Logic Fault Diagnosis of Hidden Delay Defects

Holst, Stefan; Kampmann, Matthias; Sprenger, Alexander; Reimer, Jan Dennis; Hellebrand, Sybille; Wunderlich, Hans-Joachim; Wen, Xiaoqing

Proceedings of the IEEE International Test Conference (ITC'20), Washington DC, USA, 2020, pp. 1–6

doi: https://doi.org/10.1109/ITC44778.2020.9325234

Abstract: Hidden delay defects (HDDs) are small delay defects that pass all at-speed tests at nominal capture time. They are an important indicator of latent defects that lead to early-life failures and aging problems that are serious especially in autonomous and medical applications. An effective way to screen out HDDs is to use Faster-than-At-Speed Testing (FAST) to observe outputs of sensitized non-critical paths which are expected to be stable earlier than nominal capture time.To improve the reliability of current and future designs, it is important to learn about the population of HDDs using logic diagnosis. We present the very first logic fault diagnosis technique that is able to identify HDDs by analyzing fail logs produced by FAST.Even with aggressive FAST testing, HDDs generate only very few failing test response bits. To overcome this severe challenge, we propose new backtracing and response matching methods that yield high diagnostic success rates even with very limited amount of failure data. The performance and scalability of our HDD diagnosis method is validated using fault injection campaigns with large benchmark circuits.

Logic Fault Diagnosis of Hidden Delay Defects

Stefan Holst[1], Matthias Kampmann[2], Alexander Sprenger[2], Jan Dennis Reimer[2],
Sybille Hellebrand[2], Hans-Joachim Wunderlich[3], and Xiaoqing Wen[1]

[1]Kyushu Institute of Technology, Iizuka, 820-8502, Japan
[2]Paderborn University, Warburger Str. 100, 33098 Paderborn, Germany
[3]University of Stuttgart, Pfaffenwaldring 47, 70569 Stuttgart, Germany

***Abstract*—Hidden delay defects (HDDs) are small delay defects that pass all at-speed tests at nominal capture time. They are an important indicator of latent defects that lead to early-life failures and aging problems that are serious especially in autonomous and medical applications. An effective way to screen out HDDs is to use Faster-than-At-Speed Testing (FAST) to observe outputs of sensitized non-critical paths which are expected to be stable earlier than nominal capture time.**

To improve the reliability of current and future designs, it is important to learn about the population of HDDs using logic diagnosis. We present the very first logic fault diagnosis technique that is able to identify HDDs by analyzing fail logs produced by FAST.

Even with aggressive FAST testing, HDDs generate only very few failing test response bits. To overcome this severe challenge, we propose new backtracing and response matching methods that yield high diagnostic success rates even with very limited amount of failure data. The performance and scalability of our HDD diagnosis method is validated using fault injection campaigns with large benchmark circuits.

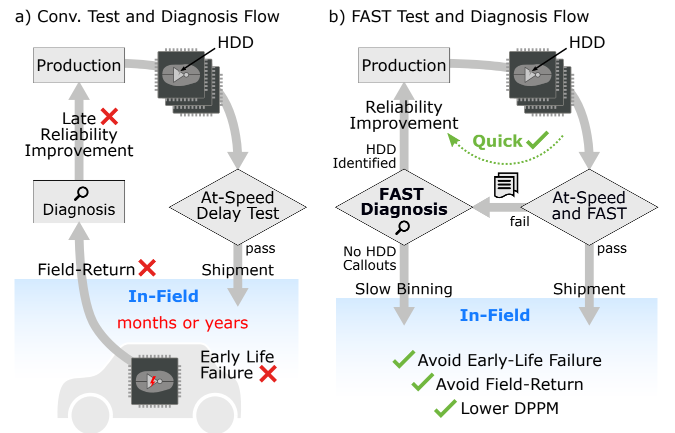

Fig. 1. Faster-than-At-Speed testing flow compared to conventional at-speed testing and how diagnosis of hidden delay defects enables quick reliability improvement.

I. Introduction

Functional safety is a major concern in autonomous and medical systems. Respective standards, e. g., the ISO 26262 standard for the automotive domain, define fault coverage requirements for manufacturing and in-system testing. In this context, marginalities pose serious challenges, as they may not be observable during manufacturing test, but can cause *early-life failures* (ELF) in the field as illustrated in Fig. 1a. In addition to safety hazards, ELFs may also lead to huge economic losses for overall equipment manufacturers. As weak devices often suffer from small additional delays, screening for *small delay defects* (SDDs) is critical in sorting out chips with potential early life failures while reducing the cost for burn-in tests [1, 2]. This is even more important for recent FinFET technologies [3, 4]. However, if small delays can only be propagated over short paths, they will remain undetected even with timing-aware test patterns applied at speed. Testing for such *hidden delay defects* (HDDs) require *Faster-than-At-Speed Testing* (FAST), typically with various frequencies [5–12].

FAST significantly increases the reliability of shipped products, but to achieve a quick *reliability ramp-up* of a manufacturing process, it must be combined with proper diagnosis techniques (see Fig. 1b). Here the goal is to assess the severity of the reliability problems and to distinguish defective chips from chips that are slow due to parameter variations. Distinguishing the effects of parameter variations from small delay defects is also crucial for avoiding unnecessary yield loss. In previous work, this problem was addressed for resistive SDDs and gate-oxide defects [13, 14]. The proposed solutions exploit the observation that slow chips show a different behavior than defective chips for varying supply voltages. However, running the test with several different supply voltages may not always be feasible. Nevertheless, diagnosis can still extract useful information. If a failure can only be explained by multiple fault locations, the chip is probably a slow chip due to parameter variations. Similarly, defective chips can be further characterized. If the diagnosis results of several failing chips always point to the same unique fault location, it is very likely that a systematic problem in the manufacturing process exists that must be fixed. In case of unique but different fault locations, random defects may be responsible for the failures.

Although delay fault diagnosis has been a topic of interest in recent years [2, 15–25], the underlying assumptions of the existing approaches do not exactly match the needs of HDD diagnosis. Our experimental analysis (shown later in Table II) revealed that HDDs typically propagate to very few outputs. Thus, diagnosis must be based on very few failing response

978-1-7281-9113-3/20/$31.00 ©2020 IEEE

bits. Furthermore, process variations can cause different observed failures even for the same HDD. This prevents an efficient pruning of candidate lists, and the reliability analysis described above will not lead to clear results. To develop an efficient diagnosis technique for HDDs is therefore an extremely challenging task. The diagnosis technique presented in this paper is specifically tailored to HDD diagnosis. It combines the variation aware scoring introduced in [26] with a new procedure for tracing the fault effect back to culprits. The backtracing procedure exploits additional timing information to reduce the candidate list.

The rest of the paper is organized as follows. Section II introduces FAST, diagnostic testing as well as the models for failure data. Section III introduces circuit and simulation models and gives an overview over the complete HDD diagnosis flow. Section IV details the new backtracing method used to generate initial HDD candidate lists. Section V details the new variation-aware scoring method to rank the HDD candidates. Section VI presents the experimental results and Section VII concludes this paper.

II. Diagnostic Faster-Than-At-Speed Testing

Logic diagnosis is performed on fail logs generated by *automated test equipment* (ATE) [27]. The fail log contains information on unexpected responses captured after applying certain test patterns. To reduce response data in high-volume testing, FAST can be combined with test response compaction [28, 29]. While highly compacted responses are sufficient for simple pass-fail decisions, logic diagnosis becomes much more challenging [30]. In this work, we assume that uncompacted response data is available. This can be readily achieved by bypassing any response compaction logic [31] or by compaction schemes that allow the restoration of uncompacted response data [32].

