# Lifetime Reliability Trojan based on Exploring Malicious Aging

Tien-Hung Tseng, Shou-Chun Li and Kai-Chiang Wu
Department of Computer Science
National Chiao Tung University, Hsinchu, Taiwan
E-mail: {eric830303.cs05g@g2.nctu.edu.tw, scli.cs02g@nctu.edu.tw and kcw@cs.nctu.edu.tw}

***Abstract*—Due to escalating complexity of hardware design and manufacturing, not only are integrated circuits (ICs) designed 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 is thus not trivial 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. Instead, in a typical design considering pessimistic corners, the resulting lifetime is usually far longer than the target.**

## I. INTRODUCTION

Due to escalating complexity of hardware design and manufacturing [1], integrated circuits (ICs) are usually designed and fabricated in multiple locations worldwide. Moreover, some design tools are also supplied from different units. With the utilization 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 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. In order to introduce the motivation of deploying HTH attacks, here we make some scenario from various aspects, i.e., attackers. 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. 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 mechanisms in its software merchandise. Therefore, no matter what kinds of military equipment D produces, they are stealthily put HTH in the inner ICs. After the insertion of HTHs, it is difficult to prove their existence since they are pervasive and inappreciable. 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 is thus not trivial to be detected, or not considered malicious. Time-de- pendent dielectric breakdown (TDDB), bias temperature in- stability (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. In this work, we propose to insert reliability Trojan into a circuit which can finely controls the circuit lifetime as specified by attackers (or even designers), based on manipulating BTI-induced aging behavior in a statistical manner.

## II. RELATED WORK

Studies about reliability Trojan have been proposed since last few years. Authors of [6] detail BTI and HCI effects which induce aging failures, and accelerates the effects by aggravating the most influential parameters of BTI and HCI. In [7], 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 [8], 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 [9] 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 [8] and [9] focus on the logic blocks

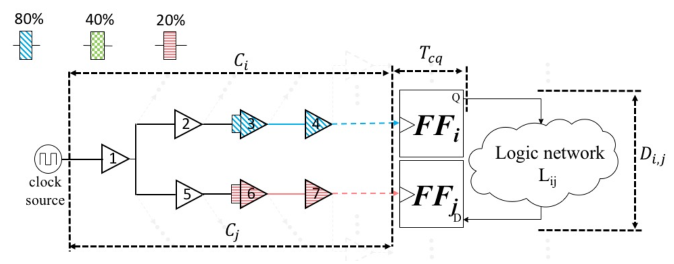


**Figure** 1. Example of DCC insertion

which highly depend on users' operational modes. However, authors of [6] does not estimate circuit lifetime in detail and the work of [7] has relatively high cost based on using counters to control lifetime. To predict circuit lifetime with Trojans, a few mathematical models are proposed in [10] to estimate circuit reliability, but it only tries on tiny circuit C17 and does not consider aging. In addition, authors of [11] propose an idea using aging effects to induce a circuit into its redundant states (i.e., operational modes) 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.

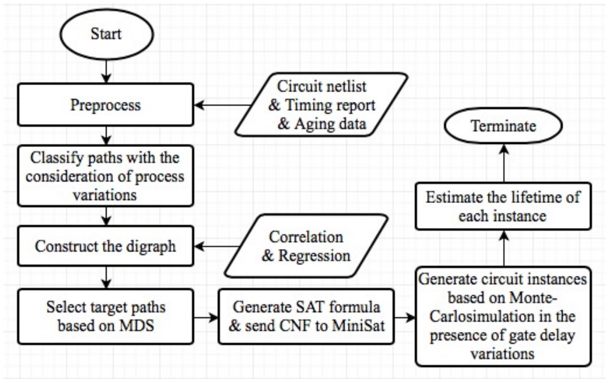
## III. MOTIVATING EXAMPLE

### 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 [8]. 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), which is introduced in [cite our paper]. It 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:



Start
Preprocess
Circuit netlist & Timing report & Aging data
Classify paths with the consideration of process variations
Construct the digraph
Correlation & Regression
Select target paths based on MDS
Generate SAT formula & send CNF to MiniSat
Generate circuit instances based on Monte-Carlosimulation in the presence of gate delay variations
Estimate the lifetime of each instance
Terminate

**Figure** 2. DCC insertion flow

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

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 occur on this critical path, causing the failure of the design circuit.

## IV. 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 IV-A discusses the classification of critical paths, Section IV-B details the workload-aware methodology to select target paths to be attacked, and Section IV-C 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 $\varepsilon$ is mximum 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.

