ENTT: A Family of Emerging NVM-based Trojan Triggers

Karthikeyan Nagarajan*
School of EECS
The Pennsylvania State University
University Park, PA, USA
kxn287@psu.edu

Mohammad Nasim Imtiaz Khan*
School of EECS
The Pennsylvania State University
University Park, PA, USA
muk392@psu.edu

Swaroop Ghosh
School of EECS
The Pennsylvania State University
University Park, PA, USA
szg212@psu.edu

***Abstract*—Hardware Trojans in the form of malicious modifications during the design and/or the fabrication process is a security concern due to globalization of the semiconductor production process. A Trojan is designed to evade structural and functional testing and trigger under certain conditions (e.g., after a number of clock ticks or assertion of a rare net) and deliver the payload (e.g., denial-of-service, information leakage). A wide variety of logic Trojans (both triggers and payloads) have been identified, however, very limited literature exists on memory Trojans in spite of their high likelihood. Emerging Non-Volatile Memories (NVMs) e.g., Resistive RAM (RRAM) possess unique characteristics e.g., non-volatility and gradual drift in resistance with pulsing voltage that make them a prime target to deploy a Hardware Trojan. In this paper, we present a delay and voltage-based Trojan trigger by exploiting the RRAM resistance drift under pulsing current. Simulation results indicate that these triggers can be activated by accessing a pre-selected address 2500-3000 times (varies with trigger designs) since the proposed trigger requires a large number of hammerings to evade test phase. Due to non-volatility, the hammering need not be consecutive and therefore can evade system-level techniques that can classify hammering as a potential security threat. We also propose a mechanism to reset the triggers. The maximum area and static/dynamic power overheads of the trigger circuit are 6.68μm^2 and 104.24μW/0.426μW, respectively in PTM 65nm technology.**

***Index Terms*—Trojan, Memory Trojan, Trojan Trigger, Payloads, Read Failure, Write Failure.**

I. Introduction

Hardware Trojan [1] is a malicious modification in a circuit that causes a chip to perform undesirable operations. Ideally, these modifications made to an Integrated Circuit (IC) should be detected during pre-Silicon verification and post-Silicon testing. In order to evade such structural and functional testing, an adversary designs the Trojan to activate only under certain rare conditions and to remain undetected during the test phase. The recently surfaced news about tampering of server motherboards by Chinese manufacturers that affected top US companies like Amazon, Apple etc. [2] serves as a strong motivation to investigate the possibility of hidden components in each step of the design and manufacturing process.

Many prior works have investigated possible hardware Trojans (both triggers and payloads). In [3], a content & timing based Trojan trigger is designed and implemented in a Basys FPGA Board which activates only when the correct input pattern is entered at the correct time. They demonstrate that the Trojan can evade the test phase even if a correct trigger pattern is provided since the timing constraints are not met. In [4], the authors have proposed a Trojan for an embedded SRAM. This Trojan is designed to evade industry standard post-manufacturing memory tests (for example, March test). The Trojan gets activated by writing a specific pattern to one/few cells (that work as a trigger) that feed into the input of the Trojan transistors (payload). The Trojan payload transistors short the data node of a victim SRAM cell to ground (data corruption). Although, the paper employs a data pattern which is not tested by conventional tests, there still lies a finite possibility of activating the Trojan during Burn-in and functional tests when random patterns are applied to the chip. This makes the Trojan detectable. In practical applications, tapping an SRAM internal node to insert Trojan transistors is extremely challenging due to the tight bitcell footprint. Furthermore, the Trojan proposed in [4] is applicable to charge based memories e.g., SRAM and DRAM since the charge stored at a data node can be used to turn ON the malicious transistors. However, these Trojans will not work on emerging memories which use cell resistance/material magnetization to store data.

Designing a small, controllable and undetectable Trojan is key in deploying an efficient one. In [5], an analog Trojan trigger, A2, is presented which is controllable, stealthy and small. It proposes a capacitor-based trigger, that aims to flip specific bits of control logic after a number of instruction issues and can escalate the adversary's privilege. If this is extended to a memory-based trigger, the capacitor value required can be prohibitively high to sustain a large number of hammerings to evade testing. In [6], a capacitor based Trojan trigger circuit is presented that is activated by writing a specific data pattern to a specific address for a number of times. The cap value in this work is high ($\sim$ 100fF) to sustain a large number of charging cycles ($\sim$ 260, denoted by N_{tr} in this work) that may be needed to evade the test phase. A large capacitor has a high footprint, making it easy to identify through optical inspection. Furthermore, the attack gets auto-reset (after 52.39μs) if the hammering is stopped due to charge leakage [6] of the cap.

Motivation for NVM-based Trojan Triggers: The exist-

*Both authors contributed equally to this work.

978-1-5386-8064-3/19/$31.00 ©2019 IEEE

ing pattern-based memory Trojan triggers can be triggered inadvertently [6] whereas A2-based triggers [5], if extended to memory, can be limited by large capacitors that can be detected. Emerging NVMs possess qualities that can evade such restrictions while maintaining a negligible area and energy footprint. For example, the resistance of Resistive RAM (RRAM) [7] or Phase Change RAM (PCRAM) [8] drifts with respect to the applied input pulse voltage (i.e., number and width of pulses). Fig. 1 shows that the resistance of RRAM, employed in this work, changes by 2.21Ω every time a 2ns pulse of 1.2V is applied. This means that the RRAM cell can be hammered and the drifting resistance can be leveraged to trigger a signal that can deliver a payload. The adversary can exploit such Trigger circuits due to, i) a small footprint (ENTTs designed in this work at most require only 6.68μm^2) making it difficult to identify; ii) a large number of pulses are needed to activate the trigger (N_{tr} = 2500 to 3000 in this work); iii) controllability over the attack i.e., the payload can be triggered and reset at will by the attacker.

Emerging NVM-based Trojan Trigger (ENTT) and the Attack Model: We present two RRAM-based Trojan trigger circuits, one of which is based on delay sensing (ENTT-1 in Fig. 2a), and the other based on voltage sensing (ENTT-2 in Fig. 2b). These trigger circuits require a specific pre-selected address enable signal (EN_{ADD}) as its input and outputs a trigger signal ($V_{Trigger}$). This output is high if the specific address is written or read (i.e. accessed) N_{tr} times. The Trojan trigger could be inserted either during the design or fabrication phase. By keeping N_{tr} high through a proper choice of RRAM initial resistance and trigger circuit design, the trigger could evade conventional March tests and random functional tests. The adversary can hammer a pre-selected addresses to activate/reset the trigger (more details in Sections III to VI). *An interesting aspect of the proposed Trojan trigger is that continuous hammering of the address is not required since the RRAM can preserve its incremental change in resistance. Therefore, the adversary can hide the hammering by occasional accesses to the desired address and evade system-level hammering detection schemes*. A brief summary of ENTT trigger and reset operations are provided below:

ENTT-1 (Delay-based): This trigger converts the RRAM's resistance drift during address hammering into an increase in path delay which is used to generate a glitch. EN_{ADD} is a

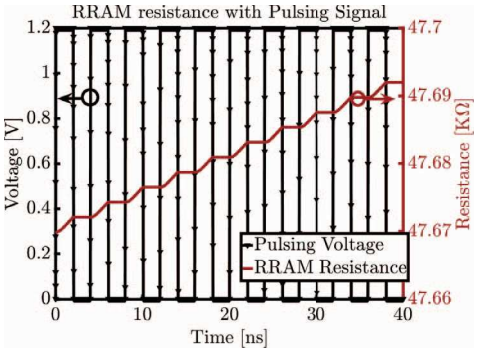
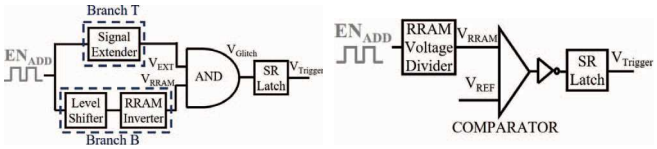


Fig. 1: Drift of RRAM resistance with pulsing signal.