The failure information obtained by FAST is different from that of a traditional at-speed delay test. An at-speed delay test is applied using a single frequency matching the functional clock speed of the design. In FAST, various subsets of delay tests are performed with multiple capture times. Fig. 2 shows an example of a test where a single delay test is applied two times and failure information is collected at two capture times c_{nom} and c_{FAST}. On the one hand, since c_{FAST} is earlier than the nominal circuit delay, outputs of long paths (o_2) need to be masked. On the other hand, the delay defect is observable at output o_0 at time c_{FAST}, but its fault effect has already vanished at time c_{nom}. Since a test has different outcomes at different clock frequencies, the information on capture times needs to be included in the fail log.

We model the FAST fail log in the way described below.

Definition 1. *Let T be the set of all delay tests (test pattern pairs), and O be the set of all* pseudo-primary outputs *(PPOs). An* observed failure *$f \in F$ is a tuple (t, o, c) consisting of a delay test $t \in T$, a PPO $o \in O$, and a capture time c. The* fail log *F is the set of all observed failures.*

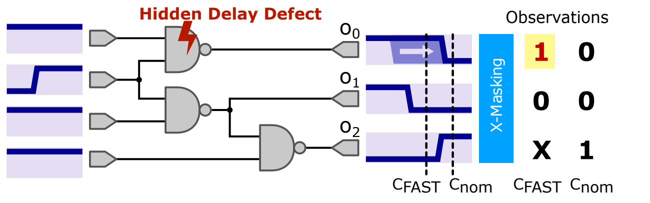

Fig. 2. Faster-than-at-speed testing with one test and two capture times discovering a hidden delay defect.

The pseudo-primary outputs are the signals that are captured by scan cells. The test in Fig. 2 generates a single entry $(t, o_0, c_{\text{FAST}})$ in the fail log. In practice, many more capture times are used for a few thousands of test patterns to obtain a reasonable coverage. A more realistic fail log looks like the following:

$$F = \{(t_{10}, o_8, 1.2\,\text{ns}), (t_{10}, o_{12}, 1.3\,\text{ns}), \\ (t_{16}, o_8, 1.2\,\text{ns}), (t_{16}, o_8, 1.3\,\text{ns})\}.$$

F shows that for test t_{10} a wrong value was observed at PPO o_8 at capture time $1.2\,\text{ns}$ and at PPO o_{12} at time $1.3\,\text{ns}$. Furthermore, test t_{16} caused a fail at scan cell o_8 that was captured both at time $1.2\,\text{ns}$ and $1.3\,\text{ns}$. It is obvious that one test can lead to wrong responses at many scan cells. Unique in FAST is that a single scan cell may capture for the same test a wrong response at multiple times and therefore lead to multiple entries in the fail log.

The masking performed during FAST ensures that the fail log contains only entries with capture times greater than the expected latest stabilization time $\text{LST}(t, o)$ of the PPO o for a delay test t. In the fail log F above for PPO o_8, both $\text{LST}(t_{10}, o_8)$ and $\text{LST}(t_{16}, o_8)$ are less than $1.2\,\text{ns}$. For PPO o_{12} the expected stabilization time is $\text{LST}(t_{10}, o_{12}) < 1.3\,\text{ns}$. In general, all observations before the expected LSTs are masked by FAST and are not available for diagnosis.

Observed failures in the fail log are not necessarily caused by a defect in the device under test. Process variations alone may delay some LSTs at the PPOs so much as to cause failures to get logged during the test. From the observed failures alone it is not apparent whether they were caused by a spot defect or by random variations. Logic diagnosis based on the fail log with an attempt to locate a fitting HDD candidate can give important insights into this question.

The fail log F generated by the ATE together with the gate-level netlist and timing information is input to the novel HDD diagnosis approach described below.

III. HDD Diagnosis Overview

Similar to previous logic diagnosis approaches for other faults and defects [26, 33], the proposed HDD diagnosis approach also consists of two main phases: *effect-cause backtracing* and subsequent *cause-effect fault simulation*. The effect-cause backtracing phase is used to identify the most suspicious circuit structures. The subsequent cause-effect inject-and-validate phase uses diagnostic fault-simulation to find the candidates whose behaviors best match the observations in the fail log.

Both phases operate on timing simulation data generated on demand by a high-performance waveform-based timing simulator [34, 35]. In preparation for timing simulation, the combinational portion of the circuit is extracted from the design and all scan cells are replaced by pairs of pseudo-primary inputs (PPIs) and pseudo-primary outputs (PPOs). The gates in the combinational portion of the *circuit under diagnosis* (CUD) are topologically ordered to facilitate the propagation of simulation data from PPIs to PPOs. The basic unit of computation is a waveform that contains the complete switching history of a signal line. First, the waveforms at PPIs are initialized with the launch transitions corresponding to a delay test pattern. The remaining waveforms in the combinational portion are calculated level by level and by using the nominal pin-to-pin and interconnect delays as illustrated in Fig. 3. As this computation progresses, the waveforms store all occurring transitions and glitches which are then available for all internal signals and PPOs.

The simulation model calculates and stores separate waveforms for fan-out stems and all fan-out branches. This is used to facilitate fine-grained backtracing and fault injection. Every stored waveform corresponds to a potential HDD candidate location. We hereafter refer to these locations also as *signal lines* $l \in L$.

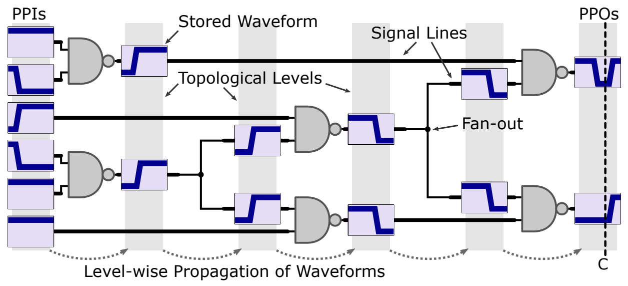

Fig. 3. Principle of waveform-based timing simulation of circuit.

A. Phase 1: Backtracing FAST Observations

Effect-cause backtracing or critical path tracing [17, 26, 36, 37] is a very popular and efficient technique used in many diagnosis approaches. Previous backtracing techniques usually operate on the results of multi-valued logic simulation [38] to determine sensitized paths. The limited expressiveness of multi-valued logic simulation can lead to overly pessimistic results as it may predict more unknowns or 'X'-values than actually present in the circuit [39]. When simulating the propagation of signal transitions using a 6-valued $\boldsymbol{H}_6$ algebra [38], for instance, the temporal order of transitions arriving at a re-convergence point is unknown. The simulator must assume that hazards may be generated at such points to include all potential propagation paths in its result. During backtracing, this pessimistic inclusion of potential propagation paths lead to a larger number of initial defect candidates. If responses from multiple failing delay tests are available, defect candidate numbers can be reduced again by intersection [26], however, this method becomes ineffective for the limited amount of failure information available for HDDs. For this reason, backtracing in the proposed HDD diagnosis approach uses timing simulation data to better estimate sensitized paths.

All observed failures are traced back to visit all potential HDD candidate locations within the combinational portion. In addition to the locations, the fault-free timing simulation data also yields information on potential fault polarities and fault sizes. The result of this first phase is a list of potential HDD candidates sorted by the number of visits to their particular locations starting from different observed PPO failures. These HDD candidates will be evaluated by fault simulation in the second phase.

