Formal Verification of Gate-Level Multiple Side Channel Parameters to detect Hardware Trojans

Imran Hafeez Abbasi, Faiq Khalid Lodhi, Awais Mehmood Kamboh, and Osman Hasan

School of Electrical Engineering and Computer Science (SEECS)
National University of Sciences and Technology (NUST)
Islamabad, Pakistan
{imran.abbasi,faiq.khalid,awais.kamboh,osman.hasan}@seecs.nust.edu.pk

Abstract. The enhancements in functionality, performance, and complexity in modern electronics systems have ensued the involvement of various entities, around the globe, in different phases of integrated circuit (IC) manufacturing. This environment has exposed the ICs to malicious intrusions also referred as Hardware Trojans (HTs). The detection of malicious intrusions in ICs with exhaustive simulations and testing is computationally intensive, and it takes substantial effort and time for all-encompassing verification. In order to overcome this limitation, in this paper, we propose a framework to formally model and analyze the gate-level side channel parameters, i.e., dynamic power and delay, for Hardware Trojan detection. We used the nuXmv model checker for the formal modeling and analysis of integrated circuits due to its inherent capability of handling real numbers and support of scalable SMT-based bounded model checking. The experimental results show that the proposed methodology is able to detect the intrusions by analyzing the failure of the specified linear temporal logic (LTL) properties, which are subsequently rendered into behavioural traces, indicating the potential attack paths in integrated circuits.

Keywords: Model Checking, Hardware Trojans, Formal Verification, Side Channel Analysis, nuXmv, Gate Level Modeling

1 Introduction

The rapid scale growth of semiconductor design and fabrication technology has raised serious concerns about integrated circuits trustworthiness and security, particularly in the military and industrial applications [28, 3, 14]. The issue of hardware trust has become prominent in the recent years due to large scale outsourcing of IC fabrication to untrusted foundries, making them vulnerable to Hardware Trojans insertion [6]. Malicious intrusion in ICs may result in change of specifications or functionality, unreliability and degraded performance, and leakage of confidential information, such as encryption keys. The effects can be catastrophic, such as failure of critical avionics system, leakage of secret encryption keys, failing of defense satellite system [1, 21] and compromise of heterogeneous network of Internet of Things (IoTs) [24]. Hardware Trojans are generally

of two types: i) functional Trojans change the system functionality by addition or deletion of functional units in a circuit with malicious purpose and ii) parametric Trojans reduce reliability of the IC to increase the likelihood of system failure by modifying physical parameters, such as modifying the power consumption resulting in faster aging than expected.

Hardware Trojan detection schemes are broadly classified into logic based testing, side channel analysis and reverse engineering [5]. Logic based testing techniques uses generation of random test vectors and implementation of different methods to trigger the Trojan circuits and observe their effects at the output [9]. Side channel analysis is based on measuring the variations in observable physical parameters, such as delay, power, electromagnetic (EM) signal analysis and current sensing in order to detect any alteration with the structural characterization of the integrated circuit design [12]. Side channel analysis techniques are more commonly used because of their higher performance, relatively lower costs and nondestructive testing capabilities. Agarwal et al. proposed a power analysis based technique by applying random patterns at inputs of ICs under test and comparing their measurements with the power signature of a golden model [2]. The golden IC model is obtained from reverse engineering of limited number of ICs. Similarly, Wang et al. proposed an approach to generate average and covariance based power traces [29] employing the singular value decomposition (SVD) algorithm and eigenvector projection analysis, respectively, to detect the malicious intrusions. The focus of delay based detection techniques [15, 26, 18] is on the delay measurements of individual paths of the circuit due to activation of Hardware Trojans and their comparison with of delay fingerprints from golden ICs. These above-mentioned delay and power signature analysis techniques are based on extensive simulations or by testing on real hardware systems, which requires immense time, cost and resources. Moreover, the measurements acquired through sensors cannot encompass all the possible input conditions for larger ICs and result in an extensive amount of data, which is difficult to handle with conventional automation techniques [17].

Formal verification [13] can overcome the above stated limitations of simulation based techniques for Hardware Trojan detection by virtue of its inherent soundness and completeness. The formal verification based methods, such as SAT solving and Model Checking, have been used with the soft intellectual property(IP) of the IC to detect Hardware Trojans, provided that user has access to a hardware description language code or netlist of the IC. In the recent past, researchers have presented different frameworks for the formalization and verification of IP core security properties. Xuehui et al. proposed an approach that applies multistage assertion based verification, equivalence and code coverage analysis, redundant circuit removal for isolation of suspicious signals, and sequential automatic test pattern generations (ATPG) [33]. Lodhi et al. have proposed to utilize model checking for analyzing the delay based vulnerabilities in integrated circuits [19]. In this approach, the timing behaviour and functionalities of IC are translated into the corresponding state-space model and LTL properties, respectively. Rathmair et al. have presented a property checking based method

which verifies functional properties deduced from system specification using a model checker. The counterexample is subsequently analyzed to detect potential attack paths [27]. Ngo et al. have presented a methodology to use assertions derived from temporal logic and converting them into a synthesizable checker [23]. This method involves identification and verification of critical behavioral invariants using assertion based property specification language (PSL). The verified behavioral invariants are used to design the hardware property checker (HPC) which is subsequently integrated in ICs to verify the properties. However, to the best of our knowledge, so far no work is being reported, which considers the formal verification of performance properties to detect intentional malicious enhancement of hardware design. Moreover, the above-mentioned assertion based property checking methods are vulnerable to Trojan insertions at netlist and layout levels, and will only be able to detect functional Trojans.

