Lifetime Reliability Trojan based on Exploring Malicious Aging

***Abstract*—Due to escalating complexity of hardware design and manufacturing, not only are integrated circuits (ICs) de-signed and fabricated in multiple nations, but also software tools may be supplied worldwide. It makes hardware security become more subject to various kinds of tampering in the supply chain. Hardware Trojan horses (HTHs) can be implanted to facilitate the leakage of confidential information or cause the failure of a system. Reliability Trojan is one of the main categories of HTH attacks because its behavior is progressive and thus hard to be detected, or not considered malicious. In this work, we propose to insert reliability Trojan into a circuit which can finely control the circuit lifetime as specified by attackers (or even designers), based on manipulating BTI-induced aging behavior in a statistical manner. Experimental results show that, given a specified lifetime target and under the influence of process variations, the circuit is highly likely to fail within a desired lifetime interval, at the cost of little area overhead.**

I. INTRODUCTION

Due to escalating complexity of hardware design and manufacturing [1], integrated circuits (ICs) are usually designed and fabricated in multiple nations worldwide. Moreover, some design tools are also supplied from different units. With the aid of third-party IP components and off-shore IC manufacturing, the overall cost and time-to-market are significantly reduced. However, hardware security becomes more subject to various kinds of tampering in the supply chain [2][3]. Typically, a hardware system does no more than its requirements. Doing more than the required, hardware Trojan horses (HTHs) can be implanted to facilitate the leakage of confidential information or cause the failure of a system [4]-[5]. Outsourcing (e.g., third-party IP components, design tools and off-shore IC manufacturing) makes malicious HTH attacks possible. Here, we take two motivating scenarioes for introducing from HTH attacks as follows:

1) Manufacturers: Given a design house A with its competitor B, in order to interfere B's commercial development, A paid B's ICs manufacturer M such that M tampered B's layout, making B's product(s) malfunction earlier than expected.
2) Design-tool suppliers: Given a country C with its imaginary enemy D, and we assume that D's design houses utilize at least one design tool supplied from C's software corporation S. For the purpose of obstructing D's high-tech military weapon development, C forced S to embed malicious functions in its software merchandise. As a consequence, no matter what kinds of military equipment D produces, D is unaware of the fact that their designs are stealthily inserted HTHs.

After the insertion of HTHs, it is difficult to be aware of their existence since they are pervasive and imperceptable. In this sense, the proposed research provides new primitives for aforementioned hardware security threats, by exploring the feasibility of different HTH attacks and associated detection/prevention countermeasures.

Reliability Trojan is one of the main categories of HTH attacks because its behavior is progressive and thus hard to be detected, or not considered malicious. Time-dependent dielectric breakdown (TDDB), bias temperature instability (BTI), and electromigration (EM) are some of the critical failure mechanisms affecting lifetime reliability. With the continuous shrinking of transistor and interconnect dimensions, the rate of such progressive wear-out failures is getting higher. In addition, due to the increasing transistor density without proportional downscaling of supply voltage, the power density and thus the operating temperature will rise significantly, which further accelerates the failure mechanisms because they are all exponentially dependent on temperature.

Among various aging mechanisms, BTI is known for prevailing over other device aging phenomena, in terms of dependence on the scaling of nanometer technologies. BTI [6] is a MOSFET aging phenomenon that occurs when transistors are stressed under bias (positive or negative, i.e., $V_{gs} = \pm V_{dd}$) at elevated temperature. As a result of the dissociation of Si-H bonds along the Si-SiO2 interface, BTI-induced MOSFET aging manifests itself as an increase in the threshold voltage (V_{th}) and decrease in the drive current (Ids) [7], which in turn lengthen the propagation delays of logic gates/paths. Experiments on MOSFET aging [8] indicate that BTI effects grow exponentially with higher operating temperature and thinner gate oxide. If the thickness of gate oxide shrinks down to 4nm, the circuit performance can be degraded by as much as 15% after 10 years of stress and lifetime will be dominated by BTI [9]. Once the performance degradation exceeds a tolerable limit, the timing specification is no longer met and then timing errors begin to occur [10][11].

In this work, we propose to insert reliability Trojan into a circuit which can controls the circuit lifetime as specified by attackers (or even designers), based on manipulating BTI-induced aging behavior in a statistical manner. Studies about reliability Trojan have been proposed since last few years. Authors of [12] detail BTI and HCI effects which induce aging failures, and accelerates the effects by aggravating the most influential parameters of BTI and HCI. In [13], a few Trojan designs are proposed to accelerate EM, BTI and TDDB effects by stressing/modifying specific interconnects and gates. Some studies also try to control the lifetime of a circuit by counters or timers. In [14], authors present a Trojan which controls lifetime by analog mechanism. It siphons charge from target wire and stores to a capacitor until voltage on the capacitor rises above the threshold and sets its output flip-flop to a desired value. The work [15] presents an unmodified Trojan by analyzing the netlist of a circuit to identify its critical paths; then they generate patterns/instructions for stressing those paths. These patterns can be fed by external programs or embedded devices to accelerate the aging and decrease the circuit performance and lifetime. The studies [14] and [15] focus on the logic blocks which highly depend on users' operational modes. However, authors of [12] does not estimate circuit lifetime in detail and the work of [13] has relatively high cost based on using counters to control lifetime. To predict circuit lifetime with Trojans, a few mathematical models are proposed in [16] to estimate circuit reliability, but it only tries on tiny circuit C17 and does not consider aging. In addition, authors of [17] propose an idea using aging effects to induce a circuit into its redundant states (i.e., operational modes)

