USENIX

THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

Be Write Back: An in-depth Study of Fault Injection Effects on FRAM Technology

Valentin Huber and Marc Schink, *Fraunhofer Institute for Applied and Integrated Security (AISEC)*

https://www.usenix.org/conference/woot25/presentation/huber

This paper is included in the Proceedings of the
19th USENIX WOOT Conference on Offensive Technologies.

August 11–12, 2025 • Seattle, WA, USA

ISBN 978-1-939133-50-2

Open access to the Proceedings of the
19th USENIX WOOT Conference on Offensive Technologies
is sponsored by USENIX.

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

Be Write Back: An in-depth Study of Fault Injection Effects on FRAM Technology

Valentin Huber
Fraunhofer AISEC

Marc Schink
Fraunhofer AISEC

Abstract

Ferroelectric random access memory (FRAM) corruption has primarily been investigated in the context of safety for space applications and the associated radiation. In this work, we present the first in-depth analysis of the effects of fault injection on FRAM technology from a security perspective. This includes the identification of potentially vulnerable signals in the control circuit. Based on a theoretical consideration of possible weak points, we carry out practical attacks on external memory devices from different manufacturers. After a detailed analysis, we demonstrate the feasibility of this attack on an FRAM-based microcontroller, namely the MSP430FR. We show how the *write-back* operation, an FRAM intrinsic operation, can be exploited to reactivate the debug interface of the microcontroller. The attack can be carried out in less than a minute and with modest equipment costs, highlighting its practicability. Based on our analysis, we conclude with different mitigations that manufacturers can implement to enhance security.

1 Introduction

In recent years, the Internet of Things (IoT) market has experienced continuous growth, with no signs of slowing down. This surge is due to several factors, including the increasing digitization of previously analog control and sensor elements in smart homes and factories. For example, smart meters for electricity and heating provide substantial benefits to energy providers by significantly reducing labor costs, as there is no longer a need for personnel to visit customer locations to read energy meters. The European Union is also supporting this trend by actively promoting the transition to smart meters [30].

These sensors share a common requirement: they must consume minimal power to avoid burdening their host systems and to extend their battery life for longer maintenance intervals. FRAM-based microcontrollers are particularly well-suited for such ultra-low power applications [14, 22]. Compared to flash memory-based devices, they consume significantly less power when writing data as they operate at lower voltages without requiring charge pumps [19]. Unlike flash memory, FRAM allows for word-granularity access for both reading and writing. It also enables substantially higher writing speeds than flash memory because it does not require an additional erase step. Furthermore, its write endurance is several orders of magnitude higher than that of flash memory [11].

Texas Instruments (TI) recognized this opportunity and launched the MSP430FR family, which utilizes FRAM instead of flash memory. By replacing the memory, this family offers an easy transition for existing systems that rely on flash memory technology. Although FRAM-based devices only hold a small share of the market compared to other technologies, they remain relevant. For instance, in 2022, TI announced an *Authentication IC* [31] that handles sensitive data and uses FRAM memory. Now, three years later, the first evaluation kits for these ICs are available for purchase.

Despite the many advantages of FRAM, the impact on the security of an embedded device must not be overlooked when introducing a new technology. Especially for security-related devices and products, the memory technology may have a major security impact. For flash memory, Schink et al. demonstrated [24] that the erase operation is susceptible to fault injection attacks. The authors show that this vulnerability can be exploited to compromise the security concept of many microcontrollers. FRAM also has potential technology-specific weak spots. One potential attack vector that we practically evaluate in this work is the destructive read mechanism of FRAM.

1.1 Related Work

Due to the aforementioned advantages of FRAM requiring very low power and having a high write endurance, it has been thoroughly studied for usage in space applications [23, 26], where ionizing radiation is a common factor. The effects of heavy ions and laser irradiation on commercial FRAM were

presented in [32]. Most of the arising errors are accounted to disturbances in the peripheral circuit rather than the memory cell itself. In [12] the impact of X-ray beams and heavy ions particularly on the peripheral circuit was investigated. The authors classify the errors in different categories and discuss potential root causes inside the circuit. Recently, Ju *et al.* [18] expanded on the topic by using an ion microbeam to inject failures in the control circuitry of a commercial FRAM IC. They provide a detailed fault model and map the different types of errors closely to the internal operations of the targeted device. These works all have their main focus in ionizing radiation and are more targeted on reliability in harsh environments. Investigations on the impact in terms of security are sparse, only in [16] it is shown that FRAM contents can be corrupted using electromagnetic fault injection (EMFI). The authors observe effects by faulting the memory write operation, but the security impact of those effects is not further examined. Additionally, no fault model is provided that explains what mechanism in the memory is affected by the electromagnetic pulse. To the best of our knowledge, there is no published work that investigates the effects of voltage based faults for off-the-shelf FRAMs.

TI's MSP430 microcontroller family was already attacked successfully in the past. In [15] a timing side-channel was discovered which efficiently breaks the bootloader (BSL) password of the MSP430 family. Additionally, a voltage glitching attack was presented to successfully reactivate a previously disabled BSL during runtime. The timing side-channel affects only certain BSL versions and the fault attack is only applicable to a completely disabled BSL. Bozzato *et al.* demonstrated in [13] that specially shaped voltage glitches can be leveraged to extract data from MSP430 microcontrollers. The glitch skips an authentication check in the BSL, to retrieve single bytes in a sequential manner. For an FRAM based device with a firmware of 8 KiB the extraction requires 50 minutes and roughly 100 k glitches.

Yet, neither of these attacks leverages the unique properties of FRAM. This work aims to address this gap.

1.2 Contributions

In this paper, we introduce the first detailed study on the effects of voltage glitches on off-the-shelf FRAMs. Contrary to previous works, we analyze chips from different manufactures. Vulnerable signals are identified inside of the memory control logic, and the fault results are associated with them. We identify the *write-back*, an FRAM intrinsic operation, as a weak spot. We demonstrate that the write-back vulnerability can be exploited in real-world systems. It allows an adversary to unlock the JTAG interface of an MSP430FR5962 microcontroller in less than a minute. Finally, we show that FRAM technology can pose a security risk for embedded products when implemented without careful consideration.

2 Write-Back Vulnerability

In this section, we describe the inner workings of an FRAM cell. We focus on the destructive read operation and the required write-back mechanism. Control signals that are crucial for memory integrity are elaborated and evaluated to their susceptibility to fault attacks.

2.1 Ferroelectric Capacitors

FRAM utilizes ferroelectric capacitors to store data. These capacitors use a ferro-electric material as their dielectric that has two stable polarization states. This is usually achieved by using lead-zirconate-titanate (PZT) crystals. These crystals have a movable zirconium or titanium atom that is held in an equilibrium state by surrounding oxygen atoms. If a strong enough external electrical field is applied, the movable atom overcomes the energy barrier in the center and moves in its second stable state, again fixated by the oxygen atoms [22]. The dipole of the structure is then aligned with the external field. This also works in the opposite direction by inverting the polarity of the electrical field, as the crystal is symmetric. Figure 1 shows the structure of a PZT crystal with an electrical field applied.

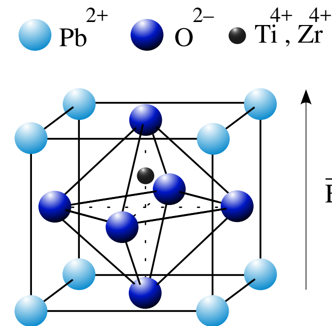

Figure 1: A PZT crystal gets polarized by an external electrical field. Adapted from [21].

FRAM cells do not lose their polarization state over time and consequently do not need constant refreshing like DRAM. Even after switching off power, the PZT holds its state, making it non-volatile.

2.2 Memory Cell

Two different base architectures exist for FRAM: the 1T-1C and 2T-2C structure [25]. The former is equivalent to a DRAM architecture where one transistor and one capacitor are connected. The evaluation of the memory cell is derived by the polarization state of the single capacitor. However, to our knowledge, most commercially available FRAM ICs use the 2T-2C structure. Here, two 1T-1C cells are combined into

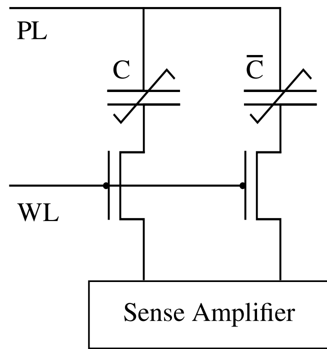

Figure 2: Example of a 2T-2C FRAM memory cell with word-line (WL), plate-line (PL) and two complementary ferroelectric capacitors (C, $\overline{C}$).

a single cell, in which the information is stored in a complementary manner. This halves the memory density, but doubles the signal margin and thereby increases reliability and noise immunity. An example of the 2T-2C structure is illustrated in Figure 2. Similar to other memory technologies, FRAM is arranged in memory rows to reduce the number of required sense amplifier circuits. Sense amplifiers are responsible for transferring the raw analog signals from the cells into the digital domain. To access single memory rows, a corresponding word-line is activated to access the cells, as seen in Figure 2.

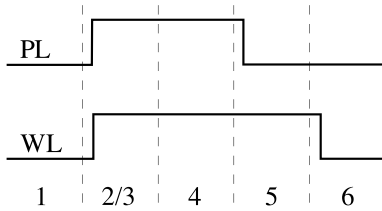

Figure 3: Plate-line (PL) and word-line (WL) transitions during the read-out of an FRAM cell.

Memory Read To read the cell's value, a sense amplifier is connected to one terminal on both capacitors. Then the following six steps [17, 25] are performed:

1. **Precharge** After each read, the sense amplifier has latched the value from the memory cell and is in a stable state. To read a new cell both amplifier inputs must be forced to the same level, as the energy from the cells is not sufficient to overwrite the previous state of the buffer. For this example the signals are pulled to ground. Once this process has finished, the inputs are left floating, setting the amplifier in an unstable state that is sensitive to minuscule differences of the input signals.