In this paper, we present a generic framework based on the behavioral model of the IC to detect malicious hardware intrusions. We assume the attack model B [32], in which we have a netlist available in the form of trusted design, but the foundry is considered untrusted to which the design is outsourced for manufacturing. The attacker in the foundry can insert Hardware Trojans in the form of addition, deletion or modification of gates. The main idea is to translate the circuit netlist to a state transition system based model and verify it against the identified set of functional and behavioral properties that can be affected by any malicious modification in the IC. The model is then intruded with the expected malicious behaviour, and counterexamples are analyzed for deducing potential attack paths. On the basis of the information extracted from the detailed analysis of counterexamples, the designers can merge protection in the original design by embedding runtime hardware monitors. The proposed LTL properties are based on system functional and physical behavior. We propose to use the symbolic model checker nuXmv [8] for analysis by virtue of its ability to handle real numbers and implicit dealing of state counters.

The rest of this paper is organized as follows: Section 2 provides an overview of the nuXmv model checker and performance parameters used in our gate models. In Section 3, we explained the proposed methodology for hardware intrusion detection followed by our gate modeling in the nuXmv model checker in Section 4. In Section 5, we have given a case study for our proposed methodology. Section 6 presents the results followed by a comparison with some of the existing schemes in section 7. Finally, the paper concludes in Section 8.

2 Preliminaries

In this section, we give a brief introduction to the nuXmv model checker and the performance parameters, i.e., dynamic power and delay, that we have used for gate level modeling in our proposed Hardware Trojan detection scheme. The intent is to facilitate the understanding of the rest of the paper for both hardware security and formal methods communities.

2.1 nuXmv Model Checker

The nuXmv symbolic model checker [8] is a recently developed formal verification tool that extends the capabilities of NuSMV model checker [10], by supporting analysis of infinite state domains. It complements NuSMV's verification techniques by sharing basic functionalities, such as symbol table, boolean encoding of scalar variables, flattening of design, and representation of finite state machines at different levels of abstraction. Moreover, it inherits all the basic model checking algorithms from NuSMV for finite domains using BDDs and SAT. For infinite state transition models, it introduces new data types of unbounded *Integers* and *Reals* and it provides the support of Satisfiability Modulo Theories (SMT), using MathSAT [20], for the analysis of such kinds of designs. The system that is required to be modeled is translated into SMV language, which supports the modular programming approach. The entire system can be distributed into several modules that interact with one another in the `MAIN` module. The properties to be checked can be expressed in nuXmv using the Linear Temporal Logic (LTL) and Computation Tree Logic (CTL). The specifications are expressed in nuXmv with the help of logical operations like, OR (`|`), AND (`&`), Exclusive OR (`xor`), Exclusive NOR (`xnor`), equality (`<->`), implication (`->`), and temporal operators, like next (`X`), Globally (`G`), Finally (`F`) and until (`U`). Similarly, the CTL specifications can be written by combining logical operations with quantified temporal operators, like forall finally (`AF`), exists globally (`EG`) and exists next state (`EX`). It is also possible to analyze quantitative characteristics of the state transition system by specifying real-time specifications. Whenever a specified property is determined to be false, a counterexample is constructed and subsequently printed by nuXmv in the form of an error trace of the state space that falsifies the property. We have chosen the nuXmv model checker because it can effectively model continuous values of power consumption and path delays of any given IC.

2.2 Performance Parameters

Gate level characterization has effectively formed the basis of side channel Hardware Trojan detection schemes, which are based on characterizing each gate in terms of its physical and performance parameters. We adopted dynamic power consumption and path delay as the side channel performance parameters for malicious intrusion detection in any given circuit. Equation 1 represents the gate level switching power model [31] that is dependent upon the activity factor α, output capacitance C_L, supply voltage V_{dd}, which has quadratic effect on dynamic power, and operating frequency f. The activity factor is the switching probability that a node of a circuit transitions from 0 to 1, because that is the only time when dynamic power is consumed by the circuit in the CMOS technology. The total output capacitance is the sum of parasitic capacitance of the individual gate and load capacitances at the output node.

$$P_{switching} = \alpha C_{total} {V_{dd}}^2 f \tag{1}$$

We have estimated gate level delays based on individual transitions at the gate inputs using the Elmore delay model [31], which computes the delay by representing each circuit in the form of RC tree. The voltage source is the root of tree, and capacitors are leaves at the ends of the branches. The delay is estimated by the model from a source switching to one of the leaf nodes changing as the sum over each node i of the capacitance C_i on the node, multiplied by the effective resistance R_{is} on the shared path from the source to the node and the leaf. Equations 2 and 3 are used in formulation of the gate level delay model.

$$\tau_{elmore} = \sum_{i} R_{is} C_i \tag{2}$$

$$t_{delay} = \ln 2 \times \tau_{elmore} \tag{3}$$

3 Proposed Methodology

In this section, we describe our proposed generic framework for the detection of malicious intrusion in any given IC. Our methodology comprises of the following five steps as depicted in Fig. 1.