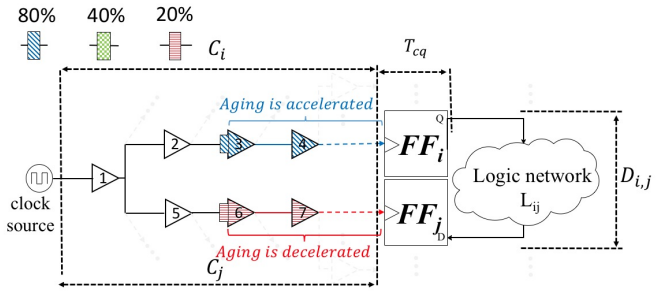


Figure 1. Example of DCC insertion

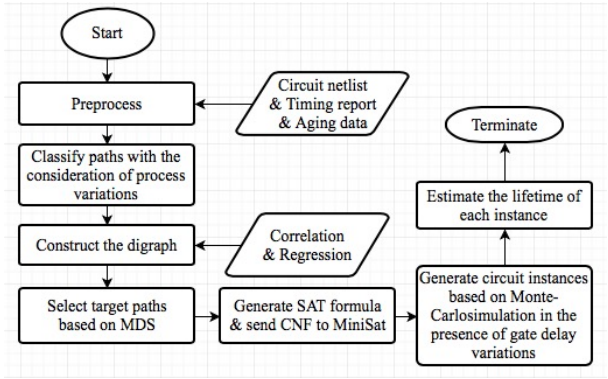


Figure 2. DCC insertion flow

and thereafter execute malicious function.

This paper proposes a method of hardware Trojan insertion to control the lifetime of a circuit based on manipulating the rate of circuit aging. We consider (i) the aging of both clock trees and combinational logical paths, and (ii) the correlation of aging rates between critical paths. These considerations ensure the effect of our proposed Trojan to be manifested on time under all possible workloads due to various users' operational conditions. More clearly, we present a methodology that deploys duty-cycle converters (DCCs) into a clock tree to accelerate the aging of predesignated clock buffers/inverters associated with critical paths. Those paths will fail around the time we set regardless of operational conditions. The advantage and contribution of our work are:

- **Low Cost and High Invisibility:** Aging-based reliability Trojan uses natural, time-consuming effect of a circuit. There is no unusual behavior on performance or function when chips are just manufactured so it cannot be detected by normal tests [13]. Our inserted Trojan modifies only a low fraction of the circuit and just changes the duty cycle of some sub clock trees. The modifications are marginal so it is hard to be found from the layout. Also, clock wave form be changed is invisible except we add test points into the sub clock trees which are impacted [13].
- **Low Dependence on User Operation:** We ensure that the proposed Trojan can attack the circuit successfully by analyzing the correlation between aging rate sof critical paths. Then we transform the problem to a Minimum Dominating Set (MDS) problem for minimizing the cost and guarantee at least one of the critical paths would fail under any operation mode.

II. MOTIVATING EXAMPLE

A. Duty-Cycle Converter (DCC)

Duty cycle is the percentage of one period in which a signal is high (i.e., logic 1). The aging of logic gates highly depends on the stress time [11]. For a clock buffer on the clock tree, its stress time is proportional to the clock duty cycle. Therefore, by adjusting the clock duty cycle, we can manipulate the aging of clock buffers and then control the effective degradation of logic paths. The unit we use to change the clock duty cycle is duty-cycle converter (DCC) [18], which can converts the duty cycle of a clock signal to a smaller/larger one (e.g., $50\% \rightarrow 20\%$ or $50\% \rightarrow 80\%$). Once a DCC is inserted into the clock tree, the downstream sub-tree of the DCC insertion point will receive a clock signal whose duty cycle is no longer 50%. This way, aging rate manipulation of downstream clock buffers can be achieved.

