Towards ISO 26262-compliant Vector Processors: Combining STLs and LBIST

Towards ISO 26262-compliant Vector Processors: Combining STLs and LBIST

Gustavo Vilar de Farias*, Dwijesh Kurada†, Sergiu-Mohamed Abed*†,
Juan-David Guerrero-Balaguera*‡, Josie E. Rodriguez Condia*, Ahmet Cagri Bagbaba†,
Felipe Augusto da Silva†, Matteo Sonza Reorda*
*Department of Control and Computer Engineering, Politecnico di Torino, Turin, Italy
†Cadence Design Systems, Munich, Germany
‡Escuela Colombiana de Ingenieria Julio Garavito, Bogota, Colombia

***Abstract*—Vector Processing Units (VPUs) are increasingly integrated into safety-critical automotive SoCs to accelerate general-purpose and Machine Learning inference workloads, making their functional safety essential in compliance with ISO26262. This work explores the effectiveness of common safety mechanisms (SMs) for permanent faults in a VPU (i.e., Software Test Libraries, or STLs, Logic Built-In Self-Test, or LBIST, and their combination). Without any SM, the VPU achieves a Single Point Fault Metric (SPFM) of only 1.2%, confirming the necessity of suitable SMs. STLs alone satisfy the SPFM threshold for ASIL B with values of 95.43%, 96.38%, and 95.98% for 2-, 4-, and 8-FU VPU configurations, respectively, while LBIST alone meets the SPFM target for ASIL D at 2-FU (99.36%) and 8-FU (99.11%), and ASIL C at 4-FU (98.69%). The hybrid approach meets the SPFM target for ASIL C across all configurations, with up to 11% lower area overhead and up to 66% shorter minimum testing interval than LBIST alone, making it a practical solution for continuous in-field automotive deployment.**

***Index Terms*—DFT, Functional Safety, FMEDA, SBST, VPUs**

I. Introduction

Currently, hardware accelerators designed for Artificial Intelligence (AI) workloads [1] are being widely adopted in the automotive industry to accommodate the computational complexity of deep learning tasks, e.g., object detection, and semantic segmentation [2] [3]. One versatile class of such accelerators comprises *Vector Processing Units* (VPUs), which are hardware units that adhere to the Single-Instruction Multiple-Data (SIMD) compute paradigm, exploiting the data-level parallelism associated with vector and matrix operations, making them well-suited for AI applications [4]. When adopted in safety-critical domains, such as automotive applications, hardware devices must comply with the stringent requirements of the ISO 26262 automotive functional safety standard [5] to ensure a certain Automotive Safety Integrity Level (ASIL) before in-field deployment. Achieving a proper ASIL requires adopting safety mechanisms (SMs) to detect faults and possibly move the system to a safe state before a fault leads to a hazardous event.

Choosing an SM requires considering aspects like area overhead, test time, and safety metrics. For example, Dual-Core Lockstep is highly effective at detecting faults but incurs an area overhead of more than 100%. Error Correction Codes (ECCs) are also popular SM, achieving high fault detection but with a significant area overhead, while mainly addressing temporary faults. For this reason, CPUs commonly employ mixtures of SMs [6] to balance area, performance overhead, and safety metrics. In this context, prior work on CPU SMs covers Software Test Libraries (STLs)-based self-test flows for automotive microcontrollers [7], Logic Built-In Self-Test (LBIST)-based in-field diagnosis [8], and hybrid hardware-software combinations targeting lockstep processors [6], [9]. For VPUs specifically, existing work had addressed fault tolerance via ECCs [10], in-field STL based-testing [11], and triple-modular redundancy [12]. However, to the best of our knowledge. the evaluation of LBIST and hybrid approaches as main SMs in VPUs has not been previously addressed.

When implementing a hybrid SMs, it is crucial to emphasize the architectural differences between CPUs and VPUs. VPUs are characterized by a more regular structure, in which Functional Units (FUs) constitute the most relevant part of the design. These FUs are particularly well-suited to be tested by STLs, which are highly attractive because they do not introduce additional hardware logic while achieving good Diagnostic Coverage (DC) and short test time. However, STLs can hardly achieve a high DC for the control part; on the other hand, LBIST can be effectively adopted to test the control logic.

To measure the efficacy of these SMs in achieving the target ASIL for the considered design, the ISO 26262 mandates that a Failure Modes, Effects, and Diagnostic Analysis (FMEDA) be performed across the hierarchy of the target design to calculate safety metrics and estimate the overall ASIL of the hardware.

