MacLeR: Machine Learning-based Run-Time Hardware Trojan Detection in Resource-Constrained IoT Edge Devices

***Abstract*—Traditional learning-based approaches for *run-time* Hardware Trojan detection require complex and expensive on-chip data acquisition frameworks, and thus incur high area and power overhead. To address these challenges, we propose to leverage the power correlation between the executing instructions of a microprocessor to establish a machine learning-based run-time Hardware Trojan (HT) detection framework, called MacLeR. To reduce the overhead of data acquisition, we propose a single power-port current acquisition block using current sensors in time-division multiplexing, which increases accuracy while incurring reduced area overhead. We have implemented a practical solution by analyzing multiple HT benchmarks inserted in the RTL of a system-on-chip (SoC) consisting of four LEON3 processors integrated with other IPs like vga_lcd, RSA, AES, Ethernet, and memory controllers. Our experimental results show that compared to state-of-the-art HT detection techniques, MacLeR achieves 10% better HT detection accuracy (i.e., 96.256%) while incurring a 7x reduction in area and power overhead (i.e., 0.025% of the area of the SoC and $< 0.07\%$ of the power of the SoC). In addition, we also analyze the impact of process variation and aging on the extracted power profiles and the HT detection accuracy of MacLeR. Our analysis shows that variations in fine-grained power profiles due to the HTs are significantly higher compared to the variations in fine-grained power profiles caused by the process variations (PV) and aging effects. Moreover, our analysis demonstrates that, on average, the HT detection accuracy drop in MacLeR is less than 1% and 9% when considering only PV and PV with worst-case aging, respectively, which is ≈10x less than in the case of the state-of-the-art ML-based HT detection technique.**

I. Introduction

Globalization in an integrated circuit (IC) design-process has exponentially increased the trend to outsource fabrication, which makes the IC designs vulnerable to security threats like Hardware Trojans (HT) [1]. These HTs can change the system functionality (i.e., the functional Trojans), reduce reliability, increase the likelihood of system failure, modify physical parameters [2][3] like power consumption, accelerate aging factor [4][5], or contribute to information leakage via side channels (i.e., the parametric Trojans). These consequences can have severe and long-lasting effects on the credibility of hardware, hence, making it imperative to develop efficient HT detection techniques.

State-of-the-art HT detection techniques utilize side-channel parameters, i.e., timing [6]–[8], power [9][10], current or electromagnetic signals [11][12], based on the *golden signatures* to detect an anomalous behavior [13]. However, *in the case of third-party-IP based designs, it is nearly impossible to extract the golden signatures because IPs can already be un-trusted*. To address this issue, various IP analysis-based approaches [14]–[16] have been proposed, but they inherently pose the following limitations:

1) A limited access to the IPs and measurement inaccuracies can compromise the accuracy of the golden signatures.
2) Reverse engineering-based techniques are costly, and the existing sensors-based techniques *cannot encompass all the possible input conditions* because of the inherent quantization loss of analog-to-digital conversion (ADC).

To address the above-mentioned limitations, different Machine Learning (ML)-based techniques [9][17]–[19] have been proposed that train the ML models for communication patterns or power profiles. However, these techniques either possess a large overhead of ML computations or a large overhead of run-time data acquisition, both of which would be infeasible in resource-constrained edge devices, especially under environmental and process variations. Therefore, these issues raise a key research question: *how to enable a lightweight ML-based HT detection technique, and consequently, what is the associated run-time data acquisition overhead and the sensitivity to the process variations?*

A. Motivational Case Study and Key Observations

To address the above question, we propose an alternative approach that exploits the interaction between the trusted and un-trusted IPs in an SoC to extract the corresponding anomalous power profile that can be leveraged to design a low-overhead ML-based HT detection technique. In the context of IoT edge/embedded devices with a shared power distribution network for multiple cores [20], *we postulate a hypothesis, "the activity in an un-trusted IP, interacting with the trusted IP, can have a detectable impact on the power consumption of the trusted IP"*.

To corroborate this hypothesis, we perform a proof-of-concept case study (see Fig. 1) using four MC8051 IPs that are connected with multiple IPs (i.e., vga_lcd, AES, Ethernet, RS232, memory controllers) in an SoC, and exchange the data with each other to execute a particular set of workloads. Among these MC8051 IPs, at least one MC8051 IP is considered trustworthy, while any of the other IPs can be intruded with (open-source) Trust-Hub HT benchmarks [21] like AES-T100, AES-T800, vga_lcd-T100, RS232-T1000, memctrl-T100, and ethernetMAC10GE-T700. In these experiments, we monitor the power consumption of the trusted MC8051 to identify the change in power consumption of each instruction in different pipeline components.

In our experimental results given in Fig. 1, the top-row shows fine-grained power profiles of the trusted MC8051 microcontroller while executing the instructions (i.e., MOV, ADD, INC) for eight different test scenarios. Each scenario is associated with the activation of a single HT except in the scenario *S0* (i.e., there is no active HT in any IP). The bottom-row shows the change in power consumption (δP) in different scenarios w.r.t. the power consumption in scenario *S0*. **From this analysis, we make the following observations:**

1) The δP in scenario *S6* is a lot higher than the δP in other scenarios (see Labels L1 and L2). The reason behind this is that in *S6*, the HT activity (i.e., constant information leakage) is higher than other scenarios. Similarly, some of the scenarios show smaller δP values. However, δP depends upon instruction. For example, in scenario *S3*, the value of δP is $< 2\%$ for the MOV instruction (see Label L4), but for the INC instruction δP is $\approx 5\%$.
2) The change in power consumption δP also varies in different pipeline stages. For example, in all instructions and all scenarios, the value δP in the Fetch pipeline stage is negligible. However, this value can be higher in unknown HTs. The value of δP in the Decode pipeline stage for MOV is higher (see labels L1 and), but the value of δP in the Decode pipeline stage for ADD is negligible. *Therefore, for complete coverage of the change in power consumption (δP)*

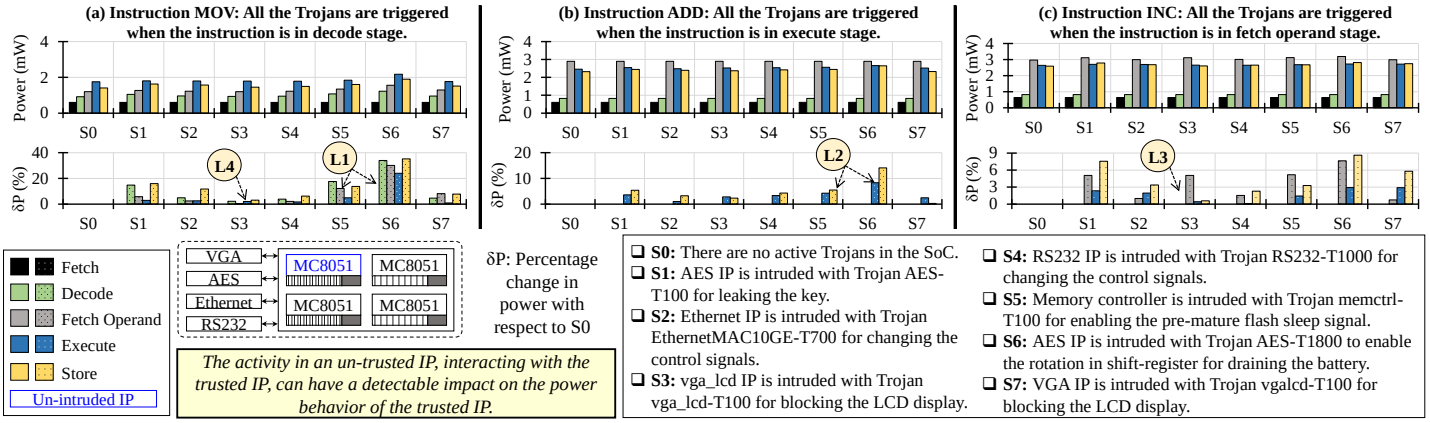


Fig. 1: Effects of Trust-Hub HT benchmarks (i.e., AES-T100, AES-T800, vga_lcd-T100, RS232-T1000, memctrl-T100, and ethernetMAC10GE-T700) on the power consumption w.r.t. different pipeline stages of an SoC consisting of four MC8051, one vga_lcd, one AES, one Ethernet, one RS232, four memory controllers IPs for different instructions, i.e., MOV, ADD, and INC. Note: these results are generated by synthesizing the MC8051 (RTL in Verilog) using the Xilinx toolchain for the Spartan 3(xc3s-1500) FPGA, as we consider resource-constrained IoT edge devices. $\delta P = \frac{|P_{S0}|-|P_{Sx}|}{|P_{S0}|}$, where P_{S0} and P_{Sx} represent the power consumption in scenario S0 and scenario Sx ($x \in \{1,2,3,4,5,6,7\}$), respectively. The workloads used for MC8051 are 32-bit encrypted multiplication, subtraction, addition, and division. The inputs are encrypted using AES, and the results are displayed on the screen using VGA as well as transmitted using Ethernet and RS232 IPs.

due to a HT, we need to consider the individual power profiles of each pipeline stage.

3) In all the scenarios, the values of δP are large enough and can be detected[1] using state-of-the-art run-time power analysis [22]. For example, in all the scenarios, the values of δP vary from 1% to 35%.

In summary, these observations strongly indicate that the extracted fine-grained power profiles can be leveraged for run-time ML-based HT detection.

B. Associated Research Challenges

The classification of the above-discussed power profiles and monitoring them during run-time, design time, or even during testing leads to the following research challenges:

1) In a contemporary SoC, for resource-constrained IoT edge devices, fine-grained power analysis for an n-stage pipeline is not straightforward as it involves a lot of dependencies. This raises a key question about how to extract the distinguishing power profiles of instructions at the granularity of different pipeline stages with a minimum overhead?
2) How to exploit these diverse fine-grained power profiles of different instructions (or instruction types) to develop a lightweight ML-based run-time HT detection technique while keeping the complexity and area overhead minimal?
3) Would the fine-grained power-analysis still be useful to accurately detect HTs at run time under the process variations and aging effects?

