Dynamic Faults based Hardware Trojan Design in STT-MRAM

Sarath Mohanachandran Nair, Rajendra Bishnoi, Arunkumar Vijayan, Mehdi B. Tahoori
Chair of Dependable Nano Computing (CDNC), Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
Email: {sarath.nair, rajendra.bishnoi, arun.v, mehdi.tahoori}@kit.edu

***Abstract*—The emerging *Spin Transfer Torque Magnetic Random Access Memory* (STT-MRAM) is seen as a promising candidate to replace conventional on-chip memories. It has several advantages such as high density, non-volatility, scalability, and CMOS compatibility. With this technology becoming ubiquitous, it also becomes interesting as a target for security attacks. As the fabrication process of STT-MRAM evolves, it is susceptible to various fault mechanisms which are different from those of conventional CMOS memories. These unique fault mechanisms can be exploited by an adversary to deploy *hardware Trojans*, which are deliberately introduced design modifications. In this work, we demonstrate how a particular stealthy circuit modification to inject a fault mechanism, namely dynamic fault, can be exploited to implement a hardware Trojan trigger which cannot be detected by standard memory testing methods. The fault mechanisms can also be used to design new payloads specific to STT-MRAM. We illustrate this by proposing a new payload by utilizing coupling faults, which leads to degraded performance and data corruption.**

I. Introduction

With the globalization of the semiconductor industry, various design and fabrication stages are outsourced to untrusted third-party vendors [1–3]. In this scenario, adversaries at any of these stages can potentially include malicious hardware as a part of the design that is hard to detect by traditional verification and test methods. This extra hardware, termed as *hardware Trojan*, usually remains dormant under normal operating conditions, and gets activated only at rare conditions defined by the adversary. The activation of a Trojan inserted on a chip under deployment can cause various malicious effects such as information leakage, Denial-of-Service (DoS), change in functionality or degradation of performance [4].

The emerging *Spin Transfer Torque Magnetic Random Access Memory* (STT-MRAM) is a leading next generation non-volatile memory technology. It has various advantageous features such as high density, scalability, high endurance, low access latency, CMOS compatibility and immunity to radiation-induced soft errors [5, 6]. As this technology is currently under widespread industrial adoption, its security implications are becoming highly relevant. Therefore, it is crucial to investigate new hardware security threats and countermeasures which exploit the unique features of this technology.

One particular vulnerability lies in the way the STT-MRAM technology is susceptible to defects and failures. The manufacturing defects (shorts and opens) lead to both *static* and *dynamic* faults in the memory operation. A static fault is sensitized by only one memory operation (read or write) whereas a dynamic fault requires multiple read/write operations to sensitize the fault. Another class of faults is called *coupling faults*, where a defect in one cell (aggressor) results in faults in a neighboring cell (victim). The defects and the corresponding faults in STT-MRAM have been explored in various previous works [7, 8]. In this work, we take advantage of them and deliberately introduce some of these defects through modifications in the circuit netlist and/or layout to cause controlled dynamic and coupling faults, which are used to design the Trojan trigger and payload.

Threat Model: We consider an adversarial model where the attacker is the fabrication house, i.e, untrusted foundry [9]. This threat model is similar to the one considered in [10] and [11]. To design the Trojan trigger, we make use of dynamic faults. The dynamic faults are observed for a particular range of resistance values for opens and shorts [8, 12, 13]. This means that, in the absence of manufacturing defects, these faults can be introduced maliciously by injecting resistances of specific values between different nodes in the bit-cell. In this way, the adversary deliberately injects controlled dynamic faults in multiple cells in a memory array, in specific memory row, activated only by a series of specific access patterns. The Trojan gets triggered only when a specific memory address, storing a specific data pattern is read multiple times. This method is also scalable, where the probability of accidentally triggering the Trojan can be reduced exponentially by increasing the number of cells with injected fault and also by tuning the value of injected resistance to increase the number of read operations required to sensitize the fault. Once the fault is triggered, various payloads can be used to cause malicious effects. To illustrate this, we propose one new payload specific to the STT-MRAM technology. It makes use of controlled injected coupling faults, which are easy to implement and have minimal footprint.

We evaluate the proposed approach using industry standard MTJ and transistor models and show that the proposed Trojan design is hard to detect and can escape the standard post-manufacturing memory tests. We also show how the proposed Trojan design can still work even in the presence of manufacturing variations.

