Variation-Aware Small Delay Fault Diagnosis on Compressed Test Responses

Holst, Stefan; Schneider, Eric; Kochte, Michael A.; Wen, Xiaoqing; Wunderlich, Hans Joachim

Proceedings of the IEEE International Test Conference (ITC'19), Washington DC, USA, 11-15 November 2019

doi: https://doi.org/10.1109/ITC44170.2019.9000143

Abstract: With today's tight timing margins, increasing manufacturing variations, and new defect behaviors in FinFETs, effective yield learning requires detailed information on the population of small delay defects in fabricated chips. Small delay fault diagnosis for yield learning faces two main challenges: (1) production test responses are usually highly compressed reducing the amount of available failure data, and (2) failure signatures not only depend on the actual defect but also on omnipresent and unknown delay variations. This work presents the very first diagnosis algorithm specifically designed to diagnose timing issues on compressed test responses and under process variations. An innovative combination of variation-invariant structural analysis, GPU-accelerated time-simulation, and variation-tolerant syndrome matching for compressed test responses allows the proposed algorithm to cope with both challenges. Experiments on large benchmark circuits clearly demonstrate the scalability and superior accuracy of the new diagnosis approach.

Preprint

Variation-Aware Small Delay Fault Diagnosis on Compressed Test Responses

Stefan Holst*, Eric Schneider‡, Michael A. Kochte§, Xiaoqing Wen*, Hans-Joachim Wunderlich‡

*Kyushu Institute of Technology, 680-4 Kawazu, Iizuka 820-8502, Japan
{holst, wen}@cse.kyutech.ac.jp
‡University of Stuttgart, Pfaffenwaldring 47, 70569 Stuttgart, Germany
schneiec@iti.uni-stuttgart.de, wu@informatik.uni-stuttgart.de
§michael.kochte@gmx.net

***Abstract*—With today's tight timing margins, increasing manufacturing variations, and new defect behaviors in FinFETs, effective yield learning requires detailed information on the population of small delay defects in fabricated chips. Small delay fault diagnosis for yield learning faces two main challenges: (1) production test responses are usually highly compressed reducing the amount of available failure data, and (2) failure signatures not only depend on the actual defect but also on omnipresent and unknown delay variations. This work presents the very first diagnosis algorithm specifically designed to diagnose timing issues on compressed test responses and under process variations. An innovative combination of variation-invariant structural analysis, GPU-accelerated time-simulation, and variation-tolerant syndrome matching for compressed test responses allows the proposed algorithm to cope with both challenges. Experiments on large benchmark circuits clearly demonstrate the scalability and superior accuracy of the new diagnosis approach.**

***Keywords*—small delay defect; logic diagnosis; test compression; process variation; GP-GPU**

I. Introduction

In modern nanometer technologies, timing-related faults have become a major concern in any high-performance or low-power device and their diagnosis is crucial for effective yield learning during production [1]. Modern multi-gate devices, such as FinFETs, are especially prone to timing-related faults as production defects that affect only some of their gates can reduce their driving strength [2]. *Small delay defects* (SDDs) [3] are a popular model to describe such additional delays and considerable amount of research has been conducted on simulation [4–6], test generation [7], and diagnosis [8–11] of SDDs. The additional delay introduced by an SDD is called its *size*. In contrast to transition or gross delay faults [12, 13], this additional delay is usually smaller than the cycle time of their clock domain. SDDs at the same location but with different sizes may lead to different test responses as the fault effects are only observable over sufficiently long paths [14, 15]. This dramatically increases the number of possible faults in the search space for a corresponding logic diagnosis algorithm. Further complications arise when the same SDD of the same size lead to different results in different chips due to manufacturing variations [16, 17]. All of the aforementioned aspects combined render logic diagnosis of SDDs an extremely challenging task.

The fault syndromes obtained through production test are usually highly compressed so as to reduce tester memory demands and test time [1]. For efficient yield learning, a diagnosis algorithm has to operate on compressed signatures only. The trade-off between compression ratios and diagnostic resolutions has been investigated previously [18] and some diagnosis approaches can cope with very high compression ratios [19]. Compressing syndromes from SDDs leads to several complications for logic diagnosis. Fig. 1 illustrates these challenges on a simple design with two scan chains of length three. The effects of the SDD is only observable at pseudo-primary outputs (PPOs) of sufficiently long paths. Even when the propagation path is long enough, the fault effect may still be masked due to reconvergency-induced hazards. The six PPOs are compressed into three signature bits, one for each scan slice. This compression can lead to additional masking as shown in the first (top) signature bit. A diagnosis algorithm has to model the activation and propagation of SDDs accurately enough in order to predict compressed fault signatures from fault candidates under the impact of hazards and process variations [4].

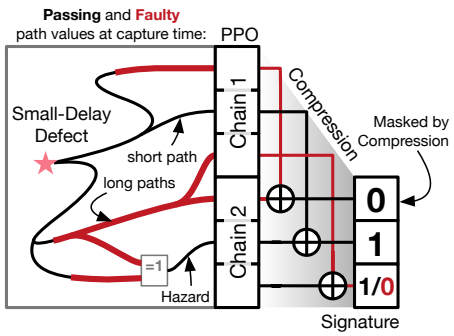

Fig. 1. Space-compacted test response of a SDD syndrome.

Approaches to diagnosing SDDs are mainly based on two distinct principles: *inject-and-validate* and *backtracing* [10].

Inject-and-validate based methods inject candidate faults and use multi-valued logic simulation [8, 20], timing-analysis combined with transition-fault simulation [11] or statistical reasoning [21] to estimate their signatures, which are then compared with the observed responses. Diagnosis approaches that work well with response compression usually focus on stuck-at or transition faults that do not require exact timing models [19]. These approaches usually do not model the precise timing of a circuit due to its computational complexity. Simpler timing models, however, easily lead to a large percentage of mismatching signature bits as shown in Fig. 2a). While only one out of six PPOs differ between the shown transition fault candidate and the SDD in Fig. 1, 33% of signature bits show a mismatch. The reason is that response compactors are designed for high observability and each change in a response bit is reflected in at least one bit of the signature.

