Exploring the Mysteries of System-Level Test

Ilia Polian[1], Jens Anders[2], Steffen Becker[3], Paolo Bernardi[4], Krishnendu Chakrabarty[5], Nourhan ElHamawy[1,2]
Matthias Sauer[6], Adit Singh[7], Matteo Sonza Reorda[4] and Stefan Wagner[3]

[1]Institute of Computer Engineering and Computer Architecture, University of Stuttgart, Germany
[2]Institute of Smart Sensors, University of Stuttgart, Germany
[3]Institute of Software Engineering, University of Stuttgart, Germany
[4]Politecnico di Torino, Department of Control and Computer Engineering, Torino, Italy
[5]Department of Electrical and Computer Engineering, Duke University, USA
[6]Advantest Europe, Boeblingen, Germany
[7]Department of Electrical and Computer Engineering, Auburn University, USA

***Abstract*—System-level test, or SLT, is an increasingly important process step in today's integrated circuit testing flows. Broadly speaking, SLT aims at executing functional workloads in operational modes. In this paper, we consolidate available knowledge about what SLT is precisely and why it is used despite its considerable costs and complexities. We discuss the types or failures covered by SLT, and outline approaches to quality assessment, test generation and root-cause diagnosis in the context of SLT. Observing that the theoretical understanding for all these questions has not yet reached the level of maturity of the more conventional structural and functional test methods, we outline new and promising directions for methodical developments leveraging on recent findings from software engineering.**

I. Introduction

System-Level Test (SLT) has emerged as an important additional test insertion in today's semiconductor lifecycle [1]. It is run by the circuit manufacturer in the final stage of production or by the buyer of the circuit, e.g., an automotive Tier-1 supplier who will integrate the circuit into a product, as part of incoming quality control. SLT can also be used during the post-silicon characterization phase where a circuit's extra-functional properties are measured on a population of several hundreds or thousands "first-silicon" circuits.

Conventional structural and functional test methods are based on established theoretical concepts, such as fault models, detection and detectability concepts, coverages. A plethora of algorithms have been invented (and tools implementing these algorithms developed) in the last decades. SLT lacks much of this fundamental understanding; in fact, even the very term "system-level test" is being used in rather different meanings. This paper aims at making first steps towards laying solid theoretical foundations for SLT. Specifically, it discusses the following questions:

- What precisely is SLT that is being used in semiconductor testing? How does it differ from the traditional structural and functional test approaches?
- What are possible reasons for *SLT-unique fails*, i.e., failures observed during SLT in circuits that passed structural and functional tests during earlier test insertions?
- How to determine the root cause of a failure during SLT, in absence of established diagnostic methods?
- How can knowledge from the software engineering domain, e.g., on coverage definitions or on stress test generation, be leveraged for SLT?

In the remainder of the paper, we describe our current knowledge with respect to these questions, touching on related scientific disciplines where necessary. Not all questions have a known answer, and we see it as our objective to capture and discuss the currently discussed explanations or hypotheses, even if they are controversial or contradictory.

II. System-Level Test

A. What is SLT?

The term "system-level test" (SLT) can stand for different types of testing. In the context of integrated circuits (ICs), the following three meanings are predominant:

1) Test of a whole system (e.g., a smartphone or an automotive electronic control unit), focusing on interactions between its components: ICs, sensors, mechanical parts, and the like.
2) Incoming quality control of ICs by a system integrator, to sort out defective ICs and to uncover systematic quality problems of a supplier. The ICs under test are put on a board that imitates the full-system setup and applies to the IC a workload that mimics real-life operation.
3) Outgoing quality control by the IC manufacturer to prevent defective ICs from delivery and to reinforce its own quality control. The procedure is similar to 2), except that the IC manufacturer has less knowledge about the full-system setup but more knowledge about the manufactured IC.

In this paper, we focus on scenario 3), even though most findings are directly applicable to scenario 2). We do not consider scenario 1), where a failing test can point to a defect in one of the system's ICs, to a defect in a different component, or to an erroneous integration. In the SLT scenarios considered here, the system around the IC is assumed to work, and the purpose of testing is to determine whether the IC is defective or not. Fig. 1 visualizes the role of SLT *within the IC manufacturer's quality assurance*, i.e., before the circuit has been shipped to a customer.