B. DCCs against a Critical Path

We use an illustrative example to explain our idea of shortening the lifespan of designs, by manipulating the aging rates of clock buffers. Consider the circuit in Figure 6, where FF_i and FF_j are edge-triggered flip-flops and there exist seven buffers in the associated clock network. If the design needs work normally, the following setup-time constraint must be satisfied:

$$C_i + T_{cq} + D_{ij} + T_{su} < C_j + T_c \tag{1}$$

where C_i is the clock latency from clock source to FF_i, C_j is the counterpart from clock source to FF_j, T_{su} is setup time, T_c is clock period, T_{cq} is clock-to-output delay, and D_{ij} is the largest path delay of logic network L_{ij}. The constraint is equivalent to the following constraint:

$$Slack = (C_j + T_c) - (C_i + T_{cq} + D_{ij} + T_{su}) > 0 \tag{2}$$

which indicates the timing slack must be greater than zero; otherwise, the design fail to work normally. Suppose that, the attacker inserts a 20% DCC and an 80% DCC at the inputs of buffer 6 and buffer 3, respectively. Over a period, C_i will gain greater than C_j does. This way, the timing slack likely decreases to a negative value, such that setup-time violation occurs on this critical path, causing the failure of the designs.

III. PROPOSED FRAMEWORK

The overall flow of the proposed framework for DCC insertion/deployment is depicted in Figure 2. The proposed framework focuses on the two issues:

1) **Overhead minimization**: Attacking all critical paths may be infeasible or may increase the used DCC count, which denotes the area overhead of the attack. Thus, the framework must filter/classify critical paths to make the attack successful and to minimize the DCC count.
2) **Workload variations**: Because users' workload highly impact the degradation of logic paths, the proposed framework must consider users' countless operational modes (i.e., workload). To be workload-aware, the problem of selecting target paths to be attacked is transform to a graph problem, which can be solved using the existing algorithms.

The section is organized as follows: Section III-A discusses the PV-aware methodology of classifying critical path, considering the correlation between PVs and BTI, which is introduced in Section III-B. Section III-C details the workload-aware methodology of selecting target paths to be attacked and Section III-E introduces the SAT-based formulation of DCC insertion/deployment.

A. Classification of Critical Paths

Given a critical path, the path is classified into three groups: *Shortlist*, *Candidate*, and *Mine*, depending on the lifetime* distribution† of the critical path, under all possible DCC deployments on the associated clock network. The lifetime distribution of the path is further analyzed with three lifetime intervals, which are defined as follows: $[0, n-\varepsilon]$, $[n-\varepsilon, n+\varepsilon]$, and $[n+\varepsilon, \infty]$, where n is the expected circuit lifetime under the proposed HTH attack and ε is mxximum tolerable error. The lifetime distributions of the path in the three intervals determine the classification of the critical path.

- *Candidate*: A critical path is defined as a candidate if there at least exists one DCC deployment, which leads the critical path to fail within $[n-\varepsilon, n+\varepsilon]$‡.
- *Mine*: A critical path is defined as a mine if it satisfies the following conditions: (i) The path is not a candidate. That is, on the associated clock paths of the critical path, there is no DCC deployment to control the path lifetime within $[n-\varepsilon, n+\varepsilon]$. (ii) On the associated clock paths, there at least exists one DCC deployment, which leads the critical path to fail within $[0, n-\varepsilon]$, i.e., it lead the critical path to fail prematurely.
- *Shortlist*: Critical paths in *shortlist* is the subset of *Candidate*, which are selected as target paths to be attacked. Attacking such paths involves deploying DCCs on their associated clock paths.

To make the classification PV-aware, we conduct Monte-Carlo simulation on the delays of logic paths from statistical perspective. The simulation is performed by imposing extra V_{th} offset (i.e., ΔV_{th_pv}) on each transistor of logic paths. Then, we check the setup-time constraint using Equation (2), considering the correlation of PVs and BTI, which is introduced in the next subsection.

B. Path Delay Estimation considering the Correlation between PVs and BTI

The correlation is a long-term phenomenon that bridge the V_{th} differences among the transistors over a period. Further, a positive/negative V_{th} offset leads to a higher/lower fresh V_{th}, causing a lower/higher aging speed. Therefore, the gap between high and low V_{th} will be gradually converged, letting threshold voltages of transistors, whose fresh ones are different, reach a convergent value. We use a model proposed in [19] to estimate the correlation between fresh V_{th} offset (caused by PVs) and BTI effects:

$$\Delta V_{th\_bti} = (1 - S_v \cdot \Delta V_{th\_pv}) \cdot A \cdot \alpha^n \cdot t^n \tag{3}$$

where ΔV_{th_pv} is the fresh V_{th} offset caused by PVs. ΔV_{th_bti} is the BTI-induced V_{th} shift, α is the stress duty cycle, A is $3.9 \times 10^{-3} V \cdot s^{-1/5}$, n is time exponential constant, 0.2 for used technology, and S_v is a constant which can be extracted by fitting HSPICE simulation results in 45nm TSMC technology. So far, given a specific ΔV_{th_pv} imposed on the V_{th} of a transistor, we use Equation (3) to derive the corresponding ΔV_{th_bti}, which can be further transformed to BTI-induced delay shift, using the model proposed in [20].

$$\Delta t_{p\_aged} = C \cdot \Delta V_{th\_bti} \tag{4}$$

*The lifetime of a critical path is defined as when the timing violation occurs on the path, in the presence of aging.

