Interactive Logic Diagnosis of Unpredicted Defects in Logic Circuits

Stefan Holst[1,a)]

Abstract: With increasingly tight integration of complex and diverse technologies (More–than–Moore) in VLSI, logic diagnosis algorithms become ineffective because of unpredictable defect mechanisms. This paper introduces a new concept called Interactive Logic Diagnosis that tackles this problem by combining automated diagnosis directly with human reasoning of expert engineers. To achieve interactive diagnosis, the diagnosis method needs to be responsive as well as transparent. For both aspects, this paper will share initial ideas and results of the ongoing research.

1. Introduction

Logic diagnosis is the process of inferring probable internal defects from the erroneous outputs of a logic circuit [1]. It is an essential part of modern VLSI chip development for identifying and eliminating issues that cause erroneous behavior, limit production yield or decrease reliability in the field [2]. Automated diagnosis methods rely on a–priori assumptions about defect mechanisms in order to produce a list of fault candidates that best explains the observed errors [3]. As VLSI technology complexity and diversity increases with the tight integration of digital-, analogue- and RF–cores [4], complex ultra–low–power management, test infrastructure and 3D–integration, defect mechanisms became much more diverse and unpredictable. Dealing with unknown defect mechanisms is extremely important for the success rate in a precision diagnosis setting. Precision diagnosis is used as preparation for destructive physical failure analysis to locate the exact circuit structure that is affected by an unknown defect. As shown in Figure 1, designers and production engineers interpret the fault candidate lists generated by logic diagnosis tools to understand the underlying defect mechanisms and take appropriate actions. However, as these fault candidate lists may already be tainted by inaccurate a–priori assumptions and engineers may be mislead in their quest to understand the true defect. Currently, post–silicon debug of prototypes usually adds many months to time–to–market [5], and cases of No trouble found (NTF) in field–returns are on the rise [6]. As engineers struggle to understand the root causes of failures, production yields remain low or product qualities fail to meet the ever–stricter Defects per Million (DPM) requirements.

This paper proposes a new logic diagnosis paradigm that fosters a much more direct interaction between the failure analysis engineer and the diagnosis algorithm. Such an *Interactive Logic Diagnosis* approach holds the promise to be much more effective in dealing with unforeseen defect mechanisms compared to traditional logic diagnosis in which the engineer can only interpret finished fault candidate lists and apply different diagnosis methods with various models in a trial–and–error fashion.

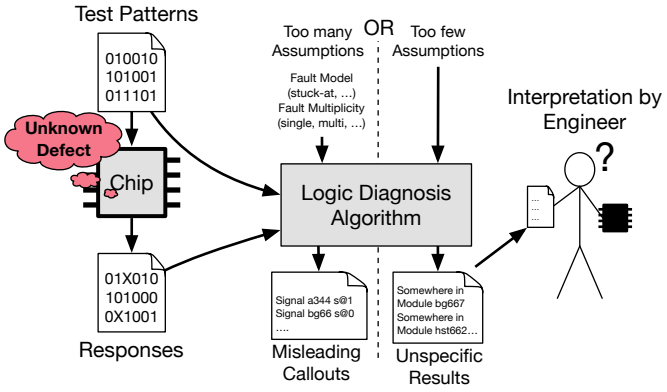

Fig. 1 Classic Diagnosis with unknown defect types.

This paper is organized as follows: Section 2 gives an overview over the state–of–the–art in logic diagnosis. Section 3 discusses the relation between a–priori assumptions and diagnosis outcomes. Section 4 introduces the new interactive diagnosis paradigm and discusses its requirements. Section 5 shows how a potential interactive diagnosis method can be made responsive using GPU–based acceleration. Section 6 concludes this paper.

2. State–of–the–Art

In the literature, logic diagnosis algorithms are often classified into cause–effect and effect–cause [7] paradigms.

Cause–effect analysis [8] starts with a set F of possible causes in the circuit. An element of this set $f \in F$ is called a fault and the set is generated according to a fault model. Diagnosis is performed by comparing for each test $t \in T$ and each fault $f \in F$ the failures generated by the fault f with the observed response of the circuit under diagnosis. The result of logic diagnosis is the set of faults, which show the same behavior as the defective circuit.