B. Phase 2: HDD Candidate Simulation

Similar to previous inject-and-validate based diagnosis methods [40–42], each HDD candidate is injected into the simulation model one-by-one and simulated. HDD diagnosis uses full timing simulation for each candidate to predict its fault effects at PPOs. These predictions therefore include the effects of all re-convergencies and hazards within the circuit. The fault simulation results are compared with the observed failures in the fail log to calculate matching scores. Since the simulation is performed with nominal timing and the observations in the fail log are potentially altered by process variations, the variation-tolerant scoring from [26] is adapted.

Finally, the HDD candidates are sorted by their scores to generate a ranked list as the final HDD diagnosis result.

IV. Backtracing FAST Observations

The goal of backtracing is to prune the vast search space of all possible HDDs down to an initial set of HDD candidates for further evaluation by diagnostic fault simulation. This phase does not require costly fault simulation and avoids any explicit enumeration of sensitization or propagation paths. Therefore, the computational resources needed for backtracing are only slightly more than those required for a fault-free timing simulation of all considered delay tests.

Let $t \in T$ be a delay test that shows some observed failures in the given fail log. Test t is simulated with full (nominal) timing, yielding for each signal line $l \in L$ a waveform with all expected transitions over time. In particular, we obtain the expected $\mathrm{LST}(t, o)$ for all PPOs $o \in O \subset L$ as these are just the times of the latest transition in each of their waveforms.

Let $f = (t, o, c)$ be an observed failure for test t at PPO o at time c. Such an observed failure implies that the actual

$\mathrm{LST}^*(t,o)$ of PPO o is later than c, while the expected $\mathrm{LST}(t,o)$ is earlier than c. For this failure to occur, the latest transition of PPO o must have been delayed by a HDD by at least $c - \mathrm{LST}(t,o)$. As mentioned previously, failures might have been observed at multiple capture times at the same PPO o for the same test t.

Definition 2. *Let T be a set of delay tests, O the set of all PPOs, and F a fail log. The* minimum required delay *$\mathrm{MRD}(t,o)$ for a test $t \in T$ and PPO $o \in O$ is defined as:*

$$\mathrm{MRD}(t,o) = \max\{c - \mathrm{LST}(t,o) | (t,o,c) \in F\}.$$

If there is no observed failure involving t and o, then $\mathrm{MRD}(t,o) = 0$.

The fail log F is converted to a set of MRDs for each failing PPO and its test. If multiple observed failures exist for a PPO o and a test t, only one MRD is included since it covers all observations at o for test t for the purpose of backtracing.

A. MRD *Back-Propagation*

For each non-zero $\mathrm{MRD}(t,o)$, an individual back-propagation is conducted to obtain a $\mathrm{MRD}(t,l)$ for each internal signal line $l \in L$. First, we initialize $\mathrm{MRD}(t,l) = 0$ for all signal lines $l \neq o$. All cells and all fan-outs in the circuit are processed in reverse topological order. Whenever a non-zero $\mathrm{MRD}(t,l_o)$ is present at an output line l_o of a cell or a fanout, new $\mathrm{MRD}(t,l_i)$ values are calculated at the respective input lines l_i. For the sake of clarity, we will only discuss this calculation for fan-outs and primitive one- and two-input gates in this paper. More complex cells are supported by decomposing them into these primitives.

1) Fan-Outs: A fan-out is a signal that connects a single driver to multiple receiving gates. During back-propagation, each branch of a fan-out as well as its stem are regarded as separate internal signal lines with separate MRD values.

If a fan-out has a non-zero MRD on at least one of its branches, the stem MRD will be set to the minimum non-zero MRD among its branches. The reason is that this delay is sufficient to cause the observed failure that is currently back-propagated. In general, an additional delay on a fan-out stem propagates to all branches of the fan-out and may be observed at multiple PPOs via paths of different lengths. Therefore, back-propagation may visit a fan-out multiple times from different PPOs with observed failures and with different MRDs. Even though the minimum MRD is propagated to the fan-out stem each time, the largest MRD will eventually be back-propagated into the input cone of the fan-out. After all back-propagations have finished, the maximum MRD on each signal line will be used for HDD candidate generation (see Section IV-B). This ensures that the HDD candidates are large enough to cause all observed failures across all fan-out branches.

2) Interconnects and One-Input Cells: Interconnects with a specified transport delay as well as one-input cells such as buffers and inverters are transparent with respect to MRD back-propagation. Any MRD at their output is simply copied to their input, because the required delay-shift of the last transition is independent of the transport or the pin-to-pin delay of the respective cell.

3) Two-Input Cells: The back-propagation at multi-input gates is more complex because of possible non-robust [43] sensitizations. If the signal at the off-path input of a two-input gate is stable and has no transitions, the MRD is propagated along the sensitized path in the same way as with one-input cells. If both inputs have transitions, robust and non-robust sensitizations need to be distinguished as shown in Fig. 4.

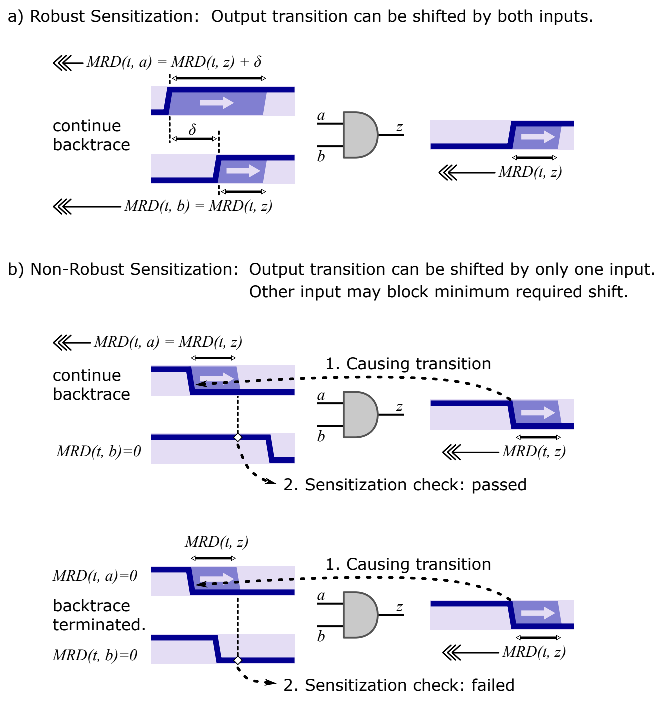
a) Robust Sensitization: Output transition can be shifted by both inputs.
MRD(t, a) = MRD(t, z) + δ
continue backtrace
δ
a
b
z
MRD(t, z)
MRD(t, b) = MRD(t, z)
b) Non-Robust Sensitization: Output transition can be shifted by only one input. Other input may block minimum required shift.
MRD(t, a) = MRD(t, z)
1. Causing transition
continue backtrace
MRD(t, b)=0
MRD(t, z)
2. Sensitization check: passed
MRD(t, z)
1. Causing transition
MRD(t, a)=0
backtrace terminated.
MRD(t, b)=0
MRD(t, z)
2. Sensitization check: failed

Fig. 4. MRD back-propagation at an AND-gate for robust sensitization (a) and non-robust sensitization (b).

In the robust case (Fig. 4a), the final value at the output is the controlled value of the gate (logic 1 for an AND-gate). The position of the latest transition at the output can therefore be altered by delaying the latest transition at either input. Let δ be the time difference between the two LST at the inputs. The input signal with the latest LST gets assigned the MRD from the output signal. The input signal with the earlier LST gets assigned $\mathrm{MRD} + \delta$.