This work explores the trade-offs among three SMs and assesses their effectiveness on a configurable VPU for Edge AI (Klessydra-T [13]), targeting ISO 26262 compliance. In detail, we consider three VPU configurations and analyze STLs, LBIST, as well as a hybrid approach cleverly combining the two (LBIST tests the control part, while STLs test the FUs). We compute the resulting cost and DC, and perform a gate-level (GL) FMEDA to obtain the Single Point Fault Metric (SPFM).

Results show that the STLs satisfy the SPFM threshold for ASIL B across all configurations, while LBIST meets the ASIL D SPFM target for 2-FU and 8-FU configurations.

This work was supported by the TIRAMISU project: EU Grant:101169378 HORIZON-MSCA-2023-DN-01.

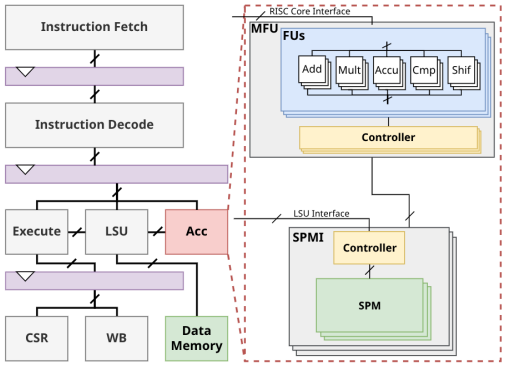

Fig. 1. A general scheme of the Klessydra-T VPU architecture

The hybrid approach achieves comparable SPFM values to LBIST at lower area overhead and with significantly reduced minimum testing interval, making it an effective solution for continuous in-field automotive deployment.

II. Background

A. Klessydra-T

Klessydra-T [13] (Fig. 1) is a highly configurable VPU integrated in a RISC-V core designed for edge AI applications. The vector accelerator is divided into two parts: the vector-oriented multipurpose functional unit (MFU), which includes the FUs and a controller, and the scratchpad-memory interface (SPMI), which includes the scratchpad memories and a controller. Inside the MFU, the FUs are the units that execute the vector operations and include an Adder/Subtractor, Multiplier, Accumulator, Shifter, and Comparator. These modules perform integer operations and can be configured to use 8-, 16-, or 32-bit data widths. Moreover, the MFU's Controller decodes and dispatches the vector instructions to the target FU. Instead, the SPMI stores the vector operands and results.

B. Safety mechanisms

Safety Mechanisms are hardware and/or software techniques for detecting faults and ensuring a system fails safely. They are crucial for achieving a target ASIL as they heavily influence the calculations of the safety metrics defined by ISO 26262. In general, the SM performance is measured by its DC, which represents the percentage of dangerous faults detected per SM. Among the SMs techniques [14], **Software Test Libraries** (STLs) are a software-based strategy to detect permanent faults without requiring design modifications. STLs use functional stimuli to activate a device and compare the outputs with an expected value. In in-field environments, STLs can be run when the hardware is IDLE exploiting their short test duration, thereby detecting faults with minimal performance overhead.

Alternatively, **LBIST** is a DfT strategy that relies on inserting modules into the design to generate stimuli and compare the results against an expected signature. For this implementation, the flip-flops (FFs) in the design are first converted to scan FFs and connected into chains. Each chain is then connected to a pseudo-random pattern generation (PRPG) and to a compressor. The PRPG feeds the scan chain with stimuli, while the compressor computes a signature from the scanned-out values and compares it with the expected value. On top of that, test points can be placed to increase the design observability and controllability. While recent industrial architectures have successfully adapted LBIST for in-system testing during key-on and key-off phases [15], [16], deploying it for continuous, periodic in-field testing without halting the application remains challenging and requires complex complementary hardware. To overcome these limitations, this work explores a cost-effective hybrid strategy.