While there is no precise definition of "system-level test", it usually refers to applying to the device under test (usually,

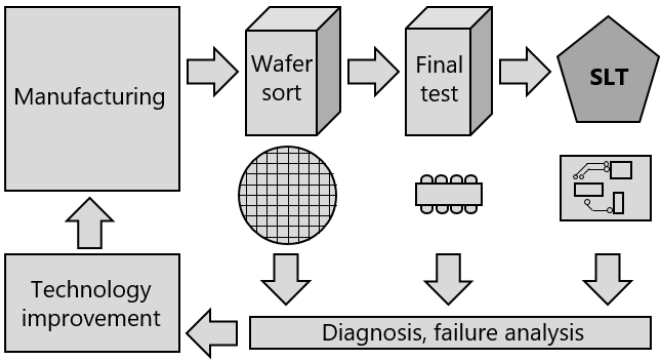

Fig. 1. SLT within the quality-assurance flow

a complex system-on-chip IC) workloads that originate from its intended usage. A popular SLT example is booting an operating system and running several software applications known to stress the system; if the system does not behave as expected (e.g., it crashes), SLT has found a failure. This implies that there is currently no test specification and generation process for SLT: the workload comes from an application scenario. This may change in the future; for example, Chen [1] proposes to use automatically generated design-validation scenarios based on the *Portable Test and Stimulus Standard* (PSS) flow.

B. SLT-induced Costs and Complexities

SLT's benefits are offset by its costs. As it is obvious from Fig. 1, SLT is an additional test insertion that also requires special hardware. The device under test usually needs to be mounted on an evaluation board which includes memories, peripherals and interfaces necessary to run the intended workloads. It is often impossible to perform SLT on regular test equipment that does not include such features, but specialized SLT testers are available. Companies are currently investing in deploying SLT-oriented tester architectures that can be re-used (at least partly) over different products. In this scenario, the more diversified a company's products are, the more they profit from the re-usability of dedicated SLT testers [2].

Another SLT cost factor is its very long test application time in the range of several minutes, a multiple of prior test insertions [1], [3], [4]. This has triggered interest in *adaptive test* methods, where SLT is applied only for a subset of circuits determined during earlier test insertions [5]. For example, Singh [3] (motivated by industrial data reported in [6]) proposes to assign the circuits to bins based on their timing variability observed during pre-SLT test insertions. Every bin corresponds to a certain expected test quality, quantified by the number of defective circuits that will go undetected if no SLT is applied (*defective parts per million* or DPPM). The expected DPPM contribution for each bin is established by a predictive model created by machine-learning from previous experiences. SLT is skipped for circuits from "non-critical" bins associated with (predicted) DPPM contribution below the desired DPPM target. When considering application areas where Burn-In test is required (e.g., automotive), cost reduction can be achieved by combining Burn-In test and SLT. In this case, the tester infrastructure developed for Burn-In (characterized by high parallelism) can be adapted to account for SLT requirements as well [7].

Fine-grained adaptive testing based on quality prediction can also be used to select a subset of chips that must undergo SLT for high-volume production [8], [9]. This strategy includes two key steps. In the first step, parametric test results from an early test insertion are used to train a machine-learning model, which can predict the quality of each chip. A random-forest model is used for quality prediction; the parametric test results of the previous test insertion are used as independent variables, and binary pass/fail results of the current test insertion are used as dependent variables. In the second step, based on the predicted quality, chips are partitioned into two groups using k-means clustering. Test selection is performed for each group individually. SLT can be limited to chips that are predicted to be of low quality. It is shown in [8], [9] using data from three lots, including 71 wafers and 230,000 dies, that fine-grained adaptive testing reduces test cost by up to 7% for a lot, and by as much as 90% for low-quality chips. Moreover, experimental results also show a strong correlation between the predicted quality and marginality of the test outcomes. Therefore, the quality-prediction model can be further used to predict the occurrence of early-life failures.