(a) (b)

Fig. 2: a) ENTT-1, b) ENTT-2. A level shifter is needed since RRAM requires 2.2V.

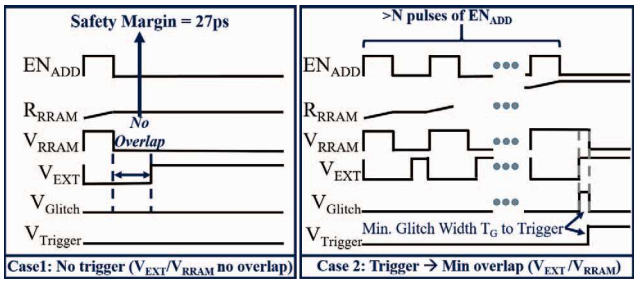


Fig. 3: Glitch generation from outputs of Branch T and B.

decoded signal for a specific address. Therefore, the EN_{ADD} signal is pulsed each time the pre-selected address is accessed. EN_{ADD} goes to two branches, namely Branch T (Top) and Branch B (Bottom) of ENTT-1 (Fig. 2a). Branch T generates an inverted signal (V_{EXT}) of EN_{ADD} where the ON period of the signal is extended (Fig. 3). Branch B generates a delayed version (V_{RRAM}) of EN_{ADD} without any modification (Fig. 3). Every time EN_{ADD} makes a $0 \rightarrow 1$ transition, the RRAM's resistance in the RRAM-Inverter increases. Therefore, the $1 \rightarrow 0$ transition of V_{RRAM} gets delayed. If there are N_{tr} pulses of EN_{ADD} (i.e. corresponding address is accessed N_{tr} times), the $0 \rightarrow 1$ transition of V_{RRAM} gets delayed enough and the AND gate generates a glitch. If the glitch width is wide enough (more than the propagation delay of the SR latch), $V_{Trigger}$ is latched to HIGH. Note that one more inverter (not shown) is connected after the RRAM-inverter to generate a clean signal.

ENTT-2 (Voltage-based): This trigger converts the RRAM's resistance drift during address hammering into a voltage change that is used by a comparator to generate the trigger signal. ENTT-2 is basically a resistance divider between the RRAM and NMOS/PMOS transistors. Each time EN_{ADD} is asserted, a current is passed through the RRAM to increase its resistance. Therefore, the voltage of the node between the RRAM and NMOS/PMOS drops. This node voltage is compared with a reference voltage. If the node voltage is lower than the reference voltage, the comparator output is 0 which is inverted and captured by an SR latch.

Resetting the trigger: We also propose a reset mechanism which can be implemented for both of the ENTTs. We call it ENTT Reset (ENTTR). This reset circuit enables the adversary to stop the attack (to evade detection) and restart again when needed. Essentially, it reverts the resistance drift by hammering the RRAM in reverse polarity.

Payloads of Memory Trojan: Once the trigger circuit is activated, i.e. $V_{Trigger}$ is asserted, it can be used to launch Denial of Service (DoS) attacks, fault injection attacks and

information leakage attacks. We keep this discussion short in this work since the payloads of memory Trojans are already explained in prior works [6] [9].

In summary, the following contributions are made in this paper. We,

(a) Propose delay-based and voltage-based memory Trojan triggers by leveraging RRAM resistance drift;
(b) Propose a Trojan reset mechanism by leveraging the RRAM resistance drift by pulsing a (reverse) current;
(c) Perform design space exploration of the trigger circuitry;
(d) Note that the ENTTs can evade test and row hammer detection routines;
(e) Present process variation analysis to show that the trigger stays inactive during the test phase under worst-case corners;
(f) Discuss the implementation of Trojan triggers and estimate the power/area overhead;
(g) Discuss assumptions, limitations and practicality of ENTT and ENTTR.

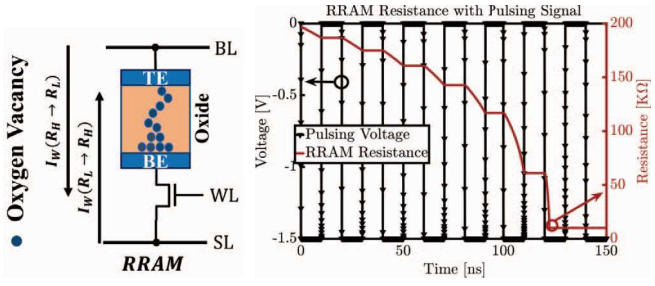
The paper is organized as follows: Section II describes the background of RRAM. Sections III and IV describe the delay-based and voltage-based ENTTs, respectively; Section V presents the proposed resetting mechanisms of the Trojan triggers; Section VI presents a discussion on the practicality, assumptions and limitations of the proposed trigger circuits; Finally, Section VII draws the conclusion.

II. Background on RRAM

In this section, we present the basics of RRAM.

A. Basics of RRAM

RRAM contains an oxide material between its Top/Bottom Electrode (TE/BE) (Fig. 4a). RRAM resistive switching is due to oxide breakdown and re-oxidation which modifies a Conduction Filament (CF). Conduction through the CF is primarily due to transportation of electrons in the oxygen vacancies. These vacancies are created under the influence of an electric field due to the applied voltage. The two states of the RRAM are termed as Low Resistance State (LRS) and High Resistance State (HRS). The process of switching the state to LRS (HRS) is known as SET (RESET).



(a) (b)

Fig. 4: (a) RRAM bitcell; and, (b) RRAM resistance drift (negative) with reverse pulsing signal.

TABLE I: RRAM Parameters Used for Simulations

Parameter	Value
Oxide Thickness	5nm
RRAM Gap (min/max)	0.53nm/1.1nm
Atomic Distance of Oxide	0.25nm
Atomic Energy for Vacancy Generation	1.501eV
Atomic Energy for Vacancy Recombination	1.5eV

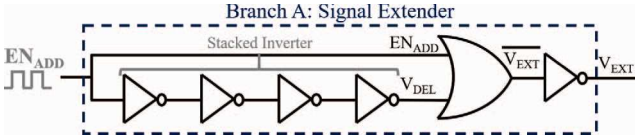


Fig. 5: Branch T: Signal Extender.

B. RRAM Resistance Drift

Fig. 1 shows that a pulsing signal can increase RRAM resistance. Furthermore, RRAM resistance can decrease if the voltage polarity of the pulsing signal is reversed (Fig. 4b). In this work, we leverage the positive and negative drift of RRAM resistance to trigger the Trojan (launch the attack) and to reset the Trojan (exit the attack).

