

HTPG: Hardware Trojan Detection based on Anomalies of Transient Behavior

Quentin TUAL
Lab-STICC
Université de Bretagne Sud
Lorient, France
quentin.tual@univ-ubs.fr

Jean-Christophe LE LANN
Lab-STICC
ENSTA
Brest, France
jean-christophe.le_lann@ensta.fr

Philippe COUSSY
Lab-STICC
Université de Bretagne Sud
Lorient, France
philippe.coussy@univ-ubs.fr

Abstract—Hardware Trojan (HT) attacks have been a recognized threat to Integrated Circuits (ICs) for more than two decades. HTs tamper with IC integrity by enabling surreptitious modifications to cause a malfunction and/or data leakage. They can be implanted during all stages of the design and manufacturing of ICs. In Cyber-Physical Systems (CPSs), where hardware directly controls physical processes, such attacks can compromise computational integrity as well as system safety, reliability, and security. In this paper, we present a method named HTPG for detecting HTs in combinational logic by analyzing the transient behavior of circuits. Our method analyzes a reference design model to generate specific test patterns that reveal anomalies of transient behavior when the chip is altered by an HT. These specific test patterns are then applied during stimulation of a design under verification while observing its transient behavior. Differences with the reference transient behavior reveal logical design modifications, such as HT insertion, which enable automated detection. Considering various HTs inserted in synthetic or real application benchmarks, HTPG reports an average detection rate of 99.99%. The experiments also show that the test patterns generated by HTPG considering a given delay model are still effective in detecting 94% HTs with a more precise one.

I. INTRODUCTION

The last few decades have seen a wide evolution of IC architectures, associated technologies, and supply chain. Today, their complexities are greater than ever, which requires unprecedented efforts to ensure the design of trusted and secure ICs. As core components of CPSs, these circuits directly affect the reliability and safety of the physical processes they monitor and control. Threats are generally divided into two major categories : Intellectual Property (IP) piracy and malicious design tampering. HTs, which belong to the latter, are stealthy alterations that can be made at all stages, from initial design phases to the fabrication steps of ICs [1] to cause data leakage or induce malfunctions.

There are many types of HT, each of them being categorized according to its trigger conditions, impact on the circuit, or architecture [2][3]. HT attacks can occur in various scenarios characterized by the entity (e.g., rogue designer, compromised design tool, etc.) that initiates the attack, the design step at which the attack occurs, and the architecture of the Trojan. Detecting HTs insertions is crucial to ensure the integrity of circuits throughout their entire design process. Unfortunately, their stealthy nature makes their detection almost impossible

with conventional testing methods such as functional tests [3]. Therefore, specialized detection methods are required.

In order to detect HTs in combinational layers, we propose an approach leveraging the Transient Behavior of signal values induced by their propagation in logic gates. Detection is performed thanks to dynamic analysis of a design model and its stimulation using associated specific test patterns. Our contributions are the following:

- We formally define Anomalies of Transient Behavior (ATBs) and show their interest for HT detection;
- We present a method to generate Test Vector Pairs (TVPs), revealing ATBs from an altered combinational circuit, and observe these ATBs in its transient behavior;
- We present a set of experiments which results demonstrate the ability of the proposed HTPG approach to detect HTs;

The remainder of this paper is organized as follows. Section II discusses the background and related works. Section III presents a pedagogical example to highlight ATBs and their potential in HTs detection. Section IV proposes a formal definition of ATBs and details the proposed method for HT detection. Section V presents the experimental evaluation. Section VI provides a summary and discusses future work.

II. BACKGROUND AND RELATED WORKS

Detecting HT classically requires a golden reference: an authentic chip, a trusted model of the circuit, or a set of intrinsic characteristics of the circuit. To cover the wide attack surface offered by current complex design flows, several complementary detection methods must be combined. Each approach aims to detect HTs for a specific threat model. Some of them, usually falling into the domain of formal verification [4], checks the functional correctness of the circuits. These methods allow HT detection on models of the circuits (RTL, gate-level...) and the generation of test vectors for detection on manufactured chips. Other approaches rely on the circuit's physical parameters (e.g. power consumption, electromagnetic radiation). These solutions are generally categorized as side channel analysis for HT detection [5][6]. There also exist approaches that use both formal test generation (usually called SAT-based) and side-channel analysis as in [7][8]. Finally, some approaches address HT detection using destructive methods, e.g., layer-by-layer IC imaging [9]. *Reference-free* detection

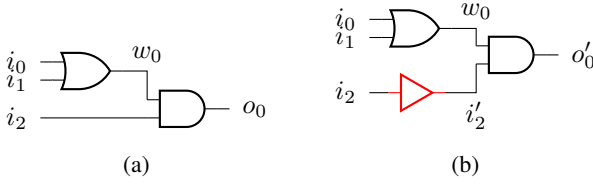


Fig. 1: Schematics of C (a) and C' (b)

methods are also available in the literature [10][11]. These methods use Machine Learning to find recognizable patterns in HT attacks, requiring large datasets and being less interpretable.

