POLITECNICO DI TORINO
Repository ISTITUZIONALE

Netlist-Independent Functional Stress Pattern generation strategy for AI HW Accelerators embedded into SoCs

Original
Netlist-Independent Functional Stress Pattern generation strategy for AI HW Accelerators embedded into SoCs / Filipponi, G., Schwachhofer, D., Angione, F., Bertani, C., Corbellini, S., Di Gruttola Giardino, N., Garozzo, G., Insinga, G., Tancorre, V., Bernardi, P.. - In: IEEE TRANSACTIONS ON COMPUTERS. - ISSN 1557-9956. - 75:2(2026), pp. 554-566.

Availability:
This version is available at: 11583/3004589 since: 2025-10-29T14:39:59Z

Publisher:
IEEE

Published
DOI:

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

19 November 2026

Netlist-Independent Functional Stress Pattern Generation Strategy for AI HW Accelerators Embedded into SoCs

Gabriele Filipponi, *Student Member, IEEE*, Denis Schwachhofer, *Student Member, IEEE*, Francesco Angione, *Member, IEEE*, Claudia Bertani, Simone Corbellini, *Senior Member, IEEE*, Nicola di Gruttola Giardino, *Student Member, IEEE*, Giuseppe Garozzo, Giorgio Insinga, *Member, IEEE*, Vincenzo Tancorre, and Paolo Bernardi, *Senior Member, IEEE*

***Abstract*—Artificial Intelligence hardware accelerators are pervading the chip market. Most of the time, they are third-party IPs integrated by silicon manufacturers. As a consequence, their design may be obfuscated, which can introduce issues from a manufacturing testing perspective. This paper illustrates how to effectively and efficiently select the most appropriate functional stress stimuli for Artificial Intelligence (AI) Hardware (HW) Accelerators embedded in System-on-Chip (SoC). The proposed methodology is netlist independent; and it is based on both current measurements from the real chip and architectural evaluations. These ingredients are heuristically used to rank and sift the optimal functional patterns to apply along the Burn-In (BI) phase. Experimental results on two different Automotive SoCs manufactured by STMicroelectronics, demonstrate the effectiveness and efficiency of the proposed method.**

***Index Terms*—Functional testing, stress functional patterns, AI hardware accelerators, burn-in, automotive testing.**

I. Introduction

ARTIFICIAL Intelligence (AI) Hardware (HW) Accelerators are pervading the Chips market [1], [2], [3]. In safety-critical domains, such modules perform the mathematical operations required by AI algorithms in hardware for speeding up the classification [4] or prediction [5] process during themission mode (i.e., pedestrian recognition, road signals detection, engine monitoring). Therefore, semiconductor manufacturers must minimize test escapes to avoid dangerous situations.

Central-Processing-Units (CPUs) control and configure the AI HW accelerators, and manage the inference execution; CPUs and AI module netlists may come from specialized third-party companies and may be integrated into a System-on-Chip (SoC) by a different system integrator. To preserve the Intellectual Properties (IPs), it is not unusual that such AI modules come as *hard macros* and are plugged in the SoC as *closed boxes*. The integration process can impact the testing activity effectiveness; no netlist and/or limited visibility of Design for Testability (DfT) features may introduce severe limitations to the generation of appropriate test patterns. In the literature, the development of functional stress approaches for AI HW accelerator is unexplored. By using functional stress in BI, additional stress can be introduced to complement scan-based stress. Functional workloads are executed at the operating frequency at which the device will run in the field, behaving as it would during mission mode [6]. In contrast, scan-oriented techniques are executed at a lower frequency and involve reconfiguring the circuit in a non-functional, non-field-like manner. Consequently, functional workloads can generate more stress since they are executed more frequently compared to scan-based techniques within the same BI duration. Moreover, the electrical stress induced by functional workloads creates a more uniform stress distribution over the gates in the DUT compared to scan- or DFT-oriented techniques [7].

This paper explores an alternative solution to netlist-based approaches that operates like an Automatic-Test-Pattern-Generation (ATPG) to select functional patterns for AI HW Accelerators, especially in the direction of Burn-In (BI) stress. Due to the current increasing trends in terms of complexity for both SoC (i.e., the gate count growing exponentially [8]) and DfT structures [7], [9], [10], the proposed method provides an effective and efficient way to synthesize a suite of functional stress programs specifically targeting arithmetic computational modules, with a primary focus on AI hardware accelerators embedded in SoC.

Received 14 November 2024; revised 5 August 2025; accepted 29 October 2025. Date of publication 3 November 2025; date of current version 15 January 2026. This work was supported in part by Advantest through the Graduate School "Intelligent Methods for Test and Reliability" (GSIMTR), University of Stuttgart. Recommended for acceptance by T. Jones. *(Corresponding author: Gabriele Filipponi.)*

Gabriele Filipponi, Francesco Angione, Simone Corbellini, Nicola di Gruttola Giardino, Giorgio Insinga, and Paolo Bernardi are with the Department of Control and Computer engineering, Politecnico di Torino, 10129 Turin, Italy (e-mail: gabriele.filipponi@polito.it; francesco.angione@polito.it; simone.corbellini@polito.it; nicola.digruttola@polito.it; giorgio.insinga@polito.it; paolo.bernardi@polito.it).

Denis Schwachhofer is with the University of Stuttgart, 70174 Stuttgart, Germany (e-mail: denis.schwachhofer@informatik.uni-stuttgart.de).

Claudia Bertani, Giuseppe Garozzo, and Vincenzo Tancorre are with the STMicroelectronics, 20864 Agrate Brianza, Italy (e-mail: claudia.bertani@st.com; giuseppe.garozzo@st.com; vincenzo.tancorre@st.com).

Digital Object Identifier 10.1109/TC.2025.3627803

© 2025 The Authors. This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

The approach aims to assemble a pool of functional stress patterns that maximize the toggle metrics of circuit nodes and uniformly distribute toggling across the AI hardware accelerator to achieve an extremely effective BI stress, without requiring the netlist.

The proposed method relies on two main measurement elements, which are then combined into a selection algorithm:

- The *current consumption* of functional stress patterns;
- The *Memory Toggle* Metrics, metrics purely based on the programmer's view of the AI HW accelerator based on the accelerator's memory map.

In brief, the proposed method evaluates a large population of functional stress patterns which could be created from scratch (i.e., randomly) or inherited from validation/verification functional program suites. Firstly, the population of functional stress patterns is executed on-chip to measure current absorption. Afterwards, the proposed method ranks the list of functional stress patterns by current consumption, selects the ones with the highest current consumption, and filters them according to the Memory Toggle Metrics.

It is crucial to highlight that the proposed method does not resort to netlist information for selecting the functional stress patterns, the netlist could not be available to the company assembling and testing SoCs with the AI HW Accelerator, the proposed method purely relies on the programmers' view of memory, control and status registers of the AI HW accelerator. Nevertheless, it can be used also in case the netlist is available, as it is largely faster, and thus more efficient, than simulation-based approaches to reach similar results [11].

The paper presents experimental results on two automotive SoCs manufactured by STMicroelectronics, including an AI HW Accelerator module. The proposed method produces excellent results compared to random selection of functional stress patterns, to which results are shown for both SoCs. Moreover, for the first SoC, the generation and evaluation costs are compared with those of a generation process based on a genetic algorithm [12] and a scan-stress suite [13].

The paper is structured as follows: in Section II the background is presented, Section III shows the proposed approach; while Section IV presents the experimental results and Section V draws some conclusions.

II. Background

A. Manufacturing Flow and Stress Metrics

A typical manufacturing test flow is divided into different stages, each one of them targeting different defects and with a specific goal to discard manufactured faulty devices [14]:

- **Wafer Sort**: it is performed at the wafer level checks for the primary electrical functionalities of the chip and executes structural test patterns [14];
- **Package Test**: it is applied after packaging, it tests all features and the electrical characteristics of the pins by executing structural tests [15];
- **Burn-In (BI)**: it exacerbates potential latent defects by "stressing" the device through environmental conditions and running specific electrical procedures [16];
- **System-Level Test**: it is an additional test for safety-critical and automotive devices [17], usually resorting to complex functional programs [18], [19];
- **Final Test**: it detects faults by applying a mix of structural and functional tests.

For the sake of this work, the focus is on the BI phase, where stress procedures are applied to exacerbate latent defects in the circuit by applying environmental stress (i.e., high temperature provided by a climatic chamber) and electrical stress (i.e., gate activation by scan patterns or the execution of a functional program). Specific metrics are adopted to evaluate the stress capabilities of test patterns [20], [21]. With newer technologies, adequate activity can be obtained only with complex methods, e.g., static electrical stress approaches (which share similarities with *IDDQ* [22] patterns).

