ForASec: Formal Analysis of Security Vulnerabilities in Sequential Circuits

Faq Khalid, Student Member, IEEE, Imran Hafeez Abbassi, Semeen Rehman, Member, IEEE,
Osman Hasan, Senior Member, IEEE and Muhammad Shafique, Senior Member, IEEE

Abstract—Security vulnerability analysis of Integrated Circuits using conventional design-time validation and verification techniques is generally a computationally intensive task and incomplete by nature, under limited resources and time. To overcome this limitation, we propose a novel methodology based on model checking to formally analyze security vulnerabilities in sequential circuits considering side-channel parameters like propagation delay, switching and leakage power. In particular, we present a novel algorithm to efficiently partition the state-space into corresponding smaller state-spaces for faster security analysis of complex sequential circuits and thereby mitigating the associated state-space explosion due to their feedback loops. We analyze multiple ISCAS89 and trust-hub benchmarks to demonstrate the efficacy of our framework in identifying security vulnerabilities.

Index Terms—Formal Verification, Hardware Trojan, Model Checking,

1 INTRODUCTION

The increasing trend of outsourcing fabrication and split manufacturing to untrusted foundries and manufacturing plants has imposed significant threats to the security of critical applications [1]. In Hardware Trojan (HT) insertion attacks, an IC design is covertly altered with a malicious intent at some point during the design or manufacturing process [2]. HTs, may lead to three types of security attacks: (1) leakage of confidential information (confidentiality attack); (2) modifying the specifications or functionality (integrity attack); and (3) reduced reliability, degraded performance and destruction of a complete IC or its module (availability attack), as shown in Fig. 1. The effects can be catastrophic, such as system failure, and leakage of secret encryption keys, making it imperative to develop effective HT detection techniques, e.g., failure of ice-detection module in the P-8A Poseidon [3], [4].

Typically, HT detection techniques employ propagation delay [5], [6] and power [7], [8], [9] signature based analyses, which inherently pose the following limitations:

1) These techniques are based on extensive simulations or testing on real hardware systems, which require immense amount of time, cost and resources (as shown in Fig. 2), and therefore cannot provide full coverage for larger circuits. For example, in Fig. 2, the circuit $F$ (ISCAS S349) requires 51.2 seconds to test all input cases, i.e., $2^9 = 512$ (because total number of inputs are "9"), but to analyze the security vulnerabilities for all the possible gates (349) and nodes (369), it requires almost $3.18 \times 10^{96}$ years and $3.64 \times 10^{102}$ years, if 10 tests are performed in second, respectively. Moreover, with an increase in the number of inputs, even security vulnerability analysis with respect to input patterns is not feasible. For example, it requires almost 109 years to perform the security vulnerability analysis for circuit $H$ (ISCAS S349) with respect to all possible input patterns, i.e., $2^{35} = 34,359,736,368$ (because total number of inputs are "35")

2) The measurements acquired through sensors cannot encompass all the possible input conditions for larger ICs because of inherent data loss during the quantization in analog-to-digital conversion [10].

To address the above problem of high demand on the resources and the time of analysis during the design phase, different analytical techniques for vulnerability assessment against potential attacks have been developed. These techniques analyze the equivalent behavioral, functional and performance model against the design constraints and functionality characteristics [11], [12], [13], [14], [15]. These model-based simulations cannot guarantee complete coverage because of the computational constraints (energy and memory) [16] and floating point inaccuracies [17]. To address these issues, mathematical modeling and formal verification [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], (i.e., SAT solvers and model checking), based approaches...
provides the completeness and accuracy to some extent. However, these techniques deploy symbolic execution or fixed approximation of side-channel parameters. Therefore, they are unable to incorporate the parametric behavior, especially leakage power, and effects of process variations, which limits their scope to Active Hardware Trojans (AHTs), that remain active even under the normal operation.

1.1 Associated Research Challenges:

The aforementioned limitations of simulation-based, model-based and formal verification-based techniques pose the following research challenges:

1) **Accuracy**: How to incorporate the parametric behavior and effects of process variations to ensure an accurate security vulnerability analysis, even in the presence of Stealthy Hardware Trojans (SHTs, that remain inactive until they receive any activation trigger)?

2) **Completeness**: How to ensure 100% coverage in security vulnerability analysis for all gates and nodes with all possible input patterns?

3) **Feasibility**: How to reduce the analysis time while keeping 100% coverage for all gates and nodes with all possible input patterns?

1.2 Our Novel Contribution

To address these research challenges, we propose a novel and generic framework for formal analysis of security vulnerabilities (ForASec, Section 4) to perform the security analysis while considering the complete coverage of parametric behavior (i.e., leakage power, dynamic power and propagation delay) and process variations. The proposed ForASec consists of the following key components:

1) **Mathematical modeling (Section 4.1)** of basic gates, i.e., NOR, NAND and NOT, with respect to side-channel parameters while considering the process variations effects on different technology parameters, e.g., switch on resistance ($R_{on}$), oxide capacitance ($t_{ox}$), carrier mobilities ($\mu_n$ and $\mu_p$), gate ($C_G$), drain ($C_D$) and source ($C_S$) capacitances.

2) **Model checking based analysis methodology (Section 4.1)** to ensure the complete coverage of security vulnerability analysis, against the multiple intrusions at different locations, for all gates, nodes and with respect to all input patterns.

3) **A verification/analysis algorithm (Section 4.2)** to address the state-space explosion while considering the uncertain behavior of side-channel parameters.

