Assessing the Effectiveness of Software-Based Self-Test Programs for Cell-Aware Test

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 8 May 2026, accepted 15 May 2026, date of publication 22 May 2026, date of current version 28 May 2026.

Digital Object Identifier 10.1109/ACCESS.2026.3696050

RESEARCH ARTICLE

Assessing the Effectiveness of Software-Based Self-Test Programs for Cell-Aware Test

REZA KHOSHZABAN[1], (Graduate Student Member, IEEE), IACOPO GUGLIELMINETTI[2], MICHELANGELO GROSSO[2], (Senior Member, IEEE), MATTEO SONZA REORDA[1], (Fellow, IEEE), AND RICCARDO CANTORO[1], (Member, IEEE)

[1]DAUIN, Politecnico di Torino, 10129 Turin, Italy
[2]STMicroelectronics, 10129 Turin, Italy

Corresponding author: Reza Khoshzaban (reza.khoshzaban@polito.it)

This work was supported by the Project Piano Nazionale di Ripresa e Resilienza (PNRR)-Next Generation EU (NGEU) from the Ministero dell'Università e della Ricerca (MUR) under Grant DR 117/2023.

ABSTRACT Software-Based Self-Test (SBST) is a widely adopted technique for ensuring the in-field reliability of safety-critical systems, particularly in the form of Software Test Libraries (STLs). While the effectiveness of SBST is traditionally assessed using classical fault models such as stuck-at and transition delay faults, these models may not fully capture the behavior of modern nanometric technologies. In contrast, Cell-Aware Testing (CAT) has emerged as a defect-oriented approach that more accurately reflects physical defects at the cell level, and is gaining traction in manufacturing test flows. This paper presents a structured framework for evaluating SBST programs under defect-oriented Cell-Aware Test (CAT) models and interpreting the results at the defect level. The proposed approach combines defect-to-conditional fault mapping, a simplified observability-based taxonomy, and a post-processing flow to derive accurate defect-level coverage. In addition, a table test percentage (TT%) metric is introduced and exploited to quantify intrinsic defect detectability and guide test improvement. Together, these techniques enable a consistent defect-level interpretation of fault simulation results, reduce the overestimation inherent in raw conditional-fault coverage, and provide actionable insight for identifying and prioritizing hard-to-detect defects. The proposed framework is validated on a RISC-V System-on-Chip using multiple Software Test Libraries and two technology libraries. Experimental results show that, while existing STLs achieve substantial coverage on CAT defects, a subset of defects remains systematically hard to detect due to intrinsic observability constraints. The proposed framework identifies critical hard-to-detect defects, enables targeted STL refinement, and remains compatible with industrial defect-oriented validation flows.

INDEX TERMS Software-based self-test, functional test, in-field test, functional safety, fault simulation, cell-aware testing.

I. INTRODUCTION

Modern electronic systems used in safety-critical applications, such as automotive or aerospace, must meet stringent quality and reliability standards. Ensuring that latent hardware defects are effectively detected and managed is essential. In this context, there is a growing need for methodologies that quantify how effective existing in-field test solutions are under advanced defect-oriented fault models. While manufacturing tests play a key role in filtering defective devices, traditional fault models—such as stuck-at fault (SAF) and transition delay fault (TDF)—have become insufficient to meet the extremely low defective parts per million (DPPM) levels now required.

To address this issue, defect-oriented fault models—like Cell-Aware Testing (CAT) [1], [2], [3] and Small Delay Defects (SDDs) [4], [5], [6]—have gained traction in automatic test pattern generation (ATPG) for production testing. These approaches have been shown to significantly improve DPPM metrics [7]. On the in-field side, functional safety standards such as ISO 26262 mandate achieving a certain level of diagnostic coverage (DC), yet remain agnostic

The associate editor coordinating the review of this manuscript and approving it for publication was Poki Chen.

regarding which fault models should be used to assess the quality of a test and to guide test generation [8], [9]. As a result, in-field test strategies still predominantly target SAFs [10], [11].

When hardware redundancy is infeasible or impractical, software-based safety mechanisms, such as Software Test Libraries (STLs), are often used [12]. STLs are based on the Software-Based Self-Test (SBST) paradigm [13], where the processor executes test programs designed to detect faults through functional behavior. STLs can operate in-system at-speed with minimal intrusion and can detect faults that may escape scan-based techniques. Despite their growing importance, little work has been devoted to assessing STL effectiveness against advanced defect-oriented models [14], [15].

The contribution of this work is not limited to extending coverage evaluation to additional fault models, but addresses a fundamental gap in SBST assessment under defect-oriented testing. Existing approaches typically rely on simulator-level fault models and implicitly assume that the resulting coverage metrics reflect defect detectability. However, for defect-oriented models such as CAT, this assumption does not hold, as multiple conditional faults may correspond to a single physical defect, and their detectability depends on complex activation and observability constraints.

To address this limitation, this work introduces a structured defect-level evaluation framework that bridges simulator-level fault representations and physically meaningful defect behavior. The proposed approach combines three key elements: (i) a defect-to-conditional-fault mapping and post-processing flow that reconstructs accurate defect-level coverage, (ii) a *Table Test Percentage* (TT%) metric that captures intrinsic defect detectability, and (iii) a decision-oriented analysis linking defect detectability to Software Test Library (STL) design, enabling targeted rather than indiscriminate test improvements.

Compared to our prior work [16], which focused primarily on static CAT evaluation and coverage reporting, this paper extends the analysis along two orthogonal dimensions. From a technical perspective, it includes dynamic CAT defects and multiple technology libraries. More importantly, from a methodological perspective, it shifts the focus from raw coverage estimation to defect-level interpretation and decision-oriented analysis. While previous studies quantify how much of the fault space is covered, the proposed framework focuses on defect-level interpretation and its use for guiding STL refinement.

Given that CAT is a superset of the SAF model, the proposed framework enables analyzing whether redundancy introduced during STL development also contributes to CAT fault detection. This framework is validated on a RISC-V System-on-Chip (SoC) using openly available STLs [16]. It also enables a comparative evaluation of SAF, TDF, and CAT behaviors. Additionally, due to the limitations of commercial CAT fault simulators in handling sequential circuits, we define a custom simulation flow including the pre-processing and post-processing steps required to derive CAT fault coverage correctly.

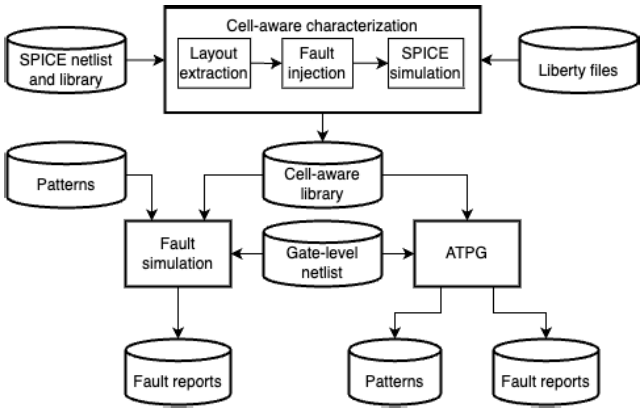

FIGURE 1. The cell-aware test flow for ATPG and fault simulation.

By enabling a consistent mapping between fault-level simulation results and defect-level behavior, the proposed framework provides a missing interpretability layer in SBST evaluation. This allows moving beyond coverage-centric assessment toward causality-driven analysis, where residual defects can be characterized, prioritized, and systematically targeted during STL development.

The remainder of this paper is structured as follows: Section II reviews relevant background on CAT and SBST. Section III describes the simulation flow used. Experimental results and their analytical interpretation are presented in Section IV, and conclusions are drawn in Section V.

II. BACKGROUND

A. CELL-AWARE TEST

Traditional fault models like SAF, TDF, bridging, or SDDs enumerate faulty locations in the circuit simply by considering the boundary of the standard cells. Such faults are excited by bringing proper logical values (or transitions, in the case of dynamic fault models) toward the faulty locations. Subsequently, the fault differences are propagated through one or more paths by forcing the off-path signals to transparent values (e.g., 1s for AND, 0s for OR gates), until reaching an observable point (e.g., a flip-flop or a primary output). Contrarily, the CAT methodology targets electrical-level defects occurring within cells, which patterns generated targeting traditional fault models may fail in detecting [1], [2], [3].

CAT requires an electrical-level characterization of each library cell. After this characterization step, the resulting cell-aware library can be imported into the ATPG or fault simulation flows (see Fig. 1), where each gate instance is associated with the corresponding defects and activation conditions. As a result, ATPG and fault simulation can still rely on conventional algorithms, but with additional constraints imposed by the defect model. Most commercial Electronic Design Automation (EDA) tools support CAT ATPG and fault simulation for scan design, while the support for functional fault simulation is currently very limited.

TABLE 1. Example of static defect matrix for a full-adder cell.

A	*B*	*CI*	*CO*	*S*	*D*1	*D*2	*D*3	*D*4	*D*5	...
0	0	1	0	1	0	0	1	2	3	...
0	1	0	0	1	3	1	1	2	1	...
1	0	1	1	0	0	2	0	2	0	...
1	1	0	1	0	0	1	0	0	3	...
...	...	...	...	...	...	...	...	...	...	...

TABLE 2. Example of dynamic defect matrix for a full-adder cell.

A	*B*	*CI*	*CO*	*S*	*D*6	*D*7	*D*8	*D*9	*D*10	...
0	1	R	R	F	0	0	0	0	3	...
R	1	0	R	F	0	1	0	2	0	...
0	F	1	F	R	2	1	3	2	3	...
0	1	F	F	R	2	0	0	1	0	...
0	R	0	0	R	2	0	0	0	0	...
1	F	1	1	F	0	1	0	0	0	...
...	...	...	...	...	...	...	...	...	...	...

The cell-aware characterization starts from the *layout extraction* for each cell. A tool analyzes the cell's transistor-level netlist in this phase with layout information. It extracts a list of possible defects, such as parasitic resistors and coupling capacitors or resistive bridges, shorts, or opens. In the subsequent *fault injection* phase, a characterization tool analyzes the transistor-level netlist, derives candidate defects, and injects them into faulty netlists. *SPICE-level simulation* is then used to determine whether each defect manifests as a *static* or *dynamic* fault and to identify the corresponding activation conditions. Once all defects have been analyzed, equivalent defects are collapsed and the resulting model is exported, typically as a user-defined fault model (UDFM) or a Cell Test Model (CTM).

In our experiments, we used the CTM format. The CTM file includes static and dynamic defect matrices for each cell. Tables 1 and 2 show a snippet of the static and dynamic defect matrix of a full-adder cell, respectively. For each row, the tables report the input values, the expected fault-free output values, and a code for each defect indicating on which outputs the defect is observable. In the dynamic case, the symbols R and F denote slow-to-rise and slow-to-fall behavior, respectively. The bit-string is used to specify whether the defect manifests itself on the corresponding output. In the example, 1 means the defect can be detected on the first output (CO), 2 on the second output (S), and 3 on both.

The defect matrix does not directly list the faulty output values. Instead, it indicates on which outputs the waveform of the defective circuit deviates from the defect-free waveform. For example, in Table 1, when defect $D1$ is activated by input [0, 1, 0], code 3 indicates observability on both outputs CO and S, meaning that both outputs differ from their fault-free values. Using the same input value but injecting $D2$ only, the code 1 indicates that the defect is observable only on CO.

Each defect can also be detectable under multiple input combinations. The same interpretation applies to the dynamic defect matrix in Table 2, where defect observability is evaluated over pairs of vectors. For example, defect $D8$ is observable on both outputs for input transition [0, F, 1], while other defects may affect only one output or require different transitions to become observable.

B. SOFTWARE-BASED SELF-TEST

In the past, various alternative and complementary methods have been developed and utilized to broaden the range of testing techniques available to designers and product engineers in lieu of scan chain-based testing. One such approach is the SBST method, which involves applying functional stimuli to an on-chip microprocessor to execute specific code and detect structural faults within the logic by observing the results produced by the code execution. This method can detect faults at the nominal circuit frequency without introducing any Design for Testability (DfT) features and acting in operational mode. However, the test generation and coverage assessment processes are not as standardized, automated, and mature as for DfT-based test. The use of advanced semiconductor technologies, particularly for safety-critical applications requiring high reliability, has led to the wide adoption of SBST for in-field testing. This is done through STLs developed by the semiconductor manufacturer and integrated by the system company into the application code. STLs are guaranteed to achieve a given fault coverage and can be activated (as a whole or in chunks) with a given frequency, depending on the required safety level.

Previous works addressed SBST generation targeting various fault models, such as SAFs [17], TDFs [18], [19], [20], [21], SDDs [14], [22] and path delays [23], [24], [25], [26], while the use of CAT is relatively new [15]. Regarding the methodology for generating test programs, manual approaches and partly or fully automated techniques have been used, based on ATPGs, SAT-solvers, and evolutionary algorithms [13].