## *B. Selection of Target Paths (Shortlist) to be Attacked*

If an attacked critical path always ages as estimated (i.e., under worst-case aging), a successful attack can be obtained just by inserting DCCs on its associated clock paths. However, uncertainty of user-dependent operational modes (e.g., watching video, playing games) can influence/vary the aging behavior. Therefore, we must ensure that the attack will succeed under any operational mode. We make the following assumption, which is also used in Section V to estimate the lifetime of attacked designs:

*"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, an operational mode, which causes one path to undergo worst-case aging, is defined as the *critical operational mode* of the path. This way, 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 nave method to guarantee the successful attack, based on the assumption. Nevertheless, it is very costly and may be impossible.

[*]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*.

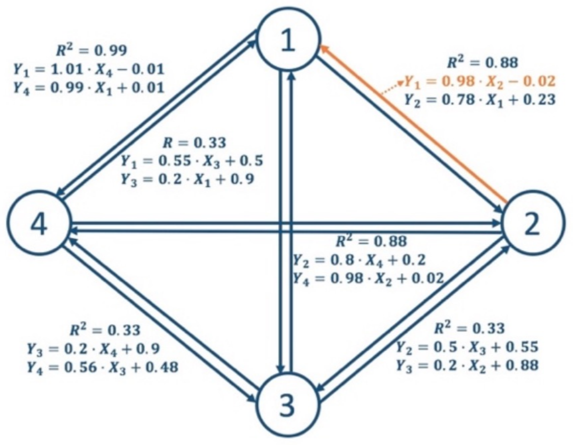

1
2
3
4
$R^2 = 0.99$
$Y_1 = 1.01 \cdot X_4 - 0.01$
$Y_4 = 0.99 \cdot X_1 + 0.01$
$R^2 = 0.88$
$Y_1 = 0.98 \cdot X_2 - 0.02$
$Y_2 = 0.78 \cdot X_1 + 0.23$
$R = 0.33$
$Y_1 = 0.55 \cdot X_3 + 0.5$
$Y_3 = 0.2 \cdot X_1 + 0.9$
$R^2 = 0.88$
$Y_2 = 0.8 \cdot X_4 + 0.2$
$Y_4 = 0.98 \cdot X_2 + 0.02$
$R^2 = 0.33$
$Y_3 = 0.2 \cdot X_4 + 0.9$
$Y_4 = 0.56 \cdot X_3 + 0.48$
$R^2 = 0.33$
$Y_2 = 0.5 \cdot X_3 + 0.55$
$Y_3 = 0.2 \cdot X_2 + 0.88$

**Figure** 3. Example of graph used in choosing targets

Therefore, after observing the relationship among aging of paths, we find that the aging behaviors of many paths are highly correlated. If several paths are highly correlated in terms of aging behavior, one operational mode can lead all of them to age to a similar extent. Thus, we can simply attack one out of those highly correlated paths to cover multiple operational modes. For example, given two critical paths $A$ and $B$. Their critical operational modes are $O_A$ and $O_B$, respectively. Assume that $A$ and $B$ are highly correlated in terms of aging behavior. Because $A$ and $B$ age similarly/closely, $O_A$ causes $A$ to age in the worst-case and also causes $B$ to age severely. Consequently, even if we simply attack path $A$, not only $O_A$ but also $O_B$ can make the attack successful (i.e., shorten the lifetime of path $A$ to the interval $[n-\varepsilon, n+\varepsilon]$). This property helps reduce the count of targeted paths to be considered/formulated. To choose the attack targets (shortlist), we transform the relationship of paths to a directed graph (also known as digraph). In Figure 3, 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, whose coefficients are obtained by running functional simulation. $X_i$ denotes the worst-case aging rate of path $i$, whose exact value will be introduced in Section [Undefined]. $Y_j$ denotes the aging rate of path $j$ predicted based on the linear regression equation. Consider the orange equation in Figure 3:

$$Y_1 = 0.98 \cdot X_2 - 0.02$$

Given the worst-case aging rate of vertex/path 2, $X_2$, the aging rate of vertex/path 1, $Y_1$, can be predicted as 0.98 multiplied by $X_2$ minus 0.02. Before the shortlist is determined by selecting a subset of candidates in the graph, we can simplify the graph by removing some arcs which indicate the relationships of weak aging correlation between pairs of paths.