2. **Row Access** After the precharge phase is completed, the word-line is pulled high to access a memory row. This connects one terminal of each of the complementary ferroelectric capacitors to the inputs of the sense amplifier.

3. **Polarization** At the same time the word-line is activated, the plate-line is pulsed. As the plate-line is connected to the other terminals of the capacitors and the precharge phase pulled the amplifier wires to ground, an electric field is applied across the capacitors. This forces both PZT crystals to polarize in the same orientation. Due to the complementary architecture, one crystal is already polarized in alignment with the field. The second crystal undergoes a polarization change that pushes charges onto the amplifier wiring. The row access and the polarization stage must happen quickly after the precharge, as otherwise the sense amplifier can drift towards a stable state due to leakage currents.

4. **Sense** Let Y be the terminal of the sense amplifier that is connected to the crystal that switched polarization and Z be the remaining terminal. Due to the additional charges on terminal Y, the strength of the signal exceeds that of terminal Z. For this reason, the amplifier latches the memory cell's content and moves to a stable state. In this state the amplifier actively drives terminal Y high and terminal Z low. The memory content is now available in CMOS logic and the actual reading is completed.

5. **Write-Back** Due to the single polarization change, both ferroelectric-capacitors are now polarized identically, resulting in an undefined state of the cell. That mechanism is the reason for the destructive read of FRAM. To restore the original state and allow for further reads, the polarization of the affected crystal must be reversed. This is achieved by pulling the plate-line to ground while keeping the word-line high. An electric field is generated across the flipped capacitor, as the sense amplifier is still driving its output high. The resulting field is opposed to the polarization of the cell, thus switching its state back to its initial pre-read state. The complementary cell experiences no electric field and keeps its state, as both terminals are pulled to the same level. After that process, the memory cell is fully restored.

6. **Close** Finally, the memory cell is closed by pulling the word-line low, allowing the sense amplifier to be reused for operations on further cells.

Memory Write In order to write a value to the memory cell, the same steps as for the read mechanism are performed. The only difference is that the precharge phase is skipped and instead the sense amplifier is forced to the state that represents the new value. Now following the remaining steps (2-6) from the read process, a new value is written to the cell. The *sense* step has no effect here, as the amplifier is already in a latched state.

2.3 Vulnerability

Three potential vulnerabilities that manipulate memory during a read access can be identified in the FRAM control flow. The focus of this work is on memory reads, because writes in FRAM involve only a subset of the steps from the reading mechanism. Additionally, attacks on memory writes do not stand out from those targeting other memory types.

- **Precharge Suppression** The precharge works by adding two transistors that connect the differential inputs of the sense amplifier to ground when precharge is enabled. If the enable signal can be suppressed or delayed, a read operation is behaving like a write operation. Interestingly, the resulting value of the memory cell is then only dependent on the previous read or write operation, as the amplifier stays in its latched state. This allows for precise control of the targeted memory cell value at the cost of a complex attack, as a single signal must be suppressed completely within tight timing constraints. The timing is restricted to step 1, as visible in Figure 3. It is possible that the precharge mechanism is activated after every read and it only gets disabled with step 2. If the disable signal is delayed, a timing violation with the plate-line occurs, resulting in the same effects as described below in *plate-line timing violation*.

- **Sense Amplifier Bit-Flip** As the sense amplifier works like a latch, it should be possible to flip its state with, e.g., EMFI. This approach is similar to inducing bit-flips in SRAM. It offers less control of the outcome compared to the precharge suppression. The fault timing is between step 2 and 4 of the read process.

- **Plate-Line Timing Violation** The plate-line, as visible in Figure 2, plays a crucial role for the read and restore mechanism as previously explained. There is a time constraint together with the word-line. The plate-line must be pulled low before the word-line is pulled low to ensure a proper write-back. Additionally, there must be enough time between the signal transitions to allow the PZT crystal to change its polarization. The timing of both signals can be seen in Figure 3, step 5 and 6. If the plate-line signal is delayed or skipped, the cell can close without completing the write-back step, leaving it in an undefined state where the complementary cells are polarized in the same direction. On the next read, the sense amplifier experiences two identical signals on its inputs, thus miniscule differences determine which input is going to dominate the other. Effectively, the following read value is mostly determined from the physical layout and the manufacturing variations of the sense amplifier circuit that slightly favors one input. From an attackers point of view it is difficult to control the actual outcome of the memory corruption from a plate-line timing violation, as it is inherent to the individual device. However, one major advantage is that closing the cell is equivalent with cutting the power to the cell or the complete device. That implies, that if an attacker can control the power of the device, he can effectively control the timing of step 6 from the read mechanism and violate the plate-line timing, causing memory corruption. The timing for the fault is right after the initial polarization happens (step 3) and lasts until the original polarization is restored (step 5).

This work focuses on the plate-line timing violation. The reason is that the vulnerability has an advantage of being non-transient which is unique to FRAM. If the device crashes or resets, the already corrupted memory is not restored. This is advantageous for the attacker for two reasons. First, the time required to mount an attack is significantly reduced, as the device is allowed to crash or reset during the fault, which reduces the complexity of the parameter space. An attacker has not to worry that the device keeps running after the fault. Second, voltage supervisors or glitch detectors that cause the device to reset if abnormalities are registered may be ineffective as the fault persists. The precharge suppression does not have this benefit, as during the precharge phase, the memory is still intact. It is to note that the bit-flip vulnerability and the plate-line timing violation cannot be differentiated for attacks that do not crash the device, as they have overlapping timing windows. For devices that do experience crashes, the faults are attributed to the plate-line timing violation, as flipping bits in the sense amplifier is only effective if the read mechanism can complete successfully.

In the following sections, we assess how the plate-line timing violation may pose threats to the security of real-world devices.

3 Experimental Setup

The following section describes the equipment and software that is used to perform the investigation on the external FRAMs as well as on the MSP430FR microcontroller.

Hardware To perform voltage glitching on the external FRAM devices and the microcontroller, a CW1200 ChipWhisperer-Pro from NewAE Technology is used. It provides an SMA port for crowbar glitches, as well as GPIO pins that can be used as trigger inputs. As the ChipWhisperer hardware allows for an adjustable delay between the trigger and the glitch, no additional delay generator is necessary to control the timing of the faults. The communication with the CW1200 is achieved through USB and the official Python software library. In order to communicate with the external memory, a USB to SPI adapter is necessary. For this analysis, we use a J-Link Plus Compact from SEGGER in combination with a custom software library derived from flashrom [2].

To ensure comparability between all FRAM devices, they are all soldered to identical custom printed circuit boards

(PCBs) featuring connectors for a power supply, the Chip-Whisperer GPIOs and the J-Link. Additionally, an SMA port on the power rail is established to enable crowbar glitches with the CW1200. The hardware setup is completed with a programmable laboratory-grade power supply. It can be controlled from the host computer to power cycle the target chip automatically.

We use the ChipSHOUTER from NewAE Technology for the experiments regarding EMFI. It is priced at 4500 USD and fits various types of coils while delivering voltages of up to 500 V. For all tests it is equipped with a 4 mm coil that has a counter-clockwise (CCW) direction. Like for the voltage faults, the CW1200 is responsible for managing the trigger. In order to place the coil of the ChipSHOUTER precisely above the target device, it is fixated on a 3-axis positioning table. We use a STEPCRAFT D-300 as our positioning table, which is controlled programmatically through software and costs around 1500 USD.

To program and debug the microcontroller, we repurpose a MSP-EXP430FR2311 LaunchPad development board, implementing an ezFET debugger. For the attack on the MSP430FR only the 3-axis table and the ChipSHOUTER are necessary. The CW1200 can be replaced through a cheap FPGA development board to generate the trigger. Suitable boards are sold for less than 100 USD. The complete attack setup costs less than 6500 USD.

Software All of the hardware components are controlled by a host computer running a custom software framework. The procedure varies for the external memories and the experiments on the microcontroller. For the external FRAMs the MOSI signal of the SPI connection is used as a trigger. One run of the analysis then performs the following steps. First, the FRAM is power cycled to set repeatable starting conditions. Next, the targeted memory row is written with either all zeros or all ones. The row is then read to verify that the content is correct and that the chip is still functioning properly. Afterwards, the chip is again power cycled to circumvent a potential caching of the just read row. Finally, the glitch setup is armed and a read command for a single byte is issued on the target address. After the fault injection, the device is power cycled again and the memory row is read in order to check if it got corrupted. This procedure is repeated for each memory row to obtain a heatmap indicating corruption for the complete address space.

The process for the MSP430FR is similar, however it must be distinguished between the vulnerability analysis and the actual attack. For the analysis we use a custom firmware on the microcontroller that establishes an UART connection with the host computer. As first step, the device is reset through its hardware pin. If the fault setup is ready and configured, the computer instructs the microcontroller to start its operation. After the fault injection, the memory integrity is reported back to the computer through a checksum. Each checksum is then compared to a reference. If a mismatch is detected, the relevant memory is dumped through the debugger for further inspection. Finally, the firmware is flashed again on to the device to ensure a clean state. We control the debugger with MSPDebug [7].

For the attack on the debug protection, no communication or firmware is required. The steps are reduced to an initial reset through the reset pin, which also serves as a trigger, and the debugger's attempt to connect to the device. The fault is evaluated based on the fact if the debugger can connect successfully or not.

4 External Memory

Table 1: Overview of the analyzed external FRAM devices.

Device	Size	Organization	Manufacturer
FM25L16B	16 KiB	256 x 64	Infineon
CY15B016Q	16 KiB	256 x 64	Infineon
MB85RS16	16 KiB	n/a	RAMXEED
MR45V032	32 KiB	2048 x 16 [1]	ROHM Semiconductor

In this section, we examine four distinct external FRAM devices from different manufacturers. Table 1 lists the evaluated devices and their properties. The organization column refers to the physical organization of the memory. That means the number of physical memory rows multiplied by the row size in bits. In contrast, the logical organization that is usually given in the datasheet, relates to the number of bits a single address represents. The logical organization is 8-bit per address for all tested memories. If one address is accessed, the complete physical memory row containing, e.g. 64 bits, is read internally, but only 8 bits are returned.