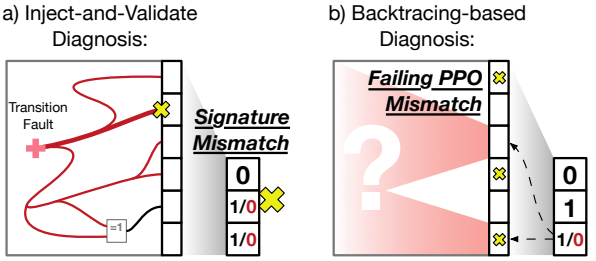

Fig. 2. Common SDD diagnosis techniques are ineffective on compressed syndromes.

Backtracing-based methods [22–25] are designed to trace back sensitized paths from failing PPOs to find candidate SDD locations. Their effectiveness is usually lost when they are directly applied to compressed signatures. This is because a single faulty signature bit can originate from numerous PPOs and a correct signature bit does not imply that all associated PPOs are correct. Fig. 2b) shows a backtracing attempt on the SDD signature from Fig. 1. It can identify only one out of three failing PPOs since the other two failing values are masked by response compression. In addition, another PPO is assumed as possibly faulty, which is totally unrelated to the original SDD.

This paper proposes a novel diagnosis algorithm that can diagnose single SDDs in the circuit under test directly from highly compressed fault signatures which are also influenced by timing variations. The primary purpose of this diagnosis algorithm is to provide additional and better fault candidate data for statistical analysis in volume diagnosis. The diagnosis system is not intended for precision diagnosis, which needs to be fault model agnostic and able to handle multiple faults. In a precision diagnosis context, access to uncompressed response data is usually available and many sophisticated precision diagnosis methods can be found in the literature [8–11]. Here, we focus on finding small delay defects that might be responsible for failing signatures. Our system can easily be used in parallel with other diagnosis approaches to generate more valuable data for volume diagnosis.

The overall diagnosis system is illustrated in Fig. 3. It comprises of two phases. The first phase generates a set of initial SDD candidates using a novel variation-invariant backtracing approach that is based on a fast 6-valued logic simulation. In the second phase, the candidates are scored using *graphics processing unit* (GPU) accelerated timing-accurate small delay fault simulation [5] combined with innovative variation-aware signature matching.

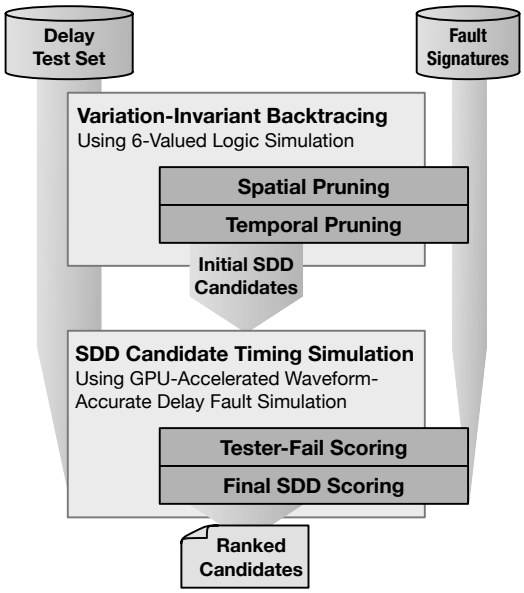

Fig. 3. Proposed SDD diagnosis flow.

The remainder of this paper is organized as follows: Section II discusses test response compression and its implications on SDD diagnosis. Section III introduces the new variation-invariant backtracing-based pruning approach used to focus the diagnosis on the most suspicious parts of the circuit. Section IV proposes a new method for efficiently scoring SDD candidates using GPU-accelerated timing simulation, and the experimental results in Section V show the diagnostic resolution and runtime efficiency of the proposed method.

II. Test Response Compaction and Diagnosis

A. Linear Space Compactors

Linear space compactors, such as parity trees [26] or X-Compact [27], are popular hardware structures to compress test responses. The proposed diagnosis approach works with any linear space compactor. A *space compactor* computes a few signature bits from each test response separately and there are no dependencies among test responses. In a *linear compactor*, each signature bit is always a linear combination (i.e., the parity or XOR-sum) of a certain subset test response bits.

Let $o_1, \ldots, o_n \in O$ be the PPOs of the circuit under diagnosis (CUD). Let $s_1, \ldots, s_m \in S$ be the signature bits, i.e. the output of the linear space compactor. The compactor is completely defined by providing for each signature bit $s_i \in S$ the set of PPOs in its fan-in: $O[s_i] \subseteq O$. Let $t \in T$ be a

delay test and $o_1(t), \ldots, o_n(t)$ its response captured in the PPOs. Its signature bits are then simply: $s_i(t) = \oplus_{o \in O[s_i]} o(t)$ $(1 \leq i \leq m)$.

B. Direct vs. Indirect Diagnosis

In *indirect diagnosis*, first the original test responses (the state of the PPOs) are reconstructed from the signatures, and then, logic diagnosis is performed on the reconstructed data. It is easily combined with any common SDD diagnosis technique. However, test response reconstruction requires the use of error-correcting compactors [27], whose signatures are quite large. Moreover, reconstruction may still fail if the PPOs contain too many failing bits, resulting in invalid information given to the logic diagnosis algorithm.

Direct diagnosis, which is used in this work, resolves the above issues by directly working on compressed signatures [18]. In the inject-and-validate approach, the simulated test responses of each fault candidate are first compressed and then compared with the observed signatures to find a match. Backtracing-based techniques also work on compressed signatures in principle, although at reduced resolution (see Fig. 2b)).

C. Model for Direct SDD Diagnosis

The proposed diagnosis algorithm uses both backtracing and timing simulation. The circuit model consists of the combinational part of the CUD with all scan cells replaced by pseudo-primary inputs (PPIs) and PPOs. The nominal delays of all combinational logic gates and interconnects are obtained from synthesis in standard delay format.

All test infrastructure including the compactor is usually tested separately (e.g., using scan flush tests) and is assumed to be defect-free. During simulations, the diagnosis algorithm calculates the compressed signatures directly from the response at PPOs. Backtracing follows every faulty signature bit s_i back to its corresponding PPOs in $O[s_i]$. Obviously, no timing information is necessary for the compactor structure itself since it is not part of the delay test.