C. Novel Contributions and Concept Overview

To address the above research questions, we propose a novel methodology, called MacLeR, to design an ML-based run-time HT detection technique that exploits the fine-grained power profiling of the microprocessor (see Fig. 2). Towards this, MacLeR employs the following analysis and methods

1) To obtain the fine-grained power profiles, we propose to measure the individual power of each pipeline stage w.r.t. a particular instruction (see **Section V**). The reason for choosing this method is that the impact of HTs on fine-grained power profiles is relatively more noticeable as compared to the overall power.
2) To reduce the complexity and detection time, we propose an off-chip monitor that collects analog power profiles and converts them into the digital domain (see **Section VI**). These power profiles are then used first for training an ML model (in our case it is a lightweight multi-layer perceptron (MLP)) at design time, and afterwards at run time for detecting HTs. The reason for choosing an MLP is because it requires fewer computations and is typically faster than other complex ML algorithms.
3) Extracting the fine-gained power profiles of a microprocessor during run time requires multiple power ports. Therefore, to reduce the number of power ports, we propose a single power-port current acquisition block (SP-CAB) and accordingly measure the current in a time-division multiplexing manner (see **Section V**).
4) To study the robustness of MacLeR (i.e., drop in HT detection accuracy), we perform a sensitivity analysis under the process variation by performing the Monte-Carlo simulation using the PV models from TSMC 65nm technology (see **Section VIII**).
5) To study the robustness of MacLeR (i.e., drop in HT detection accuracy), we also perform a sensitivity analysis under aging effects with and without different aging polices, i.e., Fast-Core-Age-First and balanced-aging profile (see **Section VIII-C**).
6) To illustrate the scalability and generalizability of the MacLeR, we evaluated MacLeR for SoC with un-trusted microprocessor IPs and non-processor SoC (see **Section IX**).

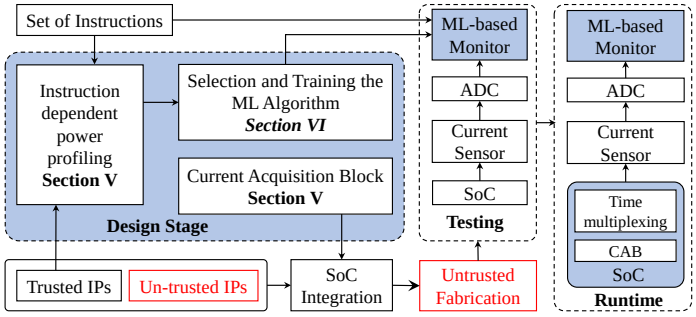


Fig. 2: Design, test, and run-time flow of our methodology for ML-based HT detection technique (MacLeR). Highlighted boxes represent the novel contributions. The evaluation/testing is done on a LEON3-based SoC.

[1]Note: The detectable change in power consumption depends upon the calibration of the current sensors and the corresponding analog-to-digital converter. For example, some state-of-the-art run-time power analyses cannot detect less than 5% change in power consumption [22].

Hardware Design: To evaluate our MacLeR framework with the above methods, we developed a LEON3-based SoC consisting of four LEON3 processor IPs integrated with one vga_lcd IP, one RS232 IP, one Ethernet IP, four memory controller IPs, one basic-RSA and one AES IP for multiple instructions using different ML-algorithms (see Section VII).

Key Results Compared to the state-of-the-art ML-based HT detection Technique: We analyzed our MacLeR on the above-mentioned SoC for multiple trust-Hub HT benchmarks and compared it with the state-of-the-art technique presented in [9]. Key results of these experiments are:

1) With only a single port power measuring block, our *MacLeR, using an MLP with two hidden layers and eight neurons per layer,* achieves HT detection accuracy of 96.256% that is ≈10% more than the maximum accuracy achieved by the technique presented in [9].
2) The area and power overheads of MacLeR are ≈7x less than the area and power overheads of the technique presented in [9]. MacLeR requires ≈ $7.01\mu m^2$ (≈ 0.15% of the area of the implemented SoC) and ≈ $15\mu W$ (< 1% of the power of the implemented SoC), respectively, when synthesized for 65nm technology using the Cadence Genus tool.

Sensitivity to the Process Variations and Aging: The side-channel parameters-based (like power consumption) HT detection techniques are vulnerable to process variations (PV) and aging. Therefore, we also analyze the impact of PV on the MacLeR by performing Monte-Carlo simulation using given PV models from TSMC 65nm technology. The aging variations based on the model presented in literature [23]–[25], i.e., change in operating frequency after Year-1, Year-2, Year-5 and Year-10. We evaluated MacLeR with no aging mitigation and for two aging policies, i.e., Fast-Core-Age-First and balanced-aging profile. Key results of this sensitivity analysis are:

1) In the microprocessor, the power variations due to HT are significantly higher compared to the PV-induced power variations or powaer variation due to aging effects.
2) The average drop in the HT detection accuracy of MacLeR is less than 1% and 9% when considering only PV and PV with worst-case aging, respectively, which is ≈10x less than in the case of the state-of-the-art ML-based HT detection technique.

II. Related Work

Typically, HT detection techniques employ *delay* [6]–[8], *power consumption* [9], [10], [13] and *operating frequency* [26][27] signature-based analysis. However, most of these techniques are payload-specific, i.e., they can only detect HTs during the design or testing phases, and require golden circuits. However, the intruder may hide HTs by *exploiting the aging behavior* [4][5] or *the switching activity* [3] of the chip. Detecting such HTs during the testing phase is very challenging, and the undetected HTs may get activated once the chip is in use. Run-time approaches, on the other hand, can monitor an IC for its entire operational lifetime, providing an important last-line of defense. Therefore, specialized techniques have been developed to detect HTs with specific payload at runtime, i.e., confidentiality [28], integrity [29] and availability [30] attacks. *The main drawback of such run-time techniques is that they incur large area and power overhead [31] and require precise calibration to cater to environmental changes and process variations.*

To address the above-mentioned limitations, ML-based techniques have emerged as a promising solution to detect the possibility of anomalies at run time, while exploiting the datasets obtained during the measurement phase for training [17]. *One of the major challenges in such techniques is generating the parametric or behavioral profile to train the ML tool.* Recently, a support vector machine (SVM) has been used to classify the intruded and un-intruded parametric behavior [18]. However, modeling and acquiring the dynamic behavior of SoC using SVM is computationally costly, e.g., it requires $m \times p$ multiplications and requires $p \times wordsize$, where m and p are the size of data and number of features, respectively. To cater to this problem, Kulkarni et al. [19] proposed to use other supervised ML online algorithms, i.e., k-NN and Modified Balanced Winnow (MBW) algorithm. However, these approaches assume that IP modules are not intruded and therefore, they are only applicable to availability attacks. The work in [9] proposed an on-chip power-based technique, which can detect HTs having direct or indirect effects on the power consumption of a microcontroller [9]. *However, this HT detection technique requires a large number of power-ports to extract the power behavior for ML training and ML inference, and is therefore infeasible to be deployed in real-world embedded systems*. Table I summarizes state-of-the-art ML-based HT detection approaches w.r.t. their features, as well as the orientation of our proposed MacLeR framework highlighting the key differences.

TABLE I: Comparison with state-of-the-art ML-based run-time HT detection (SVM: support Vector Machine, RE: Reverse Engineering, TMR: Triple Modular Redundancy, DT: Decision Tree,"n": Number of components involve in a pipeline operation, C: Confidentiality, I: Integrity, A: Availability, P: Power, D: Delay)

Techniques	Payloads			Parameters		ML Tools	On-Chip Overhead
	C	I	A	P	D		
[32], [33]			✓	✓	✓	SVM	SVM and RE overhead
[19]	✓	✓			✓	SVM	SVM Model and TMR
[9], [22]		✓	✓	✓		DT	"n" Power-ports and ADCs
MacLeR	✓	✓	✓	✓		MLP	One power-port, "n" current sensors, and Time Multiplexer

III. Threat Model

We assume that the *third-party IP (3PIP) vendors are not trustworthy*, and therefore, the specification and source code provided by the vendor may contain HTs; see Fig. 3.

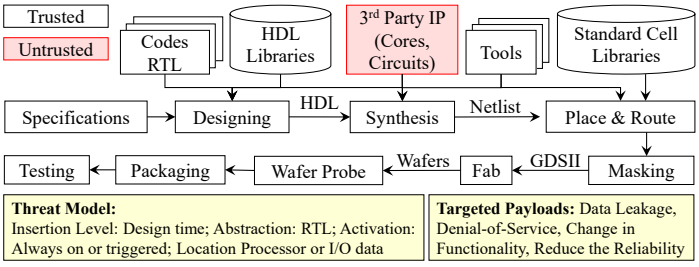


Fig. 3: Brief overview of the IC manufacturing along with the targeted threat model and the corresponding payloads.

The hardware designers/architects who integrate different 3PIPs, along with the in-house IPs (if any), to develop an SoC are considered to be the defenders. Note that among these IPs *at least one IP is trusted.* This paper targets HTs (see Fig. 3), which have a *direct or indirect impact on the shared power network between IPs of a SoC*. Although the multi-core SoCs can have multiple power grids and different voltage islands, the SoCs in battery-operated components for edge devices typically have only one power grid with multiple voltage islands (for instance, GAP-8 [20]), which is shared between different components. Therefore, in this work, *we design a low-power machine-learning-based run-time HT detection methodology for SoCs that have one power grid with multiple voltage islands.*

IV. MacLeR: ML-based Run-time Monitoring

Fig. 4 shows the complete step-by-step flow of McLeR, which consists of the following key steps:

1) The main goal of MacLeR is to design a low-power ML-based monitor for HT detection, for which a proper training dataset is required. Towards this, during the design phase, first, MacLeR generates power profiles by measuring the combined power consumption of each component involved in a particular pipeline stage. However, to generate the abnormal power profiles of a microprocessor for MLP training, we use the Trust-Hub HT benchmarks. Note, we use different sets of HT benchmarks for training and testing of MacLeR to avoid any kind of training bias.
2) During the design phase, it uses the generated power profiles to train different variants of MLPs. Then it performs a design space exploration (DSE) w.r.t. the detection accuracy and the associated overhead of the MLP, and it chooses the most appropriate MLP, which is used for run-time HT detection.
3) During the run time, MacLeR measures the power consumption of each component involved in a particular pipeline stage using multiple *current mirrors*[2], and then it collects the combined power using a single pMOS transistor. Then these *power values are collected in a time-division multiplexing manner* and used by the trained MLP (which is the best-one from the Step-2 of DSE) to detect HTs.