   To evaluate the effectiveness of ForASec, we implement some of the basic sequential circuits and ISCAS89 benchmark in the presence of trust-hub Trojan benchmarks [29], [30], i.e., s35932-T100, s35932-T200, s35932-T300, s38417-T100, s38417-T200, s38417-T300, s38584-T100, s38584-T200 and s38584-T300. The experimental results demonstrate that ForASec is able to correctly identify the most vulnerable node(s) and the minimum possible size of SHTs that can be detected while analyzing the leakage power. Moreover, it also reduces the analysis time by 5 to 20 times (See Section 5.3).

2 Related Work

Typically, **model based simulations** are used for analyzing the security vulnerabilities [14], [15], but they cannot cover all possible test cases in complex systems because of their computational constraints (energy and memory) [15] and floating point inaccuracies [17]. To ensure the completeness and accuracy, mathematical modeling and formal verification based vulnerability analysis techniques have been proposed [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], as shown in Table 1. Although, to some extent, mathematical modeling can overcome above-mentioned limitations, it is still prone to human error and increases the design time. On the other hand, formal verification based approaches, i.e., SAT solving and model checking, can overcome the above stated limitations of simulation-based techniques by virtue of their inherent soundness and completeness [28].

The **SAT solver based approaches** are used for multi-stage assertion-based verification, code coverage analysis,
redundant circuit removal for isolation of suspicious signals and sequential automatic test pattern generations in vulnerability analysis [18], [19]. However, they provide the information about the satisfaction of certain property but in case of failure they are unable to identify the reason, thus lack completeness in vulnerability analysis [31].

On the other hand, functional [20], [21], [22] and behavioral [23] model checking based approaches provide the comprehensive vulnerability analysis and cover all possible input test cases but they inherently pose the state-space explosion problem for complex systems. To overcome this issue, several approaches for performance modeling have been proposed, which model the side-channel parameters, i.e., power [24], [25], [26], propagation delay [25], [26] and temperature [27], and analyze the vulnerabilities based on temporal properties [25], [26], [27] and property specification language [24]. These approaches provide the symbolic execution or fixed approximation of side-channel parameters. Therefore, they only target the security vulnerabilities against Active Hardware Trojans and may not be able to study the security vulnerabilities due to Stealthy Hardware Trojans (SHT). These SHTs remain undetected and inactive during the testing phase and even after deployment using external, internal or time-based triggers, they can initiate any of the drastic payloads as mentioned in Fig. 3. Symbolic side-channel parameters of these approaches makes the translated model deterministic, thus overshadows the uncertain behavior due to process variations. Moreover, most of these approaches are focused on the cryptographic modules but other modules can also be affected to launch Denial-of-Service attacks (DoS) or to disable the system. For instance, in Fig. 3 the control signals of HT-1, 2, 4 and 5 can be used for DoS attacks, as a kill switch or for controlling the computation and communication. These attacks can activate their respective payload even if they are not directly affecting the cryptographic components.

3 Preliminaries

In this section, we provide a brief overview of different triggering mechanisms and payload of HTIs, gate-level modeling and model checking.

3.1 Triggers and Payloads of HTs

Fig. 3 shows different types of triggering and payload mechanisms for hardware Trojans, e.g., the payload of HTs 1, 2, 4 and 5 is to hack the signals C and ER, which can be any control signal having the combinational, sequential (counter), asynchronous (counter) and hybrid triggering mechanism, respectively. However, payload of Trojan 3 is to leak the information via side-channel parameters which in this case is dynamic or leakage power. Therefore, HTs can broadly be categorized as: (1) Functional HTs, which change the system functionality by addition or deletion of modules with malicious intent and (2) Parametric HTs, which reduce reliability, increase the likelihood of system failure, and modify the physical parameters, such as power consumption and resulting in faster aging than expected.

3.2 Model Checking and nuXmv

Model checking [32] is primarily used as a verification technique for reactive systems by translating them into a corresponding state-space model and temporal properties. Although it provides automatic and exhaustive verification but in complex systems the state-space grows exponentially, which makes it computationally impossible to explore the entire state-space with limited resources. This problem, termed as state-space explosion, is usually resolved by using efficient algorithms and bounded model checking (BMC) [32]. In this paper, we use an open-source symbolic model checker, nuXmv [33], supports rational number analysis and property language specification using computation tree and linear temporal logic, to facilitate the modeling and verification of designs that exhibit continuous behaviors.

3.3 Gate-Level Performance Parameters

In this paper, we employ the following multiple side-channel parameters for gate level modeling of complex sequential circuits:

- Dynamic Power is the power dissipated while charging or discharging of load capacitances associated with transistors, nodes and wires and usually modeled as:
  \[ P_{\text{switching}} = \alpha C_{\text{total}} V_{dd}^2 f \],
  where \( \alpha \), \( f \), \( V_{dd} \), \( C_{\text{total}} \) are the switching activity, operating clock frequency, supply voltage, and the total capacitance that is charged or discharged in a transition, respectively.

- Subthreshold Leakage Power is dissipated in an IC primarily due to the undesirable flow of subthreshold current in the channel from \( V_{dd} \) to ground nodes, when transistors are in \( \text{Off} \) state. In this paper, we modeled it as:
  \[ P_{\text{Leakage}} = V_{dd} L \cdot I_{\text{Leakage}} \].
  Equation 1 shows the subthreshold leakage current of a MOSFET per transistor width, where \( W \) is gate width, \( L \) is the effective channel length, \( n \) is subthreshold slope factor, \( C_{oz} \) is oxide capacitance,
\[ \phi_t = KT/q \] is thermal voltage, \( \mu \) is effective carrier mobility, and \( \sigma \) is drain induced barrier lowering (DIBL) factor.

\[ I_{\text{leakage}} = 2.n.\mu.C_{\text{ox}} \frac{W}{L} \phi_t^2 \exp(\frac{\sigma.V_{dd} - V_{th}}{n.\phi_t}) \]  

**Propagation Delay** is defined as the (dis)charging time of the internal parasitic and external load capacitances, it is estimated as \( Elmore \) delay based on individual input transitions \( t_{\text{delay}} = \ln 2.\tau_{\text{elmore}} \), where \( \tau \) is equivalent to the first-order time constant \( \tau_{\text{elmore}} = \sum_{k=1}^{N} R_{ik} C_k \), where \( C_k \) is the capacitance at node \( k \), and \( R_{ik} \) is shared resistance among the paths from root to node \( k \) and leaf \( i \).