C. SLT vs. Conventional Testing

It helps the understanding of SLT to contrast its properties with conventional structural and functional test approaches. Table I summarizes the discussion below and in the subsequent sections.

The main difference between SLT and *structural test* is that the latter is strongly based on the notion of a fault according to a *fault model*. Although fault models defined on various levels of abstraction have been introduced in the past [10], [11], most popular models (including stuck-at, bridging and most delay fault models) work on gate level. The stimuli used during testing are usually test patterns either applied through scan infrastructure by automated test equipment (ATE) or generated on-chip by built-in self-test (BIST) or test-compression logic. These patterns are usually produced by automatic test pattern generation (ATPG) procedures that target specific faults. The resulting test set has a fault coverage (number of detected divided by the number of all modeled faults), and there are efficient fault-simulation tools that determine fault coverage.

In contrast, SLT fault models are currently lacking, even though first ideas to define an "SL-FM" and to use it to guide "SL-TG" to generate PSS scenarios for use in SLT are discussed in [1]. Moreover, the application of conventional fault models, such as stuck-at faults, to SLT is practically infeasible because it would necessitate simulation of very long (billions of clock cycles) SLT sequences for every considered fault.

Regarding the test quality, there is no serious discussion of skipping structural test altogether, replacing it with SLT. Structural tests, in addition to covering defects directly represented by the detected faults, usually also detect a large number of unmodeled defects ("fortuitious detection"), such as manifestations of crosstalk or power-supply noise [12], [13]. Therefore, the "baseline DPPM level" is provided by structural

TABLE I
SLT COMPARED WITH STRUCTURAL AND FUNCTIONAL TEST

Aspect	Structural Test	Functional Test	System-Level Test
Level of abstraction	Gate level, sometimes incorporating additional information from layout or RTL	Instruction set architecture	None explicitly considered
Main stimuli format	0/1 values at circuit's inputs / scan flops	Assembly programs	Application or operating system code
Test application	ATE; BIST	ATE; Software-based self-test (from cache)	Evaluation board; SLT tester
Test generation	Fault-oriented ATPG; manual	Manually created test programs; automated techniques	Reuse of verification stimuli; applications; operating system
Quality metrics	Fault coverage	Instruction coverage; fault coverage; coverage used in validation domain	Representative application (no explicit metric)
How assessed?	Fault simulation; sometimes electrical simulation (Spice) of selected patterns	Instruction set level simulation, sometimes fault simulation	No systematic approach available
What defects covered?	Gross and marginal defects represented by considered fault models and beyond	Defects not targeted by structural tests (e.g., small memories without BIST); complex defects missed by structural tests; variability; defects triggered by complex interactions within processor; a major advantage of functional test compared to structural test is the fact that it is performed at speed	Defects missed by structural and functional tests; asynchronous or analog interfaces; clock-gating logic; clock domain boundaries; unknowns (Xes); timing-related failures in uncore logic

tests (it depends on the accuracy of the fault models and the throughness of ATPG). SLT is considered as an additional test insertion if this DPPM level is higher than required by a given application.

The differentiation between *functional test* and SLT is more subtle; in fact, many publications simply treat SLT as a subtype of functional test [1]. However, we believe that there are serious differences between the "traditional" functional test [14] and SLT. The former is based on running relatively small test programs, usually stored in a microprocessor's cache and written, at least partially, in assembly language. Such programs can be created manually, using evolutionary techniques [15] or even deterministic test sequence generation [16].

Functional test programs can be (and are) assessed with respect to detection of traditional gate-level fault models (stuck-at, delay faults, and the like) or special instruction-set level fault models [17]. As was discussed above, meaningful SLT sequences are far too long to be assessed using these fault models or generated by ATPG. A meaningful coverage metric or any other systematic approach to decide whether an SLT suite is "good enough" is currently lacking (see Section IV for some ideas to this end).