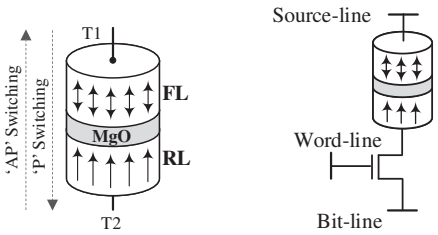
The rest of the paper is organized as follows. In Section II, the basics of hardware Trojans, STT-MRAM technology, and related work are discussed. Section III presents the proposed Trojan deployment mechanism, followed by details of the proposed Trojan trigger and payload in Section IV. Finally, Section V concludes the paper.

II. Background

A. Hardware Trojan

Hardware Trojans are stealthy modifications of a hardware by an adversary to trigger various malicious operations. Functional Trojans are defined to have two parts, namely *Trojan trigger* and *Trojan payload*.

978-3-9819263-4-7/DATE20/©2020 EDAA



(a) MTJ (b) Bit-cell

Fig. 1: STT-MRAM storing device

1) Trojan Trigger: The Trojan is designed to get activated only on a rare combination of events, such that it does not get self-exposed in normal operation conditions. To achieve this property for a Trojan activation, a trigger logic can be implemented by an adversary that asserts the malicious activity only when the input to the trigger logic satisfies a specific rare condition. For example, a trigger logic in a memory block can activate the Trojan by checking for n consecutive write operations to a specific address location with a specific data word. To realize this trigger logic, a finite state machine can be used with address and data bits as inputs.

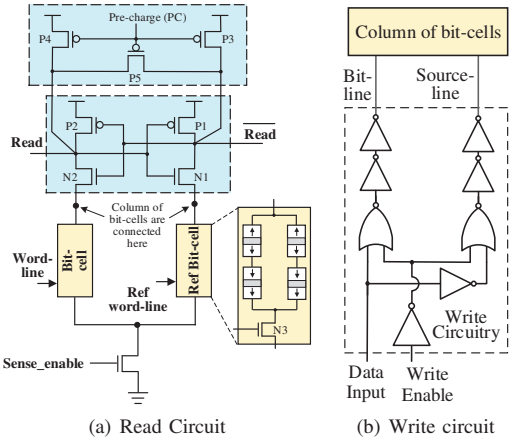
2) Trojan Payload: The Trojan payload determines the effect that the Trojan inflicts on the chip. When the trigger condition is satisfied, the payload gets activated. A payload circuit in memory can potentially disable the memory, change the stored values, or leak the values to a different address.

B. STT-MRAM technology

The storage element in STT-MRAM technology is the Magnetic Tunnel Junction (MTJ) in which the value is stored in the form of resistance states. The MTJ comprises of two ferromagnetic layers (e.g., CoFeB) separated by a thin barrier oxide layer (e.g., MgO) as shown in Fig. 1(a). One of the two ferromagnetic layers has a fixed magnetic orientation and is known as the Reference Layer (RL). The magnetic orientation of the other layer, called the Free Layer (FL), can be freely rotated. The MTJ exhibits a low resistance state when the magnetic orientations of the two ferromagnetic layers are parallel to each other ('P'configuration). Alternatively, the MTJ has a high resistance state when the magnetic orientations of FL and RL are anti-parallel ('AP' configuration). The magnetic orientation of the MTJ depends on the direction of the current flow as shown in Fig. 1(a). To read the content of the MTJ, a small read current is passed through the MTJ stack and the MTJ resistance state is sensed using a sense amplifier. A typical STT-MRAM bit-cell structure is 1T1MTJ (one transistor and one MTJ) as shown in Fig. 1(b). The bit-cell has a total of three terminals, namely source-line (SL), bit-line (BL) and word-line (WL). The read and write circuitry that we have employed in our design are shown in Fig. 2(a) and 2(b), respectively.

C. Faults in STT-MRAM

As the proposed hardware Trojan mechanism is achieved by injecting deliberate faults in the STT-MRAM circuitry, in this section, we briefly describe the read and write fault models associated with STT-MRAM. The write faults are classified as *Transition Faults* (TF), TF0 and TF1. TF0 fault occurs when a cell storing a '1' value is unable to switch to '0' during the write operation and vice versa for the case of TF1 fault. For the read operation, the faults are classified as *Incorrect Read Faults* (IRF), IRF0 and IRF1. IRF0 means that a cell storing a '0' is read as '1' by the read circuitry, whereas in IRF1 fault, a cell storing '1' is wrongly read as '0'. The TF and IRF faults are classified as static faults since only a single operation (read or write) is required to sensitize the fault.



(a) Read Circuit (b) Write circuit

Fig. 2: STT-MRAM read and write circuits

In addition to static faults, STT-MRAM is also subjected to dynamic as well as inter-cell coupling faults [7, 8]. A dynamic fault requires multiple consecutive read/write operations to sensitize and hence detect the fault. In this work, we make use of *dynamic Incorrect Read Faults* (dIRFs) which require multiple read operations from the same cell, to design a hardware Trojan trigger. A dIRF-n is a dynamic IRF which requires at least n consecutive reads to sensitize the fault. It means that with $m < n$ reads, the read result is still error-free. A *coupling fault* occurs when a defect in one cell (aggressor) results in faults in a neighboring cell (victim). In this work, we make use of coupling faults to design a Trojan payload, where the aggressor and victim cells are in the same column.