### 4 ForASec: Analysis Framework

ForASec analyzes the security vulnerabilities in a sequential circuit by translating it into its corresponding state-space model. Fig. 4 shows the different phases of vulnerability assessment in the proposed framework, which are explained below in Sections 4.1 and 4.2.

#### 4.1 State-Space Modeling and Translation

In the first phase, ForASec translates the circuits under test into their corresponding state-space and behavioral/functional/performance properties (see Fig. 4). For this translation, we developed the side-channel and technology parameters-based models for all the universal 2-input gates, i.e., NAND, NOR and NOT, that are used to model the multi-input complex gates and modules. For illustration purposes, we describe the two input NAND gate model with different parameter values based on their transistor level structure, shown in Fig. 5.

**Dynamic power** for 2-input NAND is modeled as \( P_{\text{switch}} = \alpha.C_{\text{total}}.V_{dd}^2.f \), where switching activity \( \alpha \) is computed based on the input/output transition probability and the total capacitance at the output of NAND is equal to \( C_{\text{total}} = C_{\text{load}} + C_{\text{diff}} \). The load capacitance \( C_{\text{load}} \) at the output of a single gate is the sum of gate capacitances of individual gates connected at the output node \( C_{\text{load}} = \sum_{i=1}^{N} C_{\text{DpMOS}_i} + \sum_{j=1}^{M} C_{\text{npMOS}_j} \). However, the gate capacitance for an individual pMOS/nMOS transistor is \( C_{\text{MOS}} = FO.WR.(C_{GSO} + C_{GDO} + W_{\text{min}}.L.C_{\text{ox}}) \), where \( FO \), \( WR \), \( C_{GSO} \), \( C_{GDO} \), \( W_{\text{min}} \), \( L \) and \( C_{\text{ox}} \) represent fanout, width ratio, overlap capacitances, minimum width, effective length, and oxide capacitance per unit area of the gate, respectively.

The internal diffusion capacitance \( C_{\text{diff}} \) depends on both the sidewall perimeter \( PS \) and area \( AS \) of the source (drain) diffusion region and modeled as
This resistive behavior of the transistors is modeled as the total capacitances via RC tree path from drain to source, defined as the time required for charging and discharging the transistors to incorporate the effects of process variations. It is computed as:

\[
C_{\text{diff}} = (2. FO.W_{p\text{min}}.W_{n\text{min}}.C_{p\text{admin}}) + (FO.W_{n\text{min}}.W_{p\text{min}}.C_{n\text{admin}}),
\]

where \(C_{p\text{admin}}\) and \(C_{n\text{admin}}\) represent the minimum intrinsic capacitance for the pMOS and nMOS transistors, respectively. The minimum drain diffusion capacitance \(C_{d\text{min}}\) for a single transistor is calculated as:

\[
C_{d\text{min}} = A.C_{jbd} + P.C_{jbd}d_{w},
\]

where \(C_{jbd}\) is the capacitance per unit area between body and bottom of the drain, and \(C_{jbd}d_{w}\) is the capacitance per unit length of the junction between body and side walls of the drain. Similarly, the minimum source diffusion capacitance \(C_{s\text{min}}\) can be computed as well.

We have modeled the propagation delay as Elmore’s delay estimation because it considers the linear behavior of transistors to incorporate the effects of process variations. It is defined as the time required for charging and discharging the total capacitances via RC tree path from drain to source. This resistive behavior of the transistors is modeled as:

\[
R_{on} = \frac{1}{\mu.C_{ox}.W_{p}(V_{th}-V_{gs})}.
\]

We have modeled the resistances for individual pMOS and nMOS transistors in accordance with the configuration of universal gates on different inputs. Table I indicates the Elmore delay relation for 6 possible transitions.

We have modeled the leakage current at the gate level using the components mentioned in Equation 1. The total leakage in an IC is the sum of leakage power of individual nMOS and pMOS transistors that are in the Off state, which essentially depends on the input vector. In CMOS based ICs, half of the total transistors are always in the Off state for any given vector. The stacking of two Off transistors in series significantly reduces the sub-threshold leakage compared to a single Off transistor. The two nMOS transistors in the NAND gate, shown in Fig. 5, are in series and as a result of the stack effect on input vector 00, the total leakage current reduces by a factor \(10^{-\frac{(V_{th}-V_{gs})}{V_{ts}}})\) by virtue of Kirchhoff’s current law. The leakage power for the two input NAND gates can be modeled with respect to the input states as shown in Table 2.

In case of intrusion at the output of any gate, the total capacitance \(C_{\text{total}}\) of the gate can increase with the factor \(C_{\text{intr}}\), which is the gate capacitance of the intruded gates. \(C_{\text{intr}} = C_{\text{intr1}} + C_{\text{intr2}} + ... + C_{\text{intrn}}\). Thus, the total capacitance \(C_{\text{total}}\) at the output of the gate is changed to:

\[
C_{\text{total}} = C_{\text{load}} + C_{\text{diffusion}} + C_{\text{intr}}.
\]

This increase in capacitance can affect the overall performance of the gate in terms of side-channel parameters. For example, if the intruded gate is within the propagation path then it has a major effect on the propagation delay and dynamic power, otherwise its effect is more on switching and leakage power compared to propagation delay.

