DAWN: Efficient Trojan Detection in Analog Circuits Using Circuit Watermarking and Neural Twins

Jayeeta Chaudhuri, *Member, IEEE*, Mayukh Bhattacharya, and Krishnendu Chakrabarty, *Fellow, IEEE*

***Abstract*—As the globalization of integrated circuits (ICs) continues to advance, the threat of hardware Trojans has emerged as a major concern in ensuring the security and reliability of analog circuits. While a considerable body of prior work has focused on detecting digital Trojans in digital circuits, the detection of analog Trojans in analog circuits has received significantly less attention. We present DAWN, a sensitivity analysis-based analog Trojan detection framework using neural networks to identify potential analog Trojan hotspots and prevent them from being exploited through unauthorized modifications. We incorporate circuit watermarks in these hotspots to provide an additional layer of security. With these watermarks, any malicious modification to the circuit is automatically detected with high accuracy. We target the detection of stealthy, large-delay Trojans that might be inserted either during the chip design or fabrication stages. Experimental results for analog benchmark circuits and two commonly studied analog Trojans demonstrate the effectiveness of the proposed framework.**

***Index Terms*—Analog circuits, analog Trojan, integrated circuits (ICs), mixed-signal design, security and trust, sensitivity analysis.**

I. Introduction

ANALOG integrated circuits (ICs) play a critical role in signal processing, amplifiers, sensors, and power management systems. As ICs become more complex with technology scaling, their susceptibility to analog Trojans and unauthorized modifications also increases. The globalization of the semiconductor industry and the outsourcing of analog ICs to third-party vendors have introduced security threats that significantly affect the integrity of these ICs [1]. Adversaries can introduce malicious Trojans in the design and fabrication stages [2], [3].

Analog circuits are typically designed to operate across a range of bias voltages; even a small change in the bias can substantially affect the circuit behavior. Analog Trojans can be carefully introduced in these circuits such that they remain hidden under normal operating conditions and become activated due to certain specific variations in the operating voltages. The vulnerability of analog circuits to Trojans requires effective detection methods to ensure security and integrity.

Recently, ML-based approaches have been developed to detect hardware Trojans. The work in [4] and [5] learns features extracted from Trojan-inserted (TI) gate-level netlists. A majority of prior work on Trojan detection has been limited to digital logic, which is largely due to the fact that digital circuits operate on discrete signals and can be easily monitored to detect unusual bit-flip patterns [6], [7]. In contrast, detection of analog Trojans is challenging due to the following reasons.

1) *Area:* Analog Trojans occupy small area; therefore, they can be easily embedded in the design netlist or the chip layout.
2) *Rare Trigger Conditions:* Trojans are typically inserted in the noncritical paths of a circuit. As a result, their functionality remains hidden during chip verification. In digital circuits, the signals are discrete values (0 or 1); a digital Trojan can be triggered by a rare input sequence of bits. However, in analog circuits, the signals are continuous values. Therefore, analog Trojans can be triggered by changes in the input signal level or the effects of process variations, which are difficult to predict. For example, in [8], a capacitor-based Trojan is proposed, which is activated when one of the capacitive components reaches a certain threshold voltage.
3) *Design Complexity:* Large circuits (e.g., a bandgap filter) consist of hundreds of components and paths. Trojans may be stealthily inserted in one of the less sensitive paths and triggered by a subtle change in the voltage across that path.

Traditional methods use physical analysis techniques, e.g., scanning electron microscopy (SEM) or focused ion beam, to detect the presence of additional circuitry or other anomalies in the circuit [9], [10]. However, these methods are costly and require comparison with a golden (reference) circuit. Obtaining a golden circuit can be challenging as the IP is protected and access requires permission from the IP owner. Moreover, the generation of a reference circuit requires a significant amount of time and design expertise. Although the SEM-based detection techniques can identify malicious modifications, they are often expensive as well as destructive.

Analog Trojans such as the A2 and large-delay Trojans have recently been implemented [8], [11]. These Trojans occupy a small footprint and have a high payload activation time, making them challenging to detect during standard circuit operation. In this article, we propose DAWN, a Trojan detection method for analog circuits using neural twins and circuit

Manuscript received 24 August 2023; revised 26 January 2024 and 31 March 2024; accepted 1 April 2024. Date of publication 3 April 2024; date of current version 20 September 2024. This work was supported in part by the Synopsys and in part by the National Science Foundation under Grant CNS-2011561. This article was recommended by Associate Editor G. Qu. *(Corresponding author: Jayeeta Chaudhuri.)*

Jayeeta Chaudhuri and Krishnendu Chakrabarty are with the School of Electrical, Computer, and Energy Engineering, Arizona State University, Tempe, AZ 85287 USA (e-mail: jchaudh3@asu.edu).

Mayukh Bhattacharya is with the Custom Design Manufacturing Group, Synopsys, Sunnyvale, CA 94043 USA.

Digital Object Identifier 10.1109/TCAD.2024.3384948

1937-4151 © 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: Arizona State University. Downloaded on October 09,2024 at 01:01:59 UTC from IEEE Xplore. Restrictions apply.

watermarking. We present an automated sensitivity analysis-based Trojan detection framework, which combines the analog neural twin (ANT) model from recent work [12] with a novel circuit watermarking technique. The proposed approach can automatically identify critical nodes that are least sensitive to input perturbations, and are therefore potential Trojan hotspots. In other words, this method enables the rapid identification of specific parts of the design where an adversary will most likely insert an analog Trojan. This technique also helps to localize the Trojan attack to the rarely activated nodes of the circuit. Additionally, we place curated circuit watermarks in the identified Trojan hotspots. By inserting watermarks in the least sensitive nodes of a circuit and creating an observable watermark output pin, we demonstrate a mechanism for detecting the presence of stealthy Trojans while maintaining the circuit's original functionality. Moreover, the watermark output pin detects any removal or tampering attempts on the circuit watermarks, thus enhancing the security of the watermarking scheme. Overall, the combination of the neural twin and the circuit watermark provides a comprehensive and effective technique for analog Trojan localization as well as detection.

While prior work has demonstrated the detection of A2 Trojans on the digital domain, this is the first demonstration of its impact on analog circuits. By highlighting the risk on several analog designs, we show that the A2 Trojan can be used to exploit both digital and analog systems. The proposed countermeasure performs targeted detection of insensitive nodes that are susceptible to Trojan alterations, and the countermeasure is shown to secure different types of analog circuits as well as large-scale mixed-signal SoCs.

The key contributions of this article are as follows.

1) We present an automated sensitivity analysis framework using an ANT model for identifying potential Trojan hotspots in analog circuits.
2) We design circuit-based watermarks with negligible power overhead to efficiently detect analog Trojans that have been maliciously inserted in the design. Moreover, the proposed watermark circuits effectively detect removal attacks, further strengthening the security of analog circuits.
3) We demonstrate the generalizability of the proposed method to different types of analog circuits.

The remainder of this article is organized as follows. Section II describes related prior work and establishes the motivation for this work. Section III presents the threat model. Section IV describes circuit simulation-based methods to motivate the proposed solution. Section V describes our proposed analog Trojan detection framework. Experimental results are presented in Section VI. Section VII concludes this article.

II. Background and Motivation

A. Prior Work on Analog Trojan Detection

1) Design-Time Trojan Detection: This detection approach identifies hardware Trojans that are inserted either in design or during the manufacturing of an analog IC. In [14], an information flow tracking (IFT)-based approach is presented to detect the presence of Trojans from a circuit layout. The IFT tracks the possible locations (or taint sources) where an attacker can insert a trigger for Trojan activation. This is done by carefully analyzing the capacitors in the circuit that have a size larger than a predetermined threshold value. If a taint source is identified, IFT performs taint propagation to activate an interrupt at the circuit output. However, the IFT-based security solution is limited to small circuits. For a larger and more complex analog circuit, the IFT can become computationally impractical as it needs to identify taint sources across multiple nodes in the circuit. Moreover, this detection method may not be effective against Trojans that are activated at run-time [11], thus limiting its applicability.

2) Test-Time Trojan Detection: This approach involves the detection of Trojans after an analog IC is manufactured. Pavlidis et al. [16] proposed an approach based on built-in-self-test to facilitate the detection of Trojans during functional testing. By monitoring the symmetry of the internal nodes in an analog circuit, a set of invariant signals are generated. Additionally, checkers with predefined tolerance windows are implemented. The checkers monitor the invariant signals and raise a flag when any malicious modification is detected in the circuit. However, this method requires a considerable amount of additional hardware for generating the invariances, which significantly increases circuit area. Also, several analog Trojans inserted during fabrication do not become active till after functional testing [8].

3) Run-Time Trojan Detection: This type of detection involves the monitoring of the analog circuit in order to detect any anomalous behavior during functional operation. The work in [17] uses thermal sensors to evaluate the power consumption and detect deviations in the temperature profile that might indicate the presence of a Trojan. Another approach presented in [15] uses a current sensing-based circuit to detect analog Trojans at run-time. Although this method is able to detect the presence of a Trojan, it may not be able to determine the locations of the Trojan in the circuit. Moreover, the circuitry needed for current sensing adds to significant power overhead of the circuit.

B. Motivation

Prior methods on analog Trojan detection suffer from hardware area overhead and increased test time. They may also require access to a golden netlist for comparison. Also, these methods are not aimed toward Trojan localization and are of limited applicability to large analog circuits. This motivates the need for a generalizable Trojan detection framework that supports Trojan localization as well as efficient design-time and run-time detection. Table I presents a comparison of the proposed Trojan detection framework with prior work.

Sensitivity analysis is a useful tool for Trojan detection. Prior work [18] has used a SCOAP-like approach technique for sensitizing critical nodes in digital circuits. Recently, an ANT-based framework has been developed to generate tests for different fault locations in analog circuits using a gradient-based approach [12]. We explore the sensitivity-based properties of well-known analog Trojans and use them to identify potential Trojan hotspots in analog circuits. We deploy the ANT model for accurately replicating a circuit and evaluating the sensitivity of the primary and intermediate nodes of the circuit. The ANT model is an analog representation of a neural twin, which has been used for criticality analysis in digital circuits [19]. Using sensitivity analysis, we locate the

Authorized licensed use limited to: Arizona State University. Downloaded on October 09,2024 at 01:01:59 UTC from IEEE Xplore. Restrictions apply.

TABLE I
COMPARISON OF DAWN WITH PRIOR WORK ON ANALOG TROJAN DETECTION

Parameter	[13]	[14]	[15]	**Proposed method**
Attack scenario	Fabrication phase	Fabrication phase	Fabrication phase	Both design and fabrication phases
Scalability?	No	No	No	Yes
Detection methodology	Hardware interrupt	Information flow tracking	Current sensing	Sensitivity analysis
Ease of deployment?	No	No	Yes	Yes
Trojan attack localization?	No	No	No	Yes
Run-time detection?	Yes	No	Yes	Yes
Area overhead	High	Low	Medium	Low
Automated?	No	No	No	Yes

critical nodes in the circuit that are most susceptible to Trojan insertion. Next, we embed customized, low overhead circuit watermarks in the design to detect malicious modifications.