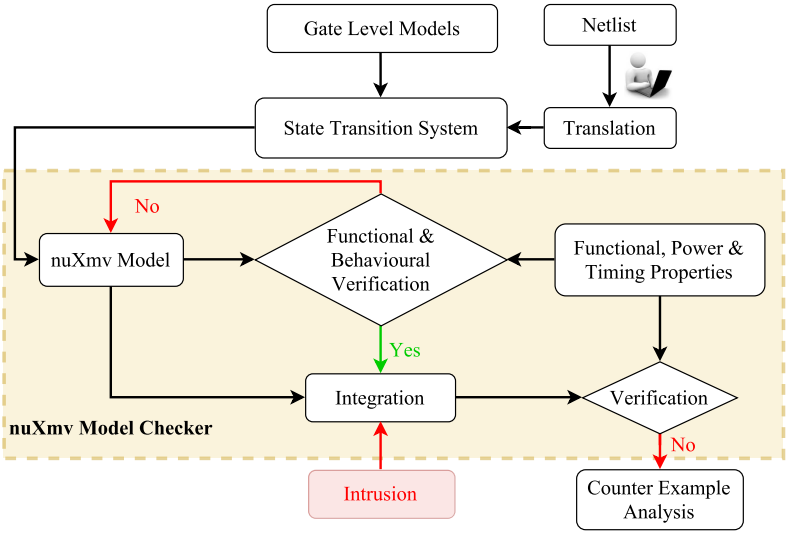


Fig. 1. Proposed Framework for Hardware Trojan Detection

1. The first step is to develop models for universal gates, including NAND, NOR and NOT. The advantage of these models is that we can build any other complex gate or a complete IC using these three basic gates. These generic models are technology independent and can be customized based on the characteristic parameters of a particular VLSI technology.
2. The next step is to develop a state transition system for any given netlist manually using the individual gate models. Based on the information in the

netlist, expressions are specified for computation of both power and individual path delays. The technology parameters, and individual gate models are passed to the main module for required computations.
3. The state-space model is verified in a model checker against LTL properties specified for the IC functionality and performance. The gate fanouts [25] are set to be of variable size, such that model checker can analyze all possible combination of gate sizes in a circuit. The minimum and maximum bounds for circuit power consumption and path delays are determined, which are used to examine the integrity of the circuit.
4. The behaviour of Hardware Trojan is integrated into the model of IC. The intruded model is subsequently verified against the specified power and timing LTL properties.
5. The verification of intruded model generates counterexamples, which are analyzed and translated into the potential attack paths in the IC.

4 nuXmv Modeling

In this section, we give the detailed description of the proposed modeling approach in our Hardware Trojan detection scheme.

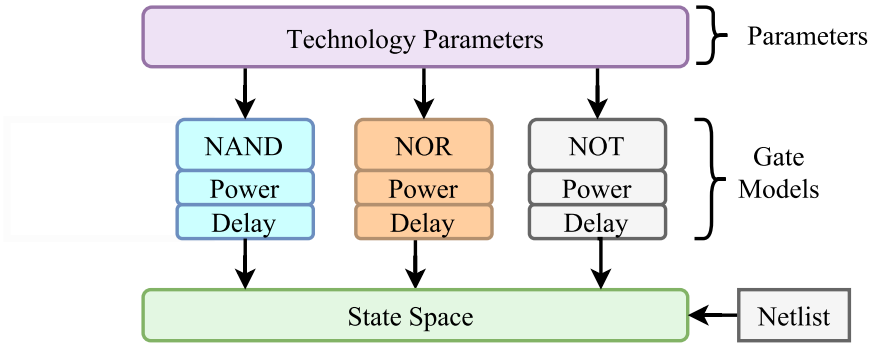


Fig. 2. Gate Level Modeling

4.1 Technology Parameters

The starting point of our work as shown in Fig. 2 is to identify the required parameters of target VLSI technology used in the manufacturing of an IC. An estimation of the gate level power consumption and delay needs parameters, such as minimum length and width of transistor's gate, source, and drain, electron and holes mobilities, threshold voltages, thickness oxide, and junction capacitances. These basic parameters can be obtained from the process specification document of the relevant technology or by plotting the DC and model parameters in a CAD tool, such as Cadence. We have defined a separate module, which uses basic parameters to calculate minimum values of MOSFET gate and drain capacitances along with the value of individual resistances of MOSFETS in the ON condition.

4.2 Universal Gate Models

Based on the technology parameters, we have developed models for the universal gates, i.e., NAND, NOR and NOT as depicted in Fig. 2, in order to estimate switching power and delay. These gate models can be in turn used to build more complex gates and circuit elements. The description of the NAND gate model is provided here and all the others have been developed similarly with different parameter values.

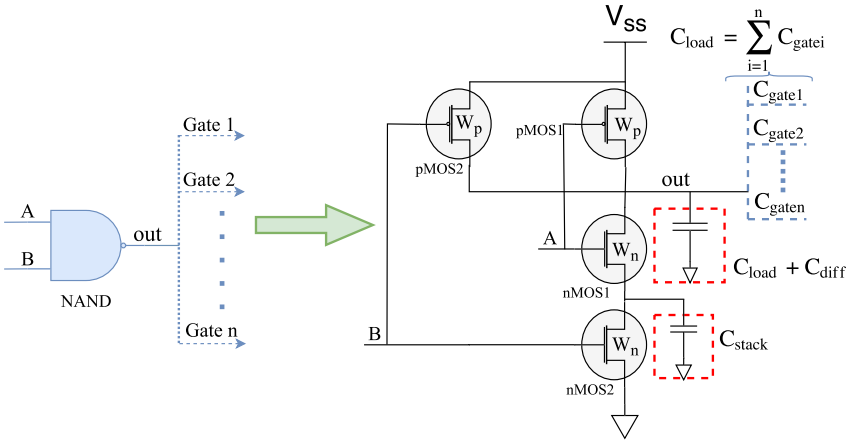


Fig. 3. Composition of two Input NAND Gate

A two input NAND gate is composed of two pMOS transistors, connected in parallel, and two nMOS transistors connected in series as shown in Fig. 3. The individual gate capacitances for pMOS and nMOS transistors are given in equations 4 and 5.