We have used ASU RRAM Verilog-A model [7] along with PTM 65nm technology for the analysis. The RRAM is bipolar HfO_x-based resistive switching memory [7]. The model parameters used in this work are shown in Table I.

III. Delay-Based Trojan Trigger

In this section, we present design and analysis of ENTT-1.

A. ENTT-1: Design and Analysis

ENTT-1 contains two branches namely, Branch T and Branch B. Both branches takes the EN_{ADD} as input. A glitch is generated after the address is accessed N_{tr} times.

Branch T: EN_{ADD} and a delayed version of EN_{ADD} (V_{DEL}) are OR'ed to generate $\overline{V_{EXT}}$ which follows EN_{ADD} but with an extended ON period (Fig. 6). This extension corresponds to the number of accesses to trigger-address, N_{tr}, needed to create a glitch. If a large N_{tr} is desired to evade post-Si testing,

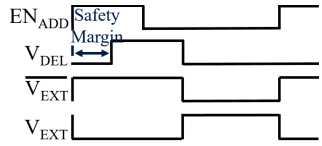


Fig. 6: Branch T: Timing Waveform.

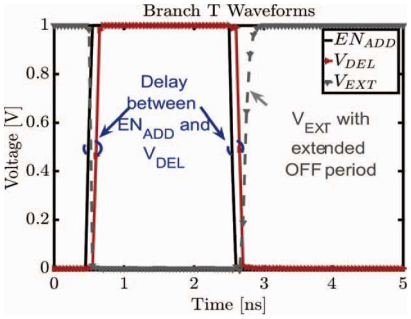


Fig. 7: Simulation timing waveform for Branch T.

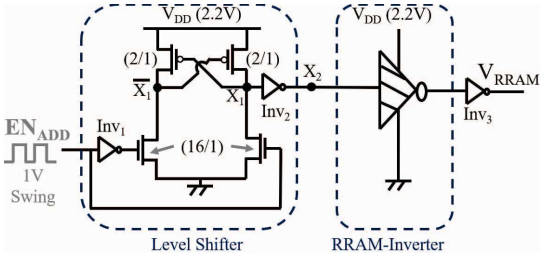


Fig. 8: Branch B: Level shifter and RRAM-inverter.

the adversary can budget a long ON period. Finally, we invert the $\overline{V_{EXT}}$ using a NOT gate. Therefore, V_{EXT} is an inverted version of EN_{ADD} with an extended OFF period. The inverters used are stacked (i.e., 2 PMOSs and 2 NMOSs in series) to get more delays from each inverter. Overall, this branch incurs 293ps of delay between the falling edge of EN_{ADD} and the rising edge of V_{EXT} of which 106ps/101ps/26ps comes from the 4 inverters/the OR gate/the last inverter respectively. Note that we can implement a NOR gate instead of one OR gate followed by one inverter to save more area. However, we used an OR-NOT combination in the block diagram for ease of explanation with the intermediate signal, $\overline{V_{EXT}}$. Fig. 7 shows the timing waveform of the input/output of the NOR gate.

Branch B: This branch (Fig. 8) also takes EN_{ADD} as an input with a 1V swing and converts it to a 2.2V swing through a level-shifter circuit [10]. The level shifting is necessary since the next stage employs an RRAM which requires at least 2.2V to operate correctly. Note that 2.2V is expected to be available in systems employing RRAM as embedded memory/discrete chip. We embed the RRAM in the PMOS path of the inverter so that a change in the RRAM's resistance can be converted into a delay of the 0→1 transition (rise delay) of the RRAM-inverter's output. As mentioned before, the RRAM's resistance changes when a voltage is applied across its terminals. The key idea is to keep the initial resistance of the RRAM small so that the AND output of Branch T and Branch B remain zero. However, the resistance of the RRAM would gradually increase every time the pre-selected address is hammered (to cause the pulsing of EN_{ADD}). This will, in turn, increase the rise delay of the RRAM-inverter in Branch B. Once the extra delay due to the RRAM is more than the safety margin budgeted in Branch T, a glitch (V_{Glitch}) will be generated.

Note that, the input of the RRAM-inverter requires a very sharp falling transition to ensure that the PMOS is fully turned ON and the RRAM experiences maximum disturb voltage. We add a skewed inverter, Inv_2, with a very strong NMOS (W/L=40) after level-shifter since it lacks drive-ability. We have implemented one more inverter after the RRAM-inverter to get V_{RRAM} which is finally AND'ed with V_{EXT} to generate the glitch.

B. RRAM-inverter Design Exploration

It is desirable to keep the rate of change of the RRAM resistance low to ensure that the N_{tr} is high. This will avoid any inadvertent triggering of the Trojan during test/normal mode of operation. However, we don't want N_{tr} to be very high since the operating system/wear leveling techniques can catch them as potential issues. Furthermore, a large N_{tr} will cause a longer delay to start the attack. Therefore, we have considered N_{tr} = 2500 to 3000 as the optimum target number of hammerings. Note that the rate of the RRAM resistance change depends on: i) product of the voltage across the RRAM and disturb time, and ii) initial gap/resistance of the RRAM. These are explained below:

i) Product of voltage across RRAM and disturb time: A current flows through the RRAM when the input of the RRAM-inverter switches from 1→0. A high current through the RRAM is needed to ensure that the voltage across the RRAM is enough to change its resistance. At the same time, if the output node of the RRAM-inverter charges up very quickly, the time duration of the current flow through the RRAM will be too short to change its resistance. Therefore, intentional contention is added in the pull-up path. The following three cases have been considered (Fig. 9): Case 1 where we have an usual inverter with an RRAM. In this case, there is no contention at the output node; Case 2 where we have stacked two PMOS transistors to ensure that the current that charges the output node reduces and the RRAM disturb time increases; and, Case 3 where we have added one more PMOS as feedback (similar to a Schmitt Trigger [11]) to add contention.

Fig. 10a shows that the charging time of the output node (i.e. RRAM disturb time) of Case 3 > Case 2 > Case 1. However, Fig. 10b shows that the resistance change of Case 3 > Case 1 > Case 2. Table II summarizes the key simulation results which explains that the Case 2 fails to trigger (does not latch even for N_{tr} = 10000) since the voltage across the RRAM is

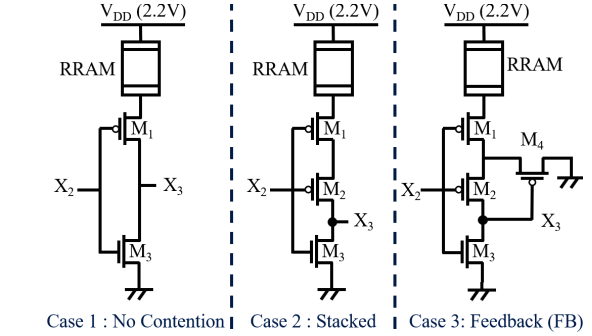
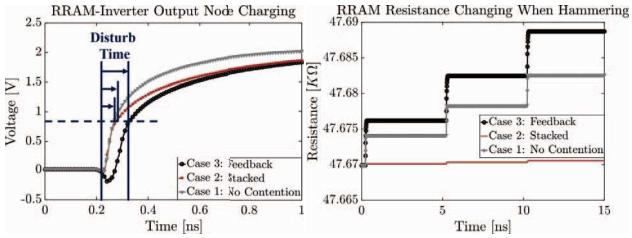


Fig. 9: Design cases for RRAM-inverter of Branch B.



