2024 NDIA MICHIGAN CHAPTER
GROUND VEHICLE SYSTEMS ENGINEERING
and TECHNOLOGY SYMPOSIUM
Modular Open Systems Approach (MOSA) Technical Session
August 13-15, 2024 - Novi, Michigan

CHALLENGES AND MITIGATIONS FOR DATA REMANANCE IN FPGA BASED SYSTEMS

Kevin Paar, Scott Harper, PhD

Graf Research Corporation, Blacksburg, VA

ABSTRACT

FPGA based electronic systems have the potential to support zeroization and sanitization of sensitive information without resorting to kinetic destruction. However, careful consideration needs to be given to the data remanence effects of both the FPGA and the attached storage media that form a complete system. SRAM, DRAM, and different forms of flash memory all have distinct data remanence characteristics that must be accounted for in the design of zeroization solutions. In this paper, we survey these characteristics across typical FPGA system media types and present a framework to enable the rapid integration of complete zeroization and sanitization capabilities in FPGA systems.

Citation: *K. Paar, S. Harper, PhD, "Challenges and Mitigations for Data Remanence in FPGA Based Systems," In Proceedings of the Ground Vehicle Systems Engineering and Technology Symposium (GVSETS), NDIA, Novi, MI, Aug. 13-15, 2024.*

1. INTRODUCTION

Deployed electronic systems invariably contain sensitive information in the form of intellectual property (IP), configuration information, and transient operational data. If those electronic systems fall outside the control of their authorized users, they can potentially be exploited in unintended ways. Preventing this exploitation in DoD ground systems has typically involved the use of kinetic means to physically destroy the equipment. However, this method can be dangerous, may draw attention to the destruction, and may fail to sufficiently destroy the system enough to prevent data recovery and exploitation. Systems based on field programmable gate arrays (FPGAs) offer a better approach in which their sensitive information can be securely cleared from the system by zeroizing (overwriting with zeros) or sanitizing (repeated overwrites) the storage media that holds that information.

FPGAs are blank slates for logic without their associated configuration data. That configuration data contains the intellectual property that forms a useful deployed system

from what is otherwise a general-purpose commercial off-the-shelf (COTS) hardware platform. This also implies that the configuration data can be a target for reverse engineering to extract the functional details of the FPGA-based system. All FPGAs must utilize non-volatile storage to maintain their configuration information across power cycles, either internally (Flash-based FPGAs, e.g., Microchip Polarfire) or externally (SRAM-based FPGAs, e.g., Xilinx Versal, Intel Stratix, etc.) Either class of FPGA can utilize cryptographic encryption and authentication mechanisms to protect their configuration information when it is external to the FPGA. However, once the configuration is distributed throughout the FPGA fabric, it will no longer be cryptographically protected and can be a target for attacks.

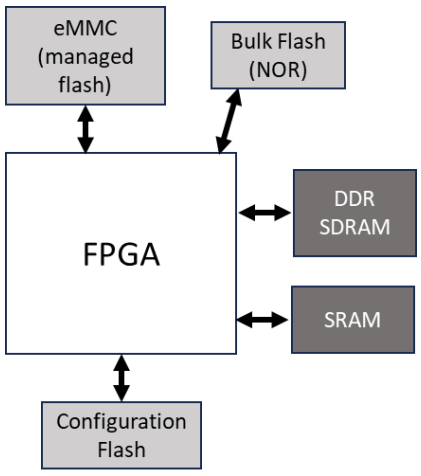

Figure 1: FPGA system with typical external media.

In addition to the FPGA configuration data, a complete FPGA-based electronic system may also contain sensitive information in the form of other IP (software, AI models, etc.), application-level provisioning information including application Critical Security Parameters (CSP) such as cryptographic communication keys, spread-spectrum coding sequences, etc., as well as transient operational data. For example, FPGA designs with CPU cores, either hard cores or soft cores, will contain boot and operational software code bases. That code may be embedded in the configuration data or may be stored in other external storage media. Those CPU cores typically also have associated external DRAM to use as transient storage. This invariably leads to system designs like that shown in Figure 1, with various external storage media (SRAM, DRAM, Flash, etc.) connected to the FPGA.

Careful consideration needs to be given to the requirements for sanitizing any sensitive data from these external components. Volatile media (e.g., SRAM, DRAM) should theoretically not require any zeroization consideration beyond simply removing their power. However, the complexities of system power architectures (redundant sources, slow decay times, low power stand-by states, etc.) may limit designer confidence that those media are fully zeroized without active means. Therefore, provisions should be available for active zeroization to ensure no residual information is left intact. Non-volatile media (e.g., FRAM, Flash) have a more obvious need for sanitization, however the methods of achieving that are not obvious. Some flash memories utilize oversized arrays with internal block remap techniques to optimize write performance, eliminate defects, and evenly distribute write accesses across the flash array. Thorough sanitization of these parts invariably requires more than a simple erase operation.