In this work, the following electrical stress metrics for BI are considered: for a circuit with N nodes, T_i defines the number of transitions taking place on the circuit node i during the test application. *Toggle Coverage* (TC) and *Toggle Activity* (TA) [23] are stress metrics resulting from monitoring T_i over the set of all nodes.

TC punctually measures how many gate ports toggle (both rising and falling edges are produced by the stress pattern) as shown in Equation 1.

$$\mathbf{TC\ [\%]} = 100 \times \frac{\sum_i^{0\to N} rising(T_i) + falling(T_i)}{2 \times N} \tag{1}$$

TA globally measures the strength of a stress pattern by two components: the $TA_{average}$, or $AVG(TA)$, and the $TA_{variance}$, $VAR(TA)$. The values of the latter metrics are displayed in Equation 2 and Equation 3. $AVG(TA)$ and $VAR(TA)$ need to be normalized in a quantum of time (e.g., 1 us).

$$\mathbf{AVG(TA)}\left[\frac{\mathbf{T}}{\mathbf{us}}\right] = \frac{\sum_i^{0\to N} T_i}{time[us]} \tag{2}$$

$$\mathbf{VAR(TA)} = \sqrt{\frac{\sum_i^{0\to N} (T_i - \hat{T})^2}{N}},\ \hat{T} = \frac{\sum_i^{0\to N} T_i}{N} \tag{3}$$

In order to meet BI stress requirements for high quality silicon devices, TC and TA have to show:

- High TC, stressing as many circuit nodes as possible;
- High $AVG(TA)$, implying high-stress effectiveness on the gates;
- Low $VAR(TA)$, uniformly stressing the gate, without introducing excessive stress in some circuit portions and low in others.

B. AI HW Accelerators

Artificial Intelligence workloads can be executed by CPUs, or Graphics-Processing-Units (GPUs), and Field-Programmable-Gate-Arrays (FPGAs). Albeit these computer architectures are suitable for High-end AI computations, i.e., Large-Language-Models (LLMS); they are not adequate for edge safety-critical applications, for example in the Automotive field. In such environments, AI computation

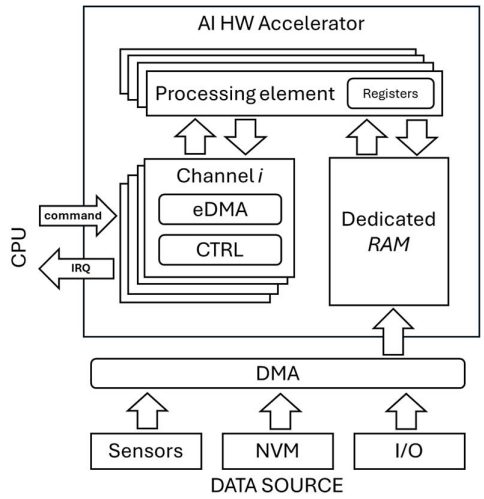


Fig. 1. Generic block diagram of an AI HW Accelerator.

requires short computing time to fulfill the real-time requirements. Stringent real-time requirements can be achieved only with Application-Specific Integrated Circuits (ASICs), or AI-oriented accelerators inside SoCs.

Fig. 1 illustrates a generic view of an AI HW Accelerator. This kind of modules is usually plugged into the SoC and configured by the CPU like a peripheral.

Given the regularity of AI computation, the architecture paradigm is shifted to a dataflow processing, where the data control is minimized and the data processing is maximized by the deeply pipelined unit of **Processing Elements (PEs)** [3]. The PEs are composed of **Arithmetic Logic Units** (ALUs) capable to perform AI-specific computations with different data formats (integer, floating point and fixed point).

Other than the computational circuits, and the simplified control unit (not reported in Fig. 1) the AI HW Accelerators include the following design blocks:

- A **dedicated RAM memory**, which can be populated from different sources, like Sensors, Non-Volatile Memories (NVM), or I/O peripherals through system Direct-Memory-Access (DMA), and from which the PEs retrieve data;
- A **Bus Interface** for different input channels, which contains a DMA for data movement to/from the accelerator, and Control and Configuration Registers for the accelerator that communicates with the CPU to receive commands or trigger Interrupt ReQuests (IRQ), in case the computation has completed or a mathematical exception raised from the computation. In addition, it provides an access to the dedicated RAM.

Access to the memory elements of the system, such as control registers and dedicated RAM blocks, is crucial for the proposed method.

III. Proposed Approach

Nowadays, more and more IPs from different sources are integrated into the same SoC, and sometimes IP vendors do not provide full visibility into their IP. The proposed approach focuses on assessing functional workloads using indirect methodologies based on current consumption and architectural metrics, without the need of the netlist. Therefore, the proposal could be valuable for system integrators and manufacturers to stress an AI accelerator without any netlist knowledge, relying solely on functional testing and the accelerator description provided in the user manual.

The proposed approach collects a set of functional patterns suitable for the execution on the AI HW Accelerator under stress. For the scope of this work, a functional stress pattern for an AI HW Accelerator is the content of the dedicated RAM, which will be sequenced to the PEs. Existing functional patterns may come from validation or verification suites, or may be created according to algebra rules about functional pattern content, for example, relying on a distribution of logic values '0' and '1'.

Once a population of functional stress patterns is available, the proposed method measures the current consumption for each functional stress pattern.

The initial population of functional stress patterns can be randomly generated with defined proportions of logic 1s and 0s in the memory. Given the functionality of the AI hardware accelerator (described as a model), their generation is based on an offline tool capable of ensuring functional validity; i.e., the functional correctness of the patterns without causing exceptions or overflows due to the numbers. Moreover, in terms of structural diversity, the tool ensures that the 0s and 1s are randomly distributed throughout the memory layout. However, it does not guarantee alignment with realistic workloads, since the focus is to functionally stress all the logic cones to/from memory available in the accelerator during the Burn-In phase. This work and its pattern creation can be the starting point to generate functional patterns by means of other tools and human intervention.

The current consumption associated to each functional stress pattern is an indirect indicator of the stress activity the AI HW accelerator is subject to [24], and the proposed method ranks the functional stress patterns in decreasing order of current consumption. The higher the current consumption, the higher the expected toggle activity of the circuit nodes.

Although current consumption is already a significant factor in guiding the selection, the mere composition of functional patterns that achieve high current consumption does not guarantee strong and uniform stress across all circuit nodes. Top-ranked programs may exhibit similar stress characteristics, covering some portions of the circuit while leaving others untouched.

In order to polarize the selection process, an heuristic algorithm is proposed that sifts the ranked list based on Memory Toggle Metrics described in Section III-B. The sifting process prunes out redundant functional stress patterns from the ranked list by looking at the content of the AI HW Accelerator memory elements (i.e., the dedicated RAM or control registers). The

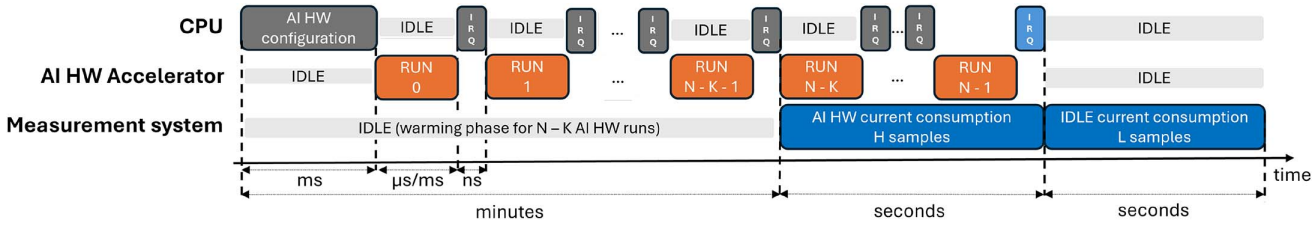


Fig. 2. Firmware execution flow to grade the current consumption of a functional pattern.

result of the algorithm is the identification of a small-sized set of functional stress patterns, whose composition effectively achieves high stress quality. The algorithm is described in Section III-C.

A. Current Consumption Measurement Flow

A current consumption measurement flow is used to grade the ability of functional stress patterns to excite the AI HW Accelerator. The required setup is a combination of measurement equipment and appropriate firmware run by the Device Under Test (DUT).

Regarding the measurement equipment, several works have already described current probing setups, which can be based on either direct methods (e.g., using inductive probes [25]) or indirect methods (e.g., using a shunt resistor and oscilloscope [24]). A major requirement is to sample at a frequency comparable to the device's working frequency; if the sampling frequency is too slow, it is suggested to repeat the measurement over multiple AI HW runs.

Once the measurement system is configured, the firmware that also controls the measurement system is executed. An explanatory view of the bare-metal firmware components is shown in Fig. 2.