In the non-robust case (Fig. 4b), the final value at the output is the uncontrolled value of the gate (logic 0 for an AND-gate). The LST at the output can only be altered by one of the inputs. This input is determined by analyzing which transition at the inputs directly caused the latest output transition. Usually, it is the last transition at an input during which the gate was non-robustly sensitized through the other input. In the example of Fig. 4b, this is the input a of the AND-gate. Next, the latest transition on the on-path input is shifted by $\mathrm{MRD}(t,z)$ and the sensitization condition is re-checked. If the off-path input has still a sensitizing value (like in the upper case in Fig. 4b), a non-zero MRD is assigned to the on-path input signal and back-propagation continues. If the off-path input transitions to a non-sensitizing value too soon (lower case in Fig. 4b), the propagation stops, since the output LST cannot be shifted by the required $\mathrm{MRD}(t, z)$ by any change to one input signal.

4) Limitations of MRD *Back-Propagation:* There are cases of limitations in which MRD back-propagation may fail to include the actual fault-site.

The first limitation arises when a fault-effect propagates through the circuit as a hazard or glitch. Since back-propagation is based on altering transitions that already exist in the fault-free case, any additional hazards caused by the HDD in the CUD are not considered.

The second limitation is possible if a fault-effect re-converges at a gate. In many cases (e. g., if the reconversion point is robustly sensitized), back-propagation will still follow some of the propagation paths and still reach the actual fault-site. However, if the reconversion point is non-robustly sensitized and a fault-effect is required on the off-path signal to obtain the output MRD, back-propagation may stop prematurely.

The aforementioned limitations are common and well-discussed challenges for critical path tracing [36]. Overcoming these limitations require a more sophisticated (and computationally more expensive) analysis. However, the particular situations in which these limitations impact the diagnosis results are rather rare. We have observed that in most cases, backtracing still works via other test patterns and observed failures and the actual fault-site is still reached via other propagation paths. Furthermore, once a fault-site is marked as a candidate, the following fault-simulation and candidate scoring will consider all hazards and re-convergencies and the final score will not be affected by the limitations of back-propagation.

B. Initial Candidate Set Generation

The results of all individual back-propagations are now combined to generate an ordered list of the most probable HDD candidates. Each HDD candidate consists of a location l, a polarity (slow-to-rise or slow-to-fall), and a delay or size. First we describe how to obtain likely candidate locations, and then we explain how to calculate polarities and delays to form the HDD candidate list.

For each $\mathrm{MRD}(t, o)$, we define two sets of candidate signal lines as:

$$L_{cand}(t, o) = \{l \in L | \mathrm{MRD}(t, l) = \mathrm{MRD}(t, o)\},$$
$$L'_{cand}(t, o) = \{l \in L | \mathrm{MRD}(t, l) > \mathrm{MRD}(t, o)\},$$

with all $\mathrm{MRD}(t, l)$ being obtained by back-propagating $\mathrm{MRD}(t, o)$. $L_{cand}(t, o)$ contains all signal lines l whose $\mathrm{MRD}(t, l)$ does not contain any slack, and $L'_{cand}(t, o)$ contains all signal lines with additional slack in their $\mathrm{MRD}(t, l)$ originating from robust sensitizations (input a in Fig. 4a).

For each signal line $l \in L$ in the circuit, we further define hit-counts $h(l)$ and $h'(l)$ as the numbers of appearances of l in all $L_{cand}(t, o)$ and $L'_{cand}(t, o)$, respectively:

$$h(l) = \sum_{(t,o,c) \in F} |\{l\} \cap L_{cand}(t, o)|,$$
$$h'(l) = \sum_{(t,o,c) \in F} |\{l\} \cap L'_{cand}(t, o)|.$$

The list of candidate signal lines is composed of two parts. First, all $l \in L$ with $h(l) > 0$ are added in descending order of h. Then, all $l \in L$ with $h(l) = 0$ and $h'(l) > 0$ are appended in descending order of h'.

Let l be a candidate signal line. The fault size $\delta(l)$ of the HDD candidate on l is determined by taking the maximum of all $\mathrm{MRD}(t, l)$ obtained during backtracing. This is a lower bound for the actual HDD fault size and ensures that the fault effect of the HDD candidate can propagate to all observation points reported in the fail log.

In addition to the calculation of $\mathrm{MRD}(t, l)$, back-propagation also records the polarity of the shifted transitions. As a signal line l may be reached by back-propagation multiple times, back-propagation may record only falling transitions, only rising transitions or both falling and rising transitions being shifted by the minimum required delay. If only falling transitions were shifted, the corresponding defect is likely of *slow-to-fall* (STF) polarity. In this case, an HDD candidate $\mathrm{STF}[l, \delta(l)]$ is added to the candidate list. If only rising transitions were shifted, a *slow-to-rise* HDD candidate $\mathrm{STR}[l, \delta(l)]$ is added to the candidate list. If both falling and rising transitions were shifted at signal line l while back-propagating from different observed failures, both $\mathrm{STF}[l, \delta(l)]$ and $\mathrm{STR}[l, \delta(l)]$ are added to the candidate list.

The list of initial HDD candidates, sorted by $h(l)$, is given to a fault simulator for final scoring. Our experiments show that among all successful candidate list generations (i. e., the real culprit is indeed among the candidates), the real culprit is almost always among the first few hundred candidates.

V. HDD Candidate Timing Simulation

The initial HDD candidates are now explicitly simulated. Each HDD candidate is injected into the circuit by adding its delay to the fault-free delay of the affected cell. All delay tests that detected the culprit in the CUD are applied to the modified circuit to calculate the potentially faulty waveforms at all PPOs. These predicted waveforms include the effect of hazards, race conditions and re-convergencies that may be present in the CUD. The simulated waveforms at the PPOs are analyzed at the FAST capture times and compared to the observations in the fail log. A score is assigned to each candidate based on the confidence in the simulation result and its similarity to the observed behavior.

A. *Confidence Estimation for Variation Tolerance*

Fault simulation is performed with nominal timing while the CUD is subject to process variations. Therefore, the observations reported in the fail log may not completely match with the simulation even for the best HDD candidate. This issue is addressed with a variation-aware matching approach from [26] summarized in the following for completeness of the paper.

Fig. 5 illustrates the basic idea of evaluating the values of and the confidences in predicted responses. The waveform at a PPO o is a step-function that alternates between 1 and 0 according to the logic values of the signal over time. This step-function is multiplied with a Gaussian probability density function with its mean at a capture time c and a chosen standard deviation σ. The area A of the resulting product gives the probability that the scan cell at PPO o in the chip captured a logic 1. The predicted logic value at the PPO o is 1 if $A > 0.5$ and 0 otherwise. The confidence in the predicted logic value is given by $|2 \cdot (A - 0.5)|$. For example, the confidence is 0.0 if $A = 0.5$ and 1.0 if $A = 0.0$ or 1.0.

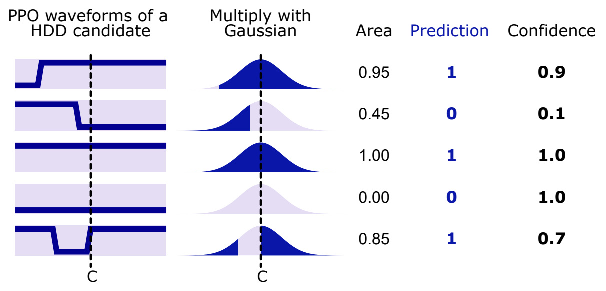

Fig. 5. Prediction of a captured value at time c and the estimation of simulation confidence [26].