Most devices are available with either an SPI or an I2C interface. For better comparability, only devices that implement their communication via SPI are selected. The devices operate within a comparable supply voltage range and are all designed to function effectively with the chosen supply voltage of 3.3 V. All tested devices implement a status register that can be configured to enable write protection for up to four sections, which together cover the complete memory. The write protection is enabled on all segments for the experiments in order to rule out other fault origins like unintentional writes from the control logic. For each device in the study, five samples are tested. The fault injection process is repeated 16 times for each memory row to gather statistical data.

Parameter Space The analysis on the external FRAMs focuses on simple and inexpensive voltage glitches using a

[1] Retrieved based on the results of the fault-analysis, as no specifications are available in the datasheet [5].

Table 2: Flip probabilities for a random selected bit on each tested FRAM device, averaged across all samples.

Device	Flip probability (%)	
	$1 \rightarrow 0$	$0 \rightarrow 1$
FM25L16B	49.6	7.3
CY15B016Q0	71.3	2.8
MB85RS16	0.0	0.0
MR45V0320	9.4	22.9

crowbar-circuit. A crowbar-circuit shorts the supply rail to ground, to cause a quick voltage drop and potentially inject a fault into the target device. For these types of glitches the most crucial parameters are timing, pulse length and glitch repetitions. In the specific case of the *plate-line timing violation*, the parameter space can be simplified to focus solely on the timing parameter. The reason for this is that we do not care if the device crashes or not, as the fault persists. Consequently, a single glitch with a pulse length long enough to shutdown the target is sufficient. Further system parameters like the impact of supply voltage and alike are not in scope for this work.

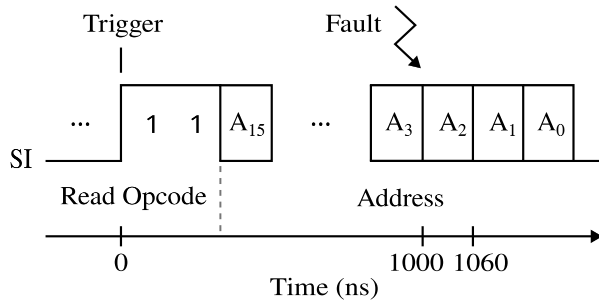

Figure 4: Timing diagram of the MOSI line issuing a read command with SPI frequency set to 15 MHz. The signal transition in the opcode is used as a trigger.

SPI Protocol All devices employ the same control flow and commands for their SPI interface, enabling the use of identical tooling for communication with all memories. To perform a read operation, the 8-bit opcode `0x03` is transmitted first. Subsequently, the device expects a 16-bit wide address indicating the location to read from. Once the address is received, the device returns the memory content from that location in the following clock cycle. Depending on the memory size, not all 16 available bits are needed for the address. For instance, memories sized at 16 KiB only require 11 address bits. The least significant bits are transmitted last, which creates a tight time constraint for the internal FRAM access. This is because a device begins transmitting the data immediately after the last address bit is received. As the tested memories support SPI speeds up to 20 MHz this would result in a maximum time of 50 ns to process the last address bit and perform all steps involved with reading from the FRAM array. This contradicts with FRAM access times, which have a comparable duration [14]. Hence, it is likely that the access to the memory row happens already before the complete address is received. That is possible if the memory rows are larger than the logical address organization. Such behavior is also explained in an user guide [8] of the predecessor from the FM25L16B memory. With Equation (1) it is possible to calculate the earliest point in time t_{acc} in which the row may be accessed:

$$t_{acc} = [N_{cc} - \log_2(\frac{B_P}{B_L}) - 1] \cdot \frac{1}{f_{spi}} \quad (1)$$

N_{cc} represents the total number of SPI clock cycles until all address bits are transmitted. B_P denotes the number of physical bits per row in the memory organization, while B_L indicates the number of bits per address. f_{spi} refers to the frequency of the SPI communication.

Since the MOSI line is utilized as the trigger source, the trigger activates two clock cycles before the address transmission begins, caused by the logic transition in the read opcode. Consequently, for a 16-bit address, N_{cc} is fixed at 18 clock cycles for our setup. To ensure comparability in the analysis, the SPI frequency is set to 15 MHz for all tested devices. This is the maximum frequency determined by the slowest device. Given that all devices store 8-bit per address, the introduced formula can be used to calculate the time between the trigger firing and the memory row being accessed earliest, if the physical row size B_P is known. This knowledge can be used to narrow the window for identifying effective timings for fault injection. The process is visualized in Figure 4.

FM25L16B The FM25L16B [3] from Infineon is a 16 KiB sized FRAM fabricated with 130 nm technology [29]. It has an SPI interface that supports speeds up to 20 MHz. A single physical memory row contains 64 bits. Using Equation (1), this translates to an offset of roughly 930 ns after which the row can be accessed. Sweeping the fault timing reveals that memory corruption starts happening at 930 ns and stops after 1180 ns, resulting in a window of 250 ns. This confirms that the cell is indeed accessed after the received address is sufficient to resolve the memory row. For all experiments regarding this device, the timing is fixed to 1000 ns after the trigger is fired from the read opcode, as depicted in Figure 4.

Figure 5 shows two heatmaps with average bit-flip probabilities for all five samples. The upper heatmap (a) represents the case that the initial value is one and it flips to zero. The lower heatmap (b) presents the inverse case. Every column of the heatmap relates to a single memory row containing 64 bits. The plots include all available bits from the memories. Further, it is visible that faulted devices overall prefer to reset their bits to zero. In fact, the flip probabilities are significantly higher if the initial state is one with 49.6 % compared to 7.3 %

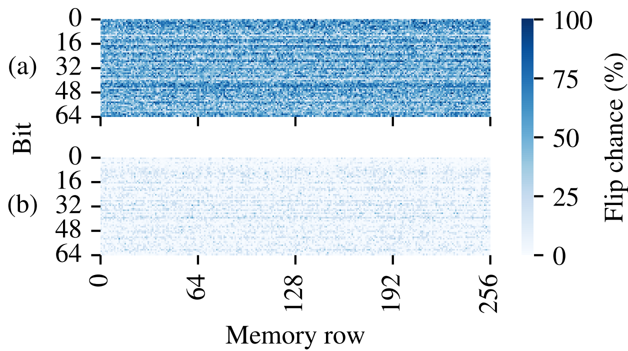

Figure 5: Average bit-flip probabilities for each individual bit, calculated across all tested FM25L16B devices. Upper heatmap shows flip direction 1 → 0. Lower heatmap shows flip direction 0 → 1.

for the inverse case, as stated in Table 2. Furthermore, the flip chances are dominated by the bit position in the memory row. This is evident by the appearance of horizontal features in the heatmaps that stretch across all memory rows. The row location only plays a secondary role on influencing the probabilities.

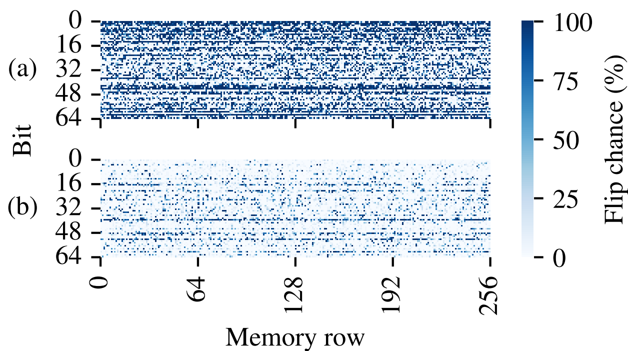

Figure 6: Bit-flip probabilities for each individual bit in the FM25L16B, based on sample device 1. Upper heatmap shows flip direction 1 → 0. Lower heatmap shows flip direction 0 → 1.

This gets even more evident by having a look on the bit-flip heatmap of a single sample, as visible in Figure 6. There, the bit position clearly determines the flip chance, with values approaching 100 %, displayed as distinct horizontal stripes. Most of them prefer to reset the bit, however there are a limited number of positions that favor to set the bit to one. This sample deviates slightly from the average with a flip chance of 42.4% if its initial state is one and a flip chance of 11.9% if its initial state is zero, for a randomly selected bit. All tested samples show a similar behavior with the appearance of those horizontal features in the heatmap and the tendency to reset bits. The individual flip probabilities and plots for all samples are given in the Appendix A. By using the fault model defined in Section 2.2, this behavior can be explained by the undefined state of the memory cells after a plate-line timing violation. The device appears to implement a single set of sense amplifiers, that is reused for each memory row, as the horizontal stripes are consistent across the entire memory without interruptions. The stripes change their bit positions for the individual devices, as they are dependent on the intrinsic device properties caused by manufacturing tolerances. However, as the averaged heatmaps in Figure 5 also show consistent features across memory rows, this indicates that the hardware layout also influences the tendencies of some sense amplifiers.

CY15B016Q Manufactured by the same company, the CY15B016Q [1] shares very similar properties with the FM25L16B. The main differences are the lower maximum SPI speed with 16 MHz and an extended temperature operating range. It is also fabricated with a 130 nm process node [27] and implements 64 bits per memory row. Hence, the timing is consistent with that of the first device, allowing the earliest row access to occur after 930 ns. A sweep of the fault timing reveals a glitch window of the same size, measuring 250 ns, starting from 930 ns. This suggests that the write-back mechanism is executed late. Again, a delay of 1000 ns is chosen for the trigger.

The similarities extend to the fault injection results, which display characteristics close to those of the FM25L16B, expressed through the presence of horizontal stripes. Consequently, the CY15B016Q assumably also implements only a single set of sense amplifiers that is shared by all memory rows. As listed in Table 2, the device prefers to reset bits, with a bit-flip probability of 71.3 % from state one to zero. The inverse transition from zero to one has a significantly lower probability of 2.8 %. This tendency to prefer the reset state is also observed for the FM25L16B, although to a lesser extent.

