# The ATPG Attack for Reverse Engineering of Combinational Hybrid Custom-Programmable Circuits


***Raza Shafiq***
Howard University
raza.shafiq@bison.howard.edu

***Hamid Mahmoodi***
San Francisco State University
mahmoodi@sfsu.edu

***Houman Homayoun***
George Mason University
hhomayou@gmu.edu

***Hassan Salmani***
Howard University
hassan.salmani@howard.edu



**Abstract:** *This paper presents the implementation and evaluation of ATPG attack on hybrid custom-programmable circuits. While functionality of programmable cells are only known to trusted parties, effective techniques for activation and propagation of the cells are introduced. The ATPG attack carefully studies dependency of programmable cells to develop their (partial) truth tables. Results demonstrate the capabilities of the ATPG attack.*



**Keywords:** Reverse-engineering; ATPG attack; hybrid custom-programmable circuits.


## Introduction

The widely practiced horizontal integrated circuit supply chain exposes a design to various types of attacks including the reverse engineering attack. The reverse engineering (RE) attack aims to obtain the detailed design implementation, and it may take different forms [1]. To protect a circuit against the RE attack, new design techniques such as split manufacturing [2], camouflaging [3], and hybrid STT-CMOS circuits [4] have been proposed. Hybrid STT-CMOS circuits combine custom and programmable cells to hide the functionality of certain number of gates.

To reverse engineer circuits that integrate unknown gates such as camouflaged circuits, the RE attack based on satisfiability (SAT) has been also investigated [5]. While the effectiveness of SAT-based attacks becomes limited when the size of circuit and the number of possible identities increase [5], this paper investigates the capability of the automatic test pattern generation (ATPG) attack on circuits that are composed of known and unknown logic gates such as camouflaged circuits and hybrid STT-CMOS circuits.

To determine the functionality of an unknown logic gate ($m$), the ATPG attack is to develop $m$'s (partial) truth table. This requires test pattern application to set inputs of $m$ to different input combinations (the rows of truth table) and to propagate the output of $m$ to some primary outputs. This work introduces a platform for an automated ATPG attack that can be executed by any ATPG tool without need for a high-level of IC testing knowledge. To evaluate the effectiveness of the ATPG attack in practice, it is being applied to hybrid combinational STT-CMOS circuits [4]. ***The contributions of this work are***: (1) introducing the ATPG attack model, (2) studying activation mechanisms for unknown gates based on the stuck-at fault model, (3) analyzing challenges for propagating the output of unknown gates to some observation points, and (4) proposing a platform to evaluate the resiliency of a circuit against the ATPG attack.

### The ATPG Attack Model

The attacker applies a set of test vectors to the circuit and observes its output(s). The attacker hopes to resolve the hybrid circuit by analyzing input and output pairs. The ATPG attack model assumes: (1) an attacker has the netlist of unresolved hybrid custom-programmable circuit, (2) the attacker obtains a configured hybrid custom-programmable circuit (the blackbox circuit), and (3) the attacker generates a set of patterns and applies it to the blackbox circuit.

It is not difficult for an attacker to obtain the netlist of unresolved hybrid custom-programmable circuit. For example, an untrusted manufacturer directly obtains the circuit from the design house. When the circuit is manufactured and released to the market, the attacker can directly order one that is configured by the design house or a trusted vendor, as well. This circuit serves as a blackbox circuit. Having the unresolved netlist and its blackbox counterpart, the attacker generates a set of test patterns by analyzing the unresolved netlist and applies it to the blackbox circuit. Then the attacker processes input-output vectors to resolve the unresolved netlist.

In the ATPG attack, the attacker is to obtain the (partial) truth table of each unknown logic gate. If all rows of truth table are obtained, the programmable logic gate becomes resolved and known. Otherwise, a pool of possible logic gates (candidates) that match the partial truth table is obtained. In such a case, the attacker may select one of candidates and then target the remainder of unknown logic gates.

To develop the truth table of a programmable gate (or called a missing gate), the ATPG attack needs to set the gate's inputs to different combinations and to propagate its output to an observation point. A missing gate can be highly correctly determined if it is independent from any other missing gates. On the contrast, it may not be possible to obtain the complete truth table of a dependent missing gate, and even the obtained partial truth table may not be necessarily correct as it is based on the assumption that some other dependent missing gates are resolved correctly.

Resolving an unknown logic gate consists of two main steps, namely activation and propagation, and Figure 1 presents the overall ATPG attack flow.