†Given a critical path, a DCC deployment on the associated clock paths results in an individual lifetime value of the critical path. Thus, numerous DCC deployments on the associated clock paths forms the lifetime distribution of the path.

‡Because of the effects of PVs and workload variations, it is impossible to precisely control the circuit lifetime at the expected lifetime n, thus, the Trojan attack, controlling the circuit lifetime within $[n-\varepsilon, n+\varepsilon]$ is accepted/desired. Therefore, the lifetime interval $[n-\varepsilon, n+\varepsilon]$ is defined as *desired lifetime interval*.

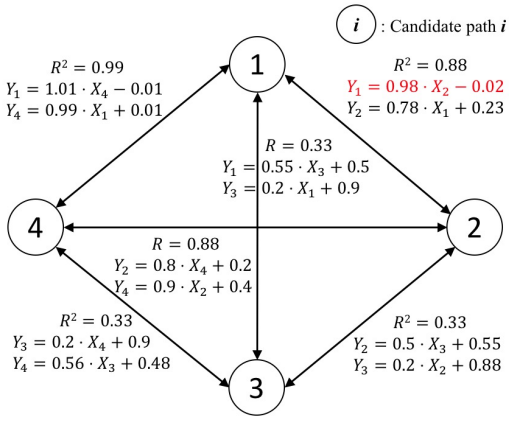


Figure 3. Example of graph used in choosing targets

where Δt_{p_aged} is BTI-induced shift of propagation delay, and C is a constant and is fitted to 0.5 after SPICE simulation. Further, the Equation (4) is modified as following Equation (5) to account for the conversion from ΔV_{th_pv} to intrinsic delay shift.

$$\Delta t_{p\_intrinsic} = C \cdot \Delta V_{th\_pv} \tag{5}$$

where $\Delta t_{p\_intrinsic}$ is the propagation delay shift caused by ΔV_{th_pv}. Up to now, a model is built to convert a given specific ΔV_{th_pv} to corresponding Δt_{p_aged} and $\Delta t_{p\_intrinsic}$. The model is not only used in the classification of critical paths, but also used in Section IV while estimating the lifetime intervals of Monte-Carlo instances of attacked designs.

C. Selection of Target Paths (Shortlist) to be attacked considering Workload Variations

The uncertainty of user-dependent operational modes (e.g., watching video, playing games) can influence/vary the aging behavior of logic paths. Therefore, we must ensure that the attack will succeed under any operational mode. We make the following assumption, which is also used in Section IV to estimate the lifetime of attacked designs:

Assumption: Every operational mode causes at least one candidate path to undergo worst-case aging.

In other words, no matter how users operate the design, at least one candidate path in the design undergoes worst-case aging. Moreover, given a path, if an operational mode leads the path to undergo worst-case aging, the operational mode is defined as the *critical operational mode* of the path:

Definition: An operational mode, which leads the path to undergo worst-case aging, is defined as a critical operational mode of the path.

This way, the following corollary can be obtained:

Corollary: The union of critical operational modes of all candidate paths is equivalent to the universe of operational modes.

Therefore, attacking all candidate paths is a naïve method to guarantee the successful attack regardless of the users' workload variations. Nevertheless, it is very costly and may be impossible to attack all candidate paths. However, while we attack the part of candidate paths, it is still possible to make the attack successful if we consider the correlation of aging behaviors among critical paths. Specifically speaking, we observe that the aging behaviors of many paths are highly correlated. For instance, given two critical paths A

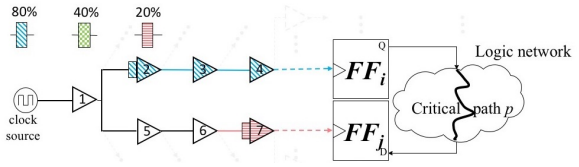


Figure 4. A DCC deployment leads path p to fail prematurely

and B, suppose that A and B are highly correlated in terms of their aging behaviors, the critical operational mode of A, O_A, must lead A to age in the worst-case, according the aforementioned definition. In addition, O_A also leads B to age to a similar extent because A and B age close/similarly. Consequently, even if we simply attack path A, both O_A and O_B can make the attack successful. In short, we can simply attack one out of those highly correlated paths to cover multiple operational modes. This property helps reduce the count of targeted paths to be attacked.

D. Selection of Target Paths using MDS (Minimum Dominating Set)

Based on the aforementioned property of aging correlation of paths, a directed graph (also known as digraph) is constructed as shown in Figure 3, where vertices represent candidates and arcs (i.e., directed edges) are correlation coefficients (R^2) and linear regression equations between each pair of vertices. Each arc has a regression equation, where X_i denotes the worst-case aging rate of path i, Y_j denotes the aging rate of path j predicted based on the linear regression equation and the other coefficients are obtained by running functional simulation. The exact value of X_i can be derived by the following predictive model presented in [20]:

$$A \cdot \alpha^n \cdot t^n \tag{6}$$

where A and n are fitted constants, α denotes the stress duty cycle, and t denotes time (unit is second). α is usually set to 0.5. A and n are fitted as 0.0039 and 0.2, respectively, after SPICE simulation.