D. Related work

Most previous work on the deployment and detection of hardware Trojans in Integrated Circuits (ICs) have focused on the logic part of the circuit [1, 14]. There is limited work which explores hardware Trojan attacks in embedded memories [11]. A new class of hardware Trojans targeting embedded memories was first proposed in [11], where the Trojan is designed to evade detection by industry standard post-manufacturing memory tests such as March tests [15]. However, there is still a probabilistic possibility that the Trojan gets activated during the test phase since the Trojan trigger depends only on the data pattern of a few bits in the memory. Moreover, the proposed Trojan is applicable to charge-based memories (e.g., SRAM and DRAM) and cannot be directly used for STT-MRAM which stores data in the form of resistance states [10].

A hardware Trojan for STT-MRAM was proposed in [10]. Here, the Trojan trigger is based on a capacitor charging mechanism, which is already proposed in [16]. The novelty in [10] is that the charging of the capacitor is done by the ground bounce generated by writing a specific data pattern to a specific address for a certain number of times. However, the ground bounce can be eliminated by various design techniques such

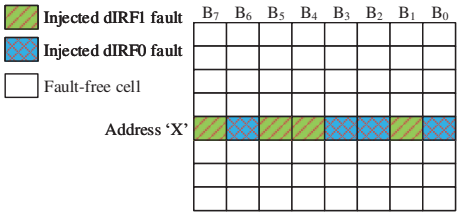


Fig. 3: Injected dynamic faults at a specific memory address to be used for Trojan trigger. In this example, the Trojan is triggered by reading the pattern of 10110010 n times at address 'X'.

as strengthening the power grid system or by using decoupling capacitors. Moreover, the actual data pattern itself can be used to charge the capacitor instead of the ground bounce [10]. The authors also propose various payloads for information leakage, fault injection and DoS. However, the Trojan trigger mechanism is generic which can be used with other memories as well and not specific to MRAMs.

In contrast to the prior works, we propose a hardware Trojan trigger which utilizes the inherent fault mechanism of STT-MRAM, namely *dIRF* (see Section II-C). In addition, we also propose one new payload based on another fault mechanism, namely *coupling fault*. These faults can be introduced by deliberately inserting resistances of specific values between different nodes of the STT-MRAM bit-cell with minimal footprint and modifications to the original memory circuitry, as will be explained in Section III.

III. Proposed Trojan Mechanism

We propose a new hardware Trojan design for STT-MRAM by leveraging the inherent fault mechanisms. We exploit dynamic faults for the Trojan trigger, whereas coupling faults are used for the payload. Specifically, we use a particular class of dynamic faults, namely dIRF0-n and dIRF1-n, which happens when the bit-cell containing the fault is storing a '0' and '1', respectively. Here, n stands for the number of operations required to sensitize the fault.

Dynamic faults for SRAM and DRAM have been observed and validated based on industrial test results [12, 17]. These faults have been linked to defects (opens and shorts) with specific resistance values and have been validated based on SPICE simulations [12, 13]. Recently, dynamic faults for STT-MRAM and the type of defects causing these faults have been analyzed in [8]. These faults occur due to RC time constants introduced as a result of additional resistances. For instance, a dIRF1 fault occurs as a result of a short across the MTJ (see Fig. 4(a)). This is because the additional resistance introduced due to the short results in charging up the SL node faster. This charge cannot discharge during consecutive reads and hence results in an increase in voltage across the MTJ. This, in turn, increases the current through the bit-cell and hence biases the read output to '0' even when the MTJ is storing a '1'. The amount of charge buildup and hence the number of read operations to cause the fault (value of n) depends on the resistance associated with the short. Similar explanations can be given for other dynamic faults as well.