$$C_{gatepMOS} = fanout \times WR_{pMOS} \times C_{gminP} \tag{4}$$

$$C_{gatenMOS} = fanout \times WR_{nMOS} \times C_{gminN} \tag{5}$$

where WR is the width ratio and C_{gmin} is the minimum gate capacitance for pMOS and nMOS transistor. C_{gmin} is calculated from the oxide capacitance C_{ox}, minimum width W_{min} and length L of respective MOSFETS. The load capacitance C_{load} is the sum of gate capacitances of individual gates connected at the output node.

$$C_{load} = \sum_{i=1}^{p} C_{gatepMOSi} + \sum_{j=1}^{n} C_{gatenMOSj} \tag{6}$$

The diffusion capacitance for NAND gate is computed as:

$$\begin{aligned} C_{diffusion} = &\left(2 \times fanout \times WR_{pMOS} \times W_{minP} \times C_{dminP}\right) + \\ &\left(1 \times fanout \times WR_{nMOS} \times W_{minN} \times C_{dminN}\right) \end{aligned} \tag{7}$$

where C_{dmin} is the minimum diffusion capacitance of a MOSFET, calculated using area, sidewall perimeters and respective junction capacitances of the drain

diffusion region [31]. The total capacitance C_{total} at the output of an individual gate is computed by addition of $C_{diffusion}$ and load capacitance C_{load} as shown in Equation 8.

$$C_{total} = C_{load} + C_{diffusion} \tag{8}$$

The total power consumption of the NAND gate is determined using Equation 1. In order to determine the individual path delays in a circuit, we have used the Elmore delay model to calculate the individual gate delay on the respective input transitions as depicted in Table 1. An accurate estimation of the delay is performed by considering all possible transitions by taking into account the capacitances, which will change or remain constant. Our proposed approach of gate level modeling also considers the effects of charging and discharging of capacitances at the internal nodes. For instance, capacitance is required to be charged at the nMOS stack of the NAND gate when upper transistor is ON and the lower transistor is OFF. The total power consumption and path delay

Table 1. Elmore Delay Calculation NAND Gate

Input	**Output**	**Elmore Delay**
00	1	$(2 \times R_p \times C_{total})$ / (Fanout $\times$ $WR_{pMOS} \times W_{minP}$)
01	1	$(R_n \times C_{total})$ / (Fanout $\times$ $WR_{nMOS} \times W_{minN}$)
10	1	$(R_n \times (C_{total} + C_{stackN}))$ / (Fanout $\times$ $WR_{nMOS} \times W_{minN}$)
11	0	$(R_n \times C_{total})$ / (Fanout $\times$ $WR_{nMOS} \times W_{minN}$)

measurements are mainly dependent upon charging and discharging of individual capacitances in an IC. We illustrate this fact by considering a behaviour of a single inverter, which drives the load of two inverters connected at its output node as shown in Fig. 4. At input low, the gate capacitances of NOT2 and NOT3, i.e., C_{g2} and C_{g3} along with the diffusion capacitance C_{d1} of gate NOT1 are charged. The output node of NOT1 transitions to logic high depicted as *state* 1. The compute power(CP) and estimate delay (ED) flags are set to high, indicating the measurement of dynamic power and time required to charge capacitance. At the input high, same capacitance is required to be discharged and output node transitions to *state* 0. The state does not change if the input remains same, indicating no change in dynamic power or path delay. Similarly, behaviour of two input NAND and NOR gates can be represented with the state diagram comprising of four states, and each state having transitions to and from, all other states.

4.3 Netlist Translation

The translation of netlist is accomplished by interconnecting the individual gate modules. A particular gate module is defined by parameters, including variable

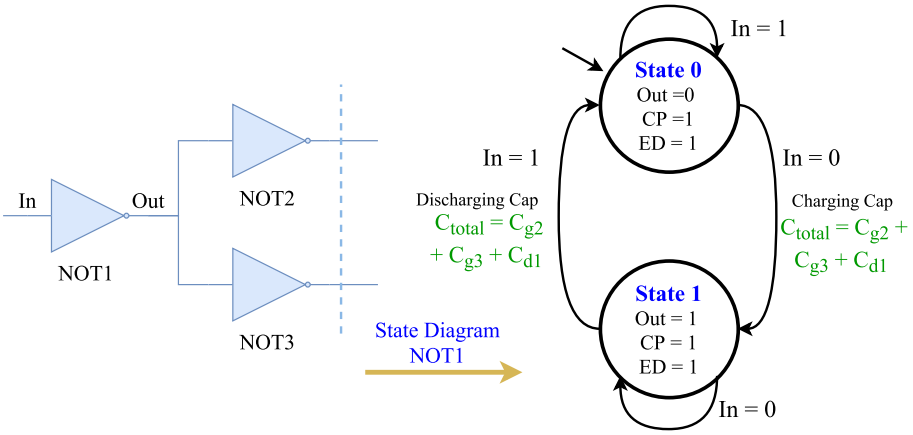


Fig. 4. The State-Space Model of an Inverter

inputs, transition probabilities, gate capacitances at output, fanout and relevant technology parameters. The transition probabilities along with outputs are passed to all gates connected at the output node of an individual gate. Other gates are constructed using the three basic gates, for instance, the AND gate is constructed using a NAND module followed by a NOT gate. Similarly, netlist of any integrated circuit can be manually translated using basic models of the three gates. The individual fanouts are swept across all the values in order to identify maximum and minimum bounds of switching power and individual path delays.