Consider the red equation in Figure 3:

$$Y_1 = 0.98 \cdot X_2 - 0.02 \tag{7}$$

where X_2 is aging rate of vertex/path 2 and Y_1 is the aging rate of vertex/path 1, which can be predicted as 0.98 multiplied by X_2 minus 0.02. Moreover, after the graph is constructed, it is further simplified by removing some arcs, which indicate the weak aging correlation between pairs of paths/vertices.

The cost of our proposed HTHs is the inserted DCC count. To minimize the cost, we select minimum-sized targets to cover/dominate all candidate paths in the digraph, such that all users' operational modes are considered based on the corollary. This problem is similar to a classical digraph problem, **Minimum Dominating Set (MDS)**:

On digraph $G = (V, E)$, find a minimum-sized set of vertices $S \subseteq V$ such that $\forall y \notin S$, $\exists x \in S$, there exists an arc from x to y. And we say that y is dominated by x.

Therefore, the problem of selecting target paths to be attacked is transformed to a MDS-related digraph problem, which can be solved using the existing algorithms proposed in [21][22].

E. SAT-based Problem Formulation and Encoding for DCC Deployment

After the shortlist (i.e., target paths to be attacked) is determined, the problem of DCC deployment on their associated clock paths is formulated as a **Boolean satisfiability (SAT)** problem. The authors of [18] also use the SAT-based approach to determine the DCC deployment in the existing clock tree, while they insert DCCs to improve aging tolerance of designs. The key of our framework is to represent the problem in *conjunctive normal form* (CNF). A CNF representation is a conjunction of one or more clauses, where each clause is a disjunction of one or more Boolean variables. Thus, DCC deployment/insertion needs to be encoded into Boolean representation before being transformed into a SAT-based formulation. Assume that a total of 3 types of DCCs can be chosen (i.e., 20%, 40%, and 80% DCCs). Including the DCC-free case where no DCC is inserted, there are 4 possibilities of DCC insertion for each clock buffer. Given a clock buffer p, the four possibilities of DCC insertion at the input of buffer p can be encoded as follows using two Boolean variables $B_{p,2}$ and $B_{p,1}$:

	DCC type	$\{B_{p,2}, B_{p,1}\}$
(1)	None	{0,0}
(2)	20%	{0,1}
(3)	40%	{1,0}
(4)	80%	{1,1}

To control the circuit lifetime within $[n - \varepsilon, n + \varepsilon]$, timing constraints of DCC deployments are formulated in the SAT-based problem. The formulations of timing constraints depend on the classification of critical paths.

1) Paths in the shortlist (i.e., targets): The lifetime (i.e., when timing violations occur) of shortlist paths dominate the lifetime of attacked designs. To control the lifetime of shortlist paths within $[n - \varepsilon, n + \varepsilon]$, on their associated clock paths, we formulate all DCC deployments, which lead the path to fail prematurely or post-maturely (i.e., within $[0, n - \varepsilon]$ or within $[n + \varepsilon, \infty]$), such that the SAT solver does not output the corresponding deployment in the result if the CNF is satisfiable.
2) Other paths (paths not in the shortlist): While attacking shortlist paths, the DCC deployment, on the clock paths of shortlist paths, may cause the premature failure of other paths not in shortlist, due to the common clock network, making the proposed Trojan attack premature and inaccurate. Thus, on the associated clock paths of other paths not in shortlist, we formulate all DCC deployments, which lead the paths to fail prematurely (i.e., within $[0, n - \varepsilon]$), such that the SAT solver does not output the corresponding deployment in the result if the CNF is satisfiable.

Consider the example in Figure 4, where the 80% and 20% DCCs are inserted at the inputs of buffer 2 and 7, respectively. Assume that the critical path p is in shortlist. If the DCC deployment will lead the path p to fail prematurely (i.e., path fail within $[0, n - \varepsilon]$), then the following clause

$$(A_1 \vee A_0 \vee \neg B_1 \vee B_0 \vee C_1 \vee C_0)$$

will be generated and added into CNF, such that the solver will not output the corresponding DCC deployment in the result if the CNF is satisfiable.

For SAT-based formulation, our proposed problem of DCC deployment is transformed into CNF clauses. The CNF clauses are solved by SAT solver such as MiniSat and we can find the locations and types of inserted DCCs by decoding the output from the solver.

IV. LIFETIME ESTIMATION

In this section, we propose a methodology to estimate the lifetime interval of attacked designs, considering the workload variations.

