2016 IEEE Symposium on Security and Privacy

A2: Analog Malicious Hardware

Kaiyuan Yang, Matthew Hicks, Qing Dong, Todd Austin, Dennis Sylvester
Department of Electrical Engineering and Computer Science
University of Michigan
Ann Arbor, MI, USA
{*kaiyuan, mdhicks, qingdong, austin, dmcs*}@*umich.edu*

***Abstract*—While the move to smaller transistors has been a boon for performance it has dramatically increased the cost to fabricate chips using those smaller transistors. This forces the vast majority of chip design companies to trust a third party—often overseas—to fabricate their design. To guard against shipping chips with errors (intentional or otherwise) chip design companies rely on post-fabrication testing. Unfortunately, this type of testing leaves the door open to malicious modifications since attackers can craft attack triggers requiring a sequence of unlikely events, which will never be encountered by even the most diligent tester.**

In this paper, we show how a fabrication-time attacker can leverage analog circuits to create a hardware attack that is small (i.e., requires as little as one gate) and stealthy (i.e., requires an unlikely trigger sequence before effecting a chip's functionality). In the open spaces of an already placed and routed design, we construct a circuit that uses capacitors to siphon charge from nearby wires as they transition between digital values. When the capacitors fully charge, they deploy an attack that forces a victim flip-flop to a desired value. We weaponize this attack into a remotely-controllable privilege escalation by attaching the capacitor to a wire controllable and by selecting a victim flip-flop that holds the privilege bit for our processor. We implement this attack in an OR1200 processor and fabricate a chip. Experimental results show that our attacks work, show that our attacks elude activation by a diverse set of benchmarks, and suggest that our attacks evade known defenses.

***Keywords*-analog; attack; hardware; malicious; security; Trojan;**

I. Introduction

Hardware is the base of a system. All software executes on top of a processor. That software must trust that the hardware faithfully implements the specification. For many types of hardware flaws, software has no way to check if something went wrong [1], [2]. Even worse, if there is an attack in hardware, it can contaminate all layers of a system that depend on that hardware—violating high-level security policies correctly implemented by software.

The trend of smaller transistors while beneficial for increased performance and lower power, has made fabricating a chip expensive. With every generation of transistor comes the cost of retooling for that smaller transistor. For example, it costs 15% more to setup the fabrication line for each successive process node and by 2020 it is expected that setting-up a fabrication line for the smallest transistor size will require a $20,000,000,000 upfront investment [3]. To amortize the cost of the initial tooling required to support a given transistor size, most hardware companies outsource fabrication.

Outsourcing of chip fabrication opens-up hardware to attack. The most pernicious fabrication-time attack is the dopant-level Trojan [4], [5]. Dopant-level Trojans convert trusted circuitry into malicious circuitry by changing the dopant ratio on the input pins to victim transistors. This effectively ties the input of the victim transistors to a logic level 0 or 1—a short circuit. Converting existing circuits makes dopant-level Trojans very difficult to detect since there are no added or removed gates or wires. In fact, detecting dopant-level Trojans requires a complete chip delayering and comprehensive imaging with a scanning electron microscope [6]. Unfortunately, this elusiveness comes at the cost of expressiveness. Dopant-level Trojans are limited by existing circuits, making it difficult to implement sophisticated attack triggers [5]. The lack of a sophisticated trigger means that dopant-level Trojans are more detectable by post-fabrication functional testing. Thus, dopant-level Trojans represent an extreme on a tradeoff space between detectability during physical inspection and detectability during testing.

To defend against malicious hardware inserted during fabrication, researchers have proposed two fundamental defenses: 1) use side-channel information (e.g., power and temperature) to characterize acceptable behavior in an effort to detect anomalous (i.e., malicious) behavior [7]–[10] and 2) add sensors to the chip that measure and characterize features of the chip's behavior (e.g., signal propagation delay) in order to identify dramatic changes in those features (presumably caused by activation of a malicious circuit) [11]–[13]. Using side channels as a defense works well against large Trojans added to purely combinational circuits where it is possible to test all inputs and there exists a reference chip to compare against. While this accurately describes most existing fabrication-time attacks, we show that it is possible to implement a stealthy and powerful processor attack using only a single added gate. Adding sensors to the design would seem to adapt the side-channel approach to more complex, combinational circuits, but we design an attack that operates in the analog domain until it directly modifies processor

DOI 10.1109/SP.2016.10

state, without affecting features measured by existing on-chip sensors.

We create a novel fabrication-time attack that is controllable, stealthy, and small. To make our attack controllable and stealthy we borrow the idea of counter-based triggers commonly used to hide design-time malicious hardware [14], [15] and adapt it to fabrication-time. To make our attack small, we replace the hundreds of gates required by conventional counter-based triggers implemented using digital logic with analog components—a capacitor and a few transistors wrapped-up in a single gate. Our attack works by siphoning charge from a target wire every time it toggles and storing that charge in a capacitor. If the wire toggles infrequently, the capacitor voltage stays near zero volts due to natural charge leakage. When the wire toggles frequently, charge accumulates on the capacitor—faster than it leaks away, eventually fully charging the capacitor. When the voltage on the capacitor rises above a threshold, it deploys the payload—whose output is attached to a flip-flop changing that victim flip-flop to any desired value.

To demonstrate that our attack works for real chips, we implement a privilege escalation attack in the OR1200 [16] open source processor. We attach our capacitor to a signal that infrequently toggles with normal software, but toggles at a high rate with specially-crafted, usermode trigger programs. For our victim flip-flop, we select the privilege bit (i.e., user or supervisor mode). Because the attack taps into both the digital layer and the analog layer, it is unable to be simulated completely using existing tools that operate at only a single layer. As such, we fabricate our malicious processor to verify its end-to-end operation. Experiments with our fabricated malicious processor show that it is trivial for a knowing attacker to activate the attack and escalate the privilege of their unprivileged process—all from usermode code, without operating system intervention. Experiments with an array of embedded systems benchmarks [17] show that it is unlikely that arbitrary software will trigger our attack.

This paper presents three contributions:

1) We design and implement the first fabrication-time processor attack that mimics the triggered attacks often added during design time. As a part of our implementation, we are the first to show how a fabrication-time attacker can leverage the empty space common to Application-Specific Integrated Circuit (ASIC) layouts to implement malicious circuits.
2) We are the first to show how an analog attack can be much smaller and more stealthy than its digital counterpart. Our attack diverts charge from unlikely signal transitions to implement its trigger, thus, it is invisible to all known side-channel defenses. Additionally, as an analog circuit, our attack is below the digital layer and missed by functional verification performed on the hardware description language. Moreover, our attack relies on a complex and unlikely analog trigger sequence, thus, it is impractical to simulate at the analog level—which motivated us to fabricate a chip to verify that our attacks worked.
3) We fabricate the first openly malicious processor and then evaluate the behavior of our fabricated attacks across many chips and changes in environmental conditions. We compare these results to SPICE simulation models [1].

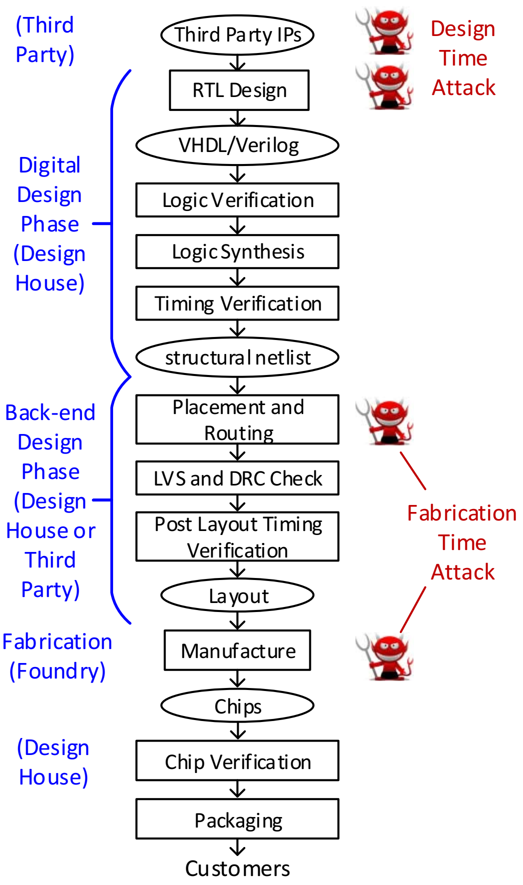


Figure 1: Typical IC design process with commonly-research threat vectors highlighted in red. Thu blue text and brackets highlights the party in control of the stage(s).

II. Background

The focus of this paper is fabrication-time attacks that leverage analog characteristics of integrated circuits as a trigger. In this section, we start with an overview of the integrated circuit (IC) design process and possible malicious attacks at different phases. Then we discuss the threat model of our proposed attack.

[1] We make both the software and hardware code pertaining to A2 publicly available [18].

A. Integrated Circuit Design Process

Figure 1 shows the typical design process of integrated circuits [19]. This process often involves collaboration between different parties all over the world and each step is likely done by different teams even if they are in same company, which makes it vulnerable to malicious attacks by rogue engineers involved in any of the above steps.

B. Threat Model

It is possible to implement our attack at either the back-end phase or at the fabrication phase. Since it is strictly more challenging to implement attacks at the fabrication phase due to limited information and ability to modify the design compared to the back-end phase, we focus on that threat model.

The attacker starts with a Graphic Database System II (GDSII) file that is a polygon representation of the completely laid-out and routed circuit. This is a very restrictive threat model as it means that the attacker can only modify existing circuits or—as we are the first to show in this paper—add attack circuits to open spaces in the laid-out design. The attacker can *not* increase the dimensions of the chip or move existing components around. This restrictive threat model also means that the attacker must perform some reverse engineering to select viable victim flip-flops and wires to tap. As detailed in Section VI-C, a public specification of the chip to be fabricated makes this process easier. After the untrusted fabrication house completes fabrication, it sends the fabricated chips off to a trusted party for post-fabrication testing. Our threat model assumes that the attacker has no knowledge of the test cases used for post-fabrication testing, which dictates the use of a sophisticated trigger to hide the attack.

Leading up to the attacker getting a GDSII file, our threat model assumes that a design house correctly implements the specification for the chip's behavior in some hardware description language (HDL). Once the specification is implemented in an HDL and that implementation has been verified, the design is passed to a back-end house. Our threat model assumes that the back-end house—who places and routes the circuit—is also trusted. This means that the delivered GDSII file represents a perfect implementation—at the digital level of abstraction—of the chip's specification. The attacker is free to modify the design at both the digital level by adding, removing, or altering circuits and at the analog level (e.g., increasing electromagnetic coupling of wires through layout or adding analog components).

Note that the chip vendor is free to run any additional tests on the fabricated chip. We assume that the attacker has no knowledge or control about post-fabrication testing. We only assume that testing is bound by the limits of practicality.

III. Attack Methods

A hardware attack is composed of a trigger and a payload. The trigger monitors wires and state within the design and activates the attack payload under very rare conditions such that the attack stays hidden during normal operation and testing. Previous research has identified that evading detection is a a critical property for hardware Trojans designers [20]. Evading detection involves more than just avoiding attack activation during normal operation and testing though, it includes hiding from visual/side-channel inspection. There is a tradeoff at play between the two in that the more complex the trigger (i.e., the better that it hides at run time), the larger the impact that trigger has on the surrounding circuit (i.e., the worse that it hides from visual/side-channel inspection).

We propose A2, a fabrication-time attack that is small, stealthy, and controllable. To achieve these outcomes, we develop trigger circuits that operate in the analog domain; circuits based on charge accumulating on a capacitor from infrequent events inside the processor. If the charge-coupled infrequent events occur frequently enough, the capacitor will fully charge and the payload is activated, which deploys a privilege escalation attack. We target the privilege bit in a processor, as privilege escalation constitutes a simple payload with maximum capability provided to the attacker. Our analog trigger similar to the counter-based triggers often used in digital triggers, except using the capacitor has the advantage of a natural reset condition due to leakage.