In many cases, the functional test (which plays also a major role when considering in-field test) is developed targeting single modules in the IC (CPU, peripherals, memories, interconnections). Hence, it basically aims at checking the correct behavior of each single module in an isolated manner. At the same time, one target of SLT is to check whether the whole device works correctly, exploring for example the effects induced by the interactions among modules. Among the different phenomena triggered by SLT, temperature-related ones play a major role: cases have been reported, where a given defect in an interconnection was only triggered when the temperature gradient between two modules was exceeding a given threshold. Clearly, this kind of defects can hardly be detected by anything different than SLT.

An interesting question is whether functional test and SLT detect the same classes of defects. Both aim at closing the coverage holes of structural tests, and yet one would expect that the expensive SLT would not be applied if the desired test quality were achievable by simpler functional tests alone. One reason for SLT's superiority might be its sheer huge number of test patterns being applied to the circuit during its billions of clock cycles, resulting in a higher chance of fortuitous detections. However, SLT may have systematic advantages, as it is defined on SoC level and incorporates functional interactions between the microprocessor and other SoC components, while functional test tends to focus on the microprocessor itself. For an ultimate answer to this question (and possibly, options to shift some of the detections from SLT to previous insertions), the nature of SLT-unique fails should be better understood. The next section discusses our current understanding of SLT-unique fails.

D. Debug and Diagnosis in SLT Context

When doing SLT, determining the root cause of an observed failure is more difficult than for conventional testing, where efficient diagnosis methods [18], [19] are available. As will be discussed further below, the exact nature of failures observed during SLT is not always clear, and the manifestation of a failure (e.g., a crash) may happen thousands of cycles after its occurrence. Post-silicon validation features such as trace buffers [20] or "quick error detection" logic [21] can alleviate this problem. It was proposed in [22] to use machine learning to establish a relationship between SLT failures and values of over one hundred check status registers within a server-grade processor SoC.

The key idea in [22] is to use a support-vector machine (SVM) to classify SLT failures in the Intel Skylake SoC into one of 18 classes, where each class corresponds to a candidate faulty core or group of cores in the chip. The SVM model was trained using a small number (1,000) of manually created training vectors. A drawback of this approach is that the training data has to be generated manually. Such a manual approach obviously is not practical for high-volume production tests. While it is desirable to use actual fail data from SLT to train the model, a challenge in this context is that SLT fails are "rare events", and it would take a considerable amount of time to generate a sufficient amount of fail data to train the model.

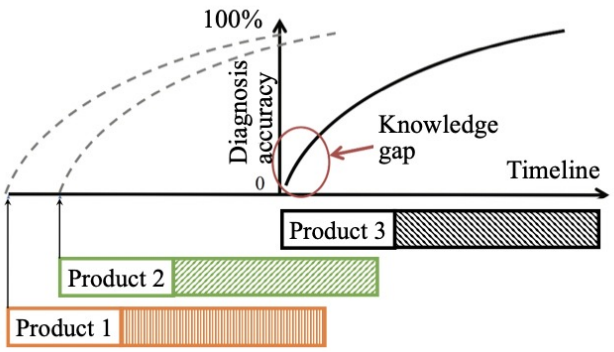

Fig. 2. Diagnostic quality over time

An attractive alternative in this context is to leverage transfer learning techniques and methods for root-cause localization methods using streaming data that have been developed for board-level fault diagnosis [8], [23].

Machine learning-based fault-identification models exploit knowledge from test results and corresponding ground-truth data, without requiring a detailed understanding of the complex functionality of chips. This problem can be formulated as a *supervised classification* problem, where a complete set of learning data consisting of pairs $f(x; y)$. Each instance x is associated with a unique label y, and we refer to each instance-label pair $(x; y)$ as a *sample*. In our application, since different chips may have different fault candidates, we train a binary classifier to diagnose each target fault. The learning algorithm constructs a classifier that outputs a class prediction for a given instance.