Just like the FM25L16B, all examined devices exhibit the horizontal stripes in their heatmaps on different locations for the individual samples. The data for the averaged flip chances looks similar to the data from the previous analyzed device, exhibiting a bias for certain bit positions across all samples. The results can be attributed to manufacturing tolerances and the hardware design, as reasoned for the first device. The individual heatmaps are available in the Appendix A.2.

The CY15B016Q shares similarity with the FM25L16B in almost all regards. This is reasonable as they are both manufactured from the same company. As they also have the same fault timings and the noticeable long window in which faults are effective, their hardware designs are likely to be similar. Nevertheless, they exhibit different probabilities for their bit-flips confirmed by multiple devices. This indicates that their designs are similar but not identical.

MB85RS16 No parameter set was identified that would corrupt the memory of the MB85RS16 [4] from RAMXEED.

This can be for multiple reasons. One reason could be that the device's internal capacitance is sufficiently large to dampen the crowbar glitch, rendering it ineffective. It is to note that all tested devices are mounted in the same manner and no external capacitors are used. In combination with the capacitance, an other reason could be that the time resolution of the fault injection setup is too coarse. However, the resolution of 10 ns should be sufficient for FRAM access times in the two digit nanosecond range. Furthermore, it is possible that the device implements countermeasures against sudden power disruptions. The motivation for this is maybe not to protect it against fault attacks but to ensure data integrity in environments with unreliable power sources. Finally, it cannot be ruled out that RAMXEED uses a different technique to read values from the memory, deviating from the approach described in Section 2.2, making it less susceptible or even immune to crowbar glitches.

Given its immunity to voltage faults, we also conduct tests using the EMFI setup on this chip. However, just like with the crowbar glitches, we were unable to identify any parameter set that alters the memory content. This finding suggests that the ineffectiveness of voltage faults is unlikely due to a dampened glitch signal.

MR45V032 The MR45V032 [5] from ROHM Semiconductor is the only tested chip with a capacity of 32 KiB. It is not available with 16 KiB density, but we want to include the manufacturer in the analysis. Comparing the memory to the other examined devices, it has the slowest operating speed with a maximum SPI clock frequency of 15 MHz. The physical organization is not described in the datasheet, however bit-flips only occur in a 16-bit boundary for single addresses. For that reason, we assume a memory row size of 16-bit.

Two different fault types are observable for this FRAM. First, failures caused by the crowbar setup with a glitch length of 200 ns lock-up the control logic. It is possible that other fault lengths also show this effect. However, this is not further evaluated as it is not in scope for this work. If the control logic locks up, consecutive reads will write a random value into the memory, independent from the address or the length of the read operation. That indicates that the sense amplifiers are not precharged correctly after the fault. Even writes will result in the same random value, thus the device does not update its amplifiers. As writes will not update the internal state, it is not possible to control the fault results from an attackers perspective. The effectiveness of the fault depends on the address that is read from. Not all addresses cause a lockup of the control logic. If the glitch time gets prolonged more, the chip is powered down and resets correctly.

The second fault type is a plate-line timing violation like the previous devices showed. The timing of the fault is more strict and it is only effective at 1060 ns after the read opcode is transmitted. This timestamp matches with Equation (1) if the physical row organization is set to 16 bits and thus supports the assumption of 16-bit wide memory rows. In contrast to the other tested devices, the short glitch window suggests that the write-back happens immediately after the row was accessed.

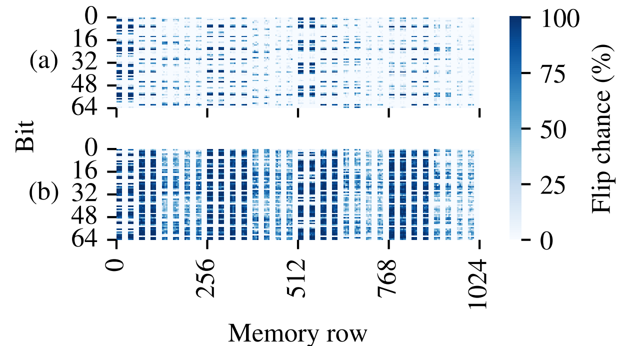

Figure 7: Bit-flip probabilities for each individual bit in the MR45V032, based on sample device 1. Upper heatmap shows direction $1 \rightarrow 0$. Lower heatmap shows $0 \rightarrow 1$.

Figure 7 shows the bit-flip probabilities for the plate-line timing violation at 1060 ns. Only half of the memory is used to mimic the same amount as the smaller FRAMs. For the sake of readability, always four 16-bit long memory rows are concatenated in the plot. Interestingly, the effectiveness of the fault depends strongly on the address. For every 64 addresses, only the first half is affected by the glitch, represented by the blank columns with 0 % glitch probability. This holds true for all tested devices, so it is not based on manufacturing tolerances but on the chip design. One reason for this could be that the physical layout delays the voltage drop for certain memory segments long enough, so that the write-back can finish correctly. Additionally, some address spaces exhibit a higher flip chance in both directions indicated by the memory blocks with darker color in the figure. That suggests, that in contrast to the other FRAM devices, this device implements multiple sets of sense amplifiers assigned to different memory regions. It is noteworthy that the device prefers bits to be set in state one, unlike the other FRAMs.

The individual heatmaps for all five MR45V032 samples are available in the Appendix A.3. Compared to the other tested devices they are less likely to flip any bits with average flip chances of 9.4 % if the initial state is one and 22.9 % for the inverse case, as presented in Table 2. The reason for this is that half of the memory is not affected by the fault injection, significantly reducing the overall flip probabilities. As the immune addresses are consistent across all samples, the flip chances double, if only the vulnerable memory rows are targeted.

Countermeasures Countermeasures may not be too relevant for external memories, however the same actions can be applied to all devices which incorporate FRAM technology. One measure is to reduce the time frame in which the de-

vice is vulnerable to glitches as much as possible. Two of the tested memories have a considerably large time window in that effective faults can be applied. We assume that the reason therefore is, that the write-back does not happen immediately after a row is read, but only once the remaining address bits are received. To address this, the design should implement a row buffer that stores all bits until the address is resolved fully and perform the write-back without a delay. The behavior of the MR45V032 chip matches this implementation. As shown, this countermeasure only mitigates the most basic attacks, however due to its negligible hardware overhead, we deemed it noteworthy. To close the timing window for crowbar glitches, a backup capacitor combined with a voltage supervisor could be implemented. This ensures that the read process can finish uninterrupted, despite external voltage fluctuations. Integrating backup power will however increase die area and thus drive cost. Other than that, common countermeasures against fault injections can be applied like error correcting codes (ECCs) or glitch detectors. It is to note that simple glitch detectors which only cause a reset are not able to protect the device, as the injected fault is persistent even after power is removed.

5 MSP430FR Microcontroller

Analyzing the fault injection behavior of external FRAM devices provides a better understanding of the underlying fault model and what can be achieved. However, it is only of little relevance in terms of security, as the devices lack any security features. If an attacker can access such a memory, he can simply reconfigure it through the unprotected SPI interface. To emphasize the security implications we move on to a more realistic target, the MSP430FR5962 microcontroller. The MSP430FR family from TI implements FRAM as its internal memory.

5.1 Attacker Model

This section describes the attacker model that is used for the analysis on the MSP430FR. The following states the competencies and the goal of the assumed attacker:

- **Time** We consider a scenario where the attacker has physical access to the target for at least a couple of hours. A dummy device for calibrating the attack setup is available, thus reducing attack time.
- **Technical Knowledge** The attacker has sufficient technical knowledge about the device under target and is able to control an EMFI setup.
- **Expenses** The targeted microcontroller resides more in the general-purpose category rather than being a dedicated security chip. Nevertheless, it implements an AES hardware accelerator and security features for IP protection, which makes it reasonable to assume that it handles sensitive data. This can be in the form of secret IP or cryptographic keys that need to be protected. Those render it an attractive target if the cost for an attack is low. For this reason a budget in the realm of a few thousand USD is set for the attacker.
- **Objective** The objective of the attacker is to either steal secrets of the device or manipulate it. For example in the case of industrial smart-meters, manipulation can be used to decrease meter readings for profit.

5.2 Device

The MSP430FR5962 [6] is advertised as an ultra low-power mixed-signal microcontroller with a maximum clock frequency of 16 MHz. It is equipped with 128 KiB of FRAM with a maximum access rate of 8 MHz. Additionally, it has 8 KiB of SRAM available. Variable data and the stack resides in SRAM, however in theory it is possible to place everything in FRAM. The 2T-2C type memory is organized into rows, each packing 64 bits of data. It features single-bit error correction and multi-bit error detection. As the CPU is able to clock faster than the FRAM, four 64-bit cache lines are available to reduce wait states.

To program and debug the device there exist two options. The first option is to use spy-by-wire (SBW) or JTAG. Internally the SBW signals are translated to JTAG so these can be seen as equivalent [10]. For the MSP430FR5 and MSP430FR6 series access to JTAG can be restricted by configuring two 16-bit signatures. These signatures reside in FRAM memory. Writing 0x5555 to both signatures locks the interface completely and writing 0xAAAA to the first signature allows to use the interface with a password that can be set by the user. All other values unlock the JTAG functionality without restrictions.[2]

The second option is to access the device through the BSL which resides in the ROM of the microcontroller. After a special entry sequence on the reset pin, an UART connection can be established to the BSL. With that connection the device can be controlled in a similar fashion to JTAG, e.g. memory can be directly accessed. To protect the device from manipulation, the BSL is always either password protected or disabled. This behavior can be configured through a signature. Setting the 32-bit signature to all bytes being 0x55 disables the BSL, setting it to any other value than all bytes being 0xAA triggers a mass erase, if a wrong password is provided.

