.2**	100	2.1	100	**1.5**	100	1.8	100	**1.3**
p239k	70	5.2	**83**	**4.0**	99	2.2	**100**	**1.4**	100	1.8	100	**1.2**
b19	69	5.8	**81**	**4.2**	87	3.5	**94**	**2.1**	94	2.8	**96**	**1.7**
p267k	91	3.9	**93**	**3.1**	100	2.0	100	**1.6**	100	1.7	100	**1.4**
p269k	92	3.7	**94**	**3.0**	100	2.0	100	**1.5**	100	1.8	100	**1.4**

Table 5 Diagnosis of byzantine bridges with limited failure information

Design	1st fail				4th fail				8th fail			
	Slat		Pointer		Slat		Pointer		Slat		Pointer	
	%s	r	%s	r	%s	r	%s	r	%s	r	%s	r
s35932	17	8.7	**53**	**6.7**	37	7.7	**83**	**4.2**	42	7.4	**90**	**3.4**
s38584	67	5.3	**97**	**3.1**	85	3.7	**100**	**1.7**	88	3.3	**100**	**1.4**
b20	70	6.6	**87**	**4.8**	78	5.0	**98**	**2.3**	89	4.0	**98**	**1.7**
b21	69	6.6	**88**	**4.6**	77	5.0	**99**	**2.2**	89	4.0	**99**	**1.7**
s38417	58	5.8	**96**	**3.4**	72	4.5	**99**	**2.0**	76	4.0	**100**	**1.6**
b22	72	6.6	**86**	**4.9**	79	4.9	**98**	**2.2**	90	4.0	**99**	**1.7**
b17	78	5.2	**93**	**3.8**	88	4.0	**99**	**2.1**	91	3.6	**99**	**1.8**
p45k	56	6.0	**87**	**4.2**	74	4.5	**97**	**2.1**	80	4.0	**99**	**1.7**
p35k	45	6.5	**75**	**4.5**	61	5.5	**85**	**3.2**	68	5.0	**89**	**2.7**
p77k	56	6.1	**80**	**4.4**	69	4.9	**89**	**2.8**	74	4.4	**92**	**2.4**
p78k	25	8.6	**77**	**5.7**	54	6.5	**98**	**2.3**	66	5.7	**100**	**1.4**
p89k	69	5.3	**92**	**3.5**	84	4.0	**99**	**1.9**	88	3.7	**99**	**1.6**
p100k	55	6.0	**80**	**4.4**	75	4.5	**96**	**2.4**	84	3.8	**98**	**1.8**
p81k	43	7.7	**73**	**5.6**	76	4.8	**98**	**2.4**	87	3.8	**99**	**1.9**
b18	72	5.8	**80**	**4.6**	80	4.7	**91**	**2.8**	87	4.1	**95**	**2.3**
p141k	65	5.7	**91**	**4.0**	86	4.0	**99**	**2.0**	90	3.5	**100**	**1.6**
p239k	47	6.6	**77**	**4.8**	72	4.6	**97**	**2.3**	80	3.9	**99**	**1.6**
b19	70	5.9	**81**	**4.6**	81	4.8	**92**	**2.7**	87	4.2	**96**	**2.2**
p267k	70	5.2	**95**	**3.3**	85	3.8	**99**	**1.9**	88	3.4	**100**	**1.5**
p269k	70	5.1	**94**	**3.3**	85	3.8	**99**	**2.0**	90	3.2	**100**	**1.6**

maximum diagnostic resolution and needs a smaller number of failed responses to be analyzed.

5.1.3 Byzantine Bridging Fault Diagnosis

Resistive bridges are often modelled by byzantine faults [26]. They are very hard to diagnose without a specialized fault model and therefore provide a good benchmark for fault model independent diagnosis approaches. The two signal lines of those bridges may interchange their values and one or both may be faulty. Therefore there may be two independent faulty signals present in the circuit.

As can be seen in Table 5, the success rate of SLAT drops dramatically because failing patterns either don't have the SLAT-property or their single explaining evidences are not pointing to the bridge.

After analysis of at least 8 fails, POINTER provides a near-perfect success rate in almost every test case. The average ranks show, that only one or two physical inspections are required in average to find the real defect.

5.2 Adaptive Debug and Diagnosis

In this section, we improve the test set based on the response of the device under diagnosis.