The firmware runs on the CPU and the AI HW Accelerator, executing independent tasks. The measurement system is controlled by the SoC, which can trigger events through digital I/O pins (e.g., to the oscilloscope's input trigger).

The firmware configures the AI HW, which includes the data transfer to the internal RAM of the AI HW accelerator. Then N AI HW runs of the same functional stress pattern are scheduled. While the AI HW runs, the CPU enters IDLE mode, to minimize the influence on the current consumption. Every time an AI HW run terminates, an IRQ wakes up the CPU which launches the next run.

As depicted in Fig. 2, the measurement system is inactive at the beginning and in idle for K AI HW runs. This inactivity period allows the SoC to warm up and reach a stable current consumption [26]. An early current measurement may report lower current consumption values than with the system "at hot". Furthermore, the evaluation of several functional patterns may show inconsistent values.

The CPU firmware is in charge of counting the warming runs and switching on the measurement system at the right time. In Fig. 2 the IRQ triggered after K runs activates the measurement for the remaining $N - K$ runs. Ideally, the value of N should be $K + 1$ if the sampling frequency matches the working frequency of the module under evaluation. Unfortunately, this is not always possible (i.e., due to instrumentation limitations), then more than a single run needs to be measured, and N is decided upon the sampling and working frequencies ratio [27], as well standard practices to reduce readings instrumentation noise and errors.

More in details, the parameter K represents the index after which the system reaches steady-state behavior, identified through the temporal evolution of power consumption. Only the current samples collected after this index, namely $\{I_K, I_{K+1}, \ldots I_{N-1}\}$, are considered for steady-state analysis. The total number of samples N is determined dynamically, ensuring that a sufficient number of steady-state readings are collected to achieve the desired level of statistical confidence. The proposed methodology ensures the reliability and statistical validity of current measurements through a structured evaluation of measurement uncertainty, incorporating both instrumental uncertainty (Type B evaluation) and measurement repeatability (Type A evaluation). The instrumental uncertainty accounts for systematic errors arising from sensor accuracy limitations, which are derived from manufacturer specifications and calibration data. The repeatability, which is analyzed by evaluating the experimental standard deviation σ_K in steady-state current measurements, accounts for random fluctuations due to noise and environmental factors.

To further reduce the influence of random noise, the final reported measurement is obtained by averaging over H steady-state samples, which reduces the noise proportionally to the inverse square root of the number of H. The relative standard uncertainty (Type A) has been therefore evaluated as:

$$\mu_{A,r}(\bar{i}) = \frac{s \times 100}{\sqrt{H} \times \bar{i}}$$

The combined treatment of Type A and Type B uncertainties, together with the dynamic determination of K and N, guarantees that the reported current values reflect both accuracy and repeatability.

As mentioned, in the measurement phase, each run generates H samples. An IRQ, handled by the CPU, is generated at the end of each run, which introduces a number of ϵ samples. However, the IRQ time is much shorter with respect to the AI HW Accelerator computation time, meaning its contribution is close to 0.

Once the $N - K$ measurement runs are completed, the resulting power consumption is the samples average. Throughout

the paper we refer to current consumption (c^i) as the average ($\mu(x)$) of the samples.

At this point the consumption of the system in idle is measured with L samples. As at this stage the SoC has reached power stability the only requirement for L is to be sufficiently large to account for the measurement error.

The samples gathered during this final phase are utilized to construct a subtractive term [28] as an average of L samples. This phase is optional when experiments are conducted in a temperature-controlled environment (e.g., a climatic chamber); however, its inclusion is strongly recommended to mitigate inaccuracies and noise arising from ambient temperature fluctuations commonly encountered in standard laboratory settings.

Despite the current consumption being very helpful to distinguish better functional stress patterns from weak ones, it may introduce some "aliasing". In fact, it may grade many functional stress patterns as effective, which are exciting the same parts of the circuit, thus, including many snippets which do not contribute to stress the circuit as uniformly as expected. Therefore, the evaluation of the accelerator memory-related metrics is taken into account.

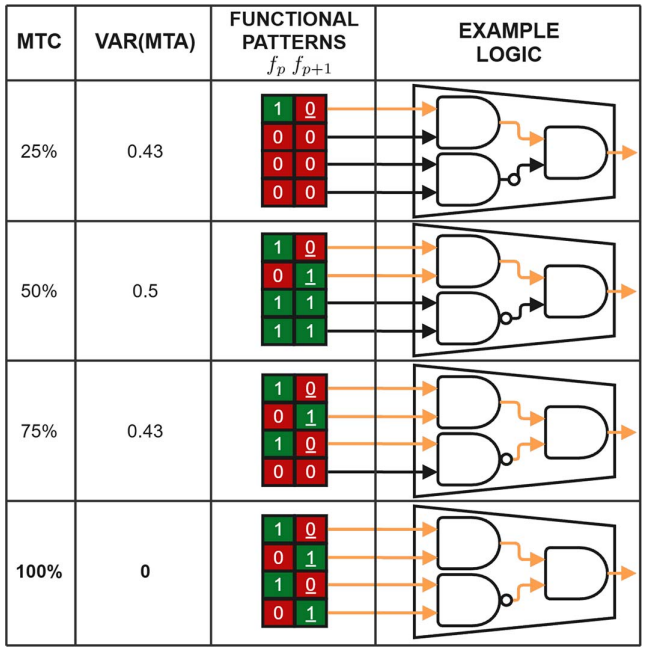


Fig. 3. Visual representation of the MTC and VAR(MTA) between two consecutive functional patterns in function to gate activity.

B. Memory Toggle Metrics

Memory Toggle Metrics are proposed to further guide the evaluation process of the functional patterns. They are based on the evaluation of memory elements visible in programmers' view of the AI HW Accelerator. These memory elements could be memory banks used to store the operands, or registers, whose content can be computed, for instance, by an emulation software (e.g., QEMU) or architectural simulator (e.g., Gem5).

The Memory Toggle Metrics aim at quantifying the activity of the memory elements along the application of a set of functional patterns. The specific metrics used to further grade each functional stress pattern are the following:

- **Memory Toggle Coverage** (MTC): for every bit in the programmers' memory elements, it measures the number of bits that show a transition from logic value '0' to '1' (rising) or '1' to '0' (falling) along the application of the functional patterns, similar to the TC Equation 1.
- **Memory Toggle Activity** (MTA): it considers the number of times every memory element is toggled during the functional stress patterns application and it is synthesized in:
  - $MTA_{Average}$ **or** $AVG(MTA)$, similar to Equation 2.
  - $MTA_{Variance}$ **or** $VAR(MTA)$, similar to Equation 3.

The objective of the proposed Memory Toggle Metrics is to establish a correlation by which elevated memory toggle coverage and activity are associated with increased circuit node toggle coverage and activity. Conversely, a functional stress pattern showing low values of MTC and MTA is expected to partially activate the circuit. Memory toggle metrics are somehow similar to high-level code metrics like statement and toggle coverage computed at RTL-level circuit description. The reasoning is that a constant value in a Flip-Flop or Memory Bit cell, may introduce low testability of the connected logic cone, as it is likely a constant "tied" value feeding the gates network. Gate activity and gate coverage are closely linked because active gates are typically those being exercised during testing. High gate coverage usually leads to higher gate activity. As illustrated in Fig. 3, higher MTC is expected to correspond to increased gate activity, while a lower variance of VAR(MTA) should indicate a more uniform distribution of gate activity. However, as shown in the figure, priority should be given to MTC, since a lower VAR(MTA) does not necessarily lead to higher gate activity. For instance, the 25% MTC case exhibits a lower VAR(MTA) compared to the 50% MTC case, yet the 50% MTC scenario yields greater gate activity.

An example of the computation of MTC is provided in Fig. 4. Memory elements are summarized in a squared table, where every cell represents a bit of observable memory elements. Such a map is produced by stacking the memory behavior of a set of functional stress patterns.

C. Functional Stress Pattern Selection Algorithm

Once all the functional stress patterns have been evaluated in terms of current consumption and the memory element content available for each, it becomes possible to run an algorithm based on the combination of these two metrics.

Fig. 5 presents the algorithm flow. The primary source of the algorithm is the list of functional patterns ordered by descending value of current consumption. The algorithm iteratively extracts the top-ranked individual and checks if it is increasing the MTC value to the current set of selected functional patterns. If so, the functional pattern is added to the functional stress pattern suite. The top-ranked functional pattern is certainly included in the suite, while the successive patterns need to be evaluated.