[2] According to the datasheet [20]. In practice, similar values also result in the locking of our sample device.

5.3 Vulnerability Analysis

In this section, we evaluate if the microcontroller is vulnerable against the write-back glitch identified for the external memories.

Firmware In order to analyze if the FRAM reads of the microcontroller are affected by fault injections, we deploy a custom firmware that reads a single byte from a fixed address in FRAM. Upon startup, the firmware calculates a CRC16 checksum from the part of main memory where the firmware resides, as well as from the target address for the read operation. The result is transmitted to the host computer, where the checksum of the original memory is saved as a reference. As the CRC16 algorithm accesses the relevant memory locations, it is possible that the targeted memory row gets stored in cache. To prevent a potential cache hit during the analysis, multiple words outside of the analyzed memory segment are read. The firmware continues by raising the voltage level of a GPIO pin to supply the trigger for the fault injection setup. Then the actual read from FRAM is performed with a `mov` instruction targeting a fixed memory address. The instruction is surrounded with a `nop` slide to account for any delays in the setup. Afterwards, the trigger pin is released to provide visual feedback on an oscilloscope, indicating the time frame for searching the parameter space.

Voltage Glitch No working parameter set was identified that manipulates the memory content through voltage glitches. Previously published work [33] on a prototype of the MSP430FR series implements an internal backup capacitor together with a voltage supervisor that disconnects the capacitor in case the supply voltage changes too quickly or drops below a threshold. This mechanism is intended to ensure that ongoing writes to the FRAM can finish even if the power supply fails suddenly. As our target device is also intended for low power scenarios it is likely that it also implements a backup capacitor that mitigates simple crowbar-style voltage glitches. However, more sophisticated approaches [13] have shown that at least the CPU can be manipulated with specifically shaped voltage glitches.

Electromagnetic Fault Injection As voltage based faults show no effect, we move on to using EMFI. All tests are carried out with a coil voltage of 500 V, effects start to arise with 490 V. Figure 8 shows the effectiveness of the EMFI pulse based on the placement of the coil. The grid scan spans an area of 10 mm x 10 mm which is identical to the package dimensions of the chip. The step size is selected as 1 mm, as it evenly divides the package area into 100 locations. For the starting location (0, 0) the center of the coil is placed on the top left corner of the microcontroller package, where pin number one is located. Firmware, as previously described in this section is used to evaluate the faults. Once the ideal timing is identified, each location is exposed to the electro-magnetic pulse 16 times to retrieve the data for Figure 8. The resulting glitch probabilities are rounded down to the next tenth place for a conservative estimate. It is visible that locations in the top left corner have a probability of up to 100 % to cause memory corruption. Placing the coil above other sections of the package has no impact except for location (0, 8) which has a probability of 30 %. Within that area, there is a supply pin that may picks up the electromagnetic pulse via the bond wire, potentially causing a glitch.

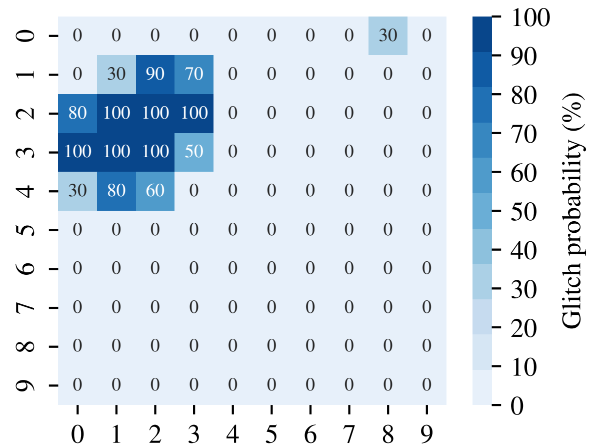

Figure 8: Effectiveness of the EMFI setup based on the placement of the coil. Pin one of the microcontroller is in the top left corner.

```
;TRIGGER HIGH
_memory_row_0: nop
_memory_row_0: nop
_memory_row_0: nop
_memory_row_0: nop
_memory_row_1: nop
_memory_row_1: nop
_memory_row_1: mov &_memory_row_8, R0
_memory_row_2: nop
_memory_row_2: nop
_memory_row_2: nop
_memory_row_2: nop
_memory_row_3: nop
_memory_row_3: nop
;TRIGGER LOW
        ⋮
_memory_row_8: 0x5555
```

Listing 1: Targeted segment of the firmware that is used to analyze the effects of fault injection. The instructions are marked with their physical memory row location, relative to the first instruction.

With the sensitive areas identified, we can analyze the impact of the fault timing on the evaluation firmware. The targeted section of the firmware is displayed in Listing 1. It outlines the assembly instructions executed during the trigger window and specifies the arrangement of these instructions in consecutive memory rows. One memory row can store four 16-bit instructions. The `mov` instruction with a fixed address requires 32 bits.

Figure 9 shows which memory rows are affected, depending on the timing of the fault injection. Single rows exhibit a glitch window of 30 ns in that faults are effective. Interestingly, not only the targeted 8th memory row, from which the microcontroller is instructed to read, is influenced, but also the rows containing the instructions itself. The reason for this is that these instructions are also placed in FRAM and must be read from the CPU in order to be executed. When comparing Figure 9 with the algorithm from Listing 1 it turns out that memory row zero is not influenced by the fault. The reason for this is on the one hand, that the EMFI setup, including the rise time of the trigger, has a delay that can prevent hitting the instructions placed in rows directly after the trigger routine. On the other hand, it seems that the microcontroller preloads complete memory rows in cache before executing its content, instead of issuing a read for every single instruction.

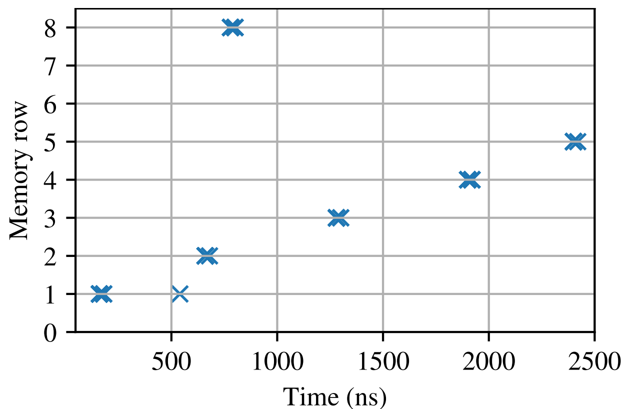

Figure 9: Impact of fault timing on memory locations in the analyzed firmware (see Listing 1).

This becomes clearer when examining the chronological order in which the memory rows are affected by increasing glitch timings. The targeted 8th row encounters upsets later than the 2nd memory row. However, the `mov` instruction accessing the 8th memory row is stored in memory row number one. In theory, due to the order of instruction storage, the 8th row should be affected by the glitch before the 2nd row. Nevertheless, since the microcontroller appears to load the rows in advance, the timing aligns with Figure 9. The preloading is also the reason why memory row four and five, which are not part of the algorithm in the trigger window, are affected, despite ensuring that the fault is only applied in said window. According to Figure 9, the preloading occurs roughly every 500 ns. This matches with the fastest time that a memory row can be processed by the CPU running at 8 MHz, if it only contains instructions that endure one clock cycle.

In order to provide comparability with the investigation on the dedicated FRAMs, Figure 10 shows the bit-flip probabilities for the microcontroller in the same way as for the external memories. However, compared to the external memories, only 64 consecutive memory rows are analyzed. Due to the integrated error-correction single bit-flips have no effect as they get corrected. It is visible that the memory is more likely to switch from zeros to ones than vice versa, thus preferring its erased state which is all ones. In heatmap (b) the higher order bits in the memory row starting with bit 48 exhibit a significantly increased flip chance. It is not clear if this is caused by the physical layout of the memory, or if the actual flip locations are masked by the ECC. Nevertheless, the flip probabilities appear to be linked to the bit positions inside of the memory rows, indicated by the horizontal lines in the heatmaps. This implies that the root cause for the corruption relates to a plate-line timing violation as described in Section 2.2.

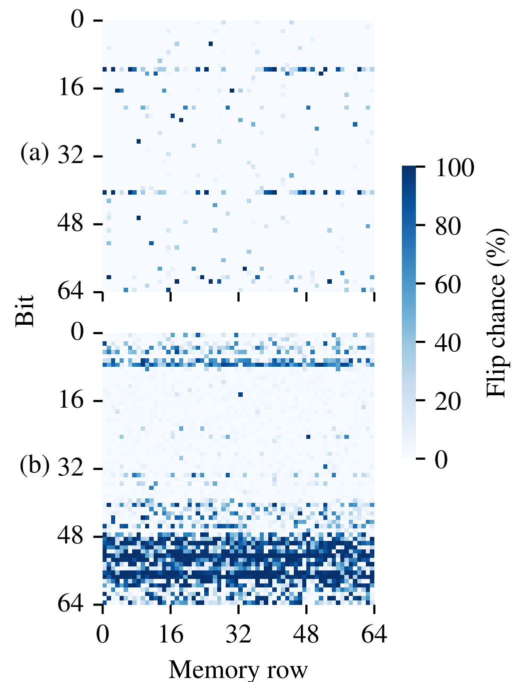

Figure 10: Bit-flip chances for a section of the FRAM inside the MSP430FR5962, targeting a read operation on each memory row. Upper heatmap (a) shows direction $1 \rightarrow 0$, lower heatmap (b) shows direction $0 \rightarrow 1$.

It is to note that the SRAM of the device can also be corrupted with the glitch. Early versions of the target firmware used plain C code and a controllable address to perform the read operation in a loop. This stores the address in a variable that is located in SRAM by default. A dump from the SRAM

indicates that glitches can alter its contents, including the variable storing the targeted address, leading to reads from random locations. Using an assembly instruction with a fixed address or a careful selection of the glitch timing fixes that issue. Further analysis on the effects were not performed as they are not in scope for this work.