We create the trigger using a custom analog circuit that a fabrication-time attacker inserts after the entire design has been placed and routed. Compared to traditional digitally described hardware Trojans, our analog trigger maintains a high level of stealth and controllability, while dramatically reducing the impact on area, power, and timing due to the attack. An added benefit of a fabrication-time attack compared to a design time attack (when digital-only triggers tend to get added) is that the fabrication-time attack has to pass through few verification stages.

To highlight the design space of our analog trigger technique, we show how an attacker can connect several simple trigger circuits to create arbitrary trigger patterns to improve secrecy and/or controllability. In addition to the number of stages, we show how an attacker can tune several design parameters to achieve trade-offs between the ease of triggering the payload and its stealthiness, even to the point of creating triggers that can only be expressed under certain process variation and/or environmental conditions. This trade-off space is only possible through the use of an analog-based trigger.

In the following sections, we describe the design and working principles of our analog trigger. We present the designs of both a base single-stage trigger and a more complex, but flexible, multi-stage trigger. We also describe our privilege escalation attack which also has analog com-

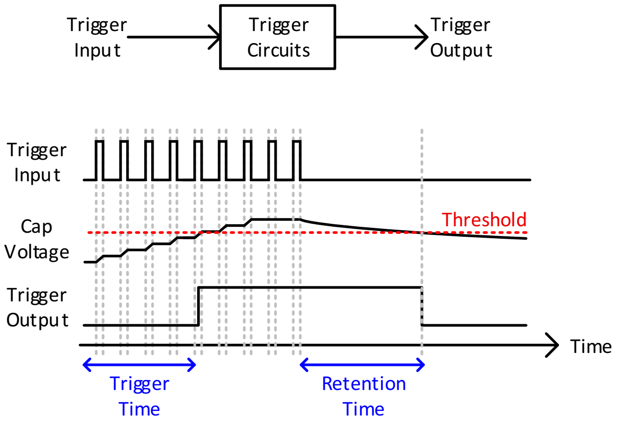


Figure 2: Behavior model of proposed analog trigger circuit.

ponents. We conclude with an analysis of how an attacker, bounded by our threat model, would go about attacking a processor.

A. *Single Stage Trigger Circuit*

Based on our threat model, the high-level design objectives of our analog trigger circuit are as follows:

1) Functionality: The trigger circuit must be able to detect toggling events of a target victim wire similar to a digital counter and the trigger circuit should be able to reset itself if the trigger sequence is not completed in a timely manner.
2) Small area: The trigger circuit should be small enough to be inserted into the empty space of an arbitrary chip layout after placement and routing of the entire design. Small area overhead also implies better chance to escape detection.
3) Low power: The trigger circuit is always actively monitoring its target signals, therefore power consumption of the design must be minimized to hide it within the normal fluctuations of entire chip's power consumption.
4) Negligible timing perturbation: The added trigger circuit must not affect the timing constraints for common case operation and timing perturbations should not be easily separable from the noise common to path delays.
5) Standard cell compatibility: Since all digital designs are based on standard cells with fixed cell height, our analog trigger circuit should be able to fit into the standard cell height. In addition, typical standard cells use only metal layer 1 [2] for routing while higher metal layers are reserved for connections between cells, therefore it is desirable for the trigger circuit to use only metal layer 1 for easier insertion into final layout and detection more difficult.

To achieve these design objectives, we propose an attack based on charge accumulation inside capacitors. A capacitor acts as a counter which performs analog integration of charge from a victim wire while at the same time being able to reset itself through natural leakage of charge. A behavior model of charge accumulation based trigger circuits comprises 2 parts.

1) Charge accumulation: Every time the victim wire that feeds the trigger circuit's capacitor toggles (i.e., changes value), the capacitor increases in voltage by some ΔV. After a number of toggles, the capacitor's voltage exceeds a predefined threshold voltage and enables the trigger's output—deploying the attack payload. The time it takes to activate fully the trigger is defined as *trigger time* as shown in Figure 2. *Trigger time* equals toggling frequency of input victim wire multiplied by the number of consecutive toggles to fill the capacitor.
2) Charge leakage: A leakage current exists over all time that dumps charge from the trigger circuit's capacitor, reducing the capacitor's voltage. The attacker systematically designs the capacitor's leakage and accumulation such that leakage is weaker than charge accumulation, but just enough to meet some desired *trigger time*. When the trigger input is inactive, leakage gradually reduces the capacitor's voltage even eventually disabling an already activated trigger. This mechanism ensures that the attack is not expressed when no intentional attack happens. The time it takes to reset trigger output after trigger input stops toggling is defined as *retention time* as shown in Figure 2.

Because of leakage, a minimum toggling frequency must be reached to successfully trigger the attack. At minimum toggling frequency, charge added in each cycle equals charge leaked away. *trigger time* is dependent on toggling frequency, lower toggling rate requires more cycles to trigger because more charge is leaked away each cycle, meaning less charge accumulated on the capacitor each cycle. *retention time* is only dependent on the strength of leakage current. *Trigger time* and *retention time* are the two main design parameters in our proposed analog trigger attack circuits that we can make use of to create flexible trigger conditions and more complicated trigger pattern as discussed in Section III-B. A stricter triggering condition (i.e., faster toggling rate and more toggling cycles) reduces the probability of a false trigger during normal operation or post-fabrication testing, but non-idealities in circuits and process, temperature and voltage variations (PVT variations) can cause the attack to fail—impossible to trigger or trivial

[2] Several layers of metal wires are used in modern CMOS technologies to connect cells together, lower level metal wires are closer to transistors at bottom and have smaller dimensions for dense but short interconnections, while higher metal layers are used for global routing. The lowest layer of metal is usually assigned as metal layer 1 and higher metal layers have correspondingly larger numbers.

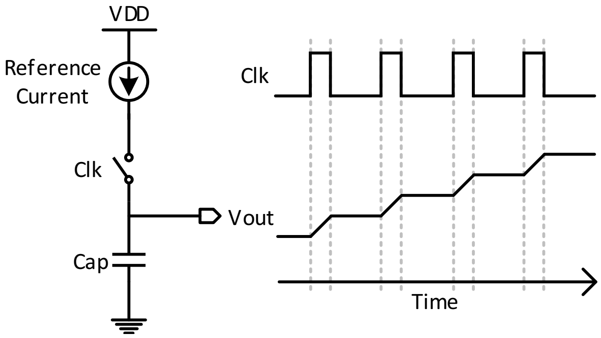


Figure 3: Concepts of conventional charge pump design and waveform.

to accidentally trigger—for some chips. As a result, a trade-off should be made here between a reliable attack that can be expressed in every manufactured chip under varying environmental conditions and a more stealthy attack that can only be triggered for certain chips, under certain environmental conditions, and/or very fast toggling rate of trigger inputs generated by software.

To find a physical implementation of the function described previously, we first try a charge pump widely used in phase locked loop (PLL) designs as shown in Figure 3. Clk in the figure represents some toggling wire that adds charge to Cap capacitor during positive phase of Clk. The voltage step added to Cap during one positive phase can be calculated as,

$$\Delta V = \frac{I_{ref} \times T_{positive}}{Cap} \tag{1}$$

This implies that the voltage on the cap can exceed a threshold in $V_{threshold}/\Delta V$ cycles. Due to our area and power requirements, we need to minimize I_{ref} and Cap size while maintaining an acceptable number of cycles to trigger the attack. There are 3 common methods to implement capacitors in CMOS technology: transistor gate oxide cap (MOS cap), metal-insulator-metal cap (MIM cap) and metal-oxide-metal (MOM cap). The other 2 options require higher metal layers and have less capacitance density, therefore we select the MOS cap option. Given the area constraints, our MOS cap can be at most tens of fF, which means the current reference should be in nA range. Such a current reference is nontrivial to design and varies greatly across process, temperature, and voltage variations. Therefore, we need a new circuit design to solve these problems for a more reliable and stealthy attack. However, the circuit in Figure 3 is useful for attacks that wish to impact only a subset of manufactured chips or for scenarios where the attacker can cause the victim wire to toggle at a high rate for hundreds of cycles.

A new charge pump circuit specifically designed for the attack purpose is shown in Figure 4. Instead of using reference current and positive phase period of Clk to control ΔV, the new scheme uses one additional small unit capacitor $Cunit$ to better control the amount of charge dumped on main capacitor each time. During the negative phase of Clk, $Cunit$ is charged to VDD. Then during positive phase of Clk, the two capacitors are shorted together, causing the two capacitors to share charges. After charge sharing, final voltage of the two capacitors is the same and ΔV on $Cmain$ is as,

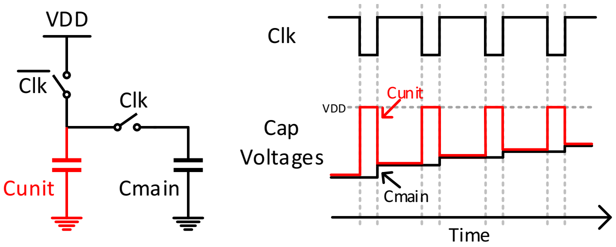


Figure 4: Design concepts of analog trigger circuit based on capacitor charge sharing.

$$\Delta V = \frac{Cunit \times (VDD - V_0))}{Cunit + Cmain} \tag{2}$$

where V_0 is initial voltage on $Cmain$ before the transition happens. As can be seen, ΔV is decreasing as the voltage ramps up and the step size solely depends on the ratio of the capacitance of the two capacitors. We can achieve different *trigger time* values by sizing the two capacitors. Compared to the design in Figure 3, the new scheme is edge triggered rather than level triggered so that there is no requirement on the duty cycle of trigger inputs, making it more universal. The capacitor keeps leaking over time and finally ΔV equals the voltage drop due to leakage, which sets the maximum capacitor voltage.

A transistor level schematic of the proposed analog trigger circuit is shown in Figure 5. $Cunit$ and $Cmain$ are implemented with MOS caps. $M0$ and $M1$ are the 2 switches in Figure 4. A detector is used to compare cap voltage with a threshold voltage and can be implemented in two simple ways as shown in Figure 6. One option is an inverter, which has a switching voltage depending on sizing of the two transistors and when the capacitor voltage is higher than the switching voltage, the output is 0; otherwise, the output is 1. The other option is a Schmitt trigger, which is a simple comparator with hysteresis. It has a large threshold when input goes from low to high and a small threshold when input goes from high to low. The hysteresis is beneficial for our attack, because it extends both *trigger time* and *retention time*.

In practice, all transistors have leakage currents even in their off state and our capacitors are very small, therefore the cap voltage is affected by leakage currents as well.

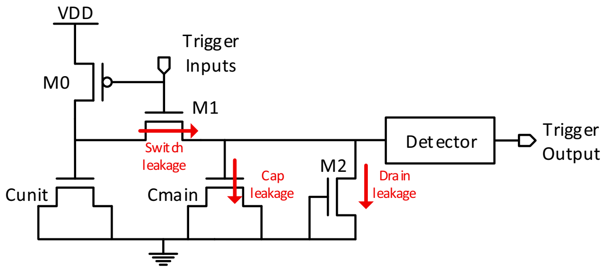
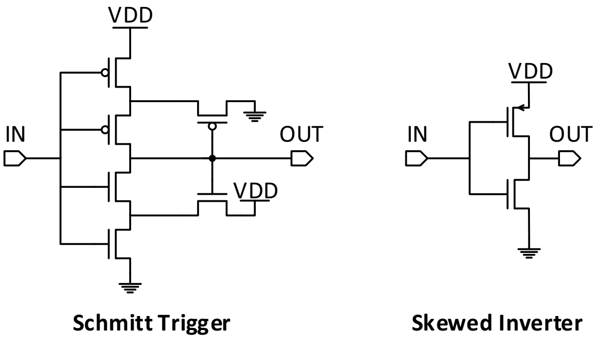


Figure 5: Transistor level schematic of analog trigger circuit.