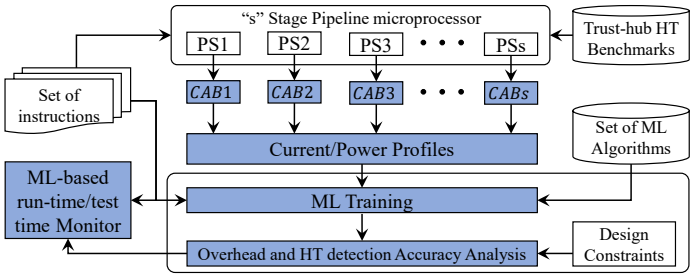


Fig. 4: MacLeR: ML-based methodology to run-time power monitoring. CABs represent the current acquisition blocks for s^{th} pipeline stages (PSs).

V. Fine-Grained Power Profiling

During design time, fine-grained power profiles are obtained by measuring the instruction-dependent power of each component associated with a pipeline stage. However, power acquisition during the run-time poses a research challenge about *how to acquire the fine-grained power profiles with a minimum area overhead*?

For this, we propose to use *multiple current mirrors* for measuring the current from each component and collect it using a single pMOS transistor-based collector, as shown in Fig 5. Since the current measuring accuracy is dependent on the sizes of the transistors, therefore, we computed these sizes using the following set of equations:

$$W'_{nsc_s} = SF_{nsc_s} \times W_{nsc_s} \quad (1)$$

$$SF_{nsc_s} = \frac{I_{nsc_s}}{max(I_{ns1}, I_{ns2}, ..., I_{nsc_s})} \quad (2)$$

Where W_{nsc_s} and W'_{nsc_s} are the widths of nMOS in the parent branch and corresponding mirror branches for c_sth component of the sth pipeline stage, respectively. In some cases, the width requirement exceeds the maximum allowed width. to address this, we introduce the scaling factor SF_{nsc_s} that normalizes the current and respective transistor width, to keep the width requirement within the limit, as shown in Equation 2. Similarly, using the model of [34], we computed the width of a single pMOS for a given component 'C', using the following equation:

$$W_{ps} = \frac{T_{wR} \times W_{n\_min}}{c} \; ; \; define \;\; T_{wR} = \frac{\mu_n}{\mu_p} \quad (3)$$

The single current sensor-based power-ports are used to measure the current of different modules involved in each pipeline stage. Typically, the number of power-ports required to cover all the modules is equal to the number of components involved in the operations of pipeline stages [9][22], e.g., in n-stage pipeline architecture, the number of power-ports is $N_p = C_1 + C_2 + ... + C_n$, where $C_1, C_2, ..., C_n$ are the number of components involved in pipeline operations of 1^{st}, 2^{nd}, ... , n^{th} pipeline stage, respectively. Having multiple power ports in an IC is very expansive for the IC packaging. Therefore, to acquire the complete power profile of the microprocessor via a single power-port, we propose to acquire the data using time multiplexing (which measures the current of each SP-CAB after every clock cycle), as shown in Fig. 6. Note, *MacLeR extracts and uses the fine-grained power profiles of a trusted microprocessor in an SoC. Therefore, it does not require any golden circuits of un-trusted IPs.*

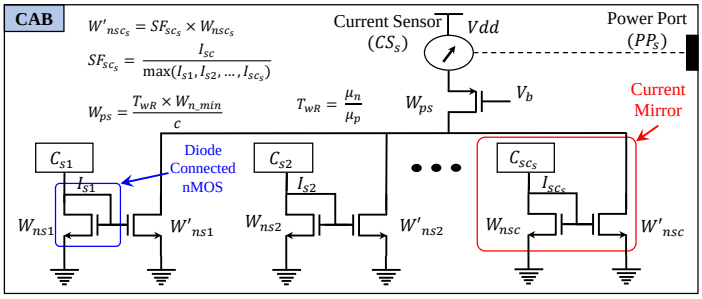


Fig. 5: The proposed single power-port current acquisition block (CAB).

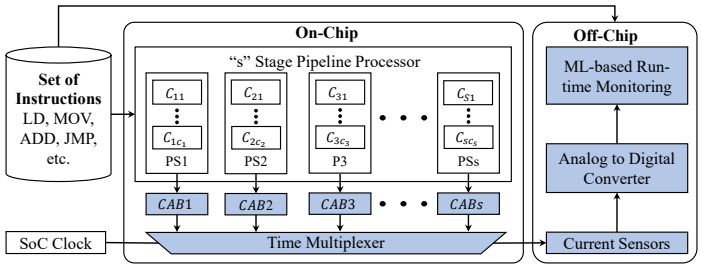


Fig. 6: Single power-port based hardware implementation of MacLeR with on-chip and off-chip hardware components.

VI. Training and Selection of An Efficient MLP Model

After acquiring the power profiles of the microprocessor, MacLeR chooses an appropriate ML algorithm based on the required HT detection accuracy and design constraints. Towards this end, we propose an iterative methodology that first trains the multiple configurations of MLPs using the following steps, as shown in Fig. 4.

1) We start by labeling different power-profiles to differentiate the intruded and un-intruded power profiles. These labelings are, in turn, used to train and validate the ML models. Note, during the design time, the abnormal power profiles are obtained using the trust-hub HT benchmarks.

[2]Current mirror is an analog circuit that copies the current of an active device using diode connected CMOS transistor. Typically, these circuits are used to sense the current of an active device.

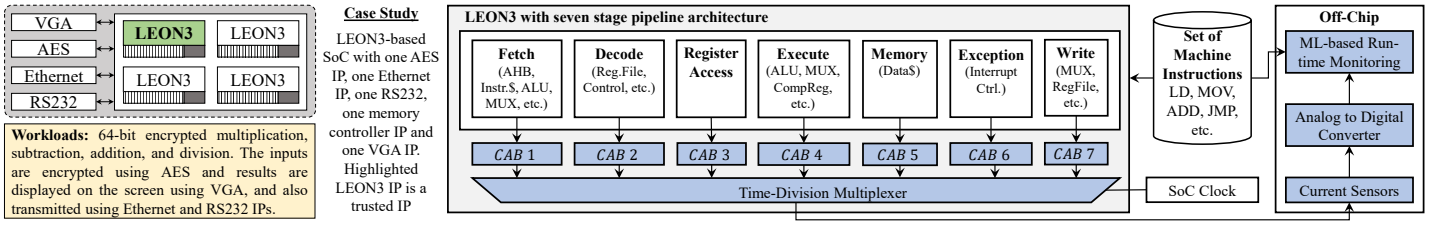


Fig. 7: Hardware implementation of the MacLeR monitoring framework for LEON3 with a seven-stage pipeline architecture.

2) Next, we categorize these power profiles w.r.t. the functional and behavioral similarity to increase the efficiency of the ML models.
3) After the categorization, we train the multiple ML models and validate them by applying the testing dataset.
4) Finally, MacLeR selects the best MLP model based on the maximum HT detection rate, associated overhead, and the given design constraints.

Note, MacLeR requires the instruction, its category, and power value, irrespective of the pipeline stage, as shown in Fig. 7. The main reason to choose the fine-grained power profiling at the pipeline stage is that MacLeR can explore multiple power values to expand its search space.

VII. Case Study: Employing MacLer to a LEON3-Based SoC

We illustrate the practicality, utilization, and effectiveness of our MacLer framework by applying it on a LEON3-based SoC, where at least one of the IP is trusted, as shown in Fig. 7. The main motivation of choosing LEON3 is that it is highly configurable and open-source, and the HT benchmarks that are provided by trusthub.org [21] can easily be integrated into it. The workloads used for LEON3 are 64-bit encrypted multiplication, subtraction, addition, and division. The inputs are encrypted using AES and results are displayed on the screen using VGA and also transmitted using Ethernet and RS232 IPs. Note, the addition, subtraction, and multiplication are used for training the ML monitor, and the division is used at the inference stage of the ML-monitor to detect the HTs, and therefore, avoiding the biasing in the testing phase.

A. LEON3: Power Profiling

For obtaining the power profile, we synthesized the LEON3 processor using Cadence Genus (Encounter) tool with the TSMC 65nm library. The power of each module involved in the pipeline stage is calculated separately for each instruction. For example, Fig. 8 shows the power consumption in each pipeline stage for a particular instruction, which is extracted by executing one instruction at a time. *It is very challenging and computationally expansive to cover all the possible combinations of instructions. Therefore, to reduce the overheads, we use the instruction categorization of SPARC V8 architecture [35],* i.e., the following five categories:

1) **Category 1 (Cat1):** *Load/store instructions* access memory and, use registers and a signed 13-bit immediate value to calculate a 32-bit, byte-aligned memory address, e.g., load (LD), load double (LDD) and load floating-point (LDF).
2) **Category 2 (Cat2):** *Arithmetic/logical/shift instructions* perform arithmetic, logical, and shift operations, e.g., subtract (SUB), add (ADD), multiply (ULMUL) and add with carry (ADCC).
3) **Category 3 (Cat3):** *Control-transfer instructions* perform PC-relative branches and calls, register-indirect jumps, and conditional traps, e.g., restore (RESTORE), call and link (CALL) and save (SAVE).
4) **Category 4 (Cat4):** *Read/Write Register instructions* read and write the contents of software visible state registers and processor registers, e.g., write to processor state register (WRPSR), write to y register (WRY) and read from y register (RDY).
5) **Category 5 (Cat5):** *Floating-point operate instructions* perform all floating-point operations, e.g., floating-point move (FMOV), floating-point operate (FCMP) and floating-point add (FADD).