[1] Kyushu Institute of Technology, Iizuka 820–8502, Japan
[a)] holst@ci.kyutech.ac.jp

In order to avoid costly fault simulations during diagnosis, a fault dictionary is constructed. This dictionary is created once by simulating every fault $f_1, \ldots, f_n \in F$ with every pattern $t_1, \ldots, t_m \in T$ in the circuit C and storing the responses in a table. Two major disadvantages with this procedure were quickly identified. First, the size of the fault dictionary became a major issue and a lot of research effort was spend on reducing its size by various means [9], [10]. As the size of the dictionary grows with the size of F, this approach is not very effective for more complex fault models. The second disadvantage is the a–priori selection of a fault model to generate F. Many defects cause a behavior of the circuit under diagnosis, that does not match with any of the faults in F. In this case, cause–effect diagnosis will fail.

As a response to the rising variety of defect mechanisms in modern VLSI technology, research in logic diagnosis has been pushing to reduce a–priori fault assumptions by means of generalized fault models, location–based diagnosis and adaptive diagnosis.

To reduce the dependency on a single fault model as well as to handle fault models with very large number of faults, effect–cause diagnosis was proposed [11], [12], [13]. Effect–cause based diagnosis algorithms derive candidate causes directly from the observed effects. The majority of modern diagnosis algorithms implement effect–cause methods and there is also a trend of combining effect–cause and cause–effect ideas and diagnose a defect in two passes. First, a fast effect–cause analysis is performed to constrain the circuits region where possible culprits may be located. Second, for each of the possible fault sites and suspected fault model, a cause–effect approach is performed for identifying those faults which match the real observed behavior [14], [15].

Inject–and–validate is a per–test analysis concept, that follows the cause–effect principle. Also here, a set of faults F are simulated and their responses are compared to the responses of the circuit under diagnosis. But inject–and–validate procedures do not restrict themselves to exact matches between the simulated faults and the defective behavior of the circuit under diagnosis. Instead, they determine fault candidates based on the similarity between the simulated and the observed responses. The diagnosis approach in [16] was one of the first proposing this procedure to localize defects. This diagnosis method reports stuck–at faults, that are able to explain most of the failing responses.

One of the most prominent technique is the *Single Location At a Time* (SLAT) method introduced in [3], [17]. A test pattern $t \in T$ has the SLAT property, if there is at least one observable stuck–at fault $f \in F$ which produces a response on that pattern identical with the response of the circuit under diagnosis. As diagnosis result, SLAT based algorithms report a set of multiplets. A multiplet M is a subset of F with the property, that each SLAT pattern observed in the circuit under diagnosis is explained by at least one fault in M.

By using multi–valued logic simulation in inject–and–validate diagnosis algorithms, defects can be located in a circuit despite incomplete knowledge on their actual behavior [18]. Such a diagnosis algorithm reports the minimum set of signal lines that need to be set to an unknown value (X–value) in the fault simulator, such that fault simulation will produce a X for every failing response bit from the circuit under diagnosis. The same idea was used more recently [19] for diagnosing delay faults. Three–valued logic simulation is pessimistic and will result in X–values at outputs which are actually fault–free. The works [20], [21] alleviate this problem by using distinct X–symbols for every signal line and a more sophisticated X–propagation method that preserves some logic properties during X–propagation (inversion and branches). After location of the defect site, the unknown values at the site are resolved by 2–valued simulation exhaustively assigning the X–symbols to logic values. Inject–and–validate diagnosis is computational expensive as it relies heavily on fault simulation of many possible candidates. Many proposals on high performance fault simulation are also applicable to inject–and–validate algorithms. Common techniques used in fast fault simulation can be found in [22]. Another canonical way is to trade computing time against memory storage by constructing a fault dictionary [8], [9]. In contrast to cause–effect approaches, the dictionary is only used to replace some fault simulation runs by table lookups. The fault simulation problem can also be mapped to emerging data parallel architectures like General Purpose Graphics Processing Units (GPGPUs) [23], [24], [25] or hybrid systems as the Cell Broadband Engine [26] to gain speedup compared to a simulation on standard hardware.