Schmitt Trigger **Skewed Inverter**

Figure 6: Schematics of detector circuits.

To balance the leakage current through $M0$ and $M1$, an additional leakage path to ground (NMOS $M2$ in Figure 5) is added to the design. An attacker must carefully calculate all leakage paths flowing to and out of the capacitor node in order to balance their effects to achieve the *trigger time* and *retention time* targets. There are three major leakage paths in our analog trigger design: sub-threshold leakage current through switch $M1$, transistor $M2$, and gate tunneling leakage current (as shown in Figure 5). Because leakage currents are sensitive to process, voltage and temperature variations, balancing all the leakage paths is the most challenging part in the implementation of a reliable trigger analog trigger.

For the trigger circuit to work, capacitor voltage without any toggling on its input wire should be low enough to not, in any manufacturing or environmental corner case, be self-triggering. Also, the maximum voltage under the fastest rate of toggling by the victim wire that the attacker can produce must be enough to have a good margin for successful attack, allowing a wider range of acceptable toggling rates that will eventually trigger the attack. These conditions should be met under all PVT variations for a reliable attack, or under certain conditions if attacker only want the attack to be successful under these conditions. No matter what the design target is, minimum voltage should always be kept lower than threshold voltage to avoid exposing the attack in normal use.

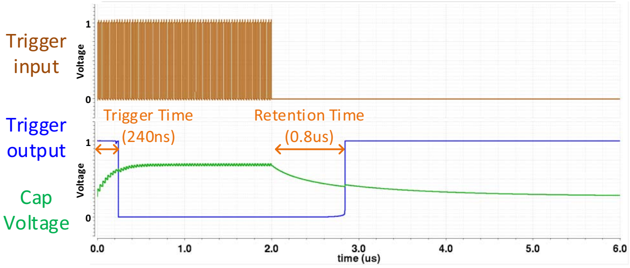


Figure 7: SPICE simulation waveform of analog trigger circuit.

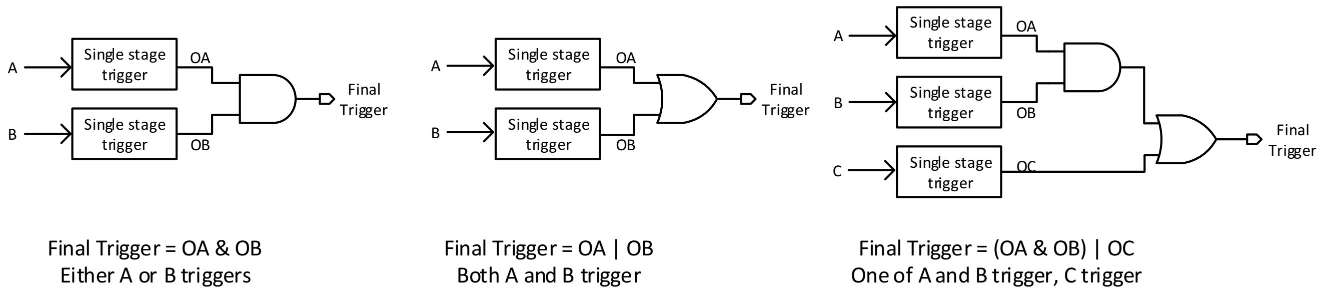
A SPICE simulation waveform is shown in Figure 7 to illustrate the desired operation of our analog trigger circuit after optimization. The operation is same as the behavioral model that we proposed in Figure 2, indicating that we can use the behavior model for system-level attack design.

B. Multi-stage Trigger Circuit

The one-stage trigger circuit described in the previous section takes only one victim wire as an input. Using only one trigger input limits the attacker in two ways:

1) False trigger activations: Because fast toggling of one signal for tens of cycles triggers the single stage attack, there is still a chance that normal operations or certain benchmarks can expose the attack. We can imagine cases where there is only a moderately controllable wire available. A single-stage trigger might be prone to false activations in such a scenario, but a multi-stage trigger could use wires that normally have mutually-exclusive toggle rates as inputs, making it stealthy and controllable.
2) Software flexibility: Certain instructions are required to cause fast toggling of the trigger input and there is not much room for flexible and stealthy implementation of the attack program. For example, some types of multi-stage triggers could support a wide range of attack programs. This would allow the attacker to repeatedly compromised a victim system.

To make the attack even more stealthy, we note that an attacker can make a logical combination of two or more single-stage trigger outputs to create a variety of more flexible multi-stage analog trigger. Basic operations to combine two triggers together are shown in Figure 8. When analyzing the behavior of logic operations on single stage trigger output, it should be noted that the single-stage trigger outputs 0 when trigger condition is met. Thus, for AND operation, the final trigger is activated when either A or B triggers fire. For OR operation, the final trigger is activated when both A and B triggers fire. It is possible for an attacker to combine these simple AND and OR-connected single-stages triggers into an arbitrarily complex multi-level multi-stage trigger. Figure 8 show what such a trigger could look



Final Trigger = OA & OB
Either A or B triggers

Final Trigger = OA | OB
Both A and B trigger

Final Trigger = (OA & OB) | OC
One of A and B trigger, C trigger

Figure 8: Basic ways of connecting single-stage triggers to form a multi-stage trigger.

like, creating a two level multi-stage trigger with the logical expression $(OA\&OB)|OC$. This third trigger fires when trigger C and one of triggers A or B fire. Lastly, it is important to note that not only can the inputs A, B, and C be different, but the internal circuit parameters for each single-stage trigger can also be different (even though we treat them as identical for simplicity).

Due to the analog characteristics of the trigger circuits, timing constraints limit the construction of multi-stage triggers, but also make accidental trigger probability vanishingly rare. A single-stage trigger circuit has two timing parameters, *trigger time* and *retention time*. For AND operation, the timing constraint is same as for a single-stage trigger, because only one of the triggers must activate. For OR operation, there are two methods to trigger the attack: 1) alternatively run the instructions to toggle victim wires A and B or 2) run the instructions to toggle A first for enough cycles to activate the trigger and then run the instructions to trigger B. For the first method, the timing constraint is minimum toggling frequency, because adding n stages reduces the toggling frequency for each trigger circuit by n times. For the second method, the timing constraint is that *retention time* of the stage n should be larger than the total *trigger time* of the following stages stages.

C. Triggering the Attack

Once the trigger circuit is activated, payload circuits activate hidden state machines or overwrite digital values directly to cause failure or assist system-level attacks. The payload can also be extra delay or power consumption of target wires to leak information or cause failure. For A2, the payload design is independent of the trigger mechanism, so our proposed analog trigger is suitable for various payload designs to achieve different attacks. Since the goal of this work is to achieve a Trojan that is nearly invisible while providing a powerful foothold for a software-level attacker, we couple our analog triggers to a privilege escalation attack [21]. We propose a simple design to overwrite security critical registers directly as shown in Figure 9. In any practical chip design, registers have asynchronous set or/and reset pins for system reset. These reset signals are asynchronous with no timing constraints so that adding one gate into the reset signal of one register does not affect functionality or timing constraints of the design. Since the analog trigger circuit output is 0 when activated, we insert an AND gate between the existing reset wire and our victim flip-flop for active-low reset flops and we insert a NOR gate for for active-high set flops. Moreover, because there are no timing constraints on asynchronous inputs, the payload circuit can be inserted manually after final placement and routing together with the analog trigger circuits in a manner consistent with our threat model.

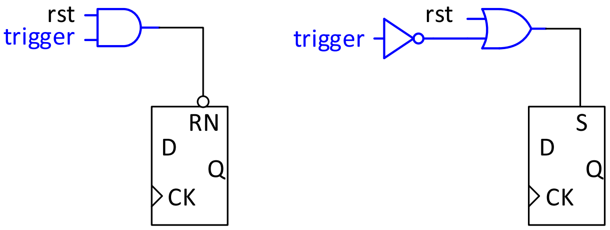


Figure 9: Design of payload to overwrite register value. Gates in blue lines are inserted for attack.

D. Selecting Victims

It is important that the attacker validate their choice of victim signal; this requires showing that the victim wire has low baseline activity and its activity level is controllable given the expected level of access of the attacker. To validate that the victim wire used in A2 has a low background activity, we use benchmarks from the MiBench embedded systems benchmark suite. We select these benchmarks due to their diverse set of workload characteristics and because they run on our resource-limited implementation. We expect that in a real-world attack scenario, the attacker will validate using software and inputs that are representative of the common case given the end use for the attacked processor. For cases where the attacker does not have access to such software or the attacked processor will see a wide range of use, the attacker can follow A2's example and use a multi-stage trigger with wires that toggle in a mutually-exclusive

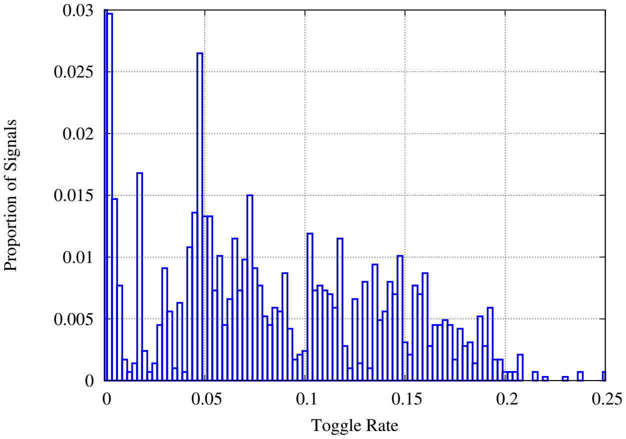


Figure 10: Distribution of paths toggling rate when running a benchmark program.

fashion and require inputs that are unlikely to be produced using off-the-shelf tools (e.g., GCC).

Validating that the victim wire is controllable requires that the attacker reason about their expected level of access to the end user system for the attacked processor. In A2, we assume that the attacker can load and execute any unprivileged instruction. This allows us to create hand-crafted assembly sequences that activate the attack—remember that we selected victim wires that off-the-shelf tools will not produce code significantly activates. While this model works for attackers that have an account on the system, attackers in a virtual machine, or even attackers that can convince users to load code, we did not explore the challenges of less controllable attack scenarios. Two examples are triggering the attack from Javascript and triggering the attack situationally (e.g., radar containing the attacked chip senses a certain type of plane). We expect that our attack supports such triggering scenarios as there is no fundamental difference from running handcrafted unprivileged code: executable code contains a multitude of different instructions and different instructions activate different sets of wires in the processor. The difference is just an extra layer of abstraction. One challenge that we anticipate is the extra layer of abstraction will likely reduce the range of activity on potential victim wires. Our experimental results show that the attacker can deal with this by changing parameters of the analog trigger or even through careful use of a multi-stage trigger.

IV. Implementation

To experimentally verify A2, we implement and fabricate it inside an open source processor with the proposed analog Trojans inserted in 65nm General Purpose (GP) CMOS technology. Because of the time and monetary cost of hardware fabrication, we include multiple attacks in each chip. One set of attacks are Trojans aimed at exposing A2's end-to-end operation, while the other set of attacks are implemented outside the processor, directly connected to IO pins so that we can investigate trigger behavior directly. In this section, we detail the selection of the trigger and attack payload in an OR1200 processor, the activity trigger insertion flow, and analog trigger testing structures.

A. Attacking a Real Processor

We implemented a complete open source OR1200 processor [16] to verify our complete attack including software triggers, analog triggers and payload. The OR1200 CPU is an implementation of the 32-bit OR1K instruction set with 5-stage pipeline. The implemented system in silicon consists of OR1200 core with 128B instruction cache and an embedded 128KB main program memory connected through a Wishbone bus. Standard JTAG interface and custom scan chain are implemented to load program, control and monitor the processor.