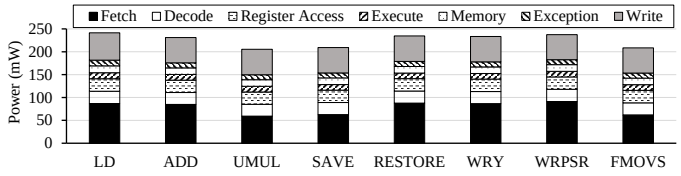


Fig. 8: Power Profiles of un-intruded LEON3 for different instructions

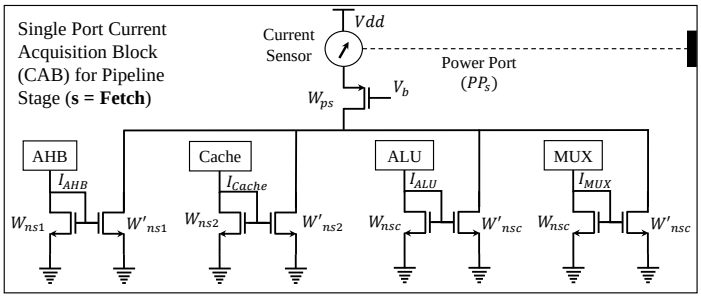


Fig. 9: Hardware Implementation of the single power-port current acquisition block for pipeline stage Fetch of the LEON3 processor.

Note, we assume all the required data is available in the on-chip memory, and thus, the difference in power profiles during cache hits or misses is not considered in this work. However, this behavior can be captured by exploiting the pipeline stall flags [36].

B. LEON3: Data Acquisition

The power consumption of a microprocessor is dependent upon the number of modules involved in the execution of instructions during a particular pipeline stage, e.g., LEON3 has seven different power profiles, i.e., one for each pipeline stage. To model the power behavior, first, we identify all the modules involved in the operation of a particular pipeline stage. Fig. 7 shows that in LEON3, the fetch stage requires instruction cache (I-Cache), AHB bus, adder, and multiplexers (MUX). Decode, register access, execute, memory and exception stages require register file, ALU, data cache (D-cache), and interrupt controller. In LEON3, the fetch stage consists of 4 components, which is the maximum number of components involved as compared to any other pipeline stage. To obtain the power profiles, we implemented the CAB for each pipeline stage of the LEON3, e.g., Fig. 9 shows the CAB for Fetch stage. To

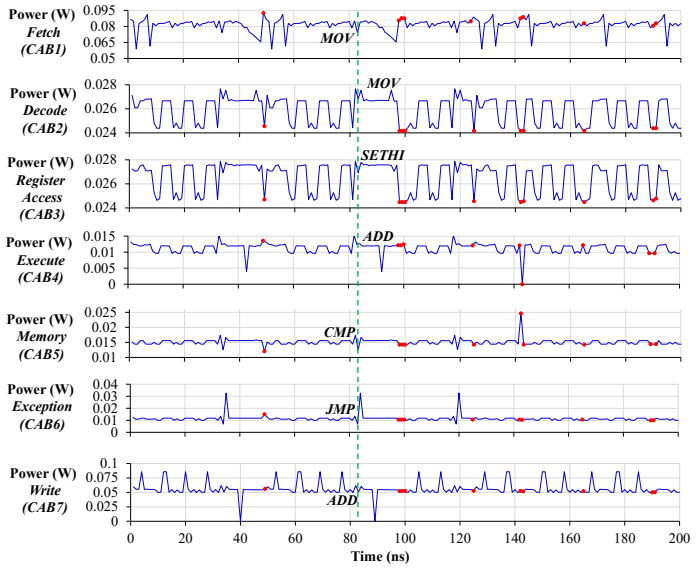


Fig. 10: Power consumption of seven pipeline stages of the LEON3 microprocessor while executing multiple instructions. The Red dots (●) in the figure represent the activations of MC8051-T200 in the SoC.

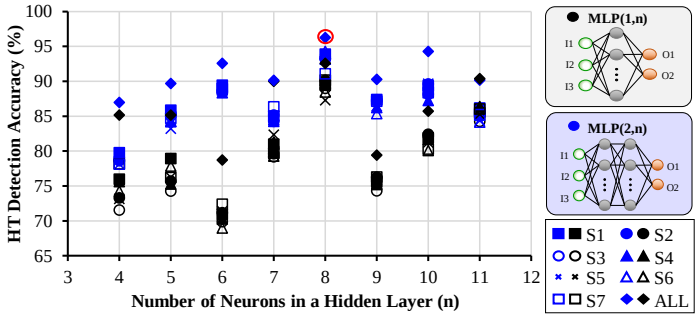


Fig. 11: Design space exploration of different configurations of implemented MLPs with one or two hidden layers. The blue color represents the MLP with two hidden layers, and the black color represents the MLP with one hidden layer. The inputs for MLP are I1: Power, I2: instruction, and I3: category, and output labels are O1: un-intruded and O2: Intruded.

generate the *anomalous power profiles*, we implemented the *trust-hub HT benchmarks*. In our experimental setup, AES-T100, AES-T800, vgalcd-T100, RS232-T1000, memctrl-T100, and ethernetMAC10GE-T700 benchmarks are used to generate the data for *training*, while the data generated using all the HT Trojan benchmarks for RS232, MC8051, AES, Basic RSA, VGA-LCD, memory controller and Ethernet IPs are used for *evaluation*.

Fig. 10 shows the power profiles of LEON3 in the presence of AES-T100 in SoC, and the red dots (●) show the triggering of AES-T100. The power value at each time unit, along with an instruction and its category, is extracted to generate the required power profile[3]. These profiles are then used to train the ML model in the next phase.

C. LEON3: Run-time Monitor for HT Detection

After extracting the power profile in the previous phase (see Fig.10), we analyze and transform it in such a way that it can be utilized for training and validation. This involves proper labeling of the extracted power behavior to differentiate between *intruded* and *un-intruded* behaviors, and pre-processing to reduce the redundant dataset. After the transformation, we trained different configurations of neural networks (multi-layer perceptrons (MLP) with one and two hidden layers, as shown in Fig. 11). Afterwards, we analyzed the trade-off between accuracy and computational cost as shown in Fig. 11. For a comprehensive analysis, we trained each configuration on the abnormal power profiles extracted from the different scenarios (explained in Fig. 1). For example, MLP (1,8) is trained for anomalous data (generated from the scenarios S1, S2, S3, S4, S5, S6, and S7 individually, and a combination of all scenarios) and the normal data (generated from the scenario S0). This analysis shows that the MLP with two hidden layers and eight neurons in each hidden layer provides maximum HT detection accuracy, i.e., 96.256% in our case study.

HT Detection Accuracy: To validate the extracted trained model, we randomly divide the labeled data into k-mutually exclusive subsets, where each subset is approximately of the same size, and performs the training and validation k times. In each iteration, one subset is used for validation, and the others are used in training. Thus, each subset is used for an equal number of times for training and once for validation. We also evaluated the trained network for unseen data generated using trust-hub HT benchmarks, as shown in Fig. 12.

The experimental analysis shows that in the case HT benchmarks for MC8051, the trained MLP with two hidden layers and eight neurons in one layer provides approximately 98% HT detection accuracy, and with a very small number of false positives and false negatives. However, for other HT benchmarks, MacLeR still provides approximately 90% HT detection accuracy. *Based on these observations, we conclude that the impact of HTs on shared power network, especially in multi-IP based SoC with at least one trusted microprocessor, can be detected by observing the fine-grained power profiles of the trusted processor.*

Moreover, for comprehensive analysis, we also compute the Mathews correlation coefficient, as shown in Fig. 12(d). MLP1(10, 100) and MLP2(50,100) using [9] gives up to 88% HT detection accuracy. However, the MCC analysis shows that MLPs trained using [9] are either randomly flipping the binary decision (because MCC is close to zero, see labels P10 and P11) or show the negative correlation. The reason for this is that number of false positives, and the number of false negatives is very large. Therefore, these MLPs cannot be considered as a good binary classifier. On the other hand, MCC values for an MLP trained using MacLeR go up 0.7 (see label P12), which is the property of a very good binary classifier.

VIII. Sensitivity Analysis of MacLeR under Process Variations

The side-channel parameters (like Power) based-HT detection techniques are vulnerable to environmental changes (e.g., temperature variations and measurement noise) and process variations (PV). To illustrate the variation-tolerance of MacLeR, we perform an analysis that consists of the following steps:

1) First, we perform the Monte-Carlo analysis using the given PV model from TSMC 65nm technology to generate power profiles with and without HT activation while using multiple workloads, i.e., the addition, subtraction, and multiplication. In this experiment, we perform 100 experiments for each workload with and without each benchmark Trojan.
2) Afterwards, we use these power profiles to train the MLP that is selected in Section VII-C based on the highest HT detection accuracy (see Fig.11). Then, the trained MLP is used to analyze the impact of PV on the HT detection accuracy.

[3]For example, at 1ns the power profile consists of the power of each pipeline stage is ([0.086649W, 0.027123W, 0.027238W, 0.013182W, 0.015111W, 0.012424W, 0.059794W], LD and Cat1).