Our works focus on HT detection based on transient behavior analysis which use a golden reference. The transient behavior of a combinational block refers to transitions (i.e. intermediate values) resulting from a sequence of values applied to its inputs. It is the semantic opposite of "stabilized state". Transient behavior analysis uses the physical characteristics of the circuit, intrinsic properties serving as a unique signature of the circuit's structure, in order to detect potential HTs. To date, there have been few contributions in the literature that have explored the potential of this property for HT detection. Two families of methods exploiting transient behavior analysis exist: at-speed path delay measurement and overclocking. *At-speed path delay measurement* methods [12][13][14] consist in measuring the delay exhibited by the multiple paths of the circuit during operation. The purpose of the path delay measurement is to detect any violations of the timing references that would be induced by HT insertion. Circuit's timing characteristics are usually known from simulation or Static Timing Analysis (STA). *Overclocking* [15][16][17] consists of driving the flip-flops with a clock signal whose period is shorter than the delay of the circuit's critical path (i.e. the clock frequency exceeds the nominal one). The overclocked sequential circuit exhibits a dysfunctional behavior used as a signature.

In this paper, we propose a HT detection method that exploits the variation of the values in the transient behavior rather than the variation in path delays induced by the insertion of HTs. To this end, the architecture of a reference design model is analyzed, and TVPs are generated. These TVPs are used to stimulate the reference design and a chip under test. Resulting transient behaviors are captured and their comparison allow to detect any architecture alteration. Unlike digital modeling, late-stage insertion detection cannot use straightforward equivalence checking, so dynamic analysis remains the only non-destructive method to verify IC integrity. The proposed detection method can be applied to each chip manufactured individually using the same reference. This work paves the way for a simple, effective, non-destructive dynamic analysis method for detecting HTs in ICs, especially those inserted in the final design phase or during manufacturing.

III. PEDAGOGICAL EXAMPLE

To illustrate what ATBs are, we first present a simple example. Let C be a combinational circuit used as a reference (see Fig. 1a), and C' a version of C altered by an inserted HT

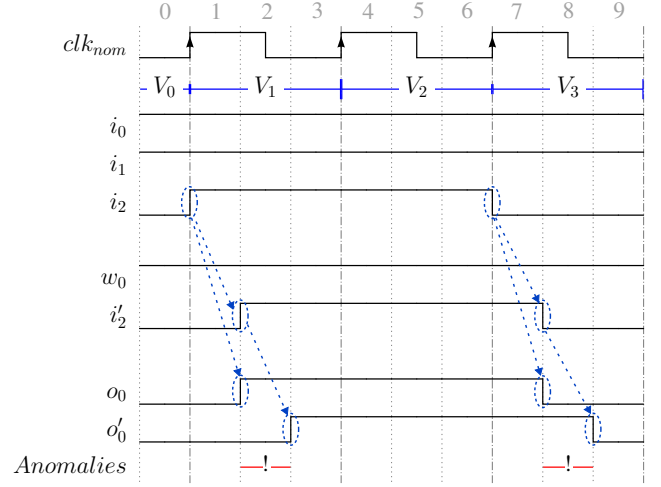


Fig. 2: Timing diagram for simulation of C and C' . Instants are indicated on the top of the diagram and arrows indicate transition dependencies

(see Fig. 1b). Assuming $delay(g)$ is the propagation delay of a gate g , in this pedagogical example we will use an uniform unit delay model with which $\forall g \in C' \cup C, delay(g) = 1$ in Arbitrary Units (AU). With i_0, i_1 and i_2 the inputs and o_0 the output of C , the Boolean function of the circuit C is defined as follows:

$$C : o_0 = (i_0 + i_1) \cdot i_2 \quad (1)$$

The altered circuit C' is the same circuit wherein a buffer is inserted over the input signal i_2 (see Fig. 1b), representing a simplified HT. This representation assumes perfect stealth with a payload that always remains inactive. The signals of C that are affected by the alteration in C' are renamed such that i_2 becomes i'_2 and o_0 becomes o'_0 . A difference is observable by temporalizing the previously defined equation (1). For a given instant $t \in \mathbb{N}$, let define :

$$C : o_0(t) = (i_0(t-2) + i_1(t-2)) \cdot i_2(t-1) \quad (2)$$

$$C' : o'_0(t) = (i_0(t-2) + i_1(t-2)) \cdot \underline{i_2(t-2)} \quad (3)$$

The equations (2) and (3) represents the transient behavior of the outputs of C and C' respectively. The difference between the two equations is underlined in equation (3): it is the mathematical representation of the structural difference visible in red in Fig. 1b. Consider that both circuits are simulated, ensuring that all signals stabilize before the input values change. Therefore, the input values are updated at each rising edge of clk_{nom} , a square wave signal operating at nominal frequency $f_{nom} = 1/(\gamma + 1)$, with γ the critical path and a margin of 1 AU. In the simulation timing diagram shown in Fig. 2, no difference can be observed on the output at each rising edge of the clock (nominal frequency). However, two differences can be seen between the transient behavior of C and C' . These dissimilarities are due to the structural difference already identified earlier in equations (2) and (3). We refer to these dissimilarities, which indicate the presence of structural

modification, as ATBs.