In a typical SLT scenario, only a limited amount of test fallout data arrives in the early stages of manufacturing. In fact, fallout data and then associated ground truth about root-cause localization arrives in a streaming format characterized by a potentially large volumes of data instances. As a result, the diagnosis accuracy tends to be low in the early stages of manufacturing. Compared to a static test and diagnosis flow, processing data streams imposes two new requirements on diagnosis algorithms: (1) the ability to adapt to concept drift, and (2) the availability of a limited amount of memory. Online incremental learning algorithms have been proposed for handling streaming data, and to deal with concept drift, classifiers implement forgetting, adaptation and drift detection mechanisms. To overcome the challenge of limited memory, classifiers record only the key information extracted from the previous round of streaming data instead of all the past samples. Moreover, classifiers can learn the target concepts incrementally instead of training from scratch to save training time. By executing online learning algorithms for streaming data, the trained model predicts more accurately when the data distributions shift. This approach has been utilized for fault diagnosis in printed circuits boards [23]. We expect a similar solution to be useful for root-cause localization in SLT.

The diagnosis accuracy improves when more instances of successful root-cause localization in SLT are available for training the diagnosis system. However, there exists a significant knowledge gap in the initial product ramp-up stage (see Fig. 2). In reality, a successful product typically experiences multiple updates and there are often similar products during a period of time. In chip manufacturing, fallout data accumulates over multiple wafers and lots. A similar problem has been addressed for board diagnosis, whereby knowledge learned from a mature board is transferred to the diagnosis model of a new board [8]. A supervised model is trained to identify board-level functional fault using a large number of samples from the mature product (i.e., source domain) and a limited number of samples from the new product (i.e., target domain).

III. SLT-Unique Fails

SLT's main *raison d'être* is its ability to detect failing ICs that are missed during the conventional test insertions. The existing literature suggests a number of sources of these failures, which can be attributed to three broad categories:

1) **Failure mechanisms that are not covered by standard fault models.** For example, traditional scan-based stuck-at and TDF timing tests are node oriented, and explicitly only target the interconnect between the standard cells (gates) in the design. More recently, it was recognized that these classical tests can miss a significant number of defects within complex standard cells, which has led to the development of Cell Aware Tests (CAT) targeting cell internal defects [6]. Scan tests generated using the CAT methodology can significantly reduce subsequent SLT fallout in some applications. However, marginal, or "soft", timing failures [3] that manifest themselves only under certain operating conditions (voltage, temperature) [1], [4], and power-supply instabilities in conjunction with complex power-management schemes [4], remain challenging to detect using the available scan-based delay test methods.
2) **Systematic ATPG coverage holes.** Complex SoC designs contain signal lines connecting several, even dozens, of different clock domains. Faults in the logic structures at the clock domain boundaries, asynchronous or analog interfaces or clock distribution networks often need careful manual expert intervention during test generation to ensure reliable detection. These may be conservatively classified as "untestable" by automated ATPG tools, even though the faults can manifest themselves during the device's operation. More mundane causes of incomplete test coverage are test time and tester memory limitations. At times some scan patterns that detect only a few faults are dropped to reduce memory requirements or test-application time; this "long-tail" problem is also alleviated by fortuitous detection by the very long SLT test sequences.
3) **Faults exposed only during system-level interactions.** This includes complex software-controlled clock- and power-domain interactions or resource contention in a multi-core system that cannot be fully replicated on an ATE [1]. Other possible situations include complex hardware-implemented protocols or "soft" failures during high-speed memory accesses [4]. The key characteristics of these failures is that the electrical and timing interaction of the IC under test with other components

in the target system are insufficiently defined, or too complex, to be accurately modelled on a tester.

Since using SLTs as an additional final test screen during post manufacturing tests imposes significant added costs, considerable efforts have focused on minimizing test escapes from traditional structural scan-based testing so as to eliminate, or at least reduce, the need for SLTs. In some applications, SLT is critically needed only during the early yield ramp phase of production, when the yield of defect free parts is low, as test escapes are obviously more numerous when there are more defective parts being tested. Once the manufacturing process matures and yields improve, SLT is only performed in these applications on sample parts for quality assurance purposes—to ensure acceptable defect levels in the shipped ICs.