It is worth reminding the previous corollary: The union of critical operational modes of all candidate paths is equivalent to the universe of operational modes. That is, after applying worst-case aging on

each candidate path, all operational modes are considered, during the lifetime estimation in our methodology.

V. EXPERIMENTAL SETTING AND RESULTS

In this section, we explain the experimental setting and demonstrate the experimental results of our proposed Trojan attack. The benchmarks in IWLS'05 and ISCAS'89 are used in the experiments. The utilized technology is TSMC 65nm GP standard cell series. The used SAT solver is MiniSAT 2.2. The section is organized as follows: Section V-A introduces the experimental setting for clock period. Section V-D demonstrates the lifetime distributions of Monte-Carlo instances of attacked designs, Eventually, Section V-E discuss the detectability of the proposed Trojan attack.

A. Clock Period Setting

Figure 5 shows the lifetime intervals of original (i.e., Trojan-free) designs with clock periods which make the designs fail at a specified time (in our experiment, 7 years) under aging. The resulting clock period is both used in Trojan-free and Trojan-included (attacked) designs. In Figure 5, lower bounds are exactly 7 years because circuit clock periods (shown in column 4) are specifically set such that the most critical path, whose slack is smallest, fails at 7^{th} year under the worst-case aging condition. The upper bound of each design differs significantly because, in the Trojan-free designs, only the most critical path is considered for determined the clock period while workload variations are disregarded.

B. Monte-Carlo Instantiation of the Attacked Designs

Given an attacked design, each Monte-Carlo instance of the design is generated by imposing extra V_{th} offset (i.e., ΔV_{th}) on each transistor. The offsets follow a normal distribution with the standard deviation of a given value (usually 10mV - 30mV [23][24]). Each instance (i.e., Monte-Carlo seed) can be considered as a die after the circuit is manufactured. In our experiment, each attacked design is instantiated for 1000 times with a specified standard deviation of V_{th}.

C. Lifetime Estimation considering the Effect of PVs on aging rates of transistors

Whenever an instance is generated, Algorithm **??** and **??** are applied to estimate its lifetime interval. Note that, because threshold voltages of transistors along the same path are not fixed due to PVs, their aging rates differ. Thus, at line 8 in Algorithm **??**, aging rate of path i (X_i) must consider the aging rate of individual transistor, instead of using the deterministic Equation (6). The aging rate of individual transistor can be obtained using the mathematical model in Section refsec:frame, which discusses discusses the correlation between PVs and BTI.

D. Lifetime Distribution of Monte-Carlo Instances

Figure 6 shows the lifetime distributions of instances of the attacked three designs (*s38417*, *des_perf* and *leo3mp*). The designs are attacked to fail at 3^{rd}, 4^{th} or 5^{th} year. Note that, there exists no SAT solution while leo3mp is attacked to fail at 3^{rd} year, whereas there exists SAT solution while it is attacked to fail at 4^{th} year. In each subfigure, there exist four distributions, each of which corresponds to one distinct value of standard deviation of V_{th} while generating Monte-Carlo instances. In our experiments, the deviations are set to 15mV, 20mV, 25mV and 30mV, respectively. Moreover, in each distribution, there exist two peaks. The left/right peak denotes the distribution of lower/upper bounds of lifetime intervals of instances. Note that, there exist two differences between Figure 5 and Figure 6. First, the designs in Figure 5 are Trojan-free ones, instead of Trojan-included ones in Figure 6. Second, because the lifetime intervals of the Trojan-free designs are not subject to the PVs, the original lifetime intervals (5^{th} column in Figure 5) do not consider the effect of PVs.

Apparently, Figure 6 shows that, as the standard deviation becomes larger, the interval between the left and right peaks becomes wider, indicating the larger standard deviation leads to a less accurate attack. Therefore, the lifetime accuracy of the proposed Trojan is impacted by the diversity of threshold voltages of transistors. However, even though the peaks of two bounds deviate from the desired lifetime interval $[n-\varepsilon, n+\varepsilon]$, it does not mean that the attacked designs must not fail in that interval. As mentioned in Section IV, the estimated lifetime interval of one instance consists of two bounds. One is lower bound; and the other is upper bound. The two bounds denote the earliest and the last time points, at which the instance will fail. But the exact time point, at which the instance fails, depends on the workload. Therefore, since the lifetime interval of each instance is overlapped with the desired lifetime interval $[n-\varepsilon, n+\varepsilon]$, the proposed Trojan is still likely to control the design lifetime in that interval.

E. Detectability

Side-channel analysis is a technique often used to detect the existence of hardware Trojan. Nevertheless, the used DCC count is marginal compared with the total gate count. On average, DCC count is less than 0.2% of total gate count. That is, the area overhead is insignificant. Also, the power overhead due to DCCs can be regarded as the power variations caused by PVs. As a result, the proposed Trojan is difficult to be detected by conventional side-channel analysis.