In order to observe ATBs, it is necessary to sample values earlier than the instants of clk_{nom} rising edges, i.e., shifting the phase or increasing the sampling frequency. To avoid sampling of non-stable values, a reliable synchronizer would be needed in hardware implementations [18]. In addition, to provoke ATBs, specific TVPs must be generated. For that purpose, the design to secure must be analyzed. The next section models formally ATBs and then presents the automated HTPG approach we propose in this paper.

IV. METHODOLOGY

This section presents HTPG, a method for detecting HT using ATBs. Initially, formal definitions for hardware architectures and ATBs are provided. Then, an automated method to generate TVPs designed to reveal ATBs is presented. Next, a method to capture transient behavior is proposed. Finally, the comparison of two captured transient behavior to detect ATBs is detailed.

A. Anomalies of Transient Behavior

Let I and O represent a set of all the inputs and outputs of C respectively with $I = \{i_0, \dots, i_{k-1}\}$ and $O = \{o_0, \dots, o_{l-1}\}$ where k is the number of inputs and l the number of outputs of C . Let $OC(s) \subset O$ (resp. $IC(s) \subset I$) denotes the *output cone* (resp. *input cone*) of s , represented as a set of outputs downstream (resp. inputs upstream) of the signal s . For a circuit C , let $path(i_x, o_y) = \{g_0, g_1, \dots, g_{n-1}\}$, with $x \in [0, k-1]$ and $y \in [0, l-1]$, the path from signal i_x to an output signal o_y represented as a sequence of n gates. The set of all paths in C is noted $\phi = \{path(i_x, o_y) : \forall x \in [0, k-1], \forall y \in [0, l-1]\}$. Therefore, the delay of a path p is $delay(p) = \sum_{g \in p} delay(g)$. Consequently, $\delta = \{delay(p) : \forall p \in \phi\}$ is a set that contains the delay of all the paths of C and $\gamma = \max(\delta)$ is the critical path delay (in Fig. 1, $\gamma = 2$ AU). We define an ATB as an instant t_α where $o_y(t_\alpha) \neq o'_y(t_\alpha)$ and $t < t_\alpha < t + \gamma$ with t the instant of a synchronous update of all inputs (typically synchronized on a clock rising edge in sequential circuit).

We use a Boolean delay equation (BDE) model for a signal s , denoted $s(t)$, indicating the value of s at any time t . The model considers $s(t)$ based on all inputs $i \in IC(s)$, factoring in gate propagation delays and data dependencies. Each operand is an input i_x , defined by t minus the total delay of gates between i_x and s , namely $delay(path(i_x, s))$.

In our example (Section III), equations (2) and (3) are the BDEs of signals $o_0(t)$ and $o'_0(t)$. The signal i_2 , attacked in C' , is delayed by an added logic gate, making $i'_2(t) = i_2(t - 1)$. This difference is noticeable when transitions on o_0 originate from i_2 . To provoke an ATB on an output, it is essential to: (a) apply a transition on a signal appearing in the differing terms (i_2 in our example), and (b) ensure this signal controls an output, maintaining identical terms as *non-controlling values* as defined in [19]. This establishes the system:

$$o_0(t_\alpha) \neq o'_0(t_\alpha) \Leftrightarrow \begin{cases} i_2(t_\alpha - 1) \neq i_2(t_\alpha - 2) & (4a) \\ i_0(t_\alpha - 2) + i_1(t_\alpha - 2) = 1 & (4b) \end{cases}$$

Equations (4a) and (4b) model the conditions (a) and (b), respectively. Regardless of its value, it is the presence of

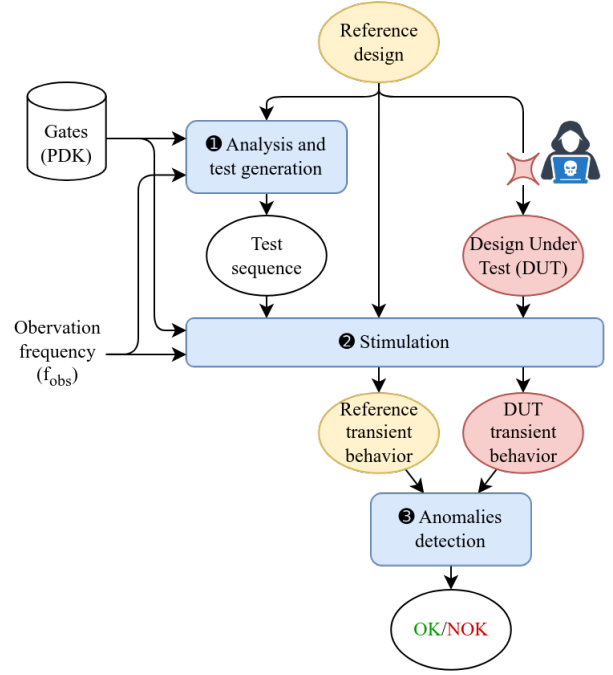


Fig. 3: Proposed HTPG flow

an additional delay w.r.t. the original design that makes the structural alteration visible during the transient behavior.