4.4 Property Specification

The verification of the IC model is carried out by validating the following properties using the bounded model checking (BMC) support for *real* numbers. The performance properties are validated using the nuXmv model checker to ascertain that the given IC remains between the specified boundaries defined for dynamic power and delay parameters. The undesired behaviour of the circuit due to any malicious alteration of circuit can be identified from the generated counterexample. The maximum and minimum bounds for power consumption are identified to validate the power property. The LTL specifications to validate the switching power is defined by adding the power consumed by individual gates in the circuit.

```
G(pwr_max >= gate1.pwr + gate2.pwr +...+ gaten.pwr >= pwr_min)    (I)
```

The attacker can intrude the IC by altering and modifying any of the individual path from input to output. Therefore, the characteristic delay of each individual path in an IC from input to output is required to be validated using the delay based properties. Suppose an IC has p number of paths from input to output, then properties to verify minimum and maximum delay for each of the i^{th} path

are required to be specified.

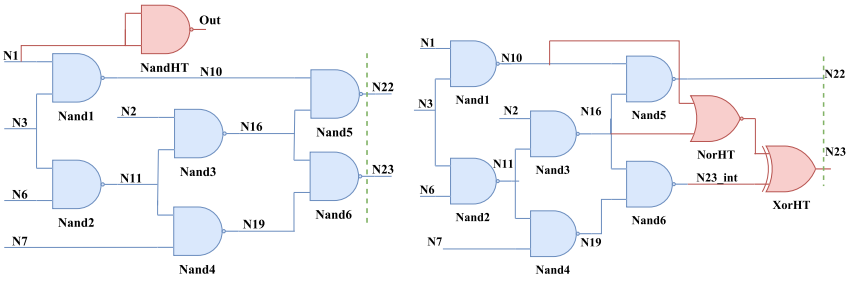
```
G(gate1(i).del + gate2(i).del +...+ gatek(i).del >= del_(path_i)min)  (II)
```

```
G(gate1(i).del + gate2(i).del +...+ gatek(i).del<= del_(path_i)max)  (III)
```

Any intrusion at the hardware level either affects power consumption of the entire circuit, or delay of an individual path which has been altered, or both. Whenever the defined bounds for maximum and minimum values of parameters are violated, a counterexample can be generated by the model checker indicating the existence of Hardware Trojans.

5 Case Studies

We illustrate usefulness of our proposed framework, by evaluating it on ISCAS85 benchmark circuit C17. We show two types of malicious intrusions on C17 given in [30], and [22] as depicted in Fig. 5a and Fig. 5b respectively. The procedure of modeling and identification of intrusions using nuXmv model checker is explained below.



(a) Power based intrusion (b) Delay based intrusion

Fig. 5. Intruded ISCAS-85 C17 benchmark circuit

5.1 Gate Level Models

The first step in our proposed framework is to acquire the basic gate level models. The ISCAS benchmark circuit C17 comprises of 6 two input NAND gates, and total number of 5 inputs. Only the NAND gate model is required, which constitutes of expressions for diffusion and load capacitances, required to estimate values for both power and path delay. Moreover, the activity factors for switching power computation are also determined for each gate in the circuit. The basic gate models are defined in separate modules in the nuXmv model checker.

5.2 State Space Modeling

The netlist gives the description of connectivity for C17 benchmark circuit, which is translated into the state space. For example, consider the NAND3 gate in Fig.

5a which has inputs N2, and NAND2.out (output of NAND2). The gate along with relevant parameters is described as:

```
NAND3:nand(N2, NAND2.out, 0.5, 0.5, NAND2.P0, NAND2.P1, Fanout3, par.Freq,
 par.Cgmin_p, par.Cgmin_n, par.Vdd, par.Wmin, par.Cdmin_p, par.Cdmin_n,
par.Csmin_p, par.Csmin_n,par.Rp, par.Rn, NAND5.Cgate, NAND6.Cgate,0, 0);
```

Using the given input signal probabilities of the circuit, we compute the probabilities and activity factor for its each node. For example, input N2 has an input probability of 0.5 for 0 and 1, $P0$ and $P1$ are the probabilities of the second input being 0 or 1, which can be used to calculate the activity factor at the pertinent node of circuit. This follows by the parameters like, operating frequency, input voltage V_{dd}, values of gate and diffusion capacitances, along with the values of ON resistances R_p and R_n for pMOS and nMOS transistors, respectively. The last part of the expression indicates the total load at the output node N16, which is the sum of gate capacitance of NAND5 and NAND6. Typically, gates have the maximum fanout of 4 and minimum fanout of 1 [25]. Similarly, all six NAND gates of C17 circuit are described in the main module of nuXmv to generate the formal model of the given circuit.

5.3 Model Verification

After the state space of the C17 benchmark circuit is defined, the next step is to check the functionality the circuit. There are a total number of $2^5 = 32$ possible input vector for C17 circuit. We verified the functionality of circuit by using certain number of input vectors. The next step is to identify the maximum and minimum bounds for switching power and delay. Our model accuracy requires the values to remain in between these bounds. The power for C17 circuit is maximum when all 6 NAND gates have a maximum fanout equal to 4. Similarly, the minimum bound is determined by computing the power with the minimum fanout of 1. For all sizes of the gates of the C17 benchmark, the power for C17 is required to remain in between the specified bound. The circuit has four individual paths from inputs to two output, and the bounds for the delay are identified for every single path by computing the combination of individual gate fanouts, which gives the maximum and minimum path delay for every path. The model is termed as verified if all the functional and performance properties are satisfied.