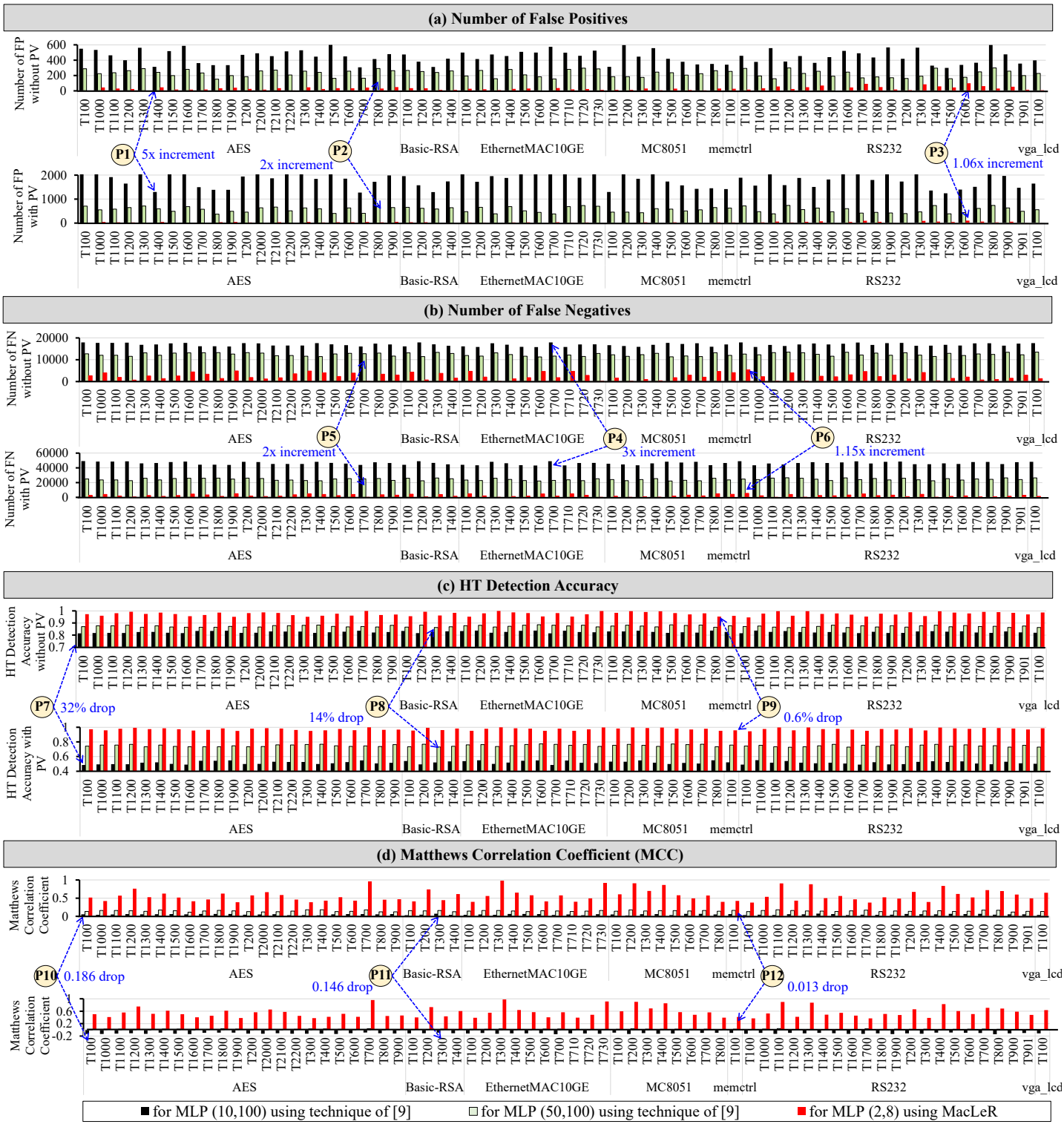


*Fig. 12: False positives, false negatives and HT detection accuracy of the MacLeR and the state-of-the-art run-time ML-based HT detection technique [9], in the presence of different HT benchmarks **with** and **without** considering the process variations. Note: these analyses are based on the 100,000 classification per HT, where the total number of activations is 1000 out of 100,000. In these experiments, all benchmarks related to MC8051 are implemented in the LEON3 microprocessor and all other HT benchmarks are also configured for the LEON3 microprocessor. In these experiments, the overall accuracy is computed as* $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$*, where TP, TN, FP, and FN represent true-positives, true-negatives, false-positives, and false-negatives, respectively. The Mathews Coefficient is computed using the standard formula,* $MCC = \frac{(TP\times TN)-(FP\times FN)}{\sqrt{(TP+FP)\times(TP+FN)\times(TN+FP)\times(TN+FN)}}$.

Note, we use the division as the workload for analyzing the HT detection accuracy of the trained MLP.

A. Impact of PV on Fine-Grained Power Profiles

The impact of process variations is not uniformly distributed for an SoC. Depending upon the fabrication conditions and variation in the fabrication process, it affects different components with different intensities. Therefore, to analyze the impact of process variation on fine-grained power profiles, we individually analyze the power profiles extracted from the CAB associated with each pipeline stage of the LEON3 microprocessor. For example, Figs. 13 show the impact of PV

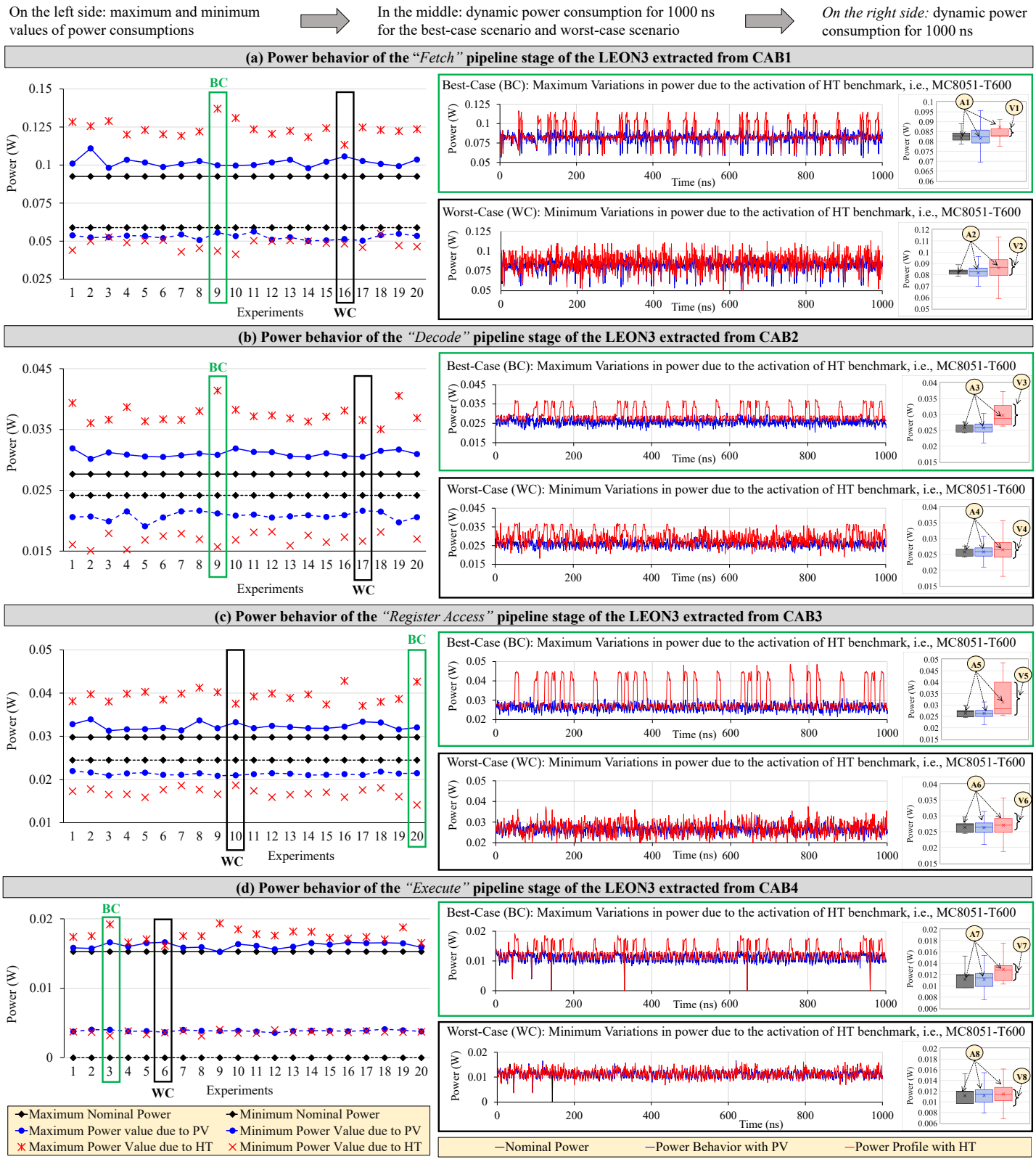


*Fig. 13: The impact of process variations and implemented hardware Trojan benchmark, i.e., MC8051-T600, on the power behavior of **Fetch, Decode, Register Access**, and **Execute** pipeline stages of LEON3 microprocessors while executing the multiple instructions. The results presented in black, blue and red colors represent the nominal power behavior, variations in power behavior due to PV, and variations in the power behavior due to HT, respectively. Note, in these analyses, Best-Case (BC) defines the scenarios in which HT is easily detectable, and Worst-Case (WC) defines the scenarios in which HT is hard to detect.*

on fine-grained power profiles of LEON3 microprocessor with and without HT benchmark, i.e., MC8051-T600[4]. From these analyses, we made the following observations:

1) The analysis presented in Fig. 13 (a) shows that in the presence of an active HT (i.e., MC8051-T600) the maximum power consumption of the "Fetch" pipeline stage

[4]For these analyses, we performed 100 experiments for each case with multiple HT benchmarks. However, due to limited space, we present analysis for 20 experiments for each case with one HT benchmark, i.e., MC8051-T600.

