New Design Strategy for Improving Hardware Trojan Detection and Reducing Trojan Activation Time

Hassan Salmani and Mohammad Tehranipoor
ECE Department
University of Connecticut
{salmani_h,tehrani}@engr.uconn.edu

Jim Plusquellic
ECE Department
University of New Mexico
jimp@ece.unm.edu

ABSTRACT

Hardware Trojans in integrated circuits and systems have become serious concern to fabless semiconductor industry and government agencies in recent years. Most of the previously proposed Trojan detection methods rely on Trojan activation to either observe a faulty output or measure side-channel signals such as transient current or charge. From the authentication stand point, time to trigger a hardware Trojan circuit is a a major concern. This paper analyzes time to (i) generate a transition in functional Trojans and (ii) fully activate them. An efficient dummy flip-flop insertion procedure is proposed to increase Trojan activity. Depending on authentication time and circuit topology, a transition probability threshold is selected so that inserted dummy flip-flops would moderately impact area overhead. The simulation results on s38417 benchmark circuit demonstrate that, with a negligible area overhead, our proposed method can significantly increase Trojan activity and reduce Trojan activation time.

1 Introduction

Outsourcing design and fabrication process has become a trend in Integrated Circuit (IC) market due to economical profit. Such trend, however, provides an opportunity for adversary to tamper IC supply chain by maliciously implanting extra logic as Hardware Trojan circuitry into an IC [1]. This raises serious concerns about security and truthworthiness of imported products employed in mission critical applications. An attacker can change a design netlist or subvert the fabrication process by manipulating design mask, without affecting main functionality of the design. [2]

Hardware Trojan detection is an extremely challenging problem and traditional structural and functional tests do not seem to be effective in targeting and detecting Trojans. Automatic Test Pattern Generation (ATPG) methods used in manufacturing test for detecting defects do so by operating on the netlist of the Trojan-free circuit. Therefore, existing ATPG algorithms cannot target Trojan activation/detection directly [3]. Trojan circuits have stealthy nature and are triggered in rare conditions. Trojans are designed such that they are silent most of their life time and have very small size relative to their host design, with featuring limited contribution into design characteristics. These suggest that they most likely connect to nets with low controllability and/or observability [3][4]. It is expected that Trojan inputs are supplied by nets with low transition probabilities to lessen its impact on side-channel signals such as power and delay.

Trojan detection makes efficient pattern generation necessary to disclose Trojan impact on design characteristics beyond process and environmental variations. Trojan detection methods using transient power analysis [5][6] require patterns that increase Trojan activity whereas keep circuit activity low to magnify Trojan contribution into the circuit power profile. Methods that are based on delay analysis [7][8] require patterns that generate transition on nets that supply Trojan inputs to reveal wiring and input gate resistance and capacitance impact of Trojan on the design delay characteristic. From authentication stand point, it is critical to (i) analyze time to generate a transition at Trojan input and in Trojan circuit and (ii) reduce authentication time.

In this paper, we develop a methodology to increase the probability of generating a transition in functional Trojan circuits and analyze the transition generation time. Transition probability is modeled using Geometric Distribution [9] and is estimated based on number of clock cycles needed to generate a transition on a net. To increase transition probability of nets whose transition probability is lower than a specific probability threshold, an efficient dummy flip-flop insertion procedure is proposed. The procedure identifies nets with low transition probability and insert dummy flip-flops such that it increases the transition probability. It should be noted that dummy flip-flops are inserted in a way that will not change the functionality of design. The effectiveness of dummy flip-flop insertion is examined by evaluating different transition probability thresholds for various Trojan circuits. The relation between authentication time, the number of required transitions in Trojan circuit, and tester clock is studied. These parameters would help determine the transition probability threshold of a design. The transition probability threshold, in turn, provides an estimation of area overhead induced by inserted dummy flip-flops.

The paper is organized as following: Section 2 describes prior work on Trojan detection. Analyzing Trojan activation time is presented is Section 3. The proposed dummy flip-flop insertion procedure is presented in Section 4. Transition probability threshold analysis and simulation results are presented in Sections 5 and 6. Finally the concluding remarks are presented in Section 7.

2 Prior Work

Authors in [5] present a method to generate a power fingerprint of genuine ICs considering various types of noise in the circuit. Random patterns are applied to IC-Under-Authentication (IUA) to generate a measurable difference between the power profile of the genuine IC and IUA. In [8], using the same basic procedure as in [5], path delay fingerprint of a design is generated. From an IC design, many chips are selected and high coverage input patterns are run on these sampled chips and a series of delay fingerprints are generated. To detect Trojans, the same input patterns are applied to IUA and compared with the delay fingerprints.

The proposed method in [6] is based on analyzing local I_{DDT} current measured from power ports on the target chip. To alleviate process variations impact during measurement, a calibration process is performed for each IUA before actual measurement. Trojan-inserted designs are distinguished using outlier analysis. In [10], a multiple supply transient current integration method is presented to detect hardware Trojans in IUA. The current is measured locally from various power pads or controlled collapse chip connections (C4s) on the die. Random patterns are applied to increase the switching in the circuit in a test-per-clock fashion [11].

In [12] [13], two methods are presented to detect and localize Trojan circuits. The methods are based on a test pattern generation technique to generate transitions in a target region while keeping other regions at minimum activity. In the first method [12], a region is defined as a group of a number of flip-flops. To induce activity in a region and keep other regions in low switching (i.e. quiet), random patterns are generated and applied. Those patterns that meet a certain switching threshold are selected and used for Trojan detection. In the second method [13], a region is formed from a group of flip-flops and gates which are logically related to a particular function. Random patterns that limit activity in a target region are selected. The presented results compared the effectiveness of selected test patterns with the random patterns. In [4], the authors present a sustained vector technique. A vector is applied to the circuit and for several clock cycles (up to 25) primary inputs are kept unchanged. In this way all transitions in the circuit would be induced because of state bits and it is expected after some clock cycles activities converge to a specific portion of the circuit. By applying the next vector another portion of circuit will be targeted.