In this paper we explore the challenges of data remanence in FPGAs and the different data storage media typically attached to them. We then propose a configurable architecture to meet these challenges and enable the rapid integration of complete zeroization and sanitization capabilities in FPGA systems.

2. THE CHALLENGE OF DATA REMANENCE

Below we discuss our findings regarding data remanence and imprinting in the different storage media we have explored, and how that applies to both the external media and the FPGAs found in typical FPGA systems. We separately explore SRAM, DRAM, bulk flash, and managed flash.

2.1. SRAM Data Remanence

SRAM circuits are generally constructed using two inverters interconnected such that each one drives the other, with one inverter driving a '1' and one inverter driving a '0', as shown in Figure 2. This creates a feedback loop that maintains the state of the circuit.

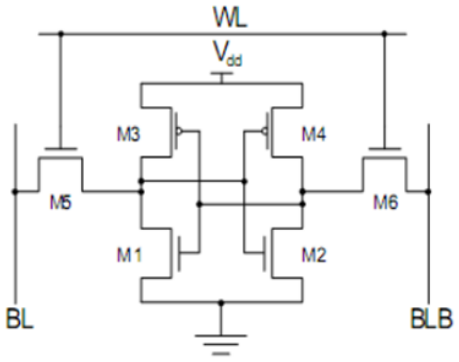

Figure 2: A typical SRAM cell [1]

Which inverter is driving the '1' can be swapped during a write, forcing the circuit to one state or the other, which is how it stores a data value. When power is first applied to an SRAM circuit (or cell), the state is undefined and goes through a period of metastability until it settles to one value or the other [2]. If the SRAM cells in an array are physically well-balanced, then the power-up state of that array will statistically come out to 50% '1' and 50% '0' values [3].

SRAM is volatile by design and can be zeroized by simply removing power. However, the data does not disappear instantaneously. Instead, it decays over time as charge accumulated in the circuit leaks out. The charge leakage is not uniform from one SRAM cell to another, so each cell in an array will decay at a different rate. For a given period of decay, some percentage of the cells will lose their charge and their stored state, with more time allowing a higher percentage of the cells to decay. If power can be restored to the SRAM before the charges completely decay, it is possible to recover some portion of the stored data.

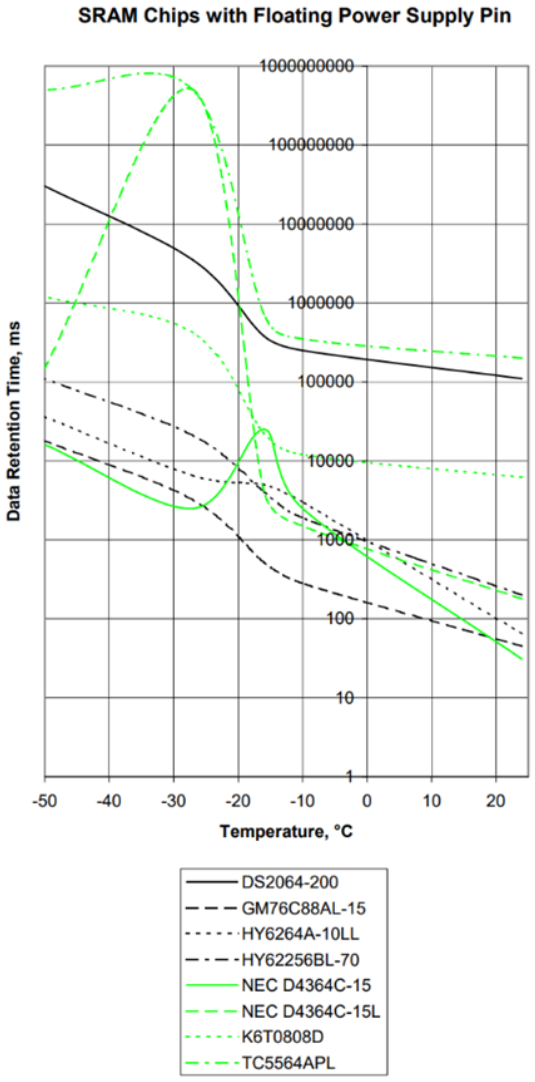

Figure 3: SRAM IC data retention times (floating power supply)[4]

Typically, this data decay is rapid enough that an attacker would have relatively little time to retrieve it. However, it has been

shown that low temperatures can significantly extend the retention time, allowing a higher percentage of the stored data to be recovered. In [5] Skorobogatov reported on experiments where complete SRAM ICs were set to either all 1's or all 0's and then subjected to temperatures ranging from room temperature to -50°C before power was removed. After a period of time the power was restored, and the data read out. Experiments were conducted on eight different SRAM ICs from six different vendors, to determine the point at which a mere 20% of the data was lost. The results of many experiments confirm that the data retention times of SRAM chips increase with lower temperatures. The results also indicated that the decay rates varied significantly from one vendor to another, ranging from 10's of milliseconds to greater than 10 seconds at room temperature. All devices exhibited greater retention with lower temperatures, ranging from more than 10 seconds to what appears to be hours, as shown in Figure 3. Additionally, it was noted that the retention time of the SRAM ICs was longer for parts that consumed less power.