For comparison reasons, we present results for stuck-at faults, and as a representative for general defects we analyze stuck-open faults. Design errors are represented by exchanging gate functions.

5.2.1 Single Stuck-at Fault Diagnosis

In this experiment, 100 test cases with randomly injected detectable single stuck-at faults are diagnosed in each circuit. A diagnosis run starts by applying first random and then deterministic test patterns to the DUD until a faulty behavior is encountered. If the algorithm finds fault candidates f with $e(f, T) = (\sigma_T, 0, 0, 0)$ and σ_T maximum, distinguishing patterns are generated until no further distinguishing is possible. Table 6 shows

Table 6 Pattern count and resolution of SSA-fault diagnosis

Circuit	Evidences	Patterns	Suspects	Rank
s38417	32527	1207.6	1.4	1.2
s38584	38945	1153.6	1.1	1.1
b20	47964	2234.0	1.1	1.1
b21	48812	2385.9	1.2	1.1
b22_1	52931	2164.7	1.2	1.1
b22	71365	2678.2	1.1	1.1
b17	84737	2696.5	1.4	1.2
b17_1	96092	3989.3	1.2	1.1
b18	285210	11037.8	1.4	1.2
p330k	558163	11526.9	1.2	1.1
b19	575193	12967.1	1.8	1.3
p286k	672496	9673.3	1.2	1.1
p418k	715945	7936.5	1.3	1.1
p388k	865000	8976.8	1.5	1.3
p951k	1618672	9825.6	1.1	1.1

the results. Columns 1 and 2 denote the circuit and its number of evidences, column 3 shows the average number of patterns used for diagnosis. The achieved diagnostic resolution, which is the average number of candidate fault classes, is shown in column 4. For completeness, column 5 shows the average rank of the fault in the DUD which is in this case $\text{rank}(s) = (s+1)/2$ with s being the number of suspects.

The diagnosis for single stuck-at faults is complete, i.e. POINTER provides an optimal resolution. Here, the average number of fault candidate classes is larger than one, because these classes were determined by simple structural fault collapsing and ATPG has proven the functional equivalence for more fault pairs during diagnosis.

5.2.2 Diagnosis of Stuck-Open Faults

Intra-gate stuck-open faults may result in a transition fault at the output signal of the faulty gate [13]. In this case, pattern analysis leads to evidences $e(f, T) = (\sigma_T, \iota_T, 0, 0)$ with σ_T maximum and $\iota > 0$ and ATPG is switched to generate pattern *pairs* to provoke possible transition faults. Additionally, during fault distinguishing, neighboring signals can be driven to different values to improve resolution for possible bridging faults.

Table 7 shows the average number of suspect fault classes in column 4. The rank, which is shown in column 5, is considerably lower than average because of the sorting by ι_T.

5.2.3 Debug of Design Faults

We now consider faulty designs in which one gate is of a wrong type. Such faults behave like either one or two conditional stuck-at faults. In the former case, the algorithm encounters only evidences of the form $e(f, T) = (\sigma_T, \iota_T, 0, 0)$ and the strategy of the last section is used. Two conditional stuck-at faults (AND replaced by XOR for instance) result in an evidence of the form $e(f, T) = (\sigma_T, \iota_T, \tau_T, 0)$ with σ_T maximum, $\iota_T > 0$, $\tau_T > 0$. In this case, all other evidences with $\sigma_T > 0$, $\iota_T > 0$ and $\tau_T > 0$ are included in the suspect list, which is then sorted first by decreasing σ_T then by increasing ι_T.

Table 8 shows the resulting average number of suspect conditional faults in column 4. s is much higher in this case, because every evidence with $\sigma_T > 0$ is included as soon as the algorithm detects a multi-site fault by observing positive τ_T among the top-ranked suspects. The average rank of the evidences leading to the faulty gates (column 5) is only marginally affected by this high suspect count.