Recent work uses graph neural networks (GNNs) to model analog circuits as graphs at the transistor level, and uses the graph-structured data to identify subcircuits within larger analog designs [20]. GNN performs tasks such as edge classification and graph classification, and is agnostic to the knowledge of a transistor's functionality. The nodes represent the transistors in the graph, and they do not capture any functional information. However, the fundamental concept of using a neural twin is to represent MOSFET functionality in terms of its *I–V* characteristics. Hence, the motivation of our work and that of prior work on analog circuit modeling using GNNs are quite different. In particular, GNNs are used to generate graphs that are unique for each circuit; hence a GNN for a circuit can be used to distinguish it from other circuits. This approach has not been used for modeling current/voltage characteristics at transistor-level. In contrast, in our work, we model each transistor *IV* characteristics using FET twins. In other words, we focus on analog simulation at transistor-level, and this modeling approach helps us to determine paths of low sensitivity in analog circuits.

III. THREAT MODEL

Analog Trojans can be inserted in one or multiple stages during design and fabrication of the chip [21]. The attack vectors for the insertion of analog Trojans are as follows.

1) *Design Netlist:* Insertion of trigger circuits, and additional input and output wires that connect and influence the other modules in the netlist.
2) *Design Layout:* Malicious Trojan insertion by third-party electronic design automation tools.
3) *Fabricated Chip:* The graphic database system (GDS) file contains proprietary information about the placed-and-routed circuit. An attacker in the foundry may reverse-engineer the layout to the corresponding design netlist and insert a Trojan in the circuit.

In this work, we assume that the adversary is any third-party user present during the design and fabrication of the chip and has unauthorized access to the design netlist or the design layout. They can insert the analog Trojan in the design to disrupt the functionality of the circuit. The threat model with the associated attack vectors is illustrated in Fig. 1.

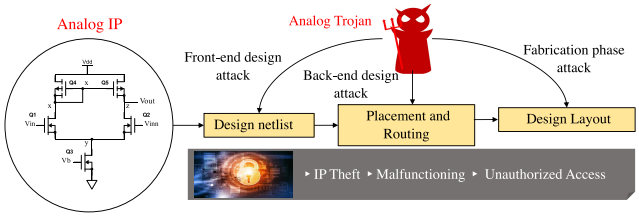

Fig. 1. Security threats associated with analog Trojan insertion.

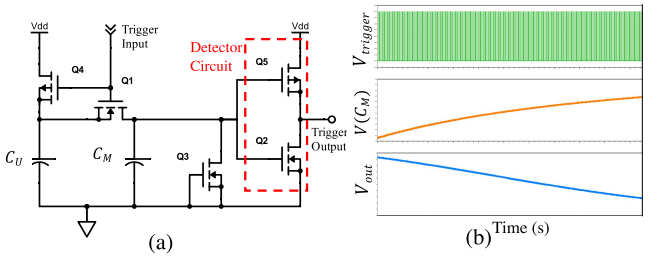

Fig. 2. (a) Schematic of the A2 Trojan [8]. (b) SPICE simulation results for the A2 Trojan.

IV. IMPACT OF ANALOG TROJANS ON CIRCUIT FUNCTIONALITY

A. Analog Trojan Characteristics

A hardware Trojan consists of a trigger and a payload. The trigger consists of a specific input signal or a series of input signals that can activate the Trojan either immediately or after a specific period of time. The payload is the malicious effect of the Trojan that affects the normal functionality of the circuit. It has been shown how the trigger and payload can be carefully designed to avoid detection by traditional verification tools [3]. A common technique for analog Trojan insertion is the modification of the existing design by inserting analog components and wires. These components can be in the form of additional transistors, resistors, or capacitors. It is also possible to insert Trojans via circuit layout modifications. These modifications can include changes to the placement or routing of components within the circuit. These alterations can affect the circuit functionality significantly and cause performance degradation. In this work, we focus on the well-known A2 and large-delay analog Trojans [8], [11].

1) A2 Trojan: A2 is a fabrication-type Trojan that can be stealthily inserted by an adversary on a completely placed and routed circuit [8]. The schematic of the A2 Trojan is shown in Fig. 2(a). When the trigger input is low, transistor $Q4$ is turned on while transistor $Q1$ remains turned off. This allows C_U to get charged. When the trigger input switches

Authorized licensed use limited to: Arizona State University. Downloaded on October 09,2024 at 01:01:59 UTC from IEEE Xplore. Restrictions apply.

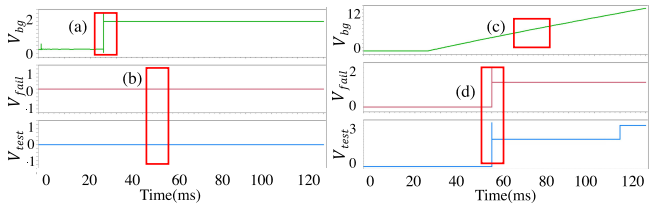

Fig. 3. HSPICE simulations for a bandgap filter. (a) TF output. (b) Value "0" indicates no fault is detected in the circuit during simulation. (c) Circuit output when A2 Trojan is inserted. (d) A2 Trojan behavior is captured at circuit output.

to high state, charge sharing takes place between C_U and C_M. The C_M voltage activates transistor $Q3$, which in turn, is fed to the detector circuit, a skewed inverter in this case. By appropriately tuning the trigger input conditions, an adversary can activate the payload through the trigger output node. Fig. 2(b) shows the behavior of the A2 Trojan when it is activated. The trigger input V_{trigger} is a pulse waveform that slowly increases the voltage across C_M. Upon reaching a specific voltage, C_M enables the detector circuit and activates the payload V_{out}.

Extending upon A2 Trojan, Gupta et al. [22] proposed DELTA, a conditionally gated glitch generator circuit that enables toggling of the trigger input only for a specific duration. Upon sufficient charge build-up at C_M, a charge detector based on cross-coupled inverters activates the Trojan payload.

2) Large-Delay Trojan: A trickle charge (TC)-based fabrication-type Trojan is presented in [11]. The TC Trojan relies on rarely activated trigger events, which are not sensitive to input patterns during production test. The TC Trojan is implemented using a reverse-biased diode and an on-chip capacitor. Due to the reverse-bias condition, the current through the diode increases slowly, which leads to delayed Trojan activation. It has been shown that the TC Trojan takes almost 47 h to generate the payload after the trigger is applied. The A2 and TC Trojans occupy small area and are activated only under specific trigger conditions, thus making them difficult to be detected using traditional verification methods.

B. Identifying Trojan Features in Netlists

Our preliminary analysis of analog Trojan detection in an analog circuit involves exploring the output voltage and output current characteristics of the circuit. The trigger input node is carefully chosen for the circuit under test (CUT). We first create a baseline HSPICE simulation sim_1 for the analog circuit without inserting a Trojan. The simulation demonstrates the normal circuit functionality in terms of the frequency response, the output *IV* characteristics, and the signal transition delays. Next, we insert the A2 Trojan shown in Fig. 2 into the original HSPICE netlist of the circuit. Note that the trigger output of the A2 Trojan is connected to the primary output (PO) node in a way such that the Trojan payload disrupts the correct circuit operation. We run a new HSPICE simulation sim_2 based on the A2 TI netlist. The goal is to identify any deviations in circuit behavior that indicate the presence of a Trojan in the design. Fig. 3 illustrates the disruptions in the normal operation of a bandgap filter circuit when an A2 trojan is stealthily inserted in one of the output nodes of the circuit. The bandgap filter is tuned such that during normal operation, its output voltage satisfies the following condition: $0 \leq V_{bg} \leq 2$. Any value beyond this range activates the signals V_{fail} and V_{test}, which indicate circuit malfunctioning and the severity of the disruption, respectively. Fig. 3(c) illustrates the activation of the A2 Trojan payload after 50 ms; Trojan activation triggers the V_{fail} and V_{test} nodes [Fig. 3(d)], indicating the presence of the A2 Trojan in the design netlist.

V. Analog Neural-Twin Framework for Trojan Hotspot Detection

We develop a two-tier analog Trojan detection method that facilitates run-time Trojan detection. The first stage involves the identification of potential Trojan hotspots in the circuit. A *Trojan hotspot* is a region of the circuit that is most vulnerable to Trojan insertion. Once these critical regions are marked, the second stage involves the insertion of circuit watermarks in the Trojan hotspots. These watermarks do not add a significant number of transistors in the original circuit and serve as a signature to detect the presence of an analog Trojan. Fig. 4 illustrates the proposed framework for Trojan localization and detection.

A. Generation of Analog FET-Twins

An FET-twin is the neural network representation of a MOSFET. In other words, it mimics the *IV* characteristics of an actual transistor. The FET-twin, ϕ_{fet}, consists of four fully connected layers followed by a forward layer including the Sigmoid activation function [12]. It can be formulated as: $\phi_{\text{fet}}(x) = \sigma(W_4\sigma(W_3\sigma(W_2\sigma(W_1x + b_1) + b_2) + b_3) + b_4)$, where x is the input to the first fully connected layer, σ is the activation function applied after each fully connected layer, and W_i and b_i represent the weight and bias of the ith fully connected layer, respectively. As shown in [12], the FET-twin is capable of generating both the drain current (I_D) and output voltage (V_{DS}) values for a specific transistor. For a given transistor, there are two variants of FET-twins: 1) current-based FET-twin ϕ_{I_D} and 2) voltage-based FET-twin $\phi_{V_{DS}}$. The ϕ_{I_D} variant predicts the drain current for the transistor corresponding to its input gate voltage (V_G) and all the primary inputs (PIs) of the circuit. The second variant $\phi_{V_{DS}}$ determines the output drain voltage (V_{DS}) of the transistor as a function of the gate input as well as all the PIs for the circuit. Both the $\phi_{V_{DS}}$ and ϕ_{I_D} FET-twin variants are trained using the mean-squared error (MSE) loss function. The MSE incorporated in ϕ_{I_D} ($\phi_{V_{DS}}$) calculates the loss between the predicted and the desired drain current (output drain voltage). The loss is propagated backward through the neural network model to adjust the model's weights and biases. The optimizer used is Adam with a learning rate of 0.01.

To generate realistic ϕ_{I_D} and $\phi_{V_{DS}}$ FET-twins, we train them on a wide range of data that we obtain from the HSPICE simulations. For training the ϕ_{I_D} ($\phi_{V_{DS}}$) FET-twin and enabling feature selection, we collect the I_D (V_{DS}) values by sweeping the PI test inputs across a wide range of voltage values. We also incorporate the FET-twin gate voltage, V_G as the feature for training the FET-twin. Note that we generate both ϕ_{I_D} and $\phi_{V_{DS}}$ FET-twins for the PO transistor and only the $\phi_{V_{DS}}$ FET twin for the PI transistors and the intermediate transistors in the circuit. We train a voltage-based FET-twin $\phi_{V_{DS}}$ for each

Authorized licensed use limited to: Arizona State University. Downloaded on October 09,2024 at 01:01:59 UTC from IEEE Xplore. Restrictions apply.