4.2 Verification and Vulnerability Analysis

In the second phase, ForASec verifies the translated model and intrudes it to analyze the vulnerable locations (see Fig. 4). First, the functional, behavior and performance tests are performed to ensure the correctness of the translated model. We assume that an attacker has access to the netlist, therefore, to model this behavior, a different number of intruded gates are applied at different locations in the model to generate counterexamples for the given functionality, behavior and performance. These counterexamples are used to analyze and identify the vulnerabilities for undesired behavior, caused due to malicious alterations.

We propose a generic set of functional, behavioral and performance properties, based on the circuit under test (CUT), to analyze the vulnerabilities against intrusions based on the performance bounds of the circuits, which are calculated by applying different distributions of technology parameters. It is important to note that the lower bounds for dynamic power and path delays are not considered because these values are always zero in the steady state condition. The above-mentioned bounds are computed under the following definitions:

1. The upper bound of dynamic power is modeled as:

\[
\sum_{i=1}^{n} DP_{\text{max}},\text{where } DP_{\text{max}} \text{ is the maximum dynamic power of an } i^{th} \text{ gate.}
\]

2. The upper bound of the delay of the \(i^{th}\) path is modeled as:

\[
\sum_{i=1}^{n} Delay_{\text{max}},\text{where } Delay_{\text{max}} \text{ is the maximum delay of the } i^{th} \text{ gate in the selected path.}
\]

3. The upper/lower bound of leakage power is modeled as:

\[
\sum_{i=1}^{n} LP_{\text{max}/\text{min}},\text{where } LP_{\text{max}/\text{min}} \text{ is the maximum/minimum leakage power dissipated by the } i^{th} \text{ gate.}
\]

Algorithm for Security Vulnerability Analysis: it analyzes the state-space model of the given circuit and generates the corresponding counterexamples using the set of

| Algorithm 1 : Security Vulnerability Analysis |
|---|
| **Input:** Maximum dynamic power \(DP_{\text{max}}\) and leakage power \(LP_{\text{max}}\), Minimum leakage power \(LP_{\text{min}}\) and Maximum delays for k paths of IC \(d_{1\text{max}}...dk_{\text{max}}\) |
| **Output:** Vulnerable locations |
| 1: while Property II = TRUE do // upper bound of dynamic power |
| 2: Generate counterexample using Property II |
| 3: Add counterexample as an exception; |
| 4: end while |
| 5: while Property IV = TRUE do // upper bound of propagation delay |
| 6: Generate counterexample using Property IV |
| 7: Add counterexample as an exception; |
| 8: end while |
| 9: while Property VI = TRUE do // upper/lower bound of leakage power |
| 10: Generate counterexample using Property VI |
| 11: Add counterexample as an exception; |
| 12: end while |

---

**Fig. 6. Modeling of a 2-input NAND Gate with intrusion**
linear temporal logical (LTL) properties. These properties are based on the upper and lower bound of side-channel parameters, like propagation delay, dynamic and leakage power. To capture the effect on dynamic power, we propose property $\text{(I)}$ which states that there exists a state in which dynamic power is out of upper bound, which can be true if any intrusion has a significant impact on dynamic power. However, if the property is false then it indicates the scenario in which either there is no intrusion in the IC or the dynamic power fails to detect the HTs.

$$F(DP \neq 0 \rightarrow DP_{\text{MAX}} < DP_1 + DP_2 + \ldots + DP_n)$$  (I)

Once the unwanted behavior of intrusions is identified (Property $\text{(I)}$ holds), we take the complement of Property $\text{(I)}$ to generate the error trace, as shown in Property $\text{(II)}$

$$G(DP \neq 0 \rightarrow DP_{\text{MAX}} >= DP_1 + DP_2 + \ldots + DP_n)$$  (II)

The counterexample driven analysis is performed to identify the vulnerable locations. After analyzing the counterexample, it is added as an exception in Property $\text{(I)}$ to generate Property $\text{(III)}$. This process repeats until the Property $\text{(I)}$ becomes false, as shown in line 1 to 4.

$$F((DP \neq 0) \& \& (C_1 \& \& \ldots \& \& C_n) \rightarrow DP_{\text{MAX}} < DP_1 + \ldots + DP_n)$$  (III)

Similarly, this iterative process is applied on the following set of LTL properties to analyze effects on the propagation delay and leakage power.

$$F(D(k) = 0 \rightarrow D(k)_{\text{max}} < D_1(k) + \ldots + D_n(k))$$  (IV)

$$G(D(k) = 0 \rightarrow D(k)_{\text{max}} >= D_1(k) + \ldots + D_n(k))$$  (V)

$$F(LP_{\text{MAX}/\text{MIN}} < / > LP_1 + LP_2 + \ldots + LP_n)$$  (VI)

$$G(LP_{\text{MAX}/\text{MIN}} >= / <= LP_1 + LP_2 + \ldots + LP_n)$$  (VII)

For complex and larger circuits, the number of variables in the model along with the individual gates increases, which augments the complexity, and thus the above-mentioned counterexample analysis may take a longer time. However, the proposed algorithm and the modular structure of gate models reduces the complexity by allowing to construct and analyze the dynamic power, leakage power and delay models separately.