5.4 Unlocking JTAG

In order for the device to know if its debug interface is locked, it must read its JTAG signatures. As these signatures reside in FRAM, the read operation is potentially vulnerable to the write-back attack, implying that the relevant memory can be altered. Due to the nature of the fault as described in Section 2.2 and from the previously analyzed external memories, it is not feasible to control the memory value resulting from the attack. However, as the JTAG interface is only locked for two specific values this is not an obstacle. As long as the signature changes to any other value, the device gets unlocked. With respect to the error-correction mechanism at least two bits must be flipped.

The user manual [9] of the microcontroller specifies that locking JTAG only takes effect after a brownout reset (BOR) occurs. Without a reset, debugging is still possible even if the locking signatures are written to memory. That means that the signature, or at least its memory location, is not checked for every JTAG command, but at the startup of the device. Furthermore, the user manual specifies that the JTAG password is copied from memory to a register after a BOR.

With a maximum startup time of 1 ms for the microcontroller after a BOR and a glitch window of 30 ns, more than 30k timestamps must be considered for the fault attack. As this is quite time intensive, a side-channel analysis (SCA) is performed on the startup phase of the device to reduce the parameter space. The time span could also be narrowed down by measuring the time required for the BOR, as it is only one step of the complete startup procedure. However, this only reduces the required effort roughly by a factor of three, so the SCA is still the preferred choice.

Side-Channel Analysis For the SCA the reset pin is used simultaneously as a trigger for an oscilloscope and to trigger the BOR, as it allows for a more precise timing compared to switching the power supply on and off. Due to the capacitive load the rising edge of the power line is smoothed out and not suitable for accurate timed triggers. The custom PCB to that the microcontroller is soldered has an option to place a shunt resistor in the supply voltage line. As the device draws about 0.4 mA current during the boot phase, a shunt resistor with 27 Ω is selected. This results in a voltage drop of about 0.1 V across the resistor. The board is powered with 3.3 V ensuring that the microcontroller is still operating in its recommended range.

With the help of the debugger, different values are written to the JTAG signatures. Figure 11 shows the measured voltage drop across the shunt resistor during the startup phase for the different values. Note that the JTAG module is in unlocked state for both cases. The value 0xBBCC is selected as it is as close as possible to the actual value of 0xAAAA, without causing a lock, to ease the measurements. Contrary to the datasheet [20], two arbitrary located 0xA in the 16-bit register are sufficient to lock the device. Furthermore, patterns such as 0xBBBB also cause the debug interface to lock. It seems that this is a bug in the startup routine of the microcontroller or an error in the documentation.

By having a look at Figure 11, it is visible that the release of the reset signal initially causes a spike. The signal triggers the device startup which causes the short pulse from inrush currents. By comparing the two traces recorded with different JTAG signatures, it is evident that both are aligned up to the 55 µs mark. It is to note that if the JTAG signature is set to all zeros, the complete startup phase is almost 64 µs shorter compared to when it is set to the non-zero value. That indicates that the algorithm which checks the JTAG signature during the startup phase may is implemented with an early exit condition. Finally, this provides timing information that allows us to significantly narrow down the time slot during which the actual readout of the JTAG signature occurs.

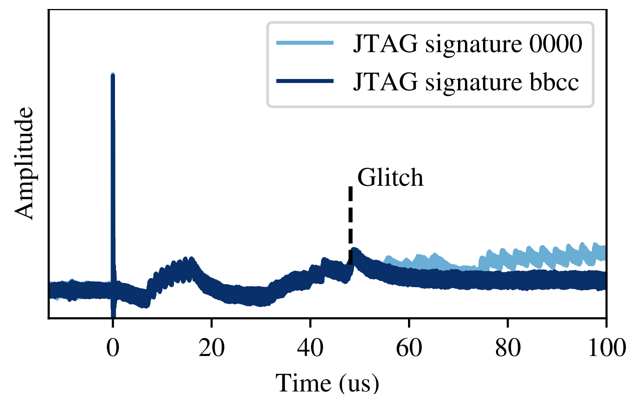

Figure 11: Power traces of the startup sequence from the MSP430FR5962 with differently configured JTAG signatures.

Unlock With the help of the SCA and the previous analysis, the EMFI parameter space can be reduced to a single location for the coil and a narrow time frame. The search reveals a 30 ns time slot in the 48 µs range after the reset signal rises, during which the contents of the JTAG signatures can be reliably modified. The timing is marked in Figure 11 and aligns well with the discoveries from the SCA. In most cases multiple glitch attempts are necessary to unlock the device, because on the one hand the integrated error-correction prevents single bit-flips to be effective and on the other hand not all values unlock the debug access. It is noteworthy that it

does not matter if the JTAG interface is password protected or completely disabled. Both cases can be unlocked with the identical procedure.

Implications The attack, exploiting the write-back mechanism of FRAM, enables full JTAG access, including all debugging features. This can be used to extract firmware or run arbitrary code on the targeted device. Once the fault location and the timing are identified, the device can be consistently unlocked in less than a minute. This allows attackers to act quickly, eliminating the need to steal potential target devices. In theory the unlock should be applicable to all devices from the MSP430FR5 and MSP430FR6 series, as they all implement the same JTAG locking mechanism. Most devices even share the same user guide, indicating strong similarity.

5.5 Countermeasures

From the side of the manufacturer the simplest fix would be to inverse the JTAG signature logic, so that the device is only unlocked if the signature matches 0x5555. All other values should lock the debug interface. Due to the random nature of the fault, it is highly unlikely that it will then unlock the microcontroller. However, as this check is implemented in hardware, this improvement would only affect future devices. In-field updates are not possible, leaving those devices vulnerable. It is unknown to the authors why this feature is implemented this way. Especially because the BSL or the flash-based microcontrollers of the same family implement the locking mechanism inverse, such that only two values unlock the interface and all other lock it. One explanation would be that devices could get locked unintentionally during soldering, as FRAM is sensitive to high temperatures [28]. Other than that, the access control could be implemented through a fuse or, if the password feature is desired, by storing the signature in dedicated memory with non-destructive read. Since our analysis showed that not all external devices are susceptible to fault injection, hardening the FRAM and its surrounding logic may also be an option.

For users of the device it may not be possible to protect it against readouts of sensitive data. Still, it is possible to detect manipulated devices by using the password feature. If a device is unlocked through glitching the FRAM, the complete memory row where the JTAG signatures are stored, is affected. In the case of locking the debug interface with a password, only two out of eight bytes from the memory row are required. The user can store a canary value in the remaining six bytes that will be destroyed during the fault.

6 Conclusion

In this work, we demonstrate that the *write-back* mechanism of FRAM reads is susceptible to fault injection, potentially leading to memory corruption. We subjected multiple external memories from different manufacturers to voltage glitches, revealing that three out of four devices are vulnerable. We attribute these faults to timing violations in the memory's peripheral circuit. Furthermore, we illustrate that this vulnerability can be exploited in real-world scenarios by unlocking the JTAG interface of an MSP430FR microcontroller in under a minute and with a modest budget of a few thousand USD. Theoretically, this unlock should be feasible for all devices in the MSP430FR5 and MSP430FR6 series. As this represents a pure hardware vulnerability, it must be fixed in the hardware design by the manufacturer. This work extends the current literature on fault effects in FRAM devices by presenting the implications of fault injection from a security perspective. Ultimately, our findings suggest that using FRAM can pose a security risk, which must be taken into account during the design phase of embedded systems.

Future Work One of the analyzed memory devices was not affected by the attacks. As the reasons for its resilience compared to its competitors remain unclear, future research could explore countermeasures based on this device. Additionally, it would be interesting to investigate whether the write-back vulnerability can be leveraged for more complex attacks, despite its inherent randomness. For example, the tendencies for faulted memory cells to prefer either the reset or set state could be used to reduce entropy in symmetric crypto keys stored in FRAM. Finally, the experiments regarding EMFI were conducted using a 4 mm coil with a CCW direction. The potential benefits of using coils with a smaller diameter or a clockwise winding direction is left for future work.

Coordinated Disclosure

As part of a coordinated disclosure process, we informed the product security incident response team (PSIRT) of TI about our finding. Technical and detailed information were provided more than 120 days prior to the publication of this paper.

Acknowledgment

This work was funded by the Bavarian Ministry of Economic Affairs, Regional Development and Energy, through the project "Trusted Electronics Center Bavaria" (TrEB).

References

[1] CY15B016Q Datasheet. https://www.infineon.com/dgdl/Infineon-CY15B016Q_16_KBIT_(2K_X_8)_SERIAL_(SPI)_AUTOMOTIVE-E_F-RAM-DataSheet-v04_00-EN.pdf?fileId=8ac78c8c7d0d8da4017d0ee3f9176a99.

[2] flashrom README — flashrom documentation. https://flashrom.org/.

[3] FM25L16B Datasheet. https://www.infineon.com/dgdl/Infineon-FM25L16B_16-Kbit_(2_K_8)_Serial_(SPI)_F-RAM-DataSheet-v11_00-EN.pdf?fileId=8ac78c8c7d0d8da4017d0ec917394180.

[4] MB85RS16 Datasheet. https://www.fujitsu.com/uk/Images/MB85RS16.pdf.

[5] MR45V032A Datasheet. https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/539/MR45V032A.pdf.

[6] MSP430FR5962 data sheet, product information and support | TI.com. https://www.ti.com/product/MSP430FR5962.

[7] MSPDebug. https://dlbeer.co.nz/mspdebug/.

[8] FRAM SPI Reads & Writes and Data Protection During Power Cycles, May 2005. https://www.cika.com/soporte/Information/Ramtron/FRAM/AppNotes/SPI_Power_Cycling.pdf.

[9] MSP430FR58xx, MSP430FR59xx, and MSP430FR6xx Family User's Guide - SLAU367P, April 2020. https://www.ti.com/lit/ug/slau367p/slau367p.pdf.