The cost of our proposed HTHs is the inserted DCC count. To minimize the cost, we select minimum-sized targets to cover all candidate paths, that is, to dominate all candidate paths in the digraph, such that all users' operational modes are considered in the proposed Trojan attack, based on the assumption. 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 [12][13].

### *C. 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 key of the 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.

**Algorithm 1:** Lifetime Estimation

```
Input: Digraph $G = \{V, A\}$
       Vertices Set $V = \{Candidate, Mine\}$
       Arcs Set $A = \forall a \in A, R_a^2 > Threshold_{R^2}$
Output: Lower and Upper Bounds of Circuit Lifetime
1  begin
2      $Vector\ Vtr$
3      for $\forall i \in Candidate$ do
4          for $\forall p \in V, p \neq i, \exists a \in A$, $a$ is the arc from $i$ to $p$ do
5              $LT_p$ = Estimate $p's$ lifetime based on binary search
6              if $LT_p < Smallest$ then
7                  $Smallest = LT_p$
8              end
9          end
10     end
11     Put $Smallest$ into $Vtr$
12 end
13 return Smallest value in $Vtr$, largest value in $Vtr$
```

Consider the example in Figure 4(a), 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.

Consider the other example in Figure 4(b), where the 80% and 20% DCCs are inserted at the inputs of buffer 4 and 5, respectively. Assume again that the critical path $p$ is in shortlist. If the DCC deployment will lead the path $p$ to fail post-maturely (i.e., path fail within $[n + \varepsilon, \infty]$), then the following clause

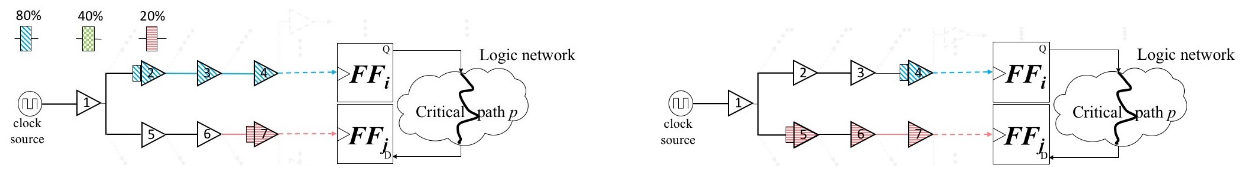
$$(A_1 \vee A_0 \vee B_1 \vee B_0 \vee C_1 \vee \neg C_0)$$

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.

## V. LIFETIME ESTIMATION

In this section, we propose two algorithms to estimate the lifetime of design circuits, which is attacked by our proposed framework using DCCs, considering the workload variations (i.e., various operational modes) from users.



(a) A DCC deployment lead path $p$ to fail prematurely (b) A DCC deployment lead path $p$ to fail post-maturely

**Figure** 4. Illustrative example for the proposed framework based on DCC deployment/insertion

```
Algorithm 2: Lifetime Estimation
  Input: Modified netlist (after DCC insertion), critical path $p$ and
         critical path $i$
  Output: Lifetime of path $p$
1  begin
2      Upper Bound $U$
3      Lower Bound $L$
4      Median $M$
5      while $U - L < 10^{-3}$ do
6          $M = \frac{U+L}{2}$
7          //Derive aging rate of path $i$ i.e., $X_i$. Path $i$ is assume
             undergo worst-case aging
8          $X_i = 0.0039 \times (0.5 \times 86400 \times 365 \times M)^{0.2}$
9          //Predict $p's$ aging rate i.e., $Y_p$ by the regression of $p$ on $i$
10         $Y_p = \alpha_{pi} \cdot X_i + \beta_{pi}$
11         $S_p$ = Estimate $p's$ slack considering an aging rate of $Y_p$
12         if $S_p > 0$ then
13             $L = M$
14         else
15             $U = M$
16         end
17     end
18     return $U$
19 end
```