The researchers also ran experiments where the SRAM input power was tied to ground instead of simply turning off the power supply and leaving it floating. This has the effect of accelerating the decay of the overall accumulated charge in the device, which also accelerates the decay of the stored data. Depending on the part, the decrease in retention times was between a factor of two and factor of five at room temperature. Details can be seen comparing the curves from Figure 3 with Figure 4 below.

These potential data remanence attacks can be mitigated by active erasure (zeroization) of the SRAM prior to power-down. However, this requires knowledge of the pending power loss, and adequate time to perform the zeroization.

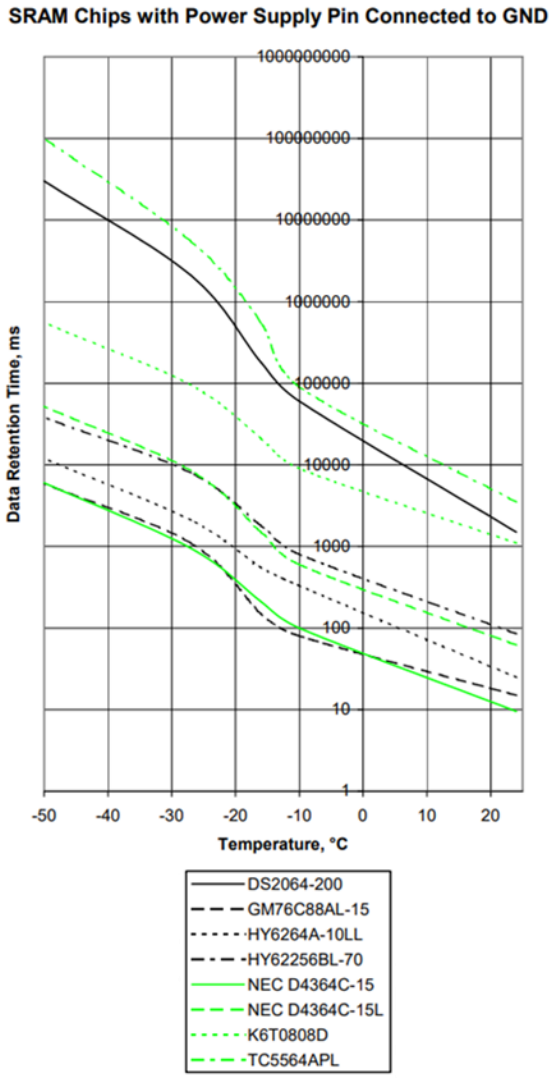

Figure 4: SRAM IC data retention times (grounded power supply)[4]

2.2. SRAM Data Imprinting

As mentioned above, a balanced SRAM cell will power up in an undefined state and then quickly settle to one value or the other with equal probability. If the cell is subject to any imbalance, it will make the cell more likely to power up in one state than the other. Those imbalances could come from the design of the cell itself, or they could be caused by operational wear in the cell. It has been found that SRAM cells can become biased simply by holding a static value over long periods of time[3][5]. Those biases can then potentially

be exploited to determine what static value had been held in the cell. This bias is sometimes referred to as imprinting [6].

The physical cause of the bias is difficult to determine. Several different mechanisms are mentioned in the literature, including electromigration and hot-carrier effects in [5], and negative bias thermal instability (NBTI) in [3]. Electromigration occurs when metal atoms in conductors relocate due to the flow of current, leading to microscopic whiskers and voids which increase the resistance of the conductor [5]. Hot-carrier effects are generally the result of strong electric fields across a transistor causing electrons to migrate into the transistor gate oxide, thereby changing the transistor characteristics [5]. Although possible, both these mechanisms apply more to transistors that are switching rather than those that are sitting at a static value. NBTI however, is caused by transistors sitting at the same value for long periods of time. Specifically, PFETs are susceptible to NBTI effects when they are left 'on' (conducting) for long periods of time. Over time the transistor threshold voltage will increase, and the transistor currents will decrease. These changes in transistor characteristics can lead to SRAM cell bias, effectively imprinting the stored value in the cell [3].

NBTI effects can be somewhat reversed by storing a complementary value in the cell. However, the longer the initial value was stored, the longer the complementary value will need to be stored to reverse the effects. Higher temperatures increase the effect in both directions. With enough time and temperature, NBTI effects can become permanent in SRAM media [7].

