This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2018.2864246, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

On-Chip Analog Trojan Detection Framework for Microprocessor Trustworthiness

Yumin Hou, Hu He, Kaveh Shamsi *Student Member, IEEE*, Yier Jin *Member, IEEE*, Dong Wu, Huaqiang Wu *Senior Member, IEEE*

***Abstract*—With the globalization of semiconductor industry, hardware security issues have been gaining increasing attention. Among all hardware security threats, the insertion of hardware Trojans is one of the main concerns. Meanwhile, many current Trojan detection solutions follow the assumption that the hardware Trojan itself should be composed of digital logic. This assumption is invalidated by recently proposed analog Trojans which are extremely small and can detect rare events. This paper proposes a runtime hardware Trojan detection method which is geared towards detecting such advanced Trojans. The principle of this method is to guard a set of concerned signals, and initiate a hardware interrupt request when abnormal toggling events occur in these guarded signals. To prove the effectiveness of this method, we design a processor based on ARMv7-A&R ISA, and insert an analog Trojan into the processor. We fabricated the design in an SMIC 130 nm process and demonstrate the effectiveness of the proposed methodology.**

I. Introduction

The globalization of semiconductor industry brings critical security, integrity, and privacy concerns. The globalization of the integrated circuit (IC) supply chain makes it difficult and costly for regulations only to maintain the integrity of the IC design through the design and fabrication process. This is especially true for the case of third party components [1].

Among all hardware security threats including reverse engineering, IP piracy, etc., the insertion of malicious logic (aka hardware Trojans) is still one of the main concerns. Upon this challenge, researchers from the government, industry and academia have proposed various techniques to help identify malicious logic both pre-fabrication, on RTL, netlist, and layout levels, as well as post-fabrication on manufactured circuits, using a combination of special testing pattern generation techniques and side-channel analysis.

Given that most of the hardware Trojans target digital circuits, almost all Trojan designing methods and detection solutions follow the same assumption that hardware Trojan itself should be composed of digital logic. This assumption largely limits the capabilities of many hardware Trojan detection method in detecting digital Trojans only. Another fundamental limitation of all the detection schemes is that the effect of a small enough Trojan on the logic and on the side-channel fingerprints of the circuit, can be masked by process variation and noise. This is exacerbated by the ever-increasing scale of integration and process-variation in advanced nodes. Recently proposed analog and RF Trojans [2] fall into this category. An analog Trojan circuit can detect an extremely rare sequential event with just a handful of transistors added to the circuit. This hurdles even invasive detection techniques and poses a real threat to the IC supply chain despite a decade of research in the area.

To guarantee the trustworthiness of a microprocessor, it is desired for designers to consider the existence of such analog Trojans, and build a corresponding detection framework in the processor design phase. In this paper, we present a novel on-chip hardware Trojan detection mechanism called R2D2 geared towards such analog Trojans. Since such analog Trojans are triggered by high frequency wire-flops in the processor, we propose an abnormal-toggling detection scheme that can easily be integrated into the processor with low overhead. We also discuss why it is difficult to remove it from the design. The main contributions of the paper are listed as follows:

- We develop an on-chip Trojan detection method. This method targets hardware Trojans triggered by a successive toggling events. We present an in-depth security analysis of the scheme;
- We design a processor based on the ARMv7-A&R ISA, and insert an analog Trojan (based on the A2 Trojan [2]), into the ARM processor. We explore the architecture of the processor and present various novel ways to integrate the Trojan and its payload;
- We provide simulation results, which demonstrate that the analog Trojan works on the ARM processor, and the R2D2 method is effective in detecting the analog Trojan;
- We also provide test results based on the taped-out chip. We fabricate a proof-of-concept chip in the SMIC 130 nm Mixed-Signal 1P7M process with the hardware Trojan and the detection mechanism. A testing platform is constructed to test the fabricated SoC. On-board test results are collected and presented.

The remainder of the paper is organized as follows: Section II presents comparison to related works. Section III presents the R2D2 detection scheme. Section IV provides an overview of the implemented processor. Section V presents simulation results and on-board test results, and Section VI concludes the paper.

This work is supported by the National Natural Science Foundation of China under Grant No. 61502032, and by Tsinghua and Samsung Joint Laboratory. The corresponding author is Hu He.

Y. Hou, H. He, D. Wu and H. Wu are with the Institute of Microelectronics, Tsinghua University, Beijing, 100084, China (email: hou-ym12@mails.tsinghua.edu.cn, hehu@tsinghua.edu.cn, dongwu@tsinghua.edu.cn, wuhq@tsinghua.edu.cn).

K. Shamsi and Y. Jin are with the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL, USA, 32611 (email: kshamsi@ufl.edu, yier.jin@ece.ufl.edu).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2018.2864246, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

II. Related Works

Hardware Trojans are typically categorized according to a) their triggering method e.g. sequential or combinational, b) their payload e.g. active modification of logic values, or passive leakage of information through side-channels [3]. Post-fabrication detection mechanisms fall into testing-based [4], [5], or side-channel-based [6], [7]. Various statistical methods have been used to extract the side-channel traces of a Trojan in a sea of other components. In addition, design time techniques can also accompany post-fabrication detection such as reducing rare-events in the circuit [8], or inserting on-chip sensors that will be measured post-fabrication for Trojan detection [9].

A. Analog Hardware Trojans

Sequentially triggered Trojans can be made extremely difficult to detect through testing patterns. A digital sequential Trojan typically requires a large FSM to detect a rare sequence of events, which in turn can reduce the Trojans resiliency to side-channel detection techniques. Analog switch-capacitor circuits on the other hand can be used to do signal shaping and detection with a much lower transistor count. The analog Trojan presented in [2] known as A2 uses a simple analog circuit to detect high frequency toggling.

A2 is a small analog circuit, which can be inserted into an already placed and routed design. It reads a digital pulse signal (the trigger input), and triggers the payload when the pulse signal has toggled with a high frequently for a certain period of time. The trigger input is connected to a signal that can be toggled with high frequency through a special code snipper running on the processor. The attacker insures that the trigger signal has a much lower toggle rate during typical workloads. This makes detecting the hardware Trojan difficult through testing, not to mention that there exists many low toggling frequency bits in modern processors.

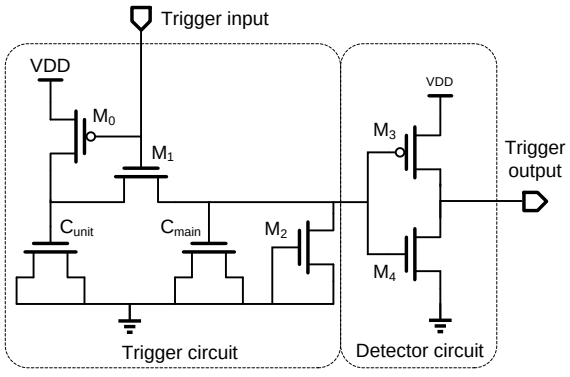


Figure 1: Transistor schematic of the analog Trojan circuit from [2]

The transistor schematic of the A2 analog Trojan [2] circuit is shown in Figure 1. When the trigger input is low, C_{unit} is charged to VDD. When the trigger input switches to high, C_{unit} shares its charge with C_{main}. This will raise the charge on C_{main} by an amount controllable by the size ratio of C_{main} and C_{unit}. When the trigger input is stationary at either high or low, the charge at C_{main} dissipates through M_2 and other stray-paths. Hence, only with sufficiently frequent toggling of the trigger input, one can raise C_{main}'s voltage. This voltage is fed to a detector circuit which is an imbalanced inverter with a controllable switching threshold. The attacker connects the trigger input to a software controllable bit which has a low toggling rate, and uses the trigger output to launch the payload.