We exploit dynamic faults to design a Trojan trigger which gets activated only when a specific memory address storing a specific data pattern is read n times. The value of n is chosen such that it is significantly larger than the typical sequence used during memory tests, hence, it cannot be detected by the standard post-manufacturing memory tests. Additionally, since the Trojan trigger is activated only by a specific access pattern, this can never happen in memory *March test patterns* [15]. As an illustration, consider a memory array as shown in Fig. 3. All the cells in the array are fault-free, except for the cells at address 'X', which is chosen for Trojan insertion (corrupted cells). At this address, we design the cells B_7, B_5, B_4 and B_1 to have injected dIRF1-n dynamic fault whereas the cells B_6, B_3, B_2 and B_0 are designed to have dIRF0-n dynamic fault. Now if the address 'X' is storing an 8-bit data pattern '10110010' in bit-cells B_7 to B_0 and the address is read n (e.g., 8) consecutive times, the Trojan gets triggered. Note that if the address 'X' is storing any other data pattern, the Trojan is not triggered. This means that the probability of accidentally triggering the Trojan during consecutive read operations is $2^{-m} \times r^{-n}$, where m is the number of corrupted cells ($m = 8$ in Fig. 3) and r is the number of rows in the memory array. For a 512×512 memory array, this probability is around 8.27×10^{-25}.

The Trojan trigger remains active as long as there is a dIRF in all of the chosen bit-cells. The trigger can be reset by a write operation to the address 'X' or by read/write operation to another address. This way the activation of the trigger mechanism is "transient", i.e., not persistent, and hence harder to detect. Please note that this Trojan cannot be detected by typical memory test patterns. The trigger is activated only by specific pattern with n consecutive accesses. Even though the memory March test could be extended to account for such n accesses, with associated significant memory test costs, the March pattern (all-0, all-1, or checkered pattern) cannot activate the specific pattern needed for this trigger. On the other hand, for designing the payload, we make use of coupling faults. We design the fault in such a way that once activated by the Trojan trigger, a fault in one cell in a column causes dIRF and also results in faulty write operation in all other cells in the same column.

Since the proposed Trojan requires only resistance insertion in some of the bit-cells, an adversary (untrusted foundry) can easily implement the proposed hardware Trojan with minimal footprint. It should be noted that resistors (non-malicious) are part of the STT-MRAM memory array. For instance, in [18], the reference circuit consists of two R_{AP} state MTJs in series with a poly resistor. Similarly, in [19], the reference resistor for the sensing circuitry is made using a thin-film precision resistor. Hence, it would be very difficult to distinguish the malicious resistors used for Trojan insertion from the non-malicious ones. Although the insertion of resistances at only certain locations for Trojan design can disturb the regular structure of the memory array, these can be hidden in the layout as in [11], where the authors show how transistors can be introduced in regular SRAM arrays without changing the cell footprint. Moreover, the untrusted foundry (attacker) can insert dummy resistances at suitable locations to regularize the memory array and hence evade detection.

IV. Designing the Trigger and Payload

A. Simulation framework

In order to illustrate the impact of the proposed Trojan trigger and payload mechanism, we have developed a simulation based framework. We employed the MTJ model proposed in [20] and the details of the MTJ parameters are given in Table I.

TABLE I: Parameters of the MTJ

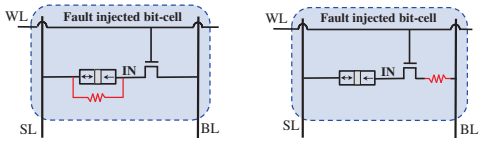
Parameter	Value
Radius	20 nm
Free layer thickness	1.3 nm
Oxide thickness	1.48 nm
Resistance-Area product (RA)	$7.5\,\Omega\mu m^2$
TMR	150%

In our CMOS circuit implementations, we have used TSMC 65 nm general purpose SPICE models with the supply voltage of 1.1 V and temperature 27 °C. The circuit schematics are simulated using *Cadence Spectre* simulator tool. For process variation evaluations, we have considered MTJ and CMOS components separately as these two use different fabrication technologies. For MTJ components, we have used a statistical model that includes variations of *Tunnel Magneto-Resistance* (TMR) and the *Resistance-Area* (RA) product. Whereas, for CMOS components, we have used the statistical model provided by TSMC models.

B. Trigger Mechanism

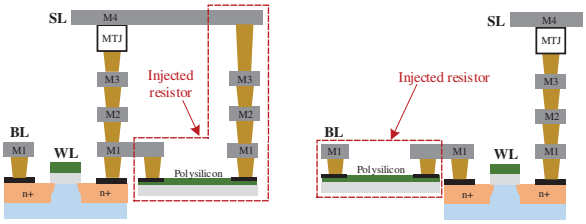
As mentioned in Section III, dynamic faults can be achieved by inserting resistances between various nodes in a bit-cell. A dIRF1 can be achieved by introducing a resistance between the *source-line* (SL) and *internal node* (IN) of the bit-cell (see Fig. 4(a)). Similarly, dIRF0 can be obtained by inserting a resistance in the *bit-line* (BL) (see Fig. 4(b)).