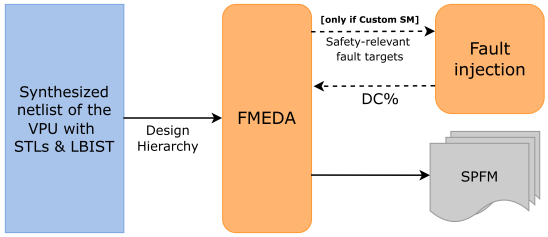

Fig. 2. Safety methodology for evaluating a VPU functional safety

C. Functional safety

ISO 26262 defines **functional safety** as the *absence of unreasonable risk due to hazards caused by malfunctioning behavior of electrical/electronic systems* [5]. As part of functional safety analysis, **FMEDA** is employed as a quantitative hardware safety analysis method following a hierarchical, bottom-up approach, starting from low-level design elements and propagating results towards higher levels of the design hierarchy. The FMEDA enables the systematic identification and classification of hardware failure modes across the design hierarchy, along with the evaluation of implemented SMs and their diagnostic coverage. By combining failure rates and diagnostic coverage, FMEDA facilitates the computation of key hardware safety metrics, including the SPFM, thereby providing quantitative evidence of the achieved safety performance. SPFM measures the robustness of the hardware against single-point faults (SPF) and residual faults (RF) [5], where SPF and RF refer to faults not covered by any safety mechanism and faults escaping the SMs, respectively. SPFM is calculated with the following formula: $SPFM = 1 - \frac{\sum(\lambda_{SPF}+\lambda_{RF})}{\sum\lambda}$, where the lambdas indicate failure rates, the denominator being the total base failure rate of the whole hardware. Note that SPFM is one of several metrics required for full ISO 26262 complaince; a complete assessment is beyond the scope of this work.

III. Adopted Methodology

This section describes our methodology for analyzing functional safety in VPUs, as illustrated in Fig. 2. First, the SMs are implemented within the VPU. The synthesized netlist, with the SMs, is then imported into an FMEDA tool, which identifies safety-relevant failure modes and, for custom SMs, conducts a fault-injection campaign to determine their DC. Lastly, these DC values are back-annotated into the FMEDA to compute the SPFM.

A. Safety mechanisms

For each target VPU architecture and configuration, three SM techniques for hardware protection are implemented and analyzed: STLs, LBIST, and a hybrid approach. The STL strategy relies on functional stimuli generated using the method described in [11] to detect faults in a VPU. In contrast, the LBIST strategy detects faults by automatically modifying the GL model to include the LBIST structures summarized in II-B.

This work explores the feasibility of a hybrid approach that combines the STL and the LBIST strategy to improve fault detection while limiting area and test duration overhead. In detail, the hybrid approach implements a reduced version of the LBIST structures targeting only the VPU module least effectively covered by STLs, i.e., the control unit. Instead, the FUs have a regular structure that is highly testable with functional stimuli.

B. FMEDA

The FMEDA is performed based on the synthesized netlist of each VPU version, ensuring that failure rate estimation and SM evaluation are directly grounded in the implemented design. The FMEDA first identifies safety-relevant fault targets—elements in the design where a fault could lead to dangerous output—and, for custom SMs, directs a fault injection campaign to determine their DC accordingly. Not all design elements require injection; only those considered dangerous are targeted, focusing the campaign on safety-critical logic.

For each element in the design, design instances from the netlist are associated with their corresponding FMEDA subpart. Failure modes—specific ways in which a system can fail to perform its intended function—are then identified and created, and each is allocated to its respective subpart. The safety architecture then defines which mechanism protects each subpart. The base failure rate (BFR)—the per-device failure rate in FIT derived from technology parameters under IEC 62380 [17] using the NAND2-equivalent cell area as a reference unit—provides the FIT rate for each element which, combined with DC values, feeds into the SPFM calculation.

While DC values for well-established SMs can be estimated from ISO 26262 guidelines, custom mechanisms such as STL and LBIST require dedicated fault injection campaigns to determine their diagnostic coverage. These DC values are back-annotated into the FMEDA. For elements protected by more than one safety mechanism, as in the Hybrid approach, the DC contributions of each mechanism are summed, where each mechanism's contribution is obtained from fault injection campaigns isolating its incremental detection over the population already covered by the other mechanism. The resulting combined DC is then used in the FMEDA to compute SPFM across all evaluated configurations.

IV. Experimental Results

For the experiments, we synthesize the Klessydra-T VPU using a 130nm PDK [18] in three configurations: with 2 FUs (2-FU), 4 FUs (4-FU), and 8 FUs (8-FU). For each configuration, we analyze and implement 3 SMs: STLs, LBIST, and Hybrid. The SMs are implemented on the MFU portion of the VPU, since memory can be protected using other strategies, such as MBIST. For each SM and VPU configuration, the FMEDA for permanent faults is performed with Cadence *Midas* using the VPU GL model. Moreover, we identify the safe fault points in the GL model with *JasperGold* (by Cadence), with around 1% of faults being safe across all configurations. These safe faults are used to compute the diagnostic coverage (DC), which is the percentage of faults detected by the SM when excluding the safe faults. Furthermore, the clock period used in the experiments is $10ns$.