This system of equations is essential : it can be automatically constructed and solved, enabling an automated approach to HT detection. According to equations (4a) and (4b), provoking one ATB requires the stimulation of the input cone $IC(o_y)$ of a given output with a specific sequence of input values referred as TVP in this paper. Hence, one TVP is associated to one given ATB and to at least one specific signal (later referred as *risky signals*). Applying the TVP provokes the associated ATB when the design architecture has been altered on an associated signal. Obviously, there are several solutions to the system equation. However, only one solution is needed to reveal an ATB and thus detect a structural alteration.

In Fig. 2 the TVP1, referring to $(\vec{V}_0, \vec{V}_1) = (110, 111)$ with $\vec{V}_z = (i_0, i_1, i_2)$, $z \in \mathbb{N}$ being an arbitrary identifier, is used to stimulate C and C' . $(\vec{V}_0, \vec{V}_1)$ verifies (4a) and (4b), as a consequence an anomaly is generated. Indeed, with respect to equation (4b), this TVP sets the signal w_0 to 1 (i.e. the *non-controlling value* of the downstream AND gate) at all instants of the stimulation. In this way, the transition introduced at the input i_2 (at instant 1), with respect to equation (4a), is propagated to the outputs o_0 and o'_0 , respectively, at instants 2 and 3. Hence, the delay difference between i_2 and i'_2 caused by structural tampering is visible at the outputs (see the marked anomaly in Fig. 2). For illustration purposes, Fig. 2 also presents TVP2, referring to $(\vec{V}_2, \vec{V}_3) = (111, 110)$, which provokes a falling edge (instead of the rising one previously presented). The proposed formalization of this phenomenon enables the development of an automated method, presented in the next subsection, to reveal and capture ATBs.

Algorithm 1 Analysis and test generation for a circuit C

```

1: function GEN_TVP( $C$ : a circuit model,  $gates\_lib$  a li-
   brary of the gates)
2:    $test\_set$  : a set of test vectors pairs
3:    $risky\_signals \leftarrow get\_risky\_signals(C, gates\_lib)$ 
4:   for all  $s \in risky\_signals$  do
5:      $C' \leftarrow dummy\_alteration(C, s, gates\_lib)$ 
6:     for all  $o \in get\_outputs\_cone(s, C)$  do
7:        $(o(t), o'(t)) \leftarrow get\_BDEs(o, C, C')$ 
8:        $constraints \leftarrow get\_constraints(o(t), o'(t))$ 
9:        $(\vec{V}_0, \vec{V}_1) \leftarrow sat\_solve(constraints)$ 
10:      if  $(\vec{V}_0, \vec{V}_1) \neq \emptyset$  then
11:         $test\_set.push((\vec{V}_0, \vec{V}_1))$ 
12:        break
13:      end if
14:    end for
15:  end for
16:  return  $test\_set$ 
17: end function

```

B. HTPG method

The method we propose exploits ATBs for HT detection. It is based on three main steps, as illustrated in Fig. 3. The first step *Analysis and test generation*, uses as inputs a gate-level netlist and a timing information from a library (typically a Process Design Kit) and generates a test sequence containing one TVP for each identified risky signal. The second step *Stimulation* consists in producing and observing transient behaviors by applying the generated TVPs to a reference design and a Design Under Test (DUT). The last step *Comparison*, compares the two transient behaviors observed to reveal the ATBs provoked.

1) *Analysis and Test Generation*: The test generation is described in Algorithm 1. It consists in computing one TVP for each signal that can be exploited by an attacker (later referred to as *risky signals*). Each TVP is generated by resolving a system of constraints deduced during the analysis of the reference (golden) model. The first step *Analysis and Test Generation* is thus broken down into four sub-steps.

- *Risky signals identification (line 3 in Algorithm 1):*

The analysis starts by detecting risky signals in the reference circuit. In this paper, a risky signal is typically a signal that has a slack allowing the insertion of any logic gate from a given gate library without altering critical path timing. Risky signals could also be mentioned by the user through pragmas, for example.

In the pedagogical example, there is only one risky signal in C which is the input signal i_2 . The process then iterates on the next steps for each risky signal identified.

- *Dummy alteration (line 5 in Algorithm 1):*

C' , a modified version of C , is created by adding a buffer to the signal s , as illustrated, for example, in Fig. 1. This buffer represents an always-off payload of a hardware Trojan (HT) and adds a delay on s . As mentioned earlier, regardless of its value, it is the added delay relative to the original circuit that reveals the structural change during transient behavior. Hence, the delay is arbitrarily set as the minimum gate library delay. The process then iterates in the next steps for each output o in $OC(s)$ until a TVP is generated for at least one output.