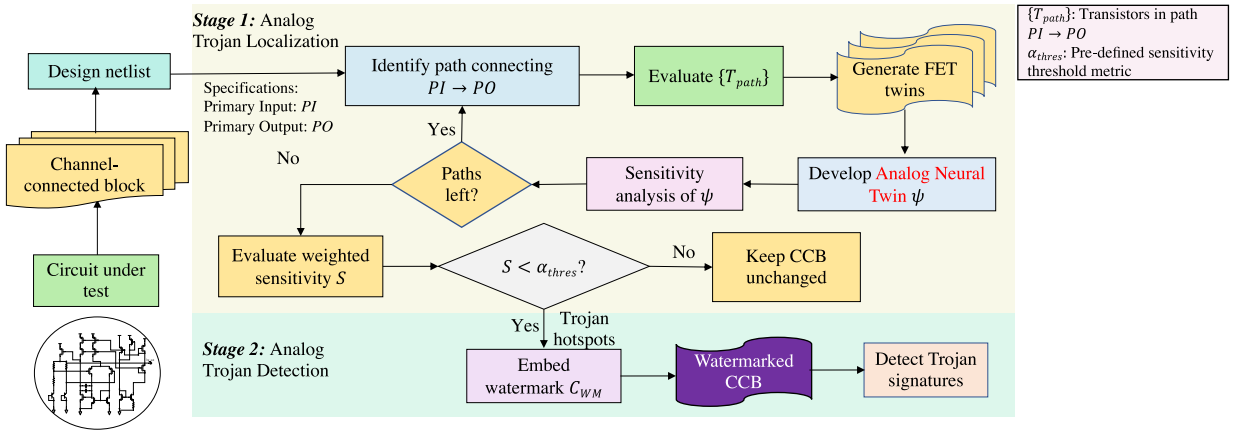

Fig. 4. Overall flow of the two-tier analog Trojan detection framework.

transistor component in an analog circuit. This training process is performed only once for each transistor in the circuit. The current-based FET twin ϕ_{I_D} generates I_D that is dependent specifically on the V_{DS} generated by $\phi_{V_{DS}}$ and the V_G values. Therefore, we train only a single ϕ_{I_D} FET twin per circuit. Depending on the specifications of the analog CUT, i.e., the input and output voltage ranges, ϕ_{I_D} is trained over a wide range of I_D values. To enhance training accuracy and achieve model convergence for ϕ_{I_D}, we train the ϕ_{I_D} using the log magnitude for I_D.

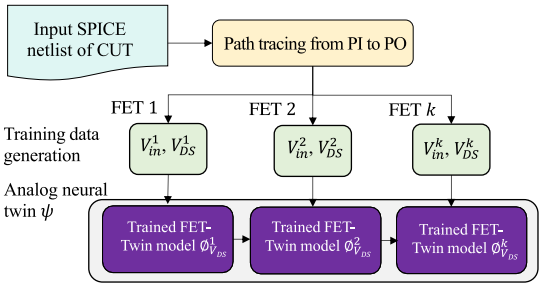

Fig. 5. Generation of the ANT [12].

B. Analog Neural Twin-Based Sensitivity Analysis

1) Automated Path-Tracing Methodology and Analog Neural Twin Generation: We develop an automated path-tracing method to quickly identify all paths that connect the PIs to POs of a circuit. The path-tracing methodology involves the following steps.

1) *Node Selection:* Select the PI transistor x and the PO transistor out of the path for which we need to construct the ANT for sensitivity evaluation. Note that x and out are a part of the HSPICE circuit netlist.
2) *Path Tracing:* Given x and out, we construct a directed acyclic graph (DAG) that starts by identifying the intermediate transistors that are connected to the drain of x. A DAG is a graph that has forward edges from one node to other, without the presence of cycles. The DAG terminates when the drain of an intermediate transistor is connected to the gate of out. Multiple paths might be encountered during the path tracing method that connect x and out. We discard the paths that contain transistors with a PI different from that of the PI transistor input. This ensures that we are evaluating the path sensitivities that are directly relevant to a specific PI voltage. Let the final number of paths be m. For a path $Path_i$, $1 \leq i \leq m$, we generate a list L_i that consists of all the transistors present in $Path_i$. Note that cyclic paths are not relevant in the particular problem of identifying Trojan hotspots; the sensitivity values of these paths are influenced by intermediate nodes present in the feedback loops and hence, do not give us relevant information about the presence of a Trojan in a circuit. Therefore, we evaluate the sensitivities of forward, directed paths from the drain of x to the gate of out for identifying probable Trojan insertion paths.
3) *Generating the FET-Twins and Constructing the ANTs:* For each path $Path_i$, we obtain the set of transistors in that path, as explained in the previous step. Consequently, we train a voltage-based FET-twin corresponding to each transistor in that path. Note that we train each voltage-based FET-twin based on *IV* characterization data of the corresponding transistor; the biasing voltages of each transistor are derived from SPICE simulation of the analog design. The ANT model of a particular path is formed by stitching together the pretrained voltage-based FET-twins; the stitching order is determined by the sequence in which the corresponding transistors are connected in that path. For each path $Path_i$, the ANT model $\Psi_i : \{\Psi\}$, $1 \leq i \leq m$ is constructed. Fig. 5 illustrates the procedure for training and evaluation of voltage-based FET-twins and the ANT for a specific circuit.
4) *Sensitivity Calculation:* We use the sensitivity analysis approach discussed above to calculate the sensitivity of the PO transistor out (as a function of the input voltage applied to x) for each ANT Ψ_i. For an analog CUT with m paths (i.e., m neural twins), the worst-case computational complexity of the sensitivity analysis procedure is $\mathcal{O}(m \times l_{\text{avg}} \times n)$, where l_{avg} is the average number of transistors present in each path and n is the number of input voltage samples. In large circuits, where there are multiple paths connecting the PI and PO nodes, parallel processing of ANT-based sensitivity analysis can

Authorized licensed use limited to: Arizona State University. Downloaded on October 09,2024 at 01:01:59 UTC from IEEE Xplore. Restrictions apply.

Input: SPICE netlist $\mathcal{N}$
Output: $\{\psi\}$ /*List of generated neural twins for $\mathcal{N}$*/
$\{\psi\} \leftarrow 0$ /*Initialize list*/
$\mathcal{CCB}^m \leftarrow \mathcal{N}$ /*Convert $\mathcal{N}$ to m number of CCB(s)*/
for $1 \leq i \leq m$ **do**
  Determine set of $\{PI\}^x, \{PO\}^y$ for $\mathcal{CCB}_i^m$ /*x and y are number of PI and PO nodes, respectively*/
  for $1 \leq j \leq x$ **do**
    for $1 \leq k \leq y$ **do**
      $\{T_{path}\} \leftarrow 0$ /*Initialize list*/
      if `path_exists`$(PI_j^x, PO_k^y, \mathcal{CCB}_i^m)$ **then**
        $\{T_{path}\} \leftarrow$ `FET_list`$(PI_j^x \rightarrow PO_k^y)$
        $\{\psi\} \leftarrow$ `twin_gen`(T_{path})
      end
    end
  end
end
return $\{\psi\}$

Fig. 6. Pseudocode for ANT generation.

Input: $\{\psi\}^x$ /*Generated x analog neural twins*/
Output: $\{\mathcal{T}\}$ /*List of analog Trojan hotspots*/
$\{\mathcal{T}\} \leftarrow 0$ /*Initialize list*/
for $1 \leq i \leq x$ **do**
  Evaluate $\mathcal{S}_i =$`sens_calc`(ψ_i^x, PI_i, PO_i) /*Sensitivity value of path $(PI_i \rightarrow PO_i)$*/
  if $\mathcal{S}_i < \alpha_{thres}$ **then**
    $\mathcal{T} \leftarrow PI_i$
  end
end
return $\{\mathcal{T}\}$

Fig. 7. Pseudocode for analog Trojan hotspot detection.

be incorporated. This is done by training and evaluating each neural twin independently based on the training data samples collected from the transistors in these paths. Parallelizing the sensitivity evaluation process reduces the overall computational time and improves scalability for large circuits with multiple paths.

Note that a realistic, larger circuit will typically consist of multiple differential amplifiers, inverters, and other analog components. In this scenario, we aim to perform sensitivity analysis on the channel connected blocks (CCBs) corresponding to the circuits. The CCBs are the building blocks of an analog circuit and consist of transistors, resistors, and capacitors [23]. They are arranged according to the connectivity and functionality of the original circuit, and hence, can be analyzed as separate subcircuits. Fig. 6 illustrates the procedure for developing ANT models for each CCB extracted from an analog circuit. Once all the neural twin models are generated, we evaluate the sensitivity values of the PO nodes of each CCB using these models. The methodology is illustrated in Fig. 7. If the sensitivity value of a node is less than α_{thres}, we classify the node to be a Trojan hotspot. The functions defined in Figs. 6 and 7 are described as follows.

1) `path_exists`$(PI_j^x, PO_k^y, \mathcal{CCB}_i^m)$*:* Given PI_j^x and PO_k^y, the input node and the output node, respectively, for the evaluated CCB, $\mathcal{CCB}_i^m$, $1 \leq i \leq m$, we determine whether a path exists between the input and output nodes of $\mathcal{CCB}_i^m$.
2) `FET_list`$(PI_j^x \rightarrow PO_k^y)$*:* Once we have determined that a path exists between PI_j^x and PO_k^y, we retrieve the set of transistors in that path and store them in the list $\{T_{\text{path}}\}$.

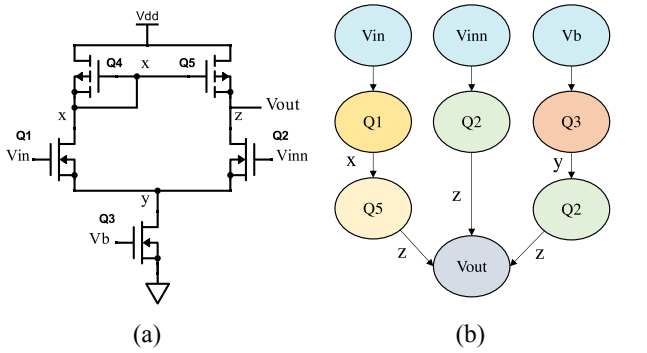

Fig. 8. (a) Single-stage differential amplifier circuit. (b) Construction of paths connecting the PI and PO nodes.

3) `twin_gen`$(\{T_{path}\})$*:* Note that we generate FET-twins for all the transistors listed in T_{path}. We train the FET-twins on a wide range of input/output data pairs, which ensures that the FET-twins are robust and accurate across a diverse range of test input voltages. We stitch the pretrained FET twins corresponding to the transistors present in T_{path} to form the ANT ψ for that path.
4) `sens_calc`(ψ_i^x, PI_i, PO_i)*:* Once the ANT has been trained for the path $PI_i \rightarrow PO_i$, we use it to perform sensitivity analysis of the PO_i node for a wide range of input voltages applied at the PI_i node. This is useful for determining the least sensitive nodes in the analog CUT that are susceptible to analog Trojan insertion.