There are several ways a resistive component can be realized during the fabrication process at wafer-scale, such as polysilicon, diffusion, thin-film and ion-implant [21]. Polysilicon resistors are the most commonly used resistors for analog and digital IC systems. In Fig. 5, we have demonstrated how a poly-based resistor can be integrated as a Trojan within a bit-cell. Fig. 5(a) shows the fault case where a resistance is connected in parallel with the MTJ for a dIRF1 fault, whereas Fig. 5(b) shows the case in which resistance is connected in series with the access transistor for dIRF0 fault. A wide range of resistors can be formed using polysilicon with acceptable accuracy, nevertheless, for high accuracy and stability, thin-film



(a) Resistance for dIRF1 (b) Resistance for dIRF0

Fig. 4: Schematic showing resistance insertion between nodes of a bit-cell for dynamic faults.



(a) Resistance for dIRF1 (b) Resistance for dIRF0

Fig. 5: Fabrication of resistances for injected dynamic faults.

TABLE II: Resistance range to sensitize different dynamic faults

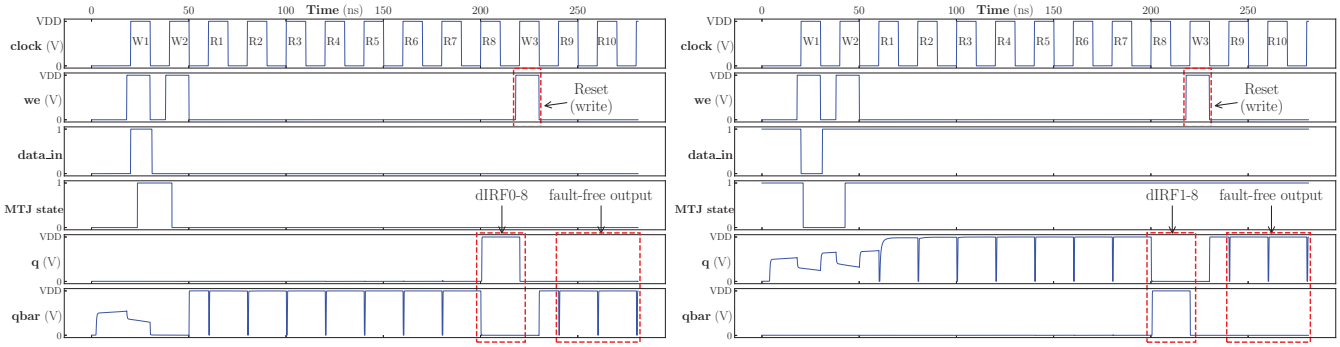
n	Resistance range	
	dIRF1	dIRF0
5	30 kΩ – 43 kΩ	4.34 kΩ – 3.39 kΩ
6	44 kΩ – 57 kΩ	3.38 kΩ – 3.00 kΩ
7	58 kΩ – 69 kΩ	2.99 kΩ – 2.78 kΩ
8	70 kΩ – 79 kΩ	2.77 kΩ – 2.65 kΩ
9	80 kΩ – 88 kΩ	2.64 kΩ – 2.51 kΩ
10	89 kΩ – 95 kΩ	2.50 kΩ – 2.43 kΩ

resistors can be used, where the resistance can be set precisely using a laser-trimming method. Adding a resistance in a bit-cell will increase the bit-cell area, however, the excessive bit-cell area can be improved using a multi-layer polysilicon resistor method [22]. Also, in order to improve the stealthiness, these resistances can be introduced using transistors. For instance, the access transistor in Fig. 5(b) can itself be modified to provide the required resistance, with negligible change in the bit-cell area.

The number of read operations required to achieve the dynamic fault (value of n) can be adjusted by adjusting the value of the injected resistance as presented in Table II. To illustrate our method, we tune the resistance values so that we obtain dIRF0-8 and dIRF1-8 (i.e., n = 8). Higher values of n can be used to further reduce the probability of Trojan detection. From the table, it can be seen that for dIRF0-8, we need a resistance value in the range from 2.65 kΩ to 2.77 kΩ. Similarly, for dIRF1-8, the resistance range required is 70 kΩ to 79 kΩ. Hence, in this work, we use a resistance of 2.7 kΩ to inject dIRF0-8 and a resistance of 75 kΩ for dIRF1-8 faults.

The waveforms for dIRF0-8 and dIRF1-8 are given in Fig. 6(a) and Fig. 6(b), respectively which show that the value of the output (q) flips after 8 consecutive reads to the bit-cell, indicating a dynamic fault. It can also be seen that introducing the resistance does not impact the write operations to the cells since the write operations (W1 and W2) complete successfully by switching the value of the MTJ state. Fig. 6(a) and Fig. 6(b) also present the reset mechanism, where a write operation (W3) resets the Trojan trigger and causes the subsequent reads (R9 and R10) to be fault-free. In order to activate the trigger again, another 8 consecutive read operations are required.