## 5 RESULTS AND DISCUSSIONS

To illustrate the practicability and usefulness of our ForASec framework, we evaluate it on a set of ISCAS89 benchmarks and some basic sequential circuits, like D-Flipflop (DFF), Shift register and counter, with intruded gates (Table 1) and trust-hub Trojan benchmarks. The following tools are used for experimentation:

1) Yosys for gate level translation of provided netlist or verilog code [34].
2) verilog2smv for gate level translation to corresponding smv mode [35].
3) nuXmv Version 1.1.1 for model checking and performance analysis.

All the simulations are executed in Centos 7 on a computing machine with Core i7 processor @3.4 GHz and 16 GB memory. For each benchmark, we consider two cases:

1) Case 1: the malicious gates are inserted randomly along the IC paths at the input, output and within critical and non critical paths.
2) Case 2: the malicious gates are inserted parallel to the IC paths.

As an example, Table 4 shows the variations in leakage power caused by single gate intrusion in benchmark circuits. To analyze the multiple counterexamples that either can be generated by single or multiple intrusions, we used the iterative Algorithm $\text{(III)}$.
5.1 Modeling and Analysis of Benchmark Circuits

The ForASec analysis of the implemented circuits is performed in two steps: (1) accuracy analysis of model translation and (2) vulnerability analysis against multiple intrusions. The correctness of the translated model is ensured by verifying all the functional, behavioral and performance characteristics.

For example, in Fig. 7 CLK-to-Q delay of a DFF is equivalent to the combined delay of NAND1 and NAND3, hold time of a DFF is equivalent to the inverter delay and the setup time of DFF is equivalent to the combined delay of NAND3 and NAND4. Similarly, the dynamic power, functionality and other characteristics are well defined in the literature. Therefore, to ensure the accuracy of model translation, the translated model of DFF must fulfill all the above-mentioned properties, design constraints and characteristics. Similarly, we translated multiple ISCAS89 benchmarks along with basic sequential circuits, i.e., a DFF, shift registers and counters, into their respective SMV models and verified them based on the above-mentioned properties, design constraints and characteristics.

5.2 Security Vulnerability Analysis of Benchmark Circuits:

Once the accuracy is ensured, then vulnerability of the translated model is analyzed by applying effects of multiple intrusions. For instance, Fig. 7 shows multiple parallel intrusions in a DFF, which can affect the side-channel parameters due to the loading effect.

We evaluated basic sequential circuits and ISCAS89 benchmarks against single/multiple gate intrusions and trust-hub Trojan benchmarks, i.e., s35932-T100, s35932-T200, s35932-T300, s38417-T100, s38417-T200, s38417-T300, s38584-T100, s38584-T200 and s38584-T300. Due to space limitation, it is not possible to discuss the vulnerable point in all circuits, therefore, in this section, the analysis of DFF and ISCAS89 benchmarks, i.e., s349, s35932, s38417 and s38584, against single/multiple gate intrusions and trust-hub Trojan benchmarks, is presented as these are some of the most complex circuits among the implemented ones. Table 5 shows the structural information of the implemented benchmarks.

| Benchmarks | Inputs | Outputs | Gates | N ANDs | N ORs | N OTs | Flip-flops |
|------------|--------|---------|-------|--------|-------|-------|------------|
| s349       | 9      | 11      | 63    | 41     | 41    | 179   | 15         |
| s35932     | 35     | 320     | 11025 | 1152   | 9045  | 1728  | 1636       |
| s38417     | 28     | 106     | 6204  | 2585   | 17830 | 15933 | 1492       |
| s38584     | 12     | 278     | 7642  | 3806   | 1152  | 63    | 1636       |

5.2.1 D flip-flop (DFF) against gate intrusions

We intruded at multiple locations, i.e., feedback paths/nodes, input nodes, output nodes and intermediate paths/nodes, as shown in Fig. 7. The feedback node in DFF is more vulnerable compared to other positions because of higher switching activity, which makes this node slightly unstable and less vulnerable with respect to intrusion. Moreover, the change in the loading effect of the feedback node can cause the setup time violation, which may result in data corruption.

5.2.2 ISCAS89 benchmark S349 against gate intrusions

For the vulnerability analysis, we intruded the S349 at multiple locations with different size (the number of intruded gates). Fig. 8 shows the effects of intrusion on the side-channel parameters but these variations can be overshadowed by the effects of process variations. Therefore, before intruding the circuits, we analyzed the effects of process variations, shown in Fig. 8 and analyzed their combined effects, which exhibit the following key observations:

1) The effect on dynamic power due to the process variations overshadows the effects of intrusions if the Trojan size is less than or equal to 7 gates (i.e., 2% of S349), as shown in Fig. 8(a) (see label DP1). However, beyond that the effects of intrusions are dominant and varies at different locations. For example, the dynamic power is affected by almost 11%, when 10 gates are intruded within the critical path. However, it is affected by almost 5%, when 10 gates are intruded at the input locations. The reason behind this behavior is that, in S349, there are no DFFs at the input, but the intrusion at intermediate stages involves effects in the feedback path of DFF, which starts the unwanted switching activity. Similar behavior can be observed in case of parallel gate intrusion (see label DP2).

2) The effect on leakage power due to the process variations overshadows the effects of intrusions if the Trojan size is less than or equal to 9 gates (2.5% of S349), as shown in Fig. 8(b) (see labels LP1 and LP2). The reason behind this behavior is that most of the physical parameters are affected by the process variations.

3) The effect on leakage power due to gate intrusions is very dominant, especially, when the intrusions are within the path, as shown in Fig. 8(c) (see label PD1). However, if intrusions are in parallel then the effect remains within the range of the process variations till the intrusion size is 6, as shown in Fig. 8(c) (see label PD2).