The standard deviation σ determines the sensitivity of the confidence prediction to nearby transitions. A reasonable estimation of σ is given by the expected standard deviation of the LST at the PPOs of the circuit under variation. This standard deviation can be determined by running about 100 monte-carlo simulations and fitting a normal distribution to a histogram of LSTs at the PPOs [26].

B. *HDD Candidate Scoring*

Each HDD candidate is assigned a score to reflect how well its simulated PPO values predict the observations reported in the fail log. Similar to [26], the total score of a candidate is the sum of individual score contributions for each response bit.

For a response bit, there are four possible situations, commonly known as TPSP, TPSF, TFSP, and TFSF.

- TFSF (*Tester-Fail Simulation-Fail*): The fail log contains an observed failure and the simulation also predicts a failure.
- TFSP (*Tester-Fail Simulation-Pass*): The fail log contains an observed failure but the simulation predicts a correct value.
- TPSF (*Tester-Pass Simulation-Fail*): No failure was observed but the simulation predicts a failure.
- TPSP (*Tester-Pass Simulation-Pass*): No failure was observed and the simulation also predicts a correct value.

Both TPSP and TFSF contribute positively to the total score, i. e., the simulated candidate HDD is more likely to correspond to the actual defect. Contributions of TFSP and TPSF show mismatches that make a candidate less likely to explain the observed behavior.

We associate a partial score value with each of these four situations: s_{TPSP}, s_{TPSF}, s_{TFSP}, s_{TFSF}. Each value is the sum of the simulation confidences for each response bit in the respective matching category. For example, the simulator predicts a faulty value for a test t, PPO o, and capture time c with a confidence of 0.8. If the fail log contains an entry (t, o, c), a value of 0.8 is added to s_{TFSF}. If the fail log does not contain such an entry, 0.8 is added to s_{TPSF}. Conversely, if the simulator predicts a fault-free response bit, the confidences are added to s_{TFSP} or s_{TPSP}, respectively.

The goal is to order all HDD candidates by their estimated probabilities of being the actual culprit. This order is defined by a single final score s for each candidate that can be calculated with a formula of the general form:

$$s = s_{\text{TFSF}} + \alpha \cdot s_{\text{TFSP}} + \beta \cdot s_{\text{TPSF}} + \gamma \cdot s_{\text{TPSP}}.$$

In practice, $s_{\text{TF}} = s_{\text{TFSF}} + s_{\text{TFSP}}$ is always very close to the number of observed failing outputs, and $s_{\text{TP}} = s_{\text{TPSF}} + s_{\text{TPSP}}$ is close to the number of passing test response bits. Therefore, both s_{TF} and s_{TP} do not vary significantly between the different HDD candidates and are not very useful for ranking. If we treat s_{TF} and s_{TP} as constants within a ranking of candidates, the general score formula can be simplified as follows. First, the constants are subtracted from all scores without affecting the ranking:

$$s = s_{\text{TFSF}} + \alpha s_{\text{TFSP}} + \beta s_{\text{TPSF}} + \gamma s_{\text{TPSP}} - \alpha s_{\text{TF}} - \gamma s_{\text{TP}}.$$

Through simple re-arrangements, two partial scores can be eliminated:

$$\begin{aligned} s &= s_{\text{TFSF}} - \alpha \cdot s_{\text{TFSF}} + \beta \cdot s_{\text{TPSF}} - \gamma \cdot s_{\text{TPSF}} \\ &= (1 - \alpha) \cdot s_{\text{TFSF}} + (\beta - \gamma) \cdot s_{\text{TPSF}}. \end{aligned}$$

Without changing the ranking, we can divide all scores by the constant $(1 - \alpha)$. Furthermore, we can write the constant $-(\beta - \gamma)/(1 - \alpha)$ simply as ω to yield:

$$s = s_{\text{TFSF}} - \omega \cdot s_{\text{TPSF}}.$$

The weight ω determines by how much simulation mispredictions degrade the ranking of HDD candidates. We will show in our experiments in Section VI that using the same ω across various diagnosis cases leads to low-quality rankings in some cases. The reason is that in some diagnosis cases, the fault-simulator produces high values of s_{TPSF} relative to s_{TFSF} for most HDD candidates, while in other diagnosis cases, s_{TPSF} is rather low for most HDD candidates. A weight ω that result in a good ranking in the latter case may degrade the HDD candidate corresponding to the actual culprit too much in the former case. To solve this problem, we propose a weight ω^* that scales a normalized value of s_{TPSF}. Let $s_{\text{TPSF}}^{\max}$ be the maximum s_{TPSF} among all HDD candidates in a diagnosis case. The final score then becomes:

$$s = s_{\text{TFSF}} - \omega^* \cdot \frac{s_{\text{TPSF}}}{s_{\text{TPSF}}^{\max}}.$$

This way, the final score is always reduced by a value between 0 and ω^*, independent from the absolute values of s_{TPSF}. We will demonstrate how to choose the weight ω^* for optimal ranking performance in Section VI-C.

C. Candidate Simulation and Ranking

Phase 1 of the diagnosis procedure generated a list of HDD candidates that are sorted by the number of times its location was visited by backtracing. These HDD candidates are fault-simulated one by one, starting from the candidate with the highest hit-count. Each candidate is simulated with all failing tests that are present in the fail log. With the simulation results, $s = s_{\text{TFSF}} - \omega^* \cdot \frac{s_{\text{TPSF}}}{s_{\text{TPSF}}^{\max}}$ for each candidate is calculated. In the calculation of the scores, both passing and failing response bits of the failing delay tests are considered.

Simulating all passing tests will yield a more accurate s_{TPSF}, which may further improve the ranking. However, this adds considerable fault-simulation runtime that may become infeasible for large designs.

VI. Experimental Results

The goals of our experiments were to evaluate the performance in terms of results and runtime of the proposed HDD diagnosis algorithm. They were conducted on large ITC'99 benchmark circuits [44] that were synthesized using the SAED 32 nm technology library and a standard commercial tool flow. For each circuit, the FAST-ATPG approach from [12] was used to generate test pattern sets, observation times as well as a list of all HDD detected by the generated test sets. To generate a diagnostic test case, we randomly picked an HDD from the set of all detected HDD, injected it into the circuit, and simulated the FAST procedure to generate the fail log. Note that all culprits are actual hidden delay defects that cannot be detected with traditional at-speed delay testing as they do not break the timing of the CUD. To the best of our knowledge, no logic diagnosis approach in the literature targets diagnosis of HDDs. Therefore, it is not possible to compare our results to any other diagnosis algorithms.

A. Statistics on benchmark circuits and FAST fail logs

The basic statistics on benchmark circuits, test sets, and test timing are shown in Table I. Column $|L|$ gives the number of signal lines, i. e., the number of possible HDD locations, in the circuit. $|T|$ shows the number of delay tests generated by FAST-ATPG, $|O|$ shows the number of PPOs, $|C|$ shows the number of capture times, and #HDDs shows the number of detected hidden delay defects detected by the FAST test. The number of detected HDDs vary among benchmark circuits and depend on the prevalence of HDDs in the circuits and the efficacy of FAST. For the remaining experiments, we randomly picked 1000 defects from detected HDDs for each benchmark circuit.

TABLE I
Statistics of benchmark circuits and test sets.