2.3. DRAM Data Remanence

Like SRAM, DRAM is volatile and loses its data when power is removed. However, the data decay of DRAM is slower than that of SRAM, owing to the capacitors that DRAM memory cells use to store their bit values, as shown in Figure 5. The data remanence of SDRAM, DDR, and DDR2 manufactured between 1999 and 2007 were studied by Halderman [8], who found that data decay in these systems can last from just a couple seconds to tens of seconds at room temperature, with newer devices exhibiting shorter retention times. However, the data retention times of DRAM could be vastly increased by cooling the memory cells to sub-freezing temperatures. Halderman [8] showed that DRAM cooled to -50°C with canned air developed almost no bit errors for at least 60 seconds after power off, and only about 1% bit errors after 10 minutes, allowing the researchers to recover the data. Their experiments went on to demonstrate that a DIMM cooled with a combination of canned air and liquid nitrogen lost only 0.17% of its data even after being removed from its machine for one hour.

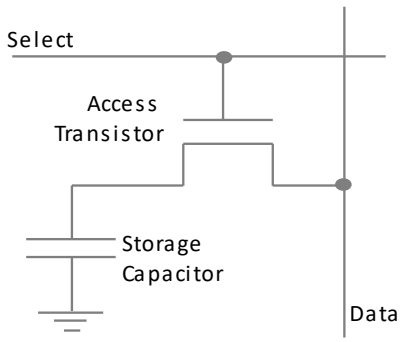

Figure 5: DRAM cell structure

In [9], Gruhn set out to replicate the results of Halderman [8], and additionally applied the same methodology to more modern DDR3 devices. The results of their DDR1 and DDR2 experiments largely aligned with those of Halderman—low temperatures can drastically increase retention times, allowing the researchers to transport the memory modules between compatible machines. However, they found that even cooling DDR1 and DDR2 systems by just 10°C could

still increase data retention, a result not reported by Halderman.

Interestingly, Gruhn found that low-temperature cold boot attacks were unsuccessful on DDR3 memory—data retention in DDR3 was unaffected by low temperatures. However, it appears Gruhn brought their memories to temperatures no lower than -10°C while Halderman brought theirs down to -50°C.

A separate study by Bauer [10] was successful at performing a cold boot attack on a DDR3 from Intel. They emphasized the fact that modern DRAM controllers may perform memory scrambling to even out the distribution of 1's and 0's across the memory, because biases can cause current spikes that lead to unreliability. Consequently, any attack must have a way to descramble the imaged RAM. They performed their cold boot attack by cooling a DDR3 system down to -30°C and note that such a low temperature was "absolutely essential" to their success. Even then, they found memory retention only lasted for about 10 seconds. Their success also heavily relied on the algorithms they developed to account for lossy acquisition, descrambling of the acquired memory image, and de-interleaving of the memory. All of this was achieved despite there being almost no documentation about the DRAM system they studied.

Because DDR3 cold boot attacks succeeded in one study while failing in another, it may be the case that these attacks depend on the DRAM system being attacked. As we suggested earlier, it could also be that DDR3 cold boot attacks require temperatures lower than what Gruhn attempted, which would explain why Gruhn was unsuccessful.

A 2017 experiment [11] demonstrated the success of yet another cold boot attack on scrambled Intel DDR4 DRAMs by cooling them to -25ºC. Despite the increasingly sophisticated methods Intel was using to scramble memory, the researchers cleaned up errors from lossy acquisition and correctly descrambled the image in just 2 hours.

Like SRAM, these attacks can be mitigated by active clearing (zeroization) of the DRAM instead of relying on natural decay. Also, like SRAM this requires notice of the power loss and sufficient time. Systems are typically designed with far greater DRAM capacity than SRAM capacity and the zeroization time for DRAM may far exceed that of SRAM. Systems which include power hold-up capacity to support zeroization need to take this into account.

2.4. Flash Memory Cells

Flash memory cells, as well as EEPROM cells, are based around a floating gate transistor. As illustrated in Figure 6, a floating gate transistor consists of a control gate and a floating gate, which are separated from each other and from the underlying substrate by insulating oxide layers. During programming, voltages are established on the control gate, source, and drain that cause electrons to tunnel into the floating gate where they are trapped by the insulating oxide layers. During erasure of the cell, different voltages are established on the control gate, source, and drain that cause electrons to tunnel back out of the floating gate. The presence of electrons in the floating gate increases the transistor's threshold voltage, which is the minimum voltage that needs to be applied to the control gate to turn on the channel between the source and drain [12].

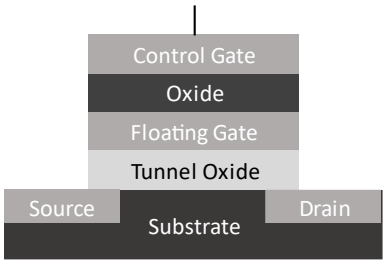

Figure 6: Floating gate transistor

In single-level cell (SLC) flash, the cell bit-value is read by applying a read voltage to the control gate and determining whether current flows through the transistor. In multi-level cell (MLC) flash, different quantities of electrons in the floating gate are used to establish multiple threshold voltage levels, each of which represent a different bit pattern. Multiple read voltages are applied to the control gate to determine not just the presence but the amount of current that flows through the transistor, and thereby determine what bit pattern was programmed into the cell. MLC flash typically utilizes four levels to store a two-bit pattern in each cell, although newer flash designs have been utilizing even more levels to store more bits per cell [12][13].