If the functional pattern currently selected from the list does not lead to an increase in the MTC, the algorithm then

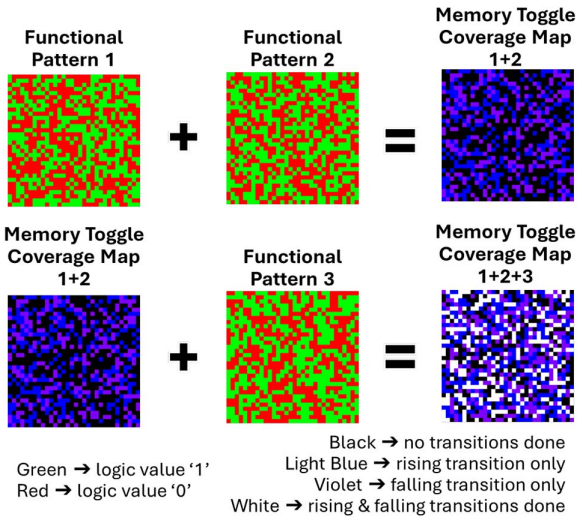


Fig. 4. Example about Memory Toggle Coverage computation.

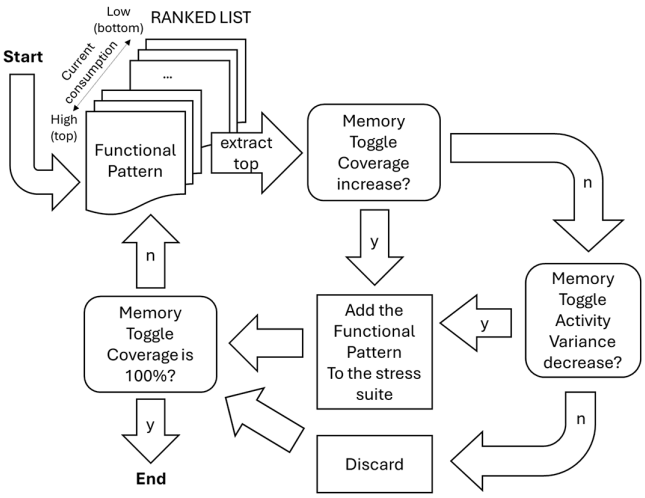


Fig. 5. Rank and Sift Selection Algorithm.

checks whether this pattern helps improve (i.e., reduce) the $MTA_{variance}$. If it does, the pattern is added to the functional stress pattern suite; otherwise, it is permanently discarded.

The process continues iteratively until the MTC reaches 100%, i.e., all memory elements accessible by the programmer toggle at least once along the application of the selected functional stress pattern suite.

The algorithm priorities the functional stress patterns demanding more current because they are the most promising in terms of circuit node activation [27]. Nevertheless, to achieve a high diversity in the functional stress suite, Memory Toggle Metrics are considered as well, to avoid selecting multiple programs that stress the same set of circuit nodes. Once the final set of functional stress patterns is selected by the heuristic rank-and-sift selection algorithm, the functional stress suite can then be executed cyclically in sequence for the desired duration during BI, depending on the required level of stress.

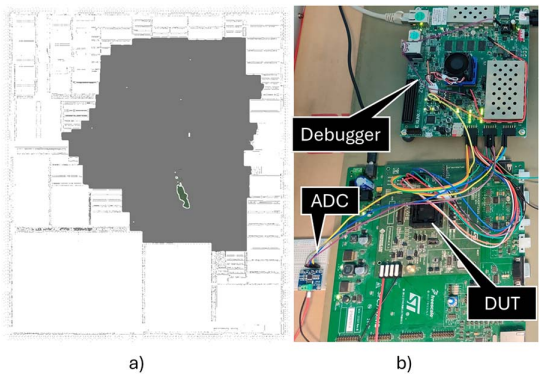


Fig. 6. a) Layout of the case of study with AI HW Accelerator gates highlighted in green and b) Measurement setup.

The rank and sift algorithm is straightforward from an algorithmic perspective, relying basically on selecting the top-ranked value from an ordered list. The strength of the proposed method lies in combining different indirect measurements (current consumption and memory toggle metrics) to grade functional stress patterns in two steps without resorting to netlist-based measurements: first, by reordering the patterns in decreasing order of current consumption, and then by using the memory toggle metrics to select the highest value.

IV. Experimental Results

A. Experimental Setup

The case of study is an automotive SoC, manufactured by STMicroelectronics, that incorporates an AI HW Accelerator [29] able to perform floating-point Multiply and Accumulate (MAC) vectorial computations and exponential evaluations. Furthermore, the AI hardware accelerator design features two independent calculation channels operating on a processing engine composed of a single ALU using a time-multiplexed scheme. The AI HW Accelerator is highlighted in green over the chip surface plot shown in Fig. 6(a), with the sea of gates in gray and the memory cores in white. The AI HW Accelerator includes around $80K$ gates and is equipped with a dedicated 32 KB RAM memory that stores the operands of the computations, from which Memory Toggle Metrics are computed.

Fig. 6(b) shows a photograph of the experimental workbench, including the real chip, a single-site FPGA-based debugger [30] to flash and start the firmware that applies the functional patterns, and a current sensor, which communicates with the debugger for retrieving the current measurement and transferring to host PC (connected through Ethernet and not visible in the image). The current (and power) sensor is based on a low-cost discrete module built with an INA219 power monitor IC coupled with a $100\,\mathrm{m}\Omega$ shunt resistor. The shunt resistor is part of the power supply line of the DUT, allowing the module to compute current and power by measuring the voltage drop across it. The INA219 is equipped with an internal Programmable Gain Amplifier (PGA). The gain is set to achieve a full range of $\pm 80\,\mathrm{mV}$. The ADC resolution of $12\,\mathrm{bit}$ paired with the PGA yields a voltage resolution of $\frac{80\,\mathrm{mV}}{2^{12}} = 0.0195\,\mathrm{mV}$. Considering the $100\,\mathrm{m}\Omega$ shunt resistance, it translates to a theoretical current

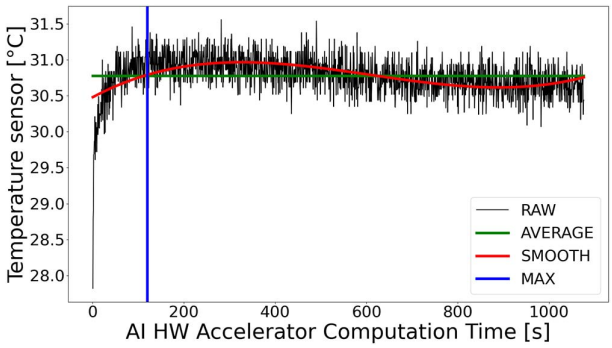


Fig. 7. Embedded thermometer's temperature profile.

resolution of $0.2\,\mathrm{mA}$. To improve measurement stability and compensate for noise, the module is configured to average 128 samples per reading in about $68\,\mathrm{ms}$, yielding $\approx 2\,\mathrm{Ksamples/s}$.

The selection algorithm based on the collected information and the Memory Toggle Metrics runs on the Host PC.

The gate-level netlist is available at both the gate and layout levels for this specific experimental setup. Although the proposed method does not require netlist information, layout data and gate-level simulations of the functional stress patterns identified by the method are presented to validate the proposed methodology.

B. Experiment Statical Confidence and Parameters

In the reported experimental setup, the Type B uncertainty, associated with the intrinsic accuracy of the current sensor, is conservatively estimated within the range of approximately $2{-}3\%$, based on manufacturer specifications and calibration assessments. The Type A uncertainty is evaluated by calculating the standard deviation, denoted as σ_K, of the steady-state current measurements during periods of stable operation. To obtain each final measurement, an average is computed over $H = 140$ steady-state samples. Experimental results shows that σ_K consistently remains below $0.5\,\mathrm{mA}$ after the $120\,\mathrm{s}$ execution time mark, leading a (Type A) standard uncertainty of 0.083%

The combined uncertainty of Type A and Type B contribution, remains sufficiently low, in the range of 3% to 4%, to ensure statistically robust and reliable measurement results, with a measurement repeatability that enables current comparison to within tens of micro-amperes. Additionally, raw temperature data obtained from the on-chip temperature sensor, as shown in Fig. 7, confirm the stabilization of temperature around the $120\,\mathrm{s}$ mark during steady-state operation. Given that a single execution run (t_{run}) requires up to $150\,\mathrm{us}$, the total number of samples within the stabilization window is $K = \frac{120\,\mathrm{s}}{t_{run}} = 800,000$. A measurement interval of $10\,\mathrm{s}$ is designated to define the transition phase, corresponding to $N - K = \frac{10\,\mathrm{s}}{t_{run}} \approx 80,000$.

C. Functional Stress Patterns Evaluation

The first experimental step consists of gathering the initial population of functional stress patterns. In this specific case, a pool of 700 functional stress programs is applying functional stress patterns corresponding to the content of the dedicated RAM in the AI HW Accelerator. Functional stress patterns are created randomly and according to a defined percentage of logic value '1'. In other words, M functional patterns are created by randomly distributing the $N\%$ of '1' over the dedicated RAM content.