5.4 Hardware Intrusion and Verification

To present the effectiveness of model checking based Hardware Trojan detection technique, we used intruded versions of C17 benchmark given by Wei et al. [30] and Mukhopadhyay et al. [22]. The intrusion of a single two input NAND gate is depicted in Fig. 5a . The addition of the gate only affects the overall power consumption and does not affect its delay since it is not in the path from input to output. The state space for the intruded model is defined with the C17 benchmark circuit along with NANDHT gate. Therefore, when the intruded model

of the circuit is validated against the property defined for maximum power, it generates a counterexample. However, intruded model satisfies delay based properties since the NANDHT does not lie in the any of active paths of the circuit. The combinational Trojan in [22] is embedded in the C17 circuit with a NOR and XOR gate. Due to the inserted Hardware Trojan, the power consumption and delay of the circuit increases and LTL properties defined for the maximum power and delay fails.

5.5 Counterexample Analysis

The counterexamples generated by the verification of intruded circuits can be analyzed to identify potential locations of the malicious intrusion. Our proposed approach has an inherent advantage of compositional analysis, as shown in Fig. 6. If the power property defined for the entire IC fails, the analysis may be extended by partitioning the IC into distinct regions or components, and specifying the power properties for the individual parts to isolate Trojan-free and Trojan-inserted regions. For example, for analyzing the power property failure in Fig. 6, we divided the IC into four distinct regions in such a way that each region approximately has an equal number of gates. The power property for each region is verified and the intruded region is subsequently identified, i.e., Region 3. In order to identify the intruded component within the identified region we can further analyze the power property for each component, e.g., component 2 of Region 3.

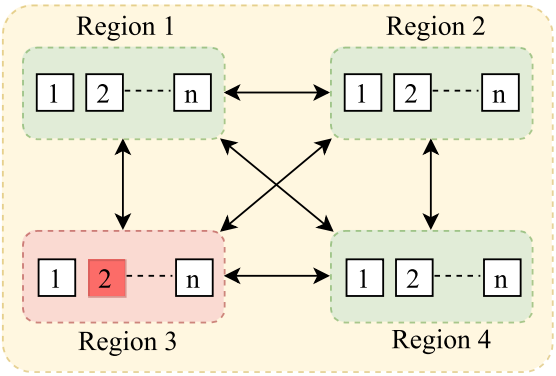


Fig. 6. Counterexample Analysis of IC

For instance, whenever input N1 in Fig. 5a switches from 0 to 1 or 1 to 0, the total dynamic power increases and corresponding property for the IC fails. On partitioning the circuit into different regions, each having two NAND gates, we can specify a power property for each individual partition. The first partition comprising of NAND1 and NAND2 will fail the power property on verification, indicating the presence of malicious intrusion along the input N1. Similarly, the delay based properties are specified for each path from input to output, and the failure of the corresponding property will indicate the path along which the

circuit has been intruded. The total delay of any path from input N6 to output N23 in Fig. 5b is the sum of the individual delays of NAND gates and intruded gates along the path, resulting in the failure of the specified delay property. The analysis of generated counterexamples for both power and delay can thus be used to identify the potential location of the intrusion in IC.

6 Results and Discussion

We used the version 1.0.1 of the nuXmv model checker along with the Windows 10 Professional OS running on a Core i7 processor, 2.67 GHz, with 6 GB memory for our experiments. We applied our verification methodology on different benchmark circuits as depicted in Table 2. The intruded variant of C17[30] has 7 NAND gates, and it has one additional gate due to which the dynamic power check fails. The power consumption at a particular node depends upon the sum of gate and diffusion capacitances. Whenever there is an activity at the input of intruded gate, switching power is consumed by delivering energy to charge capacitance at the output node, and then dumping this energy to ground. Our method is effective since we can detect any intrusion even if no load is driven by the intruded gate. The diffusion capacitance $C_{diffusion}$ depends on the size of source and drain region, with wider transistors having proportionally greater diffusion capacitances. We have successfully tested our technique by varying intruded gate sizes, since any transition at input, even with the minimum possible gate size, fails the defined power specifications. The delay property is validated in this case, since we have defined delay bounds on propagation delays of the individual logic paths, and malicious gate in this case does not contribute towards the charging and discharging of capacitances at particular nodes of any path. The technique presented in [30] performs the gate level characterization (GLC) with some error, i.e., GLC error for C17 is 0.0057 % [30], which increases with the number of gates. However, our proposed methodology accurately models the dynamic power and delay based gate level behaviour for C17.

The Hardware Trojan in [22] modifies the output at node $N23$ on rare input vectors. The total load capacitance at the output nodes $N23\_int$ and $N16$ increases due to the addition of gate capacitances of XOR and NOR gates. Consequently, the power consumption of entire circuit increases and the LTL property, defined for maximum power, fails when checked in nuXmv. Similarly, added capacitances contribute towards incrementing the propagation delays of the effected paths from input to output node $N23$. The defined LTL delay bounds in nuXmv fail, which indicates that the specified paths has increased delays.

We also tested our methodology on a full adder circuit made by universal gates. We inserted a two input XOR gate in the path of the carry out signal of the full adder with one input as a trigger signal. When the trigger signal stays low, the carry-out remains the same, however, when the trigger is activated, the logic of carry-out changes. The intruded gate, due to its inherent C_{gate}, affects the power usage and delay constraints, and subsequently identified when verified for its integrity in the nuXmv model checker. Similarly we defined the

state-space for a 4 bit ripple carry adder(RCA). To overcome complexity, we designed other gates like XOR, using universal gates in separate modules. The instances of these modules are called in the main module while defining the formal model as per the netlist information. After validating the parametric properties of the trusted RCA circuit model, the intrusion is modeled at the third stage of carry-out by inserting a three stage Ring Oscillator(RO), which is enabled using an external trigger signal through a NAND gate. Once enabled, the RO continuously oscillates between the two voltage levels, resulting in more power consumption. It is pertinent to mention that the RO is not inserted along the path of carry. The model verification against the properties indicated that it fails both checks for power and delay bounds. The capacitances of added gates contributes towards the switching power consumption. In case of delay, although an intrusion is not along the path of delay computation, the added gate act as load to the previous gate generating the carry. Thus, a counterexample is also generated in this case by the model checker.