Some Trojan defenders can insert probes in the clock network to inspect the variation of clock duty cycle. The detection method is indeed able to prove the existence of the proposed Trojan. However, the method need to be supported by extra I/O pins/ports. Therefore, it is impractical, not only because the pin counts of ICs are limited, but also the area overhead of extra pins is costly.

VI. CONCLUSION

We proposed a methodology of hardware Trojan insertion to control the circuit lifetime with the consideration of aging behavior, correlation between pairs of critical paths and process variations. The influence of Trojan heavily reduce the lifetime of circuit instances. Even though the accuracy is impacted by PVs, the lifetime of instances is still likely to fail within the desired lifetime interval $[n-\varepsilon, n+\varepsilon]$. Also, the DCC count is less than 0.2% of total gate count, implying limited area and power overhead. Therefore, the proposed Trojan is difficult to be detected.

REFERENCES

[1] L. Wilson, "International technology roadmap for semiconductors (itrs)," *Semiconductor Industry Association*, 2013.

[2] M. Tehranipoor *et al.*, "Trustworthy hardware: Trojan detection and design-for-trust challenges," *Computer*, vol. 44, no. 7, pp. 66–74, 2011.

[3] R. Karri *et al.*, "Trustworthy hardware: Identifying and classifying hardware trojans," *Computer*, vol. 43, no. 10, pp. 39–46, 2010.

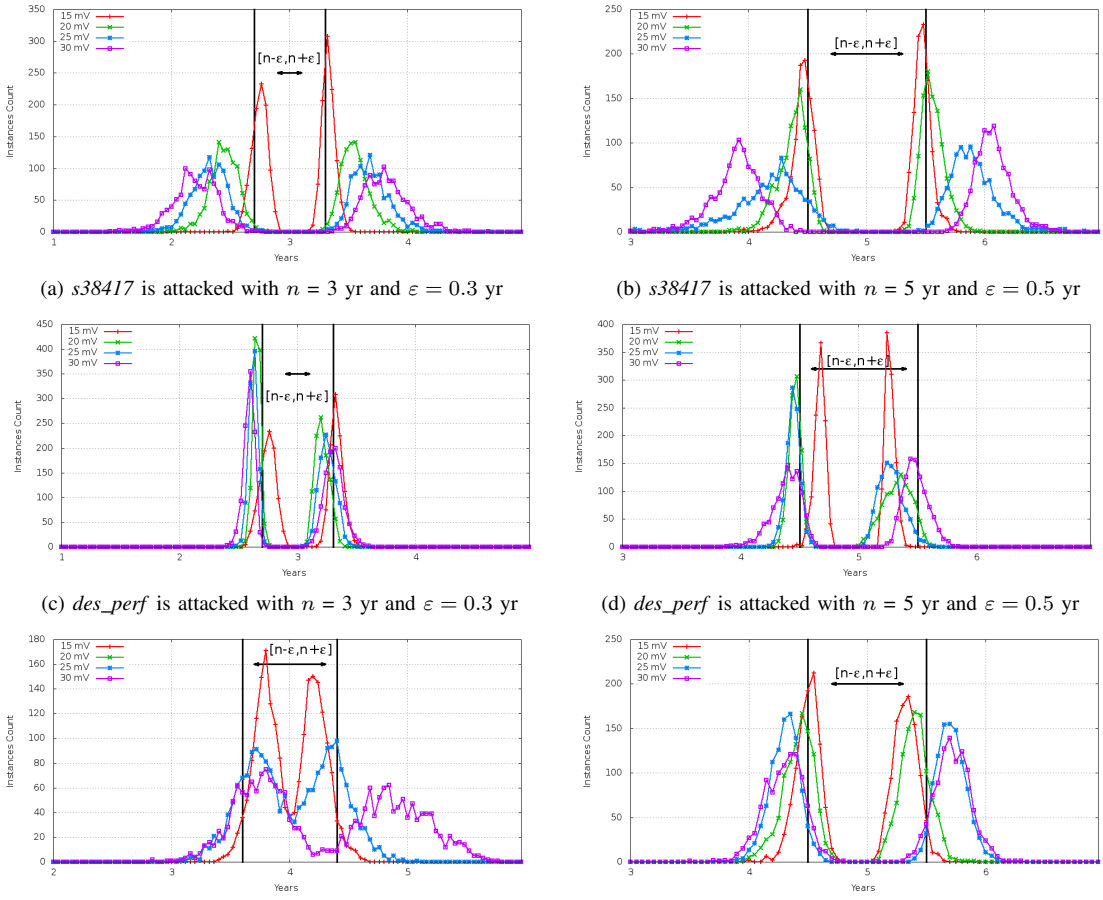
[4] S. Adee, "The hunt for the kill switch," *IEEE Spectrum*, vol. 45, no. 5, pp. 34–39, 2008.

[5] S. Bhunia *et al.*, "Hardware trojan attacks: Threat analysis and countermeasures," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1229–1247, 2014.