It is worth reminding that the previous assumption, introduced in Section IV-B, says that 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 (Line 3 in Algorithm 1), all operational modes are considered, during the lifetime estimation in our methodology. Here, Line 2 in Algorithm 1, says that a candidate path $i$ is assumed to age in the worst case. Then, in the inner for-loop (Lines 4-8), we iteratively estimate the lifetime of other paths based on prediction from the worst-case aging of $i$. For each path, say $p$ (Line 4), the following steps are applied to estimate $p's$ lifetime: ($i$, Line 6) $p's$ lifetime can be estimated based on a binary search in Algorithm 2 ($ii$, Line 7) the estimated lifetime is compared against the smallest one, since the smallest lifetime among all paths determines the circuit lifetime. In Algorithm 2, path $p's$ lifetime is estimated based on a binary search. The upper/lower bounds for the binary search are set in Lines 1-2, respectively. In Line 6, M is set as the median value of two bounds; and in Line 10, $p's$ aging rate $Y_p$ is predicted by the regression equation of $p$ on $i$,

$$Y_p = \alpha_{pi} \cdot X_i + \beta_{pi}$$

where $\alpha_{pi}$ and $\beta_{pi}$ are coefficients, and path $i$ is assumed to age under M-year worst-case condition and its aging rate $X_i$ is derived by the following predictive model, which in presented in [14]:

$$A \cdot \alpha^n \cdot t^n \tag{3}$$

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

Then, in Line 11, $Y_p$ is used to estimate $p's$ slack $S_p$, which is utilized to check whether the setup-time violations will occur on p under the aging rate $Y_p$ (Line 12). If the value of $S_p$ is negative, it denotes that, setup-time violations will occur on $p$ after M years at an aging rate of $Y_p$ (Line 14). Afterwards, according to result of the above timing check, the upper/lower bound for next iteration will be set in Line 13 or 15. While repeating the above steps (Lines 5-17), both bounds gradually converge. Eventually, the converged value of the upper bound is considered as $p's$ lifetime (Line 17), which will be returned to Algorithm 1 as the value of $LT_p$ (Line 6 in Algorithm 1).

The aforementioned procedures are repeated for each candidate path. We can derive a lifetime value by considering a specific candidate path. By considering all of the candidate paths, all operational modes are considered and we can find a group of lifetime values, i.e., $V_{tr}$ in Algorithm 1. The smallest value and the largest one within $V_{tr}$ are the resulting lifetime interval found based on this given attack.

## VI. Monte-Carlo Instantiation of attacked designs

After the locations of DCC insertions are determined in the existing clock tree, Monte-Carlo simulation is performed to demonstrate the influence of process variations (PVs) on the proposed Trojan. The section is organized as follows: Section VI-A details the procedures of instantiation of attacked designs. Section VI-B details the lifetime estimation of Monte-Carlo instances of attacked designs, considering the correlation between PVs and aging. The correlation is discussed in Section VI-C. The experimental results will be shown in the next section.

### A. Monte-Carlo Instantiation of the Attacked Designs

Given an attacked design, it is instantiated considering PV by imposing extra $V_{th}$ offset (i.e., $\Delta V_{th}$ ) on each transistor. Note that, these offsets follow a normal distribution with the standard deviation of a given value, which usually ranges from

10mV to 30mV [15][16]. In other words, if the standard deviation is set to 20mV, it implies that 68% of $V_{th}$ offsets reside between $\pm$ 20mV. Up to present, a Monte-Carlo instance of an attacked design is built. Various instances of the attacked design are generated. 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 $V_{th}$.

### *B. Lifetime Estimation Considering the Effect of PVs on aging rates of transistors*

Whenever an instance is generated, Algorithm 1 and 2 are applied to estimate its lifetime interval. Note that, because PVs is considered, the aging rates of transistors along a path will no longer be equal; in other words, 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 2, aging rate of path $i$ ($X_i$) must consider the aging rate of individual transistor, instead of using the deterministic Equation (3). The aging rate of individual transistor can be obtained using the mathematical model introduced in the next subsection, which discusses the aging rate estimation of transistors, considering the correlation between PVs and BTI.

### *C. Aging rate estimation of transistors considering the correlation between PVs and BTI*

*1) Impact of PVs on BTI:* In this section, we discuss the influence of process variation (PV) on BTI. Other works [17][18] do consider the PV effect while performing Monte-Carlo simulations, but ignore the correlation [17] between PV 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.

*2) Model of the Correlation:* A model in [19] is proposed to estimate the correlation between fresh $V_{th}$ offset and BTI effects:

$$\Delta V_{th\_nbti} = (1 - S_v \cdot \Delta V_{th\_pv}) \cdot A \cdot a^n \cdot t^n \tag{4}$$

$$V_{th} = \Delta V_{th\_nbti} + \Delta V_{th\_pv} + V_{th\_design} \tag{5}$$

$\Delta V_{th\_pv}$ denotes the fresh $V_{th}$ due to PV. $\Delta V_{th\_nbti}$ denotes the BTI-induced $V_{th}$ shift and $V_{th\_design}$ denotes the nominal threshold voltage of the design. $S_v$ depends on $\Delta V_{th\_pv}$, and can be derived by following procedures:

*a) Assume the value of $V_{th}$ is convergent:* We assume threshold voltages of all transistors will be convergent after a long period, even if their fresh values are different. In other words, $V_{th}$ is fixed regardless of various $\Delta V_{th\_pv}$, since the aforementioned correlation takes effect.