- *BDEs extraction (line 7 in Algorithm 1):*

Two BDEs noted $o(t)$ and $o'(t)$ are extracted from C and C' respectively, as explained in the previous sub-section. In the pedagogical example, the risky signal i_2 leads to $OC(i_2) = \{o_0\}$ and the corresponding output in C' is o'_0 . As already mentioned, the two BDEs extracted in this example are the equations (2) and (3).

- *TVP generation (line 9 in Algorithm 1):*

A system of constraints is built using the two BDEs extracted previously. The constraints are listed below.

- All the input values are updated synchronously, at the instant 0 and only one update is authorized; thus two consecutive test vectors are required : $(\vec{V}_0, \vec{V}_1)$.
- The application of the TVP must allow observing an ATB at the instant t_α .

Formally, these constraints are expressed as follows.

$$\exists(\vec{V}_0, \vec{V}_1), t_\alpha \left\{ \begin{array}{l} \forall x \in [0, k-1] \\ \text{and } t \in \mathbb{Z}, i_x(t) = \begin{cases} \vec{V}_{0x}, \forall t < 0 \\ \vec{V}_{1x}, \forall t \geq 0 \end{cases} \end{array} \right. \quad (5)$$

such that $o(t_\alpha) \neq o'(t_\alpha)$ with $t_\alpha \in]0; \gamma]$

Solving the constraint system gives a solution which is a TVP $(\vec{V}_0, \vec{V}_1)$ and the instant t_α at which an anomaly is observable. Then the process loops in Step 2 with another risky signal. However, if no solution exists, then the process loops in Step 3 with another output included in the fanout cone of s . Finally, if all outputs have been processed and no solution exists, the insertion point is uncovered. Once identified, these uncovered signals can easily be reported to the designer or a logic synthesis tool so that the architecture is modified for better coverage. In the pedagogical example, one of a TVP is identified as TVP1 referring to $(\vec{V}_0, \vec{V}_1) = (110, 111)$.

The complexity of the algorithm depends on: the number of risky signals (line 4 in Algorithm 1), the number of outputs in the output cone $|OC(s)|$ of each risky signal s (line 6, in Algorithm 1) and the complexity of each constraint system, i.e., the number of inputs in both BDEs, $|IC(OC(s))|$. The total complexity can be expressed as follows.

$$\sum_{\forall s \in risky_signals} |OC(s)| \times |IC(OC(s))|$$

Once all insertion points have been processed, a set of TVPs is obtained. These TVPs are then used as a test sequence to stimulate the reference design along with the design under test and to capture their respective transient behaviors. The stimulation approach we propose is presented below.

2) *Stimulation*: The transient behavior of a given design is captured during the stimulation step. In contrast to a simulation-only method, considering IC verification imposes a finite observation frequency. As stated earlier, the nominal clock that operates at frequency $f_{nom} = 1/T_{nom}$ does not allow capturing transient behavior. Indeed, the output values must be sampled before they are stable i.e. at a time t_α with $t_\alpha \in]0; \gamma]$, where γ is the delay of the critical path. For that purpose, values of the transient behavior are captured at regular time intervals through

Set	#circuits	#inputs	#outputs	#gates_CP
Synthetic	800	8-12	4-10	5-10
Real	134	3-357	3-73	1-108

TABLE I: Characteristics of each set (and subsets) of circuits

the cycle, corresponding to a resolution r and expressed as a normalized value over T_{nom} . Any anomaly that lasts less than $r \times T_{nom}$ time units is likely to go undetected. As the value of r decreases, more instants are captured, thus the lower the resolution, the higher the probability of capturing existing ATBs. By definition of inertial delay, any glitch produced by a gate g has a duration Δ that must satisfy $\Delta \geq \text{delay}(g)$. High clock frequencies required to sample such short anomalies might not be reachable due to technological limits. To observe transient behavior, we use a shadow-register-based alternative following [12], and use synchronizers to prevent metastability as advised in [18]. The proposed architecture leverages the clock generator's precision and stability by applying each TVP across multiple cycles and acquiring a single sample in each cycle, while shifting the sampling instant earlier by r in each successive cycle to capture the transient behavior. In a sequential circuit, for a combinational block with $|O|$ outputs feeding the original registers, the method requires to add $|O|$ shadow registers and a separate clock tree. It should be noted that a TVP is made up of two test vectors $\vec{V}_0$ and $\vec{V}_1$, each applied on two consecutive cycles. However, the ATBs revealed by a TVP always appear in the second cycle, i.e., after the transition from $\vec{V}_0$ to $\vec{V}_1$. Thus, it is only necessary to capture the transient behavior after this transition. In Fig. 2, $t_\alpha = 2$ for the first anomaly and $t_\alpha = 8$ for the second. Capturing these two ATBs with the proposed capture method requires $r \leq 0.5$.