2) Identifying Trojan Hotspots Based on Sensitivity Analysis: Consider the CUT, namely, a single-stage differential circuit illustrated in Fig. 8. As shown in Fig. 8, there are three possible paths that connect the PI to the PO of the circuit. The objective of the proposed sensitivity analysis method is to find the critical paths in the CUT that are potential Trojan hotspots. The sensitivity of a path connecting the PI to the PO of a circuit is given by: $\delta_{\text{path}} = (\Delta V_{\text{OUT}}/\Delta V_{\text{IN}})$, where ΔV_{OUT} is the change in the original PO voltage V_{OUT} when the PI voltage V_{IN} changes by ΔV_{IN}. The sensitivity value represents the effect of a small change in the PI voltage value on the output voltage of the circuit. An adversary strategically targets the least sensitive paths in a circuit, inserting the Trojans that are activated only for specific dc voltages. Least-sensitive paths refer to the specific paths in the circuit that do not incur significant voltage deviations at the PO due to change in the PI or intermediate node voltages. In the context of Trojan detection, sensitivity analysis becomes crucial in quantifying the impact of these voltage deviations at the circuit PO(s). Least-sensitive paths are a target for stealthy Trojan insertion because voltage deviations in the nodes of these paths are less likely to propagate to the circuit output. Hence, the Trojan does not significantly impact the circuit performance and remains undetected during design verification.

The analog Trojans explored in this work are triggered based on dc voltages. Consequently, the payloads of these Trojans impact the dc voltage of the intermediate transistors where they are inserted, as well as the PO of the circuit. The activation of these Trojans is dependent on specific voltages aligned with steady-state or dc conditions rather than transient events. In the context of Trojan detection, the focus is on identifying deviations in normal circuit behavior that indicate the presence of stealthy Trojans inserted either in rarely activated nodes or sensitive nodes of the circuit. As demonstrated in [8], when

Authorized licensed use limited to: Arizona State University. Downloaded on October 09,2024 at 01:01:59 UTC from IEEE Xplore. Restrictions apply.

TABLE II
PERFORMANCE OF FET-TWINS CONSTRUCTED FROM A SINGLE-STAGE DIFFERENTIAL AMPLIFIER CIRCUIT

Transistor	FET-Twin Type	R2 Score	No. of epochs
$Q1$	$\phi_{V_{DS}}$	1.0000	3801
$Q5$	$\phi_{V_{DS}}$	0.9728	4650
$Q5$	ϕ_{I_D}	0.9784	4999
$Q2$	$\phi_{V_{DS}}$	0.9999	3665
$Q2$	ϕ_{I_D}	0.9729	4987
$Q3$	$\phi_{V_{DS}}$	0.9899	4999

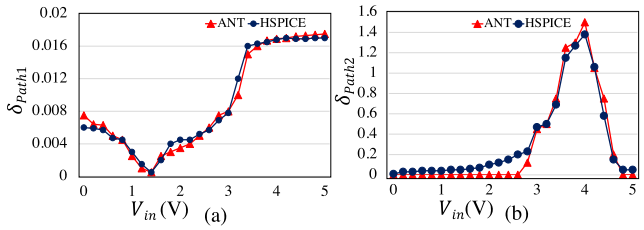

Fig. 9. Sensitivity trends for (a) Path 1 and (b) Path 2 of the single-stage differential amplifier circuit.

the A2 Trojan is inserted in least sensitive paths of a circuit, it remains dormant within certain dc voltage levels at the input. Hence, sensitivity evaluation of the circuit in the dc domain aids in identifying these least sensitive paths that are susceptible to Trojan insertion.

For each of the three paths in Fig. 8, we construct the FET-twins for the PI, intermediate, and the PO transistors that are present in the path. For example, to construct the ANT model for the first path shown in Fig. 8, we generate the $\phi_{V_{DS}}$ FET-twins for $Q1$ and $Q5$ transistors as a function of their respective V_G values and the PI voltages. We also generate the ϕ_{I_D} FET-twin for Q_5 as it is the PO transistor.

Table II lists the training accuracies for the generated FET-twin models in terms of their R2 scores. These results demonstrate that the FET-twins have been efficiently trained to replicate the actual MOSFET transistors, and are able to predict the output currents and voltages that are consistent with Ohm's law. After obtaining all the required FET-twins corresponding to the transistors in a given path, we connect them sequentially and construct the ANT model.

The ANT model is unique for every analog CUT; it provides an end-to-end differentiable representation of the analog circuit and facilitates the computation of gradients using backpropagation. The generated gradients are used for sensitivity evaluation. Note that we do not conduct explicit training of the ANT model. Instead, the ANT model is formed by connecting the pretrained FET twins. The backpropagation operation is performed once to calculate the gradients of the nodes in the model. For sensitivity evaluation on other analog circuits, the FET-twins need to be retrained and a new ANT model has to be regenerated. Importantly, the construction of the FET-twins and the ANT model does not incur any hardware cost. The only cost associated is the one-time training time overhead of the FET-twins. For the designs considered in this article, the average run-times for training the ϕ_{I_D} and $\phi_{V_{DS}}$ FET-twins to achieve improved model performance (see Table II) are only 1.3 min and 48 s, respectively.

Validating Sensitivity Values With HSPICE Simulations: In order to verify the accuracies of the pretrained FET twins and the ANT model, we compare the sensitivity values returned by the model with HSPICE simulations. This comparison is necessary because the sensitivity values generated by the ANT model will be used for identifying Trojan hotspots. Fig. 9 analyzes the sensitivity trends for different paths of a single-stage differential amplifier circuit when the PI voltage is swept from 0 to 5 V, in steps of 0.1 V. We observe that the sensitivity values returned by the ANT model are similar to the HSPICE-generated sensitivity values; this shows that the model is well trained and accurately predicts the circuit behavior. The sensitivity values of a path depend on the number of transistors in that path, their operating points, and their sensitivity to voltage deviations at the input. We observe that Path 1 demonstrates an overall lower sensitivity trend as a function of V_{IN} compared to Path 2, thus indicating that it is more susceptible to analog Trojan insertion. The computational advantage of using the ANT-based approach over HSPICE simulations is also evident. While HSPICE simulations take as much as 4 s to evaluate the sensitivity of a single path for a particular PI voltage, the pretrained ANT models perform the same task in only 0.024 s. This significant reduction in computation time is particularly advantageous when identifying Trojan hotspots in larger and more complex circuits. Note that a numerical approach, as employed in HSPICE simulations, involves rigorous calculations to determine sensitivities, which does not scale well for larger circuits due to high computational complexity. In contrast, once neural twins are trained, they provide fast evaluations, leveraging the ability to represent circuit behaviors effectively across a wide range of operating voltages.

C. Analysis of the Sensitivity Threshold

For a specific analog CUT, we determine possible paths between the PI and PO nodes and evaluate the sensitivity values for each such path using the ANT-based framework. The sensitivity threshold is determined by evaluating the sensitivity values across a range of PI test voltages and carrying out a cluster analysis for all paths in a given circuit. Next, the sensitivity values for all the paths are sorted in terms of the test input voltage, and clusters with a clear boundary are identified. The sensitivity threshold is useful as it helps us to distinguish between a Trojan hotspot and a benign node in a circuit. We refer to this boundary as α_{thres}. Any sensitivity value above α_{thres} is classified as a benign node; a value lower than α_{thres} is considered a potential Trojan hotspot. Fig. 10 presents the results of cluster analysis of different paths for a single-stage differential amplifier circuit as a function of the PI voltage. Table III lists the threshold sensitivity values that we obtained for different analog benchmark circuits. The variation in α_{thres} value among analog circuits is due to its dependence on the circuit specifications such as the range of voltages (in V) for the PI and PO nodes, the number of paths connecting PI to PO, and the circuit *IV* characteristics.

D. Circuit Watermarking Methodology

1) Design and Implementation of Circuit Watermark: Once the Trojan hotspots of an analog design are identified using sensitivity analysis, the second stage involves insertion of circuit watermarks in these hotspots. The watermark can be inserted in any of the following nodes—1) PI nodes; 2) PO

Authorized licensed use limited to: Arizona State University. Downloaded on October 09,2024 at 01:01:59 UTC from IEEE Xplore. Restrictions apply.

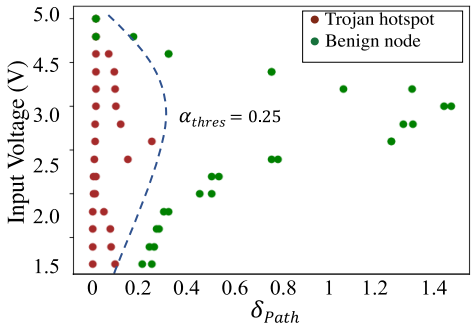

Fig. 10. Clustering analysis for determining Trojan hotspots.

TABLE III
EVALUATION OF THE SENSITIVITY THRESHOLD METRIC FOR DIFFERENT ANALOG BENCHMARK CIRCUITS

Benchmark	No. of paths	α_{thres}
Diff. amp.	3	0.25
Bandgap filter	15	0.46
OPAMP	7	0.98
SARADC	46	0.79

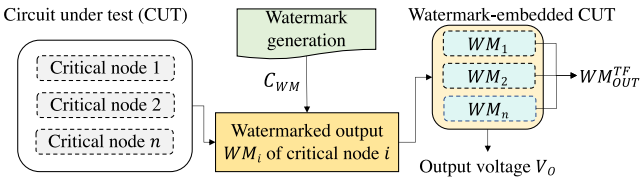

Fig. 11. Procedure for implementation of circuit watermarks in Trojan hotspots.

nodes; and 3) intermediate input and output nodes of the CCBs.

The proposed circuit watermark-based Trojan signature detection technique is illustrated in Fig. 11. A suitable watermark should have the following characteristics.

1) *Low Impact:* The watermark should have negligible impact on the circuit functionality in terms of output current and voltage characteristics and frequency response.
2) *Unique Signature:* We insert a final watermarked output pin WM_{OUT} by combining all the individual watermark outputs from the critical nodes. The WM_{OUT} should generate a unique signature at the circuit output that is indicative of malicious component insertion via analog Trojans.
3) *Low Area Overhead:* The watermark should be designed in a way such that it has a small footprint.

One efficient way to implement an analog circuit watermark is to use pass transistor-based transmission gates. Pass transistor logic (PTL), also known as transmission gate logic, uses the combination of a pMOS transistor and an nMOS transistor to allow signals to pass through when it is turned ON. The gate of each transistor is connected to a control signal, which allows the transistors to operate either in the linear region or the saturation region. The drain (source) and source (drain) of the nMOS (pMOS) are connected to the input (output) and output (input) nodes of the CUT, respectively. A schematic of the PTL-based watermark is illustrated in Fig. 12.

