Adaptive Debug and Diagnosis Without Fault Dictionaries

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

***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 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.

***Keywords*— Diagnosis, Debug, Test, VLSI**

I. Introduction

A. Debug and diagnosis

Traditionally, design, verification and diagnosis of micro-electronic 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 [1], [2]. Despite the efforts spent from 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 [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], identifying them is part of silicon debug. With growing circuit complexity and shrinking geometries, the actual behavior of the silicon is hard to model [5], [6], [7] and cannot always be predicted and simulated [8].

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 [9].

Precision diagnosis is performed on a small selected set of chips like first silicon or representants 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 between dealing with yield ramping and design for manufacturability [10], [11], [12]. Diagnosis and debug have the common objective of achieving high diagnostic resolution and especially fault model independent approaches are suitable for both of these tasks.

B. 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 [13], [14]. *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

12th IEEE European Test Symposium (ETS'07)
0-7695-2827-9/07 $20.00

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 [15], [16]. The *resolution* of a test set corresponds to the number of faults which cannot be distinguished any further [17], [18], [19].

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, and a main goal of fault diagnosis is rather finding an appropriate fault model than taking a given fault model as an initial assumption [15], [20].

C. Fault dictionaries vs. adaptive diagnosis

Cause-effect diagnosis can be speeded up, if for each fault and each failing simulated pattern the erroneous output is stored in a dictionary [21]. Even after an effect-cause pass, the size of such a dictionary may explode, and significant research effort has been spent for reducing the size of fault dictionaries [22], [23].

During debug and during diagnosis of first silicon, there exists an efficient alternative to precomputed fault dictionaries in so-called adaptive diagnosis [24]. Here, we use faulty and faultfree 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. 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.

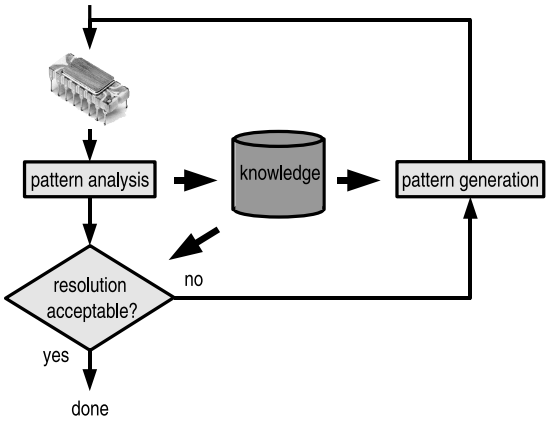

Fig. 1. Adaptive diagnosis flow

Such a diagnostic ATPG does not rely on a precomputed fault dictionary, and significant memory savings are obtained. The goal of this paper is to present a new approach for diagnostic test pattern generation and analysis, which is adaptive and fault model independent. 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. [19], [20], [25], [26], [27].

The next section of this paper introduces an analysis algorithm which identifies possible faulty or defect regions based on a given test set. Section 3 adds a pattern generation algorithm for increasing the resolution, and section 4 validates the approach experimentally by locating faults of different surrogate models.

II. Pattern Analysis

The approach presented below is an extension of the 'Single Location At a Time' (SLAT) technique introduced by [25], [26]. A diagnostic test pattern has the SLAT property, if there is a single observable 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, this approach is not based on the stuck-at fault model, but uses a set of identified stuck-at faults for describing the suspicious region of the DUD.

Since the stuck-at faults are active only for a subset of the test patterns, we can extract conditions where they occur leading to so-called conditional stuck-at faults. Figure 2 models a design error, where an AND gate is exchanged by an OR gate.

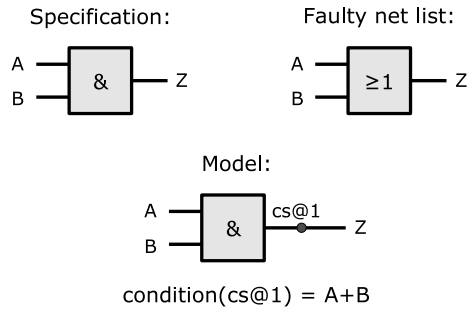

Fig. 2. Example of a conditional stuck-at fault

The main drawback of the SLAT paradigm with conditional stuck-at faults is the fact that information for fault location is only extracted from patterns with the SLAT property. All the other patterns are not taken into account, neither failing nor passing ones.

For the rest of this section, we define a measure to quantify how well a stuck-at fault reflects the behavior of the DUD 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. 3) 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 $\Delta\iota_t = \Delta\tau_t = \Delta\gamma_t = 0$.

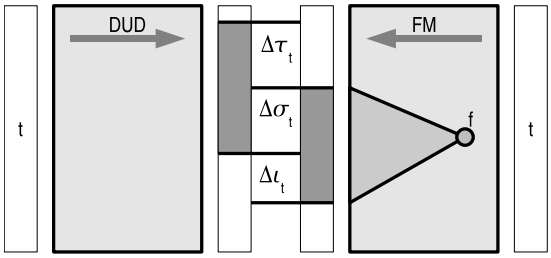

Fig. 3. Definition of evidence $e(f,t) = (\Delta\sigma_t, \Delta\iota_t, \Delta\tau_t, \Delta\gamma_t)$

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

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

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

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

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 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 figure 3, the corresponding conditional stuck-at is not a candidate.

Our 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. Such observations are very useful to classify the DUD behavior to decide for instance, if physical failure analysis is feasible. Table I shows suspect evidences for some classic models.