Critical path tracing techniques trace back error propagation paths from the outputs towards the possible culprits within the circuit. The concept of critical path tracing was first introduced in [27] as an efficient alternative to fault simulation and was first applied to delay fault diagnosis in [28], [29]. The diagnosis procedure for delay faults uses a six–valued logic algebra to describe logic values, transitions and hazards in the circuit. This approach was later refined [30], applied to bridging fault diagnosis [31], [32] as well as to crosstalk–induced delay faults [33]. Path tracing based diagnosis can be improved by taking into account the timing uncertainties (variation) of manufactured circuits [34]. The approach in [35] extends back–tracing over multiple time frames to analyze responses of functional tests. Many path tracing based diagnosis techniques fail in the case of multiple interacting faults in the circuit under diagnosis as they assume all the off–path signals of the currently traced fault propagation path to be fault–free or affected by the same fault in case of a re–convergence. If multiple faults are present, the effect of a fault x may sensitize a propagation path of a fault y at gate g. If fault x is not considered, path tracing would assume, that the fault is located between g and the outputs, because a fault effect stemming from y would not propagate over g. This challenge was addressed in [36], [37], [38] by using a more conservative path tracing approach in combination with fault simulation to diagnose multiple stuck–at, bridges and transition faults.

3. Assumptions and Diagnosis Complexity

Over the past years, the reliance on restricting fault model assumptions has been reduced to accomodate for unforeseen and more complex defect behaviors. This trend leads to three challenges. The first two challenges are a direct result of having less constraints during logic diagnosis. The lack of contstraints increases the search space of possible faults dramatically which in

turn increases the diagnosis runtime. At the same time, the diagnosis results become less expressive as less information on possible defect behaviors are available. The third challenge relates to redidual assumptions that are still present in the diagnosis algorithms themselves and can limit their effectiveness with certain types of defects.

3.1 Complexity

It is easy to see that diagnosis complexity increases with less strict assumptions. For instance, with a single–stuck–at fault assumption, the diagnosis algorithm can select from $O(n)$ different fault candidates (n being the number of gates in the circuit). Already with a slightly less strict assumption of one- or two–stuck–at faults, the number of possible faults increases to $O(n^2)$. In general, diagnosis complexity rises exponentially with fault multiplicity.

To cope with complexity, diagnosis algorithms can also exploit behavioral properties of an assumed fault model. For instance, if diagnosis is constraint to bridges like wired–AND, wired–OR or similar, diagnosis becomes much easier than a more general two–fault diagnosis. The bridge diagnosis approach in [39] exploits the fact that the two signals involved in a bridge need to carry different logic values to activate the fault. In other words, whenever two signals carry the same logic value for a failing test, the bridge cannot be located between these two signals because the fault would not be active and lead to a failing test. With this knowledge about bridge behavior, the algorithm is able to diagnose bridges with linear runtime complexity $O(n)$.

3.2 Results Expressiveness

The more general a fault modeling approach is, the less expressive diagnosis results become. Every fault model carries with it a certain expectations about the types of defects in the circuit under diagnosis. Whenever the observed evidence matches a candidate within the employed fault model (such as a bridging fault), this candidate can be directly used as result.

If the fault modeling approach is more general, fault candidates in the results become more general as well and must then be interpreted by engineers using their domain knowledge and expectations.

For example, figure 2 shows a circuit (which is just an AND–gate) with some patterns and observed responses. Stuck–at fault diagnosis (fig. 2a)) would yield no candidates, because no stuck–at fault on any signal can explain all the observed responses. The *Conditional Line Flip* (CLF) model [40] is a more general modeling approach that expresses that a signal is faulty under some (possibly unknown) fault activation condition. A diagnosis under such a CLF model (fig. 2b)) would yield the correct location of the defect but the details of the faulty behavior remains unknown. The actual faulty behavior in this example can be interpreted in various ways depending on the diagnosis context as shown in fig. 2c). In a silicon chip, the inputs of the gate may be shorted with a wired–OR bridge. In a design validation context, the gate could have been acidentally exchanged with an OR–gate.