Under normal operation of a Trojan-free (TF) analog circuit, the output of the critical node will pass through the input of

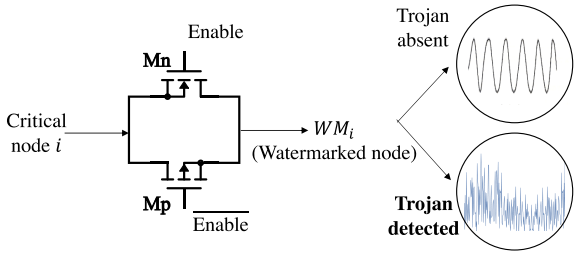

Fig. 12. Schematic of a PTL-based watermark.

the watermark. The watermark output, WM_i^{TF}, $1 \leq i \leq n$, is then further processed to generate WM_{OUT}^{TF}, which is used as a reference to detect unauthorized, malicious modifications in the CUT. However, if a Trojan is stealthily inserted in one of the least sensitive nodes of the circuit, it may remain dormant during normal operation. Upon Trojan activation, the attack payload passes through the input of the circuit watermark, and is detected at the final watermark output WM_{OUT}^{TI}. As discussed in Section IV, TC Trojans remain dormant for a long period of time before they get triggered. However, as the PTL-based watermarks are already inserted into the critical nodes of the circuit, they can effectively detect such Trojans by continuously monitoring the circuit operation. Hence, we are able to promptly detect these Trojans with large-delay triggers.

We keep Enable at the transistor gate (shown in Fig. 12) activated during normal circuit operation. This allows the outputs of the critical nodes to pass through the watermark and be captured at WM_{OUT}^{TI}. In other words, $V_{\text{out}} = V_{\text{in}}$, $V_{\text{Enable}} = V_{DD}$, where V_{out} and V_{in} are the input and output of the watermark, respectively, and V_{DD} is the supply voltage. However, keeping the gate activated for the entire duration of the detection process may cause leakage of current from the gate to the intermediate nodes of the circuit, leading to unwanted voltage deviations in these nodes. To address this behavior, we carefully size the nMOS and pMOS transistors of the PTL-based watermarks to operate in the saturation range and minimize any impact on the circuit performance. For both pMOS (threshold voltage $V_{t,p}$) and nMOS (threshold voltage $V_{t,n}$) to operate in the saturation range, $V_{gs,p} < V_{t,p}$ and $V_{gs,n} > V_{t,n}$, where $V_{gs,p}$ and $V_{gs,n}$ are the gate-to-source voltages of pMOS and nMOS, respectively.

2) Watermarking for Trojan Detection: Let us consider an analog CUT with n least sensitive paths obtained from sensitivity analysis. Let the customized circuit watermark be given by C_{WM}. Note that we do not insert the watermark directly in the insensitive (functional) path; instead, we draw out separate paths from each insensitive path, and insert the customized watermark C_{WM} in these auxiliary paths. Thus, for n such insensitive paths, we obtain n distinct watermark outputs WM_i, $1 \leq i \leq n$. Finally, all the n watermark outputs are combined to generate a final watermarked output at a watermark output pin, WM_{OUT}, where $WM_{\text{OUT}} = \sum_{i=1}^{n} WM_i$. We use a cascaded differential amplifier to generate the final watermarked output. The cascaded differential amplifier is a configuration of multiple differential amplifier stages connected in series. The schematic of the cascaded differential amplifier is illustrated in Fig. 13. The voltage signals from each of the watermark outputs are fed as inputs to each differential input pair. As the signals pass through each

Authorized licensed use limited to: Arizona State University. Downloaded on October 09,2024 at 01:01:59 UTC from IEEE Xplore. Restrictions apply.

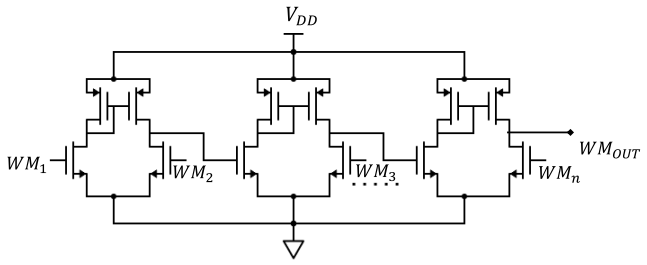

Fig. 13. Schematic of a cascaded differential amplifier circuit.

differential amplifier stage, they are amplified, making them discernible at WM_{OUT} when any unauthorized modification takes place. For a circuit having n number of Trojan hotspots, the optimal number of stages of the cascaded differential amplifier circuit is given by: $\lceil \log_2 n \rceil + k$, where k is a constant that can be fine-tuned based on the desired performance of the specific circuit that we are watermarking. The watermarked CUT has two output pins.

1) *Circuit Output* (V_O)*:* This pin represents the PO of the analog CUT.
2) *Watermarked Output* (WM_{OUT})*:* This pin represents the combination of all the watermark outputs that are obtained by placing circuit watermarks in the insensitive paths. The WM_{OUT} pin is crucial as it detects stealthy Trojan insertion even if it is not reflected in V_O.

We establish the following conditions that ensure the robustness of the watermark-based Trojan detection procedure.

1) $WM_i^{TF} \neq WM_i^{TI} \quad \forall\ 1 \leq i \leq n$, where WM_i^{TF} and WM_i^{TI} indicate the ith watermarked node of a TF netlist and a TI netlist, respectively.
2) $WM_{\text{OUT}}^{TF} \neq WM_{\text{OUT}}^{TI}$, where WM_{OUT}^{TF} and WM_{OUT}^{TI} indicate the final watermarked node of a TF netlist and a TI netlist, respectively. This ensures that the final watermarked node generates a unique signature that can detect the Trojans in the analog circuit.

Traditional methods often rely on storing the reference (golden) TF signature within the IC for comparison with the generated output. However, storing the golden signature on-chip can make it susceptible to tampering. In contrast, the proposed watermarking approach performs deviation detection in WM_{OUT}. For certain analog circuits such as a differential amplifier, there may exist twin nodes that carry similar signals, which can be used as an on-chip golden signal. This is called self-referential Trojan detection, but it is not applicable to all designs. This article aims for a generalizable methodology, and thus, we analyze the watermark output signal at WM_{OUT} externally to detect Trojan-induced deviations. In Fig. 14, we provide a schematic of the watermarked single-stage differential amplifier circuit.

E. Security Analysis

The proposed method provides resilience against the following security vulnerabilities.

Case 1 (Trojan Insertion Attack): The presence of customized watermarks in the duplicate paths enables detection of Trojan insertion attempts. The watermark output pin is continuously monitored to detect signs of Trojan activity. Any deviation in WM_{OUT} that is different from the TF output

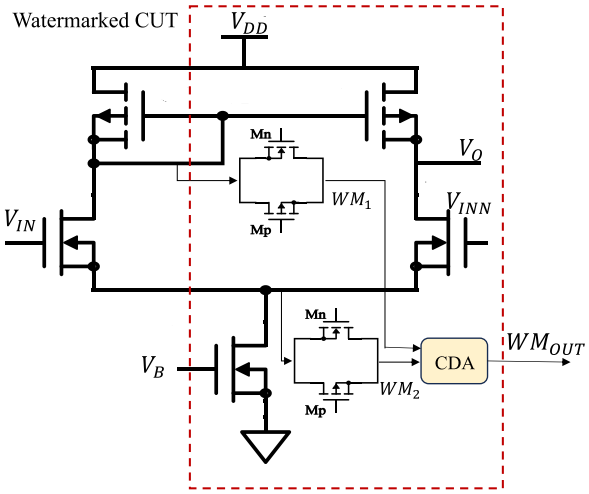

Fig. 14. Implementation of Trojan detection logic in a differential amplifier circuit (CDA: cascaded differential amplifier).

behavior is flagged, indicating that a malicious component has been added to the circuit.

Case 2 (Attacker Removes C_{WM}; Keeps Wires Open): If an attacker removes all/some of the watermarks placed in the wires drawn from insensitive paths, those specific wires will be open. This effect is immediately captured at WM_{OUT}, indicating tampering in the analog CUT.

Case 3 (Attacker Removes C_{WM} and Stitches the Wires): Note that the watermarks are not placed in the insensitive path; instead, they are placed in a separate path (say X) that has been drawn out from the insensitive path. These watermarks are detectable through X without directly interfering with the original circuit functionality. If an attacker removes the watermark and reconnects X, the watermarking design is still effective as the critical nodes are still observable at the watermark output pin. Hence, we are able to detect both watermark removal and Trojan insertion attempts.

Case 4 (Attacker Removes the Cascaded Differential Amplifier Circuit): In the watermarked CUT, the cascaded differential amplifier is directly connected to the final watermark output pin WM_{OUT}. By removing the differential amplifier circuitry, the attacker breaks the connection between the amplifier output and the WM_{OUT} pin. This removal effect will lead to an immediate change in the voltage at the WM_{OUT} pin, indicating that the CUT has been tampered with.

Case 5 (Ghost Watermark Attack): A *ghost watermark* attack is implemented by an adversary in an untrusted foundry by storing a golden watermark signature within the chip memory, and connecting the memory output to the watermark output pin. This attack can be achieved by completing removing the connection between the existing cascading differential amplifier circuit and the watermark output pin. The ghost watermark attack involves mimicking the behavior of the genuine watermark signal using the stored golden watermark signature. To address this concern, we provide a detailed analysis explaining why such an attack is not feasible within our specific Trojan detection framework.

1) We can put constant monitoring checks at WM_{OUT} by simply putting random test input voltages (other than the

Authorized licensed use limited to: Arizona State University. Downloaded on October 09,2024 at 01:01:59 UTC from IEEE Xplore. Restrictions apply.

TABLE IV
COMBINING SENSITIVITY ANALYSIS AND CIRCUIT WATERMARKING FOR TROJAN DETECTION IN ANALOG CIRCUITS

Benchmark	Trojan Location	No. of paths	Input Range (V)	Sensitivity analysis		Watermarked CUT				
				$\mathcal{S}_c$	$\mathcal{N}_c$	WM_{OUT}^{TF}	A2		TC	
							WM_{OUT}^{TI}/V_O	t_d (s)	WM_{OUT}^{TI}/V_O	t_d (s)
Differential amplifier	Q1(G)	1	(0, 5)	0.0115	1	4.98	3.45	4.95 μ	3.22	0.28
	Q2(G)	1	(0, 5)	1.1	0	4.98	1.518	4.3 μ	1.49	0.31
	Q2(D)	1	(0, 5)	1.1	0	4.98	1.507	4.27 μ	1.27	0.29
	Q3(G)	1	(0, 0.01)	0.0038	1	4.98	3.514	4.42 μ	1.506	0.36
Bandgap filter	CCB1	1	(0, 0.025)	0.23	1	2.2	1.22	0.028	1.56	0.557
	CCB3	1	(0, 0.73)	1.49	0	2.2	2.53	0.026	0.89	0.509
	CCB4	1	(0, 0.59)	0.13	1	2.2	0.65	0.025	0.8	0.504
	CCB5	4	(0, 2.5)	0.138	2	2.2	0.87	0.025	0.97	0.504
	CCB6	1	(0, 0.55)	0.151	1	2.2	0.52	0.032	0.48	0.512
	CCB7	4	(0, 2.5)	0.12	2	2.2	0.91	0.03	1.12	0.49
	CCB8	1	(0, 2.5)	1.38	0	2.2	2.526	0.022	1.89	0.556
	CCB9	3	(0, 2.5)	0.078	1	2.2	0.39	0.028	0.48	0.51
OPAMP	M1(G)	2	(0, 1.2)	0.6577	1	0.72	0.62	18.4 μ	0.55	0.27
	M2(G)	1	(0, 1.2)	1.3815	0	0.72	1.6	22.03 μ	0.69	0.275
	M3(S)	2	(0, 1.2)	1.1554	0	0.72	1.22	21.28 μ	0.574	0.32
	M4(D)	1	(0.5, 0.8)	0.8573	1	0.72	0.578	13.72 μ	0.435	0.31
	M5(S)	1	(0, 1.2)	1.2353	0	0.72	0.59	17.6 μ	0.556	0.275
SAR-ADC	CCB8	4	(0, 0.75)	0.4879	1	0.78	0.23	8.84 m	0.37	1.28
	CCB13	4	(0, 0.37)	0.432	1	0.78	0.138	10.08 m	0.42	1.3
	CCB15	4	(0, 0.8)	0.5694	2	0.78	0.72	10.08 m	0.68	1.19
	CCB17	3	(0, 0.8)	0.6577	1	0.78	0.42	11.45 m	0.75	1.3
	CCB190	2	(0, 0.75)	0.38	1	0.78	0.535	11.49 m	0.62	1.18
	CCB252	2	(0, 0.72)	0.5278	1	0.78	0.275	11.42 m	0.33	1.19
	CCB270	14	(0, 0.8)	0.751	4	0.78	0.139	10.09 m	0.25	1.29
	CCB271	13	(0, 0.8)	0.658	3	0.78	0.687	11.42 m	0.62	1.3