Table 7 Pattern count and resolution for stuck-open faults

Circuit	Evidences	Patterns	Suspects	Rank
s38417	32527	1092.0	2.3	1.3
s38584	38945	1118.8	2.4	1.1
b20	47964	2172.9	4.7	1.1
b21	48812	2349.4	3.9	1.1
b22_1	52931	2142.2	4.2	1.1
b22	71365	2837.2	3.5	1.1
b17	84737	2766.7	3.1	1.0
b17_1	96092	3977.5	3.3	1.1
b18	285210	10871.2	4.2	1.2
p330k	558163	11617.3	2.6	1.2
b19	575193	13252.0	4.2	1.3
p286k	672496	7249.9	4.0	1.1
p418k	715945	8136.7	2.0	1.0
p388k	865000	7235.5	3.2	1.1
p951k	1618672	9643.0	3.8	1.1

Table 8 Pattern count and resolution of gate debug

Circuit	Evidences	Patterns	Suspects	Rank
s38417	32527	1874.3	47.3	1.6
s38584	38945	1503.5	67.6	1.1
b20	47964	2963.7	417.6	1.4
b21	48812	3191.2	619.8	1.3
b22_1	52931	2482.3	211.7	1.2
b22	71365	3523.7	555.2	1.8
b17	84737	3123.8	342.8	1.1
b17_1	96092	4292.2	326.2	1.3
b18	285210	11531.0	301.1	1.7
p330k	558163	16614.1	195.3	1.3
b19	575193	18165.1	345.2	1.6
p286k	672496	10740.0	167.6	1.2
p418k	715945	11682.0	128.4	1.3
p388k	865000	9264.6	265.9	1.5
p951k	1618672	10347.7	256.6	1.8

6 Conclusion

A novel approach to adaptive diagnosis has been presented which combines a new effect-cause pattern analysis algorithm with high-resolution ATPG. The pattern analysis does not rely on SLAT-patterns, tolerates unmodeled behavior and therefore enables fault model independent diagnosis. By applying this approach to some diagnosis and debug problems, it has been shown, that the resolution is excellent for the used surrogate faults and an effective ranking is provided for unmodeled behavior.