3) *Comparison*: ATBs are detected by bitwise comparing the captured behaviors of the reference circuit and the circuit under test, revealing behavioral variations and thus structural changes from HT insertion.

V. EXPERIMENTAL EVALUATION

In this section, the HTPG method we propose is evaluated through various experiments considering different parameters: circuit nature, circuit complexity, HT model, gate delay model... The first subsection introduces the experimental setup. The second subsection studies the capacity of HTPG to produce TVPs. Following this, the third subsection evaluates the capability of the generated TVPs to reveal ATBs. The fourth subsection analyzes how well the HTPG method can detect ATBs. The concluding subsection evaluates the effectiveness of HTPG across delay models.

A. Experimental setup

The proposed HTPG method has been automated and implemented in Ruby. The systems of constraints previously detailed in section IV are written in SMT-LIB format and solved with *Z3 theorem prover* [20]. The simulations are performed by using GHDL open source simulator [21] running VHDL circuit descriptions. The experiments are conducted on a machine with an Intel Xeon CPU W-2125 @ 4GHz and 32GB RAM.

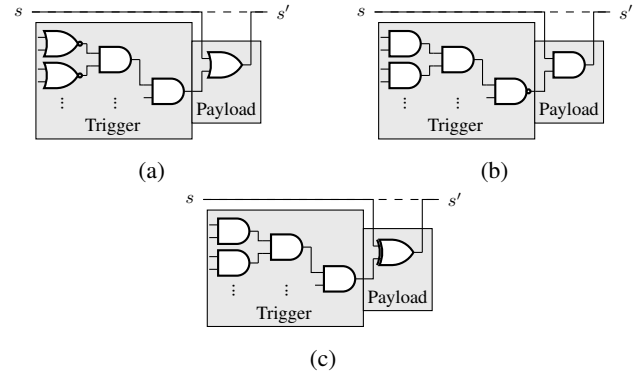


Fig. 4: HT templates: HT1 (a), HT2 (b) and HT3 (c)

Circuit set	Synthetic	Real
Minimum	0.76s	0.3ms
Maximum	3.05s	27min
Average	1.70s	1min38s

TABLE II: (Experiment 1) Time to generate TVPs

A variety of circuits, HTs and delay models are used in the experiments to evaluate the detection rates of the proposed method. The experiments focus on combinational circuits composed of classical logic gates (NAND, NOT, OR, XOR...).

The simplistic model used for pedagogical purposes is replaced in our experiments by two more precise ones:

- An *integer* delay model : a unique inertial propagation delay value is assigned to each gate type.
- A *rise/fall* delay model : the rising transitions keep the same propagation delays as in the *integer* delay model while the falling transitions have smaller ones.

The experiments cover two distinct sets of circuits which are detailed in the following list:

- The first set consists of 800 synthetic circuits, randomly generated by varying parameters as listed in Table I.
- The second set gathers 134 real circuits taken from the LGSynth91 benchmark [22].

In the experiments conducted, three explicit payload HTs are considered, each with different payload and trigger (see Fig. 4a, 4b and 4c).

B. Experiment 1 : Study of HTPG ability to generate TVPs

The objective of this experiment is to verify whether the proposed method is effective in provoking ATBs. For each circuit C , all the risky signals are identified. For each identified risky signal, a dummy alteration is performed, creating an altered version C' of C . For each pair of circuit (C, C') , BDEs are extracted as proposed in Section IV.B to build a system of constraints which resolution provides one TVP. As stated in Section V.A, each BDE and constraint is encoded in SMT-LIB format, allowing the usage of an SMT solver like Z3 [20] to resolve the system of constraints.

The Table II shows the minimum, maximum, and average time it takes to generate the TVPs for the circuits of each set. At most, TVPs generation takes 27 minutes for the largest real

Circuits set	Synthetic	Real
Minimum	43.66%	99.69%
Maximum	100.00%	100.00%
Average	95.55%	99.99%

TABLE III: (Experiment 1) Coverage rate of risky signals

Circuits set	Synthetic			Real		
HT	HT1	HT2	HT3	HT1	HT2	HT3
$f_{obs} = \infty$	100%	100%	100%	100%	100%	100%
$r = 0.2$	98.9%	98.0%	97.9%	94.0%	92.5%	95.5%
$r = 0.1$	99.2%	99.6%	99.1%	98.5%	98.5%	100%
$r = 0.05$	99.5%	99.6%	99.5%	99.3%	100%	-
$r = 0.025$	99.6%	99.8%	99.7%	100%	-	-
$r = 0.02$	100%	100%	100%	-	-	-

TABLE IV: (Experiments 2&3) Rates of detected insertion cases (at least one ATB observed) for various r values.

application benchmark which contains 7006 risky signals. For *Real* application benchmarks, the length of the generated test sequences admits a minimum of 20 test vectors, an average of 2 675, and a maximum of 23 208 test vectors.