A randomization based probabilistic approach to detect Trojan is presented in [14]. The authors show that it is possible to construct a unique probabilistic signature of a circuit using specific probability on its inputs. Input patterns are applied based on a specific probability to IUA and outputs are compared to design circuit. In case of a difference, Trojan infection is reported. Otherwise, after applying a set of N vectors if the IUA and circuit give the same output, using statistical reasoning, it is reported that the IUA is Trojan-free with a confidence interval.

A comprehensive taxonomy of Trojan circuits is presented in [3]. Trojans are classified based on *physical*, *activation*, and *action* characteristics. The *physical* characteristic studies *type*, *size*, *distribution*, and *structure* of a Trojan. In terms of *type*, Trojan can be *functional* or *parametric*. *Functional* Trojans are realized through adding or deleting of transistors or gates, while *parametric* ones are realized through modification to physical geometry designed to sabotage reliability. The number of gates or transistors which are added or deleted defines Trojan *size*. *Distribution* refers to the locations of Trojan components in physical layout of the chip. They can be tight (i.e. placed close to each other) or loose (i.e. dispersed across the layout.) Trojan insertion can affect chip dimension, delay characteristic and power profile of a circuit. Trojan *activation* characteristics refer to the criteria that causes the Trojan to become active and carry out its disruptive function. The types of disruptive behavior introduced by Trojan determines Trojan *action* characteristics. For more details on Trojan taxonomy, reader is referred to [3]. In this work, we focus on functional Trojans and targeting parametric Trojans will be part of our future work.

3 Trojan Activation Time Analysis

Since there is no information about Trojan circuit in terms of size, type, or location, from authentication standpoint, it is crucial to analyze Trojan activation time (partially or fully). In this paper, *fully activation* of Trojans refers to patterns that activate a Trojan so that they impact the circuit output and cause malfunction. However, *partial activation* refers to generating one or more transitions inside the Trojan circuit so that it improves the effectiveness of transient power-based methods [5][6][10]. In general a functional Trojan consists of two parts: Trigger and Payload [15]. The Trigger circuit is mostly inactive by nature with no Payload effect. Under certain rare conditions or events, the Trojan is activated (triggered) and then Payload injects an error to the circuit. Generating transition in Trojan circuit depends on its implementation. Switching at the first level gates of Trojan circuit depends on its preceding cells. The next levels of Trojan circuit are similar to the first level; therefore, in the following we focus on generating switching in one Trojan cell at the first level of a Trojan circuit to carry out our detailed analysis. However, the results in Section 6 will be presented for the entire Trojan circuit.

In general, the transitions in a circuit are induced by transitions in scan cells and primary inputs [16]. We define a Trojan cone as logic circuit connecting to the inputs of a Trojan gate. Note that in this section, we present one Trojan gate for our analysis, however, a Trojan may contain more than one gate. Also, note that we do not assume the location of Trojan is known to us. The procedure developed in this work is independent of location and size of hardware Trojan in integrated circuits. Trojan cone can determine the required time to generate transition in the Trojan cell. The number of gates, gate types and the structure of Trojan cone can define time to generate transition in a Trojan cell as well. Figure 1 shows two example Trojan cones. Trojans are named as *Trojan 1* and *Trojan 2*. *Trojan 1* contains three gates and two levels while *Trojan 2* contains seven gates and three levels. T_{g1} in $Trojan1$ is connected to the cone shown in Figure 1(a) and T_{g3} is connected to the cone in Figure 1(b). Other gates in the two Trojans are assumed to be connected to other parts of the circuit.

In Figure 1(a), Trojan cone consists of 17 gates in 11

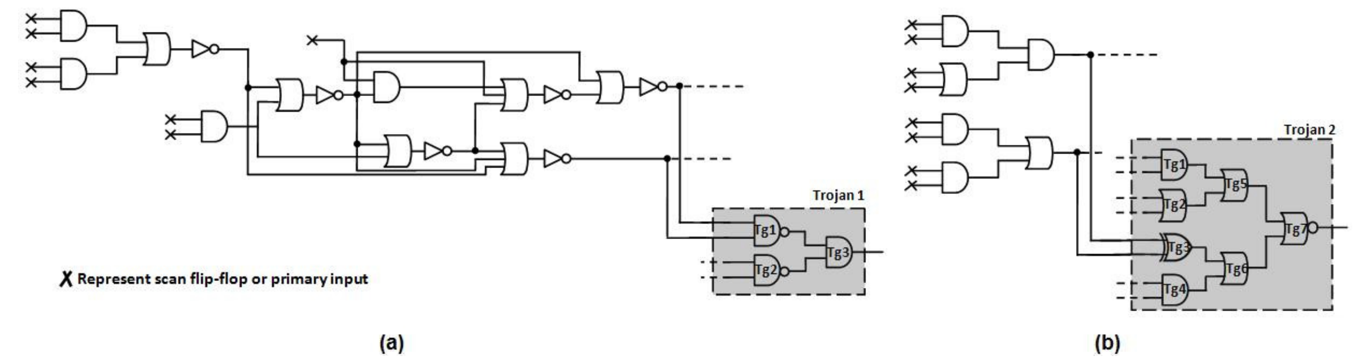


Figure 1: Two Trojan cone examples: (a) Trojans 1 and (b) Trojan 2

levels. Trojan cone contains all gates in original circuit impacting the Trojan cell and the Trojan cell itself (here T_{g1}). Simulation results show that after applying 1000 random test vectors in test-per-clock fashion, there are 67 transitions at T_{g1} output.

In Figure 1(b), Trojan cone consists of 7 gates in 2 levels. The simulation results show that there are 421 transitions at T_{g3} output after applying the same number of test vectors, i.e. 1000. Since random vectors are applied to the above circuits, the results can be slightly different from one random vector set to another. As seen from the results, the number of transitions in the two Trojan cells vary significantly. This is mainly due to the difference in Trojan cone structure, number of levels, and number of inputs (scan flip-flops and primary inputs) and the Trojan gate type.