S_c: weighted sensitivity value of a path; N_c: number of critical paths; t_d: time taken for Trojan detection; WM_{OUT}^{TF}: final watermark output for Trojan-free design; WM_{OUT}^{TI}: final watermark output for Trojan-inserted design.

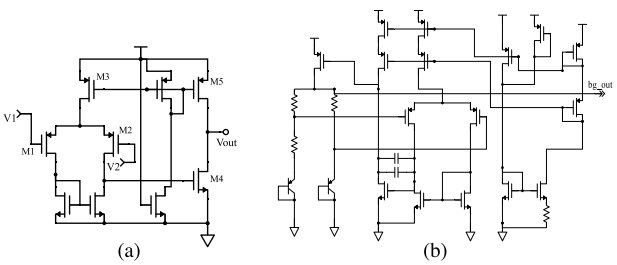

Fig. 15. (a) OPAMP. (b) Bandgap filter.

actual operational voltages used by the analog design). For different test input voltages, we observe different voltage signals at the watermark output pin. Now, if a ghost watermark is present, we will observe a constant voltage (golden watermark signature), irrespective of the test input voltage. This will detect an anomaly in the chip, making the attack infeasible. Note that we will use random test input voltages specifically for monitoring purposes to detect unexpected behavior, and not for actual circuit operation.

2) Given the continuous nature of analog signals, it is impractical to store the golden watermark values corresponding to every possible input voltage within the input range, due to limited storage capacity of the memory [24].

VI. EXPERIMENTAL RESULTS

A. Experimental Setup

We evaluate our method on four different analog benchmark circuits: 1) single-stage differential amplifier (as shown in Fig. 8); 2) bandgap filter; 3) operational amplifier (OPAMP); and 4) SAR-ADC. Fig. 15 provides a schematic of the evaluated benchmark circuits. These three circuits are widely used in real-life electronic systems [25]. All the circuits have been implemented and simulated using a HSPICE simulator. We converted the circuit-specific HSPICE netlists to the corresponding CCBs using the Synopsys Custom Compiler tool [26]. The training and evaluation of the FET-twin models and the ANT models are performed using the *Pytorch* framework. We evaluate the sensitivity values of the CCBs using the *torch.autograd* function from the *Pytorch* package [27]. The circuit watermarks are simulated using HSPICE and are embedded in the analog benchmark circuit design. The HSPICE simulations are run on a 2.4-GHz Intel Xeon Gold 5115 CPU with 768 GB of RAM. The training and inferencing of the FET-twins and the ANTs are carried out on NVIDIA GeForce RTX 1080 Ti GPU with 11-GB memory.

B. Evaluation Metrics

We use the following performance metrics to evaluate our analog Trojan detection framework.

1) $\mathcal{S}_c$ is the weighted sensitivity value of a particular CCB. If a CCB has n outgoing paths to other CCBs, the weighted sensitivity is calculated using: $\mathcal{S}_c=(1/n)\sum_{j=1}^{n}\mathcal{S}_j$, where $\mathcal{S}_j$ is the sensitivity value for the jth outgoing path.
2) $\mathcal{N}_c$ is the number of critical input/output nodes of the CCBs; the sensitivity values of these nodes should be less than α_{thres} in order for these nodes to be classified as critical. For a CCB consisting of n outgoing paths, $\mathcal{N}_c \leq n$.
3) t_d is the detection latency, i.e., the time between Trojan activation and Trojan detection. Note that the Trojan activation time depends on the stealthiness of the Trojan and the location where the Trojan is inserted. We aim to minimize t_d by embedding circuit watermarks in all the critical nodes and performing Trojan signature analysis at WM_{OUT}.
4) A_{det} is the Trojan detection accuracy of DAWN and is formulated as: $A_{\text{det}} = (m/n)$, where m is the number of times the Trojan is detected and n is the number of

Authorized licensed use limited to: Arizona State University. Downloaded on October 09,2024 at 01:01:59 UTC from IEEE Xplore. Restrictions apply.

TABLE V
LIST OF CCB CONNECTIONS IN THE BANDGAP FILTER

Input CCB	Output CCB
CCB1	CCB7
CCB6	CCB1
CCB7	CCB5
CCB{3, 9, 7, 5}	CCB4
CCB{8, 5}	CCB3
CCB{5, 9, 7, 4}	CCB8
CCB{7, 5}	CCB9
CCB9	CCB6

different locations in the analog CUT where the Trojan is placed.

C. Identifying Analog Trojan Hotspots

For our experiments, we evaluate three different Trojan designs: 1) A2; 2) TC, a large-delay Trojan; and 3) DELTA. Table IV presents the sensitivity evaluations of different Trojan locations in analog benchmark circuits and identifies critical nodes that are susceptible to Trojan insertion. For the single-stage differential amplifier circuit, the possible Trojan insertion points are between the PI and the gate of each PI transistor, namely, Q1, Q2, and Q3. Additionally, the drain of the PO transistor, Q2 can also be a potential Trojan insertion point. This is because the output of the differential amplifier is often a part of a larger circuit, which makes it a critical node for Trojan insertion. The weighted sensitivities of the critical paths are calculated using the ANT-based methodology by sweeping the test input voltages of Q1 and Q2 between (0 and 5 V). For every evaluated Trojan location, we determine the path from that location to the PO node of the circuit. We observe that the weighted sensitivities S_c of the paths Q1(G)-Q2(D) and Q3(G)-Q2(D) are less than α_{thres} (shown in Table III), thus, classifying them as critical. For OPAMP, we evaluate the sensitivities of the possible Trojan insertion points at the power supply nodes as well as the PI nodes (gate terminals) of transistors M1 and M2. We identify two unique paths from Trojan insertion location to the PO, that satisfy the condition $\mathcal{S}_c < \alpha_{\text{thres}}$ and are the potential locations where an analog Trojan may be maliciously inserted.

A bandgap filter is a larger circuit compared to the differential amplifier and has multiple subcircuits. Therefore, we break down the circuit into smaller parts that can be analyzed individually. We split the bandgap filter into nine individual CCBs using the Synopsys Custom Compiler tool, and determine the CCB connectivity graph. This gives us the number of outgoing paths from each CCB to a neighboring CCB. Table V lists the connectivity graph of a bandgap filter. Between adjacent CCBs, we develop FET twins of the PI, intermediate, and PO transistors, and generate the corresponding ANT model. For example, CCB5 has four outgoing paths to other CCBs. Therefore, we generate four distinct ANT models (one model for each path) and evaluate the path sensitivities.

SAR-ADC is widely used in programmable logic controllers and automotive SoCs [28], [29]. We use the SAR-ADC circuit provided by Synopsys for the simulation-based experiments [30]. The SAR-ADC comprises of both analog and digital modules. The analog core includes the digital-to-analog converter (DAC), comparator, DAC capacitor, and sample and hold circuitry. The digital core consists of the analog-to-digital interface, clock generator, and JTAG. We split the SAR-ADC circuit into 920 individual CCBs. Since we specifically focus on securing analog circuits from Trojan-induced malfunction, we perform sensitivity analysis and circuit watermarking only in the analog CCBs. Note that the digital portion of the SAR-ADC circuit is assumed to be protected separately using digital Trojan detection schemes [31], [32]. Table VI lists the critical paths that are marked as Trojan hotspots by the ANT model, in the bandgap filter and SAR-ADC circuits.

TABLE VI
LIST OF CRITICAL PATHS IN THE BANDGAP FILTER AND SAR-ADC CIRCUITS

Bandgap filter		SAR-ADC	
Input CCB	Output CCB	Input CCB	Output CCB
CCB1	CCB7	CCB8	CCB6
CCB4	CCB8	CCB13	CCB15
CCB5	CCB{3, 8}	CCB15	CCB{1, 17}
CCB6	CCB1	CCB17	CCB12
CCB7	CCB{4, 8}	CCB190	CCB300
CCB9	CCB8	CCB252	CCB234
		CCB270	CCB{246, 250, 260, 265}
		CCB271	CCB{247, 252, 255}

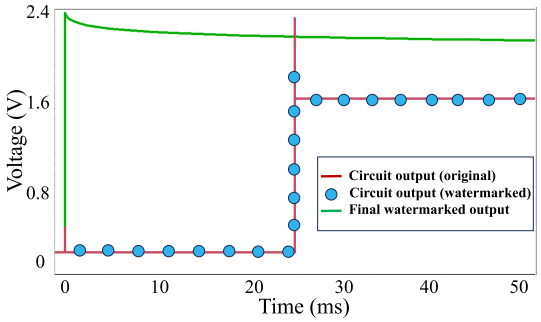

Fig. 16. Circuit outputs for the original and the watermarked scenarios in the bandpass filter circuit.

D. Watermarking-Based Analog Trojan Detection

Scenario 1 (Circuit Functionality Validation): In order to ensure that the proposed watermarking scheme does not impact the *IV* characteristics of the CUT or introduce significant delays in the circuit output, we perform HSPICE simulations of the original CUT and the watermarked CUT. The results are illustrated in Fig. 16. The selected PTL-based watermarks do not introduce substantial timing delays in the circuit. This shows that the chosen watermark circuits are curated specifically for the particular problem of detecting Trojans that are inserted either in sensitive or insensitive parts of the CUT, without impacting correct circuit functionality.

Scenario 2 (Detection of Removal Attack): We evaluate three different removal attack scenarios in a TF bandgap filter circuit: 1) attacker removes C_{WM} in the critical path CCB4→CCB8, keeping the wires open; 2) attacker removes all the watermarks; and 3) attacker removes the cascaded differential amplifier circuit. From Fig. 17, we observe that WM^{TF}_{OUT} shows significant deviation from the untampered analog CUT, indicating that the watermarking mechanism is effective in detecting removal attacks.