(a) (b)

Fig. 10: Analysis of RRAM-inverter implementation choices: (a) output node voltage with time. Longer switching time is desirable; (b) RRAM resistance drift with time due to hammering.

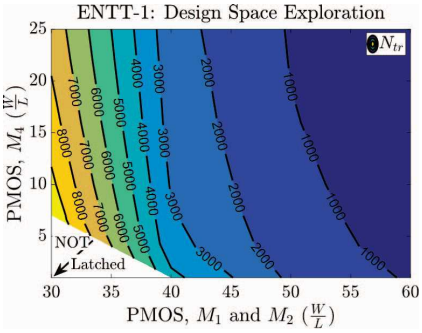


Fig. 11: ENTT-1: Design space exploration.

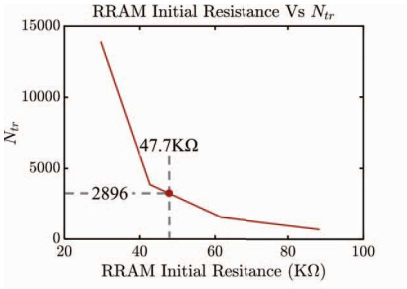


Fig. 12: ENTT-1: RRAM initial resistance vs N_{tr}.

low even though the disturb time of this case is greater than the disturb time for Case 1. We choose Case 3 over Case 1 since N_{tr} for Case 1 is too high (8340).

Fig. 11 shows that N_{tr} in Case 3 depends on the width of the PMOS transistors M_1, M_2 and M_4. This is true since the width of these transistor determines the disturb time and the voltage across the RRAM. We have chosen the (W/L) ratio for M_1, M_2 and M_3 as (40/1), (40/1) and (10/1) respectively for Case 3 since it requires N_{tr} = 2890 to trigger.

ii) RRAM initial resistance: The RRAM's initial resistance also affects the rate of change of resistance (Fig. 12). For example, if the initial gap is too low, the resistance drift will require a higher voltage/longer time and therefore the corresponding N_{tr} will be high. We have considered the initial resistance to be 47.7KΩ in this work.

We note that a delay of 128ps from the RRAM-inverter is needed to latch the glitch (corresponding glitch width is 101ps) by an SR latch. The RRAM gap changes from 530pm to 885pm during this period which translates to a resistance drift from 47.7kΩ to 173kΩ. Fig. 13 shows the timing waveform of Branch T, Branch B, V_{Glitch} and $V_{Trigger}$. Initially, there is no timing overlap between Branch T and B due to the safety margin of 27ps. However, the high to low transition of Branch B gets delayed as the target address is hammered. It takes N_{tr} = 2890 when the outputs of Branches T and B have enough overlap and the AND of these two signals generates a V_{Glitch} of 101ps. The width of this glitch is sufficient to SET the SR latch.

Note that process variations can lead to a worst-case corners where the required N_{tr} can be small and the circuit can trigger inadvertently during the test phase. Therefore, we perform a 1000 point Monte-Carlo analysis with 3σ of 5% of the RRAM initial resistance with a mean of 47.7 KΩ at T=25°C. Fig. 14a

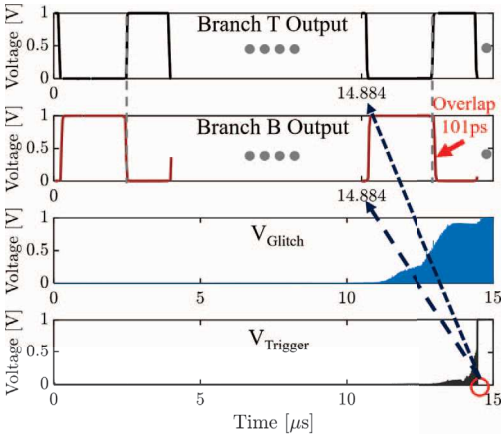


Fig. 13: ENTT-1: Timing waveforms showing Branch T and B outputs, V_{Glitch} and $V_{Trigger}$.

TABLE II: Simulation Result for ENTT-1

Parameter	Case 1	Case 2	Case 3
Peak Current (μA)	620	782	688
ΔR (Ω, per hammer)	4.1	0.2	6.1
Duration of ΔR (ps)	30	57	88
Voltage accross RRAM (V)	1.281	1.159	1.265
Cycles to latch for (W/L)	8340 for M_1=40	N/A	2890 for M_1=40, M_2=40, M_4=10

and Fig. 14b show the distribution of number of cycles for V_{Glitch} to reach 0.5V and 1V respectively and Fig. 14c shows the distribution of N_{tr} to capture V_{Glitch} by the SR latch. Note that even in the worst-case corner, N_{tr} = 1890. This is still high enough to evade testing.

IV. Voltage-Based Trojan Trigger

In this section, we present ENTT-2 by leveraging voltage change at a resistance-divider-node, composed of an RRAM, after the resistance drift.

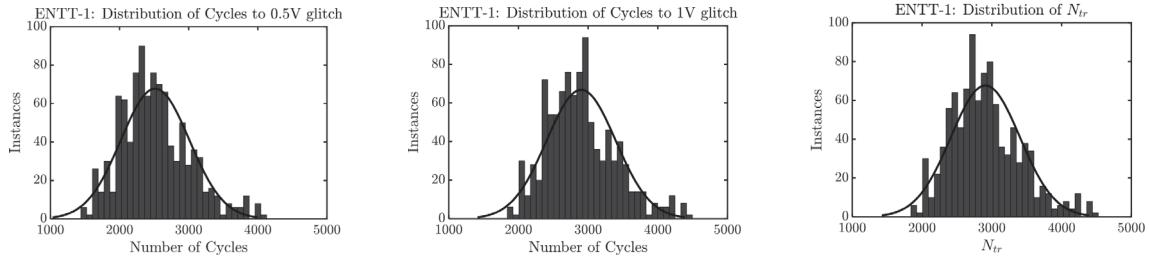
A. Design

ENTT-2 consists of two main components, namely the RRAM-voltage divider and a voltage comparator (Fig. 15). It takes the $\overline{EN}_{ADD}$ signal as an input. The RRAM voltage divider generates a voltage which depends on the RRAM resistance. The voltage remains higher (lower) than the reference voltage of the comparator if RRAM resistance is low (high). This voltage is fed to a comparator which outputs a 1 (0) by comparing this voltage with the reference voltage. When the the output of the comparator is 0, it is inverted and latched to an SR latch to generate $V_{Trigger}$. Note that RRAM resistance can be changed by hammering the pre-specified address.

The RRAM-voltage divider consists of an RRAM bitcell with its BE connected to two stacked transistors. The purpose of the stacked transistors is to reduce the current flowing through the RRAM bitcell during the hammering and increase N_{tr}. This also reduces the static current consumed by ENTT-2.

B. Design Choice and Analysis

The stacked transistors can be either NMOS (Case 1) or PMOS (Case 2).