III. FAULT SIMULATION FLOW

SBST fault grading requires simulating the top-level circuit in functional mode while injecting faults and observing their effects on the observable points (e.g., memory elements, the external bus, or specific primary outputs of the target circuit). Fault simulators for functional safety assurance exist, but the native support to CAT is limited. The fault simulator used in our flow supports the identification and simulation of *conditional faults*, faults on the output pins of a cell with constraints on the cell's inputs, using the static and dynamic defect matrices of CTM files. However, such conditional faults are specific to a single cell's output and have a fixed polarity (stuck-at-0 (sa0), stuck-at-1 (sa1), slow-to-rise (str) or slow-to-fall (stf)). As each defect can be excited by some patterns that excite more than one condition (e.g., $D4$ in the example in Table 1 or $D7$ in Table 2), possibly on multiple gate's outputs (e.g., $D1$ or $D8$ in the same example), the fault grading process must consider the defect as tested if any of the multiple conditional faults is tested.

The proposed fault simulation flow maps each CAT defect to a set of conditional faults derived from the characterization matrices, and then reconstructs defect-level coverage through post-processing. To support this mapping, defects are classified according to (i) the number of affected cell outputs and (ii) the number of conditional faults required per output. This classification captures the activation and observability complexity of each defect, which directly affects detectability.

At a high level, the taxonomy distinguishes the following classes:

- *Single-output / single-fault*: one output, one conditional fault.
- *Single-output / multiple-faults*: one output, multiple conditional faults.
- *Multiple-outputs / single-fault*: multiple outputs, one conditional fault per output.
- *Multiple-outputs / variable-faults*: multiple outputs with mixed fault multiplicity.
- *Multiple-outputs / multiple-faults*: multiple outputs, each requiring multiple conditional faults.

These classes reflect increasing activation and observability complexity. In particular, defects involving multiple outputs and multiple conditional faults are the most challenging cases for defect-level reconstruction and typically correspond to the hardest-to-detect CAT behaviors analyzed in Section IV.

The taxonomy provides a systematic way to translate CAT defects, as described by CTM matrices, into the conditional faults supported by the simulator. This ensures that the generated fault lists remain consistent with the defect model and also identifies which conditional faults must be considered jointly during post-processing. As a result, defect-level coverage can be reconstructed accurately without overestimating or underestimating the effectiveness of the test set. The complete taxonomy, including detailed definitions, examples, and bounds on the number of conditional faults per defect, is reported in Appendix.

The fault simulation produces a fault list containing conditional faults, which must then be post-processed and mapped back to CAT defects. The post-processing phase shown in Fig. 2 splits the fault list into sets of conditional faults, each one corresponding to a defect, and checks whether at least one of the faults is marked as detected. In that case, the defect is also marked as detected. If none of the faults is marked as detected, we check whether any fault is potentially detected; otherwise, the defect is marked as not detected. In more complex fault grading processes (e.g., for functional safety assurance), where additional tags can be used to mark faults, one can adapt the post-processing flow by adding additional conditions.

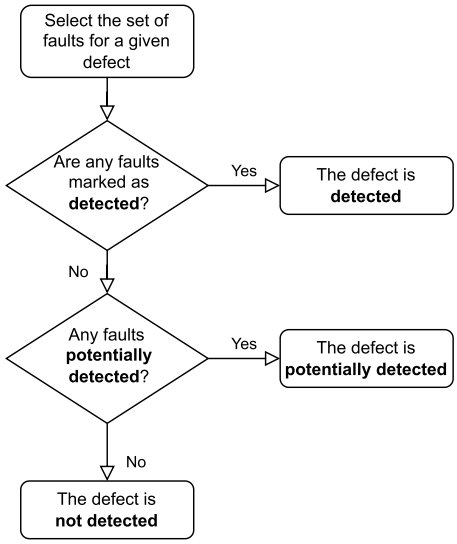

FIGURE 2. Example of fault simulation post-processing used to map conditional faults to CAT defects.

IV. EXPERIMENTAL RESULTS

A. EXPERIMENTAL SETUP AND FAULT LISTS

We have assessed the effectiveness of SBST programs in detecting CAT faults on the RI5CY processor, a 4-stage, 32-bit in-order RISC-V processor core embedded in the PULPino SoC [27]. The design was synthesized using two technology libraries: the Silvaco OpenCell 45 nm FreePDK library (denoted as $C1$) [28] and a proprietary 130 nm HCMOS technology for power applications by STMicroelectronics (denoted as $C2$). This enables a comparative study across both an open and an industrial technology library. We used a set of openly available stuck-at STLs [29].

The experimental flow uses Synopsys Design Compiler for logic synthesis, Synopsys CMGen for CAT characterization, Siemens QuestaSim for logic simulation, and Synopsys Z01X for fault simulation. Experiments on $C1$ were run on an Intel Xeon CPU E5-2680 v3 machine, while experiments on $C2$ were run on an Intel Xeon Gold 6242R CPU machine.

Due to limitations in currently available fault simulation tools, the analysis is restricted to defects affecting combinational cells and to the subset of dynamic CAT faults that can be mapped to supported conditional fault models. While this reduces the completeness of the evaluated defect space, the excluded defects account for approximately 6% of the total CAT defect set, consistently across both technology libraries considered in this work.

Combinational logic captures the dominant activation and propagation mechanisms exercised by SBST programs, and defects affecting sequential elements are ultimately observed through combinational paths. Similarly, unsupported dynamic CAT faults correspond to behaviors that cannot be expressed within the available conditional fault abstractions, but do not introduce fundamentally different observability mechanisms.

As a result, although absolute coverage values may change when extending the analysis to the full defect space, the relative trends observed in this work—particularly the relationship between TT% and defect detectability—remain representative. Extending the framework to fully support

TABLE 3. Static faults in C1 (combinational cells only).

Module	#SAF	#SCond	#SCAT
CS regs	6,158	17,537	17,537
Debug	2,596	7,093	7,093
EX/ALU	25,754	88,611	81,075
EX/MUL	30,930	204,525	144,682
ID/Control	1,322	3,593	3,593
ID/Decoder	3,422	9,219	9,219
ID/HW loop regs	3,332	12,714	12,714
ID/INT control	92	379	379
ID/regs	33,324	97,880	97,880
IF/compr, dec,	1,476	3,959	3,959
IF/HW loop control	1,628	4,449	4,446
IF/Prefetch buffer	9,648	27,134	27,133
Load Store	4,054	16,069	13,488
CPU (TOP LEVEL)	141,990	554,118	480,362

TABLE 4. Static faults in C2 (combinational cells only).

Module	#SAF	#SCond	#SCAT
CS regs	6,116	21,868	14,789
Debug	2,506	8,610	6,288
EX/ALU	29,762	102,261	71,750
EX/MUL	51,190	203,663	131,656
ID/Control	1,584	3,746	2,700
ID/Decoder	3,902	9,957	7,207
ID/HW loop regs	3,932	11,742	8,197
ID/INT control	116	324	228
ID/regs	32,640	115,503	82,014
IF/compr, dec,	1,564	4,627	3,344
IF/HW loop control	1,296	3,319	2,340
IF/Prefetch buffer	8,334	30,376	21,412
Load Store	5,692	18,607	9,920
CPU (TOP LEVEL)	172,728	613,385	421,290

sequential elements and additional dynamic defects is part of future work and would primarily increase completeness rather than alter the underlying conclusions.

Importantly, these limitations are related to the current capabilities of commercial fault simulation tools rather than to the proposed framework itself. The methodology remains applicable independently of the specific supported fault classes and can naturally extend to the excluded defects as tool support evolves, without requiring modifications to the framework.

In the considered technology libraries, cells have at most two outputs, resulting in up to four static and eight dynamic conditional faults per defect, consistent with the bounds assumed by the taxonomy in Section III.

B. FAULT LIST POST-PROCESSING AND DEFECT COLLAPSING

Z01X does not natively collapse these conditional faults to CAT defects: this mapping is instead performed by a dedicated post-processing step implemented as an external script, which follows the procedure outlined in Fig. 2.

Tables 3 and 4 report the number of SAFs, conditional faults derived from static CAT detection tables before post-processing (*SCond*), and static CAT defects after post-processing (*SCAT*) for the main modules of the core. A first key observation is that static CAT significantly expands the fault space, yielding roughly three to four times more defects than SAFs. Post-processing then collapses the conditional-fault representation to defect level, reducing the total fault count by about 15% in $C1$ and by about 32% in $C2$. This confirms that raw conditional fault counts overestimate the effective static CAT fault space, especially in the industrial library.

Tables 5 and 6 report the corresponding figures for TDFs, conditional faults derived from dynamic CAT detection tables before post-processing (*DCond*), and dynamic CAT defects after post-processing (*DCAT*). The dynamic CAT fault space is even larger, reaching about four times the TDF count in $C1$ and about eight times the TDF count in $C2$. Post-processing has a much stronger effect in this case, collapsing about 56% of DCond faults in $C1$ and about 51% in $C2$. These results highlight both the richness of the dynamic CAT model and the importance of reconstructing defect-level coverage from conditional-fault simulation results.

TABLE 5. Dynamic faults in C1 (combinational cells only).

Module	#TDF	#DCond	#DCAT
CS regs	6,118	60,724	25,427
Debug	2,576	23,209	10,083
EX/ALU	25,722	217,313	99,121
EX/MUL	30,618	185,182	86,709
ID/Control	1,322	11,164	5,054
ID/Decoder	3,418	28,645	13,011
ID/HW loop regs	3,324	37,044	18,320
ID/INT control	92	1,037	544
ID/regs	33,224	354,484	141,667
IF/compr, dec,	1,476	12,545	5,604
IF/HW loop control	1,624	14,181	6,105
IF/Prefetch buffer	9,608	90,100	38,793
Load Store	4,050	35,913	15,761
CPU (TOP LEVEL)	141,326	1,235,789	539,942

TABLE 6. Dynamic faults in C2 (combinational cells only).

Module	#TDF	#DCond	#DCAT
CS regs	6,116	107,483	48,947
Debug	2,506	45,778	21,189
EX/ALU	29,762	502,335	245,977
EX/MUL	51,190	661,107	355,855
ID/Control	1,584	26,852	13,058
ID/Decoder	3,902	70,417	33,017
ID/HW loop regs	3,932	61,675	32,851
ID/INT control	116	1,726	943
ID/regs	32,640	600,405	284,269
IF/compr, dec,	1,564	28,065	13,103
IF/HW loop control	1,296	21,838	10,049
IF/Prefetch buffer	8,334	146,939	66,993
Load Store	5,692	105,880	49,578
CPU (TOP LEVEL)	172,728	2,802,164	1,383,710

The difference in fault counts between $C1$ and $C2$ mainly reflects the characteristics of the underlying technology libraries and their CAT characterization. Compared with $C1$, the $C2$ library contains more detailed and diverse cell implementations, which in turn lead to a richer set of defect

TABLE 7. Fault simulation runtime on C1.

Test program	Clock cycles	SAF	SCAT	TDF	DCAT
STL3	35,988	7 min	10 hrs	9 min	14 hrs
STL4	43,729	10 min	16 hrs	16 min	20 hrs
STL5	115,705	29 min	204 hrs	39 min	209 hrs
STL6	17,014	4 min	5 hrs	6 min	7 hrs
STL7	80,455	21 min	65 hrs	33 min	70 hrs

TABLE 8. Fault simulation runtime on C2.

Test program	Clock cycles	SAF	SCAT	TDF	DCAT
STL3	35,988	38 min	17 hrs	42 min	22 hrs
STL4	43,729	52 min	24 hrs	59 min	31 hrs
STL5	115,705	2 hrs	210 hrs	3 hrs	216 hrs
STL6	17,014	25 min	10 hrs	34 min	13 hrs
STL7	80,455	1 hrs	68 hrs	2 hrs	73 hrs

conditions during CTM generation. As a consequence, the number of characterized defects is significantly larger in $C2$ than in $C1$.

These observations motivate the coverage analysis discussed in the following subsection, where the effectiveness of the considered STLs is examined on the reconstructed defect-level fault lists.

C. STL FAULT COVERAGE AND SIMULATION COST

This subsection evaluates the effectiveness of the considered STLs in detecting CAT defects, using the defect-level fault lists obtained after post-processing. Coverage results are compared against SAF and TDF baselines to assess how well SAF-targeted STLs extend to defect-oriented models.

Tables 7 and 8 report the STL lengths and the corresponding fault simulation runtimes for C1 and C2. CAT-based fault simulation is substantially more expensive than SAF/TDF simulation, with an overhead ranging from about 75× for the shortest STL to more than 400× for the longest one. This overhead is mainly due to the larger fault lists derived from CTM matrices and to the need to simulate the expanded set of conditional faults. The post-processing step itself is lightweight in comparison, requiring from seconds to a few minutes in our Python-based implementation. Therefore, the dominant computational cost comes from CAT fault simulation rather than from defect collapsing. While this limits the use of the proposed flow in rapid iterative optimization loops, it remains practical for offline STL validation and qualification, where the goal is to characterize STL effectiveness and identify defect classes requiring additional test effort.