[10] MSP430 Programming With the JTAG Interface - SLAU320AJ, May 2021. https://www.ti.com/lit/ug/slau320aj/slau320aj.pdf.

[11] FUJITSU SEMICONDUCTOR AMERICA. FRAM-Standalone-FS.pdf, 2011. https://www.fujitsu.com/us/Images/FRAM-Standalone-FS.pdf.

[12] A. L. Bosser, V. Gupta, A. Javanainen, G. Tsiligiannis, S. D. LaLumondiere, D. Brewe, V. Ferlet-Cavrois, H. Puchner, H. Kettunen, T. Gil, F. Wrobel, F. Saigné, A. Virtanen, and L. Dilillo. Single-Event Effects in the Peripheral Circuitry of a Commercial Ferroelectric Random Access Memory. *IEEE Transactions on Nuclear Science*, 65(8):1708–1714, August 2018. https://ieeexplore.ieee.org/document/8268559/?arnumber=8268559.

[13] Claudio Bozzato, Riccardo Focardi, and Francesco Palmarini. Shaping the Glitch: Optimizing Voltage Fault Injection Attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 199–224, February 2019. https://tches.iacr.org/index.php/TCHES/article/view/7390.

[14] Cypress Semiconductor Corp. F-RAM Technology Brief, April 2015. https://www.mouser.tw/pdfDOCS/cypress-fram-whitepaper.pdf.

[15] Travis Goodspeed. Practical Attacks against the MSP430 BSL. Berlin, December 2008. https://fahrplan.events.ccc.de/congress/2008/Fahrplan/attachments/1191_goodspeed_25c3_bslc.pdf.

[16] Bhanprakash Goswami and Manan Suri. Experimental Investigation of EM Side Channel and FI Attacks on Commercial FRAM Chips. In *2024 8th IEEE Electron Devices Technology & Manufacturing Conference (EDTM)*, pages 1–3, March 2024. https://ieeexplore.ieee.org/document/10511546.

[17] S. Sheffield Eaton Jr. Self restoring ferroelectric memory, October 1989. https://patents.google.com/patent/US4873664A/en.

[18] Anan Ju, Hongxia Guo, Fengqi Zhang, Lili Ding, Guanghua Du, Jinglong Guo, Xiangli Zhong, Jianan Wei, Xiaoyu Pan, and Hong Zhang. Failure Analysis of Commercial Ferroelectric Random Access Memory for Single Event Effect. *IEEE Transactions on Nuclear Science*, 69(4):890–899, April 2022. https://ieeexplore.ieee.org/document/9718289/?arnumber=9718289.

[19] E.M. Philofsky. FRAM-the ultimate memory. In *Proceedings of Nonvolatile Memory Technology Conference*, pages 99–104, Albuquerque, NM, USA, 1996. IEEE. http://ieeexplore.ieee.org/document/534679/.

[20] Katie Pier. MSP Code Protection Features - SLAA685, December 2015. https://www.ti.com/lit/an/slaa685/slaa685.pdf.

[21] Pinin. Perovskite structure of PZT, August 2010. https://en.wikipedia.org/wiki/Lead_zirconate_titanate#/media/File:Perovskite.svg.

[22] Volker Rzehak. Low-Power FRAM Microcontrollers and Their Applications - SLAA502, July 2019. https://www.ti.com/lit/pdf/slaa502.

[23] Claudio Sansoe and Maurizio Tranchero. Use of FRAM Memories in Spacecrafts. In Mickal Lallart, editor, *Ferroelectrics - Applications*. InTech, August 2011. http://www.intechopen.com/books/ferroelectrics-applications/use-of-fram-memories-in-spacecrafts.

[24] Marc Schink, Alexander Wagner, Felix Oberhansl, Stefan Köckeis, Emanuele Strieder, Sven Freud, and Dominik Klein. Unlock the door to my secrets, but don't forget to glitch: A comprehensive analysis of flash erase suppression attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*,

2024(2):88–129, Mar. 2024. https://tches.iacr.org/index.php/TCHES/article/view/11422.

[25] A. Sheikholeslami and P.G. Gulak. A survey of circuit innovations in ferroelectric random-access memories. *Proceedings of the IEEE*, 88(5):667–689, May 2000. http://ieeexplore.ieee.org/document/849164/.

[26] Jingyu Shen, Wei Li, and Yuanbin Zhang. Assessment of TID Effect of FRAM Memory Cell Under Electron, X-Ray, and Co- 60 \gamma Ray Radiation Sources. *IEEE Transactions on Nuclear Science*, 64(3):969–975, March 2017. https://ieeexplore.ieee.org/document/7822991/?arnumber=7822991.

[27] David Still, Jesse Siman, Madhu Joseph, Justin Beavers, Shivendra Singh, and Pramodh Prakash. Cypress Semiconductor Corporation 64-Kbit (8 K × 8), 16-Kbit (2 K x 8), and 4-Kbit (512 x 8) Serial (SPI/I2C) F-RAM Characterization Report, September 2020. https://www.mouser.com/PCN/Cypress_Semiconductor_PCN203901_1.pdf.

[28] Priya Thanigai and William Goh. MSP430 FRAM Quality and Reliability - SLAA526A, May 2014. https://www.ti.com/lit/an/slaa526a/slaa526a.pdf.

[29] Becky Thomas, Don Darling, and Rene Rodgers. Cypress Semiconductor Product Qualification Report QTP#130104 VERSION*E, June 2015. https://www.infineon.com/dgdl/Infineon-QTP_130104_AEC-Q100_Grade_3_Qual_Report_B-W_Serial_F-RAM_Family_4Kb_to_256Kb_Product_Qualification_130nm_Technology_TI_Fab-ProductQualificationReport-v06_00-EN.pdf?fileId=8ac78c8c7d710014017d714cb9af133a.

[30] Silvia Vitiello, Nikoleta Andreadou, Mircea Ardelean, and Gianluca Fulli. Smart Metering Roll-Out in Europe: Where Do We Stand? Cost Benefit Analyses in the Clean Energy Package and Research Trends in the Green Deal. *Energies*, 15(7):2340, March 2022. https://www.mdpi.com/1996-1073/15/7/2340.

[31] Nick Zahabizadeh. Authentication ICs—BQF0008 BQF0020 BQF0064, April 2022. https://www.ti.com/lit/po/slat160/slat160.pdf.

[32] Zhangang Zhang, Zhifeng Lei, Zhenlei Yang, Xiaohui Wang, Bin Wang, Jie Liu, Yunfei En, Hui Chen, and Bin Li. Single Event Effects in COTS Ferroelectric RAM Technologies. In *2015 IEEE Radiation Effects Data Workshop (REDW)*, pages 1–5, July 2015. https://ieeexplore.ieee.org/document/7336734/?arnumber=7336734&tag=1.

[33] Michael Zwerg, Adolf Baumann, Ruediger Kuhn, Matthias Arnold, Ronald Nerlich, Marcus Herzog, Ralph Ledwa, Christian Sichert, Volker Rzehak, Priya Thanigai, and Bjoern Eversmann. An 82uA/MHz microcontroller with embedded FeRAM for energy-harvesting applications. pages 334–336, March 2011. https://www.researchgate.net/publication/224228683_An_82mAMHz_microcontroller_with_embedded_FeRAM_for_energy-harvesting_applications.

A Appendix

Table 3: Bit-flip probabilities for a random selected bit on all tested FRAM devices.

Sample	Flip Chance $1 \rightarrow 0$	Flip Chance $0 \rightarrow 1$
FM25L16B		
1	42.4	11.9
2	58.5	1.3
3	39.1	8.6
4	54.7	6.8
5	53.5	7.7
$\overline{X}$	49.6	7.3
CY15B016Q0		
1	74.4	1.1
2	63.4	2.5
3	74.7	1.7
4	76.5	1.6
5	67.4	7.0
$\overline{X}$	71.3	2.8
MR45V0320		
1	8.7	28.5
2	12.4	16.7
3	7.9	27.6
4	9.3	21.9
5	8.4	19.9
$\overline{X}$	9.4	22.9