The OR1K instruction set specifies the existence of a privileged register called the Supervision Register (SR). The SR contains bits that control how the processor operates (e.g., MMUs and caches enabled) and flags (e.g., carry flag). One particular bit is interesting for security purposes; SR[0] controls the privilege mode of user, with 0 denoting user mode and 1 denoting supervisor mode. By overwriting the value of this register, an attacker can escalate a usermode process to supervisor mode as a backdoor to deploy various high-level attacks [20], [21]. Therefore, we make the payload of our attack setting this bit in the SR to 1 to give a usermode process full control over the processor. In order to evaluate both the one-stage and two-stage triggers described earlier, we have our two-stage triggered attack target SR[1]. Normally, this register bit controls whether timer-tick exceptions are enabled, but since our system does not use the timer and it SR[1] requires privileged software to change its value, it is a simple way to know if our two-stage attack works.

Our analog trigger circuits require trigger inputs that can have a high switching activity under certain (attacker) programs, but are almost inactive during testing or common case operation so that the Trojan is not exposed [3]. To search for suitable victim wires to serve as trigger inputs, we run a series of programs on the target processor in a HDL simulator, capturing the toggling rates of all wire. Figure 10 shows a histogram of wire toggling rates for the basicmath benchmark from MiBench (see Section V). As the figure shows, approximately 3% of total wires in the OR1200 have nearly zero activity rate, which provides a wide range of options for an attacker. The target signals must also be easy to control by attack programs. To find filter the low activity wires for attacker controllability, we simulate our

[3] Exposing the attack during normal operation may be acceptable as non-malicious software does *not* attempt to access privileged processor state. Additionally, current operating systems blindly trust the processor, so they are likely to miss sporadic privilege escalations.

```
{r0 is a non-zero register but reads as zero in user mode}
Initialize SR[0]=0                    {initialize to user mode}
while Attack_Success==0 do
  $i \leftarrow 0$
  while $i < 500$ do
    $z \leftarrow 1/0$
    $i \leftarrow i + 1$
  end while
  if read(special register r0) $\neq 0$ then
    $Attack\_Success \leftarrow 1$
  end if
end while
```

Figure 11: Program that activates the single-stage attack.

attack program in the same setup and identify the wires whose toggle rates increased dramatically. In our attack, we select divide by zero flag signal as the trigger for one-stage attack, because it is unlikely for normal programs to continuously perform division-by-zero while it is simple for an attacker to deliberately perform such operations in a tight loop. Fortunately, the OR1200 processor only sets a flag in the SR when a division-by-zero occurs. For the two-stage trigger, we select wires that report whether the division was signed or unsigned as trigger inputs. The attack program alternatively switches the two wires by performing signed, then unsigned division, until both analog trigger circuits are activated, deploying the attack payload. Pseudo codes for both the one-stage and two-stage attack triggering software sequences are shown in Figure 11 and Figure 12.

Triggering the attack in usermode-only code that does not alert the operating system is only the first part of a successful attack. For the second part, the attacker must be able to verify that there triggering software worked—without risk of alerting the operating system. To check whether the attack is successful, we take advantage of a special feature of some registers on the OR1200: some privileged registers are able to be read by usermode code, but the value reported has some bits redacted. We use this privilege-dependent read behavior as a side-channel to let the attacker's code know whether it has privileged access to the processor or not.

B. Analog Activity Trigger

Here we cover the implementation details of our analog triggers. To verify the first-order behavior of our analog trigger circuits, we implement, optimize, and simulate them using a SPICE simulator. Once we achieve the desired trigger behavior in simulation, we implement both the one-stage and two-stage trigger circuits in $65nm$ GP CMOS technology. Both trigger circuits are inserted into the processor to demonstrate our proposed attack. To fully characterize the performance of the trigger circuits, standalone testing structures are added to the test chip.

```
{r0 is a non-zero register but reads as zero in user mode}
Initialize SR[0]=0                    {initialize to user mode}
while Attack_Success==0 do
  $i \leftarrow 0$
  while $i < 500$ do
    $z \leftarrow a/b$                         {signed division}
    $z \leftarrow c/d$                       {unsigned division}
    $i \leftarrow i + 1$
  end while
  if read(special register r0) $\neq 0$ then
    $Attack\_Success \leftarrow 1$
  end if
end while
```

Figure 12: Program that activates the two-stage attack.

1) Implementation in 65nm GP technology: For prototype purposes, we optimize the trigger circuit towards a reliable version because we can only get a limited number of chips for measurement with no control of process variation and building a reliable circuit under process, temperature, and voltage (PVT) variations is always more challenging than only optimizing for a certain PVT range—i.e., we construct our attacks so that they work in all fabricated processors at all corner-case environments. For robustness, the Schmitt trigger shown in Figure 6 is used as detector circuit. Out of the three leakage paths shown in Figure 5, gate leakage causes most trouble because it has an exponential dependence on gate voltage, making the leakage much stronger when capacitor voltage ramps up. The gate leakage also has exponential dependence on gate oxide thickness of the fabrication technology, because gate leakage is physically quantum tunneling through gate oxide. Unfortunately, $65nm$ CMOS technology is not a favorable technology for our attack, because the gate oxide is thinner than older technologies due to dimension scaling and also thinner than latest technologies because high-κ metal gate techniques now being employed to reduce gate leakage (we use $65nm$ due to its cost savings and it is still a popular process node). Through careful sizing, it's still possible to design a circuit robust across PVT variations, but this requires trading-off *trigger time* and *retention time* as shown in in the simulation waveform of our analog activity trigger depicted in Figure 7.

To reduce gate leakage, another solution is to use thick oxide transistors commonly used in IO cells as the MOS cap for $Cmain$, which shows negligible gate leakage. This option provides larger space for configuration of *trigger time* and *retention time*, but requires larger area due to design rules. SPICE simulation results of the trigger circuits are shown in Figure 13. A zoomed waveform of the trigger operation is shown in the upper waveform, while the entire operation, including trigger and decay, is shown in the lower

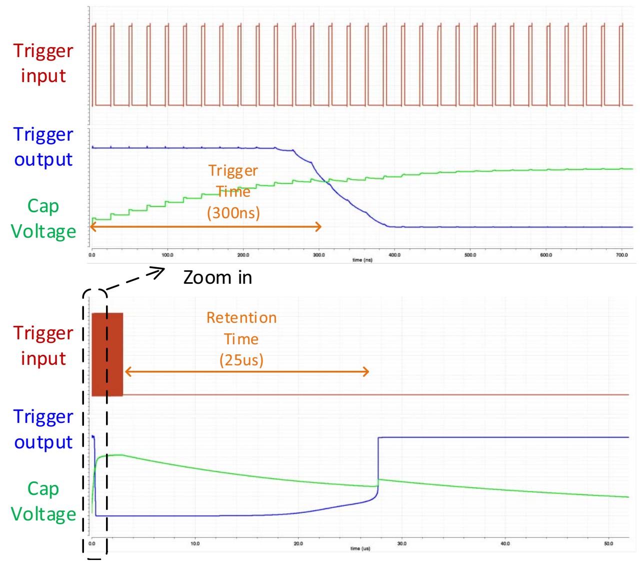


Figure 13: SPICE simulation waveform of analog trigger circuit using IO devices in 65nm CMOS.

plot. A *trigger time* of $300ns$ and *retention time* of $25\mu s$ are marked on the waveforms. Trigger circuit using IO device is implemented for two-stage attack and the one without IO device is used for one-stage attack in the system.

We also performed exploratory simulations of our trigger circuits in $65nm$ Low Power technology, which has significantly less leakage current which is better suited for low power applications. In Low Power technology, no IO device is needed to achieve robust trigger circuits with large *trigger time* and *retention time*. Thus, from an attackers perspective, Low Power technology makes implementing A2 easier and, as detailed in Section V-C, harder to detect.

2) Inserting A2 into existing chip layouts: Since A2's analog trigger circuit is designed to follow sizing and routing constraints of standard cells and have occupy the area comparable to a single standard cell, inserting the trigger circuit to the layout at fabrication time is not complicated. All digital designs nowadays are based on standard cells and standard cells are placed in predefined rows. In typical placement and routing cases, around 60% to 70% of total area is used for standard cells, otherwise routing can not complete due to routing congestion (our chip is more challenging to attack as it has 80% area utilization). Therefore, in any layout of digital designs, empty space exists. This empty space presents an opportunity for attackers as they can occupy the free space with their own malicious circuit. In our case, we requires as little space as one cell. There are 4 steps to insert a trigger into layout of a design:

1) The first step is to locate the signals chosen as trigger inputs and the target registers to attack. The insertion of A2 attack can be done at both back-end placement and routing stage and fabrication stage. Our attack model focuses on the fabrication stage because it is significantly more challenging and more stealthy compared to attack at back-end stage. The back-end stage attacker has access to the netlist of the design, so locating the desired signal is trivial. But an attack inserted at back-end stage can still be discovered by SPICE simulation and layout checks, though the chance is extremely low if no knowledge about the attack exists and given the limits of current SPICE simulators. In contrast, fabrication time attacks can only be discovered by post-silicon testing, which is believed to be very expensive and difficult to find small Trojans. To insert an attack at during chip fabrication, some insights about the design are needed, which can be extracted from layout or from a co-conspirator involved in design phase, even split manufacturing technique may not prevent the attacker from finding the target wires, as discussed in Section VI-C.
2) Once the attacker finds acceptable victim wires for trigger inputs and attack payload target registers, the next step is to find empty space around the victim wire and insert the analog trigger circuit. Unused space is usually automatically filled with filler cells or capacitor cells by placement and routing tools. Removing these cells will not affect the functionality or timing, because they are inserted as the last step after all connectivity and timing checks. Because the layout of trigger circuit only uses metal 1, inserting it to unused space will not block routed signals because metal 1 is barely used for global routing.
3) To insert the attack payload circuit, the reset wire needs to be cut as discussed in Section III-C. It has been shown that timing of reset signal is flexible, so the AND or OR gate only need to be placed somewhere close to the reset signal. Because the added gates can be a minimum strength cell, their area is small and finding space for them is trivial.
4) The last step is to manually do the routing from trigger input wires to analog trigger circuit and then to the payload circuits. There is no timing requirement on this path so that the routing can go around existing wires at same metal layer (jogging) or jump over existing wires by going to another metal layer (jumping), in order to ensure connection without shorting or design rule violation. If long and high metal wires become a concern of the attacker due to potential easier detection, repeaters (buffers) can be added to break long wire into small sections. Adding repeaters also reduces loading on the trigger input wire so that impacts on timing of original design is minimized. Furthermore, it is also possible that the attacker can choose different trigger input wires and/or payload according to the existing layout of the target design.

Function	Drive Strength	Width†	AC Power†	Standby Power†
NAND2	X1	1	1	1
NAND2	X4	3	3.7	4.1
NAND2	X8	5.75	7.6	8.1
DFF with Async Set	X1	6.25	12.7	2.9
DFF with Async Set	X4	7.25	21.8	6.8
DFF with Async Reset	X1	6	12.7	2.6
DFF with Async Reset	X4	7.75	21.8	7.2
DFF with Async Set and Reset	X1	7.5	14.5	3.3
DFF with Async Set and Reset	X4	8.75	23.6	8.1
Trigger w/o IO device	**-**	**8**	**7.7**	**2.2**
Trigger w/ IO device	**-**	**13.5**	**0.08**	**0.08**

* DFF stands for D Flip Flop. † Normalized values

Table I: Comparison of area and power between our implemented analog trigger circuits and commercial standard cells in $65nm$ GP CMOS technology.

This is possible because the proposed attack can be used to build variants of system level attacks.

In our OR1200 implementation, finding free space to insert the charge pump is trivial, even with the design's 80% area utilization, because the charge pump is small and there is no timing requirement on the attack circuits, affording us the freedom to distribute our attack components over a wide area of the chip. In our implementation, the distance between trigger and victim flip-flop is in near the mean of all interconnects. Connecting our attack components does require some jogging and jumping for the connections, but this is a common routing technique in commercial settings, so the information leaked by such wires is limited.