From a scalability perspective, the computational cost increases with both the number of faults and the length of the applied test programs. Since defect-oriented models significantly expand the fault space compared to SAF and TDF, the overall simulation effort grows accordingly. To manage this complexity, the proposed framework is intended for targeted validation phases, where a reduced set of critical modules or prioritized fault subsets (e.g., based on TT%) can be analyzed.

Tables 9 and 10 report the static fault coverage obtained on C1 and C2. The results show that STLs generated for SAFs achieve substantial coverage on static CAT defects, although CAT coverage remains generally lower than SAF coverage. For C1, all STLs detect more than 80% of static CAT defects, with STL5 reaching nearly 87%. For C2, static CAT coverage remains above 70% for all STLs, with STL5 reaching about 78%. This indicates that functional redundancy in SAF-targeted STLs naturally activates and propagates a significant portion of CAT defects, even though these defects were not explicitly targeted during STL generation.

The comparison between the SCond and SCAT columns in Tables 9 and 10 also confirms the importance of post-processing. Since multiple conditional faults may correspond to the same CAT defect, directly interpreting simulator-level conditional fault coverage can either overestimate or underestimate defect-level coverage. The maximum observed impact is a 2.5% reduction for STL5 on C1 and a 2.4% increase for STL3 on C2. These differences are moderate for static CAT faults, but they demonstrate that defect-level reconstruction is required for consistent coverage estimation.

Tables 11 and 12 report the corresponding results for TDF and dynamic CAT faults. The evaluated STLs achieve up to 80% TDF coverage on both libraries, while dynamic CAT coverage exceeds 74% on C1 and 67% on C2. Dynamic CAT coverage follows the same general trend as static CAT coverage, but it is lower overall because dynamic defects require more specific transition conditions and timing-sensitive activation patterns.

Post-processing has a stronger effect for dynamic CAT faults than for static CAT faults, as shown by the differences between the DCond and DCAT columns in Tables 11 and 12. The maximum observed coverage change reaches 11.2% on C1 and 7.2% on C2. This larger impact reflects the higher multiplicity of dynamic conditional faults per defect and further confirms the need to reconstruct coverage at the defect level.

Across both static and dynamic results, coverage is consistently lower on the industrial C2 library than on C1. This suggests that CAT detectability is strongly influenced by technology-specific cell implementations and defect characterization. The richer defect space of C2 introduces activation and observability conditions that are less frequently exercised by functional stimuli, increasing the fraction of residual undetected defects.

The results are consistent with prior SAF- and TDF-oriented SBST studies [17], [18], [19], [22], while extending the comparison to dynamic CAT defects and to an industrial technology library.

These coverage figures are further analyzed in Section IV-D using TT%-based grouping, which provides a more detailed view of which defect classes are intrinsically more difficult to test.

TABLE 9. Fault simulation results for static faults on C1.

Module	STL3 FC%			STL4 FC%			STL5 FC%			STL6 FC%			STL7 FC%		
	SAF	SCond	SCAT	SAF	SCond	SCAT	SAF	SCond	SCAT	SAF	SCond	SCAT	SAF	SCond	SCAT
CS regs	66.3	71.5	71.5	46.2	52.4	52.4	62.6	69.3	69.3	72.7	74.0	74.0	54.6	63.9	63.9
Debug	16.1	25.8	25.8	16.5	25.8	25.8	16.7	26.0	26.0	16.5	25.8	25.8	18.4	25.8	25.8
EX/ALU	78.6	79.2	79.9	82.3	86.0	84.7	91.4	92.7	92.2	84.0	84.3	84.5	87.5	90.2	89.0
EX/MUL	93.8	88.3	90.1	95.6	96.9	95.1	96.2	96.3	94.9	96.3	95.2	94.4	98.5	98.7	97.0
ID/Control	53.8	57.9	57.9	44.3	47.5	47.5	54.2	58.5	58.5	54.0	58.4	58.4	51.8	56.1	56.1
ID/Decoder	81.2	81.4	81.4	73.0	73.7	73.7	83.5	83.7	83.7	82.5	82.9	82.9	81.3	81.3	81.3
ID/HW loop regs	63.6	58.5	58.5	79.3	83.2	83.2	71.3	70.3	70.3	77.7	65.2	65.2	46.1	46.2	46.2
ID/INT control	20.7	25.6	25.6	10.9	15.8	15.8	20.7	25.6	25.6	20.7	25.6	25.6	10.9	15.8	15.8
ID/regs	91.3	97.0	97.0	84.8	91.2	91.2	84.2	90.4	90.4	82.7	95.0	95.0	82.2	87.0	87.0
IF/compr. dec.	40.8	46.7	46.7	49.6	54.5	54.5	75.8	76.8	76.8	68.7	71.6	71.6	41.1	46.8	46.8
IF/HW loop contr.	51.2	55.4	55.4	53.4	58.3	58.3	57.2	61.6	61.6	49.2	52.5	52.5	50.6	54.0	54.0
IF/Prefetch buf.	67.6	70.9	70.9	73.4	76.0	76.0	74.3	76.3	76.3	72.3	74.8	74.8	73.1	75.5	75.5
Load Store	77.4	85.7	82.1	84.6	92.0	89.6	85.0	92.6	90.3	77.8	84.7	80.9	70.6	82.4	78.2
CPU (TOP LEVEL)	80.7	82.9	83.1	80.1	86.6	84.8	82.9	88.2	86.7	81.4	86.7	85.7	80.2	86.7	84.5

TABLE 10. Fault simulation results for static faults on C2.

Module	STL3 FC%			STL4 FC%			STL5 FC%			STL6 FC%			STL7 FC%		
	SAF	SCond	SCAT	SAF	SCond	SCAT	SAF	SCond	SCAT	SAF	SCond	SCAT	SAF	SCond	SCAT
CS regs	68.3	41.2	45.1	44.6	42.1	45.7	65.4	47.7	52.6	70.9	45.1	49.4	53.6	41.7	45.4
Debug	15.4	17.7	24.3	15.9	18.1	24.8	16.0	25.9	33.6	15.9	22.2	29.3	17.8	17.9	24.6
EX/ALU	77.4	76.7	79.5	82.3	78.2	80.5	89.9	80.4	82.8	83.8	79.3	81.6	88.7	77.5	80.0
EX/MUL	93.2	72.8	71.4	92.8	74.2	72.1	95.1	76.9	75.8	92.4	75.5	73.9	96.5	73.5	71.8
ID/Control	50.7	38.1	46.9	38.6	38.9	47.7	51.2	45.0	54.3	51.1	41.8	50.7	48.7	38.5	47.4
ID/Decoder	82.1	69.3	75.3	72.4	70.7	76.3	84.3	73.8	79.0	82.9	72.2	77.7	82.6	69.8	75.7
ID/HW loop regs	71.7	77.0	82.3	82.0	78.6	83.5	69.8	80.6	85.5	83.1	79.7	84.7	55.3	77.9	83.0
ID/INT control	18.1	10.1	14.4	8.6	10.8	15.3	18.1	22.2	29.3	18.1	14.1	19.7	8.6	10.8	15.3
ID/regs	91.8	84.9	88.7	87.0	86.6	89.8	85.9	87.8	90.9	85.7	87.3	90.4	83.2	85.7	89.2
IF/compr. dec.	33.4	45.6	52.8	44.0	46.8	53.8	72.8	51.8	58.9	65.1	49.4	56.7	33.4	46.2	53.3
IF/HW loop contr.	57.8	47.7	58.5	59.7	48.6	59.4	62.9	53.5	63.8	55.6	51.2	61.7	55.9	48.0	58.9
IF/Prefetch buf.	68.4	68.8	73.9	73.8	70.2	74.8	74.1	73.3	78.0	72.6	71.7	76.3	75.7	69.5	74.4
Load Store	80.7	84.1	85.9	86.5	58.8	86.9	87.4	87.3	88.5	81.2	86.4	87.5	76.8	84.9	46.0
CPU (TOP LEVEL)	81.2	72.4	74.8	80.2	73.9	75.7	82.8	76.5	78.7	81.3	75.2	77.2	81.4	73.1	75.3

TABLE 11. Fault simulation results for dynamic faults on C1.

Module	STL3 FC%			STL4 FC%			STL5 FC%			STL6 FC%			STL7 FC%		
	TDF	DCond	DCAT	TDF	DCond	DCAT	TDF	DCond	DCAT	TDF	DCond	DCAT	TDF	DCond	DCAT
CS regs	55.9	51.3	62.1	43.2	39.5	48.1	49.0	43.9	56.1	60.1	52.8	64.5	42.1	33.8	48.2
Debug	0.0	1.3	3.3	0.0	1.3	3.3	0.0	1.3	3.4	0.0	1.3	3.3	0.0	1.7	4.4
EX/ALU	76.2	64.5	77.8	80.2	69.2	84.2	90.9	78.9	92.1	82.9	69.9	83.6	86.1	72.2	85.1
EX/MUL	92.5	73.0	81.8	93.7	77.5	86.0	94.7	79.1	87.3	95.1	77.7	86.9	97.9	79.1	85.9
ID/Control	37.6	32.1	46.9	22.6	18.3	28.5	39.7	32.6	47.2	39.7	32.9	47.3	36.0	29.1	43.8
ID/Decoder	80.5	65.4	80.4	71.2	60.0	76.2	83.1	70.7	86.6	81.6	69.4	85.7	79.9	65.5	79.8
ID/HW loop regs	57.4	47.3	60.3	77.4	66.3	78.7	64.7	51.3	60.4	74.2	60.9	75.0	32.6	26.5	34.8
ID/INT control	0.0	0.2	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ID/regs	95.3	86.1	92.4	86.5	78.7	88.4	84.8	76.8	85.8	87.5	80.4	90.1	86.1	77.2	82.9
IF/compr. dec.	23.0	20.2	34.7	36.3	30.8	49.6	72.0	61.9	79.5	63.0	53.5	71.3	23.1	20.3	34.9
IF/HW loop contr.	34.8	33.2	42.2	43.4	38.6	46.9	40.3	35.3	43.8	39.3	34.5	43.7	34.9	27.6	41.5
IF/Prefetch buf.	58.8	51.1	61.0	67.3	59.0	68.1	67.0	59.8	69.0	65.0	57.3	66.8	68.6	56.1	67.1
Load Store	75.4	64.4	75.7	84.1	71.6	82.3	84.4	73.5	84.4	76.6	65.3	76.7	65.3	55.4	69.3
CPU (TOP LEVEL)	78.3	66.6	76.5	77.9	67.0	78.2	80.0	68.9	79.7	79.6	68.3	79.5	77.9	64.9	74.6

D. DETAILED ANALYTICAL STUDY OF FAULT DETECTABILITY

To fully understand the reasons behind the high fault coverage, we performed an accurate analysis on the CAT fault lists, aimed at verifying if there are correlations between the number of lines in a cell's truth table that can be used to detect the fault and the actual fault coverage obtained by the STLs. We also performed the same analysis on SAF and TDF faults.

For each fault in the considered fault lists (SAF, TDF, static CAT, and dynamic CAT), we define a metric denoted as *table test percentage (TT%)*. TT% is the percentage of input combinations in the truth table (or, for CAT, in the

TABLE 12. Fault simulation results for dynamic faults on C2.

Module	STL3 FC%			STL4 FC%			STL5 FC%			STL6 FC%			STL7 FC%		
	TDF	DCond	DCAT	TDF	DCond	DCAT	TDF	DCond	DCAT	TDF	DCond	DCAT	TDF	DCond	DCAT
CS regs	57.4	34.6	37.8	42.1	35.1	38.1	51.2	40.3	47.9	57.3	36.2	40.1	41.7	34.1	37.6
Debug	0.0	0.8	1.7	0.0	0.8	1.8	0.0	1.8	1.7	0.0	2.4	1.3	0.0	0.8	1.7
EX/ALU	74.6	64.4	70.6	79.8	65.4	71.1	88.6	68.2	75.6	81.9	65.9	71.9	87.3	63.4	70.1
EX/MUL	91.9	70.8	76.2	91.1	71.9	76.8	93.4	74.2	80.2	90.7	72.3	77.4	95.3	69.8	75.6
ID/Control	33.6	13.3	17.7	19.0	13.5	17.8	34.7	20.5	30.1	34.2	14.8	20.3	30.6	13.0	17.5
ID/Decoder	80.2	52.6	59.0	69.6	53.4	59.3	83.2	57.2	66.0	82.1	54.1	60.6	80.8	51.9	58.6
ID/HW loop regs	64.4	60.0	67.8	82.0	60.9	68.4	61.8	64.0	73.3	80.8	61.4	69.3	41.9	58.9	67.1
ID/INT control	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ID/regs	94.2	73.0	79.1	88.7	74.1	79.6	86.9	76.2	83.2	88.5	74.5	80.3	85.6	71.9	78.6
IF/compr. dec.	18.6	26.4	33.9	33.6	26.8	34.1	70.9	32.8	44.3	61.1	27.9	36.2	18.6	26.0	33.7
IF/HW loop contr.	43.1	41.4	38.5	52.5	42.1	38.7	44.4	46.7	47.7	48.7	43.0	40.4	38.8	40.9	38.3
IF/Prefetch buf.	57.2	56.4	59.3	65.0	57.2	59.6	64.2	60.6	65.7	62.5	57.9	60.8	71.3	55.5	58.9
Load Store	79.7	66.6	74.2	85.8	67.7	74.7	87.0	70.2	78.8	80.4	68.2	75.6	72.3	65.6	73.7
CPU (TOP LEVEL)	78.3	61.6	67.4	77.9	62.6	67.9	80.0	65.6	72.8	79.0	63.1	68.8	79.2	60.7	66.9

TABLE 13. Example of static defect matrix for a 2-input AND gate.