Probability can represent characteristics of a circuit since it considers gates functionality and interconnections among the cells. The probability of switching at a node in the circuit provides a good estimation of the time to generate switching on that node. Trojan cone determines switching probability at the Trojan cell output, e.g. T_{g1}. Suppose the probabilities of having '1' and '0' at Trojan output are $Pt1$ and $Pt0$, respectively, the probability of switching from 0 to 1 or 1 to 0 at the output of a Trojan gate will be $Pt_{gi} = Pt1 \times Pt0$, where gi is the ith gate at the first level of a Trojan. For example, with assumption of applying random patterns through inputs, with probability of 1/2, the probability of generating a transition at the output of Trojan gate T_{g3} ($Pt_{T_{g3}}$) is 0.25 as shown in Figure 2. The circuit shown in this figure is the same as one depicted in Figure 1(b).

To obtain transition probability, a transition (i.e. success) can be modeled using Geometric Distribution (GD) [9]. The Geometric Distribution is a discrete distribution for $n = 0, 1, 2, \cdots$ with the probability function $p(n) = P \times (1-P)^n$.

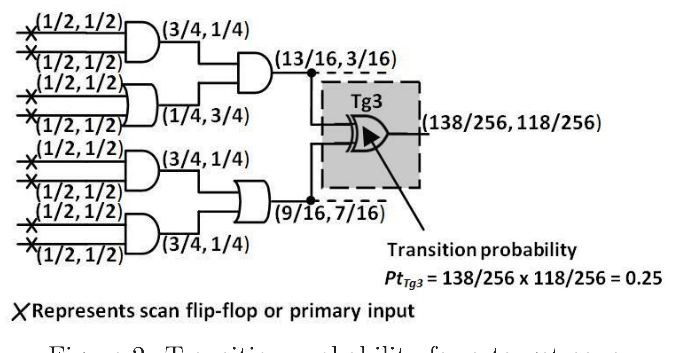
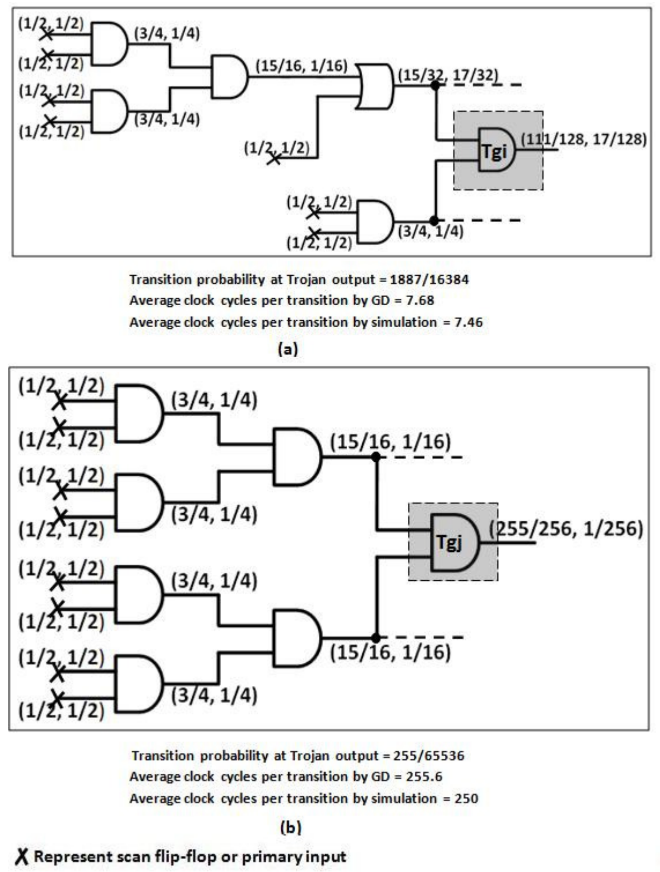


Figure 2: Transition probability for a target cone.


(1/2, 1/2)
(3/4, 1/4)
(15/16, 1/16)
(15/32, 17/32)
Tgi
(111/128, 17/128)
Transition probability at Trojan output = 1887/16384
Average clock cycles per transition by GD = 7.68
Average clock cycles per transition by simulation = 7.46
(a)
(1/2, 1/2)
(3/4, 1/4)
(15/16, 1/16)
Tgj
(255/256, 1/256)
Transition probability at Trojan output = 255/65536
Average clock cycles per transition by GD = 255.6
Average clock cycles per transition by simulation = 250
(b)
X Represent scan flip-flop or primary input
Figure 3: Comparing mathematical and simulation results.

The probability function states that after n clock cycles, finally in the $(n+1)$th clock cycle, there will be a transition, i.e. $(n+1)$th trial is the first success. The average number of experiments is $(P^{-1} - 1)$ which indicates the number of required clock cycles, *on average*, to generate a transition.

For the Trojan cell shown in Figure 2, the calculation based on Geometric Distribution shows that on average three clock cycles are required to generate a transition at the Trojan cell (T_{g3}) output. This is demonstrated by our simulation results since, on average, in each 2.37 clock cycles a transition was generated after applying 1000 test vectors. Note that the 1000 random test vectors are generated with the probability of 1/2 for '0' and '1'.

Figure 3 presents two new Trojan cones and compares the average clock cycles per transition using GD (i.e. probability analysis) and simulation. Figure 3(a) shows that the simulation result of applying 1000 random patterns is very

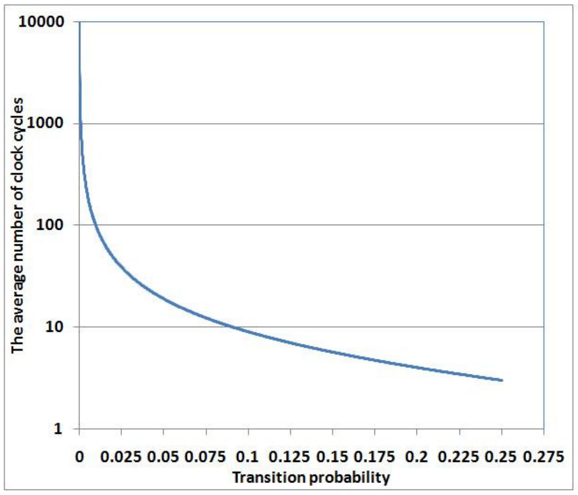


Figure 4: The number of required clock cycles versus transition probability based on geometric distribution.

close to that of GD. Trojan cone in Figure 3(b) consists only of NAND gates such that the probability of generating '1' at Trojan cell T_{gj} output is much less than that of '0' therefore, there is a small transition probability for T_{gj}. Any transition to '1' will most likely follow immediately by a transition to '0' since Trojan cone mostly provides '0' at the output of T_{gj}. The simulation results by applying 1000 test vectors show that each 250 clock cycles there is one transition at Trojan output and probability analysis show that every 255.6 clock cycles, one transition can be generated at the output of T_{gj} gate.