An increasing number of high-volume applications, e.g. high-end cell phone processor SoCs, continue to require SLT on all manufactured parts throughout the product lifetime. This has motivated considerable research on understanding the test escapes from traditional scan tests that are uniquely detected by SLTs. The goal is to improve the coverage of these failures by the scan tests, and thereby reduce dependence on SLTs. In revisiting the three categories of test escapes listed above, it is obvious that (3), faults exposed only during system-level interactions, cannot by definition be targeted using traditional test methods. Plugging coverage holes (2) caused by the inability of ATPG tools to generate some tests, e.g., across timing domain boundaries, and for asynchronous and analog mixed signal circuit structures, is a well understood and a longstanding challenge. It is currently the focus of significant development effort at major EDA companies.

However, until recently, considerable mystery has surrounded Category (1), failures that are not covered by fault-model-based scan tests but uniquely detected by SLT. The reason for this uncertainly is the poor diagnostic capability of SLTs. Unlike in scan tests where each applied test input pattern and corresponding circuit response is known, it is extremely difficult to root cause a failure observed by functional SLT down to a logic gate. Failure in functional operation may be observed thousands, even millions, of clock cycles after the underlying logic level malfunction occurs, making it impossible to trace and locate (see Section II-D). It is virtually impossible to confirm the cause of most of the SLT failures that escape scan tests. This makes it difficult to target them with new fault models.

The new Cell Aware Test generation methodology aimed at detecting shorts and opens within standard cells has been successful in reducing SLT fallout in some applications, but less so in others [6]. The distinction appears to be the susceptibility of the design to "soft" timing errors, which are not caused by the "hard" defects targeted by CAT. Particularly vulnerable to timing failures are power constrained applications such as cell phones that implement aggressive power management to ensure battery life while meeting ever increasing computational demands. This involves dynamic voltage-frequency scaling, with the circuit operated at slow clock frequencies and low energy saving voltages when computational loads are minimal. Unfortunately, the impact of manufacturing process variations on circuit timing is greatly amplified during very low voltage operation, causing some circuits to experience occasional timing errors. Ideally, such "defective" ICs containing extremely slow statistical outlier transistors from random process variations should be detected by the scan timing tests. However, the widely used transition delay fault (TDF) model explicitly only targets a single lumped delay in the circuit. It is unable to reliably detect accumulated delays along circuit paths resulting from a distribution of delays across the IC due to the effects of random process variations. What are required are effective scan path delay tests, which have so far not proven practical. Variation-aware tests [24] are, therefore, not part of the scan test set, resulting in the increasing dependence on SLTs in power constrained low voltage applications.

IV. Assessing the Quality of SLT Programs

As was already mentioned above, the standard quality metrics normally used during IC testing (fault coverages) are not practically applicable to SLT, just because the workloads that are being applied for minutes would be impossible to simulate. The only suggestion for an SLT-aware metric proposed so far is the number of used scenarios in the context of the PSS-based SLT flow [1]. In the following, we discuss coverage concepts used during software integration test and whether it can play a role during SLT.

Overall, software integration test is—similarly to SLT—not as well investigated as unit tests on the lower levels. For the latter, coverage is often used to describe the completeness of the tests. The coverage most often encompasses control flow (such as statement coverage or branch coverage) but can also relate to the data flow (such as all definitions). This translates badly to integration tests as it then still relies on which statements in the integrated components are executed, just with a relation to the interface of the components [25].

An alternative approach [26] sees the integrated components as black boxes. Motivated by the application in the automotive domain, in which many software components are developed by suppliers without revealing the source code, it aims at understanding if the components are well integrated only based on the information from the data flow between them.

To achieve this, the observations on shared data between components are described and classified into *preconditions* that represent the states of the components, *stimulations* that capture manipulations of shared data, and *verification* that represents observed data that can be used to check the behaviour based together with preconditions and stimulations. This gives rise to new coverage criteria, for example, the coverage of all usages of shared data ("shared-data-use"), or the verification usage of a shared data ("verification-data-use") in which a test must use a shared data to verify behaviour [26]. Using this approach for automotive integration testing, test gaps were identified. It appears promising to use similar coverage criteria for SLT.

Some high-level parameters are playing an important role in the detection of failures during the SLT insertion. To stress the component to facilitate the insurgence of an SLT defect is often not a pure duty of the SLT software, but it requires some extra conditions to be met. The characterization of an SLT program must also carefully consider the effects of different temperatures (both high and low) and temperature gradients.