[6] D. K. Schroder and J. A. Babcock, "Negative bias temperature instability: Road to cross in deep submicron silicon semiconductor manufacturing," *Journal of applied Physics*, vol. 94, no. 1, pp. 1–18, 2003.

Benchmark	# Gates	# FFs	# Clock buffers	Max Tree Level	Clock Period (ps)	Original Lifetime
des_perf	7401	8802	3416	11	3898	7.00 - 7.06
leo3mp	526297	108839	4924	10	3449	7.00 - 11.38
netcard	561091	97831	7904	12	922	7.00 - 8.45
vga_lcd	101496	17079	4345	10	976	7.00 - 9.07
s38417	8422	1564	376	6	999	7.00 - 10.71

Figure 5. Circuit information and estimated lifetime without Trojan insertion



(a) *s38417* is attacked with n = 3 yr and $\varepsilon = 0.3$ yr

(b) *s38417* is attacked with n = 5 yr and $\varepsilon = 0.5$ yr

(c) *des_perf* is attacked with n = 3 yr and $\varepsilon = 0.3$ yr

(d) *des_perf* is attacked with n = 5 yr and $\varepsilon = 0.5$ yr

(e) *leo3mp* is attacked with n = 4 yr and $\varepsilon = 0.4$ yr

(f) *leo3mp* is attacked with n = 5 yrs and $\varepsilon = 0.5$ yr

Figure 6. Lifetime distributions of Monte-Carlo Instances of *s38417*, *des_perf*, and *leo3mp*

[7] S. Zafar, "The negative bias temperature instability in mos devices," in *Integrated Reliability Workshop Final Report, 2006 IEEE International*, IEEE, 2006, pp. 217–217.

[8] S. Chakravarthi *et al.*, "A comprehensive framework for predictive modeling of negative bias temperature instability," in *Reliability Physics Symposium Proceedings, 2004. 42nd Annual. 2004 IEEE International*, IEEE, 2004, pp. 273–282.

[9] N. Kimizuka *et al.*, "The impact of bias temperature instability for direct-tunneling ultra-thin gate oxide on mosfet scaling," in *VLSI Technology, 1999. Digest of Technical Papers. 1999 Symposium on*, IEEE, 1999, pp. 73–74.

[10] S. V. Kumar, C. H. Kim, and S. S. Sapatnekar, "An analytical model for negative bias temperature instability," in *Proceedings of the 2006 IEEE/ACM international conference on Computer-aided design*, ACM, 2006, pp. 493–496.

[11] W. Wang *et al.*, "The impact of nbti effect on combinational circuit: Modeling, simulation, and analysis," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 18, no. 2, pp. 173–183, 2010.

[12] Y. Shiyanovskii *et al.*, "Process reliability based trojans through nbti and hci effects," in *Adaptive Hardware and Systems (AHS), 2010 NASA/ESA Conference on*, IEEE, 2010, pp. 215–222.

[13] A. Sreedhar, S. Kundu, and I. Koren, "On reliability trojan injection and detection," *Journal of Low Power Electronics*, vol. 8, no. 5, pp. 674–683, 2012.

[14] K. Yang *et al.*, "A2: Analog malicious hardware," in *Security and Privacy (SP), 2016 IEEE Symposium on*, IEEE, 2016, pp. 18–37.

[15] N. Karimi *et al.*, "Magic: Malicious aging in circuits/cores," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 12, no. 1, p. 5, 2015.

[16] S. Burman *et al.*, "Effect of malicious hardware logic on circuit reliability," in *Progress in VLSI Design and Test*, Springer, 2012, pp. 190–197.

[17] S. Wei and M. Potkonjak, "The undetectable and unprovable hardware trojan horse," in *Proceedings of the 50th Annual Design Automation Conference*, ACM, 2013, p. 144.

[18] K.-C. Wu, T.-H. Tseng, and S.-C. Li, "Maui: Making aging useful, intentionally," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2018*, IEEE, 2018, pp. 527–532.

[19] A. F. Gomez and V. Champac, "Early selection of critical paths for reliable nbti aging-delay monitoring," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 7, pp. 2438–2448, 2016.

[20] W. Wang *et al.*, "An efficient method to identify critical gates under circuit aging," in *Computer-Aided Design, 2007. ICCAD 2007. IEEE/ACM International Conference on*, IEEE, 2007, pp. 735–740.

[21] O. Ore, *Theory of graphs*. American Mathematical Society, 1962, vol. 38.

[22] K. Natarajan and L. J. White, "Optimum domination in weighted trees," *Information processing letters*, vol. 7, no. 6, pp. 261–265, 1978.

[23] S. Han *et al.*, "Statistical aging analysis with process variation consideration," in *Proceedings of the International Conference on Computer-Aided Design*, IEEE Press, 2011, pp. 412–419.

[24] C. Schlünder *et al.*, "On the influence of bti and hci on parameter variability," in *Reliability Physics Symposium (IRPS), 2017 IEEE International*, IEEE, 2017, 2E–4.