	$\|L\|$	$\|O\|$	$\|T\|$	$\|C\|$	#HDDs
b14	23.0k	274	2608	9	8.6k
b15	17.5k	499	1487	11	1.4k
b17	53.6k	1459	1916	11	3.2k
b18	166.7k	3210	3365	8	29.8k
b19	319.6k	6406	5056	7	37.9k
b20	44.0k	470	3253	8	9.7k
b21	43.6k	470	3440	8	10.1k
b22	66.0k	693	3843	8	14.4k

The randomly picked HDDs are simulated to generate the fail log for each culprit. Each case was simulated with nominal timing and the fail logs do not contain any effects from process variations. Due to space limitations, we will address the effect of process variations on HDD diagnosis results in more detail as part of our future work. Table II shows some statistics on the generated fail log data. Column %Xs shows the percentage of bits masked because they were captured before the expected LST of the PPO. Column *#resp.bits* shows the total number of unmasked response bits generated by the FAST test. This number is much larger than that of traditional at-speed testing, because the complete test set T is applied $|C|$ times.

Column *median*$|F|$ shows the median number of observed failures in each fail log. Compared to the total number of available response bits, the number of observed failures is extremely small. For instance, half of the fail logs for b22 showed less than 2 entries. This shows how elusive HDDs are even with rather aggressive diagnostic fail-data collection.

TABLE II
Statistics of test responses and fail logs.

	%Xs	#resp.bits	median$\|F\|$
b14	0.5%	6.4M	3.0
b15	0.2%	8.1M	3.9
b17	0.1%	30.7M	3.0
b18	0.1%	86.4M	6.8
b19	0.1%	226.5M	4.6
b20	0.2%	12.2M	2.2
b21	0.2%	12.9M	2.3
b22	0.1%	21.3M	1.7

B. Backtracing FAST Observations

The fail logs were then analyzed by our backtracing method to generate an initial set of HDD candidates. Table III shows the results. Column $2|L|$ gives the number of potential delay faults (slow-to-rise and slow-to-fall at every location) from which the initial candidates are selected. Column *success* shows the percentage of cases in which the real culprit was indeed included in the initial candidate list. In the vast majority (95.4%) of all cases, the real culprit was successfully included in the initial candidate list. The very few cases where backtracing failed to reach the fault site can be attributed to the backtracing limitations discussed in Section IV-A4. Column *top* shows the percentage of successful cases where the culprit is among the HDD candidates with the largest number of visits from MRD back-propagations. In other words, the real culprit was ranked on top of the initial candidate list (among other candidates with the same number of visits). We observed that this was the case in 81.6% of all experiments. This underscores the observation that complications from re-convergencies and hazards play a rather insignificant role during the first phase of diagnosis. Only b18 and b19 show a top percentage of 54% and below, but still, in over 92% of all cases, the real culprit was included in the list of initial candidates.

Column *med.rank* shows the median position of the real culprit in the HDD candidate list and column *typ.ranks* show the worst rank among the best-ranked 99.5% of all diagnosis cases. We observed that in more than half of the successful cases the real culprit can be found among the top-33, and in 99.5% of the successful cases, the real culprit can be found among the top-1000 in the initial candidate list. In relation to the number of possible candidate locations and polarities of $2|L|$, backtracing is very effective in narrowing down to a manageable candidate list for fault simulation.

Column *avg.RT* shows the average runtime for backtracing one case. The runtimes were measured with a single-threaded implementation on a 3.6 GHz Intel Xeon CPU. We observed that backtracing only took a few seconds for smaller circuits and up to a few minutes for b19.

TABLE III
BACKTRACING PERFORMANCE AND HDD CANDIDATE STATISTICS.

	$2\|L\|$	success	top	med.rank	typ.ranks	avg.RT
b14	46.1k	97.7%	91.2%	14.3	≤293	3.3s
b15	35.1k	96.4%	91.4%	23.8	≤160	4.9s
b17	107.2k	95.7%	93.2%	26.3	≤160	8.6s
b18	333.4k	93.2%	39.5%	69.0	≤2942	38s
b19	639.2k	92.0%	53.7%	47.0	≤3646	2.6m
b20	88.1k	96.1%	94.6%	25.2	≤177	3.4s
b21	87.3k	96.0%	94.8%	26.1	≤290	3.9s
b22	132.0k	96.3%	94.4%	27.9	≤267	3.8s
avg.		95.4%	81.6%	32.5	≤992	

From each initial candidate list, we considered at most 2000 (5000 for b18, b19) candidates for fault simulation. These are rather conservative numbers to include also the hard-to-diagnose cases. In most cases, the list of initial candidates was much shorter, and as shown previously, the cut-off limit could be reduced without affecting most of the results.

C. Rank Qualities vs. Scoring Weights

In this section, we introduce the notion of rank quality and explore the influence of the weights ω and ω^* on it. These calculations are not part of the HDD diagnosis procedure and do not have to be performed when deploying our algorithm.

Each initial HDD candidate was fault-simulated to calculate and store its s_{TFSF} and s_{TPSF}. From this database of partial scores, rankings were generated and statistically evaluated using various weights.

To quantify the quality of the rankings, we first define a *cumulative rank function* $\text{crf}(n) : \mathbb{N}^+ \to [0,1]$ as denoting the ratio of identified culprits after at most n picks. Fig. 6 plots the values of this function across all cases in all benchmark circuits for four different weights. The x-axis shows the number of picks n and the y-axis is $\text{crf}(n)$ given as percentage. The steeper the curve, the better the rankings. The quality of the rankings are rather low if s_{TPSF} is ignored completely ($\omega = 0$) and only s_{TFSF} is used. They improve significantly with $\omega = 0.02$, but degrade again with $\omega = 0.5$. A scoring using normalized s_{TPSF} with $\omega^* = 2$ yields slightly better quality than $\omega = 0.02$. We further define a numeric *rank quality* q as the area under $\text{crf}(n)$ for $n \leq 100$:

$$q = \sum_{n=1}^{100} \text{crf}(n).$$

The best possible rank quality $q = 100$ is achieved when all culprits are ranked as the top candidates. The rank qualities for $\omega = 0,\ 0.02,\ 0.5,\ \omega^* = 2$ are $q = 59.2,\ 63.9,\ 57.4$, and 65.4, respectively.

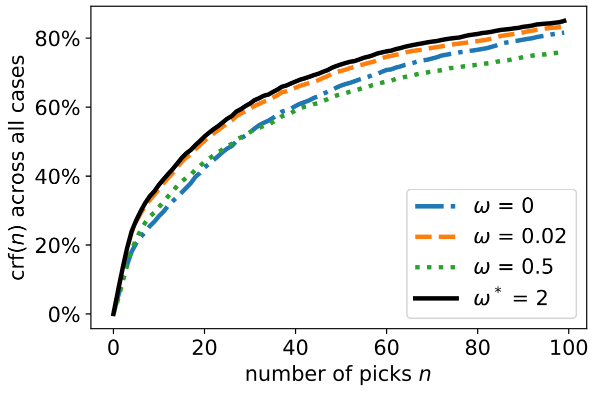

Fig. 6. Ranking results across all benchmarks with different weights.

Fig. 7 shows the rank quality across all cases across varying weights ω and ω^*. The dashed curve shows the rank quality using ω. The best rank quality of $q = 63.9$ is achieved for $\omega = 0.016$. The solid curve shows the rank quality using ω^*. We observe that scaling the normalized s_{TPSF} with a value between 0 and ω^* yields better ranking qualities of up to $q = 65.4$ for $\omega^* = 1.97$.