To trigger the Trojan, we need a mechanism to detect the activation of dynamic fault. For this, for certain selected bit-cells, we add a duplicate bit-cell with the resistance introduced to cause the fault. This means that for the bit-cells chosen for Trojan insertion, a corrupted (faulty) cell is camouflaged as redundancy to a paired healthy (fault-free) cell and both these cells would have the same address. To evade detection, an adversary can hide these extra faulty bit-cells as part of the redundant rows/columns which are provided for defect tolerance [23]. Since only a few number of cells are required for Trojan insertion, the extra power consumed during a read/write operation is not easily detectable by side-channel analysis. In addition, the parameters of the Trojan inserted cells can be tuned to minimize side-channel leakage. For a write operation, the data will be written to both the fault-free and faulty cell. For a read operation, the data would be output from the fault-free cell. This means that the data stored in the faulty and the fault-free cell will always match. Please note that the read/write operation always happens to the fault-free cell. Hence, even if a tester is explicitly checking for a dIRF-n fault, the fault would not be observed.



(a) Waveform of dIRF0 fault

(b) Waveform of dIRF1 fault

Fig. 6: Waveforms of dynamic faults. Wn – nth write operation; Rn – nth read operation; we – Write enable signal [when we = 1, write operation happens, when we = 0, read operation happens]; data_in – Input data; q – Read output; qbar – $\overline{\text{Read}}$ output

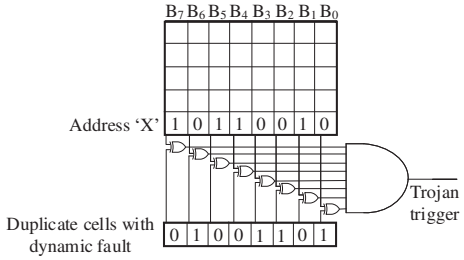


Fig. 7: Trojan trigger generation mechanism using dynamic faults from multiple cells.

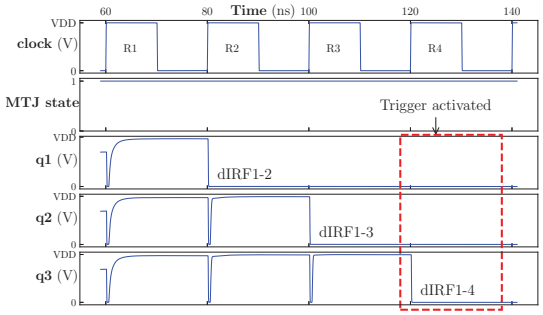


Fig. 8: Trigger signal activation in case of process variation. The trigger signal is activated during R4.

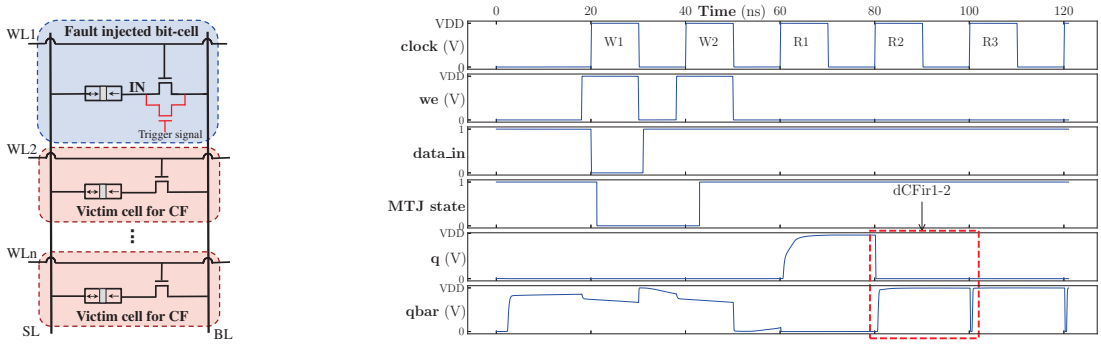
When the dynamic fault happens, the output of the read circuit for the fault-free cell would still be the correct value, whereas the output of the faulty cell will have the wrong value. These two values can be XORed together to generate the trigger signal. The trigger signal will be active as long as there is a dynamic fault in the selected address. When the dynamic fault is removed, for instance by a write operation to the cell, the fault-free and the faulty cells would have the same value at the output of the read circuitry and hence the trigger signal would be deasserted.

The above method can be scaled up by introducing dIRF0 and dIRF1 on different bit cells in a row (as explained in Section III) and the ANDing the XOR output of the different bit-cells as shown in Fig. 7. This would make the Trojan to be triggered only when the chosen memory address is storing a specific data pattern and then a dynamic fault occurs. As shown in Fig. 7, this method requires only a few additional gates, and hence can be implemented with minimal overhead compared to the size of the memory array. These gates can be hidden as part of the memory peripheral circuity by the adversary (untrusted foundry) to evade detection.