Program/Erase cycles cause wear on the flash cell over time, due to electrons becoming trapped in the floating gate and tunnel oxides that are not removed by erase operations. An increase in the trapped electrons leads to an increase in the cell's threshold voltage, eventually to the point that the cell can no longer be properly erased. When this occurs, the cell has failed and must be removed from use [49].

2.5. Flash Imprinting and Other Exploits

The shift in threshold voltage described above is particularly pronounced during the cell's first program/erase cycle, with the incremental shift caused by each subsequent program/erase cycle becoming smaller. The initial shift makes it possible to distinguish between cells that have been programmed and erased once, and cells that have never been programmed. This has the effect of imprinting the data into the device [5][12]. A proposed mitigation is pre-conditioning the flash cells by subjecting them to numerous program/erase cycles prior to using them [5]. Sanitization operations should include additional program/erase cycles to obscure any lingering data imprint.

Programmed flash memory is also at risk of being imaged directly with the IC powered down. Successful attempts have been made to read non-volatile memory charges using Scanning Electron Microscopy (SEM) on EEPROM ICs, which also utilize floating gate transistors [14]. By shaving away layers of the ICs until they reached the floating gates of the memory cells, researchers were able to image the cells with the Scanning Electron Microscope. They identified the values stored in the floating gates by examining whether the gates appeared light or dark in the SEM images. While this is not the first case of using microscopy to identify memory bit values, the researchers argue that their technique is significantly faster and easier than previous attempts. Based on these experiments, the possibility exists for these techniques to be used on flash chips that cannot be read out by non-invasive means.

2.6. Managed Flash

Larger flash components, generally referred to as solid state drives (SSD) such as eMMC, SD, and NVMe, are constructed with a flash controller that manages the data in the device. The flash controller exposes a block-based interface to the system and implements a Flash Translation Layer (FTL) that maps logical data blocks to physical flash memory blocks, as shown in Figure 7. To extend the overall life of the device, the FTL supports wear-leveling algorithms that distribute program/erase cycles across the device's physical blocks. Rather than rewriting updated data to the same physical block, the FTL simply writes the data to a new unused block and marks the old block as unused. This technique also increases write performance because blocks do not need to be erased before accepting new writes, and blocks that have permanently worn can be set aside with bad-block management. As a

result of these FTL algorithms, several latent copies or versions of the data may be present in the underlying flash array [12].

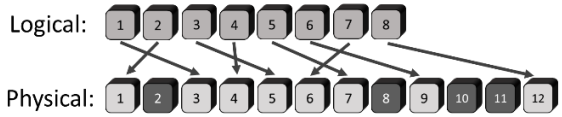

Figure 7: Logical to physical block mapping by flash translation layer (FTL)

Breeuwsma et al. [15] conducted research into forensic data recovery techniques that can uncover these old copies of data. In addition, when a system interfacing with the flash controller seeks to delete a particular set of data, the researchers showed that the flash controllers will often do little or nothing to purge the copies of that data that are sitting on previously written blocks. For example, Garg et al. [12] were able to recover files that had been supposedly deleted on flash USB storage media, which they owe to the media's FTL algorithms.

As described earlier, to reduce data imprinting one should pre-condition flash using many program/erase cycles before writing data. However, the FTL creates levels of indirection that make it difficult to achieve this in practice. As Gutmann [5] explains, attempting to create and then repeatedly overwrite a file will not repeatedly overwrite the same physical memory location, but rather results in "many copies of the data being written to different storage locations, followed by the actual data being written to yet another fresh storage location." In other words, the FTL's remapping algorithms opt to distribute these writes across many blocks and do not understand that the user wants to rewrite the same block.

Because the interfacing system does not have direct access to the physical blocks of SSD flash devices, ensuring complete and secure data erasure must be solved at the FTL controller level. Some SSD devices do claim to implement secure deletion commands. However, Wei et al. [16] studied numerous SSDs and found that some of them failed to adequately delete their data due to bugs in their implementations. In fact, some drives did not even attempt to erase the data, even though they reported that they had successfully done so.

2.7. Other Data Remanence Considerations

In Weingart's [6] list of "High Technology Attacks," he mentions a potential X-ray radiation attack against CMOS RAM that would cause the data to be burned into the memory cells. However, he does not specify SRAM or DRAM, does not elaborate on the mechanics of the attack, and does not provide any references. Smith and Weingart [17] describe a similar attack using ionizing radiation to cause data imprinting in SRAM—however, they again do not offer further explanation or references.

Weingart's [6] list also includes the threat of high-voltage attacks, which he also believes could cause imprinting in CMOS RAM. He does not present references or explanation, but it is possible that data could imprint due to rapid NBTI effects induced by the high voltages.