The scale of the y-axis shows that the differences in rank qualities are small, but normalizing s_{TPSF} improves the qual-

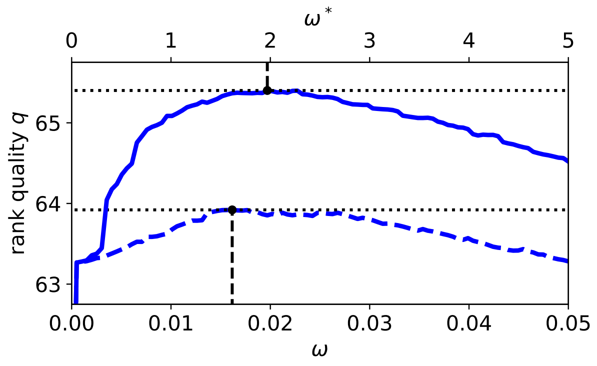

Fig. 7. Ranking quality vs. ω (dashed curve) and vs. ω^* (solid curve).

ity compared to a non-normalized scoring with ω. Weights in the range of $1 < \omega^* < 3$ will yield almost identical results. In the following, $\omega^* = 2$ will be used for generating the final rankings.

D. HDD Diagnosis Performance

Table IV shows the final diagnosis results for each of the benchmark circuits. Column *success* shows the percentage of cases that were successfully diagnosed. The first percentage is the overall success rate which also accounts for failures in backtracing. The second percentage in parentheses shows the success rate among the diagnosis cases where backtracing was successful. The slight decrease in the overall success rate is due to the cut-off limit of 2000 (5000 for b18, b19) for the initial HDD candidate list. Columns *med.rank avg.rank* show the median and average position of the culprit in the final diagnosis call-out, respectively. It can be seen that in half of the cases, the real culprit can be found among the top 17 candidates. There are two main factors that limits the ranking quality. The first is the possibility of having equivalent HDDs (e. g., a HDD in front of and after an inverter) that cannot be distinguished by diagnosis. The second is the limited diagnostic resolution of the used test set. This is evident in particular in the strong correlation between the median ranks in Table IV and the median number of observed failures in Table II. We plan to add diagnostic automatic test pattern generation to FAST for higher resolution as part of our future work.

Column *avg.RT* shows the average runtime of diagnosing one case. The first number shows the overall runtime, the number in parentheses shows the time spend on fault-simulation. While backtracing ran on a single-threaded implementation on a 3.6 GHz Intel Xeon CPU, fault-simulation was assisted by a Nvidia TITAN V GPU with 12 GB memory. Fault-simulation clearly dominated the overall runtime, but still each diagnosis case was solved within a few minutes. The memory requirements for simulations on the GPU never exceeded 2 GB.

TABLE IV
FINAL RANKING STATISTICS USING $\omega^* = 2$.

	success	(sc.only)	med.rank	avg.rank	avg.RT	
b14	97.7%	(100.0%)	10.7	29.8	1.2m	(1.1m)
b15	94.8%	(98.5%)	10.8	69.2	1.0m	(56s)
b17	95.2%	(99.5%)	11.3	19.4	1.1m	(56s)
b18	89.2%	(96.0%)	8.8	65.5	4.9m	(4.3m)
b19	89.7%	(97.8%)	10.5	45.9	7.6m	(5.0m)
b20	95.7%	(99.6%)	27.3	44.6	1.3m	(1.2m)
b21	94.7%	(98.7%)	21.8	39.9	1.4m	(1.3m)
b22	95.3%	(99.1%)	27.5	51.3	1.1m	(1.1m)
avg.	94.0%	(98.6%)	16.1	45.7		

VII. Conclusions

We have presented the very first logic fault diagnosis technique that is able to identify HDD by analyzing fail logs produced by FAST. The technique uses MRD back-propagation to obtain initial HDD candidates which are then scored and ranked using fault-simulation and variation-tolerant observation matching. The experimental results show a success rate of 94% even with very limited amount of failure data. FAST diagnosis established in this work provides the last missing component to rapidly improve yield and reliability by learning from otherwise unknown HDDs.

Acknowledgment

This work was supported in part by the Japan Society for the Promotion of Science (JSPS) under JSPS Grant-in-Aid for Young Scientists #18K18026 and JSPS Grant-in-Aid for Scientific Research (B) #17H01716 as well as by the German Research Foundation (DFG) under grants HE1686/4-1 and WU245/19-1.

References

[1] Y. M. Kim, Y. Kameda, H. Kim, M. Mizuno, and S. Mitra, "Low-cost gate-oxide early-life failure detection in robust systems," in *Proc. IEEE Symp. on VLSI Circuits (VLSIC)*, Jun. 2010, pp. 125–126.
[2] M. Tehranipoor, K. Peng, and K. Chakrabarty, *Test and Diagnosis for Small-Delay Defects*. Springer New York, 2011.
[3] F. Mesalles, H. Villacorta, M. Renovell, and V. Champac, "Behavior and test of open-gate defects in FinFET based cells," in *Proc. IEEE European Test Symp. (ETS)*, May 2016, p. 4A.1.
[4] Y. Wei, F. Baumann, S. Lucarini, K. Bartha, and Z. S. und A. Katnani, "SRAM PFET and NFET super FIN characterization," in *Proc. Int. Symp. for Testing and Failure Analysis (ISTFA)*, Nov. 2017, pp. 140–142.
[5] H. Yan and A. D. Singh, "Experiments at detecting delay faults using multiple higher frequency clocks and results from neighboring die," in *Proc. IEEE Int. Test Conf. (ITC)*, Oct. 2003, pp. 105–111.
[6] M. Amodeo and B. Cory, "Defining faster-than-at-speed delay tests," *Cadence Nanometer Test Quarterly eNewsletter*, vol. 2, no. 2, May 2005.
[7] N. Ahmed and M. Tehranipoor, "A novel faster-than-at-speed transition-delay test method considering IR-drop effects," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 10, pp. 1573–1582, Sep. 2010.

[8] T. Yoneda, K. Hori, I. Inoue, and H. Fujiwara, “Faster-than-at-speed test for increased test quality and in-field reliability,” in *Proc. IEEE Int. Test Conf. (ITC)*, Sep. 2011, p. 2.2.

[9] S. Chakravarty, N. Devta-Prasanna, A. Gunda, J. Ma, F. Yang, H. Guo, R. Lai, and D. Li, “Silicon evaluation of faster than at-speed transition delay tests,” in *Proc. IEEE VLSI Test Symp. (VTS)*, Apr. 2012, pp. 80–85.

[10] S. Hellebrand, T. Indlekofer, M. Kampmann, M. Kochte, C. Liu, and H.-J. Wunderlich, “FAST-BIST: Faster-than-at-Speed BIST targeting hidden delay defects,” in *Proc. IEEE Int. Test Conf. (ITC)*, Oct. 2014, p. 29.3.

[11] M. Kampmann, M. A. Kochte, E. Schneider, T. Indlekofer, S. Hellebrand, and H. Wunderlich, “Optimized selection of frequencies for faster-than-at-speed test,” in *Proc. IEEE Asian Test Symp. (ATS)*, Nov. 2015, pp. 109–114.

[12] M. Kampmann, M. A. Kochte, C. Liu, E. Schneider, S. Hellebrand, and H. Wunderlich, “Built-in test for hidden delay faults,” *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 10, pp. 1956–1968, Oct. 2019.

[13] X. Qian and A. D. Singh, “Distinguishing resistive small delay defects from random parameter variations,” in *Proc. IEEE Asian Test Symp. (ATS)*, Dec. 2010, pp. 325–330.