As a consequence, it can be concluded that input nodes are the most vulnerable point in the sequential circuits because the effects of such intrusions can be overshadowed by process variations, except for DFF in which the most vulnerable nodes are feedback nodes.
Intrusions within Computational Paths

| Dynamic Power Analysis | Leakage Power Analysis | Propagation Delay Analysis |
|-------------------------|------------------------|---------------------------|
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

![Fig. 8. Effects of process variations and intrusions at different positions, i.e., input (I/P), Non-Critical Path (NCP), Critical Path (CP) and output (O/P), on side-channel parameters in S349, ISCAS89 benchmark length.](image)

### Dynamic Power Analysis

1. **S35932**

2. **S38417**

3. **S35854**

### Leakage Power Analysis

1. **S35932**

2. **S38417**

3. **S35854**

### Propagation Delay Analysis

1. **S35932**

2. **S38417**

3. **S35854**

### 5.2.3 ISCAS89 benchmark against the trust-hub Trojan benchmarks

In order to evaluate the robustness and scalability of ForASec, we analyzed three of the most complicated ISCAS89 benchmarks, i.e., s35932, s38417 and s38584, against the trust-hub benchmarks and identify their respective vulnerabilities that exhibit the following key observations.

1) **Dynamic power analysis**: The effects of all Trojan benchmarks in s35932, are significant even in the presence of process variation, except s35932-T100, as shown in Fig. 8 (See label 1). Similarly, the Trojan benchmarks for s35854, the of Trojan benchmarks s38584-T200 and s38584-T300 are overshadowed by the effects of process variations, as shown in Fig. 8 (See label 2). The reason behind this behavior is that these Trojans operate in parallel and do not change the activity on any node of the circuit. Thus, s35932-T100, s38584-T200 and s38584-T300 can be considered as powerful attacks for dynamic power-based detection schemes.

2) **Leakage power analysis**: Almost all the Trojan benchmarks have a significant impact on the leakage power, except s35932-T200, as shown in Fig. 9 (See label 3). The reason behind this behavior is that the size of this Trojan is very small (15 gates, 0.04% of s35932). However, when it gets activated, it increases the switching activity and reduces the delay by creating a parallel path to skip four gates. Therefore, it has a significant impact on the dynamic power and the propagation delay, as shown in Fig. 9. In short, this Trojan (s35932-T200), can be considered as the powerful attack for leakage power-based detection schemes.

3) **Propagation delay analysis**: Process variations overshadow the effects of almost all the Trojans, as shown in Fig. 8 (See labels 4 and 5). The reason behind this behavior is that all these Trojans are inserted in parallel to the computational paths. However, s35932-T200 and s38417-T200 exhibit significant impact on propagation delay. In short, these Trojans (s35932-T200 and S38417-T200) are the weak attacks for propagation delay-based detection schemes.

### 5.3 Performance Analysis

Algorithm 1 generates a number of counterexamples during the vulnerability analysis of an IC depending upon intruded gates, their size and locations. Moreover, the time required by the NuXmv model checker to generate a single counterexample depends on the number of gates in an IC, i.e., size of the state-space model. Table 6 shows the number of counterexamples (CE), their analysis time (T) in seconds (s), and memory (M) acquired in gigabytes (GB) for the 8 NAND gates’ intrusion (within a computational path) in the ISCAS89 benchmark S349. It shows that with our proposed algorithm, the analysis time and required memory is exponentially reduced.

### 6 Conclusion

In this paper, we propose a novel formal framework (ForASec), based on the side-channel parameter analysis,
TABLE 6
Performance Comparison with respect to Analysis Time (T(s)) and required Memory(M(GB)) for 8 gates intrusion (with computational path) in S349, ISCAS89 benchmark

| Parameter          | Location | # of CEs | Without Algorithm | With Algorithm |
|--------------------|----------|----------|-------------------|----------------|
|                    |          |          | T(s)              | M(GB)          | T(s)            | M(GB)          |
| Switching Power    | Input    | 221      | 3895             | 16             | 2465            | 16             |
|                    | Output   | 304      | 4379             | 16             | 2978            | 16             |
|                    | CF       | 121      | 21642            | 16             | 741             | 13.94          |
|                    | NCP      | 102      | 17452            | 16             | 548             | 12.77          |
| Leakage Power      | Input    | 29       | 4521             | 16             | 235             | 8.02           |
|                    | Output   | 31       | 3684             | 16             | 304             | 8.01           |
|                    | CF       | 179      | 38274            | 16             | 1099            | 15.17          |
|                    | NCP      | 164      | 30167            | 16             | 992             | 14.9           |
| Propagation Delay  | Input    | 269      | 44879            | 16             | 3117            | 16             |
|                    | Output   | 356      | 52897            | 16             | 3688            | 16             |
|                    | CF       | 124      | 25291            | 16             | 987             | 13.8           |
|                    | NCP      | 103      | 18694            | 16             | 754             | 12.74          |