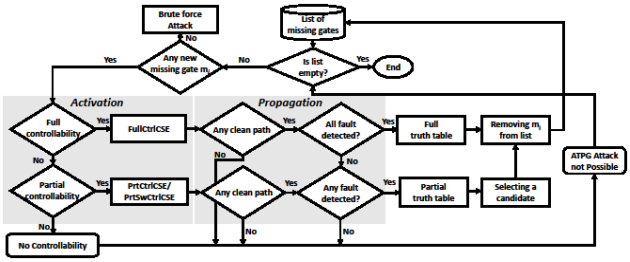


Figure 1. The ATPG Attack Flow.

## Activation for the ATPG Attack

Controllability of inputs of a missing gate through primary inputs for activation is determined by its dependency to other missing gates and can be categorized into: Full Controllability, Partial Controllability, and No Controllability. Full controllability for a missing gate is defined as the existence of no missing gate in the fan-in cone of the missing gate (referred as FullCtrlCSE in Figure 1). For a full-controllability missing gate with $|I|$ number of inputs and $|O|$ number of outputs, there are at most $2^{|I|+|O|}$ combinations that need to be examined to correctly determine its functionality. For example, Figure 2 verifies whether the output of a 2-input missing gate $m_i$ is 0 by replacing it by an AND gate whose inputs are inverted and considering the stuck-at 0 (sa0) at the output of AND gate (the AND00 case). Dependency between missing gates limits their controllability. In the partial controllability for a missing gate, some input signals of a missing gate are driven by some other missing gates while its other inputs are yet accessible through primary inputs (referred as PrtCtrlCSE/PtrSwCtrlCSE in Figure 1). For example, Figure 3 presents a case with two missing gate $m_i$ and $m_j$ where one input of $m_i$ is driven by $m_j$. To determine the row "1X" for $m_i$'s truth table (X represents don't care), the missing gate $m_i$ is replaced by an OR gate whose controllable input is inverted with a stuck-at 1 (sa1) at that input (the OR1X case). If all inputs of a missing gate are driven by some other missing gate, the missing gate holds no controllability, it cannot be resolved.

In general, if any inputs of a missing is controllable, the ATPG attack replaces the missing gate with a specific case to apply a specific input combination to the inputs of missing gate.

## Propagation for the ATPG Attack

The main challenge in propagation of the output of a missing gate to a primary output is the existence of some other missing gates in its fan-out cone. To address the challenge, the attacker needs to find a clean path defined as a path that is not passing through any other missing gate nor reachable by any other missing gate. In large circuits with gates with multiple fan-out branches, the number of paths exponentially increases. This would require considerable storage and processing time to determine clean paths. To address these challenges, obtaining clean paths can be broken in two steps: (1) clean primary outputs, and (2) path direction. A clean primary output is defined as a primary output that contains no missing gates but a target missing gate ($m_i$) in its fan-in cone. Path direction is to determine if a clean path inverts the output of the target missing gate or not.

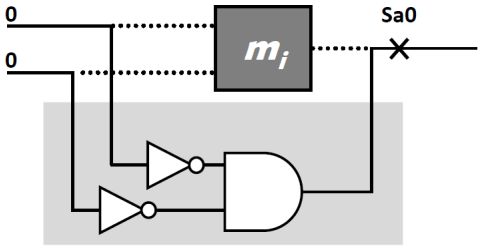


Figure 2. Full controllability (The AND00 case).

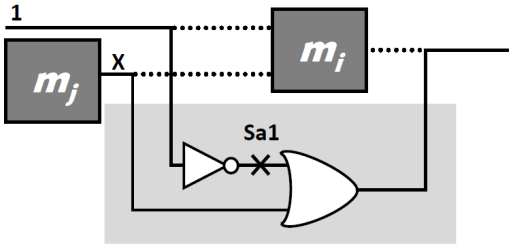


Figure 3. Partial controllability (The OR1X case).

After defining a stuck-at fault at the output of a missing gate (being done at the activation phase), an attacker can check the clean primary outputs. If any of these primary outputs reflect the fault, it means the output of missing gate reaches to this primary output. If the fault at the output of missing gate is stuck-at-0(1) and the primary output also reports stuck-at-0(1), the clean path from the missing gate to this primary output is not inverting; otherwise, the clean path is inverting.