A	*B*	*ZN*	*D*1	*D*2	*D*3	*D*4
0	0	0	1	1	1	1
0	1	0	0	1	1	1
1	0	0	0	0	1	1
1	1	1	0	0	0	1
TT%			25%	50%	75%	100%

TABLE 14. Example of dynamic defect matrix for a 2-input AND gate.

A	*B*	*ZN*	*D*5	*D*6	*D*7	*D*8	...
0	R	0	1	1	1	1	...
R	0	0	0	1	1	1	...
0	F	0	0	0	1	1	...
R	1	R	0	0	0	1	...
F	0	0	0	0	0	0	...
1	R	R	0	0	0	0	...
F	1	F	0	0	0	0	...
1	F	F	0	0	0	0	...
TT%			12.5%	25%	37.5%	50%	...

corresponding defect matrix) of the cell in which the fault or defect is detectable at the cell outputs. In the case of CAT, we first derive TT% at the defect level from the CTM matrices and then assign the same TT% to all conditional faults that implement that defect.

For SAF and TDF fault lists, no explicit defect–fault matrices are available. In these cases, TT% is computed directly from the behavior of each standard cell. For a given stuck-at fault (SAF) on an input or output line, we enumerate all input combinations of the cell and count the number of combinations that detect that stuck-at fault, i.e., those for which the faulty output differs from the fault-free output; TT% is then the ratio between this count and the total number of input combinations of the cell. For a given transition delay fault (TDF), we similarly consider all applicable input–output transitions (pairs of consecutive input vectors that sensitize and propagate a rising or falling transition at the fault location) and count how many such pairs detect the fault at the cell outputs; TT% is defined as the ratio between this count and the total number of applicable input–transition combinations for the cell. This procedure is fully consistent with the CAT case, where the same type of ratio is derived from the static or dynamic defect matrices instead of from an analytical evaluation of the cell truth table or its timing behavior.

For example, Tables 13 and 14 show the TT% calculated according to the static and dynamic detection tables of a 2-input AND gate.

After computing the TT% for each fault, faults are grouped globally according to identical TT% values, independently of their originating cell or defect class. For each group, we define its size in the original fault list as the total number of faults sharing that TT% value, and its size in the detected fault list as the subset detected by the considered STL. Based on these quantities, we derive two metrics: (i) the fault coverage of the group and (ii) the contribution of the group to the overall fault coverage. The results reported in Tables 15 to 18 are organized according to these TT%-based groups.

For SAFs and static CAT faults, TT% values are discrete, as they depend on the finite number of entries in a cell's truth table or defect matrix (e.g., 1/4, 1/8, 3/8). Consequently, many faults share the same TT% value and naturally form well-populated groups. In contrast, dynamic CAT faults produce a much larger set of distinct TT% values due to the increased number of input–transition combinations. To maintain a compact representation, dynamic TT% values are aggregated into contiguous 4% intervals, and results are reported as averages over the groups within each interval. This aggregation is also applied to static CAT faults in *C*2, where the number of distinct TT% values is similarly large.

For each TT% group (or interval), we report its fault coverage (*Fault coverage / grouped faults*) and its contribution to the overall coverage (*Fault coverage / all faults*), as shown in Tables 15 to 18. Cumulative (*cumsum*) values are also provided for both group size and coverage, allowing a progressive view of how coverage evolves as increasingly larger portions of the fault space are considered. The final

TABLE 15. Fault simulation results on groups of static faults for C1.

Grouped faults info			Fault coverage / grouped faults (%)					Fault coverage / all faults (%)					Fault coverage / all faults (% cumsum)				
TT%	Size%	Size% cumsum	STL3	STL4	STL5	STL6	STL7	STL3	STL4	STL5	STL6	STL7	STL3	STL4	STL5	STL6	STL7
Stuck-at																	
6.25	6.48	6.48	81.42	68.38	75.44	70.02	72.16	5.28	4.82	5.31	4.93	5.08	5.28	4.82	5.31	4.93	5.08
9.38	1.81	8.30	58.48	58.91	66.52	59.84	61.82	1.06	1.07	1.21	1.09	1.12	6.34	5.89	6.52	6.02	6.20
12.50	7.04	15.34	66.55	79.92	85.31	83.52	77.92	4.69	5.18	5.53	5.41	5.05	11.03	11.07	12.05	11.43	11.26
14.06	3.73	19.07	81.84	84.69	90.28	89.53	86.82	3.05	3.14	3.35	3.32	3.22	14.08	14.21	15.40	14.75	14.48
18.75	34.99	54.06	78.72	76.09	78.07	75.40	75.61	27.54	26.63	27.32	26.39	26.46	41.62	40.84	42.72	41.14	40.93
25.00	10.74	64.80	70.87	69.99	75.09	75.69	71.62	7.61	7.52	8.07	8.13	7.69	49.24	48.35	50.78	49.27	48.63
28.13	0.68	65.48	75.36	79.61	81.16	80.12	75.36	0.51	0.54	0.55	0.55	0.51	49.75	48.90	51.33	49.82	49.14
37.50	3.71	69.19	85.33	88.34	87.57	84.33	82.94	3.16	3.29	3.26	3.14	3.09	52.91	52.19	54.60	52.96	52.23
42.19	0.31	69.50	93.31	97.39	94.90	96.71	88.32	0.29	0.30	0.29	0.30	0.27	53.20	52.49	54.89	53.26	52.51
43.75	4.01	73.51	88.05	85.14	85.28	90.64	80.99	3.53	3.41	3.42	3.63	3.25	56.73	55.90	58.31	56.89	55.75
50.00	17.50	91.01	90.34	91.94	92.61	92.56	92.43	15.81	16.09	16.21	16.20	16.17	72.54	71.99	74.52	73.09	71.93
56.25	4.01	95.02	95.91	94.69	96.42	96.52	96.15	3.84	3.80	3.87	3.87	3.85	76.39	75.79	78.38	76.96	75.78
57.81	0.31	95.33	93.76	97.17	97.17	97.85	96.03	0.29	0.30	0.30	0.30	0.30	76.68	76.09	78.69	77.26	76.08
62.50	1.24	96.57	84.05	88.77	93.85	91.34	87.52	1.04	1.10	1.16	1.13	1.08	77.72	77.19	79.85	78.39	77.16
71.88	0.23	96.79	75.16	82.92	84.47	80.43	78.26	0.17	0.19	0.19	0.18	0.18	77.89	77.38	80.04	78.58	77.34
75.00	1.86	98.65	88.06	85.52	89.04	89.08	89.50	1.64	1.59	1.66	1.66	1.67	79.52	78.97	81.69	80.23	79.00
81.25	0.58	99.24	86.93	87.65	91.51	89.94	90.90	0.51	0.51	0.53	0.53	0.53	80.03	79.48	82.23	80.76	79.54
87.50	0.30	99.54	81.22	82.86	85.45	84.04	82.98	0.24	0.25	0.26	0.25	0.25	80.28	79.73	82.49	81.01	79.78
93.75	0.46	100.00	91.67	89.22	92.74	92.28	84.33	0.42	0.41	0.43	0.43	0.39	80.70	80.14	82.91	81.44	80.17
Static Cell-Aware Test																	
1.56	<0.01	<0.01	83.33	100.00	100.00	83.33	100.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3.13	<0.01	<0.01	100.00	100.00	0.00	100.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
4.69	<0.01	<0.01	100.00	100.00	100.00	100.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
6.25	3.81	3.81	81.42	82.77	89.19	83.68	82.77	3.10	3.15	3.40	3.19	3.15	3.10	3.15	3.40	3.19	3.15
9.38	1.20	5.01	58.73	59.48	67.16	60.61	63.87	0.70	0.71	0.80	0.73	0.77	3.81	3.87	4.20	3.91	3.92
12.50	10.14	15.14	68.33	77.64	79.24	73.48	77.71	6.93	7.87	8.03	7.45	7.88	10.73	11.74	12.24	11.36	11.80
14.06	2.25	17.40	83.00	90.08	89.66	84.52	84.78	1.87	2.03	2.02	1.91	1.91	12.60	13.77	14.26	13.27	13.71
18.75	22.08	39.48	81.25	78.68	80.19	82.89	75.90	17.94	17.37	17.71	18.30	16.76	30.55	31.14	31.96	31.57	30.47
21.88	<0.01	39.48	100.00	100.00	100.00	100.00	100.00	0.00	0.00	0.00	0.00	0.00	30.55	31.14	31.96	31.57	30.47
25.00	17.20	56.68	79.09	83.75	85.62	84.36	85.09	13.60	14.40	14.73	14.51	14.63	44.15	45.54	46.69	46.08	45.10
28.13	0.70	57.38	79.90	84.13	86.07	86.91	77.75	0.56	0.59	0.60	0.61	0.54	44.70	46.13	47.29	46.69	45.64
29.69	<0.01	57.38	100.00	100.00	100.00	100.00	100.00	0.00	0.00	0.00	0.00	0.00	44.70	46.13	47.29	46.69	45.64
32.81	0.06	57.44	70.59	89.62	98.27	61.94	58.48	0.04	0.05	0.06	0.04	0.04	44.75	46.18	47.35	46.72	45.68
34.38	<0.01	57.44	100.00	100.00	100.00	100.00	100.00	0.00	0.00	0.00	0.00	0.00	44.75	46.19	47.35	46.72	45.68
37.50	6.43	63.87	86.03	89.24	93.31	90.23	91.91	5.53	5.74	6.00	5.80	5.91	50.28	51.93	53.35	52.53	51.59
40.63	0.17	64.04	73.76	75.74	79.21	72.77	79.70	0.12	0.13	0.13	0.12	0.13	50.41	52.05	53.48	52.65	51.72
42.19	1.69	65.73	97.66	98.69	98.18	97.92	97.92	1.65	1.67	1.66	1.66	1.66	52.06	53.72	55.14	54.31	53.38
43.75	0.80	66.53	86.21	85.20	88.09	84.49	83.55	0.69	0.68	0.70	0.67	0.67	52.75	54.40	55.85	54.98	54.05
50.00	15.00	81.53	86.88	87.20	88.32	87.61	87.53	13.03	13.08	13.25	13.14	13.13	65.78	67.48	69.10	68.12	67.18
56.25	11.64	93.17	97.56	96.45	98.16	97.97	97.29	11.36	11.23	11.43	11.41	11.33	77.14	78.71	80.53	79.53	78.51
57.81	0.10	93.27	75.71	91.90	98.38	70.85	61.13	0.08	0.09	0.10	0.07	0.06	77.21	78.81	80.63	79.60	78.57
62.50	1.23	94.50	83.60	90.86	94.52	89.90	87.63	1.03	1.12	1.16	1.11	1.08	78.24	79.92	81.79	80.71	79.65
68.75	0.10	94.60	93.83	91.49	90.21	95.53	82.13	0.09	0.09	0.09	0.09	0.08	78.33	80.01	81.88	80.80	79.73
71.88	0.47	95.07	77.91	84.29	84.43	80.37	81.35	0.36	0.39	0.39	0.37	0.38	78.70	80.41	82.27	81.18	80.11
75.00	2.00	97.07	89.95	87.60	90.78	91.50	90.33	1.80	1.75	1.82	1.83	1.81	80.50	82.16	84.09	83.01	81.91
81.25	0.99	98.06	84.19	86.23	89.20	88.64	90.72	0.83	0.85	0.88	0.88	0.90	81.33	83.01	84.97	83.88	82.81
87.50	0.78	98.84	81.69	85.03	86.12	86.34	82.99	0.64	0.66	0.67	0.67	0.65	81.97	83.68	85.64	84.56	83.46
93.75	1.16	100.00	94.61	93.47	94.41	96.19	86.34	1.10	1.09	1.10	1.12	1.00	83.07	84.76	86.74	85.68	84.46

cumulative values correspond to the overall fault coverage achieved by the STL.