It should be mentioned that the recovery of imprinted data in RAM or flash is an inherently statistical process. None of the techniques encountered could retrieve a perfect copy of an array with one read pass. Typically, the extractions required numerous read attempts to build probable images of the data. Also, many devices contain error correcting code (ECC) mechanisms to help with read failures under normal operation. The existence of extra ECC data can aid a malicious data recovery attack by giving the attacker additional information to build confidence in their extracted values.

2.8. Data Remanence and Imprinting in SRAM-Based FPGAs

Tuan et al. [18] of Xilinx Research Labs present FPGA-specific findings that differ in some respects from the results of experiments on SRAM—the authors point out that Skorobogatov [4] tested on older chips, so his results may not apply to more modern FPGAs. Moreover, they argue that configuration memory in FPGAs ought to be a distinct focus, given that "FPGA configuration memory is very different from SRAM in terms of circuit design and layout." SRAM memory devices are organized into rows and columns of cells with specialized sense amplifiers at the bottom of the columns to accelerate read operations. FPGA configuration memory is not organized into arrays of cells but rather is comprised of logical latch structures dispersed throughout the FPGA. These latch structures directly drive the logic and interconnect they configure. Both the SRAM array cell and the FPGA latch are based on the interconnected two-inverter structure described earlier. However, the SRAM cell is typically fine-tuned to be balanced, dense, and fast, while the FPGA latch structure is designed to optimally support the logical requirements of its use.

FPGA configuration memory is typically reset to known values at power on. Without reset, it may be possible for the configuration cells to power up in illegal states that could cause physical harm to the device. Because of this, Tuan et al. [18] performed their experiments on a custom-built and unnamed 90nm Xilinx FPGA whose power-on reset was disabled. This allowed them to read back the configuration values after power cycling. Like Skorobogatov [5], they examined the data decay of bitstreams of all 0's or all 1's. However, it was found that these do not decay to 50% 1's and 50% 0's—rather, the configuration memory decay converges to 28% 1's, 72% 0's. This is a direct result of the non-balanced nature of the configuration cells. This imbalance may make the subtle effects of imprinting less likely.

It was also found that interconnect configuration cells exhibit longer data retention than logical configuration cells. They explain that this difference is a result of how logic and interconnect cells are powered: "In conventional FPGAs, logic memory is powered by the core voltage while the interconnect memory is regulated to a higher gate boosting voltage to increase performance."

In other ways, FPGAs had data remanence results like the SRAM results of Skorobogatov [4]. As with SRAM chips, data retention in FPGAs increases with lower temperatures when the device is powered off. To analyze the FPGA in terms of security, they borrow Skorobogatov's threshold of 20% data loss to measure decay times. At -40°C, 20% of the logic cells have lost their data after 1 second, while interconnect cells require 40 seconds to reach the same point. Additionally, they confirmed that FPGA memory cells discharge much more quickly when the FPGA power is grounded, as opposed to when it is left floating, like SRAM chips.

It is worthwhile to again note that these experiments required a specially modified FPGA to recover configuration data because normal FPGAs self-reset at power on. Power-on reset improves the security of the device and protects it from both low-temperature remanence attacks and any potential imprinting recovery attacks. However, if exploits are discovered that prevent power-on reset, the data is inherently vulnerable.

Any recovery of the FPGA configuration data requires the ability to read from the FPGA, typically using the JTAG port. All modern FPGAs can permanently disable the JTAG port to prevent such attacks. This is an obvious mitigation that should be used in critical designs.

2.9. Data Imprinting in Flash-Based FPGAs

As mentioned earlier, the configuration storage in an FPGA is generally dispersed throughout the IC and directly drives the circuits it is configuring. Flash-based FPGAs simply use floating-gate based cells instead of SRAM-based cells to directly configure the logic and interconnect. One implication of this is that the non-volatile configuration data cannot be encrypted. This leaves the flash-based FPGA potentially vulnerable to some of the discussed flash attacks. Therefore, it is imperative that these FPGAs be pre-conditioned before use, and properly zeroized in the field.

Like SRAM-based FPGAs, readback of the configuration data generally requires access to the JTAG port, which can be disabled.

3. CONCLUSIONS ON DATA REMANENCE

3.1. SRAM and DRAM

SRAM is susceptible to data imprinting, which biases the cells towards powering up at a particular bit value. While sanitization of imprinted data is possible, it may require significant amounts of time. To avoid imprinting, it has been recommended that SRAM data not be left at static values for long periods of time [5]. This may be possible in a system that can move the data from one memory location to another, or that can periodically invert the stored values.

DRAM does not appear to have any significant imprinting vulnerabilities and should not require additional sanitization. However, avoiding long-term static data should still be considered if possible.