It is seen from both analyses (GD and simulation) that as $P0$ or $P1$ of a net becomes too large or small, the transition probability reduces significantly. Therefore, to maximize transition probability in a net, it would be preferred to ensure that $P0$ and $P1$ values are close. The maximum transition probability on net can be 0.25 and it happens when $P0 = P1 = 1/2$. Given a cone structure and various gate types used in the cone, equalizing the transition probabilities would seem impractical but by improving controllability by inserting dummy flip-flops, we would be able to increase transition probability for both $0 \rightarrow 1$ and $1 \rightarrow 0$ transitions. This is validated by the analysis preformed using Geometric Distribution as shown in Figure 4. As seen, as the transition probability decreases, the number of clock cycles to generate transition increases exponentially.

4 Dummy Flip-Flop Insertion

When the probabilities for '0' and '1' of nets on a path in a cone becomes unidirectional, i.e. $P1 \gg P0$ or $P0 \gg P1$ similar to the example shown in Figure 3(b), transition probability of the nets ($P_i0 \times P_i1$) rapidly decreases. To ensure that $P0$ and $P1$ are greater than a specific threshold, dummy flip-flops can be inserted to keep probabilities of '1' and '0' for the nets closer to each other. Note that in this paper both terms "dummy flip-flop" and "dummy scan flip-flop" refer to the increased controllability (transition probability) in a circuit.

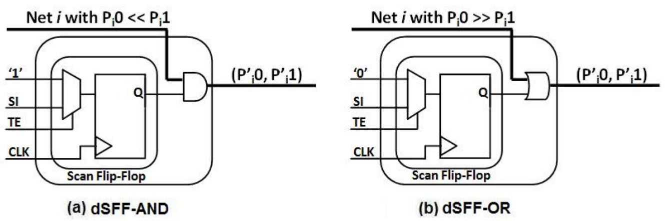


Figure 5: The dummy flip-flop structures when (a) $P_i0 \ll P_i1$ and (b) $P_i0 \gg P_i1$.

Figure 5 shows the structure of dummy scan flip-flop (dSFF) in addition to an extra gate (AND or OR). If probability of inducing '0' on target net Net_i, P_i0, is less than the probability of '1', P_i1, as in Figure 5(a), an AND gate is being used to increase P_i0. However, if P_i1 is less than P_i0, as in Figure 5(b), an OR gate is being used to increase P_i1. In this work, *dSFF-AND* and *dSFF-OR* represent dummy scan flip-flops with AND and OR gates, respectively. Adding a dSFF to a net with low transition probability would increase the net's and following nets' transition probability. When Test Enable (TE) is active, the output of scan flip-flop is supplied by Scan Input (SI). The inserted dummy scan flip-flop has no impact on the functionality of the circuit. In normal functional mode, the output of scan flip-flop is supplied by either '0' or '1' depending on the gate type at the output of scan flip-flop to avoid changing the functionality of Net_i.

The probabilities of '0' and '1' at the output of scan flip-flop are 1/2. Thus, by supplying internal nets with such high probability, the '0' and '1' probabilities on target nets can become closer and their respective transition probabilities can be increased. Assume that P_i0 of Net_i is much greater than its P_i1, where

$$P_i0 = \frac{K}{N} \qquad \text{and} \qquad P_i1 = 1 - \frac{K}{N}$$

where K and N are cardinal values. The denominators of probabilities would be the number of clock cycles in an experience and their numerators are the number of desired value.

By inserting proposed dummy flip-flop as in Figure 5(b), new probabilities are

$$P_i'0 = \frac{K}{2N}, \text{ and } P_i'1 = 1 - \frac{K}{2N}$$

As a result, $P_i'0$ will be smaller than P_i0 and $P_i'1$ will be greater than P_i1. Thus, after dummy flip-flop insertion, the transition probability of the target net and its following nets would be greater as

$$P_i'0 \times P_i'1 > P_i0 \times P_i1$$

$$\frac{K}{2N} \times (1 - \frac{K}{2N}) > \frac{K}{N} \times (1 - \frac{K}{N})$$

$$2N - K > 4K - 4N \Rightarrow K/N < 6/5$$

which is true because $P_i0 = K/N$ and is never greater than 1.

Using same analysis, it can be demonstrated that by inserting AND gate when P_i0 of a net is much lower than its P_i1, the transition probability of the net can be increased. Figure 6 shows a modified version of circuit shown in Figure 3(b) by inserting a dSFF-OR.

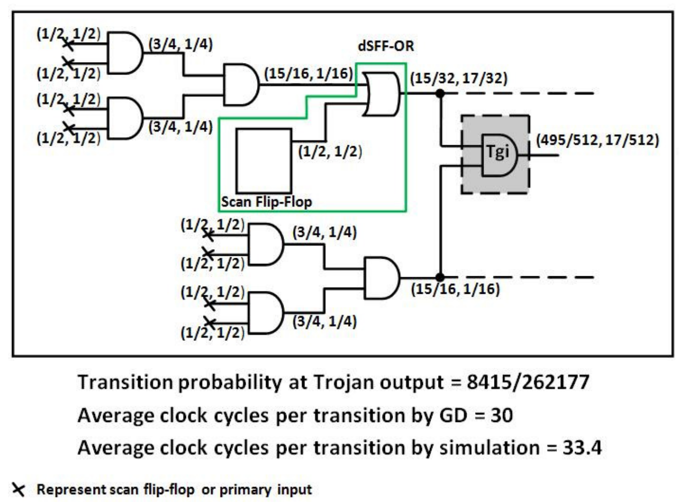


Figure 6: Increasing transition probability by inserting dSFF-OR.