Table 2. Timing and memory resources for some of the Intruded circuits detected by our technique.

Circuit	No of Gates			Hardware Trojans		Dynamic Power			Delay		
	NAND	NOR	NOT	Intrusion	Effect	Check	Memory (MB)	Time (s)	Check	Memory (MB)	Time (s)
C-17	6	—	—	—	—	✓	69	1054	✓	75	1530
C-17 [30]	7	—	—	1 NAND	Power Consumption	✗	47	215	✓	77	550
C-17 [22]	9	1	2	1 XOR 1 NOR	Rare Input Modifications	✗	55	316	✗	54	715
Full Adder	8	1	7	—	—	✓	79	1210	✓	81	1770
HT Full Adder	11	1	9	1 XOR	Externally Triggered Carry Out	✗	61	375	✗	63	886
4 Bit RCA	32	4	28	—	—	✓	103	2715	✓	115	3855
HT 4 Bit RCA	33	4	31	RO	Power Consumption	✗	77	977	✗	79	1224

The result indicates that multi-parameter based intrusion detection is more effective than just using a single parameter. If the malicious circuitry is not being detected by the defined bounds for delay then the power property may fail, indicating presence of an intrusion whenever switching activity occurs. The results further elaborate the usefulness of the nuXmv model checker for handing real numbers and bounded model checking (BMC) feature. The extensive simulations for every possible input using traditional methods is a laborious task. In contrast, once a model is defined in a model checker based on the proposed approach, then it takes significantly lesser time and resources to test the integrity of the entire system. The results in Table 2 show that modeling and verification of all the case studies require a 115MB of memory at maximum, which is around

2% of the available memory of the machine used to acquire these experimental results. Therefore, the proposed approach seems quite scalable to handle larger circuits with the availability of around 16GB memory. However, the inherent state-space explosion problem of model checking may limit the applicability of the proposed approach for larger circuits and therefore, in these cases, we plan to extend the models to a higher abstraction level.

7 Comparison with Existing Gate Modeling Techniques

The gate level time and power models have been presented previously by different researchers. The timing analysis at the gate level description of a circuit is proposed in [4], where each gate is abstracted as a set of states and individual transitions are characterized by the propagation delay of the falling or rising edge. The propagation delays are determined by using set of differential equations for capacitances or through SPICE simulations. Timing verification for asynchronous circuits, proposed in [7], is based on translating the circuit behavior in terms of transition graphs, which is checked under the assumption that the delays are bounded between two numbers. The formal verification of timed circuits is represented as symbols with unspecified delays in [11]. In this scheme, a set of linear constraints on the symbols are discovered that guarantees the correctness of the circuit. Similarly, a state transition graph based power dissipation model is presented in [16] that considers charging and discharging of the capacitance at the gate output node. The input signal probabilities are used to estimate the expected activity number of each edge in the graph, followed by computing the total consumption by summing each edge. The consumption values of transitions are obtained from SPICE simulation.

The proposed gate modeling technique in this paper takes both power and timing parameters under consideration. Our scheme has an inherent advantage in the perspective of hardware intrusion detection that we take particular VLSI technology and possible variations in fanout and gate sizes in to account as well. Moreover, our model is generic as we estimate power and timing measurements with all possible input transitions without any requirements of simulations with SPICE or other circuit simulators.

8 Conclusions

This paper presents a generic framework based on the formal verification of the integrated circuit (IC) to detect malicious hardware intrusions. Unlike the traditional methods to detect the intrusions in integrated circuits, our solution uses formal models based on multi-parameter side channel information and their validation using a model checker. The ability of nuXmv model checker to handle *real* numbers and the powerful verification methods, based on SAT and SMT solvers, has been successfully utilized to validate dynamic power and path delay parameters, to ascertain integrity of integrated circuits. In the future, we plan to enhance this work by proposing an automated method for netlist translation, and

extending the models to higher abstraction level to efficiently handle scalability for larger ICs. Moreover, additional side channel parameters, such as leakage power will also be incorporated in order to strengthen the proposed Hardware Trojan detection scheme.