(a) (b) (c)

Fig. 14: Process variation analysis of ENTT-1. (a), (b) and (c) shows the distribution of N_{tr} for V_{Glitch} to reach 0.5V and 1V and to capture the V_{Glitch} by the SR latch respectively.

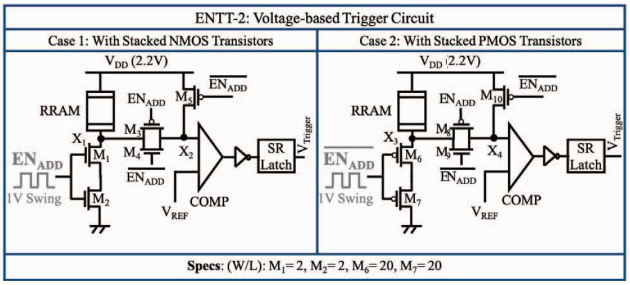


Fig. 15: ENTT-2: Voltage-based trigger circuit.

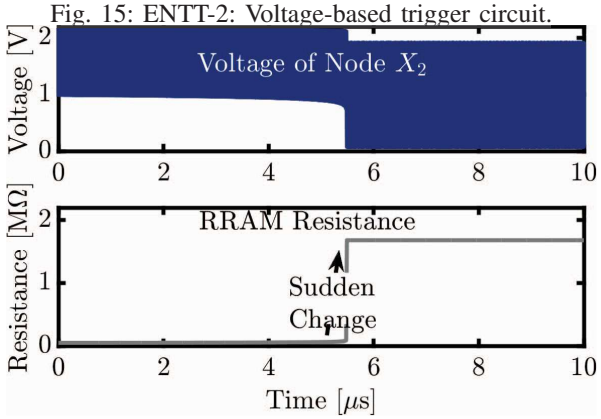


Fig. 16: ENTT-2 (NMOS-based): Voltage of node X_2 and RRAM resistance change.

i) Case 1 (NMOS-based voltage divider): The NMOS transistors M_1 and M_2 are ON when EN_{ADD} is HIGH which leads to a current flow through the RRAM bitcell. The resistance of the RRAM increases during each ON period of EN_{ADD} as shown in Fig. 1. The voltage at node X_1 decreases subsequently. We feed the voltage at node X_1 to a comparator (node X_2) through a pass transistor when EN_{ADD} is LOW. This ensures that the RRAM changes resistance during EN_{ADD}'s HIGH period and the voltage change at X_1 is sensed during its LOW period. Node X_2 is already pre-charged to V_{DD} (through NMOS M_5) when EN_{ADD} is HIGH and gradually discharges when EN_{ADD} is LOW. This voltage stays well above the reference voltage if the RRAM resistance is low and therefore, the comparator outputs a 1. However, a sufficient number of hammerings increases the RRAM resistance and the voltage at node X_2 drops below the reference voltage. This causes the comparator to output a 0. Finally, this output is inverted and fed to an SR-latch the captures the 1.

We observe that the RRAM resistance changes gradually from its low resistance, R_L of 47.7KΩ, during each instance of the hammering. However, the resistance suddenly jumps to a high resistance of 1.68MΩ from 202KΩ after only 1098 times of hammering (Fig. 16). This sudden change in resistance correspondingly leads to a rapid fall in the node voltage at X_1 from 994mV to 74.7mV (sense margin = ((994-74.7)/2)mV = 456.65mV). This result corresponds to the NMOS transistor's (W/L) = 2. Although the change in voltage is large enough to switch the $V_{Trigger}$ from $0 \rightarrow 1$, it occurs too soon. As mentioned before, we have targeted N_{tr} to be 2500-3000. Increasing the (W/L) of the NMOS transistors only leads to a faster change in RRAM resistance and leads to a lower N_{tr}. Therefore, an NMOS-based ENTT-2 provides poor design controllability.

ii) Case 2 (PMOS-based voltage divider): Two stacked PMOS transistors in the RRAM voltage divider (Fig. 15) increases N_{tr}. We provide $\overline{EN_{ADD}}$ to the gates of transistors M_6 and M_7 to ensure that the PMOS transistors are ON when EN_{ADD} is HIGH. When $\overline{EN_{ADD}}$ is LOW, M_{10} is ON and node X_4 is pre-charged to 2.2V. Since the difference between V_{REF} and X_4 is larger than the designed sense margin, the output of the comparator would be HIGH. Therefore, the output of the SR Latch is 0. When $\overline{EN_{ADD}}$ is HIGH, M_{10} is OFF, and M_8 and M_9 are ON. The node X_4 discharges to 884mV during the OFF period of the first EN_{ADD} cycle. The output of the comparator will remain HIGH and $V_{Trigger}$

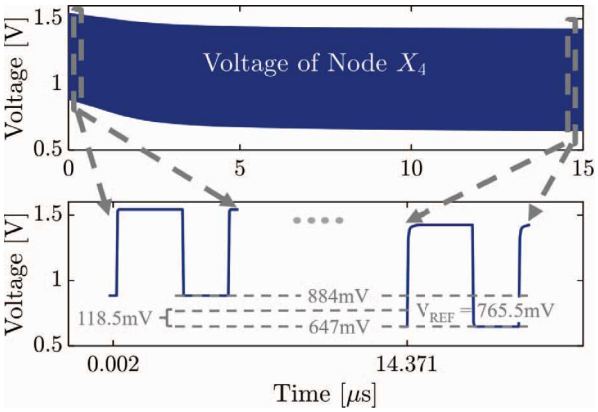
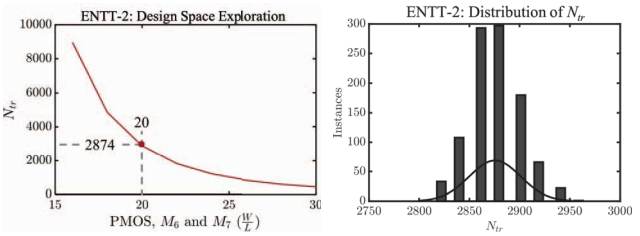


Fig. 17: ENTT-2 (PMOS-based): RRAM voltage divider output and sense margin of ENTT-2.



(a) (b)

Fig. 18: (a) ENTT-2: Design space exploration; and, (b) Process Variation for ENTT-2.

LOW. Node X_4 discharges to 647mV after 2874 cycles of hammering. The RRAM resistance gradually changes from 47.7kΩ to 749kΩ during this time. If the reference voltage, V_{REF}, of the comparator is set to 765.5mV, (gives a sense margin of 118.5mV), the output of the comparator switches to 0 and $V_{Trigger}$ makes a $0 \rightarrow 1$ transition.

Transistors M_6 and M_7 have a (W/L) = 20. To determine the appropriate (W/L) ratio of the PMOS transistors, their widths are varied to determine the corresponding N_{tr}. The minimum sense margin is set as 110mV for the comparator. The results are shown in Fig. 18a. (W/L) = 20 is chosen for M_6 and M_7 to ensure that the N_{tr} is around 3000 (2894 in actual). In order to maintain uniformity with ENTT-1 and ENTT-2, we select the initial RRAM resistance as 47.7kΩ.

We observe that a PMOS-based ENTT-2 provides better controllability. Therefore, we chose it even though the NMOS-based one provides better sense margin. We perform a 1000 point Monte-Carlo analysis with same setup as ENTT-1 and find that the N_{tr} is 2820 for the worst-case corner that is high enough to evade the testing phase.