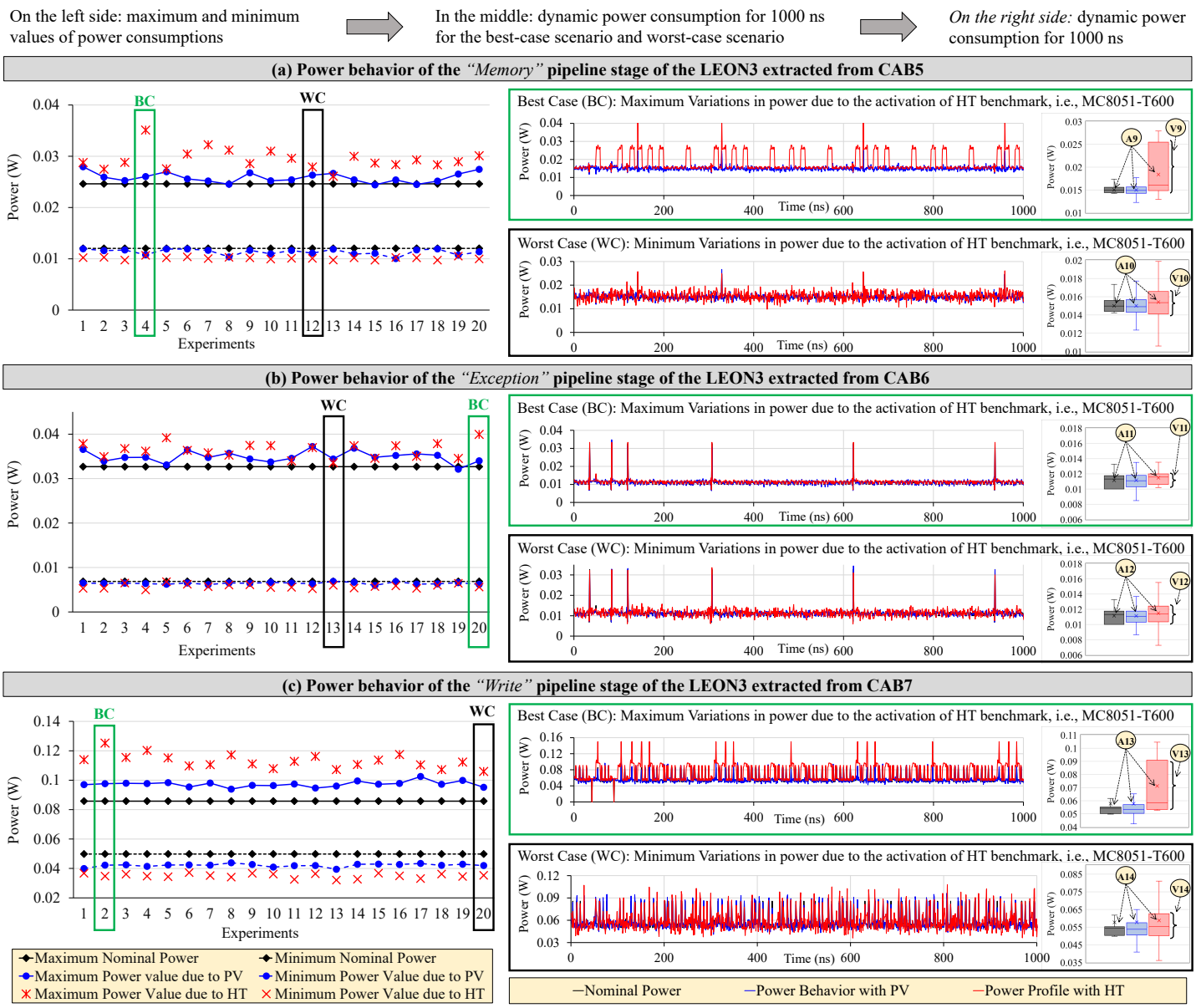


*Fig. 14: The impact of process variations and implemented hardware Trojan benchmark, i.e., MC8051-T600, on the power behavior of **Memory, Exception**, and **Write** pipeline stages of LEON3 microprocessors while executing the multiple instructions. The results presented in black, blue and red colors represent the nominal power behavior, variations in power behavior due to PV, and variations in the power behavior due to HT, respectively. Note, in these analyses, Best-Case (BC) defines the scenarios in which HT is easily detectable, and Worst-Case (WC) defines the scenarios in which HT is hard to detect.*

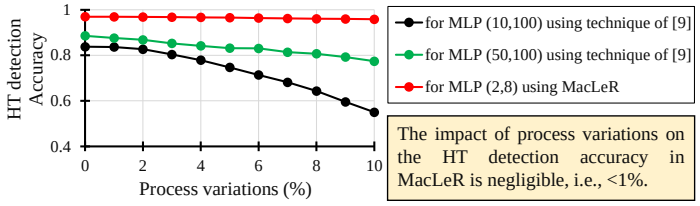


Fig. 15: Impact of different range of the process variations on the average detection accuracy. Note, this analysis is performed for the MC8051-T600.

is significantly larger than the PV boundaries[5]. For example, the best-case (Experiment 9) analysis shows that the spread of power consumption in the case of MC8051-T600 is smaller (see label V1) but is shifted towards the higher power consumption (see Label A1). This behavior shows the additive nature of MC8051-T600. However, in some cases, the variations are not significant. For example, in the worst-case (Experiment 16) analysis, the variations in power are smaller, and the mean value of power consumption in the presence of MC8051-T600 is within the PV boundaries (see label A2).

2) Fig. 13 (b) shows a similar trend for the "Decode" pipeline stage. However, variations in power consumption due to MC8051-T600 are even larger than the variations in power consumption of the "Fetch" pipeline. For example, in the best-case (Experiment 9) analysis, the power spread is large (see label V3), and it completely lies outside the PV boundaries.
3) Similar trends are observed for the other pipeline stages of the LEON3 microprocessor, i.e., "Register Access" in Fig. 13 (c)", "Execute" in Fig. 13 (d), "Memory" in Fig. 14 (a) and "Write" in Fig. 14 (c). However, the power behavior of the pipeline stage "Exception" is hardly affected by MC8051-T600. For example, the best-case analysis (see labels A11 and V11) and the worst-case analysis (see labels A12 and

[5]It is defined as the absolute difference between the maximum (P_{max}) and minimum (P_{min}) values of power consumption in the presence of PV, $|B_{PV}| = |P_{max}| - |P_{min}|$.

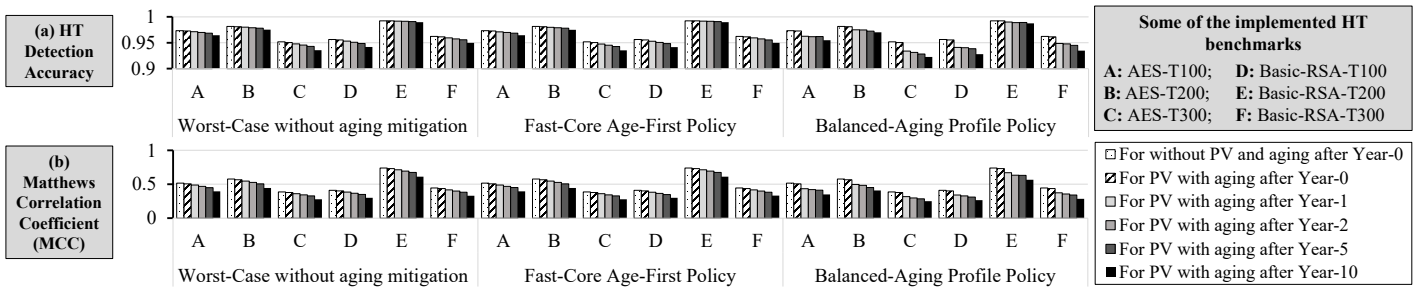


Fig. 16: Performance analysis of MacLeR for different HT benchmark under different PV and aging variations, i.e., change in operating frequency after Year-1, Year-2, Year-5, and Year-10.

V12) in Fig. 14 (b) shows that in most of the experiments, power variations due to MC8051-T600 are within the PV boundaries. The reason behind this is that MC8051-T600 does not get triggered when the instructions are inside the exception pipeline stage. Moreover, in most of the workloads, this pipeline stage remains dormant.

In summary, on average, the HT detection accuracy drops in MLP1(10, 100) and MLP2(50,100) using [9] are ≈ 29% (83.75% to 55.002%) and ≈ 11% (88.56% to 77.401%), respectively. However, on average, the HT detection accuracy drop in MacLeR (subjected to PV) is less than 1% (96.256% to 95.85%) compared to the case without PV consideration.

Based on the above-mentioned analyses, we conclude that in most cases, variations in fine-grained power profiles of the microprocessor due to HT are significant enough to detect the HT, hence captured by our MacLeR. Moreover, depending upon the applications, the frequency and significance of the pipeline stage can be adjusted to increase the HT detection accuracy.

B. Impact of PV on HT Detection Accuracy

To analyze the tolerance of MacLeR against PV, we generated the power profiles for different ranges of process variations, i.e, 1% to 10%, and analyze the MLP trained using MacLeR, as shown in Fig. 15. *This analysis shows that the drop in HT detection accuracy of MLP trained using MacLeR is negligible.* However, the drop in HT detection accuracy of MLPs that are trained using the power profiling technique in [9] is significant. *The reason behind this is that the stat-of-the-art technique [9] does not consider the correlation between the instructions with power profiles of microprocessor w.r.t. different pipelines stages.*

To further illustrate the effectiveness of the MacLeR, we analyze the MLP that is trained using MacLeR for various HT benchmarks from trust-Hub [21] in the presence of 10% PV. Fig. 12 shows the impact of 10% PV on the number of false positives, false negatives, and the HT detection accuracy of the trained MLP. From this analysis, we made the following observations:

1) For MLPs that are trained using the power profiling technique in [9], in the worst case, the number of false positives is increased by 5x (see label P1 in Fig. 12 (a)) and 2x (see label P2 in Fig. 12 (a)). However, on average, the number of false positives is increased from 500 (without considering the PV) to 1500 (when considering the PV). On the other hand, the increment in false positives for MacLeR is almost negligible, i.e., 1.06x, see label P3 in Fig. 12 (a). A similar trend can be observed for increment in the number of false negatives, i.e., 5x increment as shown by label P4 in Fig. 12 (b), 2x increment as shown by label P5 in Fig. 12 (b), and 1.15x increment as shown by label P6 in Fig. 12 (b).
2) Similar to the analysis, presented in Fig. 15, in the worst case, the HT detection accuracy MLPs that are trained using the power profiling technique in [9] is 32% (see label P7 in Fig. 12 (c)) and 14% (see label P8 in Fig. 12 (c)), respectively. On the other hand, the worst-case drop in the HT detection accuracy for MacLeR is negligible, i.e., 0.6% as shown by label P9 in Fig. 12 (c). In short, on average, the drop in HT detection accuracy is also negligible.