Both SRAM and DRAM are susceptible to low-temperature data remanence attacks, which are more difficult to prevent. Data remanence attacks have been successfully performed on SRAM and multiple generations of DDR DRAM. Consequently, some high-security systems trigger zeroization of memory if low temperatures are detected [4]. This response helps prevent such attacks, but it may have the undesirable effect of limiting the operational temperature range of the system.

Based on the results by Skorobogatov [4] and Tuan et al. [15], it is possible to accelerate the decay, and therefore the zeroization, of SRAM by grounding the power supply rather than leaving the power supply floating. This is a very hardware-dependent design consideration, but it should be explored as a method to further mitigate remanence attacks. Alternatively, active zeroization can be used if power can be maintained for sufficient time to complete the process.

3.2. Bulk Flash

Flash cells experience wear from program/erase cycles that can effectively imprint data on the part and allow for malicious recovery. For bulk (unmanaged) flash, Gutmann [5] recommends pre-conditioning of the array with 10 to 100 program/erase cycles. This is to avoid imprinted data after simple zeroization operations. Sanitization operations should involve additional program/erase cycles to eliminate any lingering data, although this is a time-intensive operation.

3.3. FPGA Configuration Memory

SRAM-based FPGAs may also be susceptible to low-temperature remanence and to a lesser extent data imprinting but are largely protected by power on resets and access controls. SRAM-based FPGAs also benefit from power supply grounding if it is available.

Flash-based FPGAs may be susceptible to bulk flash attacks and should be pre-conditioned with program/erase cycles and

protected using access controls. Zeroization in the field is essential.

3.4. Media Encryption

The best solution to prevent issues with remanence or imprinting in any external media is to encrypt all sensitive data before writing it to the media. Recovery of the encrypted data is most likely of limited use without the encryption key.

Likewise, external FPGA configuration data should be encrypted. However, configuration data within the FPGA is not encrypted. As mentioned before, locking down any external access to the configuration data is crucial for avoiding exploitation.

4. A COMPLETE ZEROIZATION SOLUTION

The research described above enumerates the data remanence threats to complex FPGA systems with numerous external storage media, all with differing sanitization requirements. Furthermore, the exact requirements, and therefore solutions, will vary significantly from one platform to another, further burdening the system designer. Graf Research has defined a generalized, platform-agnostic solution called Enforte® Zeroize™ that supports zeroization and sanitization of FPGA systems, including their various attached media. This common zeroization framework is a scalable and modular approach involving retargetable, synthesizable IP blocks and software. The blocks are designed to be independently configured or exchanged as necessary to implement zeroization and sanitization functionality for any FPGA-based system.

The zeroization system is comprised of a zeroization controller, an FPGA fabric zeroization engine, and one or more external media zeroization engines, as shown in Figure 8.

The zeroization controller is responsible for zeroization triggering and for directing the operations of the zeroization engines within the same FPGA. The zeroization engines are each responsible for effecting the necessary media access operations to zeroize and/or sanitize the media to which they are attached.

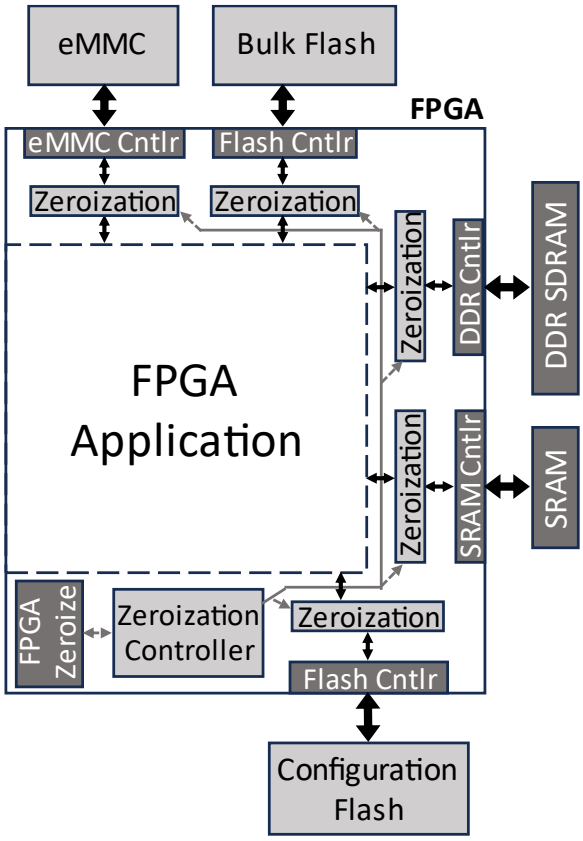

Figure 8: Zeroization solution for FPGA system

The configurable external media zeroization engine IP leverages the media interface IP already present in the system. This minimizes overhead and ensures proper control of the media during zeroization and sanitization operations without requiring numerous media-specific implementations.

The fabric zeroization engine utilizes the FPGA fabric control interface to zeroize the fabric. The fabric engine is designed to leverage the partial reconfigurability of FPGA fabrics to zeroize the application portion of the fabric while leaving the zeroization infrastructure intact to continue

with zeroization and sanitization operations of the attached external media.