*b) Obtain the convergent value of $V_{th}$:* Since $V_{th}$ is fixed regardless of various $\Delta V_{th\_pv}$, we set $\Delta V_{th\_pv}$ to 0 in Equation (5) to derive the convergent value of $V_{th}$. This way, Equation (5) can be simplified as Equation (7), where the convergent value of $V_{th}$ equals $\Delta V_{th\_nbti}$ plus $V_{th\_design}$. Here, $V_{th\_design}$ is given by technology and $\Delta V_{th\_nbti}$ can be simplified as Equation (6) because $V_{th\_pv}$ is set to 0. In Equation (7), since $V_{th\_design}$ is known and $\Delta V_{th\_nbti}$ can be derived without unknown $S_v$, the convergent value of $V_{th}$ can be obtained.

$$\Delta V_{th\_nbti} = A \cdot a^n \cdot t^n \tag{6}$$

$$V_{th} = \Delta V_{th\_nbti} + 0 + V_{th\_design} \tag{7}$$

*c) Obtain the value of $S_v$ with specific $\Delta V_{th\_pv}$:* Given a specific value of $\Delta V_{th\_pv}$, our objective is to derive corresponding $S_v$ value. Since the convergent $V_{th}$ value is obtained in the last step and $V_{th\_design}$ is known, we can derive the corresponding $\Delta V_{th\_nbti}$ using Equation (5), such that the corresponding $S_v$ can be obtained in Equation (4).

So far, the conversion from a given specific $\Delta V_{th\_pv}$ to corresponding $\Delta V_{th\_nbti}$ has been constructed. Then, $\Delta V_{th\_nbti}$ must be transformed to aging-induced delay shift. In [14], the delay shift is linearly proportional to $\Delta V_{th\_nbti}$:

$$\Delta t_{p\_aged} = C \cdot \Delta V_{th\_nbti} \tag{8}$$

where $\Delta t_{p\_aged}$ is BTI-induced delay shift, and $C$ is a constant and fitted to 0.5 after SPICE simulation. Further, the Equation (8) is modified as following Equation (9) to account for the conversion from $\Delta V_{th\_pv}$ to intrinsic delay shift.

$$\Delta t_{p\_intrinsic} = C \cdot \Delta V_{th\_pv} \tag{9}$$

where $\Delta t_{p\_intrinsic}$ is the delay shift caused by $\Delta V_{th\_pv}$. Up to now, a model is built to convert a given specific $\Delta V_{th\_pv}$ to corresponding $\Delta t_{p\_aged}$ and $\Delta t_{p\_intrinsic}$. The model is involved in Section[] and Section[].

## VII. EXPERIMENTAL SETTING AND RESULTS

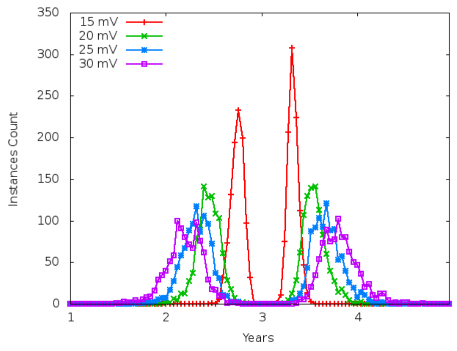
In this section, we explain the experimental setting and demonstrate the experimental results of our proposed Trojan attack. The benchmarks in IWLS'05 and ISCAS'89 are used in the experiments. The utilized technology is TSMC 65nm GP standard cell series. The used SAT solver is MiniSAT 2.2. The section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