B. Runtime Hardware Trojan Detection Methods

Off-line Trojan detection solutions, such as [10]–[12], are difficult to expose advanced Trojans like A2 because of its insertion stage and software Triggered mechanism. In this section, we mainly introduce the runtime protection against malicious attack.

Application of hardware sensors is able to measure parameters of circuits. For example, in paper [13], delay among the paths between registers are assessed and characterized to define a reasonable range. All the delays out of the scope are recognized as malicious behaviors. A shortage of this approach is much extra circuits overheads must be added to insure the monitoring effectiveness and accuracy. Whereas A2 does not disturb any paths' delays, this sensor based method does not work in detecting A2 Trojan.

Power consumption and thermal tracking are common used features in runtime abnormal behavior check. In [14], through tracking the temperature profile, the impact of Trojan activation on power consumption in a chip is analyzed. In [15], hardware Trojan activation is detected by comparison of power use between Trojan clear and Trojan embedded benchmarks using machine learning techniques. Another power tracing based approach is [16], where power monitors are inserted to chip for characterizing the power supply. However, A2 avoids all the above methods because too few gates are utilized in A2, and it will not cause any fluctuation of either thermal change or power consumption.

Runtime verification provides a comprehensive protection to ICs once the secure properties are well defined. For instance, Verifiable ASICs is proposed by Wahby et.al. [17] to verify the correctness of functionality of hardware system. In their paper, runtime (or dynamic) verification is performed by implementing an interactive encryption protocol between untrusted ICs and a trusted IC, where the untrusted ICs are called *Prover* and the trusted IC is called *Verifier*. It is the first attempt to compute proofs of correct execution through utilizing verifiable computation. However, for security purpose, their correctness checking method results in too high computational cost and overhead. Meanwhile, to verify security properties formalized from ICs permitted and prohibited behaviors, a hardware property checker in [18] is utilized to detect hardware and software Trojans, and dynamic checkers in [19] are designed to disclose Processor's malicious logic. Again, all these checkers are not effective in detecting A2 whose behaviors are hard to be formalized.

The proposed runtime detection method targets A2-alike Trojans. Compared with these methods, the proposed detection method is simple and easy to implement. It brings little area and power overhead.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2018.2864246, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

III. The R2D2 Detection Scheme

In this paper, we propose an on-chip runtime hardware Trojan detection method named R2D2 [20]. R2D2 detection scheme targets A2-alike analog hardware Trojans. It aims to detect high frequency toggling on several signals before they can activate the hardware Trojan.

A. Detection Method Overview

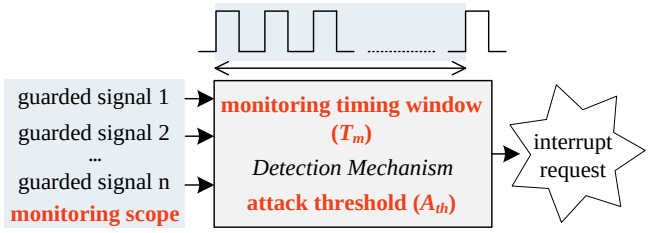


Figure 2: Mechanism of the hardware Trojan detection method

Due to the small size of the A2 Trojan, and the fact that it can be connected to any low toggling signal, we conclude that runtime detection is more feasible. Figure 2 shows the mechanism of the proposed runtime hardware Trojan detection scheme. The principle of this method is to guard a set of concerned software controllable registers or memory related signals. A hardware interrupt will be generated if abnormal toggling events occur in the guarded items. The mechanism cannot be disabled through unprivileged software.

As shown in Figure 2, R2D2 has several parameters that must be tuned in order to ensure the effectiveness of the scheme and eliminate false positives. The first parameter is the size of *monitoring timing window* denoted by T_m. During a monitoring window the detection unit counts the toggling events on the concerned signal. Throughout this time window, if the toggling frequency increases beyond the *Attack threshold* Ath, which is the second parameter, the detection circuit will generate an interrupt request. The other important parameter is the *monitoring scope* which decides the signals to be selected for monitoring. These signals must be ones that have a low toggling frequency during normal processor workloads. Each guarded signal can have a different attack threshold value.

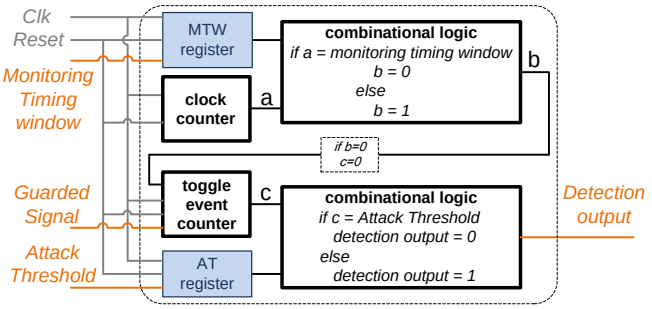


Figure 3: Runtime detection circuit design

The detection circuit is shown in Figure 3. The window size, T_m, and Ath are written into dedicated registers, Monitoring Timing Window register (MTW) and Attack Threshold register (AT) respectively. By keeping these values as software programmable registers, we can create flexibility and uncertainty to the defence mechanism and prevent the attacker from learning them through IC reverse engineering. We must ensure that only privileged software can configure these registers. The clock counter is used to count the clock cycle, and compare with the value in the MTW register. The clock counter is reset when its value reaches the value in the MTW register, and a new monitoring window starts. The toggle event counter is used to count the number of toggle events of the guarded signal, and compare them with the value in the AT register. This toggle event counter is reset when a new monitoring timing window starts. In one monitoring window, if the toggle event counter reaches the value in the AT register, the detection output (the alarm signal) will be activated.

B. Parameter Tuning

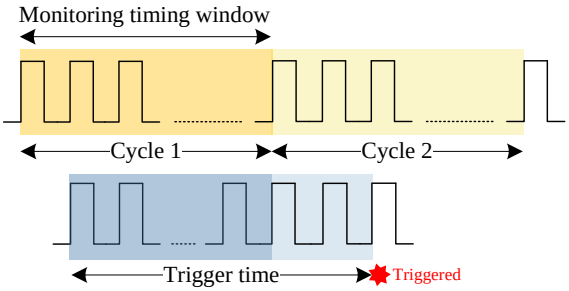


Figure 4: The detection method analysis

The first step in securing R2D2 is tuning the scheme's timing parameters. Per Figure 4, the toggling time it takes for a Trojan to trigger is denoted by T_t, and the average toggling frequency during this period is denoted by f_t. We first assume that T_t is shorter than twice the size of the monitoring window size T_m. The worst case scenario occurs when the triggering duration scope T_t falls equally into two monitoring windows. In this case, to make sure that the sequential pulse signal can be detected, the attack threshold Ath should be smaller than half of the number of toggle events N_t required to trigger the attack. In addition, the value of Ath/T_m should be higher than the average benign toggling frequency f_a of the guarded signal to avoid false detection. This request is trivial since the guarded signals are supposed to have extremely low toggling rates.