When triggered, the zeroization controller orchestrates the operations of the zeroization engines to quickly zeroize the different media to remove retained data before starting more lengthy sanitization operations to eliminate remanent data caused by imprinting effects.

This hierarchical arrangement of generalized, customizable IP blocks within the FPGA allows the framework to accommodate any arrangement of FPGA and external media that may be encountered in a system. This also allows for concurrent execution of the zeroization and sanitization operations, thereby minimizing latency.

Graf Research is developing this zeroization framework, along with the tools and methods, to enable system designers to quickly and confidently integrate zeroization and sanitization capabilities into their application designs to support the ground systems need to eliminate exploitable information quickly and securely.

5. REFERENCES

[1] Neeraj Kr. Shukla, Shilpi Birla, R. K. Singh, and Manisha Pattanaik, "Speed and Leakage Power Trade-off in Various SRAM Circuits," *International Journal of Computer and Electrical Engineering*, pp. 244–249, Jan. 2011.

[2] D. E. Holcomb, W. P. Burleson, and K. Fu, "Power-Up SRAM State as an Identifying Fingerprint and Source of True Random Numbers," *IEEE Transactions on Computers*, vol. 58, no. 9, pp. 1198–1210, Sep. 2009.

[3] M. Hutter, J. Schmidt, "The temperature side channel and heating fault attacks," Proceedings CARDIS 2013, 12th International Conference, Berlin, Germany, November 27-29, 2013.

[4] S. Skorobogatov, "Low temperature data remanence in static RAM," University of Cambridge Computer Laboratory, Cambridge, UK, Tech. Rep. UCAM-CL-TR-536, June 2002.

[5] P. Gutmann., "Data remanence in semiconductor devices.", Presented at *10th USENIX Security Symp*, 2001.

[6] S. H. Weingart. Physical security devices for computer subsystems: A survey of attacks and defenses. Presented at *CHES 2000 2nd Int. Workshop*, Worcester, MA, pp. 302-317, 2000.

[7] D. K. Schroder, "Negative bias temperature instability (NBTI): Physics, materials, process, and circuit issues," *Distinguished Lecturer Series, IEEE Solid State Circuits Society*, 2005.

[8] J. A. Halderman et al., "Lest we remember: Cold boot attacks on encryption keys," Proceedings of the *17th USENIX Security Symposium*, 2008.

[9] M. Gruhn, T. Müller, "On the practicability of cold boot attacks," Presented at *2013 Int. Conf. on Availability, Reliability, and Security (ARES 2013)*, pp. 390-397.

[10] J. Bauer, M. Gruhn, F.C. Freiling, "Lest we forget: Cold-boot attacks on scrambled DDR3 memory," Presented at *DFRWS 2016 Europe*, vol. 16, pp. S65-S74, 2016.

[11] S. F. Yitbarek, M. T. Aga, R. Das, T. Austin, "Cold boot attacks are still hot: Security analysis of memory scramblers in modern processors," Proceedings of *IEEE Int. Symp. On High Performance Computer Architecture*, pp. 313-324, 2017.

[12] A. Garg, S. Chakraborty, M. Malik, D. Kumar, S. Singh, M. Suri, "Investigation of Data Deletion Vulnerabilities in NAND Flash Memory Based Storage," arXiv:2001.07424 [cs], 2020.

[13] Y. Cai, Y. Luo, E. Haratsch, K. Mai, O. Mutlu, "Data retention in MLC NAND flash memory: Characterization,

optimization, and recovery," Presented at *2015 IEEE 21st Int. Symp. On High Performance Computer Architecture (HPCA)*, 2015.

[14] F. Courbon, S. Skorobogatov, C. Woods, "Direct charge measurement in Floating Gate transistors of Flash EEPROM using Scanning Electron Microscopy," Presented at *Int. Symp. For Testing and Failure Analysis (ISTFA)*, pp.327-335, 2016.

[15] M. Breeuwsma, M. de Jongh, C. Klaver, R. van der Knijff, M. Roeloffs,"Forensic data recovery from flash memory," in *Small Scale Digital Device Forensics Journal*, vol. 1, no. 1, Jun. 2007.

[16] M. Wei, L. Grupp, F. Spada, S. Swanson, "Reliably erasing data from flash-based solid state drives," In Proceedings of the *9th USENIX conference on File and stroage technologies (FAST)*, *USENIX Association*, 2011.

[17] S. W. Smith, S. Weingart. "Building a high-performance, programmable secure coprocessor," in *Computer Networks*, vol. 31, no. 9, pp. 831-860, 1999.

[18] T. Tuan, T. Strader, S. Trimberger, "Analysis of data remanence in a 90 nm FPGA," Presented at *IEEE 2007 Custom Integrated Circuits Conf. (CICC)*, 2007.