For SAFs, the tables show large groups with low $TT\%$ and correspondingly low fault coverage in both $C1$ and $C2$. For example, $TT\% = 25$ corresponds to faults requiring a specific input condition (e.g., inputs of AND2, OR2, NAND2, etc., and some output faults), achieving about 76% coverage in $C1$ and 80% in $C2$. Lower $TT\%$ groups (e.g., 12.5 or 6.25) exhibit reduced coverage, as expected. In $C1$, the most critical group is $TT\% = 18.75$, representing over one-third of the fault list with coverage between 75% and 79%, while in $C2$, the most critical group is $TT\% = 50$, covering about 30% of the faults with coverage close to 90%.

The static CAT sub-table shows a more uneven distribution. In $C1$, low-$TT\%$ groups are smaller than in SAF, whereas in $C2$, groups with $TT\% < 10\%$ contain more than 50% of the static CAT faults. In general, CAT groups exhibit higher variability in size than SAF groups, as reflected by the slower convergence of the cumulative size toward 100%.

For TDFs, the maximum $TT\%$ is 50%, corresponding to faults on the cell outputs. Most faults are concentrated in a small number of groups, notably $TT\% = 12.5$, 25.0, and

TABLE 16. Fault simulation results on groups of static faults for C2.

Grouped faults info			Fault coverage / grouped faults (%)					Fault coverage / all faults (%)					Fault coverage / all faults (% cumsum)				
TT%	Size%	Size% cumsum	STL3	STL4	STL5	STL6	STL7	STL3	STL4	STL5	STL6	STL7	STL3	STL4	STL5	STL6	STL7
Stuck-at																	
3.12	0.06	0.06	68.75	70.67	82.21	74.52	61.06	0.04	0.04	0.05	0.04	0.04	0.04	0.04	0.05	0.04	0.04
6.25	3.58	3.64	78.75	76.22	82.66	81.59	73.02	2.82	2.73	2.96	2.92	2.61	2.86	2.77	3.01	2.96	2.65
9.37	4.66	8.30	67.23	66.69	76.24	77.62	78.14	3.13	3.11	3.55	3.62	3.64	5.99	5.88	6.56	6.58	6.29
10.93	0.01	8.30	58.33	41.67	50.00	58.33	58.33	0.00	0.00	0.00	0.00	0.00	5.99	5.88	6.56	6.58	6.29
12.50	6.45	14.75	68.58	64.57	71.66	67.39	71.96	4.42	4.16	4.62	4.34	4.64	10.42	10.04	11.18	10.93	10.93
14.06	6.06	20.81	69.62	73.90	73.95	73.25	68.64	4.22	4.48	4.48	4.44	4.16	14.63	14.52	15.66	15.37	15.09
18.75	15.11	35.92	79.42	78.02	78.31	81.24	74.62	12.00	11.79	11.84	12.28	11.28	26.64	26.31	27.50	27.64	26.37
21.87	0.06	35.98	74.07	54.63	76.85	78.70	66.67	0.05	0.03	0.05	0.05	0.04	26.68	26.34	27.55	27.69	26.41
25.00	21.33	57.31	78.34	77.20	80.77	73.49	80.92	16.71	16.47	17.23	15.67	17.26	43.39	42.81	44.77	43.37	43.67
28.12	1.67	58.99	90.34	87.40	91.10	92.77	92.38	1.51	1.46	1.52	1.55	1.55	44.91	44.27	46.30	44.92	45.22
31.25	0.19	59.18	67.12	58.94	79.24	78.33	72.58	0.13	0.11	0.15	0.15	0.14	45.03	44.39	46.45	45.07	45.35
34.37	0.02	59.19	62.96	66.67	74.07	77.78	59.26	0.01	0.01	0.01	0.01	0.01	45.04	44.40	46.46	45.08	45.36
37.50	3.56	62.75	85.85	82.12	88.22	86.86	88.78	3.05	2.92	3.14	3.09	3.16	48.10	47.32	49.60	48.17	48.52
42.18	0.50	63.26	96.62	97.31	98.11	98.45	96.62	0.49	0.49	0.50	0.50	0.49	48.58	47.81	50.09	48.67	49.01
43.75	1.93	65.18	88.32	84.30	84.06	89.43	75.83	1.70	1.63	1.62	1.72	1.46	50.29	49.43	51.71	50.39	50.47
50.00	27.27	92.45	88.94	89.12	89.55	88.78	88.85	24.25	24.30	24.42	24.21	24.23	74.54	73.74	76.13	74.60	74.70
56.25	1.87	94.33	95.99	94.63	96.61	96.21	96.95	1.80	1.77	1.81	1.80	1.81	76.34	75.51	77.94	76.40	76.52
57.81	0.50	94.83	77.47	85.26	83.54	85.49	74.71	0.39	0.43	0.42	0.43	0.38	76.73	75.94	78.36	76.83	76.89
62.50	1.19	96.02	76.78	70.73	79.81	78.93	83.23	0.91	0.84	0.95	0.94	0.99	77.64	76.78	79.31	77.77	77.88
65.62	0.02	96.03	88.89	81.48	81.48	88.89	81.48	0.01	0.01	0.01	0.01	0.01	77.65	76.79	79.32	77.78	77.89
68.75	0.06	96.10	88.18	81.82	92.73	88.18	88.18	0.06	0.05	0.06	0.06	0.06	77.71	76.84	79.38	77.84	77.95
71.87	0.56	96.65	90.86	85.98	82.76	95.02	88.16	0.51	0.48	0.46	0.53	0.49	78.22	77.32	79.84	78.37	78.44
75.00	2.71	99.37	88.31	86.83	89.33	87.58	89.88	2.40	2.36	2.42	2.38	2.44	80.61	79.68	82.27	80.75	80.88
81.25	0.05	99.41	91.57	89.16	93.98	79.52	90.36	0.04	0.04	0.05	0.05	0.05	80.66	79.72	82.31	80.79	80.92
87.50	0.24	99.66	81.82	77.27	82.54	82.30	77.75	0.20	0.19	0.20	0.20	0.19	80.86	79.91	82.51	80.98	81.11
90.06	0.02	99.67	92.31	92.31	92.31	92.31	92.31	0.01	0.01	0.01	0.01	0.01	80.87	79.92	82.53	81.00	81.12
93.75	0.33	100.00	87.06	86.88	92.17	89.52	79.67	0.29	0.29	0.30	0.29	0.26	81.16	80.21	82.83	81.29	81.39
Static Cell-Aware Test																	
1-5	8.77	8.77	55.87	56.93	61.37	59.16	56.39	4.90	4.99	5.38	5.19	4.94	4.90	4.99	5.38	5.19	4.94
5–9	6.52	15.29	72.76	73.90	76.69	75.47	73.27	4.75	4.82	5.00	4.92	4.78	9.65	9.81	10.38	10.11	9.73
9–13	16.98	32.27	66.23	67.26	70.86	68.97	66.72	11.24	11.42	12.03	11.71	11.33	20.89	21.23	22.42	21.82	21.05
13–17	2.78	35.06	81.37	82.33	84.52	83.56	81.92	2.27	2.29	2.35	2.33	2.28	23.16	23.53	24.77	24.15	23.33
17–21	15.02	50.08	67.71	68.51	73.01	70.62	68.01	10.17	10.29	10.97	10.61	10.22	33.33	33.82	35.74	34.76	33.55
21–25	3.76	53.84	92.27	93.20	93.96	93.54	92.59	3.47	3.51	3.53	3.52	3.48	36.80	37.33	39.27	38.28	37.03
25–29	8.56	62.40	78.42	79.33	81.72	80.49	78.90	6.71	6.79	7.00	6.89	6.75	43.51	44.12	46.27	45.17	43.79
29–33	7.34	69.74	79.70	81.10	83.31	82.15	80.63	5.85	5.95	6.11	6.03	5.92	49.36	50.07	52.38	51.20	49.71
33–37	1.27	71.01	94.36	95.05	95.71	95.42	94.57	1.20	1.21	1.22	1.21	1.20	50.56	51.28	53.60	52.41	50.91
37–41	8.93	79.94	80.63	81.08	84.00	82.59	80.91	7.20	7.24	7.50	7.37	7.22	57.76	58.51	61.10	59.78	58.13
41–45	0.23	80.17	75.72	76.76	78.83	78.05	76.45	0.17	0.18	0.18	0.18	0.18	57.93	58.69	61.28	59.96	58.31
45–49	0.68	80.85	96.02	96.48	96.97	97.02	96.37	0.65	0.66	0.66	0.66	0.66	58.59	59.35	61.94	60.62	58.96
49–53	1.41	82.26	84.68	85.63	87.18	86.70	85.34	1.19	1.20	1.23	1.22	1.20	59.78	60.55	63.16	61.84	60.16
53–57	6.30	88.55	87.13	87.78	89.81	88.73	87.36	5.49	5.53	5.66	5.59	5.50	65.27	66.08	68.82	67.43	65.67
57–61	0.26	88.81	92.30	92.40	94.47	92.86	92.30	0.24	0.24	0.24	0.24	0.24	65.50	66.32	69.06	67.67	65.90
61–65	2.17	90.98	85.27	86.66	88.25	87.69	86.17	1.85	1.88	1.92	1.90	1.87	67.35	68.20	70.98	69.57	67.77
65–69	0.19	91.18	95.30	96.28	97.07	97.01	96.04	0.19	0.19	0.19	0.19	0.19	67.54	68.39	71.17	69.76	67.96
69–73	0.27	91.45	92.63	94.47	95.43	94.64	94.47	0.25	0.26	0.26	0.26	0.26	67.79	68.64	71.43	70.02	68.22
73–77	0.56	92.01	28.15	28.32	41.82	35.51	28.28	0.16	0.16	0.24	0.20	0.16	67.95	68.80	71.66	70.22	68.38
77–81	1.53	93.60	81.77	82.79	85.57	84.36	82.42	1.25	1.27	1.31	1.29	1.26	69.25	70.12	73.03	71.56	69.69
81–85	0.02	93.62	100.00	100.00	100.00	100.00	100.00	0.02	0.02	0.02	0.02	0.02	69.27	70.14	73.05	71.58	69.71
85–89	1.85	95.48	91.25	91.86	93.08	92.38	91.54	1.69	1.70	1.73	1.71	1.70	70.97	71.85	74.78	73.30	71.41
89–93	3.18	98.66	83.89	85.18	86.56	86.26	84.62	2.67	2.71	2.75	2.75	2.69	73.64	74.56	77.53	76.04	74.10
93–97	0.06	98.72	97.94	97.94	99.06	97.94	97.94	0.06	0.06	0.06	0.06	0.06	73.70	74.62	77.59	76.11	74.16
97–100	1.28	100.00	90.35	90.62	92.19	91.93	90.38	1.15	1.16	1.18	1.17	1.15	74.85	75.78	78.77	77.28	75.32

50.0. For example, $TT\% = 12.5$ (e.g., inputs of AND4, OR4, NAND4, etc.) represents about 36% of the fault list in $C1$ and about 17% in $C2$, with approximately 80% coverage in both cases.

The dynamic CAT sub-table shows a wider $TT\%$ distribution due to the combination of transition and static fault types. The majority of dynamic CAT faults fall below $TT\% = 50\%$ (about 78% in $C1$ and 87% in $C2$). As in previous cases, lower-$TT\%$ groups exhibit lower coverage while accounting for most of the fault population.

The analysis across both $C1$ and $C2$ confirms that defects with low $TT\%$ are consistently harder to test. Although these low-$TT\%$ groups dominate the residual fault set, their relative impact on total coverage

TABLE 17. Fault simulation results on groups of dynamic faults for C1.