In conclusion, we give the following parameter setting constrains:

$$\begin{cases} assume\ T_t \leqslant 2T_m \\ f_a \times T_m \leq Ath \leq N_t/2 \end{cases}$$

If the Trojan trigger time T_t is longer than twice of the monitoring window size T_m we just need to make sure that the attack threshold is smaller than the number of toggling events during one of the monitoring windows. The constraint conditions are summarized as below:

$$\begin{cases} assume\ T_t \geqslant 2T_m \\ f_a \times T_m < Ath \leq T_m \times f_t \end{cases}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2018.2864246, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

$$\Rightarrow f_a \times T_m < Ath \leq T_t/2 \times f_t = N_t/2$$

Combining these two situations, we conclude the parameter setting constrains for this detection method as below:

$$\begin{cases} f_a \leq Ath/T_m \leq 1 \\ Ath \leq N_t/2 \end{cases}$$

The first condition ensures that the detection scheme eliminates false positives and the second condition ensures that attacks will be detected. Once the attack threshold is set, it can detect hardware Trojans whose trigger time is longer than twice of the attack threshold. Only hardware Trojans whose trigger time is shorter than twice of the attack threshold have a chance of evading the detection. In addition, we give the detection rate of this method below, where n stands for the number of toggling events required to activate the hardware Trojan. The designer can obtain an estimate of n and T_t by simulating possible analog Trojan designs in the specific Technology. Note that since these values are programmable they can also be tuned post-fabrication to give the best results.

$$detection\ rate\ = \begin{cases} 0 & n \leq Ath \\ n/Ath - 1 & Ath \leqslant n \leqslant 2Ath \\ 1 & n \geq 2Ath \end{cases}$$

In theory an attacker can fine-tune the Trojan to trigger it sooner than the attack threshold. Hence, the closer the designer sets the attack threshold to the benign toggling frequency of the guarded signals the more difficult it becomes for the attacker to evade detection. Note however that this competition is in favor of the defender due to two reasons. First, the defender may be able to sustain false alarms. If a false alarm occurs, execution falls into an interrupt handler. The interrupt handler can perform a quick integrity check, e.g. verify if the security-critical state of the processor such as privilege levels have been modified or not and return to normal operation if no such violation is detected. Second, the defender is tuning the attack threshold through a precise digital counter, whereas the attacker has to fine-tune the threshold through an analog circuit which has a higher susceptibility to process variation.

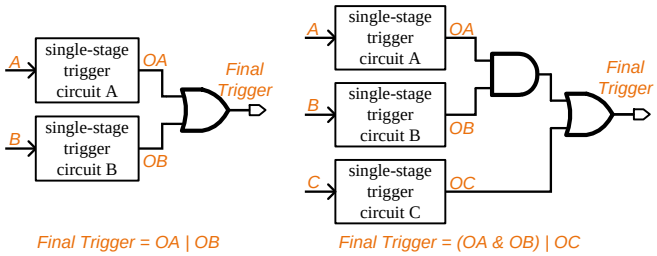


Figure 5: Multi-stage trigger method from [2]

It is possible that the hardware Trojan is triggered by a combination of multiple single-stage trigger outputs, as shown in Figure 5. Assume the single-stage trigger output is 0 when activated. As shown in the figure on the left, the final trigger will be activated when both trigger circuit A and trigger circuit B are activated. If we replace the OR gate with AND gate, the final trigger will be activated when either trigger circuit A or trigger circuit B is activated. In this case, the trigger condition is the same with single-stage trigger. But it makes the trigger more flexible, since the final trigger can be activated when one of many single-stage trigger works. The figure on the right shows a two level trigger input circuit. The final trigger will be activate when trigger circuit C and either trigger circuit A or trigger circuit B is activated.

For OR operation, there are two approaches to activate the final trigger. The first method is to toggle wire A and wire B alternatively for required cycles. This requires the Trojan designer to reduce the toggling frequency requirement of the single-stage trigger. The second method is to toggle wire A for require cycles to active activate trigger circuit A, and then toggle wire B for required cycles to active activate trigger circuit B. This method requires the retention time of the first level trigger circuit be long enough. Multi-stage trigger method can make the trigger more flexible, and can reduce false activation. But it is not easy to choose many wires suitable to trigger the attack, and to write software to make all the wires toggle as expected. If the wires are far from each other, it will be difficult to insert the Trojan in a layout. Multi-stage trigger method also means more complex analog circuit design. All these features limits the complexity of multi-stage trigger circuits, and limits the difficulty to detect them.

From the defender's perspective, we should consider the worst case (the ideal case for the attacker). The monitored parameter is the toggling frequency of possible victim wires. For R2D2 method, the advantage of multi-stage trigger method is that it can reduce the required toggling frequency of a victim wire. But anyhow, the required togging frequency is doomed to be higher than common case. For OR operation, assume it combines N single-stage outputs to trigger the attack. Ideally, it reduce the required toggling rate (f_t) by N times (f_t/N). f_t/N will be definitely above normal toggling rate. Note that if f_t/N is near or even lower than common case toggling rate, it can be easily detected. It is feasible for us to choose appropriate MTW and AT values. In the worst case scenario, we will sacrifice false positive rates for higher security level. False positive event only requires the CPU to halt for further checking, but will not affect the processor's function.

C. Scope Selection

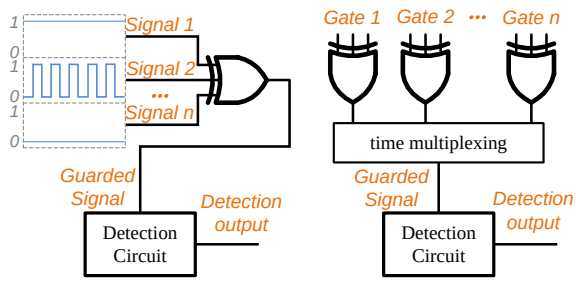


Figure 6: Detection circuit for multiple guarded signals

Selecting the set of guard signals is also a critical part of the design. The goal of the defender is to maximize the coverage of low toggling rate signals while minimizing the hardware overhead of the detection mechanism.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2018.2864246, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

We first note that since the Trojan detection circuit will be small, we can easily insert several units in the design without incurring significant overhead. In addition, we propose to XOR a set of guarded signals together to reduce hardware consumption. Figure 6 shows an example circuit. Since the guarded signals all have low toggling rates when running common programs, the XOR of the signals should also have low toggling rate. If one of the guarded signal toggles frequently, the XOR result toggles as well. But toggling events of other XORed wires will, to some extend, mask the toggling event of the victim wire. So parameters of the detection circuit need to be modified accordingly. From the Trojan designer's perspective, the difference of toggling frequency between the victim wire and other wires will be significant. So the decrease of toggling frequency of the victim wire will be trivial. We only need to lower the Ath value properly. If the attacker knows the existence of the detection circuit and the XOR solution, it is possible for them to mask the high frequent toggling event. For example, they can make one XORed wire toggle oppositely with the victim wire. To realize this goal, they need to infer from the layout, the wires are XORed together with the victim wire, and make one wire toggles as expected to mask the victim wire. This is not easy to realize. If the attacker can infer from the layout, the signals that are guarded, they can simply attach the Trojan to signals that are not guarded by the detector. To thwart this we can use two well known hardware security measures: 1) Post-fabrication in-house configuration. We can use light-weight configurable MUX-trees [21] and route the detection unit to a large number of potential victim signals through the configurable tree. The MUXes are then configured post-fabrication to select only a subset of those signals unknown to the attacker. 2) Split-manufacturing. We can perform the wiring of the detection circuitry in a trusted foundry. While the substrate devices and the bottom metal layers (including the possible Trojan circuitry) are fabricated in the untrusted process, the guarded signals are connected in a split-manufacturing style back-end-of-line (BEOL) process unknown to the attacker. The set of guarded signals can be changed on every batch of chips to create a moving target defence against the attacker. The ReRAM-CMOS technology we use to fabricate our demonstration chip perfectly embraces programmable MUX-trees, and BEOL split-manufacturing. As shown in Figure 6, we can also divide the guarded signals into several groups with each group having different detection parameters A_{th} and T_m. The XOR gates can time-multiplex to reuse the detection circuit as well. Figure 3 show how the XOR gates can reuse the toggle event counter. In this case, we need to shrink monitoring window width, and reduce attack threshold accordingly. We can also add more toggle event counters into detection circuit if necessary.