The proposed diagnosis algorithm analyzes responses for two-pattern delay tests that are applied by using launch-on-shift (LoS) or launch-on-capture (LoC) schemes. Each delay test consists of a test pattern pair with an *initialization pattern* and a *propagation pattern*. The LoS scheme loads the initialization patten, applies a single shift clock to generate the propagation pattern and loads the possibly erroneous result by a single system clock. The LoC scheme generates the propagation pattern by a system clock, and loads the result by a second system clock. Since this paper deals with SDDs, we can assume that the first system clock is applied slow enough to generate an error-free propagation pattern, and the second system clock is applied fast enough to load a possibly erroneous values. For diagnosing permanent faults or transition faults, other techniques can be found in the literature [1, 28, 29].

III. Variation-Invariant Backtracing

The analysis of a faulty signature begins with a variation-invariant backtracing approach shown in the upper part of Fig. 3. Its goal is to prune the vast search space of all possible SDDs down to an initial set of SDD candidates for further evaluation by diagnostic fault simulation. It is robust against any additional influence on the fault signature caused by timing variations under the assumption that there is only a single SDD in the CUD and the circuit without the SDD but with the same timing variations would have passed the test.

This backtracing approach is conservative in nature so as to guarantee that the generated set of SDD candidates includes all circuit structures and fault sizes that could possibly lead to the observed signatures. It exploits the following facts on the defective behavior of the CUD:

- If a signature bit $s_i(t)$ is faulty, then at least one PPO in $O[s_i]$ is faulty and the culprit is located on a structure with a possibly-sensitized path (defined below) to any PPO in $O[s_i]$.
- A failing signature cannot be explained by an SDD candidate that is located in an internal cell showing a stable signal in fault-free simulation of that delay test.
- The size of an SDD candidate must be sufficient to reach all faulty signature bits.

Variation-invariant backtracing consists of two phases: *spatial pruning* and *temporal pruning*. In spatial pruning, all failing signature bits are traced back along possibly-sensitized paths (similar to [23, 25]) to mark all possible SDD locations in the circuit. For each possible SDD location, temporal pruning is used to calculate upper and lower bounds on its fault size.

A. Six-Valued Logic Simulation

Both spatial and temporal pruning rely on a fault-free logic simulation to determine possible SDD activation and propagation paths. This logic simulation uses a 6-valued $\boldsymbol{H}_6$ algebra [30] to pessimistically analyze signal transitions and static hazards. This algebra is defined over the set $\boldsymbol{H}_6 = \{\mathrm{S0}, \mathrm{S1}, \mathrm{T0}, \mathrm{T1}, \mathrm{H0}, \mathrm{H1}\}$, where the symbols S0 and S1 represent *static signals*, T0 and T1 represent *non-static signals* with transitions or dynamic hazards, H0 and H1 represent *non-static signals* with possible hazards. When there is a path of non-static signal values between an internal signal a and a PPO o, we call it a *possibly-sensitized path*. Possibly-sensitized paths can serve as SDD propagation paths and lead to erroneous PPOs in the CUD. Figure 4 shows an example of a 6-valued simulation of one test pattern pair identifying two possibly-sensitized paths.

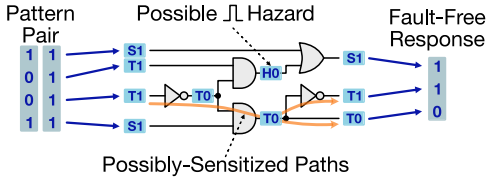

Fig. 4. Fault-free 6-valued logic simulation of a pattern pair and two identified possibly-sensitized paths.

Again, the simulation is pessimistic, meaning that the set of possibly-sensitized paths always includes all the actually sensitized paths in the defective CUD. This is also true in particular for defective CUD under timing variations. Timing variations may change non-robustly [31] sensitized paths as the arrival times of off-path values differ. Still, all possible non-robust sensitizations are marked appropriately as non-static signals during 6-valued simulation.

B. Spatial Pruning

In spatial pruning, all erroneous signatures are analyzed to determine which circuit structures (i.e., cells or signals) may be responsible for the erroneous bits in the signatures.

Every signal in the combinational circuit model is associated with a hit-counter value whose initial value is set to 0. The tester fails are analyzed one by one and the counter of a signal a is increased whenever both of the following two conditions become true:

1) There is a possibly-sensitized path between the signal a and a PPO that may be responsible for an erroneous signature bit.
2) There is possibly at least one transition on the signal a itself.

Let t be a failing delay test and $S_f(t) = \{s \in S | s^{\mathrm{CUD}}(t) \neq s^{\mathrm{good}}(t)\}$ the set of erroneous signature bits. The set of all possibly-erroneous PPOs is then: $O_f(t) = \cup_{s \in S_f(t)} O[s]$.

The delay test t is simulated to obtain the $\boldsymbol{H}_6$ value for each internal signal in the fault-free circuit. Each non-static signal at a PPO in $O_f(t)$ is flagged as suspicious. These flags are now back-propagated in reverse topological order towards the PPIs. For each cell with a flagged output signal all non-static input signals are flagged suspicious as any of them might carry the delayed transition that caused the erroneous signature in the CUD. After back-propagation is completed, the hit-counter of each flagged signal is incremented, the flags are reset and the next test with erroneous signature is analyzed.

Figure 5 shows an example structural pruning performed on a small circuit and with two tests. The outputs of the fault-free 6-valued logic simulation are compared to the observed responses. Starting from each mismatching output, all driving signals showing a value of T0, T1, H0, or H1 are flagged.

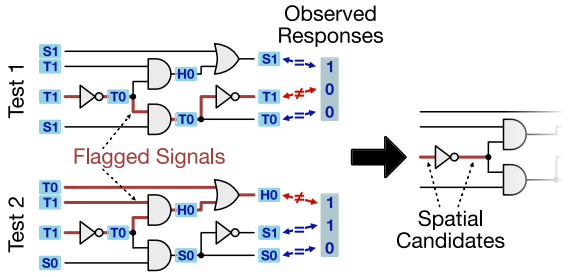

Fig. 5. Back-propagation of possibly-sensitized paths from failing outputs.

By design, the pessimistic 6-valued fault-free simulation combined with the described variation-invariant backtracing approach guarantees that the signal with the real culprit is always flagged suspicious for every failing test. As the set of possibly-sensitized paths determined by 6-valued simulation is always a superset of the paths actually sensitized in the CUD, there cannot be a SDD in the CUD that fails a test but is not flagged by backtracing. Therefore, the SDD will be located on a signal with its hit-counter matching the number of failing tests. These signals are called *spatial candidates*. The right-hand side of figure 5 shows the two spatial candidates obtained by structural pruning.