Knowledge about the diagnosis context and possible defective behaviors is important to introduce into the diagnosis process. However, one has to be very careful not to overconstrain the search space and then ending up with no or even misleading results.

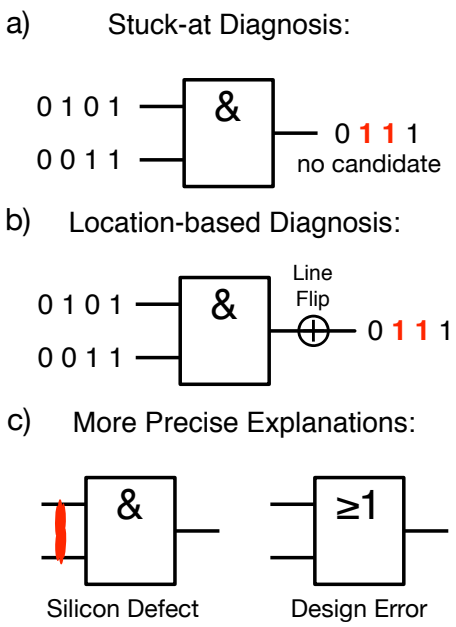

Fig. 2 Expressiveness of diagnosis results with various fault models.

3.3 Remaining Assumptions

All diagnosis algorithms found in literature still include numerous assumptions regarding the behavior of the circuit under diagnosis.

For example, inject–and–validate approaches use a certain circuit model to evaluate the propagation behavior of faulty internal signal values. For performance reasons such circuit models usually do not include gate and signal delays and are therefore unsuitable for defects with timing–related fault effects. Gross–delay or transition–fault models work around this issue as they demand, at the time the delay fault is active, that the response is the same as a stuck–at fault at the same location. If a defect behaves like a small–delay fault, however, its response may be different from that of a stuck–at fault and diagnosis approaches that do not model accurate timing in the activation and propagation paths may fail. Specialized algorithms are available for delay faults [28], but they are again inefficient with permanent faults or other defective behavior.

Modeling the actual physical behavior of the circuit and the defects in every detail is clearly infeasible. However, care must be taken to model sufficient detail whenever necessary for a given diagnosis problem.

4. Interactive Diagnosis

The main cause of the aforementioned challenges is that assumptions are made before the logic diagnosis process starts and the only way an engineer can judge the validity of the assumptions is by interpreting the diagnosis results after logic diagnosis has finished. There is currently no way to manage and adapt assumptions during the logic diagnosis process itself other than restarting diagnosis from scratch with different assumptions or

different algorithms if the previous results are unsatisfactory.

The key idea of interactive diagnosis is to enable engineers to gain insight into the diagnosis process itself, follow its reasonings and guide the process by giving additional information such as carefully considered assumptions, additional evidence, or defect hypotheses. This interaction between engineers and diagnosis algorithm has two advantages: (1) Assumptions can be changed and adapted during the diagnosis process which alleviates the problem of misguided a–priori assumptions, and (2) the engineer gains a better understanding of the defective behavior within the circuit under diagnosis.

Interactive diagnosis requires two major features: (1) The diagnosis process has to be transparent so that the engineer can observe and guide, and (2) the interaction has to be responsive for smooth user experience. Both aspects are discussed below in more detail.

4.1 Transparency

In general, diagnosis algorithms process some observed evidence such as passing tests, failing tests and their erroneous responses and produce internally a series of implications. For example, if only one output of the circuit shows an erroneous value for the whole test, it implies that all defects are located in the structural input cone of this erroneous output. If multiple outputs are incorrect during the test, all defects are located in the union–set of all structural input cones of all failing outputs. With a single–defect–assumption, the implied circuit part can be further reduced to the intersection of all these structural input cones. The implications are therefore strongly dependent on the assumptions used during the diagnosis process. For interactive diagnosis, all used assumptions have to be made explicit and clear to the user. The user can then decide to change or remove some assumptions, or, if some assumptions cannot be changed, be at least informed on the limitations of the diagnosis algorithm and work around them.