A.1 FM25L16B

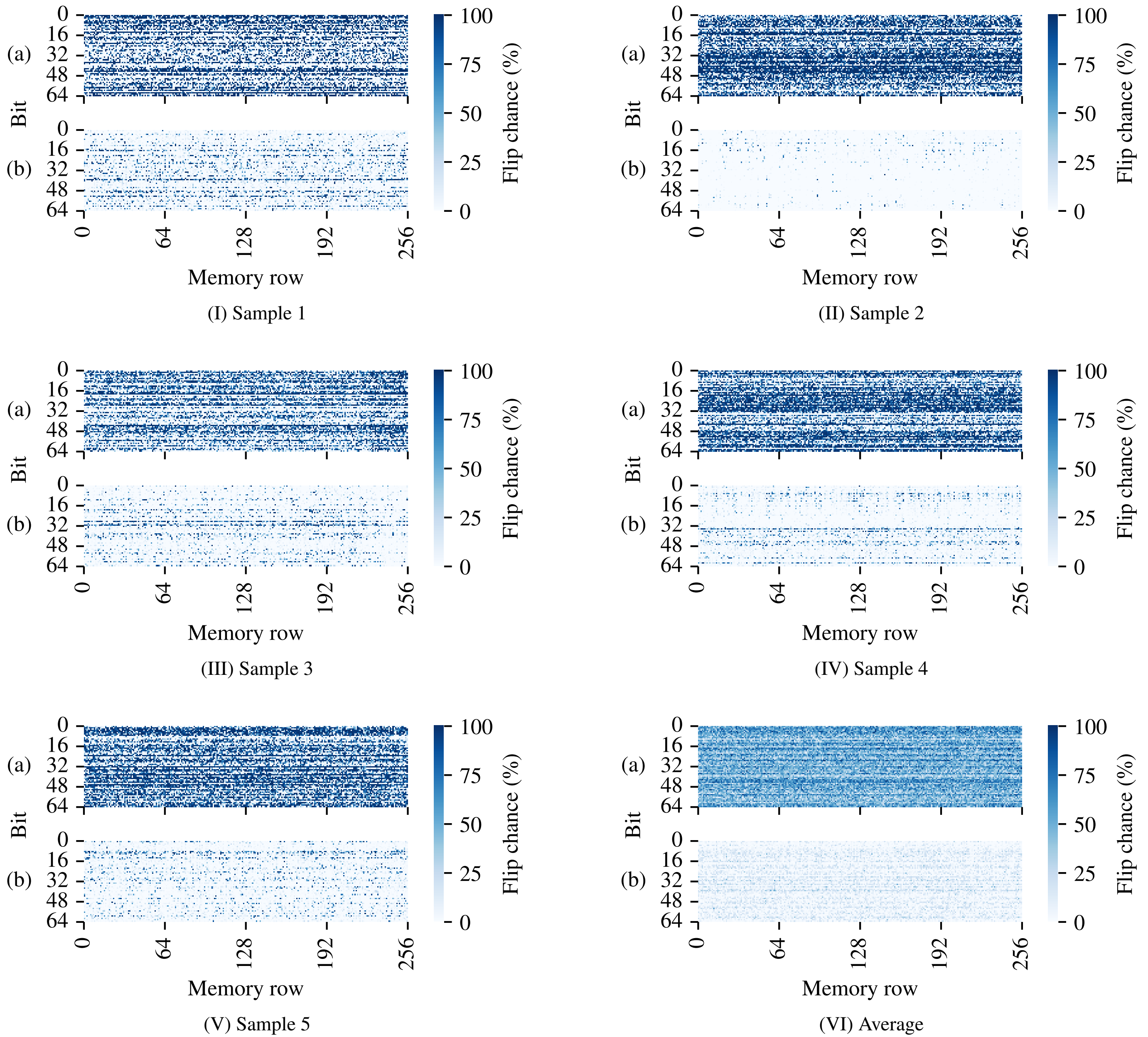

Figure 12: Heatmaps of all FM25L16B samples.

A.2 CY15B016Q

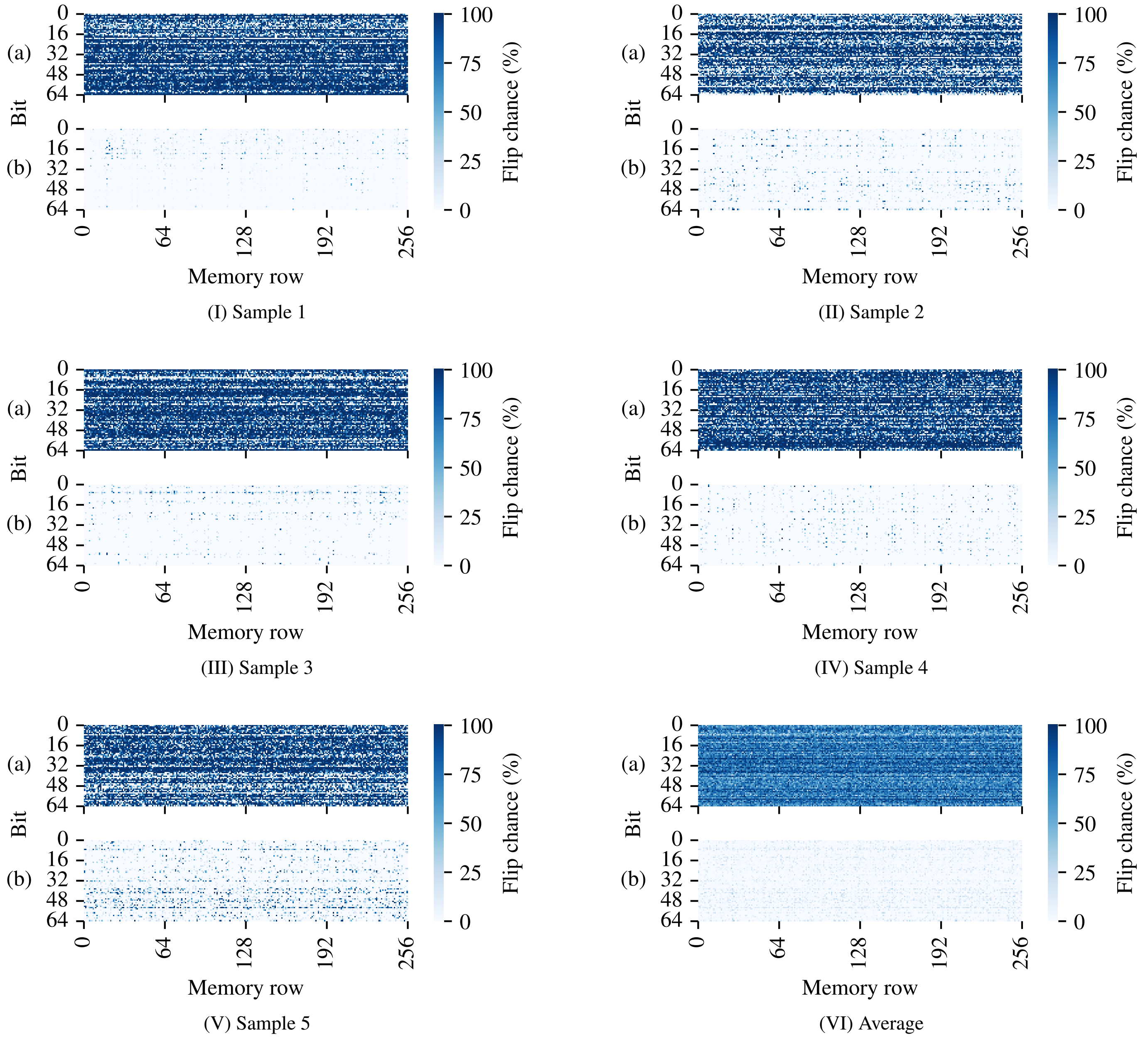

Figure 13: Heatmaps of all CY15B016Q samples.

A.3 MR45V032

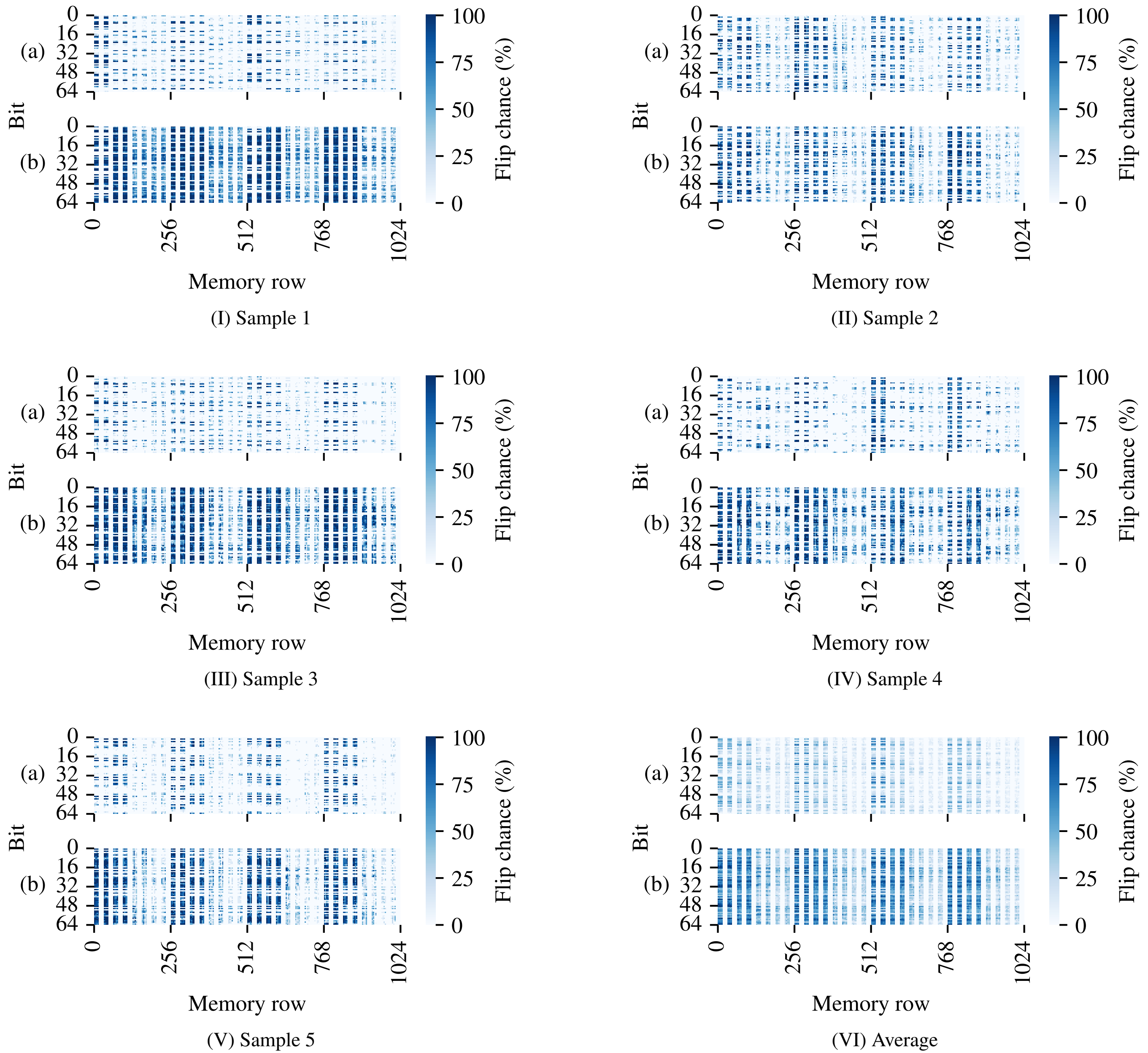

Figure 14: Heatmaps of all MR45V032 samples.