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 I

FAULT MODELS AND EVIDENCE FORMS FOR $e(f,T)$ WITH $\sigma_T > 0$

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.

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.

III. PATTERN GENERATION

If the resolution provided by the evidences of a test pattern set T is not sufficient, the evidences are used to guide 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.

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:

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

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.

Faults with $e(f,T) = (\sigma_T, \iota_T, \tau_T, \gamma_T)$ and $\sigma_T > 0$ are called *suspect*. 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 [17], [18] 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 [15].

IV. Experimental Results

In this section, we discuss the diagnostic resolution of the approach 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 the largest industrial circuits we had available. The charactericties of the circuits are given in table II. 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.

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. At the end of this section, performance data is given.

A. 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 III shows 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.

circuit	gates	faults
s38417.FS	24079	32527
s38584.FS	22092	38945
b20.FS	22557	47964
b21.FS	23100	48812
b22_1.FS	24385	52931
b22.FS	33569	71365
b17.FS	37446	84737
b17_1.FS	44544	96092
b18.FS	130949	285210
p330k.FS	312666	558163
b19.FS	263547	575193
p286k.FS	332726	672496
p418k.FS	382633	715945
p388k.FS	433331	865000
p951k.FS	816072	1618672

TABLE II
CIRCUIT CHARACTERISTICS

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

TABLE III
PATTERN COUNT AND RESOLUTION OF SSA-FAULT DIAGNOSIS

The diagnosis for single stuck-at faults is complete, i.e. the proposed approach 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.

0-7695-2827-9/07 $20.00

B. Diagnosis of Stuck-Open Faults

Intra-gate stuck-open faults result in a transition fault at the output signal of the faulty gate [28]. 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 IV 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.

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

TABLE IV

PATTERN COUNT AND RESOLUTION FOR STUCK-OPEN FAULTS

C. Debug of Design Faults

We now consider faulty designs in which one gate is of a wrong type. Such faults behave like either one (figure 2) 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 V shows the resulting average number of suspect conditional faults in column 4. This number 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.

D. Pattern Analysis Performance

Table VI shows the performance characteristics of the pattern analysis implementation. Well known fault simulation techniques [29], [30] have been adapted to backtrace value changes of evidences through fanout-free regions and over independent fanouts. Therefore, only about 10% of the evidences need actually be simulated explicitly (column 3) to analyze a pattern. The values for all the remaining evidences are derived from values of simulated evidences in combination with observability considerations.

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

TABLE V

PATTERN COUNT AND RESOLUTION OF GATE DEBUG

circuit	evidences	expl. sim.	patterns/s
s38417.FS	32527	4909	388
s38584.FS	38945	4471	496
b20.FS	47964	3847	85
b21.FS	48812	3877	82
b22_1.FS	52931	4695	82
b22.FS	71365	5704	53
b17.FS	84737	6981	80
b17_1.FS	96092	7255	117
b18.FS	285210	25324	15
p330k.FS	558163	58258	15
b19.FS	575193	51560	8
p286k.FS	672496	70418	12
p418k.FS	715945	71610	13
p388k.FS	865000	102805	9
p951k.FS	1618672	206766	6

TABLE VI

PATTERN ANALYSIS PERFORMANCE

Column 4 shows the achieved pattern analysis rate for all evidences using a single-threaded implementation which runs on an AMD Opteron(tm) 850 2.4 GHz. We observe a much better performance for the industrial circuits compared to ITC99 benchmarks due to their shorter average path lengths. Figure 4 underlines this observation and furthermore shows a linear complexity of the proposed algorithm.

V. 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

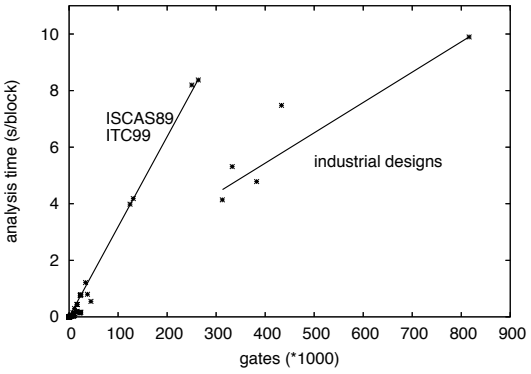

Fig. 4. Analysis times for blocks of 64 patterns

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.

VI. Acknowledgement

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

We would like to thank Guenter Bartsch for his support during development of the first ideas for this approach and Christian Zoellin for many enlightening and clarifying discussions in later stages.

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] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

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

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

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

[19] 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.

[20] V. Boppana and M. Fujita, "Modeling the unknown! Towards model-independent fault and error diagnosis." in *Proceedings IEEE International Test Conference 1998, Washington, DC, USA, October 18-22, 1998*, 1998, pp. 1094–1100.

[21] 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.

[22] 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.

[23] 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.

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

[25] 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.

[26] 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.

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

[28] X. Fan, W. Moore, C. Hora, and G. Gronthoud, "Stuck-open fault diagnosis with stuck-at model," in *Proceedings European Test Symposium, Tallin, Estonia*, 2005, pp. 182–187.

[29] J. A. Waicukauski, E. B. Eichelberger, D. O. Forlenza, E. Lindbloom, and T. Mc-Carthy, "Fault simulation for structured VLSI," *VLSI Systems Design*, vol. 6, no. 12, pp. 20–32, Dec 1985.

[30] M. H. Schulz, *Testmustergenerierung und Fehlersimulation in digitalen Schaltungen mit hoher Komplexität*, ser. Informatik-Fachberichte. Springer, 1988, vol. 173.