In A2, we select the victim processor and we also synthesize the chip. This means that we can bridge the semantic gap between names (and by extension functionality) at the hardware description level and traces in the mask. This level of information is representative of what back-end design house attackers would have. We also expect that it is possible for a foundry-level attacker to implement A2. This is more difficulty because a foundry-level attacker only has access to the chip layout. To accomplish the attack, the attacker must be able to identify a victim wire and to identify the victim flip-flop. Viable victim wires must have a low baseline rate of activity (given the expected use of the processor) and be controllable by the attacker to have a high enough activity to fill the trigger's capacitor. We observe that for processors, the existence of such a wire is not an issue. For the attacker to identify the wire, they must convert the chip layout back in to a purely digital representation, i.e., the structural netlist. Fortunately, this is an existing part of the back-end house design process known as Physical Verification. Thus, a foundry-level attacker can also use such a tool to obtain a netlist of the chip suitable for digital simulation. Once an attacker can simulate a chip, finding a suitable victim wire is a matter of simulating the expected workload and possible attack triggers; this is how we found viable victims for A2. Identifying the desired victim flip-flop in the generated netlist is challenging due to the lack of meaningful names. For A2, we are able to identify the victim flip-flop in a netlist with no meaningful names by writing test cases that expose the flip-flop by making it change value at a series of specific clock cycles.

3) Side-channel information: For the attack to be stealthy and defeat existing protections, the area, power and timing overhead of the analog trigger circuit should be minimized. High accuracy SPICE simulation is used to characterize power and timing overhead of implemented trigger circuits. Comparisons with several variants of $NAND2$ and $Dflip-flop$ standard cells from commercial libraries are summarized in Table I. The area of the trigger circuit not using IO device is similar to a X4 strength $Dflip-flop$. Using an IO device increases trigger circuit size significantly, but area is still similar to the area of 2 standard cells, which ensures it can be inserted to empty space in final design layout. AC power is the total energy consumed by the circuits when input changes, the power numbers are simulated by doing SPICE simulation on a netlist with extracted parasitics from our chip layout. Standby power is the power consumption of the circuits when inputs are static and comes from leakage current of CMOS devices.

In A2, the analog trigger circuit is directly feeds off of the victim wire, which is the only part in the attack that creates a timing disturbance to the original design. Before and after inserting the A2, we extract parasitics from the layouts to do high accuracy simulation of the victim wire's delay. Results show that rising and falling delay before trigger insertion are $19.76ps$ and $17.18ps$ while those after trigger insertion are $20.66ps$ and $18.45ps$. Extra delay is $1.2ps$ on average, which is the timing overhead of the attack. $1.2ps$ is only 0.33% of $4ns$ clock period and well below the process variation and noise range. Besides, in practical measurement, $1.2ps$ is nearly impossible to measure. unless high resolution time to digital converter is included on chip, which is impractical

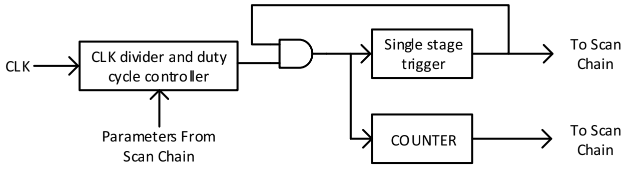

CLK
CLK divider and duty cycle controller
Parameters From Scan Chain
Single stage trigger
COUNTER
To Scan Chain
To Scan Chain

Figure 14: Testing structure to characterize the *trigger time* and *retention time* of implemented analog trigger circuits.

due to its large area and power overhead.

4) Comparison to digital-only attacks: If we look at a previously proposed, digital only and smallest implementation of a privilege escalation attack [20], it requires 25 gates and $80\mu m^2$ while our analog attack requires as little as one gate for the same effect. Our attack is also much more stealthy as it requires dozens of consecutive rare events, where the other attack only requires two. We also implement a digital only, counter-based attack that aims to mimic our A2. The digital version of A2 requires 91 cells and $382\mu m^2$, almost two orders-of-magnitude more than the analog counterpart. These results demonstrate how analog attacks can provide attackers the same power, control, and more stealthiness as existing digital attacks, but at a much lower cost.

5) Trigger characterization: To fully characterize the fabricated trigger circuit, a standalone testing structure as shown in Figure 14 is included in the test chip. A digital clock divider and duty cycle controller takes parameters from the scan chain to generate a simulated victim wire for the trigger. A feedback loop connected to an AND gate is used to stop the trigger input when the trigger output is activated. A counter counts the number of transitions of the trigger input. It stops when the trigger output is activated. The counter value is read out using the scan chain. Combining the count, clock frequency and clock divider ratio (i.e., the toggle rate of the victim wire), we can calculate the *trigger time*. After the trigger activates and victim wire stops toggling due to the AND gate, the capacitor voltage will slowly leak away until the trigger is deactivated. Once it is deactivated, the counter will restart. By taking readings fast, we can roughly measure the time interval between counter stops and restarts, which is the *retention time* of the trigger circuit.

V. Evaluation

We perform all experiments with our fabricated malicious OR1200 processor. We implement the processor using $65nm$ CMOS technology in an area of $2.1mm^2$. Figure 15 shows this processor, including where the different functional blocks are located within the processor. Figure 15 also shows where we add A2, with two levels of zoom to aide in understanding the challenge of identifying A2 in a sea of non-malicious logic. In fact, A2 occupies less than 0.08% of the chip's area. Our fabricated chip actually contains two sets of attacks: the first set of attacks are one and two-stage triggers baked-in to the processor that we use to assess the end-to-end impact of A2. The second set of attacks exist outside of the processor and are used to fully characterize A2's operation.

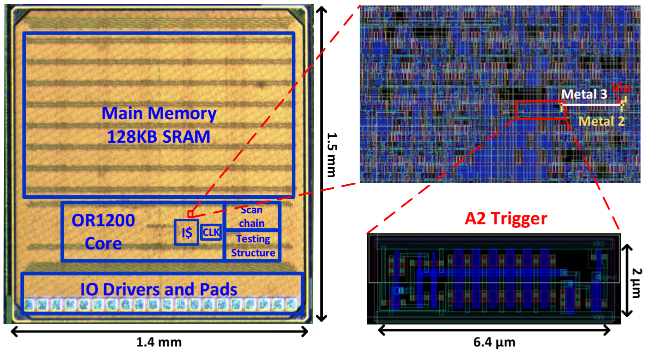


Figure 15: Die micrograph of analog malicious hardware test chip with a zoom-in layout of inserted A2 trigger.

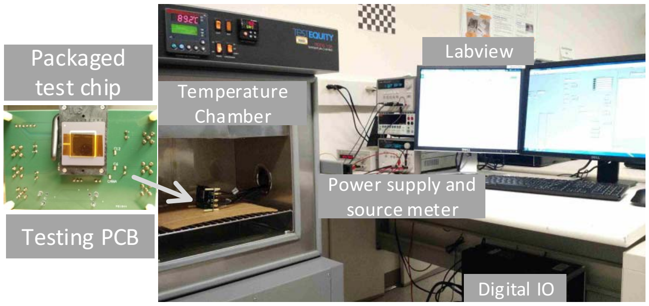
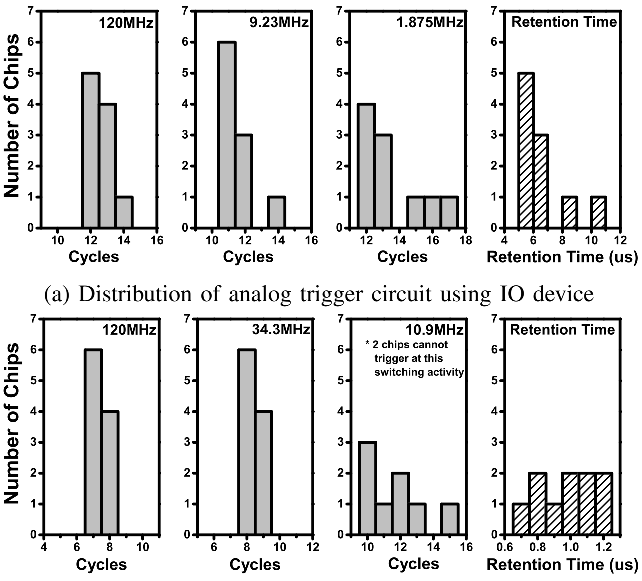


Figure 16: Testing setup for test chip measurement.

We use the testing setup shown in Figure 16 to evaluate our attacks' response to changing environmental conditions and a variety of software benchmarks. The chip is packaged and mounted on a custom testing PCB to interface with personal computer. We use the LabVIEW program to control a digital interface card that reads and writes from the chip through a custom scan chain interface. The scan chain interface enables us to load programs to the processor's memory and also to check the values of the processor's registers. The testing board is kept in a temperature chamber to evaluate our attacks under temperature variations. To clock the processor, we use an on-chip clock generator that generates a $240MHz$ clock at the nominal condition ($1V$ supply voltage and $25°C$). We use a programmable clock divider to convert the $240MHz$ clock into the desired clock frequency for a given experiment.



(a) Distribution of analog trigger circuit using IO device

(b) Distribution of analog trigger circuit using only core device

Figure 17: Measured distribution of retention time and trigger cycles under different trigger input divider ratios across 10 chips at nominal 1V supply voltage and $25°C$.

A. Does the Attack Work?

To prove the effectiveness of A2, we evaluate it from two perspectives. One is a system evaluation that explores the end-to-end behavior of our attack by loading attack-triggering programs on the processor, executing them in usermode, and verifying that after executing the trigger sequence, they have escalated privilege on the processor. The other perspective seeks to explore the behavior of our attacks by directly measuring the performance of the analog trigger circuit, the most important component in our attack, but also the most difficult aspect of our attack to verify using simulation. To evaluate the former, we use the in-processor attacks and for the later, we use the attacks implement outside the processor with taps directly connected to IO pins.

1) System attack: Malicious programs described in Section IV-A are loaded to the processor and then we check the target register values. In the program, we initialize the target registers $SR[0]$ (the mode bit) to user mode (i.e., 0) and $SR[1]$ (a free register bit that we can use to test the two-stage trigger) to 1. When the respective triggers deploys the attack, the single-stage attack will cause $SR[0]$ to suddenly have a 1 value, while the two-stage trigger will cause $SR[1]$ to have a 0 value—the opposite of their initial values. Because our attack relies on analog circuits, environmental aspects dictate the performance of our attack. Therefore, we test the chip at 6 temperatures from $-25°C$ to $100°C$ to evaluate the robustness of our attack. Measurement results confirm that both the one-stage and two-stage attacks in all 10 tested chips successfully overwrite the target registers at all temperatures.

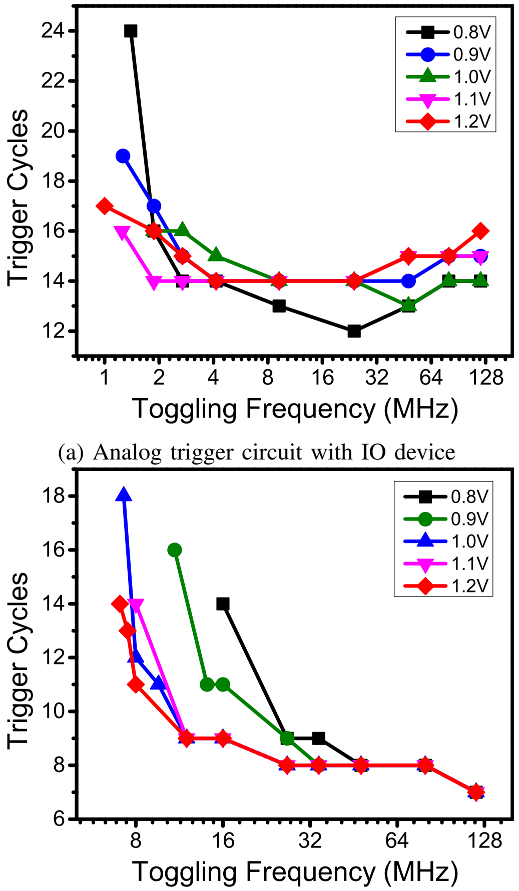
Trigger Circuit	Toggle Rate (MHz)	Measured (10 chip avg)	Simulated (Typical corner)
w/o IO device	120.00	7.4	7
w/o IO device	34.29	8.4	8
w/o IO device	10.91	11.6	10
w/ IO device	120.00	12.6	14
w/ IO device	9.23	11.6	13
w/ IO device	1.88	13.5	12

Table II: Comparison of how many cycles it takes to activate fully the trigger for our fabricated chip (Measured) and for HSPICE (Simulated) versions of our analog trigger circuit.