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

References

1. Abramovici M, Breuer MA (1980) Fault diagnosis based on effect-cause analysis: an introduction. In: 17th conference on design automation, pp 69–76
2. Amyeen ME, Nayak D, Venkataraman S (2006) Improving precision using mixed-level fault diagnosis. In: Proceedings IEEE international test conference 2006, Santa Clara, 24–26 October 2006, p 22.3
3. Arnaout T, Bartsch G, Wunderlich H-J (2006) Some common aspects of design validation, debug and diagnosis. In: Third IEEE international workshop on electronic design, test and applications (DELTA 2006), Kuala Lumpur, 17–19 January 2006, pp 3–10
4. Bartenstein T (2000) Fault distinguishing pattern generation. In: Proceedings IEEE international test conference 2000, Atlantic City, pp 820–828
5. Bartenstein T, Heaberlin D, Huisman LM, Sliwinski D (2001) Diagnosing combinational logic designs using the single location at-a-time (SLAT) paradigm. In: Proceedings IEEE international test conference 2001, Baltimore, 30 October–1 November 2001, pp 287–296
6. Bhatti NK, Blanton RS (2006) Diagnostic test generation for arbitrary faults. In: Proceedings IEEE international test conference 2006, Santa Clara, 24–26 October, 2006, p 19.2
7. Boppana V, Fujita M (1998) Modeling the unknown! towards model-independent fault and error diagnosis. In: Proceedings IEEE international test conference 1998, Washington, DC, 18–22 October 1998, pp 1094–1100
8. Boppana V, Hartanto I, Fuchs WK (1996) Full fault dictionary storage based on labeled tree encoding. In: 14th IEEE VLSI test symposium (VTS'96), Princeton, 28 April–1 May 1996, pp 174–179
9. Chen KC (2003) Assertion-based verification for SoC designs. In: Proceedings 5th international conference on ASIC, vol 1, pp 12–15
10. Chess B, Larrabee T (1999) Creating small fault dictionaries. IEEE Trans Comput-aided Des Integr Circuits Syst 18(3):346–356
11. Chess B, Lavo DB, Ferguson FJ, Larrabee T (1995) Diagnosis of realistic bridging faults with single stuck-at information. In: Proceedings of the 1995 IEEE/ACM international conference on computer-aided design, 1995, San Jose, 5–9 November 1995, pp 185–192
12. Desineni R, Poku O, Blanton RDS (2006) A logic diagnosis methodology for improved localization and extraction of accurate defect behavior. In: Proceedings IEEE international test conference 2006, Santa Clara, 24–26 October 2006, p 12.3
13. Fan X, Moore W, Hora C, Gronthoud G (2005) Stuck-open fault diagnosis with stuck-at model. In: Proceedings European test symposium, Tallin, pp 182–187
14. Fitzpatrick T (2005) Realizing advanced functional verification with questa. Mentor Graphics Corporation (white paper)
15. Gong Y, Chakravarty S (1995) On adaptive diagnostic test generation. In: Proceedings IEEE international conference on computer-aided design, p 181
16. Henderson CL, Soden JM (1997) Signature analysis for ic diagnosis and failure analysis. In: Proceedings IEEE international test conference 1997, Washington, DC, 3–5 November 1997, pp 310–318
17. Holst S, Wunderlich H-J (2007) Adaptive debug and diagnosis without fault dictionaries. In: 12th European test symposium (ETS 2007), 20 May 2007, Freiburg, pp 7–12
18. Hora C, Segers R, Eichenberger S, Lousberg M (2002) An effective diagnosis method to support yield improvement. In: Proceedings IEEE international test conference 2002, Baltimore, 7–10 October 2002, pp 260–269
19. Huisman LM (2004) Diagnosing arbitrary defects in logic designs using single location at a time (SLAT). IEEE Trans Comput-aided Des Integr Circuits Syst 23(1):91–101
20. Klein R, Piekarz T (2005) Accelerating functional simulation for processor based designs. Mentor Graphics Corporation (white paper)
21. Krstic A, Wang L-C, Cheng K-T, Liou J-J, Abadir MS (2003) Delay defect diagnosis based upon statistical timing models—the first step. In: 2003 design, automation and test in Europe conference and exposition (DATE 2003), Munich, 3–7 March 2003, pp 10328–10335
22. Lavo DB, Chess B, Larrabee T, Hartanto I (1998) Probabilistic mixed-model fault diagnosis. In: Proceedings IEEE international test conference 1998, Washington, DC, 18–22 October 1998, pp 1084–1093
23. McPherson JW (2006) Reliability challenges for 45nm and beyond. In: Proceedings of the 43rd design automation conference, DAC 2006, San Francisco, 24–28 July 2006, pp 176–181
24. Millman SD, McCluskey EJ, Acken JM (1990) Diagnosing CMOS bridging faults with stuck-at fault dictionaries. In: Proceedings IEEE international test conference, pp 860–870
25. Pomeranz I, Reddy SM (1992) On the generation of small dictionaries for fault location. In: IEEE/ACM international conference on computer-aided design, ICCAD92, Santa Clara, 8–12 November 1992, pp 272–279
26. Renovell M, Huc P, Bertrand Y (1994) CMOS bridging fault modeling. In: 12th IEEE VLSI test symposium (VTS), 25–28 Apr 1994, pp 392–297
27. Riley M, Chelstrom N, Genden M, Sawamura S (2006) Debug of the CELL processor: moving the lab into silicon. In: Proceedings IEEE international test conference 2006, Santa Clara, 24–26 October 2006, p 26.1
28. Roy K, Mak TM, Cheng K-TT (2006) Test consideration for nanometer-scale cmos circuits. IEEE Des Test Comput 23(2):128–136
29. Ubar R (2003) Design error diagnosis with resynthesis in combinational circuits. J Electron Test Theory Appl 19:73–82
30. Veneris AG, Chang R, Abadir MS, Amiri M (2004) Fault equivalence and diagnostic test generation using atpg. In: Proceedings IEEE international symposium on circuits and systems, 2004, pp 221–224
31. Waicukauski JA, Lindbloom E (1989) Failure diagnosis of structured VLSI. IEEE Des Test Comput 6(4):49–60
32. Wunderlich H-J (2005) From embedded test to embedded diagnosis. In: Proceedings European test symposium, Tallin, pp 216–221

Stefan Holst received his Diploma in Computer Science from the University of Stuttgart in 2005. He joined the Institute for Computer Architecture and Computer Engineering in 2006.

Hans-Joachim Wunderlich received a Diploma in Mathematics from the University of Freiburg in 1981 and the Dr. rer. nat. (Ph.D.) with distinction from the University of Karlsruhe in 1986. Since 1991 he has been a full Professor and since 2002 he has been the director of the Institute of Computer Architecture and Computer Engineering at the University of Stuttgart. He is editor of various international journals and program committee member a variety of IEEE conferences on design and test of electronic systems. In 2009 he became an IEEE Fellow.