TABLE I
Area overhead, Minimum test times, and Diagnostic Coverage for the evaluated SMs and VPU Configurations

VPU config	SM	Area ov. (%)	MinTest (ns)	DC (%)
2-FU	STLs	0	31,110	95.43
	LBIST	52	1,759,200	99.29
	Hybrid	46	1,000,700	98.05
4-FU	STLs	0	31,110	96.36
	LBIST	39	2,691,900	98.56
	Hybrid	28	1,164,700	97.92
8-FU	STLs	0	31,110	95.98
	LBIST	29	4,670,200	98.99
	Hybrid	19	1,595,200	97.26

For fault detection, STLs are analyzed via fault injection campaigns (FC) across all configurations. The STLs target fault detection across the entire VPU and are based on the execution of two AI core programs (i.e., matrix multiplication and convolution) and Ad Hoc routines targeting the hard-to-detect faults. For the two applications, the matrices used are 32x32 for the 2- and 4-FU configurations, and 64x64 for the 8-FU architecture. The total STLs time are 1,342,450, 1,125,270, and 2,380,960 ns for 2-, 4-, and 8-FU, respectively.

In contrast, for the LBIST approach, a new GL model featuring LBIST structures in all internal modules is generated for each VPU configuration resorting to a commercial tool, and FC is then performed. The LBIST features 10 scan chains and can generate and evaluate 1,024 pseudo-random patterns.

The proposed Hybrid approach combines STLs and LBIST by executing both mechanisms in series. For this strategy, we first evaluate the STLs effectiveness to obtain STLs DC and the list of faults that could not be detected. We then use these results to perform the LBIST insertion targeting the VPU controller, which yields a STLs DC below 90% across all the configurations. This strategy reduces LBIST area overhead by applying the structures only to the VPU controller. The LBIST in the hybrid approach aims to detect faults that the STLs do not detect, thereby improving the overall DC. Notably, the LBIST infrastructure is placed on the top block, and a GL model with the reduced LBIST is generated for each configuration before FC is performed.

Table I reports the area overhead (ov.), the minimal test time (MinTest), and DC for the considered hardware configurations (HW Config) and SMs. The area ov. is calculated based on the baseline VPU, which corresponds to 192,112, 348,237, and 656,428 μm^2 for the 2-, 4-, and 8-FU, respectively. The MinTest represents the minimum required testing time, i.e., the minimum time slot duration which the application must be suspended for test purposes. For the STLs, the MinTest

is the time required to execute a minimal STLs procedure, including the time to perform one STLs operation and to save and restore the VPU memory state. For the LBIST and Hybrid approaches, MinTest equals the LBIST execution time, including the time to perform the test and to compute and compare the signature. The VPU memory does not need to be saved or recovered, as it is a separate module that is isolated during any LBIST execution. After LBIST execution, the VPU state is recovered via a simple reset and restoration of two configuration registers (i.e., data width and vector size); this recovery time is negligible and excluded from MinTest.

Clearly, STLs do not introduce any area ov. Instead, the SMs that modify the design (LBIST and Hybrid) impose an area ov. that depends on the HW Config. For all the configurations, the hybrid approach shows a lower area ov. than LBIST alone, because LBIST structures are not added to the entire VPU. The area savings of using the hybrid approach compared with LBIST are 6%, 11%, and 10% for 2-, 4-, and 8-FU, respectively. Area savings grow with the FU count since the controller does not scale with the number of FUs, while FUs dominate the overall design area.

For the in-field test, higher MinTest times lead to lower HW availability, thereby increasing the impact of the test on the application performance. While LBIST can be adapted for in-field testing, doing so requires integrating complex complementary hardware mechanisms to partition tests and reduce continuous offline time, which increases area costs. However, the results show that the hybrid approach provides a more balanced trade-off, compared with traditional LBIST, it can significantly reduce MinTest (by around 43%, 57%, and 66% for 2-, 4-, and 8-FU, respectively) while achieving comparable DC, with minor differences of about 1.25%, 0.64%, and 1.73% for the 2-, 4-, and 8-FU, respectively. This reduction in the MinTest is due to the smaller scan chains in the Hybrid approach compared with the LBIST strategy. Notably, for in-field use, the VPU's MFU should be disconnected from the RISC core and from memory during LBIST execution.