D. Removal Attack Resiliency

Another important security concern is that the attacker may find the detection circuit and modify/remove it. We assume that the attacker has extracted a netlist from the design layout. Then it would be feasible for the attacker to find the detection unit and the alarm signal and simply disconnect it from the interrupt unit. We need to verify whether the detection circuit works as expected. A low-cost solution to this problem would be to perform testing-based verification of the detection circuitry post-fabrication. This can be done by performing reads and writes to the AT and MTW registers and the scope variables. Then various toggling benchmark codes can be used on multiple signals inside the detection scope to test whether the correct interrupt line is triggered at the correct clock cycles. The only way for the attacker to circumvent this stress-testing, is to clone all the software behavior of the detection circuitry by modifying the digital circuitry. This task would be impossible without significantly disrupting side-channel fingerprints. In essence, such modifications will fall into the realm of digital Trojans for which there are an array of available effective Trojan detection schemes.

IV. Demonstration Design

We demonstrate the effectiveness of the proposed detection scheme and the operation of the hardware Trojan itself on an in-house designed ARMv7 compatible processor which is described herein. ARM processors are the most popular processors in the mobile and embedded system domain. Hence, the security of systems based on ARM architectures is of critical concern.

A. The ARM-compatible Processor

We propose a fused microarchitecture [22]–[24] based on the ARMv7-A&R ISA [25]. ARMv7-A&R was the up to date ARM ISA when we started this project. This ARM processor is named Merlin. Merlin integrates in-order superscalar and VLIW [26]. Normally, Merlin works under dual-issue in-order superscalar mode. It can be switched to six-issue VLIW mode when the task is compute-intensive. It was evaluated based on the gem5 simulator [27], and proved to be feasible, before real hardware design [28]. Using michroarchitectural techniques [29], Merlin expands the digital signal processing capabilities of the ARM processor. Merlin supports most traditional ARM instructions, but does not support some ISA extensions, such as Thumb, ThumbEE, Jazelle, Floating-point, and Advanced SIMD. We realize 181 ARM instructions in total, which is enough to run common benchmarks, such as DhryStone, CoreMark, DSPStone, and EEMBC telecom. There are 7 execution modes defined in ARMv7-A&R ISA, while Merlin works only under user mode.

Branch prediction plays a significant role in improving the processor performance, especially for processors with deep pipeline stages. Merlin has a 10-stage pipeline. Branch prediction happens at the first stage of the pipeline. Until the first execution stage (the eighth stage in the pipeline) stage, the correct branch direction can be acquired, and the predicted direction can be verified to be correct or not. In this design, we propose a combined Bimodal and PAp branch prediction method [30]–[33]. This method achieves 94% prediction accuracy, with limited hardware budget.

Merlin has 16 KB of on-chip L1 instruction Cache, with a 256-bit wide port, and 32 KB dual-port data memory, with each port being 64-bits wide. Merlin has six functional units,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2018.2864246, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

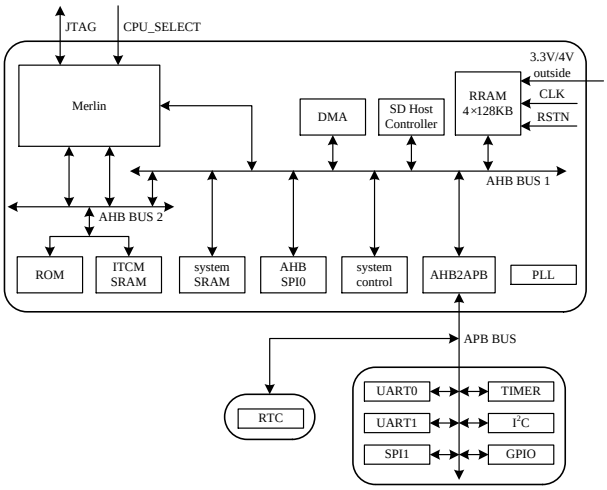


Figure 7: The SoC chip diagram

consisting of two arithmetic units, two multiply units, and two load/store units. An SoC [34] is designed, where Merlin is used as an MCU. The chip diagram is shown in Figure 7. On this chip, Merlin is integrated with DMA, ROM, SRAM, four 128KB embedded ReRAM, and a variety of peripheral I/O. The Dhrystone performance of Merlin is 1.9 DMIPS/MHz, which is comparable to ARM Cortex-A8 processors.

B. A2-like Analog Trojan in Merlin

When inserting the analog Trojan trigger circuit into the Merlin processor, we should first select a viable trigger input. The trigger input should have low toggling rate in common cases. It should be controllable through software, so that the trigger code can make it toggle at high frequency to launch the attack. We also discuss what can be utilized as the payload of the trigger in the Merlin processor.

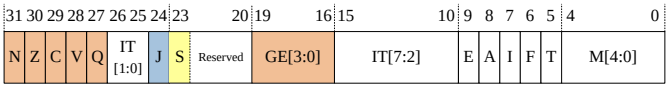


Figure 8: ARM CPSR register

1) Select the Trigger Input: In ARMv7-A&R ISA, there are 16 core registers including R_0-R_{12}, SP (Stack Pointer), LR (Link Register), and PC (Program Counter) under user mode. For each core register, it is possible that the register is frequently used during a time period. So it is not safe to utilize these registers to trigger the attack.

Another software reachable register is CPSR (Current Program Status Register). The definition of CPSR is shown in Figure 8. APSR (Application Program Status Register) is the same register as the CPSR in ARMv7-A&R ISA, but the APSR must be used only to access the N, Z, C, V, Q, and GE[3:0] bits. N, Z, C, and V are condition flags. Q is the overflow or saturation flag. GE[3:0] are the greater-than or equal flags. In the Merlin processor, we realize all these flag registers as part of APSR. All the flag registers can be modified directly using the `MSR` instruction, or be modified indirectly using arithmetic or logic instructions.

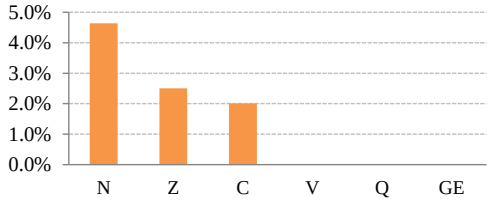


Figure 9: Toggling rate of the NZCV, Q, and GE registers when running the MFCC program

Table I: Some special instructions in ARMv7-A&R ISA

Instruction	Introduction
`PLD, PLDW, PLI`	Preloading caches
`CLREX`	Clear local exclusive access record
`DBG`	Provide a hint to debug and related systems
`DMB, DSB`	Memory barriers that regulate memory accesses

Figure 9 shows the toggling rate of the NZCV, Q, and GE registers when running MFCC, which is a speech recognition program. We can see that the toggling rate of N, Z, C registers are all below 5%. V, Q, GE registers does not toggle at all in this benchmark. So these registers can be utilized as the trigger input. We also utilize one reserved bit, CPSR[23], as the mode switch flag in Merlin. We name it CPSR_S. The toggle rate of CPSR_S is decided by the users. CPSR_S always has very low toggling rate, since it may degrade the performance if the processor is to switch between the two modes too frequently. So the CPSR_S bit can also be used as the trigger input.