V. Resetting the Trojan Triggers

We implement an SR latch to keep $V_{Trigger}$ asserted. However, the attacker would like to reset at the trigger at will for better controllability and stealthiness. In this section, we present the design and analysis of a reset circuit. Resetting the Trojan requires two operations namely, resetting the RRAM resistance and generating a reset glitch for the SR latch.

A. Resetting the RRAM Resistance:

We can generate and provide a reset pulse to the SR latch to reset the trigger output. However, V_{Glitch} (for ENTT-1) is generated or the comparator (for ENTT-2) outputs a 0 just after a single access to the pre-selected address. This will set the SR latch again. Therefore, we need to reset the RRAM resistance to its initial low value in order to disable the trigger. The adversary must hammer the address again (N_{tr} times) to change the RRAM resistance and assert the $V_{Trigger}$, as before, to restart the attack. Fig. 19 shows the proposed circuit for resetting the RRAM. Note that the EN_{ADD2} can be generated by hammering another address.

EN_{ADD1} is hammered while EN_{ADD2} stays LOW during the trigger operation (Fig. 19). Therefore, transistors M_2 and M_3 stay OFF while M_1 stays ON (since EN_{ADD2} is LOW). The RRAM resistance increases due to hammering address

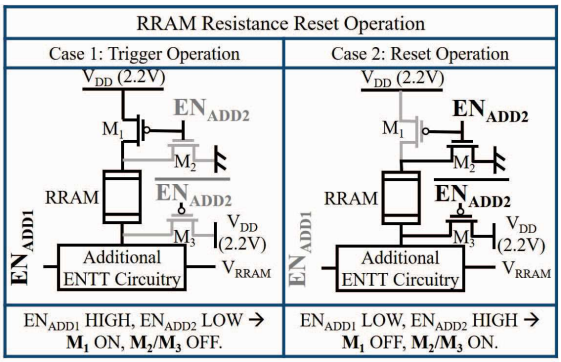


Fig. 19: RRAM resistance reset implementation.

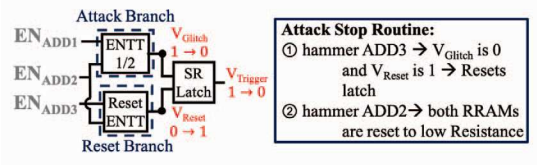


Fig. 20: Implementation of ENTTR.

ADD1 and $V_{Trigger}$ is SET as described in Sections III and IV. During the reset operation, EN_{ADD1} stays LOW and EN_{ADD2} is hammered. M_2 and M_3 are ON and M_1 is OFF (during ON period of EN_{ADD2}). This causes a reverse current to flow through the RRAM and reduces its resistance. EN_{ADD2} is hammered until the RRAM resistance is reset to its initial low resistance, R_L. Here we assume that EN_{ADD1} and EN_{ADD2} are non-overlapping. This ensures that M_1, M_2 and M_3 are never ON at the the same time. This assumption is correct if ADD1 and ADD2 are selected from a single bank since two addresses of the same bank cannot be accessed simultaneously.

B. Generate Reset Glitch for SR Latch

The following mechanism can generate the reset glitch:

Design and analysis of ENTTR (Applicable to both ENTTs): The reset signal can be generated using a circuit similar to the one used for attack generation (ENTT 1/2). However, it requires a different EN_{ADD} signal from a different memory address. EN_{ADD1} is hammered to generate V_{Glitch} (switches $V_{Trigger}$ from 0 to 1) and start the attack. To reset $V_{Trigger}$, at first, EN_{ADD3} is hammered to generate V_{RESET}. Note that we assume address ADD1, ADD2 and ADD3 all are non overlapping since they are from same memory bank (Fig. 20). Therefore, when EN_{ADD3} is hammered, EN_{ADD1} is LOW which keeps V_{Glitch} LOW. The generated V_{RESET} resets the latch ($V_{Trigger}$ from $1 \rightarrow 0$). Finally, EN_{ADD2} is hammered to reset the RRAMs of both ENTT and ENTTR.

VI. Discussions and Limitations

In this section, we present discussion on the practicality, assumptions and limitations of the proposed trigger circuits.

A. Comparison of ENTTs

Table III summarizes the key performance metrics of the proposed ENTTs. Delay-based trigger (ENTT-1) is susceptible to process variation. ENTT-2 consumes lower area (0.46X),

static power (0.51X), dynamic power (0.06X), and total trigger energy (0.06X) compared to ENTT-1 and is less susceptible to process variation as shown in Fig. 18b. Note that the static power can be reduced by gating the trigger circuit with EN_{ADD}. The maximum area and static power overhead of the proposed ENTTs are 6.68μm^2 and 104.24μW, respectively which are less than 0.001% each for a typical memory [12].

TABLE III: Comparison of Different ENTTs

Parameter	ENTT-1	ENTT-2
Dynamic Power (mW)	0.426	0.024
Static Power (μW)	104.24	53.48
Energy/hammer (pJ)	2.1	0.12
Area (μm^2)	6.68	3.06
Target N_{tr}	2890	2874
Worst-Case N_{tr}	1890	2820

B. ENTT Comparison with other Sequential Triggers

Table IV compares the worst-case performance metrics of the proposed ENTTs with other sequential triggers. It can be seen that that [4] requires significantly less design overheads. However, the Trojan will be triggered by writing a pattern to a particular address only after one time. Furthermore, it works for only a charged-based memory and most importantly requires bitcell modifications which are practically very difficult due to the compact nature of SRAM bitcells for high density. However, the proposed ENTTs work for wide range of NVMs. It incurs a higher dynamic and static power (97.6X and 118.6X, respectively) but a significantly lower area (0.045X) compared to [6]. Additionally, the proposed ENTTs incurs 0.026X, 0.017X and 0.005X lower dynamic power, static power and area respectively compared to [5].

TABLE IV: Comparison of various Sequential Triggers

Parameter	This Work	[6]	[5]	[4]
Dynamic Power (μW)	426	4.363	16550	-
Static Power (μW)	104.24	0.879	6210	-
Energy/hammer (nJ)	0.002	87.26	-	1e-6
Area (μm^2)	9.15	203.5	$\sim$1680	-
Target N_{SET}	2874	260	-	1
Worst-Case N_{SET}	1890	-	-	-

C. Memory Trojan Payload

Following payloads can be triggered:

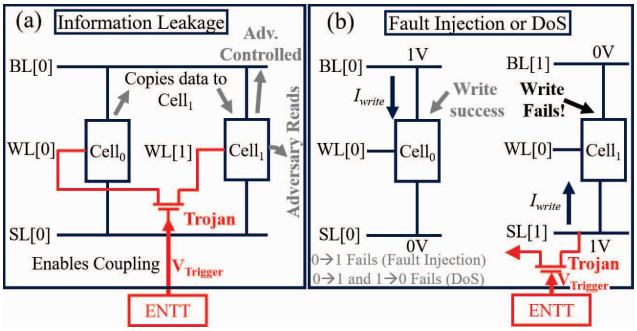


Fig. 21: Malicious memory Trojan causing, (a) information leakage attack; and (b) fault injection or DoS attack. [6].