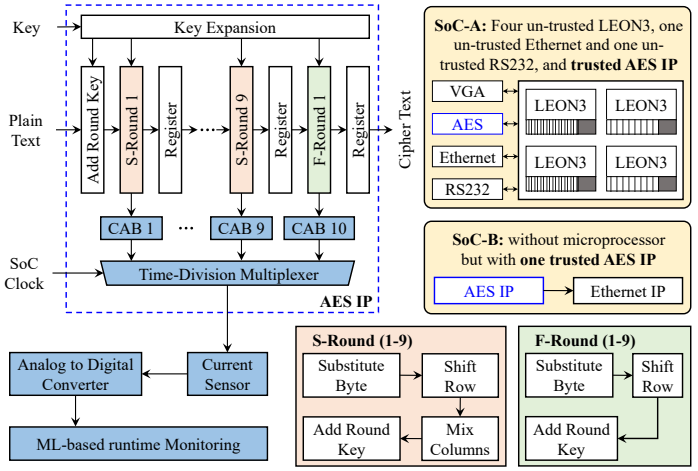


Fig. 17: SoCs to evaluate the scalability and generalizability of MacLeR. SoC-A consists of 4 un-trusted LEON3 processors , one un-trusted VGA IP, one un-trusted Ethernet IP, one un-trusted RS232 and one trusted AES IP. SoC-B consists of one trusted AES IP and one un-trusted Ethernet IP. SoC-B takes the input, encrypts the input and transmits it via Ethernet.

C. Sensitivity Analysis of MacLeR under Aging Effects

For the comprehensive analysis, we evaluated MacLeR for different HT benchmark under different PV and aging variations based on the model presented in literature [23]–[25], i.e., change in operating frequency after Year-1, Year-2, Year-5, and Year-10. Moreover, we also evaluated MacLeR with no aging mitigation and for two aging policies, i.e., Fast-Core-Age-First and balanced-aging profile.

1) Fig 16 shows that in the case of no aging mitigation technique, HT detection accuracy drop of MLP trained using MacLeR subjected to PV is 0.44% and subjected to aging after Year-10 is 8.936% when compared to the case without any variations. However, the decreasing rate of MCC values due to aging is steeper than HT detection accuracy, which shows that the number of false positives and false negatives increase with aging.
2) Fig 16 shows that in the case of the aging mitigation technique, the detection accuracy drop of MLP trained is relatively smaller. Similarly, the decreasing rate of MCC values is significantly lower than the worst-case aging scenario.

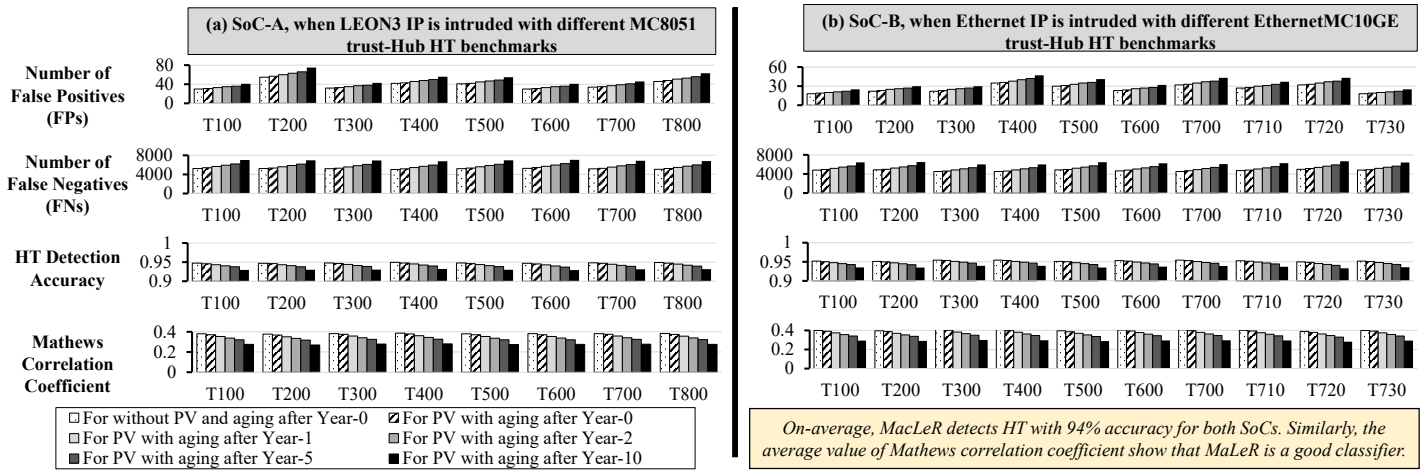


Fig. 18: Performance analysis McLeR for SoC-A and SoC-B in terms of false positives, false negatives and HT detection accuracy in different scenarios, i.e., without PV and aging, PV without aging, PV with aging after Year-0, 1, 2, 5 and 10. Note, results for SoC-A are for the case when LEON3 is intruded with different MC8051 trust-Hub HT benchmarks, and results for SoC-B are for the case when Ethernet IP is intruded with different EthernetMC10GE trust-Hub HT benchmarks.

IX. Scalability and Generalizability of MacLeR

To evaluate the scalability and generalizability of MacLeR, we also evaluated MacLeR on SoC-A (with untrusted LEON3 and trusted AES IP) and SoC-B (non-microprocessor SoC with trusted AES), see Fig, 17. Note, in both SoCs, the fine-grained power profiles of trusted AES IP are obtained by computing the power consumption of computation blocks for each round separately, as shown in Fig. 17. Experimental results in Fig. 18 show the effectiveness of MacLeR for IPw with and without trusted microprocessors. Even in the presence of non-microprocessor trusted IP, MacLeR effectively use the fine-grained power profiles of other non-microprocessor trusted IP, i.e., AES IP, to detect the HT with ≈95% accuracy.

A. Different Workloads running on trusted LEON3 IP

We also evaluated MacLeR, for different input vector to a division workload that is running on a trusted LEON3 IP in LEON3-based SoC. Note, we also considered the PV and different aging effects to evaluate MacLeR for the un-predicted real-world. Fig. 19 shows the HT detection accuracy when three different input vectors are used for the division workload. The experimental analysis shows that for all input vectors, MacLeR behaves very similar trends, high HT detection accuracy and shows very small variations with respect to input vectors. Hence, MacLeR can effectively detect HTs irrespective of input vectors.

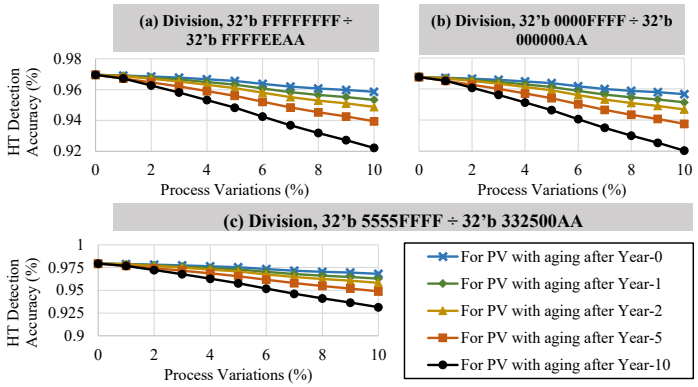
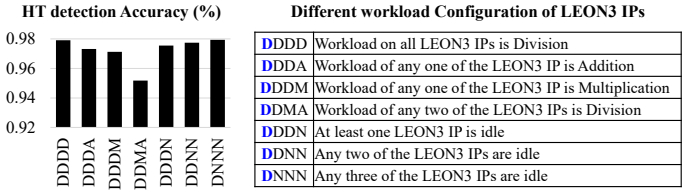


Fig. 19: HT detection accuracy of MacLeR when different input vectors are running on the trusted LEON3 IP in different scenarios, i.e., without PV and aging, PV without aging, PV with aging after Year-0, 1, 2, 5 and 10



Different workload Configuration of LEON3 IPs

DDDD	Workload on all LEON3 IPs is Division
DDDA	Workload of any one of the LEON3 IP is Addition
DDDM	Workload of any one of the LEON3 IP is Multiplication
DDMA	Workload of any two of the LEON3 IPs is Division
DDDN	At least one LEON3 IP is idle
DDNN	Any two of the LEON3 IPs are idle
DNNN	Any three of the LEON3 IPs are idle

Fig. 20: HT detection accuracy of MacLeR when different workloads are running on other un-trusted LEON3 IPs. Note, in these experiments, the work load running on trusted LEON3 IP is division.

B. Different Workloads running on other un-trusted LEON3 IPs

We also evaluated MacLeR for different workload configurations, as shown in Fig. 20. In these experiments, the workload in trusted LEON3 IP is fixed to division, and the rest of the un-trusted LEON3 IPs can remain idle, can run multiplication, addition or division, see the different workload configurations in Fig. 20. These experimental results show that variation in the workload across the chip have negligible impact (i.e., 2% to 3% decrease) on the HT detection accuracy.

X. On-chip and off-chip Overhead of our New Hardware Components of MacLeR

To analyze the on-chip overhead of the proposed MacLeR methodology, we synthesized the RTL of the complete LEON3-based SoC using a 65nm technology in Cadence Genus. On the other hand, the power consumption of off-chip components is estimated based on commonly used SoCs, e.g., TMS3280x SoC for ADC. However, the off-chip area overhead is specific to a given platform. Therefore, in this paper, our primary focus is overhead with respect to power consumption. Table II provides the on-chip and off-chip area and power overhead for LEON3-based SoC and SoC-A[6].