The diagnosis process lends itself to a contract–based knowledge model. Simply put, a *contract* is an implication that goes from a set of *demands* to a set of *guarantees*. Demands can be certain observed signal values at the outputs for different patterns (evidence), assumptions regarding defect multiplicity, model, or faulty value propagation. Guarantees can be the values of certain internal signals, parts of the circuit that must be free of any defects, or aspects of the behavior of the defect itself. In interactive diagnosis, all such implications have to be filtered for relevancy and bundled appropriately such that an engineer can understand them and make informed descisions on the employed assumptions, the collection of additional evidence or educated guesses (hypotheses) of some demands to explore and understand the circuit under diagnosis.

4.2 Responsiveness

The second novel requirement for interactive diagnosis is its responsiveness to new inputs, demands, and hypotheses from the user. Traditional diagnosis systems work independently from a user's attention and are free to work on a diagnosis problems for several minutes or even hours. Interactive diagnosis, however, work in conjunction with human engineers and are therefore required to provide evaluations and results within a matter of seconds from an input. In addition, the diagnosis system has to be flexible enough to accomodate various assumptions (or thelack thereof) for a variety of different defect types. How to approach this level of performance in logic diagnosis is still an unsolved problem. Using data–parallel architectures like general–purpose computing on graphics processing units (GPGPU) [41] holds some promise on delivering the necessary responsiveness for an interactive diagnosis system. Preliminary results on such a simulation approach is discussed in the next section.

5. GPU–Acceleration for Logic Diagnosis

As an essential part of logic diagnosis, diagnostic fault simulation is a prime candidate for acceleration using graphics processing units (GPUs). Fault simulation is a very common bottleneck in current diagnosis algorithms which will become even more pronounced with added flexibility requirements such as timing–accurate simulations.

In recent works, we explored using GPUs for timing simulations of combinational circuits and for small–delay–fault simulations [25], [42]. The main idea of this simulation approach is to process a massive amount of independent patterns or faults in a data–parallel manner (see figure 3).

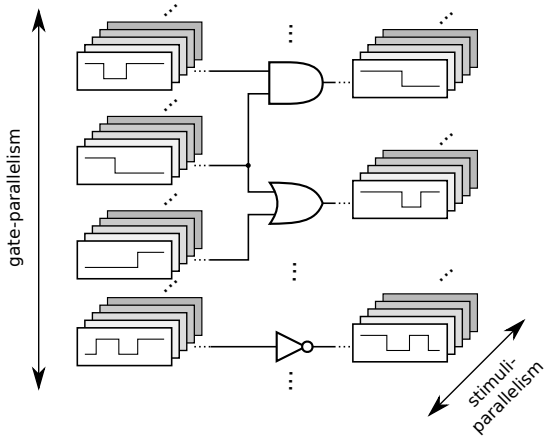

Fig. 3 Multiple dimensions of parallelism during waveform propagation enables high–throughput simulation performance.

By using complete waveforms instead of simple logic values at each stage of the simulation, full timing simulation is supported within the combinational part of the circuit without compromizing on performance. Compared to traditional event–driven timing simulation, GPU–based simulation provides 500X–1000X faster performance [25] even for million–gate designs. This enables potential interactive logic diagnosis algorithms to quickly simulate many suspects for many patterns under various conditions. Furthermore, this simulator supports keeping the full simulation state in GPU memory so the a user can directly manipulate gate or interconnect timings and re–simulate the whole circuit again without any initialization overhead.

6. Concluding Remarks

With increasingly tight integration of complex and diverse technologies (More–than–Moore) in VLSI, logic diagnosis algorithms become ineffective because of unpredictable defect mechanisms. Too strict a–priori assumptions during diagnosis lead to no or even misleading results, too few a–priori assumptions lead to increased diagnosis complexity and less expressive results.