TABLE I
POPULATION DETAILS OF THE FUNCTIONAL PATTERNS

Class	Amount of Randomly Distributed Values '1'	Number of Individuals	HIGHEST Current Consumption [mA]	AVERAGE Current Consumption [mA]
A	50%	200	50.90	46.63
B	40%	200	50.02	45.72
C	30%	100	49.96	43.67
D	20%	100	47.94	40.74
E	10%	100	42.19	37.52
F	0%-100%	5	37.00	34.25
G	provoking errors	20	9.99	7.36
B'	60%	10	48.21	45.57
C'	70%	2	45.56	44.45

Table I reports full details about the initial population characteristics, divided into classes according to the amount of randomly distributed logic values '1' or because of provoking error conditions.

Patterns belonging to classes A to F have been shown to avoid triggering exceptions such as overflows or underflows. Parameter N, remains below 50%, since a higher proportion of logical '1's results in larger numerical values being processed by the ALU of the AI HW Accelerator, thereby increasing the likelihood of arithmetic exceptions. Furthermore, as illustrated in Fig. 3, power consumption is expected to follow a Gaussian distribution with respect to the percentage of '1's in memory, peaking around 50%. This suggests that targeting approximately 50% of '1's allows for achieving representative power profiles while reducing the risk of arithmetic overflows.

Moreover, while power measurements at 60% and 70% '1's were feasible, program synthesis at 70% required over six hours to yield only a few valid individuals, and no viable programs were obtained at 80% within the same time-frame. Notably, despite the increased synthesis effort at higher proportions of '1's, the resulting power profiles at 60% and 70% were comparable to those observed at 40% and 30%, suggesting diminishing returns in program generation efficiency beyond the 50% threshold.

Table I reports the highest and average current consumption measured for every class of functional patterns. The current measurement flow takes about 6 minutes per functional pattern, distributed as follows: 3 minutes for flashing the firmware, 2 minutes for warming time, 10 seconds for current consumption measurement time, few seconds for measurements transmission to the host PC and to compute memory toggle metrics computations. The automated setup took about 4 wall-clock days to complete the evaluation of the pool of all 700 functional stress programs.

TABLE II
FUNCTIONAL STRESS PATTERN SELECTION RESULTS

Ranked Functional Patterns	Class	Current Consumption [mA]	Incremental		
			MTC [%]	MTA Average	MTA Variance
1	A	50.9	-	-	-
2	A	50.84	25.0183	0.50	0.50
3	A	50.63	50.0429	1.00	0.71
4	A	50.56	68.7063	1.50	0.86
5	A	50.2	81.2610	2.00	1.00
6	A	50.16	89.0542	2.50	1.12
7	B	50.02	93.7459	3.00	1.22
8	C	49.96	96.1398	3.44	1.32
9	B	49.95	97.5783	3.87	1.42
10	B	49.94	98.4885	4.34	1.52
11	B	49.9	99.0370	4.81	1.61
12	A	49.71	99.4692	5.31	1.69
13	B	49.67	99.6708	5.81	1.77
14	A	49.52	99.8214	6.31	1.84
15	A	49.49	99.9073	6.81	1.91
16	B	49.48	99.9435	7.31	1.97
17	B	49.38	99.9665	7.77	2.03
18	A	49.36	99.9825	8.27	2.09
19	B	49.29	99.9896	8.77	2.16
20	A	49.26	99.9949	9.27	2.22
21	B	49.23	99.9968	9.77	2.28
22	B	49.22	99.9978	10.23	2.33
23	A	49.19	99.9978	10.51	2.17
24	B	49.11	99.9980	10.99	2.22
25	C	49.04	99.9986	11.43	2.28
26	A	49.03	99.9988	11.93	2.34
27	A	49.02	99.9994	12.43	2.39
28	A	49.01	99.9996	12.93	2.44
29	A	48.99	99.9998	13.43	2.49
discarded	A	48.98	-	-	-
discarded	B	48.94	-	-	-
30	B	48.93	**100**	**13.93**	**2.54**
31	G	9.99	100	14.17	2.38
32	G	9.42	100	14.41	2.24

TABLE III
APPROACHES RESULTS COMPARISON

Number of Patterns	Type of Patterns	Generation Methodology	TC[%]*	AVG(TA) [T/us]	VAR(TA)
40	Functional	Random Selection	94.94	3009505	234014
39			94.69	2903340	227795
38			94.69	3015525	222603
37			94.90	2957831	219646
36			94.82	3011621	212613
35			94.79	2940629	205545
34			94.88	2994593	201015
33			94.84	3036523	195981
32			94.62	2939122	193504
32	*Structural*	*Scan-stress [13]*	*95.09*	*1497837*	*829*
32	*Functional*	*Netlist-based [12]*	*89.62*	*1763422*	*1639*
32	**Functional**	**Proposed**	**94.73**	**4545416**	**170932**
31	Functional	Random Selection	94.52	2994104	182261
30			94.63	2931979	176218
29			94.38	3070885	175612
28			94.41	2974546	165449
27			94.64	2857780	158744
26			94.51	2954290	156581
25			94.42	2854391	150557
24			94.44	2884231	136099
23			94.30	2936833	135296
22			94.44	2962143	131748
21			94.16	2988284	125653
20			94.48	2895112	115462
512	Structural	*Scan-stress [13]*	*98.03*	*1508718*	*11581*
32+**32**	Structural + Functional	Scan-stress [13] + **Proposed**	**96.57**	**4455521**	**171629**
512+**32**			**99.05**	**3774564**	**180789**

*for Random Selections it reports the average value over ten pattern sets.

Table II reports the results obtained by applying the selection algorithm presented in III-C. The algorithm selected 30 valid (i.e., not producing exceptions along the computation) functional stress patterns to compose the functional stress suite. The table shows the class of the chosen patterns, the corresponding current consumption, and the incremental values of MTC and MTA, including both average and variance. Patterns that are provoking errors are important, too, and added to the stress suite according to a separate but similar selection process, which looks for increased $MTA_{average}$ or decreased $MTA_{variance}$. Two functional patterns that trigger an overflow and underflow conditions are sufficient to optimize the Memory Toggle Metrics.

In order to further demonstrate the effectiveness of the proposed method, the identified functional stress patterns are evaluated through gate-level netlist simulation to extract toggle coverage and toggle activity, and the results are compared with other approaches, such as the scan-stress suite. Table III shows the comparison with random selections, a generation by a netlist-based approach in [12] and a scan-stress suite [13]. The first column reports the number of selected functional patterns; the second column the pattern typology: structural- or functional-based; for the random selection strategy, every line summarizes the results for ten sets with the corresponding number of patterns. Reported TC and TA values for random selections are averaging the results for each of the ten pattern set per line.

The proposed approach shows a circuit node Toggle Coverage of 94.73%, which is slightly worse then the average value of 94.94% reached by using 40 functional patterns, but inline with the same number of patterns. Nevertheless, TA values reached by the proposed selection methods are significantly better than those reached by random selection.

Moreover a comparison with the approach in [12] is shown. A similar setup based on a genetic algorithm is used to create stress functional programs encompassing 32 functional patterns, which are then graded directly in terms of circuit node toggle metrics over the netlist (which must be available to implement such a solution).

For a fair comparison, the netlist-based approach is executed for the same amount of time required to electrically grade the initial population of the proposed methodology, which is 4 wall-clock days.

Due to the complexity of the circuit to simulate, the netlist-based approach with 12 parallel simulations could complete the evaluation of only 200 functional stress programs in the same amount of time as the electrical grading. With this limitation, the values reached are way lower than those of the proposed approach.

A scan-based ATPG stress pattern set encompassing both 32 and 512 patterns [13] was graded over the netlist (which must be available to implement such a solution) in terms of toggle coverage and activity. The scan-stress with 32 patterns

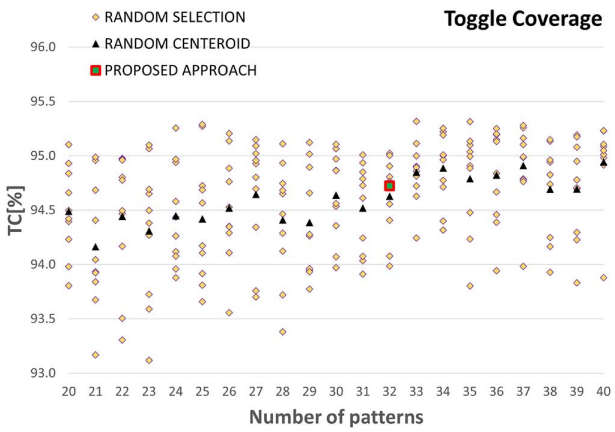


Fig. 8. Toggle Coverage (TC) comparison.