4.1 Removing Rare Triggering Conditions

An able adversary would ensure that Trojans are activated only under very rare conditions. It could be a rare circuit state, certain temperature or noise, etc. This is necessary to avoid Trojan detection accidentally using structural or functional patterns. As an example, for functional Trojans [3], a Trojan can have $q \gg 1$ trigger inputs which can be nets with (i) very low transition probabilities and (ii) rare combinations. When the transition probability of Net_i is low, either P_i0 is much greater than P_i1 or vice versa, as discussed in Section 3. With q number of trigger inputs, the probability of generating a specific trigger vector is

$$P_{trigger-vector} = \prod_{i=1}^{q} \{P_i0|P_i1\} \qquad (1)$$

It is expected that $P_{trigger-vector}$ to be very low if P_i0 or P_i1 is low. By inserting dummy scan flip-flop, the transition probability of nets would increase since P_i0 and P_i1 values becomes closer. As a result, $P_{trigger-vector}$ also increases and the trigger vector will not be a rare event anymore. By increasing the transition probability of nets with low transition rate, we will eliminate hard-to-activate sites in a design. This would result in increasing the probability of switching in Trojan circuit. If fully activated, Trojan's output can impact design functionality and it will be detected. In case of increasing switching in the Trojan, called *partial activation* in this paper, the Trojan can be detected much easier using transient power or charge-based analysis methods [5][6][10]. This method eliminates the need to focus on rare conditions as proposed in [15].

For example, Table 1 shows probability of two nets in s38417 benchmark before and after scan dummy flip-flop insertion. Assuming that Trojan needs trigger vector $\{01\}$ on $Net1$ and $Net2$, as seen in the table, the probability of the trigger vector would be $P_{trigger-vector} = P_{Net1}0 \times P_{Net2}1$=4.079e-06 in the original circuit without dummy flip-flop. However, the probability increases to 0.094 after dummy flip-flop insertion.

4.2 Dummy Scan Flip-Flop Insertion Procedure

Figure 7 shows the proposed dSFF insertion procedure. To increase transition probability of nets in the circuit, we set a threshold as P_{TH} to select nets that are slightly above this threshold to increase the transition probability of nets that are below P_{TH}. After setting P_{TH} and original design as $CurrentDesign$ (Lines 1-2), the procedure will calculate transition probability of all nets in the design (Line 3). Nets are then divided into two groups: 1) nets with transition probability higher than P_{TH}, and 2) nets with transition probability less than P_{TH}. Nets in the first group are stored permanently in $HighTransition$ array (Line 4). Nets in $HighTransition$ array are sorted based on their transition probabilities in an increasing order (Line 5). The number of nets in the second group is stored in $\#MinLowTranNet$ variable. The procedure, in Line 7, selects an $Unchecked$ net with the lowest transition probability from $HighTransition$ array. It modifies the net by inserting dSFF (dSFF-AND or dSFF-OR depending on target net's P_0 and P_1 values) and set the design as $UpdatedDesign$. The selected net is removed from $HighTransition$ array. Transition probabilities of nets in $UpdatedDesign$ are then calculated in Line 10. If the number of low transition nets decreases, the inserted dummy flip-flop is considered to be effective and kept in the database and $UpdatedDesign$ is set as $CurrentDesign$. If there is no reduction in the number of low transition nets, $CurrentDesign$ is used again and the next net from $HighTransition$ is selected. The procedure terminates in two cases (Lines 12-13): 1) the number of nets with transition probability less than P_{TH} is zero, or 2) there is no any unchecked net left with transition probability higher than P_{TH} ($HighTransition = \emptyset$).

We acknowledge that inserting dummy scan flip-flop increases the delay of paths and can impact design performance. Note that it is unlikely that adversary uses nets on critical paths as input since it can impact the path delay and can be easily detected using path delay fault test patterns. Using the above procedure, it is possible to avoid inserting dummy flip-flops on critical paths by eliminating nets on the critical paths from $HighTransition$.

5 Transition Probability Threshold Analysis

Inserting dummy flip-flops to increase transition probability of nets would increase circuit area. The area overhead mainly depends on transition probability threshold (P_{TH}). By setting a P_{TH}, our proposed procedure ensures that all nets in the circuit have transition probability greater or equal to this threshold. P_{TH} would impact both area overhead (i.e. the number of dSFFs) and Transition generation time in hardware Trojan cells. In general, setting smaller P_{TH} would result in smaller number of dSFFs but would require more time, on average, to generate switching in Trojan cells. However, setting larger P_{TH} would require more number of dSFFs but reduces the transition generation time in hardware Trojan cells.

From above discussions, it is clearly seen that there are several parameters that should be taken into consideration

Table 1: Probability of two nets in s38417 benchmark before and after dSFF insertion.

	Before dSFF insertion			After dSFF insertion		
	$P0$	$P1$	$P_{Net1}0 \times P_{Net2}1$	$P0$	$P1$	$P_{Net1}0 \times P_{Net2}1$
$Net\ 1$	0.999995317077	4.6e-06	4.079e-06	0.989	0.011	0.094
$Net\ 2$	0.999959170737	4.08e-06		0.905	0.095	

```
01: Set transition probability threshold ($P_{TH}$).
02: Set original circuit as $CurrentDesign$.
03: Calculate transition probability of nets in $CurrentDesign$.
04: Identify nets with transition probability greater than $P_{TH}$, store them in $HighTransition$ array and mark them as $Unchecked$.
05: Sort nets in $HighTransition$ array nets based on their transition probability in an increasing order.
06: Store the # of nets with transition probability less than $P_{TH}$ as $\#MinLowTranNet$.
07: Select an $Unchecked$ net with the lowest transition probability in $HighTransition$ array called $TargetNet$.
08: Insert dSFF.
09: Update $CurrentDesign$ (called $UpdatedDesign$.)
10: Calculate transition probability of nets in $UpdatedDesign$.
11: Calculate the # of nets with transition probability less than $P_{TH}$ as $\#LowTranNet$.
12: If $\#LowTranNet$ is less than $\#MinLowTranNet$, $\#MinLowTranNet = \#LowTranNet$ and set $UpdatedDesign$ as $CurrentDesign$.
13: If $\#MinLowTranNet$ is zero, return $CurrentDesign$ and $\#MinLowTranNet$.
    Otherwise, if there is any net in $HighTransition$ array marked $Unchecked$, go to Step 6.
```

Figure 7: dSFF insertion procedure.

when setting P_{TH}. They can be grouped into two main categories namely authentication and circuit parameters. Authentication parameters are of authentication characteristics and consist of two sub-parameters: 1) authentication time of each integrated circuit, T_{Au}, and 2) the clock period of tester, T_{Tester}. Circuit parameters represent circuit characteristics and consist of two sub-parameters: 1) the number of required transitions in Trojan circuit, N_{Tr}, and 2) the average number of clock cycles per transition which can be modeled using Geometric Distribution. Note that N_{Tr} is an important parameter when using transient power analysis methods for detecting hardware Trojans since it indicates the contribution that Trojans power makes to the total circuit power. The larger the N_{Tr} the easier the detection of a Trojan would be.