This paper introduced an alternative concept called Interactive Logic Diagnosis that tackles this problem by combining automated diagnosis directly with human reasoning of expert engineers. This way, assumptions remain flexible and changeable by the user. Interactive diagnosis requires responsiveness as well as transparency. As an initial idea to achieve transparency, a contract–based knowledge model was proposed. To achieve the necessary responsiveness, GPU–accelerated simulators hold a lot of promise due to their extremely high performance compared to traditional simulation methods.

Acknowledgments This work was supported by JSPS Grant–in–Aid for Young Scientists #18K18026.

References

[1] Wang, L.-T., Wu, C.-W. and Wen, X.: *VLSI Test Principles and Architectures: Design for Testability (Systems on Silicon)*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2006).

[2] Wagner, L. C.: *Handbook of Semiconductor Manufacturing Technology*, chapter 29: Failure Analysis, p. 29.1, CRC Press London (2008).

[3] Huisman, L. M.: Diagnosing Arbitrary Defects in Logic Designs Using Single Location at a Time (SLAT), *IEEE Trans. on Computer Aided Design of Integrated Circuits and Systems*, Vol. 23, No. 1, pp. 91–101 (online), DOI: 10.1109/TCAD.2003.816206 (2004).

[4] Arden, W., Brillout, M., Cogez, P., Graef, M., Huizing, B. and Mahnkopf, R.: More-than-Moore White Paper, Technical report, International Technological Roadmap for Semiconductors, Annual Report (2011).

[5] Abramovici, M.: In-System Silicon Validation and Debug, *IEEE Design Test of Computers*, Vol. 25, No. 3, pp. 216–223 (online), DOI: 10.1109/MDT.2008.77 (2008).

[6] Conroy, Z., Richmond, G., Gu, X. and Eklow, B.: A practical perspective on reducing ASIC NTFs, *IEEE International Test Conference*, pp. 7 pp.–349 (online), DOI: 10.1109/TEST.2005.1583992 (2005).

[7] Abramovici, M., Breuer, M. A. and Friedman, A. D.: *Digital Systems Testing and Testable Design*, IEEE Press (1990).

[8] Richman, J. and Bowden, K. R.: The modern fault dictionary, *Proc. IEEE International Test Conference (ITC)*, pp. 696–702 (1985).

[9] Pomeranz, I. and Reddy, S. M.: On the Generation of Small Dictionaries for Fault Location, *Proc. IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 272–279 (online), DOI: 10.1109/ICCAD.1992.279361 (1992).

[10] Chess, B. and Larrabee, T.: Creating small fault dictionaries, *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 18, No. 3, pp. 346–356 (online), DOI: 10.1109/43.748164 (1999).

[11] Breuer, M. A., Chang, S.-J. and Su, S. Y. H.: Identification of Multiple Stuck-Type Faults in Combinational Networks, *IEEE Trans. on Computers*, Vol. C-25, No. 1, pp. 44–54 (online), DOI: 10.1109/TC.1976.5009204 (1976).

[12] Abramovici, M. and Breuer, M. A.: Fault Diagnosis Based on Effect-Cause Analysis: An Introduction, *Proc. 17th IEEE/ACM Design Automation Conference (DAC)*, pp. 69–76 (online), DOI: 10.1145/800139.804514 (1980).

[13] Abramovici, M. and Breuer, M. A.: Multiple Fault Diagnosis in Combinational Circuits Based on an Effect-Cause Analysis, *IEEE Trans. on Computers*, Vol. C-29, No. 6, pp. 451–460 (online), DOI: 10.1109/TC.1980.1675604 (1980).

[14] Desineni, R., Poku, O. and Blanton, R. D.: A Logic Diagnosis Methodology for Improved Localization and Extraction of Accurate Defect Behavior, *Proc. IEEE International Test Conference (ITC)*, p. 12.3 (online), DOI: 10.1109/TEST.2006.297627 (2006).

[15] Amyeen, M. E., Nayak, D. and Venkataraman, S.: Improving Precision Using Mixed-level Fault Diagnosis, *Proc. IEEE International Test Conference (ITC)*, p. 22.3 (online), DOI: 10.1109/TEST.2006.297661 (2006).