The complexity of this approach is $\mathcal{O}(|T_f| \cdot \#\mathrm{nets})$ with T_f being the set of failing tests. The computational effort is equivalent to a simple logic simulation of T_f.

C. Temporal Pruning

Temporal pruning determines the upper bound $\mathrm{size}_{\max}$ and the lower bound $\mathrm{size}_{\min}$ on the fault size of each spatial candidate. If the size of an SDD is less than its lower bound $\mathrm{size}_{\min}$, the additional delay will be insufficient to reach all observed erroneous signature bits. If the size of an SDD is larger than its upper bound $\mathrm{size}_{\max}$, it will result in exactly the same responses as an SDD of the size $\mathrm{size}_{\max}$. Therefore, sizes outside these bounds do not need to be simulated in the following inject-and-validate phase.

The bounds are calculated based on nominal timing information without considering variations directly. This is a deliberate choice, because the SDD candidates will later be fault-simulated with nominal timing as well. The variation-aware scoring approach described in the next section compensates for the discrepancies between simulated signatures and real observations.

The lower bound $\mathrm{size}_{\min}$ is based on the *latest stabilization times* (LSTs) [32] at each possibly-erroneous PPO $O_f(t)$, $t \in T$. After the 6-valued simulation of a failing test t, the circuit is traversed in topological order from the PPIs to the PPOs. The LST at each PPI is set to 0. The LST at the output of a cell is calculated based on the LSTs at its inputs, the cell delays, and the $\boldsymbol{H}_6$ values of the corresponding signals. For each input pin of a cell that carries a non-static signal, a LST candidate is obtained by adding the pin-delay to the input-LST. If the output is non-static, its LST is the maximum of the LST candidates; otherwise, if the output of the cell is static, its LST is set to 0.

After this propagation, $\mathrm{LST}(t, o)$ gives for a test t and a PPO o the point in time after which the signal is guaranteed to remain static in the fault-free circuit. The slack at the PPO o for the test t is the difference between its $\mathrm{LST}(t, o)$ and the capture time C: $\mathrm{slack}(t, o) = C - \mathrm{LST}(t, o)$. The slack of a signature bit $s \in S$ equals the minimum slack of its PPOs: $\mathrm{slack}(t, s) = \min\{\mathrm{slack}(t, o) | o \in O[s]\}$. For a signature bit $s \in S_f(t)$ to be erroneous, there must be a SDD f in the circuit with a size larger than the slack at s ($\mathrm{size}(f) \geq \mathrm{slack}(t, s)$).

The lower bound $\mathrm{size}_{\min}$ is the maximum slack at a *failing* signature bit of the whole test:

$$\mathrm{size}_{\min} = \max\{\mathrm{slack}(s, t) | t \in T, s \in S_f(t)\}.$$

This lower bound does not depend on fault location and is therefore the same for all spatial candidates. Figure 6 shows an example of computing the lower bound based on two tests.

To compute the lower bound, the circuit has to be traversed one time for each failing test. Therefore, its complexity is again equivalent to a simple logic simulation of all failing tests: $\mathcal{O}(|T_f| \cdot \#\text{nets})$.

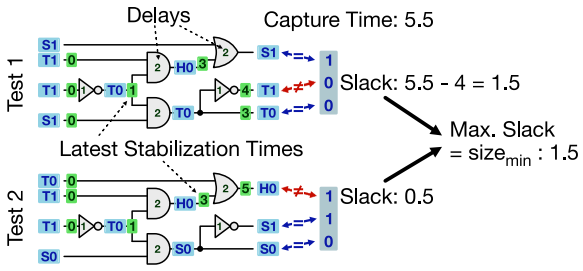

Fig. 6. Example of computing the lower bound with two tests.

The upper bound $\text{size}_{\max}$ of a SDD candidate is based on the *earliest arrival times* (EATs) at all of its reachable PPOs. The earliest arrival time $\text{EAT}(t, o)$ at a PPO o for a test t is an estimated point in time before which the signal at o is guaranteed to be stable. These EATs are calculated in the same way as the LSTs. The only difference is that at each cell the smallest EAT candidate is propagated to the cell's output signal. If a SDD is large enough to push the EAT of a PPO beyond the capture time C, it leads to the same captured value as any larger fault at the same location. Therefore, the upper bound $\text{size}_{\max}(f)$ of a fault candidate f is the minimum delay required to push the EATs of all potentially observing PPOs beyond C. Let $O(t, f)$ be the set of PPOs with a possibly-sensitized path originating at f under test t. The upper bound for the candidate f is:

$$\text{size}_{\max}(f) = C - \min\{\text{EAT}(t, o) | t \in T, o \in O(t, f)\}$$

Figure 7 shows an example of computing the upper bound of a spatial candidate with two tests.

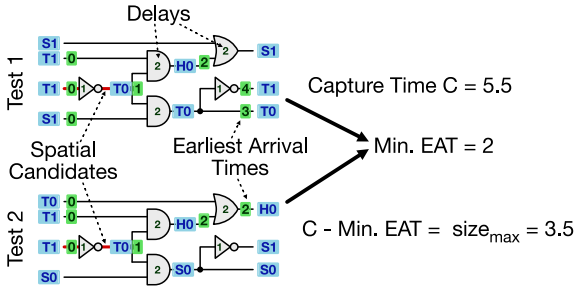

Fig. 7. Example of computing the upper bound of a spatial candidate with two tests.

A few timing-aware diagnosis techniques also use passing responses of a CUD to estimate $\text{size}_{\max}(f)$ [8, 9, 11, 24]. This is not applicable here because a passing signature bit s does not imply that all PPOs in $O[s]$ are correct. If $\text{size}_{\max}(f)$ would be estimated using the passing bits in a compressed signature, the result might be too low since the possible masking within the compactor is not considered. Furthermore, the role of $\text{size}_{\max}(f)$ in our approach is not to estimate the size of the real culprit but to determine the size where the behavior of a SDD candidate transitions to a gross delay fault.

The upper bounds are unique to each spatial candidate because each of them can propagate to different PPOs. They are, however, independent of the failing signature bits S_f and can therefore be pre-computed for all possible SDD candidate locations using only the design information and the used test set. During diagnosis itself, the upper bound for a spatial candidate is then simply selected from the pre-computed information without any impact on runtime.