The bits in CPSR that we do not implement in Merlin include IT[7:0], J, T, E, A, I, F, and M[4:0]. These bits cannot be modified by `MSR` instruction directly, but some of these bits can still be used as the trigger input. J and T compose the instruction set state register. It indicates whether the processor is working under ARM, Thumb, ThumbEE, or Jazelle mode. The `BLX` instruction calls a subroutine at a PC-relative address, and changes instruction set from ARM to Thumb, or from Thumb to ARM. Exchange of the instruction set between ARM and Thumb can make the T bit toggle frequently. While there is little chance that this happens in common cases. So the `BLX` can be utilized to trigger a Trojan in an ARM processor. IT[7:0] is the IT block state register. This field holds the If-Then execution state bits for the Thumb `IT` instruction. It is possible to make one bit in IT[7:0] toggle by exchanging between IT mode and normal mode. E is the endianness mapping register. Normally, endianness is not changed in one application, so the E bit almost does not toggle at all. But we can use the `SETEND` instruction to set and clear this bit, to make E bit toggle frequently. A (Asynchronous abort), I (IRQ), and F (FIQ) are mask bits. These bits are less software controllable, and it could be dangerous to use these bits to trigger a Trojan attack.

There are also many special instructions which are rarely used in common programs. Signals related to these instructions are predicted to have low toggling rates. So these instructions can also be used to trigger the attack. We list some of the special instructions in Table I. For example, the `PLD` instruction signals the memory system that data memory accesses from a specified address are likely to happen in the near future. The

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2018.2864246, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

memory system can respond by preloading the cache line into the data cache (pre-fetching). In Merlin, `PLD` is decoded in the D unit, and then it signals the DMMU. Continuous execution of `PLD` can make the related signals toggle frequently.

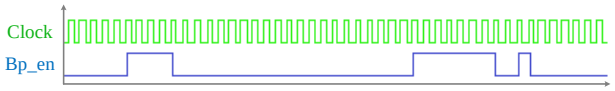


Figure 10: Branch prediction enable signal toggling rate when running MFCC program

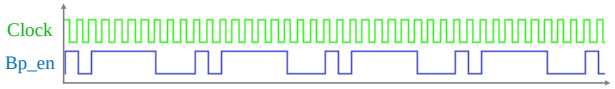


Figure 11: The branch prediction enable signal toggles more frequently if triggered by software

Regarding Merlin, there is another method to trigger the attack. As we mentioned before, Merlin adopts a combined Bimodal and PAp branch prediction method. Merlin fetches 256bit instructions each time, including 8 to 16 instructions. This is called an instruction packet. Once branch instructions are found in the instruction packet, the branch prediction mechanism is enabled. As shown in Figure 10, Bp_en stands for branch prediction enable signal. We can see that the Bp_en signal rarely toggles when running MFCC program. Whereas, we can make the branch predictor work more frequently simply by adding branch instructions into the program. Figure 11 shows that, when running the designed program, the Bp_en signal toggles more frequently than running the MFCC program.

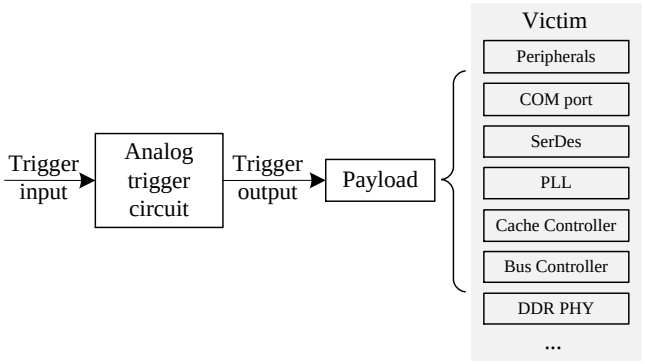


Figure 12: Possible payloads in Merlin processor

2) Select the Payload: As shown in Figure 12, for the target SoC, a hardware Trojan can bring many security issues. For example, the hardware Trojan may attack memory mapped I/O to invalidate the peripherals, or attack the COM port, or SerDes to cut the communication of the SoC. It can also attack the PLL to slow down the chip, or attack the cache controller, bus controller, AD/DA converter, DDR PHY, and so on, to make the chip lose its functionality.

In the hardware implementation, we select the CPSR_J bit as the trigger input. Since Merlin does not support ISA extensions, this bit has no function. We use R_0 as the attack

```
while success==0 do
  i ← 0
  R0 ← 0
  while i<200 do
    CPSR_J ← 0
    CPSR_J ← 1
    i ← i+1
  end while
  if read(R0) ≠ 0 then
    success ← 1
  end if
end while
```

Figure 13: Trojan trigger code

payload. Once the attack is triggered, the value store in R_0 will be modified. The trigger code is shown in Figure 13. We generate the trigger input by frequently writing 0 and 1 to CPSR_J alternatively. When the Trojan is triggered, it changes the value stored in R_0 from 0 to 1 so that we can observe the change through a register read.

V. Experimental Results

Considering that the MCU is digital logic, and the hardware Trojan is analog circuitry, we use Synopsys CustomSim to run simulation. By declaring the name of the analog top-level cell and the analog netlist, CustomSim will simulate the digital logic via VCS [35], and call the related analog simulator to simulate the analog logic. In this section, we will give the simulation result, and introduce the SoC fabrication.

A. Simulation Results

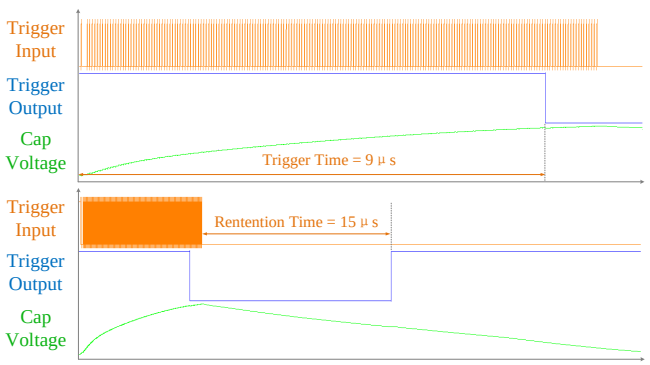


Figure 14: A2 Trojan simulation result

Figure 14 shows the simulation result of the A2 analog circuit. The frequency of the processor is 150MHz. We choose the CPSR_J as the trigger input. The toggling frequency of the trigger input signal is 20MHz. The trigger time is 9 μs. It means that the analog Trojan is activated after 180 toggling events of the trigger input. This result demonstrates that the analog hardware Trojan works in the ARM processor.

Figure 15 shows the simulation result of the R2D2 detection circuit. The detection circuit guards the CPSR_J register. We set the attack threshold as 64, and the monitoring timing window is 256 clock cycles. From Figure 15, we can see that

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2018.2864246, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

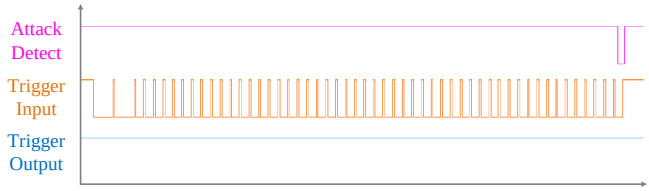


Figure 15: R2D2 detection circuit simulation result

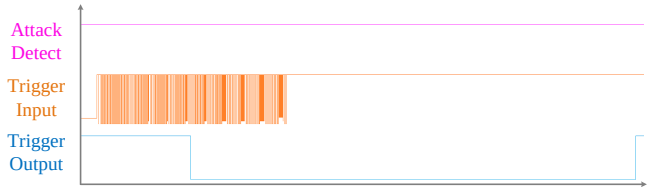


Figure 16: A2 attack simulation without detection

the attack-detected signal generates a low level pulse, after several toggling events of the trigger input. No trigger output is generated. It means that the Trojan is detected and the attack is prevented. The simulation result when we turn the detection circuits off is shown in Figure 16. The attack-detected signal remains 1. Trigger output is generated after several number of toggling events of the trigger input. The result shows that the R2D2 detection method is effective in detecting the analog Trojan.