References

1. Adee, S.: The Hunt for the Kill Switch. IEEE Spectrum 45(5), 34–39 (2008)
2. Agrawal, D., Baktir, S., Karakoyunlu, D., Rohatgi: Trojan Detection using IC Fingerprinting. In: Symposium on Security and Privacy. pp. 296–310. IEEE (2007)
3. Anderson, M.S., North, C., Yiu, K.K.: Towards Countering the Rise of the Silicon trojan. In: Annual Report. Defence Science and Technology Organisation, DSTO-TR-2220, Australia (2008)
4. Bara, A., Bazargan-Sabet, P., Chevallier, R., Encrenaz: Formal Verification of Timed VHDL Programs. In: Specification & Design Languages. pp. 1–6. IET (2010)
5. Bhasin, S., Regazzoni, F.: A Survey on Hardware Trojan Detection Techniques. In: Circuits and Systems. pp. 2021–2024. IEEE (2015)
6. Bhunia, S., Hsiao, M.S., Banga, M., Narasimhan, S.: Hardware Trojan Attacks: Threat Analysis and Countermeasures. Proceedings of the IEEE 102(8), 1229–1247 (2014)
7. Bozga, M., Jianmin, H., Maler: Verification of Asynchronous Circuits using Timed Automata. Electronic Notes in Theoretical Computer Science 65(6), 47–59 (2002)
8. Cavada, R., Cimatti, A., Dorigatti, M., Griggio, A., Mariotti, A., Micheli, A., Mover, S., Roveri, M., Tonetta, S.: The nuXmv Symbolic Model Checker. In: Computer Aided Verification, LNCS, vol. 8559, pp. 334–342. Springer (2014)
9. Chakraborty, R.S., Wolff, F., Paul, S., Papachristou, C., Bhunia, S.: MERO: A Statistical Approach for Hardware Trojan Detection. In: Cryptographic Hardware and Embedded Systems, pp. 396–410. Springer (2009)
10. Cimatti, A., Clarke, E., Giunchiglia, E., Giunchiglia, F., Pistore, M., Roveri, M., Sebastiani: NuSMV 2: An Opensource Tool for Symbolic Model Checking. In: International Conference on Computer Aided Verification. pp. 359–364. Springer (2002)
11. Clarisó, R., Cortadella, J.: Verification of Timed Circuits with Symbolic Delays. In: Asia and South Pacific Design Automation Conference. pp. 628–633. IEEE (2004)
12. Di Natale, G., Dupuis: Is Side-Channel Analysis Really Reliable for Detecting Hardware Trojans? In: Design of Circuits and Integrated Systems. pp. 238–242 (2012)
13. Drechsler, R., et al.: Advanced Formal Verification. Springer (2004)
14. Force, T.: High Performance Microchip Supply. In: Annual Report. Defense Technical Information Center (DTIC), USA (2005), `http://www.acq.osd.mil/dsb/reports/ADA435563.pdf`
15. Jin, Y., Makris, Y.: Hardware Trojan Detection using Path Delay Fingerprint. In: Hardware-Oriented Security and Trust, 2008. pp. 51–57. IEEE (2008)
16. Lin, J.Y., Liu, T.C., Shen, W.Z.: A Cell-Based Power Estimation in CMOS Combinational Circuits. In: Computer-Aided Design. pp. 304–309. IEEE (1994)
17. Lodhi, F.K., Abbasi, I., Khalid, F., Hasan, O., Awwad, F., Hasan, S.R.: A Self-learning Framework to Detect the Intruded Integrated Circuits. In: International Symposium on Circuits and Systems. pp. 1702–1705 (2016)

18. Lodhi, F.K., Hasan, S.R., Hasan, O., Awwad, F.: Hardware Trojan Detection in Soft Error Tolerant Macro Synchronous Micro Asynchronous (MSMA) Pipeline. In: Midwest Symposium on Circuits and Systems. pp. 659–662 (2014)
19. Lodhi, F., Hasan, S., Hasan, O., Awwad, F.: Formal Analysis of Macro Synchronous Micro Asychronous Pipeline for Hardware Trojan Detection. In: Nordic Circuits and Systems Conference & International Symposium on System-on-Chip. pp. 1–4. IEEE (2015)
20. MathSAT 5: (2016), `http://mathsat.fbk.eu/`
21. Mitra, S., Wong, H.S.P., Wong, S.: The Trojan-Proof Chip. IEEE Spectrum 52(2), 46–51 (2015)
22. Mukhopadhyay, D., Chakraborty, R.S.: Hardware Security: Design, Threats, and Safeguards. CRC (2014)
23. Ngo, X.T., Danger, J.L., Guilley, S., Najm, Z., Emery, O.: Hardware Property Checker for Run-time Hardware Trojan Detection. In: Circuit Theory and Design (ECCTD), 2015 European Conference on. pp. 1–4. IEEE (2015)
24. Qu, G., Yuan, L.: Design THINGS for the Internet of Things-An EDA Perspective. In: International Conference on Computer-Aided Design (ICCAD). pp. 411–416. IEEE (2014)
25. Rabaey, J.M., Chandrakasan, A.P., Nikolic, B.: Digital Integrated Circuits, vol. 2. Prentice Hall (2002)
26. Rai, D., Lach, J.: Performance of Delay-Based Trojan Detection under Parameter Variations. In: Hardware-Oriented Security and Trust. pp. 58–65. IEEE (2009)
27. Rathmair, M., Schupfer, F.: Hardware Trojan Detection by Specifying Malicious Circuit Properties. In: Electronics Information and Emergency Communication. pp. 317–320. IEEE (2013)
28. Tehranipoor, M., Koushanfar, F.: A Survey of Hardware Trojan Taxonomy and Detection. IEEE Design and Test of Computers 27(1), 10–25 (2010)
29. Wang, L., Xie, H., Luo, H.: Malicious Circuitry Detection using Transient Power Analysis for IC Security. In: Quality, Reliability, Risk, Maintenance, and Safety Engineering. pp. 1164–1167. IEEE (2013)
30. Wei, S., Meguerdichian, S., Potkonjak, M.: Malicious Circuitry Detection using Thermal Conditioning. IEEE Transactions on Information Forensics and Security 6(3), 1136–1145 (2011)
31. Weste, N., Harris, D.: CMOS VLSI Design: A Circuits and Systems Perspective. Pearson (2011)
32. Xiao, K., Forte, D., Jin, Y., Karri, R., Bhunia, S., Tehranipoor, M.: Hardware Trojans: Lessons Learned After One Decade of Research. ACM Transactions on Design Automation of Electronic Systems Vol 22(1) (2016)
33. Zhang, X., Tehranipoor, M.: Detecting Hardware Trojans in Third-Party Digital IP Cores. In: Hardware-Oriented Security and Trust (HOST). pp. 67–70. IEEE (2011)