Grouped faults info			Fault coverage / grouped faults (%)					Fault coverage / all faults (%)					Fault coverage / all faults (% cumsum)				
TT%	Size%	Size% cumsum	STL3	STL4	STL5	STL6	STL7	STL3	STL4	STL5	STL6	STL7	STL3	STL4	STL5	STL6	STL7
Transition Delay																	
6.25	2.35	2.35	62.98	68.28	80.33	71.11	71.75	1.48	1.60	1.89	1.67	1.69	1.48	1.60	1.89	1.67	1.69
7.14	1.82	4.17	55.20	56.13	62.69	56.91	58.19	1.01	1.02	1.14	1.04	1.06	2.49	2.63	3.03	2.71	2.75
8.33	3.74	7.92	80.27	86.81	86.09	82.63	82.56	3.01	3.25	3.22	3.09	3.09	5.49	5.88	6.25	5.80	5.84
10.00	4.97	12.88	62.96	67.85	74.15	67.98	71.50	3.13	3.37	3.68	3.38	3.55	8.62	9.25	9.94	9.18	9.39
12.50	35.92	48.81	79.67	75.91	76.58	76.16	75.02	28.62	27.27	27.51	27.36	26.95	37.24	36.52	37.45	36.54	36.34
16.66	7.06	55.86	79.14	82.02	81.58	83.22	78.93	5.58	5.79	5.76	5.87	5.57	42.82	42.31	43.20	42.41	41.91
18.75	2.35	58.21	74.82	79.58	81.66	82.29	78.86	1.76	1.87	1.92	1.93	1.85	44.58	44.18	45.12	44.34	43.76
21.42	0.46	58.67	65.37	69.72	69.41	70.03	68.63	0.30	0.32	0.32	0.32	0.31	44.88	44.49	45.44	44.66	44.07
25.00	11.30	69.96	74.90	75.90	80.51	80.01	78.19	8.46	8.57	9.09	9.04	8.83	53.34	53.07	54.53	53.70	52.91
30.00	2.48	72.45	76.61	75.67	83.65	83.36	78.77	1.90	1.88	2.08	2.07	1.96	55.24	54.95	56.61	55.77	54.86
50.00	27.55	100.00	83.57	83.32	85.04	86.73	83.87	23.03	22.96	23.43	23.90	23.11	78.27	77.90	80.04	79.67	77.97
Dynamic Cell-Aware Test																	
1–5	3.23	3.23	71.09	73.29	79.43	77.39	69.45	2.30	2.37	2.57	2.50	2.24	2.30	2.37	2.57	2.50	2.24
5–9	5.54	8.77	79.75	82.03	85.59	83.59	80.26	4.42	4.54	4.74	4.63	4.45	6.71	6.91	7.31	7.13	6.69
9–13	8.30	17.07	72.07	75.16	77.71	74.82	74.76	5.98	6.24	6.45	6.21	6.20	12.70	13.15	13.76	13.34	12.89
13–17	9.11	26.18	68.83	72.00	75.64	70.84	70.15	6.27	6.56	6.89	6.46	6.39	18.97	19.71	20.65	19.80	19.29
17–21	11.95	38.14	78.62	77.99	78.75	79.52	73.19	9.40	9.32	9.41	9.51	8.75	28.37	29.03	30.06	29.30	28.04
21–25	11.50	49.63	74.66	77.66	75.52	79.53	68.98	8.58	8.93	8.68	9.14	7.93	36.95	37.96	38.75	38.45	35.97
25–29	5.11	54.74	81.11	78.80	79.82	78.17	73.81	4.14	4.02	4.08	3.99	3.77	41.09	41.99	42.82	42.44	39.74
29–33	8.45	63.19	69.05	71.18	75.83	74.08	69.82	5.84	6.02	6.41	6.26	5.90	46.93	48.00	49.23	48.70	45.64
33–37	1.70	64.89	78.05	83.89	76.34	85.67	73.56	1.32	1.42	1.29	1.45	1.25	48.25	49.42	50.52	50.15	46.88
37–41	6.97	71.86	83.16	80.03	81.23	84.27	77.39	5.80	5.58	5.67	5.88	5.40	54.05	55.01	56.19	56.03	52.28
41–45	6.29	78.15	83.02	85.81	88.64	88.22	82.96	5.22	5.40	5.58	5.55	5.22	59.27	60.40	61.77	61.58	57.50
45–49	0.07	78.22	83.17	90.84	83.66	89.11	88.61	0.06	0.06	0.06	0.06	0.06	59.33	60.47	61.82	61.64	57.56
49–53	1.54	79.77	92.80	91.84	91.63	94.49	88.60	1.43	1.42	1.41	1.46	1.38	60.76	61.88	63.24	63.10	58.93
53–57	0.43	80.20	88.58	91.74	93.37	93.45	92.52	0.39	0.40	0.41	0.41	0.40	61.15	62.28	63.64	63.50	59.33
57–61	9.33	89.53	62.15	67.78	66.88	67.76	61.83	5.80	6.32	6.24	6.32	5.77	66.95	68.61	69.88	69.83	65.10
61–65	6.01	95.54	96.58	95.47	97.08	97.28	96.40	5.80	5.73	5.83	5.84	5.79	72.75	74.34	75.71	75.67	70.89
65–69	0.44	95.97	35.71	46.24	55.93	37.72	35.40	0.16	0.16	0.24	0.17	0.15	72.90	74.54	75.96	75.83	71.05
69–73	1.54	97.51	88.91	90.99	93.73	93.59	89.69	1.37	1.42	1.44	1.44	1.38	74.27	75.94	77.40	77.27	72.42
73–77	0.11	97.62	75.50	83.46	83.61	80.40	80.09	0.08	0.09	0.09	0.09	0.09	74.35	76.03	77.49	77.36	72.51
77–81	1.17	98.79	85.55	84.31	86.40	87.32	87.11	1.00	0.99	1.01	1.02	1.02	75.35	77.02	78.50	78.38	73.53
81–85	0.46	99.25	88.66	90.14	92.27	91.31	91.76	0.41	0.41	0.42	0.42	0.42	75.76	77.43	78.93	78.80	73.95
85–89	<0.01	99.25	100.00	89.29	75.00	75.00	100.00	0.00	0.00	0.00	0.00	0.00	75.77	77.44	78.93	78.93	73.96
89–93	0.29	99.55	91.93	97.12	93.89	96.86	97.03	0.27	0.28	0.29	0.28	0.28	76.04	77.74	79.13	79.09	74.24
93–97	0.42	99.97	92.85	95.84	98.01	95.99	91.62	0.38	0.39	0.39	0.41	0.39	76.44	78.14	79.29	79.49	74.63
97–100	0.03	100.00	93.68	95.40	97.89	87.36	93.68	0.02	0.02	0.02	0.03	0.02	76.46	78.15	79.66	79.52	74.64

is less pronounced for CAT faults than for SAFs and TDFs.

Beyond providing a compact detectability ranking, the TT% metric captures how activation conditions are distributed across the fault space. Medium- and high-*TT%* groups generally exhibit higher grouped coverage, while low-*TT%* groups contain the most difficult faults to detect. This highlights that improving coverage beyond current levels requires targeting increasingly specific cell-level activation conditions.

A further observation is that the lowest *TT%* groups are largely associated with multi-output and multi-fault CAT classes, indicating that TT% reflects structural test complexity. These defects require more constrained activation and propagation conditions, making them inherently more difficult to detect. In addition, TT% distributions differ across fault models and technology libraries, showing a more heterogeneous profile for CAT compared with SAF and TDF.

E. PRACTICAL IMPLICATIONS FOR STL DESIGN

The analysis presented in this work enables translating defect-level observations into concrete guidelines for improving Software Test Libraries (STLs). In particular, the combined use of CAT fault modeling, post-processing, and the TT% metric provides a structured way to identify coverage limitations and guide targeted enhancements.

- **Prioritize low-TT% defects:** The results show that defects characterized by low TT% values consistently exhibit lower detection rates across all evaluated STLs. Although these defects represent a limited fraction of the overall fault space, they dominate the residual undetected set. Therefore, STL optimization should focus on increasing the activation probability of these low-detectability defects rather than extending test length indiscriminately. The TT% metric provides a systematic way to identify and prioritize these critical defects during STL refinement. This

TABLE 18. Fault simulation results on groups of dynamic faults for C2.

Grouped faults info			Fault coverage / grouped faults (%)					Fault coverage / all faults (%)					Fault coverage / all faults (% cumsum)				
TT%	Size%	Size% cumsum	STL3	STL4	STL5	STL6	STL7	STL3	STL4	STL5	STL6	STL7	STL3	STL4	STL5	STL6	STL7
Transition Delay																	
4.54	0.06	0.06	64.90	64.90	78.37	72.60	49.52	0.04	0.04	0.05	0.04	0.03	0.04	0.04	0.05	0.04	0.03
5.00	0.42	0.49	45.84	39.71	57.02	55.11	52.11	0.19	0.17	0.24	0.23	0.22	0.23	0.21	0.29	0.28	0.25
6.25	0.20	0.69	51.58	70.83	75.14	60.34	63.65	0.10	0.14	0.15	0.12	0.13	0.34	0.35	0.44	0.40	0.38
6.52	0.09	0.78	56.79	47.53	56.17	59.88	36.42	0.05	0.04	0.05	0.06	0.03	0.39	0.40	0.49	0.46	0.41
7.14	4.46	5.24	62.88	64.06	74.86	74.49	76.62	2.80	2.86	3.34	3.32	3.42	3.20	3.25	3.83	3.78	3.83
8.33	6.07	11.31	66.15	71.64	71.32	70.48	64.81	4.01	4.34	4.33	4.27	3.93	7.21	7.60	8.16	8.05	7.76
10.00	4.74	16.05	66.00	63.18	69.89	63.29	71.63	3.13	3.00	3.31	3.00	3.40	10.34	10.59	11.47	11.05	11.16
12.50	17.38	33.43	79.84	78.33	77.86	81.50	74.24	13.88	13.62	13.53	14.17	12.90	24.22	24.21	25.01	25.22	24.06
13.63	0.09	33.52	50.64	50.64	67.31	66.03	47.44	0.05	0.05	0.06	0.06	0.04	24.26	24.26	25.07	25.28	24.11
15.00	0.13	33.65	56.59	46.14	76.14	74.32	66.14	0.07	0.06	0.10	0.09	0.08	24.33	24.31	25.16	25.38	24.19
15.21	0.06	33.71	73.15	53.70	73.15	78.70	62.96	0.05	0.03	0.05	0.05	0.04	24.38	24.35	25.21	25.42	24.23
16.66	12.35	46.06	84.79	83.51	83.96	74.99	83.09	10.47	10.32	10.37	9.26	10.26	34.85	34.66	35.58	34.69	34.49
18.75	0.19	46.26	75.90	77.11	84.04	83.73	65.36	0.15	0.15	0.16	0.16	0.13	35.00	34.81	35.74	34.85	34.62
21.42	1.12	47.37	80.71	79.98	81.39	84.29	83.88	0.90	0.89	0.91	0.94	0.94	35.90	35.70	36.65	35.79	35.55
25.00	16.68	64.05	75.22	76.44	80.43	77.68	81.12	12.55	12.75	13.42	12.96	13.53	48.45	48.45	50.07	48.75	49.09
30.00	2.37	66.42	73.84	67.54	77.58	74.13	82.51	1.75	1.60	1.84	1.76	1.96	50.20	50.06	51.91	50.51	51.04
35.00	0.06	66.48	50.00	53.06	55.10	55.10	55.10	0.03	0.03	0.03	0.03	0.03	50.23	50.09	51.94	50.54	51.07
50.00	33.52	100.00	83.89	82.87	83.72	84.58	83.90	28.12	27.78	28.06	28.35	28.12	78.35	77.86	80.00	78.89	79.20
Dynamic Cell-Aware Test																	
1.3–5.3	6.710	6.71	47.71	48.21	55.62	49.68	47.17	3.20	3.23	3.73	3.33	3.17	3.20	3.23	3.73	3.33	3.17
5.3–9.3	13.942	20.65	58.33	58.52	65.73	59.90	58.10	8.13	8.16	9.16	8.35	8.10	11.33	11.39	12.90	11.68	11.27
9.3–13.3	17.289	37.94	59.64	60.12	66.12	61.32	59.20	10.31	10.39	11.43	10.60	10.24	21.64	21.79	24.33	22.29	21.50
13.3–17.3	17.576	55.52	70.09	70.48	75.05	71.35	69.68	12.32	12.39	13.19	12.54	12.25	33.96	34.18	37.52	34.83	33.75
17.3–21.3	12.577	68.09	83.26	83.68	86.06	84.11	82.78	10.47	10.52	10.82	10.58	10.41	44.44	44.70	48.34	45.41	44.16
21.3–25.3	5.767	73.86	67.04	67.51	72.81	68.49	66.54	3.87	3.89	4.20	3.95	3.84	48.30	48.59	52.54	49.36	48.00
25.3–29.3	4.943	78.80	65.65	65.79	72.12	67.00	65.54	3.24	3.25	3.56	3.31	3.24	51.55	51.85	56.11	52.67	51.24
29.3–33.3	0.865	79.67	83.73	84.17	87.68	84.92	83.30	0.72	0.73	0.76	0.73	0.72	52.27	52.57	56.86	53.40	51.96
33.3–37.3	0.452	80.12	85.03	85.08	88.94	86.52	84.91	0.38	0.38	0.40	0.39	0.38	52.65	52.96	57.27	53.79	52.34
37.3–41.3	2.982	83.10	74.12	75.23	78.85	75.89	73.05	2.21	2.24	2.35	2.26	2.18	54.87	55.20	59.62	56.06	54.52
41.3–45.3	2.240	85.34	87.50	87.82	91.54	88.57	87.48	1.96	1.97	2.05	1.98	1.96	56.83	57.17	61.67	58.04	56.48
45.3–49.3	0.310	85.65	21.59	21.59	28.17	23.15	21.32	0.07	0.07	0.09	0.07	0.07	56.89	57.23	61.76	58.11	56.55
49.3–53.3	10.216	95.87	70.10	71.04	74.26	71.73	69.21	7.16	7.26	7.59	7.33	7.07	64.05	64.49	69.34	65.44	63.62
53.3–57.3	0.934	96.80	90.10	90.17	93.33	91.00	90.02	0.84	0.84	0.87	0.85	0.84	64.89	65.33	70.21	66.29	64.46
57.3–61.3	0.595	97.40	90.98	91.04	94.06	91.83	90.95	0.54	0.54	0.56	0.55	0.54	65.44	65.88	70.77	66.83	65.00
61.3–65.3	0.165	97.56	81.66	82.07	85.14	83.13	80.79	0.14	0.14	0.14	0.14	0.13	65.57	66.01	70.91	66.97	65.13
65.3–69.3	0.270	97.83	64.20	64.43	73.97	68.11	64.37	0.17	0.17	0.20	0.18	0.17	65.74	66.19	71.11	67.16	65.30
69.3–73.3	0.680	98.51	87.19	87.28	91.08	88.11	87.03	0.59	0.59	0.62	0.60	0.59	66.34	66.78	71.73	67.76	65.90
73.3–77.3	0.605	99.12	59.66	59.70	68.91	61.61	59.58	0.36	0.36	0.42	0.37	0.36	66.70	67.14	72.15	68.13	66.26
77.3–81.3	0.014	99.13	69.02	69.02	82.07	73.37	69.02	0.01	0.01	0.01	0.01	0.01	66.71	67.15	72.16	68.14	66.27
81.3–85.3	0.020	99.15	76.30	76.30	82.96	76.30	76.30	0.02	0.02	0.02	0.02	0.02	66.72	67.17	72.18	68.15	66.28
85.3–89.3	0.109	99.26	87.93	87.93	89.60	88.42	87.93	0.10	0.10	0.10	0.10	0.10	66.82	67.26	72.28	68.25	66.38
89.3–93.3	0.077	99.34	60.20	60.59	66.01	62.07	59.01	0.05	0.05	0.05	0.05	0.05	66.87	67.31	72.33	68.30	66.42
93.3–97.3	0.040	99.38	77.88	77.88	83.18	79.40	77.69	0.03	0.03	0.03	0.03	0.03	66.90	67.34	72.36	68.33	66.45
97.3–100	0.623	100.00	93.27	93.90	94.86	94.01	92.95	0.58	0.58	0.59	0.59	0.58	67.48	67.92	72.95	68.91	67.03