As explained in Section III, even with 8 bit-cells chosen for Trojan insertion, there is a very low probability of accidental Trojan trigger. This requires only eight 2-input XOR gates and one 8-input AND gate in addition to the 8 duplicate bit-cells. For a 512 KB memory, the area overhead due to these additional gates and bit-cells is less than 0.01%. The area of the memory was estimated from NVSim [24], whereas the area of the logic gates was obtained from the TSMC 65 nm library. Please note that while estimating the overhead, we have used an area optimized design. For latency or energy optimized designs, the overhead would be much lower.

1) Impact of variations on designed Trojan: The value of the designed resistance for Trojan insertion may vary due to various factors such as manufacturing process variations or aging. To compensate for such effects, the Trojan is designed to be triggered when a dynamic fault occurs for the cell which requires the most number of read operations. This is illustrated in Fig. 8. Here, q1, q2 and q3 are read outputs from 3 different cells at the same memory address having injected dIRF1 faults. However, due to process variation (or aging) in the injected resistance values, some of the bits require 2, some others 3 and the other bits require 4 read operations for the dIRF to happen. In this case, the Trojan trigger will be activated on the fourth read operation (R4) (see Fig 8). Please note that if the fault in the bit-cell is triggered after m consecutive read operations, the subsequent read operations remain faulty until a write operation is performed or the cell is not read again. The simulation results shown in Fig. 8 confirm that these dIRF-n faults are dynamic but persistent.

The variations in the parameters of the MTJ can also affect the value of resistance required to cause the dynamic fault. For instance, for the parameters of the MTJ given in Table I, the resistance value required for dIRF1-8 fault is 75 kΩ. However, if the radius of the MTJ is increased from 20 nm to 30 nm, our analysis shows that a 75 kΩ resistor would instead cause a dIRF1-6 fault. For a dIRF1-8 fault, a resistance of 230 kΩ is required. Therefore, similar to the situation explained above, the value of n is set based on worst case condition, and the trigger is activated to account for such margins.



(a) Injecting coupling fault

(b) Waveform showing the signals of a victim cell affected by coupling fault

Fig. 9: Payload design using coupling fault. The fault affects all cells in a column.

C. Payload Mechanism

We propose a novel payload that can induce malicious effects in the embedded memory exploiting properties of STT-MRAM. We manipulate the memory design to inject coupling faults as payload. A coupling fault can be introduced by having a short between the bit-line (BL) and internal node (IN) (see Fig. 9). This causes dIRF in other bit-cells in the same column. This also reduces the write current to another bit-cell in the same column, thus affecting the write performance by increasing the write latency.

To design the payload, we use a transistor switch connecting the bit-line (BL) and internal node (IN) in a bit-cell as shown in Fig. 9(a). Once the Trojan trigger is activated, the transistor switch would be ON, shorting the bit-line (BL) and internal node (IN) and hence introducing coupling faults in the other cells in the same column (victim cells). This increases the write latency of the victim cell, since a part of the write current is bypassed by the fault-injected cell (aggressor). This can cause the write operation to the victim cell to fail (fault attack). This coupling fault also introduces dynamic Incorrect Read Coupling Fault (dCFir) on the victim cells causing the reads to the victim cells to be faulty (fault attack) as long as the Trojan trigger is active. This is shown in Fig. 9(b), when the second read (R2) from the cell fails when the cell is storing a '1', causing a dCFir1-2 dynamic fault. Please note that this fault does not happen when the value stored in the bit-cell is '0'. In other words, this coupling fault happens only when a victim cell is read at least two consecutive times after the Trojan is triggered, when the victim cell is storing a '1'. This makes it harder for this payload to be detected by Trojan detection techniques. On resetting the trigger, the transistor switch would turn OFF and hence the normal operation resumes. The overhead of designing this payload is negligible since it requires the addition of only a single transistor. Please note that the other payloads proposed for STT-MRAM in [10] would also work with our proposed Trojan trigger since these payloads only require a trigger signal to get activated.

V. Conclusions

In this work, we propose a hardware Trojan for STT-MRAM which makes use of the unique fault mechanisms of this technology. Specifically, we expose how a low-level illicit modification to STT-MRAM bit-cells, based on purposefully implemented dynamic faults, can be used to design a hardware Trojan trigger that is activated after 'n' (variable) consecutive read operations from a specific memory address storing a specific data pattern. The Trojan trigger and payload have minimal footprint, would work even in the presence of variations and can evade industry standard post-manufacturing/validation tests. This work motivates the need for more comprehensive Trojan detection techniques for this emerging non-volatile memory technology.