### *A. Clock Period Setting*

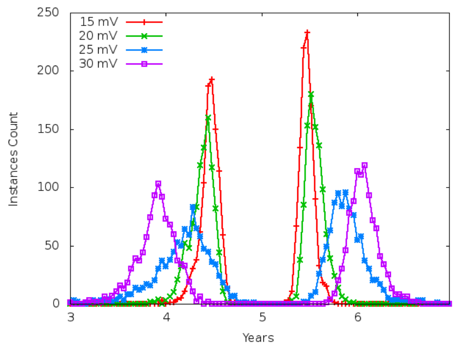
Figure 5 shows the lifetime intervals of original (i.e., Trojan-free) designs with clock periods which make the designs fail at a specified time (in our experiment, 7 years) under aging. The resulting clock period is both used in Trojan-free and Trojan-included (attacked) designs. In Figure 5, lower 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 7th year under the worst-case aging

| 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 |

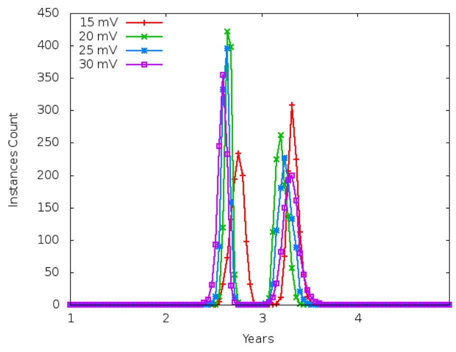
**Figure** 5. Circuit information and estimated lifetime without Trojan insertion



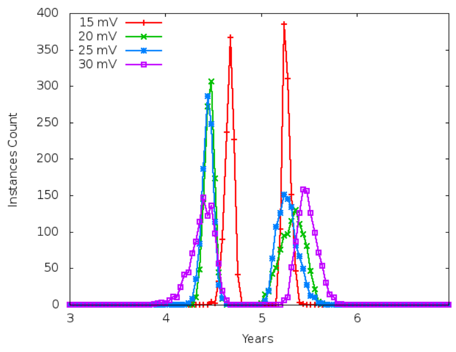
(a) *s38417* is attacked with $n$ = 3 yr and $\varepsilon = 0.3$ yr



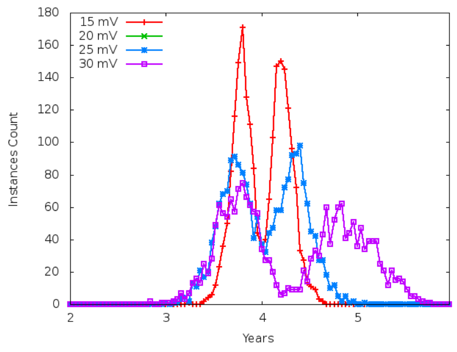
(b) *s38417* is attacked with $n$ = 5 yr and $\varepsilon = 0.5$ yr



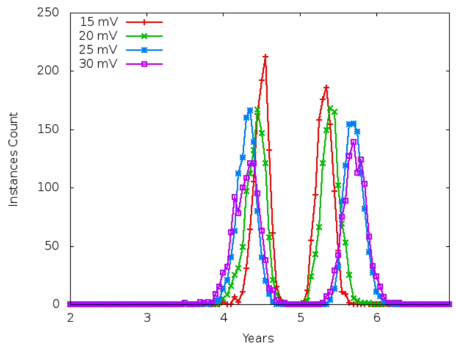
(c) *des_perf* is attacked with $n$ = 3 yr and $\varepsilon = 0.3$ yr



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



(e) *leo3mp* is attacked with $n$ = 3 yr and $\varepsilon = 0.3$ yr



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

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

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. 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 3rd year, whereas there exists SAT solution while it is attacked to fail at $4^{th}$ year. In each left/right subfigure, there exist four distributions. The distributions of various colors differ in the standard deviations of $V_{th}$ while generating instances. That is, each color corresponds to one distinct value of standard deviation of $V_{th}$. In our experiments, the deviations are set to 15mV, 20mV, 25 mV and 30 mV, 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 the 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 process variations, 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; that is, 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. 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 V, 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.

### C. Detectability

When it comes to the detectability of the proposed Trojan, side-channel analysis is 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 PV. Therefore, the proposed Trojan framework 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 over- head of extra pins is costly.

## VIII. 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] 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.

[7] 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.

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

[9] 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.

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

[11] 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.

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

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

[14] 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.

[15] 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.

[16] 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.

[17] S. Kiamehr *et al.*, "The impact of process variation and stochastic aging in nanoscale vlsi," in *Reliability Physics Symposium (IRPS), 2016 IEEE International*, IEEE, 2016, CR–1.

[18] J. Chen and M. Tehranipoor, "A novel flow for reducing clock skew considering nbti effect and process variations," in *Quality Electronic Design (ISQED), 2013 14th International Symposium on*, IEEE, 2013, pp. 327–334.

[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.