2) Analog trigger circuit measurement results: Using the standalone testing structure shown in Figure 14, *number of cycles until trigger* and *retention time* can be characterized. We use the $240MHz$ on-chip clock to simulate the toggling of a victim wire that feeds the trigger circuits under test. To show how our attack triggers respond to a range of victim activity levels, we systematically sweep clock division ratios which simulates a similar range of victim wire activities.

Figure 17 shows the measured distribution of *retention time* and trigger cycles at 3 different trigger toggle frequencies across 10 chips. The results show that our trigger circuits have a regular behavior in the presence of real-world manufacturing variances, confirming SPICE simulation results. *retention time* at the nominal condition ($1V$ supply voltage and $25°C$) is around $1\mu s$ for trigger with only core devices and $5\mu s$ for attacks constructed using IO devices. Compared to SPICE simulation results, in Figure 7 and Figure 13, trigger without IO devices has close results while trigger with IO device shows 4 times smaller retention time than simulations suggest. This is reasonable because gate leakage of IO devices is negligible in almost any designs and the SPICE model is a rough estimation. Table II provides the number of cycles until triggering for both trigger circuits (i.e., with and without IO devices) from fabricated chip measurements and SPICE simulations to validate the accuracy of simulation. An attacker wants the simulator to be as accurate as possible as the cost and time requirement of fabricating test chips make it impractical to design analog attacks without a reliable simulator. Fortunately, our results indicate that SPICE is capable at providing results of sufficient accuracy for these unusual circuits based on leakage currents.

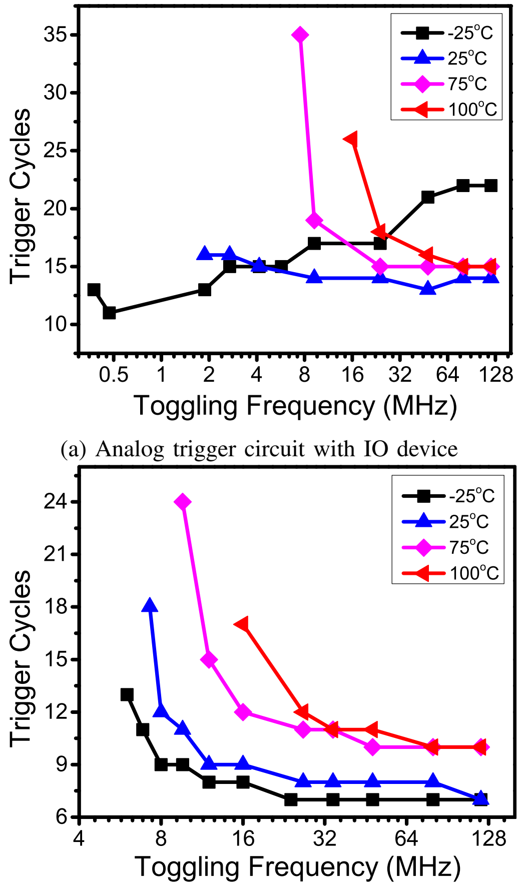
To verify the implemented trigger circuits are robust across voltage and temperature variations (as SPICE simulation suggests), we characterize each trigger circuit under different supply voltage and temperature conditions. Figure 18 and Figure 19 show how many cycles it takes (on average) for each trigger circuit to activate fully when the simulated victim wires toggles between $.46MHZ$ and



(a) Analog trigger circuit with IO device

(b) Analog trigger circuit with only core device

Figure 18: Measured trigger cycles under different input frequency at different supply voltages.



(a) Analog trigger circuit with IO device

(b) Analog trigger circuit with only core device

Figure 19: Measured trigger cycles under different input frequency at different ambient temperatures.

$120MHz$, when the supply voltage varies between $0.8V$ and $1.2V$, and when the ambient temperature varies between $-25°C$ and $100°C$.

As expected, different conditions yield different minimum toggling rates to activate the trigger. It can be seen that temperature has a stronger impact on our trigger circuit's performance because of leakage current's exponential dependence on temperature. At higher temperature, more cycles are required to trigger and higher switching activity is required because leakage from capacitor is larger. The exception to this happens with the trigger constructed using IO devices, at very low temperature. In this case, leakage currents are so small that the change in trigger cycles comes mainly from the setup time of Schmitt trigger, higher toggling inputs spend more cycles during the setup time. SPICE simulation predicts these results as well.

Lastly, once the trigger activates, it will only remain in the activated state for so long, barring continued toggling from the victim wire. The window of time that a trigger stays activated is critically important for series-connected multi-stage trigger circuits. This window is also controlled by manufacturing variances and environmental conditions. Variation of retention time across $-25°C$ to $100°C$ is plotted in Figure 20, which shows that the retention time of both trigger circuits is long enough to finish the attack across wide temperature range. Trigger circuits constructed with IO devices have a larger dependence on temperature because of different temperature dependencies for different types of devices. The variation of *cycles until triggering* and *retention time* across PVT variations implies the possibility that an attacker can include the environmental condition as part of the trigger. For example, a low activity trigger input can only trigger the attack at low temperatures according to the measurement results; great news if you want your attack to work in the North Pole, but not the tropics. Attackers can also tune the circuits towards stricter requirement to trigger so that the attack is never exposed at higher temperatures to further avoid detections.

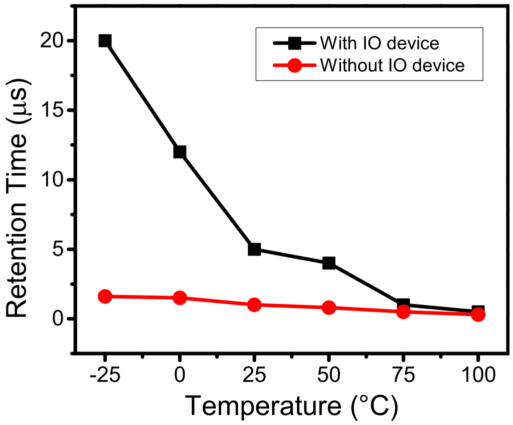


Figure 20: Measured retention time of analog trigger circuits across temperatures.

B. Is the Attack Triggered by Non-malicious Benchmarks?

Another important property for any hardware Trojan is not exposing itself under normal operations. Because A2's trigger circuit is only connected to the trigger input signal, digital simulation of the design is enough to acquire the activity of the signals. However, since we make use of analog characteristics to attack, analog effects should also be considered as potential effects to accidentally trigger the attack. Therefore, we ran 5 selected programs from the MiBench embedded systems benchmark suite. We select MiBench [17] because it targets the class of processor that best fits the OR1200 and it consists of a set of well-understood applications that are popular system performance benchmarks in both academia and in industry. MiBench consists of 31 applications, spread across 6 resource-usage-based classes. To validate that A2's trigger avoids spurious activations from a wide variety of software, we select 5 benchmark applications from MiBench, each from a different class. This ensures that we thoroughly test all subsystems of the processor—exposing likely activity rates for the wires in the processor. Again, in all programs, the victim registers are initialized to opposite states that A2 puts them in when its attack is deployed. The processor runs all 5 programs at 6 different temperatures from $-25°C$ to $100°C$. Results prove that neither the one-stage nor the two-stage trigger circuit is exposed when running these benchmarks across such wide temperature range.

C. Existing Protections

Existing protections against fabrication-time attacks are mostly based on side-channel information, e.g., power, temperature, and delay. In A2, we only add one gate in the trigger, thus minimizing power and temperature perturbations caused by the attack.

Table III summarizes the total power consumption measured when the processor runs our five benchmark programs, at the nominal condition ($1V$ supply voltage and $25°C$). A Keithley 2400 sourcemeter is used to measure the power consumption of the processor, which can measure down to $1\mu A$ in our measurement range. All the values in Table III are average values across the entire program execution. The variation of power consumption in all cases are limited to $\pm 3\mu W$. Direct measurement of trigger circuit power is infeasible in our setup, so simulation is used as an estimation. It was shown earlier that SPICE model matches measurement results in terms of trigger performance. Simulated trigger power consumption in Table I translates to $5.3nW$ and $0.5\mu W$ for trigger circuits constructed with and without IO devices. These numbers are based on the assumption that trigger inputs keep toggling at 1/4 of the clock frequency of $240MHz$, which is the maximum switching activity that our attack program can achieve on the selected victim wire. In the common case of non-attacking software, the switching activity is much lower—approaching zero—and only lasts a few cycles so that the extra power due to our trigger circuit is even smaller. In our experiments, the power of the attack circuit is orders-of-magnitude less than the normal power fluctuations that occur in a processor while it executes different instructions.

Besides side-channel information leaked by attack circuit itself, parasitic impacts of attack circuits on original design should also be considered. Adding new transistors around existing ones introduces negligible differences to the existing devices, because manufacturing steps like doping, lithography, and planarization are well controlled in modern CMOS IC manufacturing through the use of dummy doping/poly/metal fill. This fill maintains a high density of materials over large windows. The tiny inserted A2 trigger will not significantly change the overall density in a window and therefore does not cause systematic device variations. Besides, isolation between transistors avoids their coupling.

Coupling between malicious and original wires may cause cross-talk and more delay to the original wires. However, in CMOS manufacturing, the metal fill step adds floating metal pieces to empty spaces in the chip layout so that the unit parasitic capacitance of all wires are similar. An attacker can limit cross-talk effects through careful routing to avoid long parallel wires.

VI. Discussion

Now that we know A2 is an effective and stealthy fabrication-time attack, we look forward to possible defenses, including a discussion of the impact of split manufacturing and 3D-IC on our attacks. Before delving into defending against A2, we qualitatively address the challenge of implementing an A2-like attack in x86 processors.

A. Extending A2 to x86

We implement A2 on the OR1200 processor because it is open source. While the OR1200 processor is capable enough

Program	Power (mW)
Standby	6.210
Basic math	23.703
Dijkstra	16.550
FFT	18.120
SHA	18.032
Search	21.960
Single-stage Attack	19.505
Two-stage Attack	22.575
Unsigned Division	23.206

Table III: Power consumption of our test Chip running a variety of benchmark programs.

to run Linux, its complexity is closer to a mid-range ARM part, far below that of modern x86 processors from Intel and AMD. A natural question is if our attack technique applies to x86 processors and, if so, how does the attack's complexity scale with processor complexity.

We expect a A2-like attack in x86 processors to be much harder to detect and easier to implement than its OR1200 counterpart. While there are more viable victim registers in x86, A2 still only needs to target a single register to be effective. Also, A2's overhead comes primarily from its trigger circuit, but the complexity of the trigger is much more dependent on how "hidden" the attacker wants the attack to be than on the complexity of the processor. In fact, we expect that there are far more viable victim wires (highly-variable and controllable activity) due to the internal structure of complex, out-of-order processors like the x86. The only aspect of scaling to an x86-class processor that we anticipate as a challenge is maintaining controllability as there are many redundant functional units inside an x86, so a trigger would either need to tap equivalent wires in all functional units or be open to some probabilistic effects.