REFERENCES

[1] H. Li et al., “A survey of hardware trojan threat and defense,” Integration, the VLSI Journal, vol. 55, pp. 426–437, 2016.
[2] S. Bhunia et al., “Hardware Trojan Attacks: Threat Analysis and Countermeasures,” vol. 102, no. 8. IEEE, 2014, pp. 1229–1247.
[3] J. Villasenor and M. Tehranipoor, “The hidden dangers of chop-shop electronics: Clever counterfeiters sell old components as new threatening both military and commercial systems,” IEEE Spectrum (cover story), 2013.
[4] S. Adee, “The Hunt for the Kill Switch,” IEEE Spectrum, vol. 45, no. 5, pp. 34–39, 2008.
[5] D. Ismail et al., “On detecting delay anomalies introduced by hardware trojans,” in ICCD. IEEE, 2016, pp. 1–7.
[6] G. Zarrinchian et al., “Latch-based structure: A high resolution and self-reference technique for hardware trojan detection,” Trans. C, vol. 66, no. 1, pp. 100–113, 2017.
[7] F. K. Lodhi et al., “Power profiling of microcontroller’s instruction set for runtime hardware trojans detection without golden circuit models,” in DATE. IEEE, 2017, pp. 294–297.
[8] T. Hoque et al., “Golden-free hardware trojan detection with high sensitivity under process noise,” JETTA, vol. 33, no. 1, pp. 107–124, 2017.
[9] F. K. Lodhi et al., “A Self-Learning Framework to Detect the Intruded Integrated Circuits,” in ISCAS. IEEE, 2016, pp. 1702–1705.
[10] M. Pelgrom, “Nyquist analog-to-digital conversion,” in Analog-to-Digital Conversion. Springer, 2017, pp. 285–403.
[11] R. S. Chakraborty, S. Pagliarini, J. Mathew, S. R. Rajendran, and M. N. Devi, “A flexible online checking technique to enhance hardware trojan horse detectability by reliability analysis,” IEEE Transactions on Emerging Topics in Computing, vol. 5, no. 2, pp. 260–270, 2017.
[12] S. P. Mohanty, A. Srivastava, S. Hu, and P. Ghosal, “Guest editorial–special issue on hardware assisted techniques for iot and bigdata applications,” 2017.
[13] R. Rad et al., “Sensitivity analysis to hardware trojans using power supply transient signals,” in Hardware-Oriented Security and Trust, 2008. HOST 2008. IEEE International Workshop on. IEEE, 2008, pp. 3–7.
[14] P. Mishra et al., “Security and trust vulnerabilities in third-party ips,” in Hardware IP Security and Trust. Springer, 2017, pp. 3–14.
[15] G. K. Contreras et al., “Security vulnerability analysis of design-for-test exploits for asset protection in socs,” in ASP-DAC. IEEE, 2017, pp. 617–622.
[16] O. Balci, “Verification validation and accreditation of simulation models,” in CWS. IEEE Computer Society, 1997, pp. 135–141.
[17] E. et al., “Design of embedded systems: Formal models, validation, and synthesis,” Proceedings of the IEEE, vol. 85, no. 3, pp. 366–390, 1997.
[18] X. Zhang et al., “Detecting Hardware Trojans in Third-Party Digital IP Cores,” in HOST. IEEE, 2011, pp. 67–70.
[19] L. Feiten et al., “Formal vulnerability analysis of security components,” Trans. CAD, vol. 34, no. 8, pp. 1388–1369, 2015.
[20] M. Rathmair et al., “Hardware Trojan Detection by Specifying Malicious Circuit Properties,” in EJEC. IEEE, 2013, pp. 317–320.
[21] F. K. Lodhi et al., “Formal Analysis of Macro Synchronous Micro Asynchronous Pipeline for Hardware Trojan Detection,” in NorCAS. IEEE, 2015, pp. 1–4.
[22] A. Nahiyani et al., “Avsm: a framework for identifying and mitigating vulnerabilities in fms,” in DAC. IEEE, 2016, pp. 1–6.
[23] S. R. Hasan et al., “Translating Circuit Behavior Manifestations of Hardware Trojans using Model Checkers into Run-Time Trojan Detection Monitors,” in Asian-HOST. IEEE, 2016, pp. 1–6.
[24] N. Veeranna et al., “Hardware trojan detection in behavioral intellectual properties (ips) using property checking techniques,” Trans. ETC, 2016.
[25] I. Abbasi et al., “Formal Verification of Gate-Level Multiple Side Channel Parameters to Detect Hardware Trojans,” in FTSSCS. Springer, 2016, pp. 75–92.
[26] C. S. Pasareanu et al., “Multi-run side-channel analysis using symbolic execution and max-smt,” in CSF. IEEE, 2016, pp. 387–400.
[27] Y. T. Ngo et al., “Hardware Property Checker for Run-Time Hardware Trojan Detection,” in Circuit Theory and Design, 2015. IEEE, 2015, pp. 1–4.
[28] R. Drechsler et al., Advanced Formal Verification. Springer, 2004, no. ISBN 9781402077210.
[29] H. Salmani, M. Tehranipoor, and R. Karri, “On design vulnerability analysis and trust benchmarks development,” in Computer Design (ICCD), 2013 IEEE 31st International Conference on. IEEE, 2013, pp. 471–474.
[30] B. Shakyia, T. He, H. Salmani, D. Forte, S. Bhunia, and M. Tehranipoor, “Benchmarking of hardware trojans and maliciously affected circuits,” Journal of Hardware and Systems Security, vol. 1, no. 1, pp. 85–102, 2017.
[31] F. S. Bao, C. Gutierrez, J. J. Charles-Blount, Y. Yan, and Y. Zhang, “Accelerating boolean satisfiability (sat) solving by common subclause elimination,” AI Review, pp. 1–15, 2017.
[32] O. Hasan et al., “Formal verification methods,” Encyclopedia of Information Science and Technology, IGI Global Pub, pp. 7162–7170, 2015.
[33] R. Cavada et al., “The nuXmv Symbolic Model Checker,” in CAV, ser. LNCS. Springer, 2014, vol. 8559, pp. 334–342.
[34] C. Wolf, “Yosys open synthesis suite,” 2016.
[35] A. Irfan, A. Cimatti, A. Griggio, M. Roveri, and R. Sebastiani, “Verilog2smv: A tool for word-level verification,” in Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 2016, pp. 1156–1159.