Equation 2 shows how authentication and circuit parameters are related to each other.

$$T_{Au} = N_{Tr} \times (P_{TH}^{-1} - 1) \times T_{Tester} \tag{2}$$

T_{Au} is a user-defined parameter that depends on time-to-market and criticality of the application in which the circuit will be used. The equation is based on the time-to-generate a specific number of transitions in a Trojan cell. From Geometric Distribution analysis, on average, $(P_{TH}^{-1} - 1)$ clock cycles are required for each transition, and each clock cycle takes T_{Tester} time unit.

In the following, the impact of each parameter is studied in more details. Assume that the clock frequency of tester is 250MHz (T_{Tester}=4×10^{-3} second). Further, assume that user sets T_{Au}=120 seconds. If N_{Tr}=100, then using Equation 2, P_{TH} is 0.0033. Inversely, if the user sets P_{TH}=0.002, then the authentication time T_{Au}=199.6 seconds. This analysis shows that decreasing P_{TH} would increase T_{Au} since lower number of dummy scan flip-flops would be inserted. When the number of dummy flip-flops decreases, more time is needed to generate the same number of transitions on low transition nets. Similarly, by increasing P_{TH} to 0.004, the T_{Au} decreases to 99.6 seconds.

We believe that, for high-risk and secure applications, it would be possible to devote more time to each chip for authentication and that the area overhead may not be a big concern. Continuing with the above analysis, assuming that T_{Au}=3 minutes, P_{TH} would decrease to 0.0022. This means that with increasing authentication time (T_{Au}), the overhead can be reduced by reducing P_{TH}. Figure 8 shows that for a target authentication time, P_{TH} increases by the number of required transitions at Trojan output; therefore, area overhead increases. Further, P_{TH} decreases at any specific number of transitions by increasing authentication time. The minimum P_{TH} is obtained when the number of transitions is minimum and authentication time is maximum.

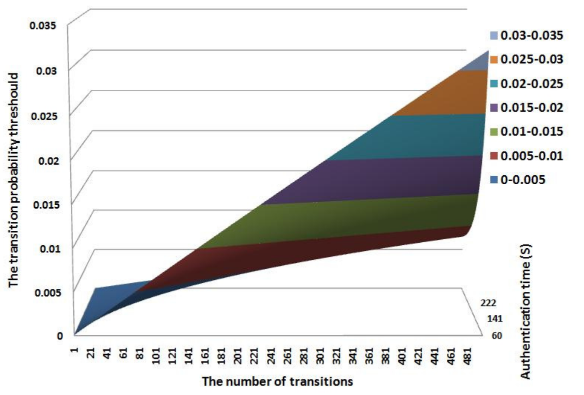


Figure 8: Probability threshold versus authentication time and the number of transitions.

6 Simulation Results

We apply our dummy flip-flop insertion procedure to s38417 benchmark which contains 1564 flip-flops and 4933

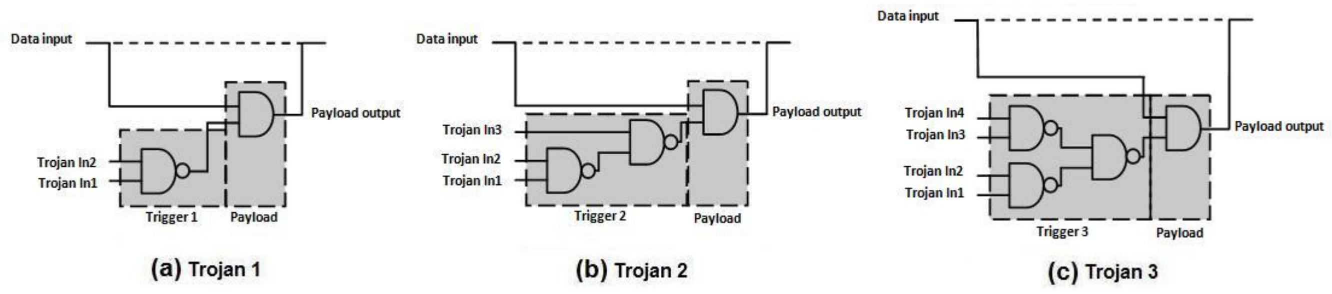


Figure 9: Trojan circuits.

Table 2: Trojans activity analysis before dSFF insertion.

	TP In1	TP In2	TP In3	TP In4	Average N_{cycle} per Trigger output change	N_{Tr}	Trigger transition count	Payload transition count	POC count
Trojan 1	4.6e-06	2.4e-05	-	-	100	3629	3593	26	0
Trojan 2	4.6-e06	2.4e-05	4.6e-06	-	91	7554	3935	26	0
Trojan 3	4.6e-06	2.4e-05	4.6e-06	4.5e-05	100	7186	3592	0	13

gates. Two different transition probability thresholds are examined in this work (P_{TH}=10e-05 and 10e-04). The amount of area overhead (number of dSFFs) to ensure all nets have transition probabilities lower than P_{TH} is evaluated. Further, three small Trojan circuits, presented in Figure 9, are inserted into the benchmark circuit. Nets with lowest transition probability are selected to be connected to the Trojans. Each Trojan circuit consists of two parts: Trigger and Payload. The Payload inputs come from Trigger output and data input which is part of the original circuit. Based on '0' and '1' probabilities at Trigger output and Payload data input, an AND gate is selected as Payload for each of the three Trojans. Dash lines above Trojans in the figure represent the connection in the original circuit which is assumed to be restitched through Trojan's Payload by adversary. The simulation results show the average number of clock cycles to generate a transition at Trigger output. The total number of transitions in Trojan circuit and the number of transitions on Trigger output that can potentially cause functional failure are reported. Additionally, the number of transitions on Payload output is also obtained and we will investigate the difference between payload output and its data input to further analyze the number of erroneous logic values injected into the circuit.