The size of each spatial candidate SDD f must fall within the calculated bounds: $\text{size}_{\min} < \text{size}(f) \leq \text{size}_{\max}(f)$. If $\text{size}(f)$ is smaller than $\text{size}_{\min}$, it cannot reach all failed signature bits. Any fault with $\text{size}(f)$ larger than $\text{size}_{\max}(f)$ leads to the same test response as the fault with $\text{size}(f) = \text{size}_{\max}(f)$. This spatially and temporally constrained set of SDD candidates are called the *initial candidates*.

IV. SDD Candidate Timing Simulation

The initial SDD candidates are now explicitly simulated using the GPU-accelerated waveform-accurate small delay fault simulation engine in [5]. To correctly simulate all hazards and race conditions that might be caused by the SDD, each candidate is simulated with a concrete size and polarity. The fault sizes at each candidate fault location are chosen based on the bounds determined in temporal pruning and the characteristics of the timing variations of the CUD.

First, we will briefly present the simulation engine itself (Subsection IV-A), followed by our new simulation result confidence estimation (Subsection IV-B) and our variation-aware SDD candidate scoring approach (Subsection IV-C). In the final Subsection IV-D we will explain, how the search space defined by the initial candidates is explored by carefully selecting concrete SDD candidates for scoring.

A. GPU-Accelerated Timing Simulation

Our diagnosis approach requires an extremely high performance timing simulator that is able to efficiently process thousands of SDD candidates per diagnosis case. A traditional timing simulator is insufficient here as it would lead to prohibitively long diagnosis run-time. Recently, a waveform-accurate small delay fault simulation engine was proposed [5] that offers 3-4 orders of magnitude speedup over event-based simulation. This is the first work that applies this simulation engine to logic diagnosis.

The basic unit of computation is a waveform [33] that contains all transitions of a specific signal in a circuit. The waveforms for the PPIs are initialized with the test data and the remaining waveforms in the combinational circuit are calculated in topological order and using the nominal pin-to-pin and interconnect delays. As this computation progresses,

the waveforms store all occurring transitions and hazards which are then available at the PPOs. During propagation, the simulator exploits two dimensions of parallelism as shown in Fig. 8. Multiple test stimuli are processed in parallel and all data-independent gates are calculated in parallel as well. One gate for one stimulus is handled by a single thread in the GPU. The two dimensions of parallelism then produce the massive number of threads required to saturate the compute resources and enable the aforementioned speedups.

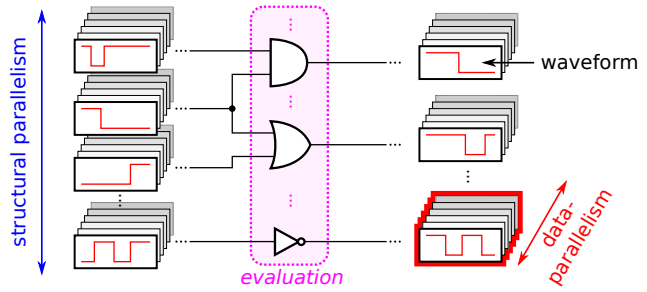

Fig. 8. Two-dimensional waveform processing.

Before the first simulation, the design data, nominal timing information and the test set is transferred to the global memory (on-board G-RAM) of the GPU. This information stays in GPU memory and does not need to be transferred again. To simulate a SDD candidate, the appropriate pin or interconnect delay is modified in GPU memory, the whole test set is propagated to the PPOs as described before, and the modified delay is restored to its original value.

For each test and PPO, the simulation delivers the full switching history over time including all glitches. This information is used in two ways. First, a predicted signature is calculated by directly computing the parity bits corresponding to the used on-chip compactor. In the presence of circuit variation, however, a single capture of the output responses can be misleading and might eventually result in the wrong candidates, especially when matching compressed signatures. Therefore, the full switching history is also used to determine the confidence in each PPO value by analyzing the signal transition times surrounding the nominal capture time. If a PPO switches close to the capture time during nominal-time simulation, chances are high that a different value has been captured in the CUD due to process variations. Therefore, the confidence in this simulated value is lower. Finally, a ranked list of SDD candidates is produced by matching simulated and observed signatures as to be detailed below.

B. Confidence Estimation for Variation Tolerance

After the timing simulation of a SDD candidate f with a test t, the full waveform with all the signal changes are analyzed for each PPO. 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 PPO o in the CUD captured a logic 1. The predicted logic value at the PPO o is $o^f(t) = 1$ if $A > 0.5$ and $o^f(t) = 0$ otherwise. The confidence in the predicted logic value is given by:

$$c(o^f(t)) = |2 \cdot (A - 0.5)|.$$

The confidence is $c = 0.0$ if $A = 0.5$ and $c = 1.0$ if $A = 0.0$ or 1.0.

From the predicted logic values and confidences at each PPO, it is easy to calculate the values and confidences for each signature bit:

$$s^f(t) = \bigoplus_{o \in O[s]} o^f(t) \text{ and } c(s^f(t)) = \prod_{o \in O[s]} c(o^f(t))$$

The remaining question is how to choose the standard deviation σ that determines the sensitivity of the confidence prediction to nearby transitions. A reasonable estimate of σ is given by the expected standard deviation of the latest stabilization times (LST) at the PPOs of the circuit under variation. We determine this standard deviation by simulating a large population of circuits with random timing variations and recording the LST for each test and each PPO. The difference between the nominal LSTs and the LSTs under variations are accumulated into a histogram. After this, we fit a normal distribution to this histogram using the least-squares method and use the fitted standard deviation for σ.

C. SDD Candidate Scoring

Each SDD candidate is assigned a score to reflect how well its simulated signatures predict the observed ones from the CUD. For a SDD candidate f and a test t, the score contribution of a single signature bit s is calculated as:

$$\text{score}(s^f(t)) := \begin{cases} c(s^f(t)) & \text{if } s^f(t) = s^{\text{CUD}}(t), \\ -c(s^f(t)) & \text{otherwise.} \end{cases}$$

A score close to 1 denotes a confident match between the CUD signature bit and the fault simulation. A score close to 0 signifies that fault simulation is unable to predict a reliable value. A score close to -1 shows a confident mismatch between simulation and observed behaviors.