For example, if $m_1$ is the target missing gate in a circuit shown in Figure 4, the following propagation paths for the $m_1$ missing gate exist: $P_1$ = {$g_3$, $g_5$, $g_{11}$}, $P_2$ = {$g_3$, $g_5$, $m_2$, $g_{12}$}, $P_3$ = {$g_3$, $g_5$, $m_2$, $g_{13}$}, and $P_4$ = {$g_3$; $g_{14}$}. The $P_2$ and $P_3$ paths are not clean as both contain the $m_2$ missing gate. On the other hand the $P_1$ and $P_4$ paths are clean paths.

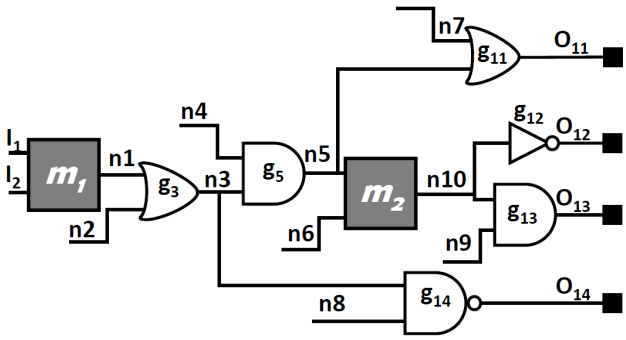


Figure 4. Clean paths for propagation in the ATPG attack.

## Results

The selection algorithms [4] are applied on the ISCA'S 85 combinational circuits [6]. The HOPE fault simulation tool [7] is used to launch the ATPG attack. While HOPE is originally to generate patterns randomly for fault detection, proposed activation and propagation techniques forces HOPE to apply a specific combination of values on the inputs of missing gates and propagate the output of missing gate to some primary outputs. The limit of number of try for HOPE to detect a fault is set to 5000.

Table 1. The ATPG attack on the Independent Selection.

| Circuit Name | # Missing Gate | Average No Patterns | Average Time (min:sec) |
|---|---|---|---|
| C880 | 9 | 786.60 | 00:20.71 |
| C1908 | 4 | 169.10 | 00:12.44 |
| C2670 | 4 | 550.90 | 00.36.75 |
| C3540 | 5 | 491.50 | 00:29.94 |
| C5315 | 3 | 439.20 | 00:41.87 |

Table 1 presents results for the ATPG attack on missing gates selected by the Independent Selection where gates are not reachable from each other directly or indirectly. The results indicate there are few circuits with independent gates, and there are small number of independent missing gates in such circuits (# Missing Gates). Table 1 reports Average No Pattern and Average Time after 10 rounds of HOPE simulation. There exists a roughly high correlation (≈ 0.74) between # Missing Gates and Average No Pattern. Table 1 shows that, Average Time is dependent on the Circuit Size (number of gate in the circuit) and Average Pattern, and there is a high correlation (≈ 0.93) between Average Time and (Circuit Size × Average Pattern).

Table 2. The ATPG attack on the Dependent Selection.

| Circuit Name | # Missing Gate | # Attackable | Average No Patterns | Average Time (min:sec) |
|---|---|---|---|---|
| C499 | 7 | 7 | 439.90 | 09:51.94 |
| C880 | 10 | 2 | 5516.30 | 00:32.60 |
| C1355 | 19 | 2 | 33.80 | 02:29.11 |
| C1908 | 9 | 1 | 73.50 | 03:20.10 |
| C2670 | 9 | 2 | 38.80 | 03:40.81 |
| C3540 | 8 | 6 | 764.00 | 58:54.11 |
| C5315 | 23 | 6 | 10631.90 | 44:35.47 |

Table 2 shows the results of ATPG attack on missing gates selected based on the Dependent Selection (DS) where all gates on a timing path are replace with missing gates. In this paper, only one path in benchmark is constantly targeted. If there exists reconvergence paths between the missing gates in DS, an attacker cannot apply the ATPG attack on all missing gates. Only missing gates that have a clean propagation path with full controllability (FullCtrlCSE) or partial controllability (PrtCtrlCSE) can be attacked. In Table 2, # Attackable shows the number missing gates for which either FullCtrlCSE or PrtCtrlCSE can be evaluated. Except the benchmarks C499 and C3540, considerable portion of missing cannot be targeted due to either lack of controllability or a clean path for propagation. Average Time indicates that the ATPG attack takes longer time for larger circuits.