B. Possible Defenses

There are a few properties that make our attacks hard to detect: 1) we require adding as little as a single gate 2) our attack has a sophisticated trigger and 3) our trigger works in the analog domain, gradually building charge until it finally impacts the digital domain. Given these properties, defenses that measure side-channel information (e.g., current and temperature) have little hope of detecting the impact of a single gate in a sea of 100,000 gates. The same holds true for defenses that rely on visual inspection. Even if a defender were to delayer a chip and image it with a scanning electron microscope, our malicious gate is almost identical to all the other gates in a design. One option might be to focus the search on the area of the chip near security-critical state holding flip-flops.

If it is impractical to expect defenders to visually identify our attacks or to be able to detect them through measuring current or temperature, what about testing? One of the novel features of A2 is the trigger. In our implementation (Section IV), we carefully design the trigger to make it extremely unlikely for unknowing software—including validation tests—to trigger the attack. In fact, we built a trigger so immune to unintended activations that we had to employ sleds of inline assembly to get an activity ratio high enough to trigger our attack. This indicates that anything short of comprehensive testing is unlikely to expose the trigger [4].

Given that post-fabrication testing is unlikely to expose our attack and our attack's impact on known side-channels is buried within the noise of a circuit, we believe that a new type of defense is required: we believe that the best method for detecting our attack is some form of runtime verification that monitors a chip's behavior in the digital domain.

C. Split Manufacturing

One promising future defense to malicious circuits inserted during fabrication is split manufacturing [22]–[25] and 3D-IC [26]. The idea behind defenses incorporating split manufacturing is to divide a chip into two parts, with one part being fabricated by a cheap, but untrusted, fabrication house, while the other part gets fabricated by an expensive, but trusted, fabrication house (that is also responsible for putting the two parts together in a single chip). The challenge is determining how to divide the chip into two parts.

One common method is to divide the chip into gates and wires [26]. The idea behind this strategy is that by only moving a subset of wires to the trusted portion of the chip, it will be cheaper to fabricate at the trusted fabrication house, while serving to obfuscate the functionality from the untrusted fabrication house. This obfuscation makes it difficult for an attacker to determine which gates and wires (of the ones they have access to) to corrupt.

From this description, it might seem as if current split manufacturing-based defenses are a viable approach to stopping A2. This is not the case as A2 changes the state in a flip-flop, but only wires are sent to the trusted fabrication house. Future split manufacturing approaches could move a subset of flip-flops to the trusted part of the chip along with a subset of wires, but that increases the cost of an already prohibitively expensive defense. Additionally, recent research shows that even when a subset of wires are missing, it is possible to reverse engineer up to 96% of the missing wires using knowledge of the algorithms used in floor-planning, placement, and layout tools [22]. In fact, we already take advantage of some of this information in identifying the victim wire that drives our trigger circuit and in identifying the victim flip-flop.

[4] Even if test cases somehow activated our attack, the onus is on the testing routines to catch our malicious state change. Observe that most non-malicious software runs the same regardless of privilege level.

Previous works [23], [24] also proposed splitting manufacturing at low-level metal layers, even down to lowest metal layer. Splitting at metal 1 is a potentially effective method to defend against A2 attack if carried out by untrusted manufacturer. However, this approach introduces an extremely challenging manufacturing problem due to the small dimension of low-level metal layers and tremendous amount of connections to make between two parts, not to mention the expense to develop a trusted fabrication house with such capabilities. There has been no fabricated chips demonstrating that such a scheme works given the constraints of existing manufacturers.

VII. Related Work

A2 is a fabrication-time attack. There is almost 10 years of previous work on fabrication-time attacks and defenses. In this section, we document the progression of fabrication-time attacks from 100-gate circuits targeted at single-function cryptographic chips, aimed at leaking encryption keys to attacks that work by corrupting existing gates aimed at more general functionality. The historical progression of fabrication-time attacks highlights the need for a small, triggered, and general-purpose attack like A2.

Likewise, for fabrication-time defenses, we document the progression of defenses from complete chip characterization with a heavy reliance on a golden reference chip to defenses that employ self-referencing techniques and advance signal recovery (removing the requirement of a golden chip). We conclude with defenses that move beyond side-channels, into the real of on-chip sensors aimed at detecting anomalous perturbations in circuit performance presumably due to malicious circuits. The historical progression of fabrication-time attack defenses shows that while they may be effective against some known attacks, there is a need for a new type of defense that operates with more semantic information.

A. Fabrication-time Attacks

The first fabrication-time hardware attack was the addition of 100 gates to an AES cryptographic circuit aimed at creating a side-channel that slowly leaks the private key [27]. The attack circuit works by modulating its switching activity (i.e., increasing or decreasing the attack's current consumption) in a way that encodes the secret key on the current consumed by the chip as a whole. This method of attack has four disadvantages: 1) the attack has limited scope 2) the attacker must have physical access to the victim device and 3) the attack is always-on, making it more detectable and uncontrollable. To mute their attack's effect on the circuit, the authors employ a spread-spectrum technique to encode single bits of the key on many clock cycles worth of the power trace of the device. This technique helps conceal the attack from known, side-channel based, fabrication-time defenses at the cost of increased key recovery time.

Another fabrication-time method for creating malicious circuits is to modify the fabrication process so that natural process variation is shifted outside the specified tolerances. Process reliability Trojans [28] show how an attacker can cause reductions in reliability by accelerating the wearing out mechanisms for CMOS transistors, such as Negative Bias Temperature Instability (NBTI) or Hot Carrier Injection (HCI). Process reliability Trojans affect an entire chip and affect some chips more than others (the effect is randomly distributed the same way as process variation); the goal is to cause the entire chip to fail early. While the paper does not implement a process Trojan, the authors explore the knobs available for implementing a process reliability Trojan and discuss the theory behind them. The value of this attack is that it is very difficult to detect as a defender would have to destructively measure many chips to reverse-engineer the fabrication parameters. A2 represents a different design point: a targeted attack that is controllable by a remote attacker.

A targeted version of a process reliability Trojan is the dopant-level Trojan [4]. Instead of adding additional circuitry to the chip (e.g., the side-channel Trojan) or changing the properties of the entire chip (e.g., the process reliability Trojan), dopant-level Trojans change the behavior of existing circuits by tying the inputs of logic gates to logic level 0 or logic level 1. By carefully selecting the logic value and the gates attacked, it is possible to mutate arbitrary circuits into a malicious circuit. This approach is incredibly stealthy because there are no extra gates or wires, but comes with limitations. First, while there are no extra gates or wires added for the attack, more recent work shows that removing additional layers (down to the contact layers) of the chip reveals the added connections to logic 0 and logic 1 [6]. Note that removing these extra layers and imaging the lower layers is estimated to be 16-times more expensive that stopping at the metal layers. A second limitation is that the attacker can only modify existing circuits to implement their attack. This makes it difficult to construct attack triggers resulting in an exposed attack payload—making detection more likely. Recent defenses seek to prevent dopant-level attacks by obfuscating the circuit and using split manufacturing [26]. A2 trades-off some detectability in the metal layers of the chip for less detectability by testing. The observation driving this is that every chip has its functionality tested after fabrication, but it is prohibitively expensive to delayer a chip and image it with a scanning electron microscope. By using analog circuits, A2 makes it possible to implement complex attack triggers with minimal perturbations to the original circuit.

The most recent fabrication-time attack is the parametric Trojans for fault injection [5]. Parametric Trojans build on dopant-level Trojan by adding some amount of controllability to the attack. Parametric Trojans rely on power supply voltage fluctuations as a trigger. For example, imagine a dopant-level attack that only drives the input of a logic gate to 1 or 0 when there is a dip in the supply voltage. Because this requires that the attacker has access to the power supply of a device, the goal is to facilitate fault-injection attacks (e.g., erroneous result leaks part of the key as in RSA attacks [29]).

B. Fabrication-time Defenses

There are three fundamental approaches to defend against fabrication-time malicious circuits: 1) side-channel-based characterization 2) adding on-chip sensors and 3) architectural defenses. This section covers example defenses that use each approach and qualitatively analyze how A2 fares against them.

1) Side-channels and chip characterization: IC fingerprinting [7] is the first attempt to detect malicious circuits added during chip fabrication. IC fingerprinting uses side-channel information such as power, temperature, and electromagnetic measurements to model the run time behavior of a golden (i.e., trusted) chip. To clear untrusted chips of possible malice, the same inputs are run on the suspect chip and the same side-channel measurements collected. The two sets of measurements are then compared, with a difference above a noise threshold causing the chip to be labeled as malicious. The more golden chips available, the better the noise modeling. IC fingerprinting works well when there are a set of trusted chips, the chip's logic is purely combinational, and it is possible to exercise the chip with all possible inputs. The authors also point out that their approach requires that Trojans be at least .01% of the circuit; in A2 the Trojan is an order of magnitude smaller than that—not to mention that we attack a processor.

Another side-channel-based approach is to create a path delay fingerprint [8]. This is very similar to IC fingerprinting, except with a heavier reliance on the chip being purely combinational. To create a path delay fingerprint, testers exercise the chip with all possible test cases, recording the input-to-output time. The observation is that only malicious chips will have a path delay outside of some range (the range depends on environmental and manufacturing variances). Even if it is possible to extend this approach to sequential circuits and to meaningfully train the classifier where comprehensive testing is impractical, A2 minimizes the impacts on the delay of the surrounding circuit to hide into environmental variation and noise (Section IV-B3) and the attack modifies state directly.

Building from the previous two defenses is gate-level characterization [9]. Gate-level characterization is a technique that aims to derive characteristics of the gates in a chip in terms of current, switching activity, and delay. Being a multi-dimensional problem, the authors utilize linear programming to solve a system of equations created using non-destructive measurements of several side-channels. A2 evades this defense because it operates in the analog domain.

Electromagnetic fingerprinting combined with statistical analysis provides a easier approach to measure local side-channel information from small parts of a chip and suppress environmental impacts [30]. Because EM radiation from A2 only occurs when the attack is triggered, it evades defenses that assume EM signals are different in attacked designs even if the Trojan is dormant.

One major limitation of characterization-based defenses is the reliance on a golden reference chip. TeSR [10] seeks to replace a golden chip with self-referencing comparisons. TeSR avoids the requirement of a golden chip by comparing a chip's transient current signature with itself, but across different time windows. Besides eliminating the need for a golden chip, TeSR also enables side-channel techniques to apply to more complex, sequential circuits. Unfortunately, TeSR requires finding test cases that activate the malicious circuit to be able to detect it. While TeSR may work well against dopant-level Trojans, we include a complex trigger in A2 that avoids accidental activations. Additionally, results in Section IV suggest that the assumption underlying TeSR—that malicious and non-malicious side-channel measurements are separable—is not true for A2-like attacks.

2) Adding on-chip sensors: As mentioned, using side-channel information to characterize chip delay is limited to combinational circuits. One defense suggests measuring delay locally through the addition of on-chip sensors [11]. The proposed technique is able to measure precisely the delay of a group of combinational paths—these paths could be between registers in a sequential circuit. Much like in the side-channel version, the sensors attempt to characterize the delay of the monitored paths and detect delays outside an acceptable range as potential malice. The increased accuracy and control over the side-channel version comes at the cost of added hardware: requires the addition of a shadow register for every monitored combinational path in the chip and a shadow clock that is a phase offset version of the main clock. A comparator compares the main register and the shadow register, with a difference indicating that the combinational delay feeding the main register has violated its setup requirement. This approach is similar to Razor [31], but here the phase shift of the shadow clock is gradually adjusted to expose changes in delay. A2 avoids this defense because it modifies processor state directly, not affecting combinational delays.