The candidate score for one test t is just the sum of the scores for all individual signature bits: $\text{score}(f,t) = \sum_{s \in S} \text{score}(s^f(t))$. The candidate score for a test set is again the sum of the scores for all tests: $\text{score}(f,T) = \sum_{t \in T} \text{score}(f,t)$.

Figure 9 shows an example of computing predictions, confidences and final score of an SDD candidate.

The search space exploration described in the next section also makes use of a score variant called a *tester-fail score*. The difference of the tester-fail score to the score above is that it considers only *failing* signature bits $S_f(t)$ of each test t: $\text{tfscore}(f,t) = \sum_{s \in S_f(t)} \text{score}(s^f(t))$, $\text{tfscore}(f,T) = \sum_{t \in T} \text{tfscore}(f,t)$. The tester-fail score is used in the early phases where the size of the simulated SDD candidates are larger than the size of the culprit and likely to produce more failing signature bits during simulation. The tfscore ignores these simulator-mispredictions.

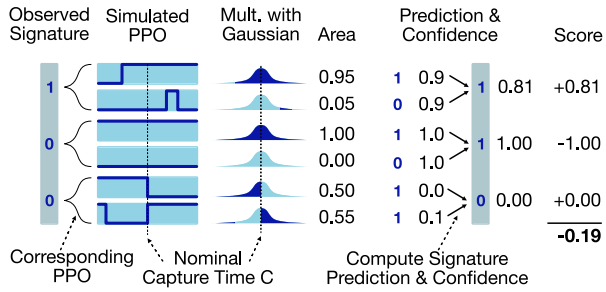

Fig. 9. Example of computing predictions, confidences and final score of an SDD candidate.

D. Candidate Simulation and Ranking

Even with efficient pruning and fast GPU-accelerated simulation, simulating all candidates with all tests would take prohibitively long time in most cases. To explore the search space defined by variation-aware backtracing more efficiently, candidate simulation is divided into *gross delay tester-fail scoring* and *final SDD scoring*.

Gross delay tester-fail scoring focuses on reducing the number of candidate SDD locations and polarities. For each spatial candidate, two SDD candidates are generated; one slow-to-rise and one slow-to-fall. The size of each spatial SDD candidate f is set to its upper bound $\mathrm{size}_{\max}(f)$, so they behave essentially as gross delay faults. The tester-fail scoring $\mathrm{tfscore}$ is used for the initial ranking to ignore simulator-mispredictions. Using tester-fail scoring here has the additional benefit that only failing tests have to be timing simulated for each candidate and not the complete test set. This way, more candidates can be scored in the same amount of time.

Spatial pruning looses effectiveness with higher compression ratios, leading to some cases where the number of spatial candidates is too large. In this case, the candidates are ranked in two passes. The first pass only scores SDD candidates on fanout-stems. The fanout-stems with the highest $\mathrm{tfscore}$ mark the suspicious fanout-free regions and the spatial candidates in these regions are scored in the second pass.

The result of gross delay tester-fail scoring is a list of gross delay fault candidates sorted by their tester-fail scores. The real culprit location is likely near to top of this list.

For *final SDD scoring*, the top gross delay fault candidates (location and polarity for each candidate) are chosen to generate the final set of SDD candidates. Let n be a cutoff count. We select the n top-most candidates and include all remaining candidates with $\mathrm{tfscore}$ equal to the top-nth candidate. This increases the probability that the real culprit is among the candidates even if tester-fail scoring was unable to distinguish them.

For each gross delay fault candidate f, we know its size interval $[\mathrm{size}_{\min}, \mathrm{size}_{\max}(f)]$ from temporal pruning. The interval is sampled in steps of 3σ to generate the concrete fault sizes for final scoring. The choice of 3σ provides a good trade-off where the Gaussian weights during scoring are still somewhat overlapping. A finer sampling may lead to more candidates and longer simulation times with no benefit as the delay differences are indistinguishable from timing variations. A coarser sampling may reduce diagnostic success as the real culprit's delay may be in between two samples and scoring is less effective.

All final SDD candidates are now scored with all passing and failing tests. The final ranking is obtained by sorting these final SDD candidates by their scores.

V. Experimental Results

The goals of our experiments were to show the diagnostic performance in terms of resolution and runtime, as well as the robustness against process variations and response compression of the proposed diagnosis algorithm. They were conducted on large ITC'99 benchmark circuits that were synthesized using the SAED 90nm technology library and a standard commercial tool flow. For each circuit, a transition-fault LoC delay test set was generated using a commercial ATPG. The response compactor was not considered during ATPG as its impact on fault coverage is negligible.

First, we describe how we obtained and characterized benchmark circuits with random timing variations that will serve as circuits under diagnosis. After that, we compare the performance of our diagnosis algorithm with the state-of-the-art.

A. Benchmarks with Timing Variations

On the basis of the nominal timing information of each cell and wire, we generated variants for a benchmark circuit by randomly changing the cell and wire delays according to a normal (Gaussian) distribution. For each nominal cell and wire delay d, a random value d' was picked from a normal distribution with standard deviation $\sigma = 0.2 \cdot d$ and mean $\mu = d$. To avoid unreasonable short or negative delays, the minimum of each randomly picked delay d' was set to 50% of the original delay d, so that $d' \geq 0.5 \cdot d$ for all cases. These parameters were chosen as typical examples for the experiments here because actual variation data is not available for the benchmark circuits.

All benchmark circuit variants were simulated with full timing using the transition-fault test set and the latest stabilization time was measured at each PPO and for each test. This information was used in two ways. First, the capture time C was set so that 95% of all generated variants pass the transition-fault test. Second, the variations of all latest transition times were analyzed to determine the standard deviation σ used in our scoring method as detailed in section IV-B.

The basic statistics on circuit netlists, test sets, and timing variations are shown in Table I. Column #nets gives the number of signal nets, i.e. the number of possible SDD locations, in the circuit. $|T|$ shows the number of transition-fault tests generated by ATPG, and $|O|$ shows the number of PPOs. Over 100 variants were generated and simulated for each benchmark circuit to determine capture time and LST standard deviation. Column C shows the capture time that

pass 95% of all variants, and column σ shows the standard deviation of all LST at the PPOs.

TABLE I
STATISTICS OF BENCHMARKS AND TEST SETS.