Table 3 shows the results of ATPG attacks on missing gates selected based on the Parametric-aware Dependent Selection where some gates on a timing path are replaced with missing gates and any gate driven by or drive known gates on the timing path are replaced with missing gates. The results indicate the existence of large number of missing gates (# Missing Gate) while small number of them are really attackable (#Attackable). While Average No Pattern presents smaller number pattern compared to Table 2 because of smaller attackable missing gates, considerable increase in Average Time is being observed.

Table 3. The ATPG attack on the Parametric-aware Dependent Selection.

| Circuit Name | # Missing Gate | # Attackable | Average No Patterns | Average Time (min:sec) |
|---|---|---|---|---|
| C432 | 11 | 2 | 35.20 | 06:30.54 |
| C499 | 13 | 3 | 51.30 | 13:17.52 |
| C880 | 15 | 2 | 586.80 | 19:13.68 |
| C1355 | 21 | 3 | 89.80 | 79:06.96 |
| C1908 | 22 | 1 | 1.70 | 130:45.66 |
| C3540 | 23 | 4 | 95.30 | 1065:56.94 |
| C5315 | 20 | 2 | 21.70 | 658:05.88 |

## Discussion

Assuming idealistically any required pattern to apply a specific values on inputs of a missing and to propagate the output of the missing gate can be deterministically obtained, we can estimate how many patterns may take for the ATPG attack to determine the functionality of missing gates.

Given |M| the number of independent missing gates in a combinational circuit, the minimum number of required test patterns for the ATPG attack to determine all missing gates ($N_{indep}$) is equal to

$$N_{indep} = \sum_{i=1}^{|M|} \alpha_i$$

where α is the average number of required patterns to determine an independent missing gate.

The value of α is determined based on the similarity of the output of the gates. For example, the similarity of 2-input AND gate and 2-input NOR gate is 2 since for two input combinations they produce the same output, and the similarity of 2-input AND gate and 2-input NAND gate is 0

as their output are completely opposite. For 2-input gates, the average similarity of gates is 1.45, so the average required patterns to determine a 2-input missing gate (α) is 2.45. For 3-input gates and 4-input gates, α is equal to 4.2 and 7.4, respectively.

Dependency between missing gates considerably increases the ATPG attack's effort. Given $M$ the number of missing gates that one input of any missing gate is driven by the output of another missing gate, and $|P_i|$ is the size of pool of candidates for a missing gate mi, the average number of required test patterns to determine all missing gates in the dependent selection ($N_{dep}$) is equal to

$$N_{dep} = \prod_{i=1}^{|M|} \alpha_i \times |P_i|$$

Assuming the existence of test patterns to apply different combinations at the accessible inputs of missing gates and to propagate the output of missing gates, the size of pool ($|P_i|$) is 2.7, 2, and 2 for 2-input, 3-input, and 4-input missing gates, respectively. The value of $|P_i|$ is obtained based on the truth table of gates. For example, by ignoring one input of 2-input gates, the truth table of AND would be similar to those of OR, NOR, and XNOR gates. In another case, the truth tables of XOR would be similar to NAND and NOR gates.

If missing gates are selected such that all of their inputs are driven with some other missing gates, the ATPG attack' effort exponentially increases. A more plausible approach for the attacker is to launch SAT-based attack or a brute force attack. Given $|M|$ the number of missing gates, $|I|$ inputs accessible to drive missing gates, and $|P|$ the number of all available gates in a technology library with the same number of inputs as a missing gate, the maximum number of test patterns to determine the missing gates in a brute force attack ($N_{bf}$) is equal to

$$N_{bf} = 2^{|I|} \times |P|^{|M|}.$$

## Reference


[1] R. Torrance and *et. al*., The state-of-the-art in semiconductor reverse engineering, DAC' 2011.

[2] R. Jarvis and *et. al*., Split manufacturing method for advanced semiconductor circuits, May 27 2004. US Patent App. 10/305,670.

[3] R.P. Cocchi, and *et. al*., Method and apparatus for camouflaging a standard cell based integrated circuit with micro circuits and post processing, June 7 2012. US Patent App. 13/370,118.

[4] Theodore Winograd and *et. al*., Hybrid STT_CMOS designs for reverse-engineering prevention, DAC 2016.

[5] M. E. Massad and *et. al*. Integrated circuit (IC) decamouflaging: reverse engineering camouflaged ICs within minutes, NDSS 2015.

[6] ISCAS benchmarks:

*http://pld.ttu.ee/~maksim/benchmarks/*

[7] H. K. Lee and *et. al*. HOPE: an efficient parallel fault simulator for synchronous sequential circuits, TCAD 1996.