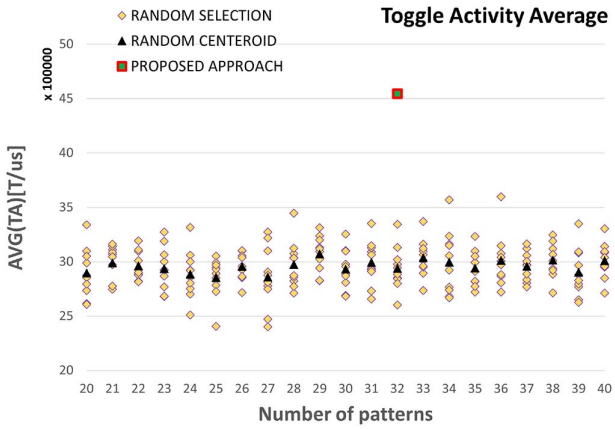


Fig. 9. Toggle Activity Average ($TA_{average}$) comparison.

manages to reach 95.09% of TC, which is slightly better then the proposed approach. However it exhibits worse TA, but better distributed as the VAR(TA) is lower. The scan-stress suite with a higher number of patterns, 512, reported for completeness, shows the highest TC value by a single methodology, 98.02%. However, this comes at the expense of memory requirements for the ATE, which often makes this approach unfeasible.

The last two rows of Table III report the incremental stress metrics with scan-stress suites and the proposed functional stress patterns. When adding the 32 selected functional patterns to the 32 of the scan-stress, the TC manages to reach 96.57%, where 1.48% of increment was provided only by the functional patterns. The same goes for the 512 scan-based pattern set reaching an astounding TC of 99.05%, with 1.02% increment only from the proposed functional patterns. For both the functional counterpart provides a solid increase in terms of TC while maintaining a high TA.

Figs. 8–10 show the behavior of the proposed approach with random selection populations.

The proposed approach TC is not the best but acceptable. Concerning AVG(TA), the proposed approach stands out from mediocre values reached by random selections. About VAR(TA), the value obtained with the proposed strategy is

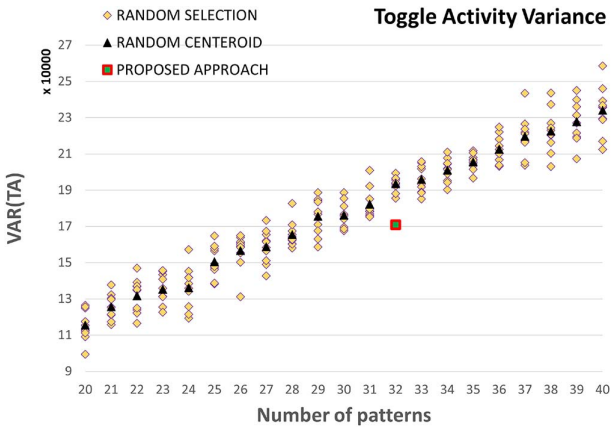


Fig. 10. Toggle Activity Variance ($TA_{variance}$) comparison.

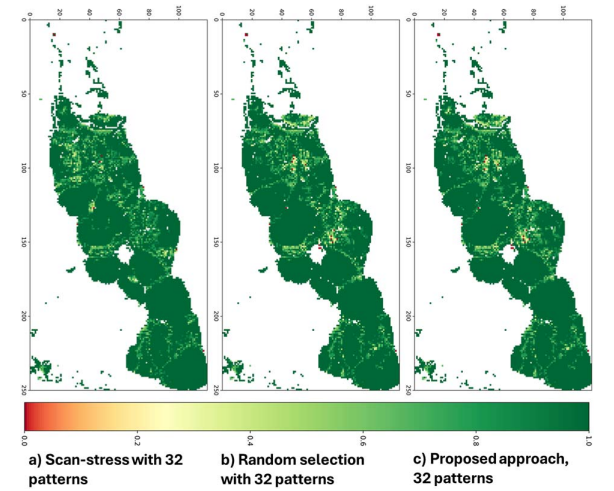


Fig. 11. Stress TC heatmap comparison.

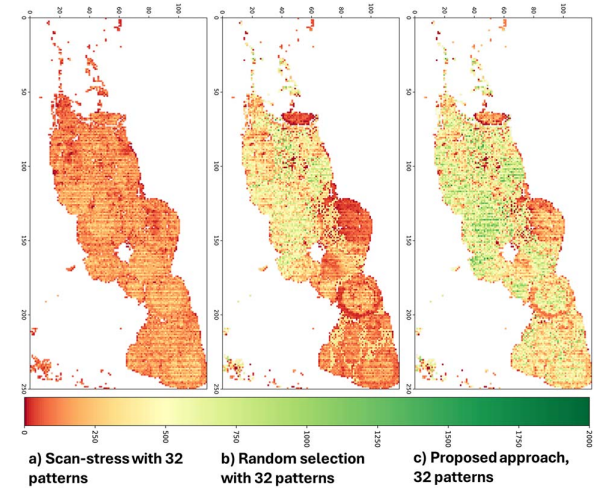


Fig. 12. Stress TA heatmap comparison.

again much better than the random selections, especially if the number of functional patterns grows.

Figs. 11, and 12 show respectively the node coverage and activity generated over a layout heatmap by (a) the scan-stress

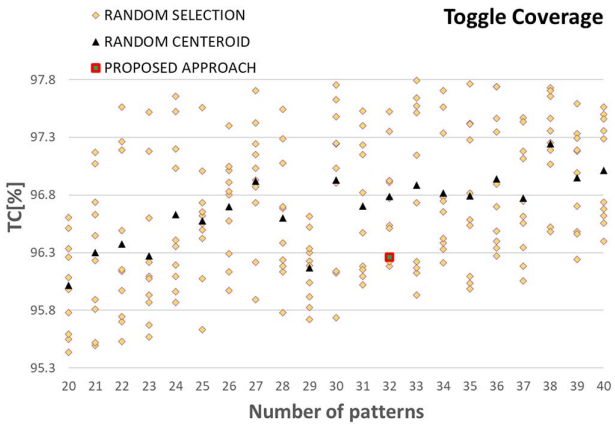


Fig. 13. (TC) comparison for DUT2.

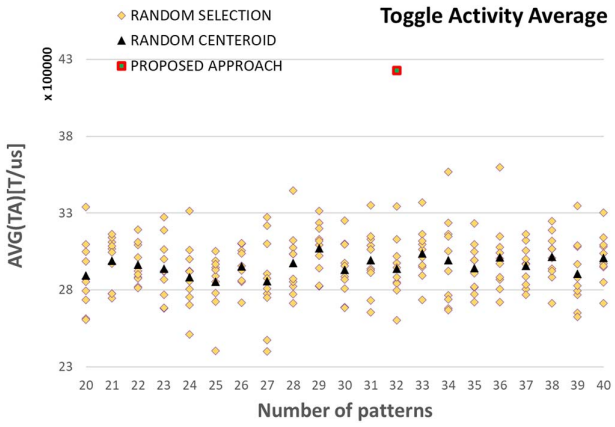


Fig. 14. $(TA_{average})$ comparison for DUT2.

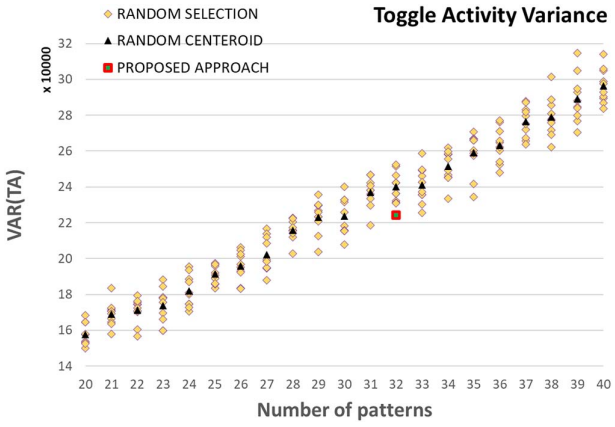


Fig. 15. $(TA_{variance})$ comparison for DUT2.

suite with 32 scan-based patterns, (b) the random selection which includes 32 functional patterns, and (c) the generated functional stress program.