	#nets	$\|T\|$	$\|O\|$	C	σ
b14	15839	814	270	7.076	0.195
b15	20854	1217	488	6.016	0.189
b17	58794	1413	1415	8.985	0.223
b18	154289	1764	2823	16.202	0.237
b20	33581	986	453	8.697	0.200
b21	33157	1077	453	7.572	0.199
b22	49872	1124	636	9.233	0.204

In our diagnosis experiments, the defects were injected into a randomly selected variant that had passed the transition-fault test in the fault-free case. The specific timing variation was of course unknown to the diagnosis system and all analyses and simulations were carried out with nominal timing.

B. Small Delay Fault Diagnosis

For measuring the diagnostic performance, a set of small delay defects are required that can be injected into the simulated CUD. In our experiments, we used only small delay defects that could produce different signatures from gross delay faults. To find such defects, randomly selected SDDs were simulated and the properties of the test responses were checked. If an SDD was detected by the test set and its behavior was different from a gross delay fault at the same location, it would be considered as a culprit. From all the randomly selected SDD candidates, on average, 55.5% were not detected by the test set, 28.1% showed gross delay behavior, and 16.4% showed unique SDD behavior. The diagnosis experiments were conducted only on the 16.4% of faults with SDD behaviors. This was done to highlight the improvement over the state-of-the-art specifically for these hard-to-diagnose cases. If there are SDD candidates with gross delay behavior that fit the syndromes, our algorithm will report them in the same way as transition fault diagnosis. For each circuit, 100 of such culprits were generated and subsequently diagnosed by our method.

A culprit is considered "found", if its location is among the top 10 in the final ranking of candidates. The average rank "avg.rank" is the average number of picks from the ranked list of candidates until the position of the real culprit is found or the the set number of maximum tries is exceeded. For a successful diagnosis, the culprit's position in the ranking is considered for the average, for an unsuccessful diagnosis, the 10 unsuccessful picks is added to the average. For avg.rank, 1.00 is the best and 10.00 is the worst possible value.

As there are no SDD diagnosis approaches available that are designed to work with compressed failure data, we will compare our method with a transition fault diagnosis approach that supports highly compressed test responses [19] representing the state-of-the-art. Table II compares the base-line diagnostic success rate and resolution on the full response without any test compression.

TABLE II
DIAGNOSTIC RESULTS FOR 100 TEST CASES FOR EACH CIRCUIT.

	TF Diagnosis [19]		Proposed		
	found	avg.rank	found	avg.rank	avg.RT
b14	85	4.16	87	3.29	67s
b15	75	5.40	91	3.17	60s
b17	75	4.96	91	3.17	132s
b18	75	5.03	88	3.04	589s
b20	78	4.58	89	3.13	82s
b21	74	4.83	88	3.10	116s
b22	79	4.80	88	3.25	131s
avg.	77.4	4.82	88.8	3.16	

We observe that the average success rate of our approach is more than 88.8% while transition fault diagnosis can only find about 77.4% of the culprits. This is due to the fact that the responses of transition faults do not always match well with the culprit's responses especially when the CUD is affected by random timing variations. The use of small delay fault simulation together with variation-aware scoring clearly leads to a much higher diagnosis success rate. Still, 11.2% of culprits were not found by our method. In the vast majority of these cases, the number of failing PPOs were extremely low (i.e. less than 10 failing response bits for the whole test) and the diagnosis algorithm is not able to distinguish between SDD candidates sufficiently for a good ranking. This behavior is common to all diagnosis approaches and typically remedied by improving the test using diagnostic ATPG approaches.

Column "avg.RT" shows the average runtime of merely a few minutes for the diagnosis of one culprit measured on an Intel Xeon CPU (single-threaded implementation for the parts running on the CPU) and an nVidia Titan V GPU for accelerated timing simulation. The required memory never exceeded 8GB.

Again, we used only culprits with small delay fault behaviors. The same experiment with gross delay faults would yield the same results for the proposed algorithm and the traditional transition fault diagnosis. As the diagnosis results would always be the same, we chose to exclude gross delay defects from the comparison.

C. Impact of Response Compression

We also conducted experiments to explore the relation between response compression ratios and diagnostic success. We explored a scenario where each benchmark circuit contained n parallel scan chains and a parity-tree space compactor [26]. The space compactor computes for each scan slice a single signature bit, leading to a compression ratio of $1 : n$. The compression ratios in our experiments ranged from $n = 10$ to $n = 100$. The small delay behavior of the culprits were checked on the their uncompressed response. In a few cases, response compression led to the same signatures as their gross delay fault counterparts. These culprits were still included in the following results.

The experimental results are shown in Table III. As expected, average run-times slightly increase and diagnostic

TABLE III
DIAGNOSTIC RESULTS FOR 100 TEST CASES FOR EACH CIRCUIT UNDER 1:n RESPONSE COMPRESSION.

		TF Diagnosis [19]		Proposed		
	n	found	avg.rank	found	avg.rank	avg.RT
b14	10	77	4.88	83	3.64	78s
	20	71	5.20	81	3.70	98s
	30	66	5.53	80	4.10	101s
	50	59	6.02	79	4.18	115s
	100	37	7.50	68	4.86	129s
b15	10	60	6.18	92	3.36	83s
	20	56	6.28	92	3.42	62s
	30	51	6.79	89	3.54	68s
	50	46	7.12	87	3.63	78s
	100	31	8.07	77	4.38	96s
b17	10	65	5.64	91	3.03	96s
	20	62	5.84	88	3.20	115s
	30	56	6.09	89	3.27	119s
	50	56	6.21	88	3.19	115s
	100	48	6.61	85	3.61	145s
b18	10	70	5.67	88	3.13	350s
	20	64	5.77	86	3.25	390s
	30	64	5.90	86	3.25	429s
	50	64	6.04	86	3.35	410s
	100	57	6.45	87	3.32	477s
b20	10	75	4.98	87	3.30	125s
	20	69	5.25	87	3.35	143s
	30	66	5.51	82	3.57	144s
	50	60	5.94	80	3.92	174s
	100	51	6.50	71	4.31	249s
b21	10	70	5.32	87	3.33	134s
	20	64	5.58	85	3.49	141s
	30	61	5.71	83	3.52	160s
	50	58	5.96	78	4.03	175s
	100	50	6.51	72	4.48	234s
b22	10	72	5.21	86	3.55	135s
	20	69	5.51	82	3.73	185s
	30	68	5.44	81	3.89	205s
	50	61	6.06	76	4.14	240s
	100	53	6.58	73	4.51	336s
avg.	10	69.8	5.41	87.7	3.33	
	20	65.1	5.63	85.7	3.45	
	30	61.7	5.85	84.2	3.59	
	50	57.8	6.19	81.9	3.78	
	100	46.7	6.89	76.2	4.21	

successes decrease with higher compression ratios. However, while the success rate with traditional transition fault diagnosis decreases rapidly with increased compression ratios, our proposed method can maintain fairly high success rates. The average success rates from Table III are plotted in Fig. 10.