[16] Waicukauski, J. A. and Lindbloom, E.: Failure Diagnosis of Structured VLSI, *IEEE Design & Test of Computers*, Vol. 6, No. 4, pp. 49–60 (online), DOI: 10.1109/54.32421 (1989).

[17] Bartenstein, T., Heaberlin, D., Huisman, L. M. and Sliwinski, D.: Diagnosing Combinational Logic Designs Using the Single Location At-a-Time (SLAT) Paradigm, *Proc. IEEE International Test Conference (ITC)*, pp. 287–296 (online), DOI: 10.1109/TEST.2001.966644 (2001).

[18] Boppana, V. and Fujita, M.: Modeling the Unknown! Towards Model-independent Fault and Error Diagnosis., *Proc. IEEE International Test Conference (ITC)*, pp. 1094–1100 (online), DOI: 10.1109/TEST.1998.743310 (1998).

[19] Poku, O. and Blanton, R. D.: Delay Defect Diagnosis Using Segment Network Faults, *Proc. IEEE International Test Conference (ITC)*, p. 15.1 (online), DOI: 10.1109/TEST.2007.4437602 (2007).

[20] Wen, X., Tamamoto, H., Saluja, K. K. and Kinoshita, K.: Fault diagnosis for physical defects of unknown behaviors, *Proc. 12th Asian Test Symposium (ATS)*, pp. 236–241 (online), DOI: 10.1109/ATS.2003.1250816 (2003).

[21] Wen, X., Kajihara, S., Miyase, K., Yamato, Y., Saluja, K. K., Wang, L.-T. and Kinoshita, K.: A Per-Test Fault Diagnosis Method Based on the X-Fault Model, *IEICE Transactions*, Vol. 89-D, No. 11, pp. 2756–2765 (online), DOI: 10.1093/ietisy/e89-d.11.2756 (2006).

[22] Schulz, M. H.: *Testmustergenerierung und Fehlersimulation in digitalen Schaltungen mit hoher Komplexität*, Informatik-Fachberichte, Vol. 173, Springer (1988).

[23] Li, M. and Hsiao, M. S.: FSimGP^2: An Efficient Fault Simulator with GPGPU, *Proc. 19th IEEE Asian Test Symposium (ATS)*, pp. 15–20 (online), DOI: 10.1109/ATS.2010.12 (2010).

[24] Li, H., Xu, D., Han, Y., Cheng, K.-T. and Li, X.: nGFSIM: A GPU-based Fault Simulator for 1-to-n Detection and its Applications, *Proc. IEEE International Test Conference (ITC)*, pp. 12.1/1–12.1/10 (online), DOI: 10.1109/TEST.2010.5699235 (2010).

[25] Schneider, E., Kochte, M. A., Holst, S., Wen, X. and Wunderlich, H.-J.: GPU-Accelerated Simulation of Small Delay Faults, *IEEE Trans. on Computer Aided Design of Integrated Circuits and Systems*, Vol. 36, pp. 829–841 (online), DOI: 10.1109/TCAD.2016.2598560 (2017).

[26] Kochte, M. A., Schaal, M., Wunderlich, H.-J. and Zoellin, C. G.: Efficient Fault Simulation on Many-Core Processors, *Proc. 46th IEEE/ACM Design Automation Conference (DAC)*, pp. 380–385 (online), DOI: 10.1145/1837274.1837369 (2010).

[27] Abramovici, M., Menon, P. R. and Miller, D. T.: Critical Path Tracing - An Alternative to Fault Simulation, *Proc. 20th IEEE/ACM Design Automation Conference (DAC)*, pp. 214–220 (online), DOI: 10.1109/DAC.1983.1585651 (1983).

[28] Girard, P., Landrault, C. and Pravossoudovitch, S.: Delay-fault diagnosis by critical-path tracing, *IEEE Design Test of Computers*, Vol. 9, No. 4, pp. 27–32 (online), DOI: 10.1109/54.173329 (1992).