The Hybrid approach's DC and area ov. are directly tied to the effectiveness of the STLs; in the Klessydra-T case, the STLs achieve high DC due to the VPU's integer-only units and relatively small controller. For the FMEDA, the Klessydra-T VPU hierarchy is decomposed in six elements, each with an associated failure mode: Adder/Subtractor (incorrect sum), Multiplier (incorrect product), Accumulator (incorrect accumulated value), Shifter (incorrect shift result), Comparator (incorrect comparison), and Controller (incorrect control signal).

Fig. 3 summarizes the resulting SPFM and DC values. Without any SM, the VPU yields an SPFM of around 1.2%, confirming that the bare hardware falls far below any SPFM threshold under ISO 26262 Part 5 [19], which reflects the computational nature of VPU, so nearly all faults directly corrupt the vector output. Instead, when STLs are applied as the sole safety mechanism, SPFM reaches 95.43%, 96.38%, and 95.98% for 2-FU, 4-FU, and 8-FU, respectively, satisfying the SPFM ASIL B requirement (90% covered faults).

Applying LBIST yields 99.36%, 98.69%, and 99.11% for

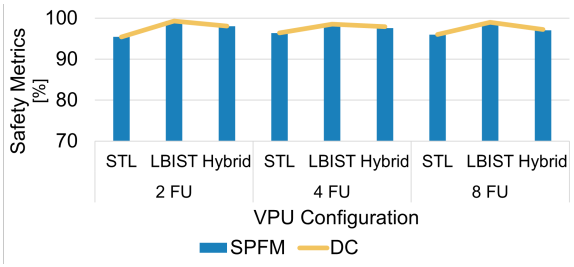

Fig. 3. SPFM and DC for the considered SMs and HW configs

2-FU, 4-FU, and 8-FU respectively, satisfying the SPFM target for ASIL D at 2-FU and 8-FU and for ASIL C at 4-FU. However, LBIST requires the design to be taken fully offline: its minimum test chunk equals its total test duration, which grows substantially with FU count from 1,759,200 ns at 2-FU to 4,670,200 ns at 8-FU, and it incurs the highest area overhead across all configurations: 52%, 39%, and 29% respectively.

The Hybrid approach achieves SPFM of 98.06%, 97.61%, and 97.07% for 2-FU, 4-FU, and 8-FU respectively, satisfying the SPFM target for ASIL C across all configurations. In the Hybrid configuration, LBIST targets exclusively the Controller block—a fixed-size block whose relative weight in the total design's hardware budget decreases as FU count grows; since LBIST's incremental SPFM contribution is anchored to this fixed block, Hybrid SPFM decreases slightly with parallelism. This scoping decision is precisely what enables the growing practical advantages reported in Table I: area overhead savings of up to 11% and MinTest reductions of up to 66% compared with LBIST alone, enabling in-field STLs-based detection alongside significantly shorter offline LBIST intervals across all configurations.

V. Conclusions

This work evaluated the functional safety of the Klessydra-T VPU via an FMEDA across three VPU configurations—2-, 4-, and 8-FUs—each evaluated under three SMs—STLs, LBIST, and Hybrid. STLs satisfy the ASIL B target for SPFM across all configurations. LBIST meets the SPFM's ASIL D threshold at 2- and 8- FUs and ASIL C at 4-FUs, but requires long halting times on the VPU for testing and incurs the highest area overhead among all configurations. The Hybrid approach achieves the ASIL C SPFM threshold across all configurations by combining STL-based in-field operation with a controller-scoped LBIST. While scoping LBIST to the Controller causes Hybrid SPFM to decrease slightly with FU count—the Controller's relative hardware weight diminishes with increasing parallelism—the practical advantages over LBIST grow with scale: area overhead is reduced by up to 11% and minimum test time by up to 66%, making Hybrid a viable solution for scalable automotive deployment.

Future work includes optimizing STLs for the Hybrid approach by removing controller-targeting routines to further reduce testing time, extending the analysis to compute the Latent Fault Metric (LFM) — which quantifies coverage of multi-point faults — and validating SM compliance with fault-tolerant time interval — maximum time from fault to hazard — in ADAS scenarios.

REFERENCES

[1] B. Dally, "Hardware for deep learning," in *2023 IEEE Hot Chips 35 Symposium (HCS)*. IEEE Computer Society, 2023, pp. 1–58.