It can be clearly seen that the advantage of our proposed diagnosis approach increases with higher compression ratios. While the success rate is about 10% higher on uncompressed responses, it is almost 30% higher under 1 : 100 response compression. It is evident that our advanced, variation-aware scoring approach copes much better with highly compressed variation-affected failure data than previous approaches.

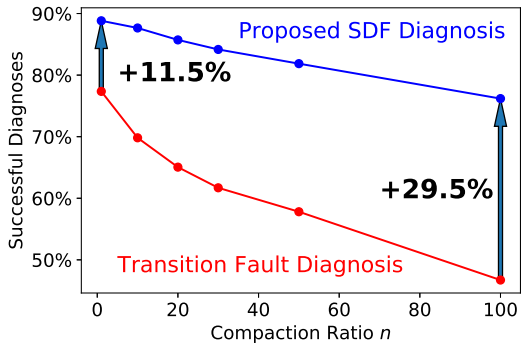

Fig. 10. Relation between compression ratio and diagnosis success rate.

VI. CONCLUSIONS

This paper has proposed an innovative and effective method for the direct diagnosis of single small delay faults with compressed test responses. The proposed method combines conservative structural and temporal pruning techniques with a high-throughput high-precision GPU accelerated timing simulator for analyzing the precise behaviors of individual SDD candidates. The proposed method is the first that can correctly locate small delay faults from highly compressed responses with a high success rate. Furthermore, the proposed method is robust against process variations through the unique consideration of the prediction confidences during fault simulation. In future works, we intend to extend our approach to layout-aware diagnosis, cell-aware diagnosis and diagnosis of multiple SDDs by supporting additional fault models in the accelerated timing simulator.

ACKNOWLEDGMENT

This work has been funded by the German Research Foundation (DFG) under contract WU 245/19-1 as well as 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.

REFERENCES

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

[2] Y. Liu and Q. Xu, "On modeling faults in FinFET logic circuits," in *Proc. IEEE Int. Test Conf. (ITC)*, Nov. 2012, pp. 1–9.

[3] A. K. Pramanick and S. M. Reddy, "On the fault coverage of gate delay fault detecting tests," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 16, no. 1, pp. 78–94, Jan. 1997.

[4] E. Schneider, S. Holst, M. A. Kochte, X. Wen, and H.-J. Wunderlich, "GPU-accelerated small delay fault simulation," in *Proc. ACM/IEEE Conf. on Design, Automation Test in Europe (DATE)*, Mar. 2015, pp. 1174–1179.

[5] 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 (TCAD)*, vol. 36, pp. 829–841, May 2017.

[6] A. Czutro, N. Houarche, P. Engelke, I. Polian, M. Comte, M. Renovell, and B. Becker, "A simulator of small-delay faults caused by resistive-open defects," in *Proc. IEEE European Test Symp. (ETS)*, May 2008, pp. 113–118.

[7] M. Sauer, A. Czutro, I. Polian, and B. Becker, "Small-delay-fault ATPG with waveform accuracy," in *Proc. IEEE/ACM Int. Conf. on Computer-Aided Design (ICCAD)*, Nov. 2012, pp. 30–36.

[8] 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)*, 2007, pp. 223–234.

[9] 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 (TCAD)*, vol. 28, no. 2, pp. 245–258, Feb. 2009.

[10] M. Tehranipoor, K. Peng, and K. Chakrabarty, *Test and Diagnosis for Small-Delay Defects*. Springer New York, 2011.

[11] 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)*, 2011, pp. 291–296.

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

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

[14] G. L. Smith, "Model for delay faults based upon paths," in *Proc. IEEE Int. Test Conf. (ITC)*, 1985, pp. 342–351.

[15] A. K. Pramanick and S. M. Reddy, "On the computation of the ranges of detected delay fault sizes," in *IEEE Int. Conf. on Computer-Aided Design. Digest of Technical Papers*, Nov. 1989, pp. 126–129.

[16] A. Srivastava, D. Sylvester, and D. Blaauw, *Statistical Analysis and Optimization for VLSI: Timing and Power*. Springer, 2005.

[17] M. Sauer, I. Polian, M. E. Imhof, A. Mumtaz, E. Schneider, A. Czutro, H.-J. Wunderlich, and B. Becker, "Variation-aware deterministic ATPG," in *Proc. IEEE European Test Symp. (ETS)*, May 2014, pp. 1–6.

[18] W.-T. Cheng, K.-H. Tsai, Y. Huang, N. Tamarapalli, and J. Rajski, "Compactor independent direct diagnosis," in *Proc. IEEE Asian Test Symp. (ATS)*, Nov. 2004, pp. 204–209.

[19] 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)*, 2009, pp. 1355–1360.

[20] 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)*, 2002, pp. 117–125.

[21] A. Krstic, L.-C. Wang, K.-T. Cheng, and J.-J. Liou, "Diagnosis of delay defects using statistical timing models," in *Proc. VLSI Test Symposium (VTS)*, 2003, pp. 339–344.

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

[23] 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.

[24] 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.

[25] 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.

[26] H. P. E. Vranken, S. K. Goel, A. Glowatz, J. Schlöffel, and F. Hapke, "Fault detection and diagnosis with parity trees for space compaction of test responses," in *Proc. ACM/IEEE Design Automation Conf. (DAC)*, 2006, pp. 1095–1098.

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

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

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

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

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

[32] V. S. Iyengar, B. K. Rosen, and J. A. Waicukauski, "On computing the sizes of detected delay faults," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 9, no. 3, pp. 299–312, Mar. 1990.

[33] 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. 1–22, Article 37, Jun. 2015.