When the value of Trigger output is dormant (i.e. '1' for AND/NAND Payloads and '0' for OR/NOR Payloads), the Payload output is the same as Payload's data input; otherwise, the Payload output depends on values of both Trigger output and data input. If both are the same, then the output will be similar to the both inputs. However, a different Payload input combination assuming the Trigger is active would mean that the Payload output is due to Trigger input. This is called *Fully Activation* of Trojan since the Payload output change (POC) can cause functional failure.

The POC rate depends on transition rate of Trigger output and Payload data input. It is expected when both Payload inputs have low transition probability the POC rate to be unpredictable (small or large). For example, if Payload is an AND gate and data input and Trigger output have high '1' probability, low POC rate is expected. On the other hand, if one of the Payload inputs has higher transition probability than the other, larger POC rate is expected. If Trigger output is active for many clock cycles, a large Payload output change is expected.

The proposed method can help Trojan detection in two ways:

1. **Transient Power Analysis:** By increasing the number of transitions in Trojan circuits, the proposed method can help improve the previously proposed power-based methods [5][6][9][11]. In this case, the vectors are applied in a test-per-clock (TPC) fashion since no observation is made by the flip-flops. In fact, the power pads and C4s are the observation points since transient current is being measured. Suppose N_{sff} is the number of scan flip-flops and N_{vec_tpc} is the number of vectors, the total number of clock cycles $N_{total_{cycle}} = N_{vec} + N_{sff} - 1$. When $N_{vec} \gg N_{sff}$, the total number of clock cycles equals the number of test vectors $N_{total\_cycle\_tpc} = N_{vec}$.

2. **Fully Activation:** By increasing the probability of fully activation of a Trojan (making the data input to be different from Payload output) the probability of observing an incorrect response to the applied vectors would also increase. In this case, the test vectors are applied in a test-per-scan (TPS) fashion since the response of a test vector pair must be captured and scanned-out. The test vectors are applied similar to launch-off-shift (LOS) method used for delay testing with no requirement on at-speed scan enable signal. The second vector is only 1-bit shifted version of the first vector (i.e. initialization vector). If N_{sff} is the number of scan flip-flops and N_{vec_tps} is the number of vectors, the total number of clock cycles $N_{total\_cycle\_tps} = (N_{sff} + 1) \cdot N_{vec\_tps}$.

6.1 Without Dummy Flip-Flop

Simulations are run for $N_{vec\_tpc} = N_{vec\_tps} = 360000$ test vectors. Selecting a large numner of random test vectors provides a good average for the results we report. It also makes the results comparable to what is obtained using probability analysis such as Geometric Distribution. However, as the

Table 3: Trojans activity analysis after dSFF insertion with P_{TH} = 10e-5.

	TP In1	TP In2	TP In3	TP In4	Average N_{cycle} per Trigger output change	N_{Tr}	Trigger transition count	Payload transition count	POC count
Trojan 1	0.04	2.4e-05	-	-	17	20429	20389	40	2
Trojan 2	0.04	2.4e-05	0.04	-	17	40820	20389	42	1
Trojan 3	0.04	2.4e-05	0.04	0.24	17	40768	20374	4	20

Table 4: Trojans activity analysis after dSFF insertion with $P_{TH} = 10e - 4$.

	TP In1	TP In2	TP In3	TP In4	Average N_{cycle} per Trigger output change	N_{Tr}	Trigger transition count	Payload transition count	POC count
Trojan 1	0.1	0.009	-	-	3	105362	105330	32	4
Trojan 2	0.1	0.009	0.24	-	4	203097	97733	34	2
Trojan 3	0.1	0.009	0.24	0.08	3	213138	107797	10	15

results show, a much lower number of test vectors would be needed in practice. Table 2 shows the results for Trojans switching activity for the original circuit. Columns 2 to 5 show the transition probabilities for the Trojans inputs. Trojan 1 has two inputs, Trojan 2 has three inputs and Trojan 3 has four inputs. Column 6 shows the number of clock cycles, on average, needed to generate a transition at Trojan's Trigger output. The seventh column in the table indicates the total number of transitions in the Trojan circuits. The number of transitions on Trigger output is reported in the eighth column. The number of transitions at the Payload output and the number of difference between Payload data input and output (POC count) are presented in the Columns 9 and 10, respectively.

Simulation results show that for all Trojans, on average, almost 100 clock cycles are needed to generate a transition at the Trojans Trigger output. Trigger transition count is part of the total number of transitions in a Trojan circuit. As seen, Payload outputs of Trojan 1 and Trojan 2 experience more number of transitions than that for Trojan 3. Studying Payload data input and Trigger output in the circuit, when there is a transition at Payload output, shows that in all cases the logic values for both are same, thus no Payload output change occurs. However, because of Trigger output value of Trojan 3, 13 Payload output change (POC) has been observed which could cause circuit malfunction.

6.2 P_{TH} = 10e-05

After running our dSFF insertion procedure considering P_{TH} = 10e-05, it has been observed that there are four nets in s38417 benchmark with transition probability lower that P_{TH}. Using our procedure, 4 dSFFs are inserted to increase transition probabilities of these nets. The 4 dSFFs induce an area overhead about 0.2%. Table 3 shows the transition probability of Trojans inputs after dSFF insertion. Comparing with Table 2, the transition probability of inputs 1, 3, and 4 are increased to above P_{TH}.

The simulation results in Table 3 show that the number of clock cycles to generate a transition at Trojan output is reduced in all cases by about 6 times. The number of Transitions in Trojan circuits and Trigger outputs are increased by about 5 times in all cases. There is also increase in the number of transitions on Payload output and POC count. Same argument can be made about the Payload transition and Payload output change as in Table 2. Also note that if Trojan's inputs are connected to nets with transition probabilities greater than P_{TH}, the average number of clock cycles is assumed to be within the acceptable range defined by user which can be estimated using Geometric Distribution.