[14] X. Qian, Chao Han, and A. D. Singh, “Detection of gate-oxide defects with timing tests at reduced power supply,” in *Proc. IEEE VLSI Test Symp. (VTS)*, Apr. 2012, pp. 120–126.

[15] M. Abramovici and M. A. Breuer, “Multiple fault diagnosis in combinational circuits based on an effect-cause analysis,” *IEEE Trans. on Computers*, vol. 29, no. 6, pp. 451–460, Jun. 1980.

[16] H. Cox and J. Rajski, “A method of fault analysis for test generation and fault diagnosis,” *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 7, no. 7, pp. 813–833, Jul. 1988.

[17] P. Girard, C. Landrault, and S. Pravossoudovitch, “Delay fault diagnosis by critical-path tracing,” *IEEE Design and Test of Computers*, vol. 9, no. 4, pp. 27–32, Dec. 1992.

[18] A. Krstic and K.-T. Cheng, *Delay Fault Testing for VLSI Circuits*. Springer, 1998.

[19] H.-B. Wang, S.-Y. Huang, and J.-R. Huang, “Gate-delay fault diagnosis using the inject-and-evaluate paradigm,” in *Proc. IEEE Int. Defect and Fault-Tolerance in VLSI Systems Symp. (DFT)*, Nov. 2002, pp. 117–125.

[20] K. Yang and K.-T. Cheng, “Timing-reasoning-based delay fault diagnosis,” in *Proc. ACM/IEEE Conf. on Design, Automation Test in Europe (DATE)*, Mar. 2006, pp. 418–423.

[21] T. Aikyo, H. Takahashi, Y. Higami, J. Ootsu, K. Ono, and Y. Takamatsu, “Timing-aware diagnosis for small delay defects,” in *Proc. IEEE Int. Defect and Fault-Tolerance in VLSI Systems Symp. (DFT)*, Sep. 2007, pp. 223–234.

[22] W. Cheng, B. Benware, R. Guo, K. Tsai, T. Kobayashi, K. Maruo, M. Nakao, Y. Fukui, and H. Otake, “Enhancing transition fault model for delay defect diagnosis,” in *Proc. IEEE Asian Test Symp. (ATS)*, Nov. 2008, pp. 179–184.

[23] V. J. Mehta, M. Marek-Sadowska, K.-H. Tsai, and J. Rajski, “Timing-aware multiple-delay-fault diagnosis,” *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 2, pp. 245–258, Feb. 2009.

[24] P.-J. Chen, W.-L. Hsu, J.-M. Li, N.-H. Tseng, K.-Y. Chen, W.-P. Changchien, and C. Liu, “An accurate timing-aware diagnosis algorithm for multiple small delay defects,” in *Proc. IEEE Asian Test Symp. (ATS)*, Nov. 2011, pp. 291–296.

[25] H. Tang, T. Tai, W. Cheng, B. Benware, and F. Hapke, “Diagnosing timing related cell internal defects for FinFET technology,” in *Proc. VLSI Design, Automation and Test (VLSI-DAT)*, Apr. 2015.

[26] S. Holst, E. Schneider, M. A. Kochte, X. Wen, and H.-J. Wunderlich, “Variation-aware small delay fault diagnosis on compressed test responses,” in *Proc. IEEE Int. Test Conf. (ITC)*, Nov. 2019, p. 3.1.

[27] M. L. Bushnell and V. D. Agrawal, *Essentials of Electronic Testing for Digital, Memory, and Mixed-Signal VLSI Circuits*. Springer, Nov. 2000.

[28] A. Sprenger and S. Hellebrand, “Divide and compact - stochastic space compaction for faster-than-at-speed test,” *Journal of Circuits, Systems and Computers*, vol. 28, Suppl. 1, p. 1940001, Apr. 2019.

[29] M. U. Maaz, A. Sprenger, and S. Hellebrand, “A hybrid space compactor for adaptive X-handling,” in *Proc. IEEE Int. Test Conf. (ITC)*, 2019, p. 3.3.

[30] S. Holst and H.-J. Wunderlich, “A diagnosis algorithm for extreme space compaction,” in *Proc. ACM/IEEE Conf. on Design, Automation Test in Europe (DATE)*, Apr. 2009, pp. 1355–1360.

[31] L. Wang, C. Wu, and X. Wen, *VLSI Test Principles and Architectures: Design for Testability*. Morgan Kaufmann, 2006.

[32] S. Mitra and K. S. Kim, “X-compact: An efficient response compaction technique,” *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, no. 3, pp. 421–432, Mar. 2004.

[33] R. Desineni, O. Poku, and R. D. Blanton, “A logic diagnosis methodology for improved localization and extraction of accurate defect behavior,” in *Proc. IEEE Int. Test Conf. (ITC)*, Oct. 2006, p. 12.3.

[34] S. Holst, M. E. Imhof, and H.-J. Wunderlich, “High-throughput logic timing simulation on GPGPUs,” *ACM Trans. on Design Automation of Electronic Systems (TODAES)*, vol. 20, no. 3, pp. 37:1–37:22, Jun. 2015.

[35] E. Schneider, M. A. Kochte, S. Holst, X. Wen, and H.-J. Wunderlich, “GPU-accelerated simulation of small delay faults,” *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, pp. 829–841, May 2017.

[36] M. Abramovici, P. R. Menon, and D. T. Miller, “Critical path tracing: An alternative to fault simulation,” *IEEE Design and Test of Computers*, vol. 1, no. 1, pp. 83–93, Feb. 1984.

[37] B. Bosio, P. Girard, S. Pravossoudovitch, and A. Virazel, “A comprehensive framework for logic diagnosis of arbitrary defects,” *IEEE Trans. on Computers*, vol. 59, no. 3, pp. 289–300, Mar. 2010.

[38] J. P. Hayes, “Digital simulation with multiple logic values,” *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 5, no. 2, pp. 274–283, Apr. 1986.

[39] D. Erb, M. A. Kochte, M. Sauer, S. Hillebrecht, T. Schubert, H.-J. Wunderlich, and B. Becker, “Exact logic and fault simulation in presence of unknowns,” *ACM Trans. Des. Autom. Electron. Syst.*, vol. 19, no. 3, Jun. 2014. [Online]. Available: https://doi.org/10.1145/2611760

[40] T. Bartenstein, D. Heaberlin, L. Huisman, and D. Sliwinski, “Diagnosing combinational logic designs using the single location at-a-time (SLAT) paradigm,” in *Proc. IEEE Int. Test Conf. (ITC)*, Nov. 2001, pp. 287–296.

[41] Y. Benabboud, A. Bosio, L. Dilillo, P. Girard, S. Pravossoudovitch, A. Virazel, L. Bouzaida, and I. Izaute, “A fault-simulation-based approach for logic diagnosis,” in *Proc. Design and Technology of Integrated Systems in Nanoscale Era Conf. (DTIS)*, 2009, pp. 216–222.

[42] S. Holst and H.-J. Wunderlich, “Adaptive debug and diagnosis without fault dictionaries,” *Journal of Electronic Testing – Theory and Applications*, vol. 25, no. 4-5, pp. 259–268, Aug. 2009.

[43] C. J. Lin and S. M. Reddy, “On delay fault testing in logic circuits,” *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 6, no. 5, pp. 694–703, Sep. 1987.

[44] F. Corno, M. S. Reorda, and G. Squillero, “RT-level ITC’99 benchmarks and first ATPG results,” *IEEE Design and Test of Computers*, vol. 17, no. 3, pp. 44–53, Jul. 2000.