B. Fabrication

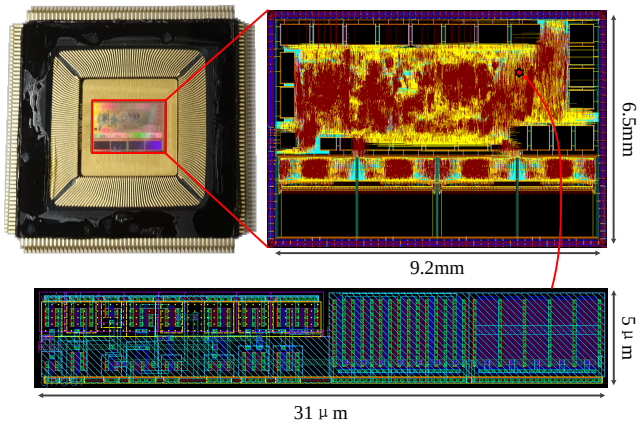


Figure 17: The SoC chip layout

The demonstration SoC is fabricated using the SMIC 130 nm Mixed-Signal 1P7M process. The layout of the MCU and the analog hardware Trojan is shown in Figure 17. Figure 18 shows the photo of the SoC silicon die. In the SoC, the MCU is integrated with 16 KB on-chip L1 program Cache, and 32 KB dual-port data SRAM. ROM is used to store the boot loader. It also embraces 4Mb embedded ReRAM. As shown in Figure 17, the chip area is 9.2 mm by 6.5 mm. The area of the analog hardware Trojan is 31 μm by 5 μm. Trojan-to-circuit ratio is 2.6×10^{-6}.

The detection circuitry is also included in the fabricated SoC, and it is guarding the CPSR_J signal in the MCU. The

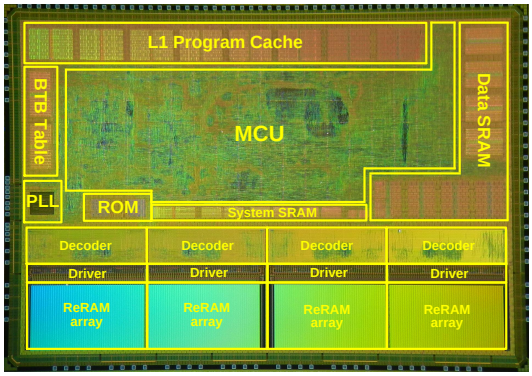


Figure 18: Photo of SoC silicon die

detection circuit is included in the MCU, by automatic place and route, the gates composing the detection circuits are scattered in the layout. The post-layout area of the detection circuit is about 2225 μm^2. Detection-to-circuit ratio is 3.7×10^{-5}. For the detection circuit, the main area consumption comes from the counters. The size of the counters is related to parameters setting. We set T_m=256, A_{th}=64, so the clock counter width is 8, and the toggling event counter width is 6. This includes about 70 gates. The AT register and MTW register include about 12 gates. With these parameters we were able to verify the operation of the Trojan and detection circuitry successfully.

While we did not implement the ReRAM MUX-trees in the demonstration chip we were able to verify that there is sufficient routing area to route the detection unit to more guard signals through MUX-trees. Note that the detection scheme is implemented in digital logic and uses gates similar to other components. This helps better hide the detection circuit from an attacker. Furthermore, the placement tool distributes the detection units making it more difficult to spot them in the layout.

C. On-board Evaluation

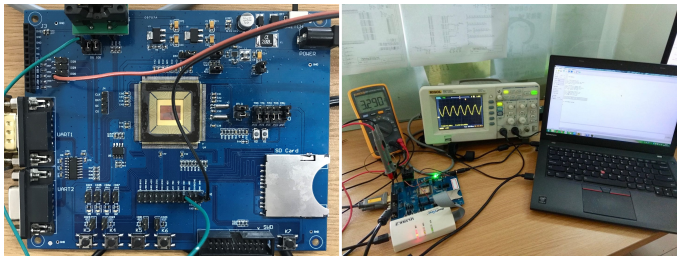


Figure 19: The PCB board and test environment

1) Test environment: Figure 19 shows the PCB board and the test environment. The test program is stored into SPI Flash ROM. When the board is powered on, the processor will start executing the program stored in flash automatically. We can see printed information on PC screen through UART.

2) Trojan functionality: The trigger output signal of the analog Trojan circuit is connected to a GPIO, we can capture the waveform of the Trojan output signal via a oscilloscope, as shown in Figure 20. The Trojan output signal generates a

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2018.2864246, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

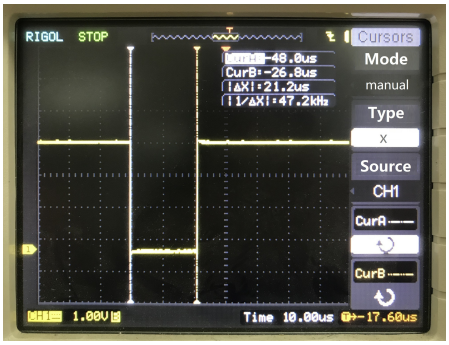


Figure 20: Analog Trojan circuit on-board test result

21.2 us width low level pulse. This result is consistent with the simulation result shown in Figure 14 and Figure 16. In different experiments, the width of the low level pulse signal varies between 20 us and 30 us.

When the analog Trojan is triggered, it will change the value of R0 from `0X0` to `0X1`. We can see the printed value of R0 through UART. As shown in Figure 21, after Trojan attack, the value of R0 is changed from `0X0` to `0X1`.

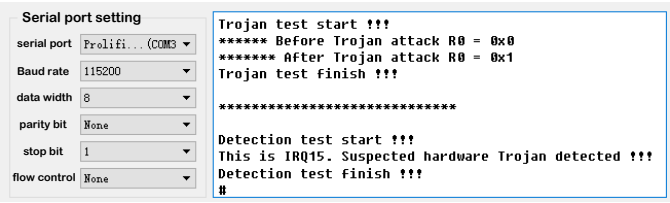


Figure 21: Printed information of the analog Trojan and R2D2 detection circuit

3) R2D2 detection functionality: The detection circuit will generate a hardware interrupt when possible Trojan is detected. The IRQ number is 15 in this design. As shown in Figure 21, when the detected wire is made toggle frequently by running the test program, the detection circuit initiate IRQ 15.

The results demonstrate the feasibility of analog Trojan in the ARM-compatible processor, and the effectiveness of R2D2 detection method.

Table II: Area comparison between Merlin processors with and without the R2D2 detection circuit

	slice LUTs	slice registers
Merlin	104330	212243
Merlin with detection	104354	212268
increase	0.023%	0.012%

4) Area analysis: In section V-B, we estimate the area of the detection circuit in gate level. Based on the parameters of the detection circuit, we estimate that it includes about 70 gates, and the post-layout area of the detection circuit is about 2225 μm^2. There is no easy way to verify this estimation, because the circuits are synthesized from the hardware description language, and it is not easy to find out which gates compose the detection circuit, considering that the gates are scattered into the layout by automatic placement and routing. To provide a more intuitional illustration of the area increase, we synthesize two designs on the same FPGA board. The two designs are the Merlin processor with and without the R2D2 detection circuit, separately. The FPGA device we use is Xilinx XC7Z045FFG676-2. As shown in Table II, compared with the Merlin processor, Merlin processor with R2D2 detection circuit utilizes 0.023% and 0.012% more slice LUTs and slice registers separately. The area increase is trivial. The area synthesize result shows, quantitatively, how much hardware resource is used by the detection circuit, but it is not proportional to the chip area. Many factors affects the area of the chip layout, and such amount of hardware resource increase is negligible.