Adding to the list of tell tale features is Temperature Tracking [12]. Temperature Tracking uses on-chip temperature sensors to look for temperature spikes. The intuition is that when malicious hardware activates, it will do so with an unusually high (and moderate duration) burst of activity. The activity will increase current consumption, that then produces temperature increases. Unfortunately, results from Section V show that this intuition is invalid for our malicious processor. A2 is a single gate in a sea of 100,000 gates, so its current consumption is muted. Also, A2's trigger gradually builds charge and the payload lasts for a very short duration not able to be capture at the slow rate of thermal variation. In general, it is possible for other attackers to hide their attacks from this approach by placing their malicious circuits in an active area of the chip, or by making their attack infrequently active and active for short durations.

The most recent on-chip sensor proposal targeted at detection malicious circuits added during fabrications hearkens back to IC fingerprinting in that the goal is to monitor the power rails of the chip [13]. The authors propose adding power supply monitoring sensors that detect fluctuations in a supply's characteristic frequencies. As has been noted with previous approaches, our results show that there are cases where there is *no* difference in power supply activity between the case where the malicious circuit is active versus inactive.

A2 defeats defenses that rely on characterizing device behavior through power, temperature, and delay measurements by requiring as few as one additional gate and by having a trigger that does not create or destroy charge, but redirects small amounts of charge. In addition, A2's analog behavior means that *cycle-to-cycle changes are small, eventually accumulating to a meaningful digital change*.

3) Eliminating unused space: BISA [32] is a promising defense against fabrication-time attacks that seeks to prevent attackers from adding components to a design by eliminating all empty space that could be used to to insert attack logic. A perfect deployment of BISA does indeed make implementing A2 more challenging. Unfortunately, the small area of A2 presents a challenging problem to any BISA implementation, because *all* empty space must be filled by BISA cells with no redundant logic or buffers—as an attacker can replace these with their attack circuit and the behavior of the design remains. Also, a perfect BISA implementation requires 100% test coverage—an impractical requirement, otherwise an attacker can replace logic not covered in the tests. In addition, implementing BISA significantly reduces routing space of the original design and prevents designers from doing iterative place and route. Limiting designers in this way results in performance degradation and possibly an unroutable design. All of these requirements dramatically increase the cost of chip fabrication and time-to-market.

VIII. Conclusion

Experimental results with our fabricated malicious processor show that a new style of fabrication-time attack is possible; a fabrication-time attack that applies to a wide range of hardware, spans the digital and analog domains, and affords control to a remote attacker. Experimental results also show that A2 is effective at reducing the security of existing software, enabling unprivileged software full control over the processor. Finally, the experimental results demonstrate the elusive nature of A2: 1) A2 is as small as a single gate—two orders of magnitude smaller than a digital-only equivalent 2) attackers can add A2 to an existing circuit layout without perturbing the rest of the circuit 3) a diverse set of benchmarks fail to activate A2 and 4) A2 has little impact on circuit power, frequency, or delay.

Our results expose two weaknesses in current malicious hardware defenses. First, existing defenses analyze the digital behavior of a circuit using functional simulation or the analog behavior of a circuit using circuit simulation. Functional simulation is unable to capture the analog properties of an attack, while it is impractical to simulate an entire processor for thousands of clock cycles in a circuit simulator—this is why we had to fabricate A2 to verify that it worked. Second, the minimal impact on the run-time properties of a circuit (e.g., power, temperature, and delay) due to A2 suggests that it is an extremely challenging task for side-channel analysis techniques to detect this new class of attacks. We believe that our results motivate a different type of defense; a defense where trusted circuits monitor the execution of untrusted circuits, looking for out-of-specification behavior in the digital domain.

Acknowledgment

We thank our shepherd Eran Tromer for his guidance and the anonymous reviewers for their feedback and suggestions. This work was supported in part by C-FAR, one of the six SRC STARnet Centers, sponsored by MARCO and DARPA. This work was also partially funded by the National Science Foundation. Any opinions, findings, conclusions, and recommendations expressed in this paper are solely those of the authors.

References

[1] M.-L. Li, P. Ramachandran, S. K. Sahoo, S. V. Adve, V. S. Adve, and Y. Zhou, "Understanding the Propagation of Hard Errors to Software and Implications for Resilient System Design," in *International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS. Seattle, WA: ACM, Mar. 2008, pp. 265–276.

[2] M. Hicks, C. Sturton, S. T. King, and J. M. Smith, "Specs: A lightweight runtime mechanism for protecting software from security-critical processor bugs," in *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS. Istanbul, Turkey: ACM, 2015, pp. 517–529.

[3] S. S. Technology. (2012, Oct.) Why node shrinks are no longer offsetting equipment costs. [Online]. Available: http://electroiq.com/blog/2012/10/why-node-shrinks-are-no-longer-offsetting-equipment-costs/

[4] G. T. Becker, F. Regazzoni, C. Paar, and W. P. Burleson, "Stealthy Dopant-level Hardware Trojans," in *International Conference on Cryptographic Hardware and Embedded Systems*, ser. CHES. Berlin, Heidelberg: Springer-Verlag, 2013, pp. 197–214.

[5] R. Kumar, P. Jovanovic, W. Burleson, and I. Polian, "Parametric Trojans for Fault-Injection Attacks on Cryptographic Hardware," in *Workshop on Fault Diagnosis and Tolerance in Cryptography*, ser. FDT, 2014, pp. 18–28.

[6] T. Sugawara, D. Suzuki, R. Fujii, S. Tawa, R. Hori, M. Shiozaki, and T. Fujino, "Reversing Stealthy Dopant-Level Circuits," in *International Conference on Cryptographic Hardware and Embedded Systems*, ser. CHES. New York, NY: Springer-Verlag, 2014, pp. 112–126.

[7] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi, and B. Sunar, "Trojan Detection Using IC Fingerprinting," in *Symposium on Security and Privacy*, ser. S&P. Washington, DC: IEEE Computer Society, 2007, pp. 296–310.

[8] Y. Jin and Y. Makris, "Hardware Trojan Detection Using Path Delay Fingerprint," in *Hardware-Oriented Security and Trust*, ser. HOST. Washington, DC: IEEE Computer Society, 2008, pp. 51–57.

[9] M. Potkonjak, A. Nahapetian, M. Nelson, and T. Massey, "Hardware Trojan horse detection using gate-level characterization," in *Design Automation Conference*, ser. DAC, vol. 46, 2009, pp. 688–693.

[10] S. Narasimhan, X. Wang, D. Du, R. S. Chakraborty, and S. Bhunia, "TeSR: A Robust Temporal Self-Referencing Approach for Hardware Trojan Detection," in *Hardware-Oriented Security and Trust*, ser. HOST. San Diego, CA: IEEE Computer Society, Jun. 2011, pp. 71–74.

[11] J. Li and J. Lach, "At-speed Delay Characterization for IC Authentication and Trojan Horse Detection," in *Hardware-Oriented Security and Trust*, ser. HOST. Washington, DC: IEEE Computer Society, 2008, pp. 8–14.

[12] D. Forte, C. Bao, and A. Srivastava, "Temperature Tracking: An Innovative Run-time Approach for Hardware Trojan Detection," in *International Conference on Computer-Aided Design*, ser. ICCAD. IEEE, 2013, pp. 532–539.

[13] S. Kelly, X. Zhang, M. Tehranipoor, and A. Ferraiuolo, "Detecting Hardware Trojans Using On-chip Sensors in an ASIC Design," *Journal of Electronic Testing*, vol. 31, no. 1, pp. 11–26, Feb. 2015.

[14] A. Waksman and S. Sethumadhavan, "Silencing Hardware Backdoors," in *IEEE Security and Privacy*, ser. S&P. Oakland, CA: IEEE Computer Society, May 2011.

[15] X. Wang, S. Narasimhan, A. Krishna, T. Mal-Sarkar, and S. Bhunia, "Sequential hardware trojan: Side-channel aware design and placement," in *Computer Design (ICCD), 2011 IEEE 29th International Conference on*, Oct 2011, pp. 297–300.

[16] OpenCores.org. OpenRISC OR1200 processor. [Online]. Available: http://opencores.org/or1k/OR1200_OpenRISC_Processor

[17] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, "MiBench: A free, commercially representative embedded benchmark suite," in *Workshop on Workload Characterization*. Washington, DC: IEEE Computer Society, 2001, pp. 3–14.

[18] K. Yang, M. Hicks, Q. Dong, T. Austin, and D. Sylvester, "A2: Analog malicious hardware," https://github.com/impedimentToProgress/A2, 2016.

[19] M. Rostami, F. Koushanfar, J. Rajendran, and R. Karri, "Hardware security: Threat models and metrics," in *Proceedings of the International Conference on Computer-Aided Design*, ser. ICCAD '13. San Jose, CA: IEEE Press, 2013, pp. 819–823.

[20] M. Hicks, M. Finnicum, S. T. King, M. M. K. Martin, and J. M. Smith, "Overcoming an Untrusted Computing Base: Detecting and Removing Malicious Hardware Automatically," *USENIX ;login*, vol. 35, no. 6, pp. 31–41, Dec. 2010.

[21] S. T. King, J. Tucek, A. Cozzie, C. Grier, W. n. Jiang, and Y. Zhou, "Designing and implementing malicious hardware," in *Workshop on Large-Scale Exploits and Emergent Threats*, ser. LEET, vol. 1, Apr. 2008.

[22] J. Rajendran, O. Sinanoglu, and R. Karri, "Is split manufacturing secure?" in *Design, Automation and Test in Europe*, ser. DATE, 2013, pp. 1259–1264.

[23] K. Vaidyanathan, B. Das, and L. Pileggi, "Detecting reliability attacks during split fabrication using test-only BEOL stack," in *Design Automation Conference*, ser. DAC, vol. 51. IEEE, Jun. 2014, pp. 1–6.

[24] K. Vaidyanathan, B. Das, E. Sumbul, R. Liu, and L. Pileggi, "Building trusted ICs using split fabrication," in *International Symposium on Hardware-Oriented Security and Trust*, ser. HOST. IEEE Computer Society, 2014, pp. 1–6.

[25] K. Vaidyanathan, R. Liu, E. Sumbul, Q. Zhu, F. Franchetti, and L. Pileggi, "Efficient and secure intellectual property (IP) design with split fabrication," in *International Symposium on Hardware-Oriented Security and Trust*, ser. HOST. IEEE Computer Society, 2014, pp. 13–18.

[26] F. Imeson, A. Emtenan, S. Garg, and M. V. Tripunitara, "Securing Computer Hardware Using 3d Integrated Circuit (IC) Technology and Split Manufacturing for Obfuscation," in *Conference on Security*, ser. Security. USENIX Association, 2013, pp. 495–510.

[27] L. Lin, M. Kasper, T. Gneysu, C. Paar, and W. Burleson, "Trojan Side-Channels: Lightweight Hardware Trojans Through Side-Channel Engineering," in *International Workshop on Cryptographic Hardware and Embedded Systems*, ser. CHES, vol. 11. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 382–395.

[28] Y. Shiyanovskii, F. Wolff, A. Rajendran, C. Papachristou, D. Weyer, and W. Clay, "Process reliability based trojans through NBTI and HCI effects," in *Conference on Adaptive Hardware and Systems*, ser. AHS, 2010, pp. 215–222.

[29] A. Pellegrini, V. Bertacco, and T. Austin, "Fault-based attack of rsa authentication," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2010*, March 2010, pp. 855–860.

[30] J. Balasch, B. Gierlichs, and I. Verbauwhede, "Electromagnetic circuit fingerprints for hardware trojan detection," in *Electromagnetic Compatibility (EMC), 2015 IEEE International Symposium on*, Aug 2015, pp. 246–251.

[31] D. Ernst, N. S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge, "Razor: a low-power pipeline based on circuit-level timing speculation," in *Microarchitecture, 2003. MICRO-36. Proceedings. 36th Annual IEEE/ACM International Symposium on*, Dec 2003, pp. 7–18.

[32] K. Xiao, D. Forte, and M. Tehranipoor, "A novel built-in self-authentication technique to prevent inserting hardware trojans," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 33, no. 12, pp. 1778–1791, Dec 2014.