(a) Information Leakage: An example of this case is shown in Fig. 21a [6]. It is assumed that victim and adversary have control over WL[0] and WL[1], respectively. The WLs share the same bitline (BL[0]) and sourceline (SL[0]) and are coupled through a Trojan transistor (switch). If the switch is activated (by $V_{Trigger}$ from ENTTs), the data will be copied to WL[1] whenever the victim writes to WL[0]. The adversary can read WL[1] to leak the victims write data.

(b) Fault Injection: The Trojan can target memory addresses to prevent writing one particular data polarity (either 0→1 or 1→0). In Fig. 21b [6], we see that 0→1 fails since the headroom voltage between bitline and sourceline is not sufficient to write the cell [13], [14]. However, writing 1→0 is successful. Such fault injections can leak system assets such as cryptographic keys. One example is when an adversary induces single-bit or multi-bit faults in a cryptographic system and performs differential fault analysis by observing correct and faulty pairs of inputs and outputs and subsequently derives simplified equations to extract the keys. Multiple methods for extracting keys using fault injection have been extensively studied and demonstrated [15].

(c) Denial of Service (DoS): If Trojan targets both write polarities (1→0 and 0→1), the victim will not be able to write anything to the memory. This results in a DoS attack.

D. Effectiveness of Trojan Detection Techniques

i) Using failure analysis tools: Sophisticated failure analysis tools like light induced voltage alternation are proposed to detect Trojans. However, this requires significant time and effort, especially for nanometer technologies [16].

ii) Using Automatic Test Pattern Generation (ATPG): ATPG does not work for logic Trojans where the malicious inserted logic is unknown. Therefore, it cannot detect the trigger mechanism.

iii) Side channel analysis (SCA): SCA is also ineffective against the proposed memory Trojans since the trigger circuits only consumes dynamic power when the the Trojan addresses are accessed. For example, an RRAM memory word of 512 bits is written (read) when EN_{ADD} is asserted which consumes around 51.2mA (5.12mA) of current considering I_{write} (I_{read}) per bit is 100μA (10μA). In the inactive mode, ENTT draws a maximum current of 0.62μA (by ENTT-1) which is negligible and cannot be detected using SCA. In the active mode, ENTTs draw an additional 116.7μA of current which is not noticeable. Furthermore, ENTT can be toggled statistically during normal mode if the specified address is accessed. This makes detecting side channel signatures extremely challenging. ENTTs draw a high peak current of 700μA for less than 100ps at the rising edge of EN_{ADD}. However, 512 bits of RRAM also draw a peak current of 66.56mA/9.73mA during write/read operation which hides the peak current drawn by the ENTTs.

iv) Detection by Inspection: For a large memory it is impractical to visually inspect due to the presence of a large number of sub-arrays and interfaces where the Trojan could be hidden. For example, if the sub-array size is 32kB and each

memory bank is 512kB, the total number of banks and sub-arrays in a 0.5GB memory will be 1024 and 16K, respectively. Each sub-array will have ∼8 interfaces where the Trojan could be hidden. This will lead to 128K interfaces that would require inspection. Similarly, the address generation is also present in plurality in the memory array. Automated image analysis of each subarray image with machine learning based detection can potentially detect the Trojan (see countermeasures below) however it will increase Post-Si validation cost.

v) Detection using March Test: It might be possible to trigger the proposed Trojan by hammering each address many times. However, this increases the test time and time to market the chip significantly. The designed ENTTs activate only after 2500-3500 hammerings. For a 4MB RRAM LLC, having 4 banks with 16384 addresses and write latency of 10ns, the required time to just check all addresses for the Trojan is $4\times16384\times3500\times10n$ = 2.29s. This is equal to the total allocated test time including march tests, endurance tests and latency tests [17]. Furthermore, an unique data pattern along with the pre-selected address can be used to trigger the Trojan like [4] making it exponentially difficult to detect using March tests.

E. Effectiveness Against System Level Protocols

i) Wear leveling techniques: Memories with limited endurance such as, RRAM employs wear leveling techniques to even out the wearing of bits. Hammering of a certain address may wear them in reality. The wear leveling techniques map the address to healthy banks so that wearing occurs evenly. Therefore, the address can be changed internally. However, adversary can tap the raw address before the mapping to bypass wear leveling techniques.

ii) Hammering detection routines: Special routines can be implemented to detect/flag hammering as a security violation. However, the adversary can interleave the Trojan triggering address among other valid accesses. Therefore, the hammering will occur at an extremely slow rate. This could be a challenge for capacitor-based triggers such as [6] but ENTT can work due to its non-volatility and bypass the hammering detection.

iii) Detection by ECC: Extra overhead will be imposed to avoid detection due to ECC e.g., ECC bits can also be copied along with data to avoid detection (costs extra transistors). In case of a fault injection, ECC must be computed to match with the data and the faults will be injected into the ECC adding complexity. For DoS, ECC of faulty data will be different than the original ECC. Therefore, ECC may catch errors and will flag cache-misses and page-faults to bring the data to the memory again, imposing a performance penalty. If the errors lead to a silent data corruption, DoS will be successful. In either case the service will be unavailable.

F. Countermeasures against ENTT

Since the ENTTs are designed to evade traditional functional and structural testing techniques, the following techniques can be adopted as countermeasures.

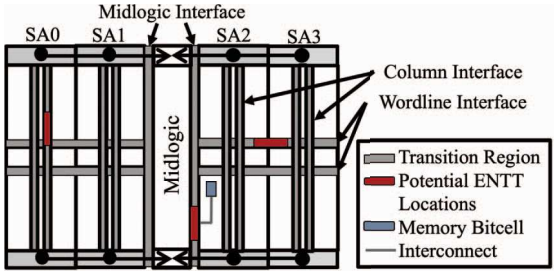


Fig. 22: Placement of ENTT within RRAM memory array.

i) Address Scrambling: Since the adversary exploits a predefined memory address to trigger the Trojan, we can scramble the logical to physical address mapping (fixed or generated from a physically un-clonable function). This will add a layer of complexity on the attacker to hit the predefined physical address.

ii) Small validated ECC: A carefully validated and optically inspected ECC (free of Trojan) can be used to store the ECC for each memory word. If the Trojan performs fault injections/DoS, the ECC will detect it.

iii) Analysis of memory images: Memory Trojans are visually tedious to identify due to replication of large number of memory instances. Machine learning can be applied to analyze the memory bank images to identify anomalies. This approach will worsen the test/validation time.

iv) Temperature/voltage modulation to screen Trojan: Higher operating voltage will accelerate the drift of the RRAM's resistance in the trigger circuit. Therefore, the Trojan could be triggered quickly and can be detected. Similarly, higher temperatures will lower the HRS of the RRAM and aid in detection.

G. Placement of Trojan Trigger

The proposed ENTT and ENTTR can be hidden inside the RRAM memory array as shown in Fig. 22. Note that sacrificial bitcells are used at the interface of array and column areas and, array and wordline areas for smooth transition to logic and to maintain high yield. These non-functional sacrificial bits can be repurposed to hide the Trojan trigger (by the designer or the fabrication house). The additional logic e.g., inverter chains and/or comparators can be hidden in the filler areas of the non-memory logic (e.g., address pre-decoding and pipelining units), also called midlogic [18] and connected to the RRAM. Parasitic capacitance due to routing (∼fF) is not accounted for simplicity. Although it will not change the power consumption significantly, delay may be affected. Floating metals are abundant in the address generation logic and can be reused to route the trigger signal without causing any area-overhead.