to analyze/identify the vulnerabilities in sequential circuits against multiple types of intrusions at different locations while considering the effects of process variations on technology parameters. For illustration, the proposed methodology has been applied on multiple ISCAS89 benchmarks and standard sequential circuits. The experimental results show that unlike existing techniques, it successfully identifies operational boundaries and vulnerable nodes, i.e., due to EC/PV the change i power is 2 to 5%, which must be considered while designing. The feedback nodes in a DFF are the vulnerable towards the side-channel attacks. Another key outcome is the number of gates required by an HT that can go undetected for a given design and variability conditions.
Faq Khalid (S’18) received the M.S degree in electrical engineering and the B.E. degree in electronics engineering from the National University of Sciences and Technology (NUST), Pakistan in 2011. He is currently pursuing the Ph.D. degree in hardware security with Computer Architecture and Robust, Energy-Efficient Technologies, Vienna University of Technology, Austria. He was a Research Assistant with the System Analysis and Verification (SAVe) Lab, NUST School of Electrical Engineering and Computer Sciences. His research interests include formal analysis and verification of embedded systems, hardware design security, design reliability of VLSI circuits, and machine learning. He is a member of the Pakistan Engineering Council. He has received the Quaid-e-Azam Gold Medal for his academic achievements, the Best Researcher Award at the SAVe Lab in 2014, and the prestigious Richard Newton Fellowship at DAC 2018. His research interests include formal analysis and verification of embedded systems, hardware design security, design reliability of VLSI circuits and machine learning.

Imran Hafeez Abbassi received his MS degree in Information Security from the National University of Sciences and Technology (NUST), Pakistan in 2011. He is currently pursuing his PhD degree in electrical engineering from the same institute, with research focused on hardware intrusion detection using parametric analysis. His research interests include hardware security, cryptography, formal verification and embedded system design. Mr. Abbassi is a member of Pakistan Engineering Council. He is also an awardee of PhD scholarship from Higher Education Commission.

Semeen Rehman (S11) received the PhD degree in computer science from the Karlsruhe Institute of Technology, Germany, in 2015. She is a post-doc researcher with the TU Dresden, since 2015. From 2005 to 2007, she was an information systems manager with Mardan Surgical Centre Pvt. (Ltd.), Pakistan. Her current research interests include cross-layer reliability modeling and optimization covering various system layers like compiler and run-time system, approximate computing and embedded systems. She received the CODES+ISSS 2011 and 2015 Best Paper Awards and several HiPEAC Paper Awards. She is a student member of the IEEE.

Osman Hasan received the M.Eng. and Ph.D. degrees from Concordia University, Montreal, Canada, in 2001 and 2008, respectively. Currently, he is an Assistant Professor at the National University of Science and Technology (NUST), Islamabad, Pakistan. He is the founder and director of System Analysis and Verification (SAVe) Lab at NUST, which mainly focuses on the design and formal verification of safety-critical systems, including e-health and digital systems. He has received several awards and distinctions, including the Pakistan’s Higher Education Commission’s Best University Teacher (2010) and Best Young Researcher Award (2011) and the President's gold medal for the best teacher of the University from NUST in 2015. Dr. Hasan is a senior member of IEEE, member of the ACM, Association for Automated Reasoning (AAR) and the Pakistan Engineering Council.

Muhammad Shafique (M11 - SM16) received the Ph.D. degree in computer science from the Karlsruhe Institute of Technology (KIT), Germany, in 2011. He was a Senior Research Group Leader with KIT for over five years. He was with Streaming Networks Pvt. Ltd., where he was involved in research and development of advanced video coding systems for several years. He is currently a Full Professor with the Institute of Computer Engineering, Department of Informatics, Vienna University of Technology, Austria. He is also directing the group on Computer Architecture and Robust, Energy-Efficient Technologies.

His research interests are in computer architecture, power- and energy efficient systems, robust computing covering various aspects of dependability and fault tolerance, hardware security, emerging computing trends, such as neuromorphic and approximate computing, neuroscience, emerging technologies, and nanosystems, self-learning and intelligent/cognitive systems, FPGAs, MPSoCs, and embedded systems. His research has a special focus on cross-layer analysis, modeling, design, and optimization of computing and memory systems covering various layers of the hardware and software stacks, as well as their integration in application use cases from Internet-of-Things, Cyber-Physical Systems, and ICT for Development domains.

He holds one U.S. patent and over 180 papers in premier journals and conferences. He is a Senior Member of the IEEE Signal Processing Society and a member of the ACM, SIGARCH, SIGDA, SIGBED, and HiPEAC. He received the prestigious 2015 ACM/SIGDA Outstanding New Faculty Award, six gold medals in educational career, and several best paper awards and nominations at prestigious conferences, such as DATE, DAC, ICCAD, and CODES+ISSS, the Best Master Thesis Award, and the Best Lecturer Award. He served as the TPC Co-Chair of ESTIMedia and LPDC, the General Chair of ESTIMedia, and the Track Chair at DATE and FDL. He has given several invited talks, tutorials, and keynotes. He has also organized many special sessions at premier venues, such as DAC, ICCAD, DATE, and ESWeek, and served as the Guest Editor for the IEEE Design and Test Magazine and the IEEE TRANSACTIONS ON SUSTAINABLE COMPUTING. He has served on the program committees of several IEEE/ACM conferences, such as ICCAD, ISCA, DATE, CASES, FPL, and ASPDAC.