[2] D. Feng, C. Haase-Schütz, L. Rosenbaum, H. Hertlein, C. Gläser, F. Timm, W. Wiesbeck, and K. Dietmayer, "Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges," *Trans. Intell. Transport. Syst.*, vol. 22, no. 3, p. 1341–1360, Mar. 2021. [Online]. Available: https://doi.org/10.1109/TITS.2020.2972974

[3] Y. Li, L. Ma, Z. Zhong, F. Liu, M. A. Chapman, D. Cao, and J. Li, "Deep learning for lidar point clouds in autonomous driving: A review," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 8, pp. 3412–3432, 2021.

[4] C. Wang, C. Fang, X. Wu, Z. Wang, and J. Lin, "A scalable risc-v vector processor enabling efficient multi-precision dnn inference," in *2024 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2024, pp. 1–5.

[5] ISO, *ISO 26262 Road Vehicles - Function Safety*, International Standardization Organization Std., Dec. 2018.

[6] A. Floridia and E. Sanchez, "Hybrid on-line self-test strategy for dual-core lockstep processors," in *2018 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, 2018, pp. 1–6.

[7] P. Bernardi, R. Cantoro, S. De Luca, E. Sánchez, and A. Sansonetti, "Development flow for on-line core self-test of automotive microcontrollers," *IEEE Transactions on Computers*, vol. 65, no. 3, pp. 744–754, 2016.

[8] P. Bernardi, G. Filipponi, G. Iaria, C. Bertani, and V. Tancorre, "Logic diagnosis based on logic built-in self-test signatures collected in-field from failing system-on-chips," *Electronics*, vol. 13, no. 21, 2024. [Online]. Available: https://www.mdpi.com/2079-9292/13/21/4234

[9] A. Floridia and E. Sanchez, "On-line self-test mechanism for dual-core lockstep system-on-chips," *Microelectronics Reliability*, vol. 112, p. 113770, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0026271420300317

[10] M. Barbirotta, F. Minervini, C. R. Morales, A. Cristal, O. Unsal, and M. Olivieri, "Enhancing fault tolerance in high-performance computing: A real hardware case study on a risc-v vector processing unit," *IEEE Open Journal of the Computer Society*, vol. 5, pp. 553–565, 2024.

[11] G. Vilar de Farias, J. E. Rodriguez Condia, and M. Sonza Reorda, "A flexible framework for vector accelerators in-field testing," in *44th IEEE VLSI Test Symposium 2026*.

[12] M. Barbirotta, A. Cheikh, A. Mastrandrea, F. Menichelli, and M. Olivieri, "Analysis of a fault tolerant edge-computing microarchitecture exploiting vector acceleration," in *2022 17th Conference on Ph.D Research in Microelectronics and Electronics (PRIME)*, 2022, pp. 237–240.

[13] A. Cheikh *et al.*, "Klessydra-t: Designing vector coprocessors for multithreaded edge-computing cores," *IEEE Micro*, vol. 41, no. 2, 2021.

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

[15] P. K. Datla Jagannadha, M. Yilmaz, M. Sonawane, S. Chadalavada, S. Sarangi, B. Bhaskaran, S. Bajpai, V. A. Reddy, J. Pandey, and S. Jiang, "Special session: In-system-test (ist) architecture for nvidia drive-agx platforms," in *2019 IEEE 37th VLSI Test Symposium (VTS)*, 2019, pp. 1–8.

[16] T. McLaurin, "Periodic online lbist considerations for a multicore processor," in *2018 IEEE International Test Conference in Asia (ITC-Asia)*, 2018, pp. 37–42.

[17] International Electrotechnical Commission, "Reliability data handbook – universal model for reliability prediction of electronics components, pcbs and equipment," International Electrotechnical Commission (IEC), Geneva, Switzerland, Technical Report IEC/TR 62380:2004, Aug. 2004.

[18] K. Herman, N. Herfurth, T. Henkes, S. Andreev, R. Scholz, M. Müller, M. Krattenmacher, H. Pretl, and W. Grabinski, "On the versatility of the ihp bicmos open source and manufacturable pdk: A step towards the future where anybody can design and build a chip," *IEEE Solid-State Circuits Magazine*, vol. 16, no. 2, pp. 30–38, 2024.

[19] ISO, *Road vehicles — Functional safety — Part 5: Product development at the hardware level*, International Organization for Standardization Std. ISO 26 262-5:2018, Dec. 2018.