On-chip area overhead: The overall on-chip area overhead for SP-CAB, time multiplexing in LEON3-based SoC is approximately 70.23μm^2, and the on-chip area overhead of SoC-A is approximately 103.26μm^2. The reason behind the extra overhead for the latter is that there are more SP-CABs in SoC-A as compared to SP-CABs in LEON3-based IP. In summary, the area overheads for LEON3-based SoC and SoC-A *is less than 0.025% of the total area, thus negligible.* This analysis also shows that area overheads of the trusted IPs, i.e., LEON3 in

[6]Note, this same SoC but the trusted IP is AES instead of LEON3.

TABLE II: On-chip and off-chip area and power overhead analysis

	Area (μm^2)			
	LEON3-based SoC	SoC-A	LEON3 IP	AES IP
Without overhead	464851.302	464851.302	108330.348	20332.8
With on-chip overhead	464921.532	464954.562	108400.578	20436.06
With on-chip & off-chip overhead	N/A	N/A		

	Power (mW)			
	LEON3-based SoC	SoC-A	LEON3 IP	AES IP
Without overhead	734.180657	734.180657	183.2784452	0.688
With on-chip overhead	734.6559309	734.755157	183.3972637	0.6915
With on-chip & off-chip overhead	770.9199309	764.444157	N/A	N/A

LEON3-based SoC and AES in SoC-A, is also 0.06% in LEON3 IP and 0.5% in AES, thus, negligible.

On-chip and Off-chip power overhead: Similarly, the on-chip power overhead for LEON3-based SoC is ≈0.47 mW, and the on-chip power overhead of SoC-A is 0.57 mW. *which is less than 1% of the total power, thus negligible*. In summary, the on-chip power overheads for LEON3-based SoC and SoC-A *is less than 0.07% of the total area, thus negligible.* However, the off-chip power overheads for LEON3-based SoC and SoC-A are 36.74 mW and 30.26 mW, respectively. The overall on-chip and off-chip power overhead is less than 5% of the total power consumption; thus, it can be considered as tolerable [37]. Similarly, the power overheads of the trusted IPs, i.e., LEON3 in LEON3-based SoC and AES in SoC-A, are also 0.06% in LEON3 IP and 0.5% in AES, respectively, thus negligible.

Summarizing the key benefits:

1) The area overhead of MacLeR is 7x less in terms of power ports as compared to state-of-the-art [9][32][33][12]. Moreover, the best-selected MLP configuration for MacLeR consists of two hidden layers, and each layer has only eight neurons. On the other hand, the MLP in [9] with maximum HT detection accuracy (i.e., 85.12%) consists of 50 hidden layers, and each layer has 100 neurons.
2) The overhead in terms of physical area is larger than the technique of [32][33]. However, this technique uses reverse engineering, which increases the complexity, time, effort, and cost substantially. Our technique, in contrast, uses a very simple design and a minimal area overhead, making it practical for fast deployment for the IoT edge and CPS devices.

XI. Conclusion

This paper presents a methodology to design an ML-based run-time HTs detection (MacLeR) for resource-constrained IoT edge devices. MacLeR first extracts the instruction-dependent fined-grained power profile of different pipeline stages. Afterwards, it addresses the challenging problem of run-time data acquisition by designing a single power-port based current acquisition block. The power profiles are used to train and explore the design space of multiple ML models, and to select the model providing the highest HT detection accuracy. To illustrate the scalability and generalizability of the MacLeR, we evaluated it on different SoCs with microprocessors as trusted IP, non-microprocessor trusted IP, and non-microprocessor SoCs. Our experimental results show that as compared to the state-of-the-art HT detection technique, MacLeR achieves a 10% increase in HT detection accuracy (i.e., 96.256%), while incurring 7x reduction area and power overhead. We also analyzed the impact of process variations and aging variations on MacLeR. The analysis shows that With proper aging policies and PV consideration can effectively, MacLeR can handle the unpredictability of real-world applications. Hence, this simple design with negligible area/power overhead and high tolerance against process variation makes MacLeR feasible in real-world IoT-edge and CPS devices.

References

[1] M. Tehranipoor et al., "A survey of hardware Trojan taxonomy and detection," *IEEE **D&T of computers***, vol. 27, no. 1, pp. 10–25, 2010.
[2] H. Li et.al., "A survey of hardware trojan threat and defense," *Integration, the VLSI*, vol. 55, pp. 426–437, 2016.
[3] I. H. Abbassi et al., "TrojanZero: Switching Activity-Aware Design of Undetectable Hardware Trojans with Zero Power and Area Footprint," in *IEEE **DATE***. IEEE, 2019, pp. 914–919.
[4] S. F. Mossa et al., "Hardware trojans in 3-d ics due to nbti effects and countermeasure," *Integration, the VLSI*, vol. 59, pp. 64–74, 2017.
[5] ——, "Self-triggering hardware trojan: Due to nbti related aging in 3-d ics," *Integration, the VLSI*, vol. 58, pp. 116–124, 2017.
[6] S. K. Nandhiniet al., "Delay-based reference free hardware trojan detection using virtual intelligence," in *Information Systems Design and Intelligent Applications*. Springer, 2018, pp. 506–514.
[7] X. Cuiet al., "Hardware trojan detection using the order of path delay," *ACM **JETC***, vol. 14, no. 3, p. 33, 2018.
[8] N. S. Babuet al., "Wire load variation-based hardware trojan detection using machine learning techniques," in *Soft Computing and Signal Processing*. Springer, 2019, pp. 613–623.
[9] F. K. Lodhi et al., "Power profiling of microcontroller's instruction set for runtime hardware trojans detection without golden circuit models," in *IEEE **DATE***, 2017, pp. 294–297.
[10] ——, "A self-learning framework to detect the intruded integrated circuits," in *IEEE **ISCAS***, 2016, pp. 1702–1705.
[11] O. Söll et al, "Em-based detection of hardware trojans on fpgas," in *IEEE **HOST***. IEEE, 2014, pp. 84–87.
[12] J. He et.al., "Hardware trojan detection through chip-free electromagnetic side-channel statistical analysis," *IEEE **TVLSI***, 2017.
[13] F. K. Lodhi et al., "Hardware trojan detection in soft error tolerant macro synchronous micro asynchronous (msma) pipeline," in *IEEE **MWSCAS***, 2014, pp. 659–662.
[14] J. Zhang et al., "Veritrust: Vrification for Hardware Trust," *IEEE **TCAD***, vol. 34, no. 7, pp. 1148–1161, 2015.
[15] X.-T. Ngo et al., "Hardware trojan detection by delay and electromagnetic measurements," in *IEEE **DATE***, 2015, pp. 782–787.
[16] X. Chen et al., "Hardware trojan detection in third-party digital intellectual property cores by multilevel feature analysis," *IEEE **TCAD***, vol. 37, no. 7, pp. 1370–1383, 2017.
[17] R. Elnaggar et al., "Machine learning for hardware security: Opportunities and risks," ***JETTA***, vol. 34, no. 2, pp. 183–201, 2018.
[18] D. Jap et al., "Supervised and unsupervised machine learning for side-channel based trojan detection," in *IEEE **ASAP***, 2016, pp. 17–24.
[19] A. Kulkarni et al., "Adaptive real-time trojan detection framework through machine learning," in *IEEE **HOST***, 2016, pp. 120–123.
[20] E. Flamand et al., "Gap-8: A risc-v soc for ai at the edge of the iot," in *IEEE **ASAP***, 2018, pp. 1–4.
[21] M. Tehranipoor et al., "trust-HUB," 2020. [Online]. Available: https://www.trust-hub.org/
[22] F. Khalid et al., "Behavior profiling of power distribution networks for runtime hardware trojan detection," in *IEEE **MWSCAS***, 2017, pp. 1316–1319.
[23] D. Gnad et al., "Hayat: Harnessing dark silicon and variability for aging deceleration and balancing," in *ACM/EDAC/IEEE DAC*, 2015, pp. 1–6.
[24] A. Tiwari et al., "Facelift: Hiding and slowing down aging in multicores," in *IEEE/ACM MICRO*, 2008, pp. 129–140.
[25] S. Rehman et al., "dtune: Leveraging reliable code generation for adaptive dependability tuning under process variation and aging-induced effects," in *ACM/EDAC/IEEE DAC*, 2014, pp. 1–6.
[26] Y. Hou et al., "On-chip analog trojan detection framework for microprocessor trustworthiness," *IEEE **TCAD***, 2018.
[27] ——, "R2d2: Runtime reassurance and detection of a2 trojan," in *IEEE **HOST***, 2018, pp. 195–200.
[28] N. Veeranna et al., "Trust filter: Runtime hardware trojan detection in behavioral mpsocs," *Springer **JHSS***, pp. 1–12, 2017.
[29] A. Malekpour et al., "TrojanGuard: Simple and effective hardware Trojan mitigation techniques for Pipelined MPSoCs," in *ACM/EDAC/IEEE DAC*, 2017, pp. 1–6.
[30] H. Zhao et al., "Applying chaos theory for runtime hardware trojan detection," in *IEEE **CISDA***, 2015, pp. 1–6.
[31] S. Bhunia et al., "Protection against hardware trojan attacks: Towards a comprehensive solution," *IEEE **D&T***, vol. 30, no. 3, pp. 6–17, 2013.
[32] C. Bao et al., "On application of one-class svm to reverse engineering-based hardware trojan detection," in *Quality Electronic Design*, 2014, pp. 47–54.
[33] ——, "On reverse engineering-based hardware trojan detection," *IEEE **TCAD***, vol. 35, no. 1, pp. 49–57, 2016.
[34] B. Razavi, *Design of analog CMOS integrated circuits*. Tata McGraw-Hill Education, 2002.
[35] C. SPARC International, Inc., *The SPARC Architecture Manual: Version 8*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1992.
[36] M. Schoeberl et al., "Patmos: A time-predictable microprocessor," *Real-Time Systems*, vol. 54, no. 2, pp. 389–423, 2018.
[37] K. Xiao, D. Forte, Y. Jin, R. Karri, S. Bhunia, and M. Tehranipoor, "Hardware trojans: lessons learned after one decade of research," *ACM **TODAES***, vol. 22, no. 1, p. 6, 2016.