6.3 P_{TH} = 10e-04

When P_{TH} = 10e-04, the dSFF insertion procedure identifies 28 nets with transition probability less than P_{TH}. In this case, 16 dSFFs are inserted to ensure these nets have greater than P_{TH} transition probability. Dummy flip-flop insertion causes 0.82% area overhead. The simulation results in Table 4 show that in all cases the number of clock cycles to generate a transition at Trojan output reduces significantly by about 30 times. Similarly, the number of transitions in the Trojan circuits and Trigger outputs have increased.

Inserting dSFF causes area overhead in the circuit. In fact, P_{TH} determines the amount of area overhead; the higher the threshold the more the area overhead. Table 5 shows the number of nets with lower than P_{TH} transition probability and the respective number of inserted dSFFs. As P_{TH} increases, the number of target nets increases and as a result the number of dSFFs increases. The total number of dSFFs depends also on the circuit topology. If the nets lower than P_{TH} are not connected to each other, more dSFFs are needed. However, if a net in the middle of a path has lower than P_{TH} transition probability and so are the following nets on the same path, then adding one dSFF can not only increase the transition probability of all these nets but also could impact those nets that branch out to other paths.

Also, note that large P_{TH} does not seem to be necessary. Even with P_{TH}=10e-4 and P_{TH}=10e-5, the number of transitions in the Trojans are very large that can help transient power-based analysis methods detect them easily. Thus, for larger designs, the area overhead would stay reasonably low.

Table 5: P_{TH} analysis.

Pth	10e-05	10e-04	10e-03
# nets	4	28	129
# dSFFs	4	16	54

• **Comment:** Please note that we are currently running our procedure on a larger benchmark circuit from ITC'99, called b19 which contains 230K gates and 7K flip-flops. If the paper gets accepted in HOST-2009, we include the results in the final version of the paper for various transition probability thresholds.

7 Conclusion

In this paper, we developed a novel dummy flip-flop insertion procedure to increase the probability of Trojan detection using transient power-based analysis or fully activation. The transitions are modeled using geometric distribution and we analyzed time to generate a transition in Trojan circuits or fully activate them. The dummy flip-flop insertion procedure is aim at reducing authentication time by increasing switching activity in Trojan circuits. The area overhead has been analyzed for various transition probability threshold. Simulation results for s38417 benchmark demonstrated that with negligible area overhead it is possible to significantly increase switching activity in the Trojan circuits and reduce time required to generate transitions.

Acknowledgements

The work of Hassan Salmani and Mohammad Tehranipoor was supported in part by the National Science Foundation Grant CNS-0716535. The work of Jim Plusquellic was supported in part by NSF grant CNS-0716559.

References

[1] U.S.D. Of Defense. "Defense science board task force on high performance microchip supply," *http://www.acq.osd.mil/dsb/reports/2005-02-HPMS_Report_Final.pdf*

[2] S. Adee "The Hunt for the Kill Switch," *http://www.spectrum.ieee.org/print/6171*

[3] X. Wang, M. Tehranipoor and J. Plusquellic, "Detecting malicious inclusions in secure hardware: Challenges and solutions," in Proc. of the *IEEE International Workshop on Hardware-Oriented Security and Trust(HOST 2008)*, pp. 15-19, 2008.

[4] M. Banga and M. S. Hsiao "A Novel Sustained Vector Technique for the Detection of Hardware Trojans," in Proc. of the *International Conference on VLSI Design*, pp. 327-332, 2009.

[5] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi and B. Sunar, "Trojan Detection using IC Fingerprinting," in Proc. of the *Symposium on Security and Privacy*, pp. 296-310, 2007.

[6] R. Rad, X. Wang, J. Plusquellic and M. Tehranipoor, "Power Supply Signal Calibration Techniques for Improving Detection Resolution to Hardware Trojans," in Proc. of the *International Conference on Computer-Aided Design (ICCAD08)*, pp. 632-639, 2008.

[7] J. Li and J. Lach, "At-speed delay characterization for IC authentication and Trojan Horse detection," in Proc. of the *IEEE International Workshop on Hardware-Oriented Security and Trust(HOST 2008)*, pp. 8-14, 2008.

[8] Y. Jin and Y. Makris, "Hardware Trojan detection using path delay fingerprint," in Proc. of the *IEEE International Workshop on Hardware-Oriented Security and Trust(HOST 2008)*, pp. 51-57, 2008.

[9] D. D. Wackerly, W. Mendenhall III and R. L. Scheaffer, "Mathematical Statistics with Application, 7th edition" Thomson Learning, Inc., 2008.

[10] X. Wang, H. Salmani, M. Tehranipoor and J. Plusquellic, "Hardware Trojan Detection and Isolation Using Current Integration and Localized Current Analysis," in Proc. of the *International Symposium on Fault and Defect Tolerance in VLSI Systems (DFT08)*, pp. 87-95, 2008.

[11] M. Bushnell and V. Agrawal, "Essentials of Electronics Testing," Kluwer Publishers, 2000.

[12] M. Banga, M. Chandrasekar, L. Fang and M. Hsiao, "Guided Test Generation for Isolation and Detection of Embedded Trojans in ICs," in Proc. of the *Symposium on Very Large Scale Integration*, pp. 363-366, 2008.

[13] M. Banga and M. Hsiao, "A Region Based Approach for the Detection of Hardware Trojans," in Proc. of the *IEEE International Workshop on Hardware-Oriented Security and Trust*, pp. 43-50, 2008.

[14] S. Jha and S. K. Jha, "Randomization Based Probabilistic Approach to Detect Trojan Circuits," in Proc. of the *IEEE High Assurance Systems Engineering Symposium(HASE08)*, pp. 117-124, 2008.

[15] F. Wolff, C. Papachristou, S. Bhunia and R.S. Chakraborty, "Towards Trojan-Free Trusted ICs: Problem Analysis and Detection Scheme," in Proc. of the *Design, Automation and Test in Europe(DATE '08)*, pp. 1362-1365, 2008.

[16] R. Sankaralingam, R. R. Oruganti and N. A. Touba, "Static Compaction Techniques to Control Scan Vector Power Dissipation," in Proc. of the *IEEE VLSI Test Symposium (VTS'00)*, pp. 35-40, 2000.