Scenario 3 (Detection of Ghost Watermark Attack): In Fig. 18, we show that the final watermark output voltage

Authorized licensed use limited to: Arizona State University. Downloaded on October 09,2024 at 01:01:59 UTC from IEEE Xplore. Restrictions apply.

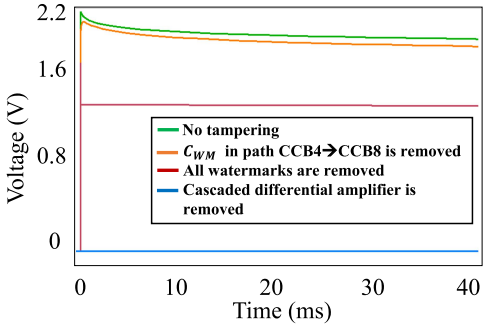

Fig. 17. Evaluating the final watermark node output WM^{TF}_{OUT} for different removal attack scenarios.

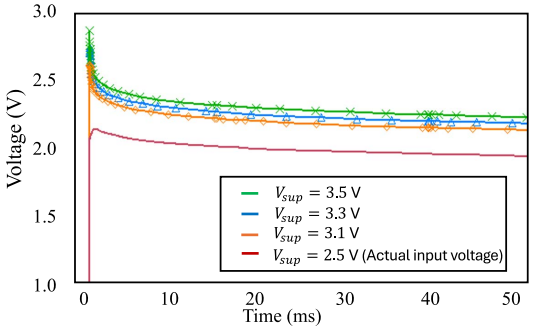

Fig. 18. Final watermark output behavior as a function of test input voltage V_{sup}.

behavior deviates with changes in one of the PIs, V_{sup}, for a watermarked TF bandgap filter circuit. Therefore, constant monitoring checks with random test input voltages can detect a ghost watermark attack.

Scenario 4 (WM_{OUT} Catches Trojan Behavior): Next, we evaluate the effectiveness of the final watermark output pin in capturing the Trojan behavior when it is stealthily inserted in one of the insensitive nodes of the circuit. We insert Trojans specifically in locations where connections between CCBs exist. Table IV shows the performance of the proposed method in terms of the detection latency t_d and the watermark output pin WM^{TI}_{OUT} behavior. In Table IV, WM^{TF}_{OUT} indicates the voltage at the final watermark output pin for a TF baseline circuit. Deviations from the TF behavior indicate possible Trojan insertion. Concurrently, we show that the PO voltage remains unaffected by the Trojan insertion. Fig. 19 illustrates this behavior; the Trojan behavior is captured at the watermark output pin 32 ms after the Trojan is inserted in one of the insensitive nodes of the circuit. The average t_d of the watermarked bandgap filter is 27 ms for the A2 Trojan and 0.517 s for the TC Trojan. The average t_d of the watermarked SAR-ADC circuit is 10.6 ms for the A2 Trojan and 1.25 s for the TC Trojan. Table VII shows the final watermark output voltages when the DELTA Trojan is inserted in specific locations of the analog benchmark circuits.

Scenario 5 [Circuit Output With and Without Trojan (PO Node Does Not Capture Stealthy Trojan Behavior in Insensitive Node)]: Based on condition 2) explained in Section V-D, we identify the Trojan signatures at the final watermarked output, WM^{TI}_{OUT}. Figs. 19 and 20 illustrate the detection of the A2 Trojan behavior at WM^{TI}_{OUT}, for the

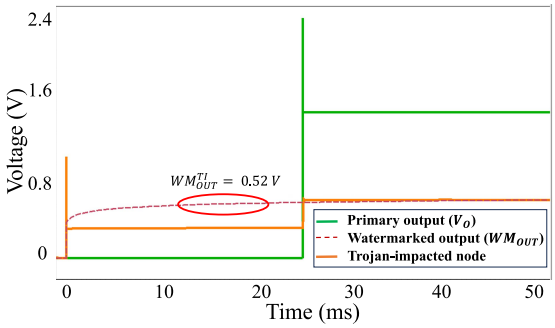

Fig. 19. A2 Trojan detection at the final watermarked output pin of a TI bandgap filter. Note that the PO V_O does not reflect the Trojan behavior, thus highlighting the effectiveness of the added watermark output pin.

TABLE VII
EVALUATING PROPOSED METHOD ON DELTA TROJAN

Benchmark	Trojan location	WM^{TF}_{OUT}	WM^{TI}_{OUT}	t_d(s)
Differential amplifier	Q1(G)	4.98	2.7	3.8 μ
	Q2(D)	4.98	1.43	4.3 μ
Bandgap filter	CCB4	2.2	0.83	0.023
	CCB6	2.2	0.66	0.025
SAR-ADC	CCB8	0.78	0.16	9.22 m
	CCB252	0.78	0.38	10.3 m

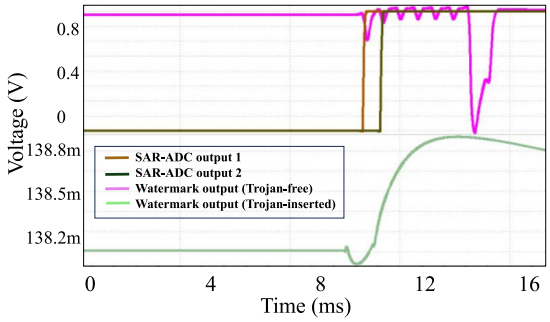

Fig. 20. A2 Trojan behavior captured by the watermark output pin upon Trojan activation in a SAR-ADC circuit. The Trojan is inserted in the insensitive path CCB13 → CCB15. Note that the POs of the SAR-ADC circuit (labeled as SAR-ADC output 1 and SAR-ADC output 2) do not reflect the Trojan behavior when it is inserted in the insensitive path.

TABLE VIII
TRANSISTOR COUNT AND POWER OVERHEADS FOR THE PROPOSED METHOD

Circuit	Original		Baseline [15]		**Proposed method**	
	#T	Power (W)	#T	Power (W)	#T	Power (W)
Diff. amp.	5	6.4 p	21	1.44 μ	11	11.2 p
Bandgap filter	39	86.4 μ	55	125.9 μ	54	123.4 μ
OPAMP	9	7.45 p	25	9.15 p	15	8.3 p
SAR-ADC	974	240.47 μ	990	356.64 μ	998	247.3 μ

bandgap filter and SAR-ADC circuits. For the noncritical nodes, marked by $\mathcal{N}_c = 0$, we perform the Trojan signature analysis at the PO of the circuit, V_O. While TC Trojans are accompanied by a longer trigger activation time compared to the A2 Trojan, the proposed method still effectively detects the TC Trojans.

E. Impact on Performance

Figs. 16 and 20 show that the watermarked CUT does not introduce substantial delays in the timing response compared

Authorized licensed use limited to: Arizona State University. Downloaded on October 09,2024 at 01:01:59 UTC from IEEE Xplore. Restrictions apply.

TABLE IX
DETECTION COMPARISON WITH BASELINE [15]

Circuit	n	Baseline [15]		**Proposed method**	
		m	$A_{det}(\%)$	m	$A_{det}(\%)$
Diff. amp.	4	3	75	4	100
Bandgap filter	8	6	75	8	100
OPAMP	5	3	60	5	100
SAR-ADC	36	21	58.3	34	94.4

n: total number of Trojan insertion points; m: number of times the Trojan is detected.

to the original TF CUT, and generates a similar output response. In other words, the watermarks have negligible impact on circuit performance.

F. Area Overhead

Table VIII compares the area overhead (in terms of transistor count #T) and the power (in W) overhead of the proposed method with a baseline Trojan detection technique [15]. We insert the POWER directive in the HSPICE netlist files of the analog benchmark circuits to calculate the power values. We observe that the proposed method has a lower area overhead and consumes less power, compared to the baseline. For SAR-ADC, which comprises both digital and analog designs, the area overhead is only 2.4% and power overhead is 2.8%. The area overhead compared to the original transistor count in the analog portion of the SAR-ADC is 4.15%. The evaluated analog circuits, namely, the differential amplifier, bandgap filter, and OPAMP, are typically part of larger mixed-signal SoC designs. Therefore, the total area overhead after adding the watermarks is still insignificant ($< 2.4\%$).

For the SAR-ADC circuit, the number of critical paths detected after sensitivity evaluation is 14. These critical paths are distributed among 20 analog CCBs. The number of CCBs comprising analog components is 476. Therefore, the percentage of CCBs that need to be secured using watermarks is 4.2%, which shows that even for larger designs, the number of Trojan hotspots is small. Although the SAR-ADC is composed of a considerable number of transistors, a relatively small fraction of CCBs is critical. Consequently, it becomes crucial to watermark all the critical paths within these CCBs to detect stealthy Trojan attacks, regardless of their insertion locations. Therefore, the importance of securing potential Trojan hotspots in these CCBs offsets the associated overhead.

G. Comparison With Existing Trojan Detection Method

We compare the Trojan detection performance of the sensitivity analysis-based framework with a baseline method that uses current sensors for Trojan detection [15]. We implement the current-sensing circuit proposed in [15] and simulate it using HSPICE. Table IX compares the Trojan detection accuracies A_{det} (in %) between the proposed method and the baseline method. For fair comparison, the total number and locations of A2 Trojan insertion points are kept same for both the methods and are chosen from Table IV. For all the evaluated Trojan insertion locations, our framework outperforms the baseline method. Unlike the baseline method, which is unable to detect an A2 Trojan when it is inserted in one of the least sensitive nodes [M1(G) and M4(D)] in the OPAMP, the proposed method successfully detects the Trojans, irrespective of their insertion points.

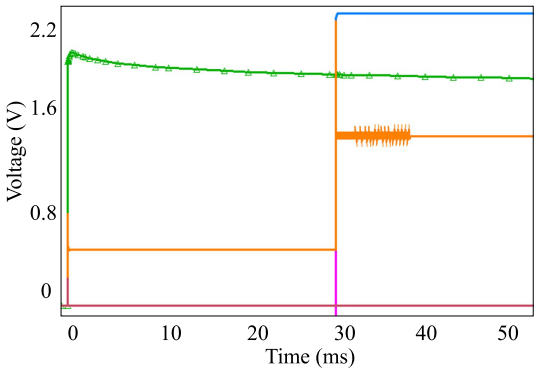

Fig. 21. Evaluating the impact of thermal noise in a bandgap filter: we introduce random noise generated by resistor models into various nodes of the bandgap filter and observe the impact on V_O (colored in orange, magenta, blue, and red lines) and WM_{OUT} (marked in green). For all the evaluated circuit locations, we observe a distortion in V_O, while WM_{OUT} exhibits negligible impact under the introduced thermal noise.