can be achieved by introducing instruction sequences specifically designed to sensitize rare internal signal combinations.

- **Increase activation diversity for multi-condition defects:** Defects requiring multiple activation conditions—particularly those mapped to multiple conditional faults—are inherently more difficult to detect. The analysis indicates that existing STLs often fail to simultaneously satisfy all required conditions. To address this, STL design should incorporate diversified execution patterns that exercise different operand values, instruction mixes, and control-flow scenarios, thereby increasing the likelihood of activating complex defect conditions.
- **Perform technology-aware STL validation:** The comparative analysis across technology libraries shows that defect detectability and fault multiplicity vary significantly with library characteristics. As a result, STL effectiveness cannot be assumed to be portable across technologies. It is therefore necessary to validate and, if needed, adapt STLs for each target library using defect-oriented evaluation flows.

To illustrate how TT%-based analysis can guide STL improvement, we consider the group of defects with TT% below 10%, which consistently exhibits the lowest detection rates across both technology libraries. These defects require highly specific input conditions at the cell level, which are

TABLE 19. Taxonomy of the defects in Table 1 according to the number of outputs and conditional faults per output to inject. In dark gray, the signals where both faults are needed.

(a) Single output – Single fault

A	*B*	*CI*	*CO*	*S*	*D*1	*D*2	*D*3	*D*4	*D*5	...
0	0	1	0	1	0	0	1	2	3	...
0	1	0	0	1	3	1	1	2	1	...
1	0	1	1	0	0	2	0	2	0	...
1	1	0	1	0	0	1	0	0	3	...
...	...	...	...	...	...	...	...	...	...	...

(b) Single output – Multiple faults

A	*B*	*CI*	*CO*	*S*	*D*1	*D*2	*D*3	*D*4	*D*5	...
0	0	1	0	1	0	0	1	2	3	...
0	1	0	0	1	3	1	1	2	1	...
1	0	1	1	0	0	2	0	2	0	...
1	1	0	1	0	0	1	0	0	3	...
...	...	...	...	...	...	...	...	...	...	...

(c) Multiple outputs – Single fault

A	*B*	*CI*	*CO*	*S*	*D*1	*D*2	*D*3	*D*4	*D*5	...
0	0	1	0	1	0	0	1	2	3	...
0	1	0	0	1	3	1	1	2	1	...
1	0	1	1	0	0	2	0	2	0	...
1	1	0	1	0	0	1	0	0	3	...
...	...	...	...	...	...	...	...	...	...	...

(d) Multiple outputs – Variable faults

A	*B*	*CI*	*CO*	*S*	*D*1	*D*2	*D*3	*D*4	*D*5	...
0	0	1	0	1	0	0	1	2	3	...
0	1	0	0	1	3	1	1	2	1	...
1	0	1	1	0	0	2	0	2	0	...
1	1	0	1	0	0	1	0	0	3	...
...	...	...	...	...	...	...	...	...	...	...

(e) Multiple outputs – Multiple faults

A	*B*	*CI*	*CO*	*S*	*D*1	*D*2	*D*3	*D*4	*D*5	...
0	0	1	0	1	0	0	1	2	3	...
0	1	0	0	1	3	1	1	2	1	...
1	0	1	1	0	0	2	0	2	0	...
1	1	0	1	0	0	1	0	0	3	...
...	...	...	...	...	...	...	...	...	...	...

TABLE 20. Taxonomy of the defects in Table 2 according to the number of outputs and conditions per output to inject. In dark gray, the signals where both conditions are needed.

(a) Single output – Single fault

A	*B*	*CI*	*CO*	*S*	*D*6	*D*7	*D*8	*D*9	*D*10	...
0	1	R	R	F	0	0	0	0	3	...
R	1	0	R	F	0	1	0	2	0	...
0	F	1	F	R	2	1	3	2	3	...
0	1	F	F	R	2	0	0	1	0	...
0	R	0	0	R	2	0	0	0	0	...
1	F	1	1	F	0	1	0	0	0	...
...	...	...	...	...	...	...	...	...	...	...

(b) Single output – Multiple faults

A	*B*	*CI*	*CO*	*S*	*D*6	*D*7	*D*8	*D*9	*D*10	...
0	1	R	R	F	0	0	0	0	3	...
R	1	0	R	F	0	1	0	2	0	...
0	F	1	F	R	2	1	3	2	3	...
0	1	F	F	R	2	0	0	1	0	...
0	R	0	0	R	2	0	0	0	0	...
1	F	1	1	F	0	1	0	0	0	...
...	...	...	...	...	...	...	...	...	...	...

(c) Multiple outputs – Single fault

A	*B*	*CI*	*CO*	*S*	*D*6	*D*7	*D*8	*D*9	*D*10	...
0	1	R	R	F	0	0	0	0	3	...
R	1	0	R	F	0	1	0	2	0	...
0	F	1	F	R	2	1	3	2	3	...
0	1	F	F	R	2	0	0	1	0	...
0	R	0	0	R	2	0	0	0	0	...
1	F	1	1	F	0	1	0	0	0	...
...	...	...	...	...	...	...	...	...	...	...

(d) Multiple outputs – Variable faults

A	*B*	*CI*	*CO*	*S*	*D*6	*D*7	*D*8	*D*9	*D*10	...
0	1	R	R	F	0	0	0	0	3	...
R	1	0	R	F	0	1	0	2	0	...
0	F	1	F	R	2	1	3	2	3	...
0	1	F	F	R	2	0	0	1	0	...
0	R	0	0	R	2	0	0	0	0	...
1	F	1	1	F	0	1	0	0	0	...
...	...	...	...	...	...	...	...	...	...	...

(e) Multiple outputs – Multiple faults

A	*B*	*CI*	*CO*	*S*	*D*6	*D*7	*D*8	*D*9	*D*10	...
0	1	R	R	F	0	0	0	0	3	...
R	1	0	R	F	0	1	0	2	0	...
0	F	1	F	R	2	1	3	2	3	...
0	1	F	F	R	2	0	0	1	0	...
0	R	0	0	R	2	0	0	0	0	...
1	F	1	1	F	0	1	0	0	0	...
...	...	...	...	...	...	...	...	...	...	...

rarely exercised by existing STLs due to limited operand diversity and constrained instruction sequences.

TABLE 21. Number of conditional static faults per defect in the example of Table 1. Note: only the visible lines in the example have been considered.

Defect	C0 sa0	sa1	S sa0	sa1	#Faults
D1		✓	✓		2
D2	✓	✓		✓	3
D3		✓			1
D4			✓	✓	2
D5	✓	✓	✓	✓	4

TABLE 22. Number of conditional dynamic faults per defect in the example of Table 2. Note: only the visible lines in the example have been considered.

Defect	C0 str	stf	sa0	sa1	S str	stf	sa0	sa1	#Faults
D6					✓				1
D7	✓	✓	✓						3
D8		✓			✓				2
D9		✓			✓	✓			3
D10	✓	✓			✓	✓			4

For example, a subset of low-TT% defects in arithmetic and logic units requires simultaneous activation of rare input combinations combined with specific propagation conditions. Current STLs, which are primarily designed for SAF coverage, tend to favor frequently occurring operand patterns and do not systematically generate these rare conditions. As a result, these defects remain undetected even when overall coverage is high.

A targeted improvement strategy consists of introducing instruction sequences that explicitly increase operand diversity and internal signal variation, such as combining dependent arithmetic operations with varied operand values or introducing controlled control-flow variations. These modifications increase the probability of activating low-TT% conditions without significantly increasing test length.

This example demonstrates that TT% can be used not only as an analysis metric but also as a practical guide for STL refinement, enabling engineers to focus on the most critical defect classes rather than extending test programs indiscriminately.

V. CONCLUSION

This paper presented a structured framework for evaluating the effectiveness of SBST programs under defect-oriented CAT models. Unlike traditional approaches focused on fault coverage reporting, the proposed methodology combines defect mapping, post-processing, and TT%-based analysis to enable accurate and interpretable defect-level evaluation.

The results obtained using the proposed framework show that existing SAF-targeted STLs achieve substantial coverage on CAT defects, largely due to inherent functional redundancy. However, a subset of defects remains systematically hard to detect, primarily due to intrinsic observability constraints. The TT% metric proves effective in identifying

these critical defects and in guiding targeted improvements in STL design.

The analysis further highlights the impact of technology library characteristics on defect detectability and confirms the importance of defect-oriented evaluation for in-field test strategies. Although current tool limitations restrict the analysis to combinational cells and supported dynamic faults, the excluded defects account for only about 6% of the CAT fault space, and the observed trends remain representative. Future work will extend the framework to sequential elements and additional dynamic defects, and will integrate TT%-based analysis into STL optimization flows, enabling a closed-loop defect-oriented methodology and further assessing scalability on larger industrial designs.

Overall, this work establishes a practical and extensible framework for defect-aware evaluation of SBST programs, moving beyond traditional coverage-driven methodologies. By enabling systematic identification and targeting of hard-to-detect defects, the proposed approach provides a concrete foundation for improving STL quality in safety-critical systems and supports the integration of defect-oriented models into industrial SBST validation flows.

APPENDIX
DETAILED CAT FAULT TAXONOMY

In the following, we classify each defect based on the number of cell outputs it affects and on the number and type of conditional faults that must be injected on those outputs. We assume that, for each cell output, the characterization may require up to two static conditional faults (sa0 and sa1) and up to four dynamic conditional faults (sa0, sa1, str, stf). Let n_{out} denote the number of outputs of the cell.