D. An Additional Case Study

To further demonstrate the effectiveness of the proposed methodology, an additional case study is introduced. This case involves another automotive SoC manufactured by STMicroelectronics, belonging to the same chip family as the first case of study. This SoC targets low-power applications, with a more complex structure including additional CPUs and peripheral cores. It includes the same AI HW accelerator as the first case study SoC; the design is differently synthesized, using a different optimization and undergoing different timing closure and DfT optimization steps than the first one. Hereinafter, this device is referred to as DUT2. Since both devices belong to the same family, they share the same package form factor, allowing the second case study to be accommodated on the same evaluation board as shown in Fig. 6(b). The debugger running on

TABLE IV
FUNCTIONAL PATTERNS DETAILS FOR DUT2

Class	Amount of Randomly Distributed Values '1'	Number of Individuals	HIGHEST Current Consumption [mA]	AVERAGE Current Consumption [mA]
A	50%	200	26.01	19.33
B	40%	200	25.00	18.35
C	30%	100	22.55	16.08
D	20%	100	21.00	12.55
E	10%	100	13.07	8.48
F	0%-100%	5	6.03	5.18
G	provoking errors	20	2.02	0.91

TABLE V
FUNCTIONAL STRESS PATTERN SELECTION FOR DUT2

Ranked Functional Patterns	Class	Current Consumption [mA]	Incremental		
			MTC [%]	MTA Average	MTA Variance
1	A	26.01	-	-	-
2	B	25.00	25.0077	0.50	0.50
3	A	24.06	49.9685	1.00	0.71
4	A	24.01	68.6957	1.50	0.87
5	B	24.00	81.1838	2.00	1.00
6	B	24.00	88.4701	2.46	1.12
7	B	24.00	92.8063	2.93	1.24
8	A	23.96	95.8177	3.43	1.34
9	A	23.68	97.5934	3.93	1.43
10	B	23.26	98.6066	4.43	1.51
11	A	23.22	99.2075	4.92	1.59
12	A	23.00	99.5468	5.42	1.67
13	A	22.98	99.7334	5.92	1.74
14	C	22.55	99.8486	6.42	1.81
15	A	22.01	99.9130	6.92	1.88
16	A	22.00	99.9569	7.42	1.94
17	C	22.00	99.9783	7.92	2.01
18	C	22.00	99.9864	8.31	2.07
19	C	22.00	99.9898	8.69	2.14
20	B	21.99	99.9933	9.13	2.21
21	A	21.97	99.9951	9.63	2.27
22	A	21.95	99.9965	10.13	2.32
23	A	21.90	99.9978	10.63	2.38
24	A	21.83	99.9984	11.13	2.43
25	B	21.48	99.9992	11.63	2.48
discarded	B	21.23	-	-	-
26	C	21.08	99.9994	12.06	2.53
27	B	21.06	99.9994	12.34	2.40
28	B	21.00	99.9996	12.80	2.43
discarded	C	21.00	-	-	-
29	A	21.00	99.9998	13.30	2.48
30	A	21.00	**100**	**13.80**	**2.53**
31	G	1.86	100	14.03	2.39
32	G	1.13	100	14.29	2.27

TABLE VI
PATTERN APPROACHES COMPARISON FOR DUT2

Number of Patterns	Generation Methodology	TC[%]*	AVG(TA) [T/us]	VAR(TA)
40	Random Selection	97.01	2679626	296403
39		96.95	2634621	289057
38		97.24	2600884	278947
37		96.77	2597179	276297
36		96.94	2567170	263147
35		96.79	2554242	258912
34		96.81	2636314	251348
33		96.88	2593437	241087
32		96.79	2650194	240249
32	**Proposed**	**96.26**	**4228201**	**224234**
31	Random Selection	96.71	2598327	237056
30		96.93	2720859	223581
29		96.17	2638669	223196
28		96.60	2646928	215920
27		96.92	2583732	202084
26		96.70	2650416	195918
25		96.57	2733406	191445
24		96.63	2612034	181865
23		96.27	2656088	173847
22		96.37	2651801	171152
21		96.30	2709068	169113
20		96.01	2631578	157602

*for Random Selections it reports the average value over ten pattern sets.

the single-site FPGA-based tester [30] has been configured to access this additional device through its TAP controller. This additional example strengthens the experimental evidence, and it highlights the ease of porting the method to other platforms within the same chip family.

The same methodology and experimental parameters are retained to ensure consistency. Specifically, the number of averaged samples is set to $H = 140$, with a post-K standard deviation of $\sigma_K = 0.22\,\mathrm{mA}$ observed after $120\,\mathrm{s}$. The maximum recorded current value is $26.01\,\mathrm{mA}$. The Type A uncertainty results in a relative uncertainty of approximately 0.072%, thereby maintaining a high level of statistical confidence.

Table IV details the initial population characteristics, categorized by the proportion of randomly distributed logic '1's or by error-inducing conditions.

Table V presents results from the proposed method, where the proposed methodology selected 30 valid functional stress patterns for the stress suite. The table details each pattern's class, current consumption, and incremental MTC and MTA with averages and variances. Patterns causing errors are also included through a similar selection process focused on increasing average MTA or decreasing its variance.

Table VI presents a comparison between the proposed method, and random selections. It includes the number of selected functional patterns, and, for random selections, averages results from ten sets per pattern count. The table summarizes TC and TA values accordingly.

The proposed approach achieves a circuit node Toggle Coverage of 96.26%, slightly lower than the average 96.79% obtained using 32 functional patterns but comparable when using the same number of patterns. However, TA values from the proposed method are significantly better than those from random selection.

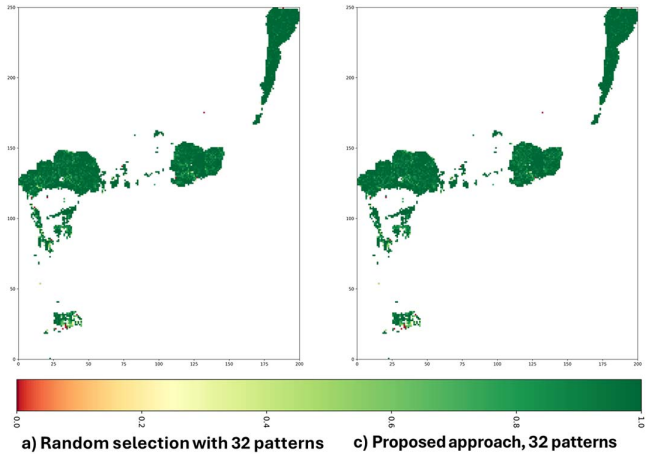


Fig. 16. Stress TC heatmap comparison for DUT2.

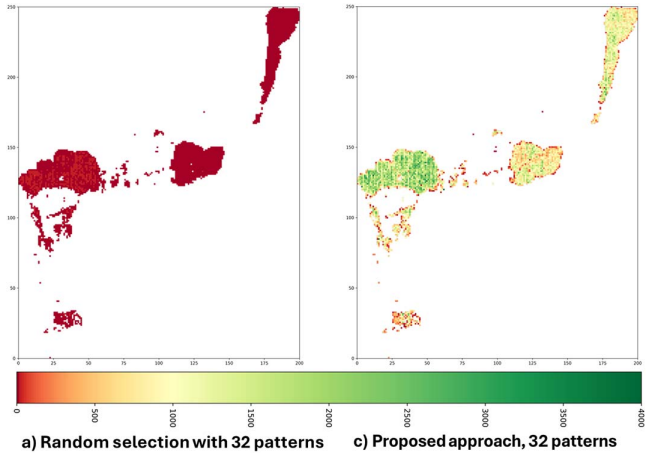


Fig. 17. Stress TA heatmap comparison for DUT2.

Figs. 16 and 17 show respectively the node coverage and activity generated over a layout heatmap by (a) the random selection which includes 32 functional patterns, and (c) the generated functional stress program.

V. CONCLUSION

This work shows that highly effective functional stress patterns can be selected without the necessity of using the circuit netlist. The heuristic approach of ranking and sifting two indirect metrics allows reaching a suboptimal solution within a reasonable amount of time and achieves better results compared to other netlist-based approaches, such as scan-based stress or evolutionary optimizers, in terms of toggle coverage and activity on the gate-level netlist. It is important to highlight that the proposed method does not require an expensive experimental setup, as it is based on a low-cost FPGA-based single-site tester [30] and an affordable discrete current sensor (approximately $5). Despite utilizing low-cost equipment, satisfactory results

were obtained, indicating that even better performance can be expected with higher-quality instruments.

Additionally, the proposed approach could be valuable in System-Level Test scenarios for companies that integrate obfuscated third-party IPs into their systems and want to ensure incoming quality by complementing existing scan-stress suites. Finally it can be used to complement structural pattern-set or be the starting point for a pattern generation.

ACKNOWLEDGMENT

The authors would like to thank **Giusy Iaria** for her incredible help on the additional case of study used to test and prove the proposed methodology.