[29] Girard, P., Landrault, C. and Pravossoudovitch, S.: A novel approach to delay-fault diagnosis, *Proc. 29th ACM/IEEE Design Automation Conference (DAC)*, pp. 357–360 (online), DOI: 10.1109/DAC.1992.227778 (1992).

[30] Rousset, A., Bosio, A., Girard, P., Landrault, C., Pravossoudovitch, S. and Virazel, A.: DERRIC: A Tool for Unified Logic Diagnosis, *Proc. 12th European Test Symposium (ETS)*, pp. 13–20 (online), DOI: 10.1109/ETS.2007.16 (2007).

[31] Rousset, A., Bosio, A., Girard, P., Landrault, C., Pravossoudovitch, S. and Virazel, A.: Fast Bridging Fault Diagnosis using Logic Information, *Proc. 16th Asian Test Symposium (ATS)*, pp. 33–38 (online), DOI: 10.1109/ATS.2007.75 (2007).

[32] Wang, L., Gupta, S. K. and Breuer, M. A.: Diagnosis of Delay Faults due to Resistive Bridges, Delay Variations and Defects, *Proc. 15th Asian Test Symposium (ATS)*, pp. 215–224 (online), DOI: 10.1109/ATS.2006.261023 (2006).

[33] Mehta, V. J., Marek-Sadowska, M., Tsai, K.-H. and Rajski, J.: Timing Defect Diagnosis in Presence of Crosstalk for Nanometer Technology, *Proc. IEEE International Test Conference (ITC)*, p. 12.2 (online), DOI: 10.1109/TEST.2006.297626 (2006).

[34] Yang, K. and Cheng, K.-T.: Timing-Reasoning-Based Delay Fault Diagnosis, *Proc. Design, Automation and Test in Europe (DATE)*, pp. 418–423 (online), DOI: 10.1145/1131595 (2006).

[35] Yen, C.-C., Lin, T., Lin, H., Yang, K., Liu, T. and Hsu, Y.-C.: A General Failure Candidate Ranking Framework for Silicon Debug, *Proc. 26th IEEE VLSI Test Symposium (VTS)*, pp. 352–358 (online), DOI: 10.1109/VTS.2008.60 (2008).

[36] Yu, X. and Blanton, R. D. S.: Multiple Defect Diagnosis Using No Assumptions on Failing Pattern Characteristics, *Proc. 45th IEEE/ACM Design Automation Conference (DAC)*, pp. 361–366 (online), DOI: 10.1145/1391469.1391567 (2008).

[37] Yu, X. and Blanton, R. D. S.: An Effective and Flexible Multiple Defect Diagnosis Methodology Using Error Propagation Analysis, *Proc. 39th IEEE International Test Conference (ITC)*, p. 17.1 (online), DOI: 10.1109/TEST.2008.4700595 (2008).

[38] Yu, X. and Blanton, R. D.: Diagnosis of Integrated Circuits With Multiple Defects of Arbitrary Characteristics, *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 29, No. 6, pp. 977–987 (online), DOI: 10.1109/TCAD.2010.2048352 (2010).

[39] Vogels, T. J., Maly, W. and Blanton, R. D. S.: Progressive Bridge Identification, *Proc. IEEE International Test Conference (ITC)*, pp. 309–318 (online), DOI: 10.1109/TEST.2003.1270853 (2003).

[40] Wunderlich, H.-J. and Holst, S.: *Models in Hardware Testing*, chapter 5 Generalized Fault Modeling for Logic Diagnosis, pp. 133–156 (online), DOI: 10.1007/978-90-481-3282-9, Springer (2010).

[41] Owens, J. D., Houston, M., Luebke, D., Green, S., Stone, J. E. and Phillips, J. C.: GPU Computing, *Proc. of the IEEE*, Vol. 96, No. 5, pp. 879–899 (online), DOI: 10.1109/JPROC.2008.917757 (2008).

[42] Holst, S., Imhof, M. E. and Wunderlich, H.-J.: High-Throughput Logic Timing Simulation on GPGPUs, *ACM Transactions on Design Automation of Electronic Systems*, Vol. 20, No. 3, pp. 37:1–37:22 (online), DOI: 10.1145/2714564 (2015).