In most cases when temperature is a factor to consider, the requested temperature conditions are provided by climatic

chambers. Anyway, a stressful (therefore valuable) SLT workload should also show self-heating capabilities to reach a specific junction temperature as indicated in mission profiles. Furthermore, it should also implement heat control mechanisms to avoid over stress and sometimes the thermal overrun.

Testing at harsh power supply conditions is another very common industrial practice in SLT; power supply voltage for core and other power domains is increased or decreased even to make the test running out of the functional specifications of the product. This power supply variation is aimed at screening out latent faults that may easily show up in the early life of the IC. Obviously, this may turn into a dangerous practice if the SLT workload was not very carefully graded in term of punctual power demands.

V. SLT Program Generation

While the current state of the art for SLT is to use existing software (thus requiring no explicit generation), testing for specific problems can call for SLT programs with specific characteristics. For instance, if we know that a particular part of the system is vulnerable to subtle failures, an SLT workload that stresses this part of the system will be useful. This is related to the "power-virus" generation considered (for much smaller circuits) in the past [27].

Thermal measurement on physical samples like in [28] is an important practice to also provide physical findings about the implemented SLT workload. Warming up the silicon surface of a device is often asking quite a long time (up to minutes) before the circuit shows the desired (usually high) temperatures. Such a thermal characterization is usually done by thermocamera-based experiments. This experimental observation can be supported by suitable firmware read-out of the chip the temperature recorded by the sensors embedded in it.

The creation of a functional oriented program able to control the thermal behaviour should not address the entire circuit in one shot, but rather focus on different circuit zones at different times. Differently from scan testing, the focused functionalities during SLT should be triggered by a suite of workloads targeting relevant circuit portions.

VI. Conclusion

Looking back at conventional integrated circuit test several decades ago, we see a scientific success story, which made quality assurance feasible and kept its cost reasonable throughout the long period of exponential circuit complexity growth. A (largely) common understanding of terms and concepts throughout a large community, both industrial and academic, has led to sophisticated and yet practical solutions that could be adopted by most relevant players. We believe that SLT needs to reach a similar level of widely agreed-upon understanding to enable comparable progress and overcome its foreseeable limitations. To answer the questions posed in this paper, a new level of cross-sector collaboration between semiconductor manufacturers, system integrators, test equipment manufacturers, EDA companies and academia will be necessary. For instance, reliable information on SLT programs that are practically effective, and on SLT-unique fails identified by such programs, will be helpful in giving the necessary research, e.g., on improved generation of SLT programs and systematic assessment of their quality, a meaningful direction.

Acknowledgment

This work was supported by Advantest as part of the Graduate School "Intelligent Methods for Test and Reliability" (GS-IMTR) at the University of Stuttgart.