H. Impact of Noise on Watermarked Output

Trojan insertion has a nonlinear impact on the analog circuit behavior (see Figs. 19 and 20). For the SAR-ADC, when the A2 Trojan is inserted in one of the critical nodes (CCB13→CCB15), we obtain fewer pulses at the PO as well as at the critical node upon HSPICE simulation. This indicates a significant disruption in the normal circuit behavior. Fewer pulses signify loss of information, which is critical in SAR-ADC-based applications where accurate signal-to-data conversion is required. Trojan insertion in the nodes of the analog circuit impacts the dc operating point, thus impacting the small-signal parameters of the circuit. This effect is captured at WM_{OUT}. In analog circuits, noise is described as an additive term, with its effect being dependent on the specific type, e.g., flicker or thermal noise. While not causing a loss of data pulses, noise may introduce a distortion in the PO voltage V_o and/or WM_{OUT}. The nature of this distortion varies based on the type of noise but is generally considered linear. We use a voltage source and a resistor to simulate the thermal noise using the Johnson–Nyquist noise model, formulated as: $V_{tn} = \sqrt{4k_bTR'B}$, where k_b is the Boltzmann constant, T is the absolute temperature, R' is the resistance which is impacted with thermal noise, and B is the bandwidth across which we observe the thermal noise. Fig. 21 shows the impact of thermal noise on V_O and WM_{OUT} by sweeping R' value between 1 and 10 kΩ.

I. Discussion

As shown in Table IV, the number of critical nodes $\mathcal{N}_c$ increases with the size and complexity of a circuit. Therefore, it is important to effectively curate the circuit watermarks in the critical node locations that minimize the area overhead while maintaining the Trojan detection accuracy. Also, while Trojans placed in any location of a circuit is detected, we do not identify the specific location in the circuit where the Trojan is inserted, or diagnose the Trojan characteristics. In future work, we aim to use anomaly-based techniques for Trojan diagnosis, and explore techniques for optimizing the number of circuit watermarks for large circuits.

Authorized licensed use limited to: Arizona State University. Downloaded on October 09,2024 at 01:01:59 UTC from IEEE Xplore. Restrictions apply.

VII. Conclusion

We have performed sensitivity analysis to identify critical nodes in an analog CUT that are potential Trojan hotspots. The watermarked CUT effectively detects removal attacks as well as analog Trojans inserted by an untrusted foundry. We have evaluated our method on multiple analog benchmark circuits, using well-known analog Trojans. The ANT-based sensitivity analysis framework in conjunction with circuit watermarking outperforms a baseline Trojan detection method to provide a robust and secure defense for analog circuits.

References

[1] U. Guin et al., "Counterfeit integrated circuits: A rising threat in the global semiconductor supply chain," *Proc. IEEE*, vol. 102, no. 8, pp. 1207–1228, Aug. 2014.

[2] Q. Wang et al., "Hardware Trojans embedded in the dynamic operation of analog and mixed-signal circuits," in *Proc. NAECON*, 2015, pp. 155–158.

[3] Q. Wang, D. Chen, and R. L. Geiger, "Transparent side channel trigger mechanism on analog circuits with paast hardware Trojans," in *Proc. ISCAS*, 2018, pp. 1–4.

[4] K. Hasegawa, M. Yanagisawa, and N. Togawa, "Trojan-feature extraction at gate-level netlists and its application to hardware-trojan detection using random forest classifier," in *Proc. ISCAS*, 2017, pp. 1–4.

[5] J. Yang, Y. Zhang, Y. Hua, J. Yao, Z. Mao, and X. Chen, "Hardware trojans detection through RTLfeatures extraction and machine learning," in *Proc. AsianHOST*, 2021, pp. 1–4.

[6] T. Inoue et al., "Designing hardware Trojans and their detection based on a SVM-based approach," in *Proc. ASICON*, 2017, pp. 811–814.

[7] T. F. Wu, K. Ganesan, Y. A. Hu, H.-S. P. Wong, S. Wong, and S. Mitra, "TPAD: Hardware Trojan prevention and detection for trusted integrated circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 35, no. 4, pp. 521–534, Apr. 2016.

[8] K. Yang, M. Hicks, Q. Dong, T. Austin, and D. Sylvester, "A2: Analog malicious hardware," in *Proc. IEEE SP*, 2016, pp. 18–37.

[9] F. Courbon et al., "A high efficiency hardware Trojan detection technique based on fast SEM imaging," in *Proc. DATE*, 2015, pp. 788–793.

[10] Q. Shi et al., "Golden gates: A new hybrid approach for rapid hardware Trojan detection using testing and imaging," in *Proc. HOST*, 2019, pp. 61–71.

[11] T. Yang, A. Mittal, Y. Fei, and A. Shrivastava, "Large delay analog Trojans: A silent fabrication-time attack exploiting analog modalities," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 29, no. 1, pp. 124–135, Jan. 2021.

[12] J. Talukdar et al., "Automatic structural test generation for analog circuits using neural twins," in *Proc. ITC*, 2022, pp. 145–154.

[13] Y. Hou, H. He, K. Shamsi, Y. Jin, D. Wu, and H. Wu, "R2D2: Runtime reassurance and detection of A2 Trojan," in *Proc. HOST*, 2018, pp. 195–200.

[14] X. Guo, H. Zhu, Y. Jin, and X. Zhang, "When capacitors attack: Formal method driven design and detection of charge-domain Trojans," in *Proc. DATE*, 2019, pp. 1727–1732.

[15] M. Abedi et al., "High-precision nano-amp current sensor and obfuscation based analog Trojan detection circuit," in *Proc. ISCAS*, 2022, pp. 3324–3328.

[16] A. Pavlidis, M.-M. Louërat, E. Faehn, A. Kumar, and H.-G. Stratigopoulos, "SymBIST: Symmetry-based analog and mixed-signal built-in self-test for functional safety," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 68, no. 6, pp. 2580–2593, Jun. 2021.

[17] D. Forte et al., "Temperature tracking: An innovative run-time approach for hardware Trojan detection," in *Proc. ICCAD*, 2013, pp. 532–539.

[18] Z. Pan and P. Mishra, "Automated test generation for hardware Trojan detection using reinforcement learning," in *Proc. 26th ASP-DAC*, 2021, pp. 408–413.

[19] A. Chaudhuri, C.-Y. Chen, J. Talukdar, S. Madala, A. K. Dubey, and K. Chakrabart, "Efficient fault-criticality analysis for AI accelerators using a neural twin," in *Proc. ITC*, 2021, pp. 73–82.

[20] K. Kunal et al., "GANA: Graph convolutional network based automated netlist annotation for analog circuits," in *Proc. DATE*, 2020, pp. 55–60.

[21] M. Rostami, F. Koushanfar, J. Rajendran, and R. Karri, "Hardware security: Threat models and metrics," in *Proc. ICCAD*, 2013, pp. 819–823.

[22] N. Gupta et al., "DELTA: Designing a stealthy trigger mechanism for analog hardware Trojans and its detection analysis," in *Proc. 59th DAC*, 2022, pp. 787–792.

[23] L. Yang and C.-J. Shi, "Frosty: A fast hierarchy extractor for industrial CMOS circuits," in *Proc. ICCAD*, 2003, pp. 741–746.

[24] A. R. Díaz-Rizo, H. Aboushady, and H.-G. Stratigopoulos, "Leaking wireless ICs via hardware Trojan-infected synchronization," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 5, pp. 3845–3859, Sep./Oct. 2023.

[25] S. Sunter and P. Sarson, "A/MSbenchmark circuits for comparing fault simulation, DFT, and test generation methods," in *Proc. IEEE ITC*, 2017, pp. 1–7.

[26] "Custom Compiler." Synopsys. Accessed: Apr. 10, 2023. [Online]. Available: http://bit.ly/40MNuXS

[27] "Automatic differentiation package." PyTorch. Accessed: Apr. 10, 2023. [Online]. Available: http://bit.ly/411Nenz

[28] G.-Y. Huang, S.-J. Chang, C.-C. Liu, and Y.-Z. Lin, "10-bit 30-MS/s SAR ADC using a switchback switching method," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 3, pp. 584–588, Mar. 2013.

[29] Y.-H. Chung and Q.-F. Zeng, "A 12-bit 100-ks/s SAR ADCfor IoT applications," in *Proc. VLSI-DAT*, 2020, pp. 1–4.

[30] (Synopsys, Inc., Sunnyvale, CA, USA). *Analog-to-Digital Converters*. Accessed: Apr. 10, 2023. [Online]. Available: https://rb.gy/ld2kt

[31] J. Cruz, F. Farahmandi, A. Ahmed, and P. Mishra, "Hardware Trojan detection using ATPG and model checking," in *Proc. 31st Int. Conf. VLSI Design*, 2018, pp. 91–96.

[32] M. Rathmair, F. Schupfer, and C. Krieg, "Applied formal methods for hardware Trojan detection," in *Proc. ISCAS*, 2014, pp. 169–172.

Jayeeta Chaudhuri (Member, IEEE) received the B.E. degree from the Department of Electronics and Telecommunication Engineering, Jadavpur University, Kolkata, India, in 2020, the M.S. degree from Duke University, Durham, NC, USA, in 2023, and the first Ph.D. degree from Arizona State University, Tempe, AZ, USA, in 2023. She is currently pursuing the second Ph.D. degree with the Department of Electrical and Computer Engineering, Duke University.

Her current research interests include cloud FPGA security, analog IC security, and ML-assisted countermeasures for hardware Trojan detection.

Mayukh Bhattacharya received the B.Tech. degree from the Indian Institute of Technology Kharagpur, Kharagpur, India, in 1992, the M.S. degree in electrical engineering from Virginia Tech, Blacksburg, VA, USA, and the Ph.D. degree in electrical engineering and computer science from The University of Michigan at Ann Arbor, Ann Arbor, MI, USA, in 1999.

He has been with Synopsys, Sunnyvale, CA, USA, since 2003. He made many technical contributions to FastSPICE circuit simulator PrimeSim XA. As a Synopsys Scientist, he currently leads research and development teams for PrimeSim Custom Fault and PrimeSim Design Robustness products. He led the development of a FastSPICE option tuner that was one of the first AI/ML applications in EDA. His interests also include circuit optimization and analog hardware security. He has 11 granted (and 4 pending) patents, 7 journal papers, and around 20 conference publications. He has participated in multiple panels in DAC, International Test Conference, European Test Symposium, and HSPICE SIG in Machine Learning and Analog Test domains.

Krishnendu Chakrabarty (Fellow, IEEE) received the B.Tech. degree from the Indian Institute of Technology Kharagpur, Kharagpur, India, in 1990, and the M.S.E. and Ph.D. degrees from the University of Michigan at Ann Arbor, Ann Arbor, MI, USA, in 1992 and 1995, respectively.

He is the Fulton Professor of Microelectronics with the School of Electrical, Computer and Energy Engineering, Arizona State University (ASU), Tempe, AZ, USA, and the Chief Technology Officer with the Department of Defense Microelectronics Commons, Southwest Advanced Prototyping Hub. He is also the Director of the ASU Center on Semiconductor Microelectronics.

Prof. Chakrabarty is a Fellow of ACM and AAAS, and a Golden Core Member of the IEEE Computer Society.

Authorized licensed use limited to: Arizona State University. Downloaded on October 09,2024 at 01:01:59 UTC from IEEE Xplore. Restrictions apply.