REFERENCES

[1] B. Keller et al., "A 95.6-TOPS/W deep learning inference accelerator with per-vector scaled 4-bit quantization in 5 nm," *IEEE J. Solid-State Circuits*, vol. 58, no. 4, pp. 1129–1141, Apr. 2023.
[2] J. Zhuang et al., "Charm: Composing heterogeneous accelerators for matrix multiply on versal ACAP architecture," 2023, pp. 1129–1141. [Online]. Available: https://arxiv.org/abs/2301.02359
[3] N. P. Jouppi et al., "In-datacenter performance analysis of a tensor processing unit," in *Proc. 44th Annu. Int. Symp. Computer Archit.*, 2017, pp. 1–12.
[4] K. He et al., "Deep residual learning for image recognition," in *Proc. IEEE CVPR*, 2016, pp. 770–778.
[5] S. Kangralkar and R. Khanai, "Machine learning application for automotive emission prediction," in *Proc. IEEE I2CT*, 2021, pp. 1–5.
[6] P. Bernardi et al., "Improving stress quality for SoC using faster-than-at-speed execution of functional programs," in *VLSI-SoC*, ser. IFIP Advances in Information and Communication Technology, vol. AICT-508. Tallinn, Estonia: Springer Int. Publishing, Sep. 2016, pp. 130–151.
[7] F. Angione et al., "An optimized burn-in stress flow targeting interconnections logic to embedded memories in automotive systems-on-chip," in *Proc. IEEE ETS*, May 2022, pp. 1–6.
[8] M. Campbell, "Plenary presentations: Keynote: The product complexity and test — How product complexity impacts test industry," in *Proc. 15th IEEE Eur. Test Symp.*, 2010, p. 9.
[9] D. Tille et al., "Towards an automated flow for implementation of dedicated LBIST scan chains for functional safety," in *Proc. TUZ*, 2020.
[10] F. Angione et al., "Automatic generation of system-level test for un-core logic of large automotive SoC," *IEEE Trans. Comput.*, vol. 79, no. 9, pp. 3195–3209, Sep. 2025.
[11] F. Corno et al., "Efficient machine-code test-program induction," in *Proc. CEC*, vol. 2, May 2002, pp. 1486–1491.
[12] V. Turco et al., "Uncovering hidden vulnerabilities in CNNs through evolutionary-based image test libraries," in *Proc. IEEE DFT,* 2023, pp. 1–6.
[13] F. Angione et al., "A low-cost burn-in tester architecture to supply effective electrical stress," *IEEE Trans. Comput.*, vol. 72, no. 5, pp. 1447–1459, May 2023.
[14] C. He et al., "Wafer level stress: Enabling zero defect quality for automotive microcontrollers without package burn-in," in *Proc. IEEE ITC*, 2020, pp. 1–10.
[15] D. Appello et al., "System-in-package testing: Problems and solutions," *IEEE Des. Test Comput.*, vol. 23, no. 3, pp. 203–211, May/Jun. 2006.
[16] A. Benso et al., "ATPG for dynamic burn-in test in full-scan circuits," in *Proc. IEEE ATS*, 2006, pp. 75–82.
[17] H. H. Chen, "Beyond structural test, the rising need for system-level test," in *Proc. VLSI-DAT*, 2018, pp. 1–4.
[18] D. Appello et al., "System-level test: State of the art and challenges," in *Proc. IEEE IOLTS,* 2021, pp. 1–7.
[19] I. Polian et al., "Exploring the mysteries of system-level test," in *Proc. IEEE ATS*, 2020, pp. 1–6.
[20] F. Angione et al., "A toolchain to quantify burn-in stress effectiveness on large automotive system-on-chips," *IEEE Access*, vol. 11, pp. 105655–105676, 2023.
[21] D. Appello et al., "A comprehensive methodology for stress procedures evaluation and comparison for burn-in of automotive SoC," in *Proc. IEEE DATE*, 2017, pp. 646–649.
[22] R. Kawahara et al., "The effectiveness of IDDQ and high voltage stress for burn-in elimination [CMOS Production]," in *Proc. IEEE Int. Workshop IDDQ Testing*, 1996, pp. 9–13.
[23] M. d. Carvalho et al., "An enhanced strategy for functional stress pattern generation for system-on-chip reliability characterization," in *Proc. MTV*, 2010, pp. 29–34.
[24] D. Schwachhofer et al., "Optimizing system-level test program generation via genetic programming," in *Proc. IEEE ETS*, 2024, pp. 1–4.
[25] L. D. Abbati et al., "Industrial best practice: Cases of study by automotive chip- makers," in *Proc. IEEE DFT*, 2021, pp. 1–6.
[26] K. DeVogeleer, G. Memmi, P. Jouvelot, and F. Coelho, "Modeling the temperature bias of power consumption for nanometer-scale CPUs in application processors," in *Proc. SAMOS XIV*, 2014, pp. 172–180.
[27] C. Guo et al., "A survey of energy consumption measurement in embedded systems," *IEEE Access*, vol. 9, pp. 60516–60530, 2021.
[28] D. Schwachhofer et al., "Automating greybox system-level test generation," in *Proc. IEEE ETS*, 2023, pp. 1–4.
[29] STMicrolectronics. *RM0391 Reference Manual*. (2024). [Online]. Available: https://www.st.com/resource/en/reference_manual/rm0391-spc58-eg-line--32-bit-power-architecture-automotive-mcu-triple-z4-cores-180-mhz-6-mbytes-flash-hsm-asild-stmicroelectronics.pdf
[30] N. di Gruttola Giardino et al., "A flexible FPGA-based test equipment for enabling out-of-production manufacturing test flow of digital systems," in *Proc. IEEE DFT*, 2024, pp. 1–6.



Gabriele Filipponi (Student Member, IEEE) is a currently working toward the third year Ph.D. degree in computer and control engineering with Politecnico di Torino. He is a part of the Electronic CAD & Reliability Group, with a focus on Embedded electronics testing. His field of research regards ultra reliability for in-field automotive SoC, specifically exploiting the LBIST.



Denis Schwachhofer (Student Member, IEEE) received the B.Sc. and M.Sc. degrees in computer science, in 2018 and 2021, respectively, from the University of Stuttgart, where he is currently working toward the Ph.D. degree. He is a member of the Graduate School "Intelligent Methods for Test and Reliability" (GS-IMTR). His research interests include generating test programs for system-level test and finding the best method between fuzzing, genetic programming, and LLMs.



Francesco Angione (Member, IEEE) received the M.Sc. degree in embedded systems track, in 2020, and the Ph.D. degree in system-level-test techniques for automotive SoCs from CAD & Reliability Group, Politecnico di Torino, in 2025. He is currently a Research Fellow with the same institution. His research interests include real-time operating systems, computer architectures, and their dependability.



Claudia Bertani received the graduate degree in electronic engineering from Politecnico di Milano. She is a Product and Test Engineer Manager with STMicroelectronics. She is 20+ years of expertise in the semiconductor industry, high volume products ramp-up and management, semiconductor manufacturing flow, DfT and testability analysis, ATE, new product introduction technical support, and team management. She is currently managing the team in charge of system-level test introduction on automotive ADAS SOC products.



Simone Corbellini (Senior Member, IEEE) received the M.S. degree in electronic engineering and the Ph.D. degree in metrology from Politecnico di Torino, Turin, Italy, in 2002 and 2006, respectively. He is an Assistant Professor with Politecnico di Torino. He is involved in the development of measuring systems based on the accurate determination of resonances in microwave resonators. His research interests include DSP, distributed measurement systems, and wireless sensor networks.



Nicola di Gruttola Giardino (Student Member, IEEE) received the M.Sc. degree in computer engineering, embedded systems track from Politecnico di Torino, in 2024. He is currently working toward the Ph.D. degree in aerospace engineering. His main interests are digital design, real-time operating systems, and firmware development. He is part of the Electronic CAD & Reliability Group, as well as the STAR Team.



Giuseppe Garozzo was born in Catania, Italy, in Dec. 1972. He received the M.S. and Ph.D. degrees in physics from the University of Catania, in 1996 and 2001, respectively. From 1998, he is employed in ST-Microelectronics and works on data analysis and management. He is the author of some papers on plasma process simulation and statistical process control.



Giorgio Insinga (Member, IEEE) received the Ph.D. degree from Politecnico di Torino, in 2025. He is a Research Fellow with Politecnico di Torino. His research centers on the test and reliability of automotive systems-on-chips (SoCs) and their embedded memories. He also investigates innovative applications of augmented reality to aid in debugging PCB prototypes and repairing PCBs.



Vincenzo Tancorre received the B.S. degree in electronics engineering from Politecnico di Bari. He is a Yield Enhancement Engineer with the Automotive Group, STMicroelectronics. His research interests include test-related process monitoring using diagnostic solutions for memories and unstructured logic based on DfT methodologies.



Paolo Bernardi (Senior Member, IEEE) is an Associate Professor with Politecnico di Torino University, where he works in the Electronic CAD and Reliability Research Group. His interests include system-on-chip tests and reliability. He is the Program Chair of the Automotive Reliability, Test and Safety (ARTS) Workshop and General Chair of the IEEE European Test Symposium 2023.