- *Single output – Single fault:* the defect can be detected by targeting exactly one conditional fault (either sa0, sa1, str or stf) on a single cell output. This is the only category that includes a single conditional fault and does not require post-processing. In the example cell, $D3$ requires testing a conditional sa1 on output CO (see Table 19a), and $D6$ requires testing a conditional str on output S (see Table 20a). Since by definition only one conditional fault is associated with the defect, the number of faults in this category is always equal to one for both static and dynamic defects.
- *Single output – Multiple faults:* the defect can be detected by targeting more than one of the fault types (sa0, sa1, str, stf) on a single cell output. In the example cell, $D4$ requires testing conditional sa1 or sa0 on S (see Table 19b), and $D7$ requires testing at least two of str, stf, or sa0 on CO (see Table 20b).
  For static defects, each output can be associated with at most two conditional faults (sa0 and sa1). Since this class requires more than one conditional fault on a single output, and at most two static fault types are available, the number of faults in this category is always equal to two for static defects. For dynamic defects, up to four conditional fault types can exist on a single output (sa0, sa1, str, stf). Therefore, the number of faults in this category is always between two and four, depending on how many of these fault types are actually needed to detect the defect.
- *Multiple outputs – Single fault:*
  the defect can be detected by targeting exactly one conditional fault per affected output, but on more than one cell output. In the example cell, $D1$ requires testing a conditional sa1 on CO *or* a conditional sa0 on S (see Table 19c), and $D8$ requires testing a conditional str on S or a conditional stf on CO (see Table 20c).
  By definition of this class, each affected output contributes exactly one conditional fault. Let k be the number of outputs on which the defect is observable. Then the number of conditional faults is k, with $2 \leq k \leq n_{\text{out}}$ because at least two outputs must be involved for the defect to be classified as "multiple outputs", and at most all outputs can be affected. Consequently, the number of faults in this category is between two and n_{out}, for both static and dynamic defects.
- *Multiple outputs – Variable faults:* the defect can be detected by targeting a combination in which some outputs require a single conditional fault, while at least one other output requires more than one conditional fault. In the example cell, $D2$ requires testing conditional sa1 or sa0 on CO, and only sa1 on S (see Table 19d), whereas $D9$ requires testing conditional str or stf on S, and only stf on CO (see Table 20d).
  For static defects, each output can contribute either one or two conditional faults (sa0 and/or sa1). To belong to this class, at least two outputs must be affected, and at least one of them must require two faults. The minimum number of conditional faults therefore occurs when one output contributes two faults and another output contributes one, leading to a total of three faults. The maximum number of faults is obtained when one output contributes only one fault and all the remaining $n_{\text{out}} - 1$ outputs contribute two faults each, resulting in $1 + 2 \cdot (n_{\text{out}} - 1) = 2\, n_{\text{out}} - 1$ faults. Hence, for static defects in this category, the number of faults is between three and $2n_{\text{out}} - 1$.
  For dynamic defects, each output can contribute between one and four conditional faults (sa0, sa1, str, stf). Again, at least two outputs must be affected, and at least one output must require more than one conditional fault. The minimum is still three faults (one output with two faults and another with one). The maximum is obtained when one output has a single dynamic fault and all the remaining $n_{\text{out}} - 1$ outputs have four faults each, leading to $1 + 4 \cdot (n_{\text{out}} - 1) = 4\, n_{\text{out}} - 3$ faults. Thus, for dynamic

defects in this category, the number of faults is between three and $4n_{\text{out}} - 3$.

- *Multiple outputs – Multiple faults:* the defect can be detected by targeting more than one conditional fault on more than one cell output. In the example cell, $D5$ requires testing conditional sa1 or sa0 on CO or S (see Table 19e), and $D10$ requires testing both str and stf on CO or S (see Table 20e).
  For static defects, each affected output must contribute at least two conditional faults, and each output can contribute at most two. Therefore, a defect in this class must involve at least two outputs, each with two faults, leading to a minimum of four faults. If all n_{out} outputs are affected, each with two faults, the maximum is $2n_{\text{out}}$ faults. For dynamic defects, an affected output must contribute at least two dynamic faults (e.g., a pair among sa0, sa1, str, stf), and at most four. The minimum is again four faults (two outputs with two faults each), while the maximum, when all n_{out} outputs are affected with four dynamic faults each, is $4n_{\text{out}}$.

The specific conditional static faults for each defect for the examples are summarized in Table 21 and Table 22. For convenience, the last column in the tables reports the number of faults.

REFERENCES

[1] F. Hapke, R. Krenz-Baath, A. Glowatz, J. Schloeffel, H. Hashempour, S. Eichenberger, C. Hora, and D. Adolfsson, "Defect-oriented cell-aware ATPG and fault simulation for industrial cell libraries and designs," in *Proc. Int. Test Conf.*, Nov. 2009, pp. 1–10.

[2] F. Hapke, W. Redemund, J. Schloeffel, R. Krenz-Baath, A. Glowatz, M. Wittke, H. Hashempour, and S. Eichenberger, "Defect-oriented cell-internal testing," in *Proc. IEEE Int. Test Conf.*, Nov. 2010, pp. 1–10.

[3] F. Hapke, J. Schloeffel, W. Redemund, A. Glowatz, J. Rajski, M. Reese, J. Rearick, and J. Rivers, "Cell-aware analysis for small-delay effects and production test results from different fault models," in *Proc. IEEE Int. Test Conf.*, Sep. 2011, pp. 1–8.

[4] M. Yilmaz, K. Chakrabarty, and M. Tehranipoor, "Test-pattern grading and pattern selection for small-delay defects," in *Proc. 26th IEEE VLSI Test Symp. (Vts)*, Apr. 2008, pp. 233–239.

[5] S. K. Goel, N. Devta-Prasanna, and R. P. Turakhia, "Effective and efficient test pattern generation for small delay defect," in *Proc. 27th IEEE VLSI Test Symp.*, May 2009, pp. 111–116.

[6] S. K. Goel and K. Chakrabarty, *Testing for Small-Delay Defects in Nanoscale CMOS Integrated Circuits*. Boca Raton, FL, USA: CRC Press, 2013.

[7] F. Hapke and J. Schloeffel, "Introduction to the defect-oriented cell-aware test methodology for significant reduction of DPPM rates," in *Proc. 17th IEEE Eur. Test Symp. (ETS)*, May 2012, pp. 1–6.

[8] *Road Vehicles—Functional Safety—Part 1-12*, Standard ISO 26262-1:2018, 2018.

[9] A. Nardi and A. Armato, "Functional safety methodologies for automotive applications," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2017, pp. 970–975.

[10] A. Nardi, S. Camdzic, A. Armato, and F. Lertora, "Design-for-safety for automotive IC design: Challenges and opportunities," in *Proc. IEEE Custom Integr. Circuits Conf. (CICC)*, Apr. 2019, pp. 1–8.

[11] F. A. da Silva, A. Cagri Bagbaba, S. Hamdioui, and C. Sauer, "An automated formal-based approach for reducing undetected faults in ISO 26262 hardware compliant designs," in *Proc. IEEE Int. Test Conf. (ITC)*, Oct. 2021, pp. 329–333.

[12] F. Pratas, T. Dedes, A. Webber, M. Bemanian, and I. Yarom, "Measuring the effectiveness of ISO26262 compliant self test library," in *Proc. 19th Int. Symp. Quality Electron. Design (ISQED)*, Mar. 2018, pp. 156–161.

[13] M. Psarakis, D. Gizopoulos, E. Sanchez, and M. Sonza Reorda, "Microprocessor software-based self-testing," *IEEE Design Test Comput.*, vol. 27, no. 3, pp. 4–19, May 2010.

[14] A. Riefert, L. Ciganda, M. Sauer, P. Bernardi, M. S. Reorda, and B. Becker, "An effective approach to automatic functional processor test generation for small-delay faults," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2014, pp. 1–6.

[15] P. Bernardi et al., "Recent trends and perspectives on defect-oriented testing," in *Proc. IEEE 28th Int. Symp. On-Line Test. Robust Syst. Design (IOLTS)*, Sep. 2022, pp. 1–10.

[16] R. Cantoro, M. Grosso, I. Guglielminetti, R. Khoshzaban, and M. S. Reorda, "Assessing the effectiveness of software-based self-test programs for static cell-aware test," in *Proc. IEEE Eur. Test Symp. (ETS)*, May 2024, pp. 1–4.

[17] D. Gizopoulos, M. Psarakis, M. Hatzimihail, M. Maniatakos, A. Paschalis, A. Raghunathan, and S. Ravi, "Systematic software-based self-test for pipelined processors," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 16, no. 11, pp. 1441–1453, Nov. 2008.

[18] R. Cantoro, F. Garau, P. Girard, N. Kolahimahmoudi, S. Sartoni, M. S. Reorda, and A. Virazel, "Effective techniques for automatically improving the transition delay fault coverage of self-test libraries," in *Proc. IEEE Eur. Test Symp. (ETS)*, May 2022, pp. 1–2.

[19] R. Cantoro, P. Girard, R. Masante, S. Sartoni, M. S. Reorda, and A. Virazel, "Self-test libraries analysis for pipelined processors transition fault coverage improvement," in *Proc. IEEE 27th Int. Symp. On-Line Test. Robust Syst. Design (IOLTS)*, Jun. 2021, pp. 1–4.

[20] M. Grosso, S. Rinaudo, A. Casalino, and M. S. Reorda, "Software-based self-test for transition faults: A case study," in *Proc. IFIP/IEEE 27th Int. Conf. Very Large Scale Integr. (VLSI-SoC)*, Oct. 2019, pp. 76–81.

[21] K.-H. Chen, B.-Y. Yang, J.-R. Liang, H.-L. Chen, and J.-L. Huang, "Automatic test program generation for transition delay faults in pipelined processors," in *Proc. IEEE Int. Test Conf. Asia (ITC-Asia)*, Aug. 2021, pp. 1–6.

[22] M. Grosso, M. Sonza Reorda, and S. Rinaudo, "Software-based self-test for delay faults," in *VLSI-SoC: New Technology Enabler*, C. Metzler, P.-E. Gaillardon, G. De Micheli, C. Silva-Cardenas, and R. Reis, Eds., Cham, Switzerland: Springer, 2020, pp. 1–19.

[23] Singh, Inoue, Saluja, and Fujiwara, "Software-based delay fault testing of processor cores," in *Proc. Test Symp.*, 2003, pp. 68–71.

[24] K. Christou, M. K. Michael, P. Bernardi, M. Grosso, E. Sanchez, and M. S. Reorda, "A novel SBST generation technique for path-delay faults in microprocessors exploiting gate- and RT-level descriptions," in *Proc. 26th IEEE VLSI Test Symp. (VTS)*, Apr. 2008, pp. 389–394.

[25] C. H.-P. Wen, L.-C. Wang, K.-T. Cheng, K. Yang, W.-T. Liu, and J.-J. Chen, "On a software-based self-test methodology and its application," in *Proc. 23rd IEEE VLSI Test Symp. (VTS)*, May 2005, pp. 107–113.

[26] L. Anghel, R. Cantoro, R. Masante, M. Portolan, S. Sartoni, and M. S. Reorda, "Self-test library generation for in-field test of path delay faults," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 42, no. 11, pp. 4246–4259, Nov. 2023.

[27] (2022). *PULPino Microcontroller System*. [Online]. Available: https://github.com/pulp-platform/pulpino

[28] *Open-Cell 45nm FreePDK*. Accessed: Mar. 1, 2025. [Online]. Available: https://si2.org/open-cell-library/

[29] (2023). *Stuck-At STLs for Pulpino-RI5CY*. Accessed: CAD Group, Politecnico di Torino. [Online]. Available: https://github.com/cad-polito-it/pulpinori5cystls

REZA KHOSHZABAN (Graduate Student Member, IEEE) received the M.Sc. degree in electronics engineering from Politecnico di Torino, in 2023, where he is currently pursuing the industrial Ph.D. degree with the Department of Control and Computer Engineering (DAUIN), in collaboration with STMicroelectronics, Italy. His research interests include embedded systems and HPC testing and reliability, with a specific emphasis on defect-oriented testing and cell-aware testing (CAT) methodologies.

IACOPO GUGLIELMINETTI received the M.Sc. degree in computer and control engineering from Politecnico di Torino, in 2022. In 2022, he joined ST-POLITO s.c.ar.l., as a Digital Test Engineer, where he focuses on functional testing methods and automatic test pattern generation techniques. His research interests include functional and structural testing.

MICHELANGELO GROSSO (Senior Member, IEEE) received the M.Sc. degree in electronics and the Ph.D. degree in computer and systems engineering from Politecnico di Torino, Italy, in 2004 and 2008, respectively. From 2008 to 2011, he was a Postdoctoral Researcher with Politecnico di Torino. In 2012, he joined ST-POLITO s.c.ar.l., as an Application Development Engineer. In 2017, he moved to STMicroelectronics, where he is currently a Digital Designer and a Senior Member of the Technical Staff. His research interests include the test and reliability of digital circuits and systems, and design automation.

MATTEO SONZA REORDA (Fellow, IEEE) received the M.Sc. degree in electronics and the Ph.D. degree in computer engineering from Politecnico di Torino, Italy, in 1986 and 1990, respectively. He is currently a Full Professor with the Department of Control and Computer Engineering, Politecnico di Torino. He published more than 400 articles in the area of test and fault-tolerant design of reliable circuits and systems, receiving several best paper awards at major international conferences. He is involved in numerous research projects with companies and other research centers worldwide.

RICCARDO CANTORO (Member, IEEE) received the M.Sc. and Ph.D. degrees in computer engineering from the Politecnico di Torino, Turin, Italy, in 2013 and 2017, respectively. He is currently an Associate Professor with the Department of Computer Engineering, Politecnico di Torino. His research interests include software-based functional testing of SoC and memories, and machine learning applied to test and diagnosis.