References

[1] H. H. Chen, "Beyond structural test, the rising need for system-level test," in *VLSI-DAT*. IEEE, 2018, pp. 1–4.
[2] "ATS 5034 System Level Test (SLT) Platform," https://www.advantest.com/system-level-test-systems/ats-5034, accessed: 2020-07-20.
[3] A. D. Singh, "An adaptive approach to minimize system level tests targeting low voltage DVFS failures," in *ITC*. IEEE, 2019, pp. 1–10.
[4] P. Bernardi, M. Restifo, M. S. Reorda *et al.*, "Applicative system level test introduction to increase confidence on screening quality," in *DDECS*, 2020.
[5] S. Letchumanan *et al.*, "Adaptive test method on production system-level testing (SLT) to optimize test cost, resources and defect parts per million (DPPM)," in *VLSI-DAT*, 2018, pp. 1–3.
[6] F. Hapke *et al.*, "Defect-oriented test: Effectiveness in high volume manufacturing," *IEEE Trans. CAD*, 2020 (Early Access).
[7] F. Almeida, P. Bernardi *et al.*, "Effective screening of automotive SoCs by combining burn-in and system level test," in *DDECS*, 2019.
[8] M. Liu, X. Li, K. Chakrabarty, and X. Gu, "Knowledge transfer in board-level functional fault identification using domain adaptation," in *International Test Conference*. IEEE Computer Society, 2019.
[9] M. Liu, R. Pan, F. Ye, X. Li, K. Chakrabarty, and X. Gu, "Fine-grained adaptive testing based on quality prediction," *ACM Transactions on Design Automation of Electronic Circuits and Systems*, pp. 1–1, 2020.
[10] S. Natarajan, S. K. Gupta, and M. A. Breuer, "Switch-level delay test," in *ITC*. IEEE Computer Society, 1999, pp. 171–180.
[11] J. Yi and J. P. Hayes, "High-level delay test generation for modular circuits," *IEEE Trans. CAD*, vol. 25, no. 3, pp. 576–590, 2006.
[12] L. Wang, S. K. Gupta, and M. A. Breuer, "Modeling and simulation for crosstalk aggravated by weak-bridge defects between on-chip interconnects," in *Asian Test Symp.*, 2004, pp. 440–447.
[13] I. Polian, "Power supply noise: Causes, effects, and testing," *J. Low Power Electron.*, vol. 6, no. 2, pp. 326–338, 2010.
[14] P. C. Maxwell, I. Hartanto, and L. Bentz, "Comparing functional and structural tests," in *ITC*. IEEE Computer Society, 2000, pp. 400–407.
[15] E. Sánchez, M. Schillaci, M. S. Reorda, G. Squillero, L. Sterpone, and M. Violante, "New evolutionary techniques for test-program generation for complex microprocessor cores," in *GECCO*, 2005, pp. 2193–2194.
[16] A. Riefert, R. Cantoro, M. Sauer, M. S. Reorda, and B. Becker, "A flexible framework for the automatic generation of SBST programs," *IEEE Trans. VLSI*, vol. 24, no. 10, pp. 3055–3066, 2016.
[17] S. M. Thatte and J. A. Abraham, "Test generation for microprocessors," *IEEE Trans. Computers*, vol. 29, no. 6, pp. 429–441, 1980.
[18] S. Venkataraman and S. B. Drummonds, "Poirot: Applications of a logic fault diagnosis tool," *IEEE Design&Test*, vol. 18, no. 1, pp. 19–30, 2001.
[19] S. Holst and H. Wunderlich, "Adaptive debug and diagnosis without fault dictionaries," *J. Elec Testing*, vol. 25, no. 4-5, pp. 259–268, 2009.
[20] K. Iwata, A. M. Gharehbaghi, M. B. Tahoori, and M. Fujita, "Post silicon debugging of electrical bugs using trace buffers," in *ATS*. IEEE Computer Society, 2017, pp. 189–194.
[21] D. Lin *et al.*, "Effective post-silicon validation of system-on-chips using quick error detection," *IEEE Trans. CAD*, vol. 33, no. 10, pp. 1573–1590, 2014.
[22] L. D. Rojas, K. Hess, and C. Carter-Brown, "Effectively using machine learning to expedite system level test failure debug," in *ITC*, 2019.
[23] M. Liu, F. Ye, X. Li, K. Chakrabarty, and X. Gu, "Board-level functional fault identification using streaming data," in *VLSI Test Symp.*, 2019.
[24] M. Sauer *et al.*, "Variation-aware deterministic ATPG," in *ETS*, 2014.
[25] A. Spillner, "Test criteria and coverage measures for software integration testing," *Software Quality Journal*, vol. 4, no. 4, pp. 275–286, 1995.
[26] D. Hellhake, T. Schmid, and S. Wagner, "Using data flow-based coverage criteria for black-box integration testing of distributed software systems," in *IEEE Conf. on Software Testing, Validation and Verification*, 2019, pp. 420–429.
[27] K. Najeeb *et al.*, "Power virus generation using behavioral models of circuits," in *VTS*, 2007, pp. 35–42.
[28] D. Appello, P. Bernardi *et al.*, "A comprehensive methodology for stress procedures evaluation and comparison for burn-in of automotive SoC," in *DATE*, 2017, pp. 646–649.