H. Endurance Issues with Hammering

RRAM endurance is known to be limited e.g., 10^6 [19]. Since ENTT requires hammering of the RRAM 2-3K times, it may be stipulated that the adversary can trigger the Trojan only limited number of times before it wears out. However, it should be noted that the hammering is incremental due to

weak writes. In fact the RRAM is written only once after 2-3K hammerings. Therefore, the RRAM in ENTT is expected to sustain triggering of up to 10^6 times.

I. Tapping and Pulse Width of EN_{ADD} Signal

In this work, we have considered a pulse of 5ns with T_{ON} = 2ns for EN_{ADD}. From Fig. 10b we observe that the RRAM resistance drifts only during the first 100ps (Table II) of ON period of EN_{ADD}. Therefore, the raw address which is a single cycle signal (i.e., T_{ON} = 0.5ns considering 2Ghz clock) or the subarray-level address enable which is phase-extended to 20 cycles to match the write latency of RRAM (10ns for [7]) can both be tapped as EN_{ADD}. In both cases, N_{tr} remains same since T_{ON} is more than 100ps.

J. RRAM Drift During Test and Normal Mode

Since RRAM is non-volatile, the ENTT resistance can drift in a positive or negative polarity during normal accesses when the specified addresses are accessed by valid programs. Assuming equal probability of accessing each address, the resistance of RRAMs in ENTT and ENTTR are not expected to divert significantly from their base values. Even if slight deviations occur, it will affect the N_{tr} and the ENTT may trigger slightly earlier or later. By keeping the target N_{tr} high we can avoid any unwanted triggering of ENTT.

K. Modulating N_{tr}

An adversary might want to design ENTTs with a higher N_{tr}. We have shown that the value of N_{tr} depends on the followings: i) (W/L)s of PMOS/NMOS of the RRAM-inverter (ENTT-1)/RRAM-voltage-divider (ENTT-2); (ii) RRAM initial resistance (both ENTTs); and, iii) safety margin delay (ENTT-1). The first two parameters can be adjusted to get a high N_{tr}. Adversary can also add an even number of inverters in the Branch T (of ENTT-1/2) to design ENTT-1/2 with a higher N_{tr}.

L. Limitation of SR latch

This work implements a NOR-based SR latch which has the following limitations, i) SET=1 and RESET=1 is prohibited. We design the ENTTRs in this way that both V_{glitch} and V_{RESET} are not 1 at the same time; ii) SET=0 and RESET=0 can lead to an uncertain output when the latch is powering up. We can use ($System_{RESET}$ || V_{RESET}) as the RESET input of the latch to initialize the latch.

VII. Conclusions

We proposed 2 flavors of Emerging NVM Trojan Trigger (ENTT) by exploiting RRAM properties. The salient features of ENTT are, (i) ability to enter and exit the attack at will; (ii) small footprint to evade optical inspection; and, (iii) ability to evade testing due to need of asserting the address a large number of times and system-level hammer detection routines due to non-volatility. ENTT uncovers a new attack surface that adversaries can use to potentially launch extremely menacing attacks including DoS, information leakage and fault injection.

Acknowledgement

This work is supported by SRC (2847.001), NSF (CNS-1722557, CCF-1718474, DGE-1723687 and DGE-1821766) and DARPA Young Faculty Award (D15AP00089).

References

[1] S. Bhunia et al, *The Hardware Trojan War: Attacks, Myths, and Defenses*. 2018.

[2] "The Big Hack: How China Used a Tiny Chip to Infiltrate U.S. Companies. [Online]. Available: https://bloom.bg/2owldii. Accessed: Oct 28, 2018," 2018.

[3] "Hardware Trojan Designs on Basys Fpga Board. [Online]. Available: http://isis.poly.edu/vikram/vt.pdf. Accessed: Oct 28, 2018," 2018.

[4] T. Hoque et al, "Hardware trojan attacks in embedded memory," in *2018 IEEE 36th VLSI Test Symposium (VTS)*, pp. 1–6, April 2018.

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

[6] M. N. I. Khan et al, "Hardware trojans in emerging non-volatile memories," in *Design, Automation Test in Europe Conference Exhibition (DATE)*, 2019.

[7] P. Y. Chen et al, "Compact modeling of RRAM devices and its applications in 1T1R and 1S1R array design," *IEEE Transactions on Electron Devices*, vol. 62, pp. 4022–4028, Dec 2015.

[8] S. Kim et al , "Resistance and threshold switching voltage drift behavior in phase-change memory and their temperature dependence at microsecond time scales studied using a micro-thermal stage," *IEEE Transactions on Electron Devices*, vol. 58, pp. 584–592, March 2011.

[9] R. S. Chakraborty et al, "Hardware trojan: Threats and emerging solutions," in *2009 IEEE International High Level Design Validation and Test Workshop*, pp. 166–171, Nov 2009.

[10] J. M. Rabaey et al, *Digital Integrated Circuits*. Upper Saddle River, NJ, USA: Prentice Hall Press, 3rd ed., 2008.

[11] J. M. Jorgensen, "Cmos Schmitt Trigger," Oct 1976. US Patent, 3984703A.

[12] M. Chang et al , "Technology comparison for large last-level caches: Low-leakage sram, low write-energy stt-ram, and refresh-optimized edram," in *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*, pp. 143–154, Feb 2013.

[13] M. N. I. Khan et al , "Information leakage attacks on emerging non-volatile memory and countermeasures," in *Proceedings of the International Symposium on Low Power Electronics and Design*, ISLPED '18, (New York, NY, USA), pp. 25:1–25:6, ACM, 2018.

[14] M. N. I. Khan et al , "Fault injection attacks on emerging non-volatile memory and countermeasures," in *Proceedings of the 7th International Workshop on Hardware and Architectural Support for Security and Privacy*, HASP '18, (New York, NY, USA), pp. 10:1–10:8, ACM, 2018.

[15] S. Bhasin et al, "Fault injection attacks: Attack methodologies, injection techniques and protection mechanisms," in *Security, Privacy, and Applied Cryptography Engineering* (C. Carlet, M. A. Hasan, and V. Saraswat, eds.), (Cham), pp. 415–418, Springer International Publishing, 2016.

[16] X. Wang et al, "Detecting malicious inclusions in secure hardware: Challenges and solutions," in *2008 IEEE International Workshop on Hardware-Oriented Security and Trust*, pp. 15–19, June 2008.

[17] "Addressing Test Time Challenges. [Online]. Available: https://semiengineering.com/addressing-test-time-challenges/. Accessed: Oct 28, 2018," 2017.

[18] U. Bhattacharya et al, "45nm SRAM Technology Development and Technology Lead Vehicle," in *Intel Technology Journal*, vol. 12, pp. 111–120, June 2008.

[19] C. Nail et al, "Understanding rram endurance, retention and window margin trade-off using experimental results and simulations," in *2016 IEEE International Electron Devices Meeting (IEDM)*, pp. 4.5.1–4.5.4, Dec 2016.