References

[1] M. Tehranipoor and F. Koushanfar, "A survey of hardware trojan taxonomy and detection," *IEEE design & test of computers*, vol. 27, no. 1, pp. 10–25, 2010.

[2] R. Karri *et al.*, "Trustworthy hardware: Identifying and classifying hardware trojans," *Computer*, vol. 43, no. 10, pp. 39–46, 2010.

[3] R. S. Chakraborty, S. Narasimhan, and S. Bhunia, "Hardware trojan: Threats and emerging solutions," in *HLDVT*, pp. 166–171, 2009.

[4] J. Rajendran *et al.*, "Towards a comprehensive and systematic classification of hardware trojans," in *ISCAS*, pp. 1871–1874, 2010.

[5] H.-S. P. Wong and S. Salahuddin, "Memory leads the way to better computing," *Nature nanotechnology*, vol. 10, no. 3, p. 191, 2015.

[6] A. D. Kent and D. C. Worledge, "A new spin on magnetic memories," *Nature nanotechnology*, vol. 10, no. 3, p. 187, 2015.

[7] A. Chintaluri *et al.*, "Analysis of defects and variations in embedded spin transfer torque (stt) mram arrays," *ETCAS*, vol. 6, no. 3, pp. 319–329, 2016.

[8] S. M. Nair *et al.*, "Defect Injection, Fault Modeling and Test Algorithm Generation Methodology for STT-MRAM," in *ITC*, pp. 1–10, 2018.

[9] K. Xiao *et al.*, "Hardware Trojans: Lessons learned after one decade of research," *TODAES*, vol. 22, no. 1, p. 6, 2016.

[10] M. N. I. Khan, K. Nagarajan, and S. Ghosh, "Hardware Trojans in Emerging Non-Volatile Memories," in *DATE*, pp. 396–401, 2019.

[11] T. Hoque *et al.*, "Hardware trojan attacks in embedded memory," in *VTS*, pp. 1–6, 2018.

[12] Z. Al-Ars and A. J. Van de Goor, "Static and dynamic behavior of memory cell array opens and shorts in embedded DRAMs," in *DATE*, pp. 496–503, 2001.

[13] S. Borri *et al.*, "Defect-oriented dynamic fault models for embedded-SRAMs," in *The Eighth IEEE European Test Workshop*, pp. 23–28, 2003.

[14] T. Hoque *et al.*, "Golden-free hardware Trojan detection with high sensitivity under process noise," *Journal of Electronic Testing*, vol. 33, no. 1, pp. 107–124, 2017.

[15] A. J. Van de Goor and A. Van De Goor, *Testing semiconductor memories: theory and practice*, vol. 225. J. Wiley & Sons, 1991.

[16] K. Yang, M. Hicks, Q. Dong, T. Austin, and D. Sylvester, "A2: Analog malicious hardware," in *IEEE symposium on security and privacy (SP)*, pp. 18–37, 2016.

[17] S. Hamdioui *et al.*, "Importance of dynamic faults for new SRAM technologies," in *The Eighth IEEE European Test Workshop*, pp. 29–34, 2003.

[18] Y.-C. Shih *et al.*, "Logic Process Compatible 40-nm 16-Mb, Embedded Perpendicular-MRAM With Hybrid-Resistance Reference, Sub-μA Sensing Resolution, and 17.5-nS Read Access Time," *JSSC*, 2019.

[19] L. Wei *et al.*, "A 7Mb STT-MRAM in 22FFL FinFET Technology with 4ns Read Sensing Time at 0.9 V Using Write-Verify-Write Scheme and Offset-Cancellation Sensing Technique," in *ISSCC*, pp. 214–216, 2019.

[20] A. Mejdoubi *et al.*, "A compact model of precessional spin-transfer switching for MTJ with a perpendicular polarizer," in *MIEL*, pp. 225–228, 2012.

[21] L. Klibanov, "Integrated Resistors for an Advanced Node CMOS." https://www.eetimes.com/author.asp?section_id=36&doc_id=1330751#, 2016.

[22] C.-H. Su *et al.*, "High resistance polysilicon resistor for integrated circuits and method of fabrication thereof," Dec. 24 1996. US Patent 5,587,696.

[23] S. M. Nair *et al.*, "A Comprehensive Framework for Parametric Failure Modeling and Yield Analysis of STT-MRAM," *TVLSI*, 2019.

[24] X. Dong *et al.*, "Nvsim: A circuit-level performance, energy, and area model for emerging nonvolatile memory," *TCAD*, vol. 31, no. 7, pp. 994–1007, 2012.