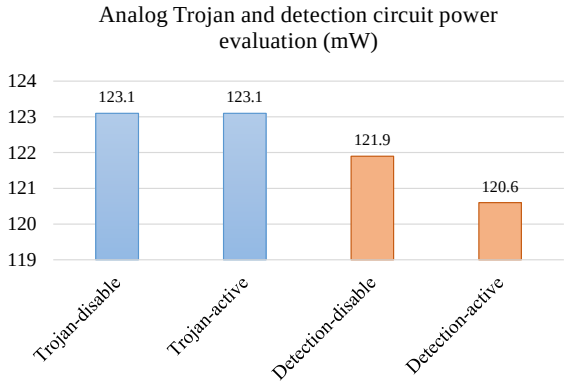


Figure 22: Power analysis of the analog Trojan and the R2D2 detection circuit

5) Power analysis: Figure 22 shows the power analysis of the analog Trojan and the R2D2 detection circuit. To separate the experiments on the analog Trojan and the R2D2 detection circuit, we use one wire as the Trojan trigger input, and use another wire as the guarded signal of the detection circuit. As a result, the detection circuit does not work when the Trojan is triggered, and the Trojan will not be triggered when the guarded signal of the detection circuit toggles. Thus, we can test the power consumption of the analog Trojan and the detection circuit separately.

The first two columns (Trojan-disable and Trojan-active) show the Trojan power experiments. Trojan-disable and Trojan-active run the same test program (program 1). The only difference is that Trojan enable signal is set as 0 for Trojan-disable test, and Trojan enable signal is set as 1 for Trojan-active test. The detection circuit does not work in these two experiments. These two tests shows the power comparison when the processor works normally and when the analog Trojan is triggered. As Figure 22 shows, power consumption under these two occasions are both 123.1 mW. The analog Trojan causes no power increase to the processor.

The last two columns (Detection-disable and Detection-active) show the R2D2 detection circuit power experiments. Detection-disable and Detection-active run the same test program (program 2). The only difference is that the detection enable signal is set as 0 for Detection-disable test, and the detection enable signal is set as 1 for Detection-active test. Program 1 and program 2 are also similar, except that they make different wires toggle. The trigger input signal of the analog Trojan does not change in these two experiments. As shown in

Figure 22, the power consumption in detection-disable test and detection-enable test is 121.9 mW and 120.6 mW separately. When the detection circuit is active, the power of the processor is slightly smaller than in normal scenario. That is because the detection circuit will trigger a interrupt request when suspect Trojan is detected, and it causes the processor to stall for a few cycles.

D. Discussion

R2D2 Trojan detection scheme can be integrated into EDA tools. As shown in Section V-B, the detection circuit is small, and it will bring little overhead to insert many of the detection circuit into the design. EDA tools can collect toggling frequency data of wires through functional simulation, and decide the detection scope. Then, DEA tools can add detection circuits into the design. It can XOR the target wires nearby to reduce area consumption. EDA tools can also help to make the detection circuits more stealthy by scattering the circuits into the layout and adding configurable MUX-trees to mask the detected wires.

VI. Conclusion and Future Work

In this paper, we implement an analog Trojan in a in-house designed ARM processor. We also propose a runtime Trojan detection method. The method targets Trojans triggered by toggling events, overcoming a significant limitation of existing Trojan detection schemes in detecting A2-alike Trojans. The chip is also fabricated using SMIC 130 nm Mixed-Signal 1P7M process. A PCB board is fabricated. Both simulation result and on-board evaluation result prove that this method is effective in detecting an analog Trojan inserted in the ARM processor. We intend to continue this research direction by exploring topics such as optimal parameter tuning, post-fabrication configuration using ReRAMs, and split-manufacturing.

References

[1] H. Li, Q. Liu, and J. Zhang, "A survey of hardware trojan threat and defense," *Integration the VLSI Journal*, vol. 55, pp. 426–437, 2016.

[2] K. Yang, M. Hicks, Q. Dong, T. Austin, and D. Sylvester, "A2: Analog malicious hardware," in *Security & Privacy*, 2016, pp. 18–37.

[3] M. Tehranipoor and F. Koushanfar, "A survey of hardware Trojan taxonomy and detection," *Design Test of Computers, IEEE*, vol. 27, pp. 10–25, 2010.

[4] Y. Jin and Y. Makris, "Hardware Trojan detection using path delay fingerprint," in *IEEE International Workshop on Hardware-Oriented Security and Trust (HOST)*, 2008, pp. 51–57.

[5] S. Narasimhan, D. Du, R. Chakraborty, S. Paul, F. Wolff, C. Papachristou, K. Roy, and S. Bhunia, "Hardware Trojan detection by multiple-parameter side-channel analysis," *IEEE Transactions on Computers*, vol. 62, no. 11, pp. 2183–2195, 2013.

[6] M. Banga and M. Hsiao, "A novel sustained vector technique for the detection of hardware Trojans," in *22nd International Conference on VLSI Design*, 2009, pp. 327–332.

[7] S. Saha, R. S. Chakraborty, S. S. Nuthakki, D. Mukhopadhyay *et al.*, "Improved test pattern generation for hardware trojan detection using genetic algorithm and boolean satisfiability," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2015, pp. 577–596.

[8] H. Salmani, M. Tehranipoor, and J. Plusquellic, "A novel technique for improving hardware trojan detection and reducing trojan activation time," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 1, pp. 112–125, 2012.

[9] 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, 2015.

[10] X. Guo, R. G. Dutta, and Y. Jin, "Eliminating the hardware-software boundary: A proof-carrying approach for trust evaluation on computer systems," *IEEE Transactions on Information Forensics and Security (TIFS)*, vol. 12, no. 2, pp. 405–417, 2017.

[11] X. Guo, R. G. Dutta, P. Mishra, and Y. Jin, "Automatic code converter enhanced pch framework for soc trust verification," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 12, pp. 3390–3400, 2017.

[12] A. Waksman, M. Suozzo, and S. Sethumadhavan, "Fanci: identification of stealthy malicious logic using boolean functional analysis," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 2013, pp. 697–708.

[13] J. Li and J. Lach, "At-speed delay characterization for ic authentication and trojan horse detection," in *Hardware-Oriented Security and Trust, 2008. HOST 2008. IEEE International Workshop on*. IEEE, 2008, pp. 8–14.

[14] D. Forte, C. Bao, and A. Srivastava, "Temperature tracking: An innovative run-time approach for hardware trojan detection," in *Computer-Aided Design (ICCAD), 2013 IEEE/ACM International Conference on*. IEEE, 2013, pp. 532–539.

[15] T. Iwase, Y. Nozaki, M. Yoshikawa, and T. Kumaki, "Detection technique for hardware trojans using machine learning in frequency domain," in *Consumer Electronics (GCCE), 2015 IEEE 4th Global Conference on*. IEEE, 2015, pp. 185–186.

[16] 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, 2015.

[17] R. S. Wahby, M. Howald, S. Garg, A. Shelat, and M. Walfish, "Verifiable asics," in *Security and Privacy (SP), 2016 IEEE Symposium on*. IEEE, 2016, pp. 759–778.