Results from Table III show that the proposed *HTPG* method is able to generate a TVP for 95.55% of the risky signals in average when considering synthetic benchmarks, reaching 99.99% when considering *Real* benchmarks. This is explained by the fact that non-synthetic benchmarks are more elaborated and thus have fewer *non-sensitizable* signals.

C. Experiment 2 : Study of TVPs' ability to provoke ATBs

To confirm that each generated TVP provokes one ATB, additional experiments have been carried out. For each HT model listed in Section V.A, an altered version of each circuit is created by inserting the HT on a randomly selected risky signal. The trigger conditions are selected in such a way that *only one* test vector allows the HT to alter the output values of the initial circuit in stable state. Using the generated TVPs, both original and altered circuits were tested through simulation. The transient behavior is observed with an infinite observation frequency (i.e. $f_{obs} = \infty$).

The first line ($f_{obs} = \infty$) from the Table IV shows that all the 934 altered circuits exhibit an ATB during stimulation. Therefore, each generated TVP revealed an ATB when the circuit was altered on the corresponding risky signal. On the basis of the risky signals automatically identified, generated TVPs are thus effective in revealing HT insertion (shown in Table IV) with a success rate of 100% for both synthesis and real application benchmarks. However, using $f_{obs} = \infty$ is equivalent to $r = 0$, which may suit a purely simulation-based approach, but is inappropriate when observing ICs.

D. Experiment 3: Study of HTPG ability to capture ATBs

This third experiment studies how the finite observation frequency (i.e. $r > 0$) affects the ability to capture revealed ATBs. The setup is kept unchanged, except for the resolution that varies from $r = 0.2$ to $r = 0.02$.

Based on the results in Table IV, this experiment shows that 98.5% of ATBs are detected with $r = 0.1$, i.e., at one

Circuits set & HT	Real & HT1		
Delay model	A	B	Relevance A $\rightarrow$ B
$f_{obs} = \infty$	100%	94.0%	94.0%
$r = 0.2$	94.0%	77.6%	82.5%
$r = 0.1$	98.5%	87.3%	88.6%
$r = 0.05$	99.3%	91.0%	91.6%
$r = 0.025$	100%	94.0%	94%

TABLE V: (Experiments 4) Rates of detected insertion cases in real benchmarks with delay models A and B for various r . The last column shows the relevance rates from A to B.

tenth of the nominal period T_{nom} . The 100 % detection rate is achieved with $r = 0.02$. Given that (Experiment 1) TVPs are generated for 99.99% of the risky signals, (Experiment 2) that 100% of the generated TVPs reveal ATBs and (Experiment 3) that 100% of the ATBs appearing can be captured with the proposed observation technique, the detection rate of the HTPG method is thus 99.99% for these experiments.

E. Experiment 4: Efficiency of HTPG accross delay models

The objective of this last experiment is to evaluate the ability of TVPs, generated by using a delay model A, to provoke ATBs when considering a more precise delay model B during stimulation. Model A is the integer delay model used in the previous experiments. Model B is a *rise/fall* delay model (see section V.A). This experiment is performed with *Real* benchmarks and HT1 only, since the previous experiments have shown similar results in the various considered configurations.

The results obtained for various resolutions are shown in Table V which first column is the same as the column 'Real, HT1' from Table V for comparison purpose. Detection rate with Model B mildly declines, as changing propagation delays can alter transitions in downstream signals, affecting outputs. Thus, some ATBs present with model A are not with model B. However, the finer resolution still hits 94% of detection after the delay model changed (as stated with $f_{obs} = \infty$). In other words, 94% of the TVPs are still relevant from our model A to our model B. This implies that TVPs can be calculated using a basic delay model, applied to a more detailed model, while retaining much of their effectiveness.

VI. CONCLUSION & DISCUSSION

This paper has introduced HTPG, a method using ATBs to identify HT insertions. Experiments confirm that transient behavior holds useful information for HT detection. The proposed approach is capable of generating a TVP for each identified risky signal. The implementation bypasses technological limits on high clock frequencies by sampling transients across multiple cycles. Overall, HTPG achieves a detection rate of 99.9% based on ATBs captured during TVPs stimulation.

Future works aim to explore more accurate delay models, localized or scalable use for sequential architectures, and actual ICs for experiments. Furthermore, optimization efforts will focus on reducing TVP generation and design stimulation time by optimizing the formulation passed to the SMT-solver, using scan timing frames and merging independent TVPs into a single one for paths containing multiple risky signals.

REFERENCES

- [1] B. Shakya, T. He, H. Salmani, D. Forte, S. Bhunia, and M. Tehranipoor, "Benchmarking of Hardware Trojans and Maliciously Affected Circuits," *Journal of Hardware and Systems Security*, vol. 1, no. 1, pp. 85–102, Mar. 2017.
- [2] M. Tehranipoor and F. Koushanfar, "A Survey of Hardware Trojan Taxonomy and Detection," *IEEE Design & Test of Computers*, vol. 27, no. 1, pp. 10–25, Jan. 2010.
- [3] K. Xiao, D. Forte, Y. Jin, R. Karri, S. Bhunia, and M. Tehranipoor, "Hardware Trojans: Lessons Learned after One Decade of Research," *ACM Transactions on Design Automation of Electronic Systems*, vol. 22, no. 1, pp. 6:1–6:23, May 2016.
- [4] X. Guo, R. G. Dutta, Y. Jin, F. Farahmandi, and P. Mishra, "Pre-silicon security verification and validation: A formal perspective," in *Proceedings of the 52nd Annual Design Automation Conference*, ser. DAC '15. New York, NY, USA: Association for Computing Machinery, Jun. 2015, pp. 1–6.
- [5] X. Cui, E. Koopahi, K. Wu, and R. Karri, "Hardware Trojan Detection Using the Order of Path Delay," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 14, no. 3, pp. 1–23, Jul. 2018.
- [6] M. Lecomte, J. Fournier, and P. Maurine, "An On-Chip Technique to Detect Hardware Trojans and Assist Counterfeit Identification," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 12, pp. 3317–3330, Dec. 2017.
- [7] A. Shabani and B. Alizadeh, "PMTP: A MAX-SAT-Based Approach to Detect Hardware Trojan Using Propagation of Maximum Transition Probability," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 1, pp. 25–33, Jan. 2020.
- [8] M. Sabri, A. Shabani, and B. Alizadeh, "SAT-Based Integrated Hardware Trojan Detection and Localization Approach Through Path-Delay Analysis," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 68, no. 8, pp. 2850–2854, Aug. 2021.
- [9] N. Vashistha, H. Lu, Q. Shi, M. T. Rahman, H. Shen, D. L. Woodard, N. Asadizanjani, and M. Tehranipoor, "Trojan Scanner: Detecting Hardware Trojans with Rapid SEM Imaging Combined with Image Processing and Machine Learning," in *ISTFA 2018*, Phoenix, Arizona, USA, Nov. 2018, pp. 256–265.
- [10] R. Yasaei, S. Faezi, and M. A. Al Faruque, "Golden Reference-Free Hardware Trojan Localization Using Graph Convolutional Network," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 30, no. 10, pp. 1401–1411, Oct. 2022.
- [11] S. Faezi, R. Yasaei, and M. A. Al Faruque, "HTnet: Transfer Learning for Golden Chip-Free Hardware Trojan Detection," in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Feb. 2021, pp. 1484–1489.
- [12] Jie Li and J. Lach, "At-speed delay characterization for IC authentication and Trojan Horse detection," in *2008 IEEE International Workshop on Hardware-Oriented Security and Trust*. Anaheim, CA, USA: IEEE, Jun. 2008, pp. 8–14.
- [13] Y. Lyu and P. Mishra, "Automated Test Generation for Trojan Detection using Delay-based Side Channel Analysis," in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. Grenoble, France: IEEE, Mar. 2020, pp. 1031–1036.
- [14] D. Ismari, J. Plusquellic, C. Lamech, S. Bhunia, and F. Saqib, "On detecting delay anomalies introduced by hardware trojans," in *Proceedings of the 35th International Conference on Computer-Aided Design*. Austin Texas: ACM, Nov. 2016, pp. 1–7.
- [15] V. R. Surabhi, P. Krishnamurthy, H. Amrouch, K. Basu, J. Henkel, R. Karri, and F. Khorrami, "Hardware Trojan Detection Using Controlled Circuit Aging," *IEEE Access*, vol. 8, pp. 77 415–77 434, 2020.
- [16] V. R. Surabhi, P. Krishnamurthy, H. Amrouch, J. Henkel, R. Karri, and F. Khorrami, "Exposing Hardware Trojans in Embedded Platforms via Short-Term Aging," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 11, pp. 3519–3530, Nov. 2020.
- [17] X. Meng, R. Hassan, S. M. P. Dinakr Rao, and K. Basu, "Can Overclocking Detect Hardware Trojans?" in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2021, pp. 1–5.
- [18] R. Ginosar, "Metastability and Synchronizers: A Tutorial," *IEEE Design & Test of Computers*, vol. 28, no. 5, pp. 23–35, Sep. 2011.
- [19] H.-C. Chen and D.-C. Du, "Path sensitization in critical path problem (logic circuit design)," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 12, no. 2, pp. 196–207, Feb. 1993.
- [20] L. De Moura and N. Bjørner, "Z3: An efficient SMT solver," in *Proceedings of the Theory and Practice of Software, 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, ser. TACAS'08/ETAPS'08. Berlin, Heidelberg: Springer-Verlag, Mar. 2008, pp. 337–340.
- [21] T. Gingold, "Ghdl, free and open source vhdl simulator," [online] Available: <https://github.com/ghdl/ghdl>, Apr. 2025.
- [22] S. Yang, "Logic synthesis and optimization benchmarks user guide," *Technical Report 1991-IWLS-UG-Saeyang*, 1991.