[18] X. T. Ngo, J.-L. Danger, S. Guilley, Z. Najm, and O. Emery, "Hardware property checker for run-time hardware trojan detection," in *Circuit Theory and Design (ECCTD), 2015 European Conference on*. IEEE, 2015, pp. 1–4.

[19] M. Bilzor, T. Huffmire, C. Irvine, and T. Levin, "Evaluating security requirements in a general-purpose processor by combining assertion checkers with code coverage," in *Hardware-Oriented Security and Trust (HOST), 2012 IEEE International Symposium on*. IEEE, 2012, pp. 49–54.

[20] Y. Hou, H. He, K. Shamsi, Y. Jin, D. Wu, and H. Wu, "R2D2: Runtime reassurance and detection of A2 trojan," in *Hardware-Oriented Security and Trust (HOST), 2018 IEEE International Symposium on*. IEEE, 2018.

[21] X. Tang, G. De Micheli, and P.-E. Gaillardon, "A high-performance fpga architecture using one-level rram-based multiplexers," *IEEE Transactions on Emerging Topics in Computing*, 2017.

[22] C. Villavieja, J. A. Joao, R. Miftakhutdinov, and Y. N. Patt, "Yoga: A hybrid dynamic VLIW/OoO processor," 2014.

[23] C. Fallin, C. Wilkerson, and O. Mutlu, "The heterogeneous block architecture," in *IEEE International Conference on Computer Design*, 2014, pp. 386–393.

[24] Khubaib, M. A. Suleman, M. Hashemi, W. Chris, and Y. N. Patt, "Morphcore: An energy-efficient microarchitecture for high performance ILP and high throughput TLP," in *Annual IEEE/ACM International Symposium on Microarchitecture*, 2012, pp. 305–316.

[25] ARM, "ARM information center," 2017. [Online]. Available: http://infocenter.arm.com

[26] Z. Shen, H. He, X. Yang, D. Jia, and Y. Sun, "Architecture design of a variable length instruction set VLIW DSP," *Tsinghua Science & Technology*, vol. 14, no. 5, pp. 561–569, 2009.

[27] N. Binkert, B. Beckmann, G. Black, A. Saidi, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, and S. Sardashti, "The gem5 simulator," *ACM Sigarch Computer Architecture News*, vol. 39, no. 2, pp. 1–7, 2011.

[28] Y. Hou, H. He, X. Yang, D. Guo, X. Wang, J. Fu, and K. Qiu, "Fumicro: A fused microarchitecture design integrating in-order superscalar and VLIW," *VLSI Design*, 2016.

[29] J. L. Hennessy and D. A. Patterson, *Computer architecture: a quantitative approach*. Elsevier, 2012.

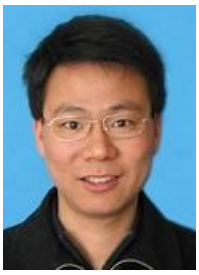
[30] J. K. F. Lee, "Analysis of branch prediction strategies and branch target buffer design," *Computer*, vol. 17, no. 1, pp. 6–22, 1984.

[31] J. E. Smith, "A study of branch prediction strategies," *Proceedings of the 8th annual symposium on Computer Architecture*, vol. 29, no. 6, pp. 135–148, 1981.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2018.2864246, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

[32] J. Hoogerbrugge, "Dynamic branch prediction for a vliw processor," in *International Conference on Parallel Architectures & Compilation Techniques*, 2000, pp. 207–214.
[33] G. Palermo, M. Sam, C. Silvan, V. Zaccari, and R. Zafalo, "Branch prediction techniques for low-power vliw processors," in *ACM Great Lakes Symposium on Vlsi 2003, Washington, Dc, Usa, April*, 2003, pp. 225–228.
[34] S. Furber, "ARM system-on-chip architecture," *Network IEEE*, vol. 14, no. 6, p. 4, 2000.
[35] G. Nunn, F. Delguste, A. Khan, A. Verma, and B. Geden, "White paper using digital verification techniques on mixed-signal socs with customsim and vcs," *Synopsys, Tech. Rep.*, 2011.

Yumin Hou received the B.S. degree in Microelectronics from the School of Microelectronics and Solid-State Electronics, University of Electronic Science and Technology of China, Chengdu, China, in 2012. She is currently a Ph.D. candidate in Electronics Science and Technology at Tsinghua University, Beijing, China, under the supervision of Dr. Hu He. Her main research interests include computer architecture design, Processing-in-Memory architecture and programming model design, and processor security.

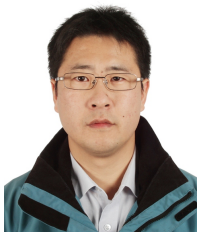


Hu He received his B.S. and Ph.D. degree from the Department of Automation and the Institute of Microelectronics, Tsinghua University respectively. He is currently an associate professor with the Institute of Microelectronics, Tsinghua University, Beijing, China. His research interests include DSP architecture, compiler, processor hardware security, and Processing-in-Memory architecture. His group has developed several generation DSPs in last decade. One VLIW DSP developed for wireless communication and finally used in video surveillance system for cryptography processing was volume produced. He is also interested in spiking neural network learning method, neuromorphic computing, and neural network hardware accelerator.

Kaveh Shamsi received the B.S. degree in Electrical Engineering from the Sharif University of Technology, Tehran, Iran, in 2014, the M.S. degree in Computer Engineering from the University of Central Florida, Orlando, US in 2017. He is currently pursuing a Ph.D. degree in Electrical and Computer Engineering at the University of Florida Florida, Gainesville, US under the supervision of Dr. Yier Jin. His research focuses on analog and digital circuit design, formal methods, and algorithms in hardware security. He is the recipient of the HOST'17 best paper award and the GLSVLSI'18 best paper candidate.

Yier Jin is currently an associate professor in the ECE Department at the University of Florida. He received his PhD degree in Electrical Engineering in 2012 from Yale University after he got his B.S. and M.S. degrees in Electrical Engineering from Zhejiang University, China, in 2005 and 2007, respectively.

His research focuses on the areas of trusted embedded systems, trusted hardware intellectual property (IP) cores and hardware-software co-protection on computer systems. He proposed various approaches in the area of hardware security, including the hardware Trojan detection methodology relying on local side-channel information, the post-deployment hardware trust assessment framework, and the proof-carrying hardware IP protection scheme. He is also interested in the security analysis on Internet of Things (IoT) and wearable devices with particular emphasis on information integrity and privacy protection in the IoT era. He is awarded the DoE Early CAREER Award in 2016 and is the best paper award recipient of DAC'15, ASP-DAC'16, HOST'17, ACM TODAES'18, and GLSVLSI'18.



Dong Wu received the B.S. degree in electronic engineering from Xi'an Jiaotong University, Xi'an, China, in 2001, and Ph.D. degree in microelectronics from Tsinghua University, Beijing, China, in 2006. He is an associate professor in Tsinghua University, and his research focus is circuit design for sensors and memories.

Huaqiang Wu is presently the deputy director of the Institute of Microelectronics, Tsinghua University, Beijing, China. Dr. Wu received his Ph.D. degree in electrical engineering from Cornell University, Ithaca, NY, in 2005. Prior to that, he graduated from Tsinghua University, Beijing, China, in 2000 with double B.S. degrees in material science engineering and enterprise management. From 2006 to 2008, he was a senior engineer in Spansion LLC, Sunnyvale, CA. He joined the Institute of Microelectronics, Tsinghua University in 2009. His research interests include emerging memory and neuromorphic computing technologies. Dr. Wu has published more than 100 technical papers and owns more than 60 patents. Dr. Wu is also served as the director of Micro/Nano Fabrication Center and deputy director of Beijing Innovation Center for Future Chips.