Securing SoCs With FPGAs Against Rowhammer Attacks

Rana Elnaggar, *Member, IEEE*, Siyuan Chen, Peilin Song, *Senior Member, IEEE*, and Krishnendu Chakrabarty, *Fellow, IEEE*

***Abstract*—Heterogeneous SoCs integrate FPGAs and microprocessor cores on the same fabric to accelerate applications, such as cryptography and deep learning. Since FPGAs share resources with the microprocessor cores, they can launch noncacheable synchronous DRAM (SDRAM) transactions through direct FPGA-to-microprocessor SDRAM interface. Therefore, if the FPGA 3rd party IPs (3PIPs) are malicious, they can launch rowhammer attacks on the SDRAM. Today's countermeasures based on performance counters cannot detect these attacks because memory transactions from FPGAs do not pass through the cache. In addition, today's countermeasures that count the frequency of activation of memory rows cannot identify the intellectual property (IP) that launches the attack from the FPGA. We present a security solution that monitors the SDRAM transactions from IPs on the FPGA to each bank of the microprocessor SDRAM through the FPGA-to-microprocessor SDRAM interface. The proposed monitor is implemented on the FPGA fabric. It can detect attempts to launch a rowhammer attack before it causes bit flips in the SDRAM. It utilizes 6.3% of the adaptive logic modules (ALMs) available in an Intel Cyclone V FPGA, when multiple IPs are monitored.**

***Index Terms*—Computer security, memory, synchronous DRAM (SDRAM), system-on-chip.**

I. Introduction

With the increasing popularity of Internet-of-Things (IoT) and machine learning-based applications [1], [2], FPGA-SoCs are being developed to meet the growing need for powerful and energy-efficient computing platforms. These SoCs integrate general-purpose microprocessors, FPGA, and other intellectual property (IP) blocks [3]. Security is already an important consideration for such systems; however, current countermeasures are not sufficient to ensure that the untrusted IPs that are dynamically configured on the FPGA cannot maliciously impact trusted IPs.

Manuscript received July 15, 2020; revised December 17, 2020 and June 3, 2021; accepted July 18, 2021. Date of publication August 3, 2021; date of current version June 20, 2022. This work was supported in part by the National Science Foundation under Grant CNS-2011561; in part by the Semiconductor Research Corporation under Contract 2994; and in part by IBM Faculty Award. A preliminary version of this article was published as a two-page poster at the IEEE European Test Symposium, 2020. This article was recommended by Associate Editor A. Chatterjee. *(Corresponding author: Rana Elnaggar.)*

Rana Elnaggar, Siyuan Chen, and Krishnendu Chakrabarty are with the Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708 USA (e-mail: rana.elnaggar@duke.edu; krish@ee.duke.edu).

Peilin Song is with the Circuit Test and Diagnostics Technology, IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598 USA.

This article has supplementary downloadable material available at https://doi.org/10.1109/TCAD.2021.3102004, provided by the authors.

Digital Object Identifier 10.1109/TCAD.2021.3102004

In an FPGA-SoC, the IP blocks implemented on the FPGA fabric share resources with microprocessor cores. Intel refers to a microprocessor core as the hard processor system (HPS), while Xilinx calls it the processing system (PS). In many Intel and Xilinx FPGA-SoC products, the FPGA accesses the microprocessor synchronous DRAM (SDRAM) either through cache-coherent interconnects or directly through cache-incoherent interconnects [4]–[6]. The noncache-coherent interconnects between the FPGA and the SDRAM can allow IPs to directly access the physical address space of the SDRAM [4]. In addition, in SoCs with FPGAs, physical address to row address mapping is clearly documented in the technical manuals published by the manufacturers [7]. Therefore, these malicious IPs can launch rowhammer attacks by repetitively accessing specific SDRAM rows, which can result in bit flips in neighboring rows [8].

Prior work detects rowhammer attacks by monitoring the cache-miss rates recorded by the performance counters in a microprocessor [9], [10]. However, in FPGA-SoCs, the FPGA IPs can access microprocessor SDRAM directly without accessing the cache. Therefore, a rowhammer attack can be launched from these IPs without affecting the cache-miss rates. Counters can be integrated in the DRAM chip or in the memory controllers to measure the frequency of access to each memory row [11]–[14]. These countermeasures cannot identify the attacking IP. Thus, the attack will remain active, and the victim memory rows will be continuously refreshed.

FPGA-SoCs are currently integrated in critical applications, e.g., avionics applications [15] and in the automotive industry [2], while current countermeasures cannot secure these applications against rowhammer attacks. Therefore, we propose a security countermeasure against rowhammer attacks that can be integrated in the FPGA-SoCs that are available in the market. We present an FPGA-to-microprocessor SDRAM security monitor that observes the memory transactions between the FPGA IPs and SDRAM within a sliding window of ψ addresses. This monitor detects and blocks malicious IPs that request N_{th} accesses to a specific row within ψ consecutive memory accesses. The value of ψ is determined by the minimum required duration between two activations of the same row to avoid bit flips when the row is repetitively accessed, and the time taken by the DRAM to activate a row. The value of N_{th} is chosen to ensure that the number of row activations does not exceed the minimum threshold for causing bit flips, reported in [16]. As shown in Section VI, for current DRAM chips, $N_{th} = 2$ and $\psi = 10$ and 20 to detect single-sided and double-sided rowhammer attacks, respectively. The attack detection and IP blocking occur early enough, before the attack modifies the SDRAM contents. Our proposed approach also identifies stealthy IPs that collude to launch an attack that is not detected if each IP is monitored independently.

The main contributions of this article are listed as follows.

1) This is the first method that can detect rowhammer attacks launched from malicious FPGA IPs. These IPs can severely compromise the system because they have direct access to the physical address space of the SDRAM. The proposed method detects the rowhammer before its effect is propagated through the system. Therefore, refreshing of the victim rows is not required.
2) We propose and evaluate a memory-access policy for benign FPGA IPs through the FPGA-to-microprocessor SDRAM interface to avoid false positives.
3) We locate malicious FPGA IPs and identify stealthy IPs that collude to launch combined attacks on the system.
4) We evaluate the overhead of the proposed security monitor. It scrutinizes memory requests to a specific memory bank with maximum FPGA utilization of only 6.3%.

The remainder of this article is organized as follows. Section II reviews FPGA-to-microprocessor SDRAM interconnects in FPGA-SoCs and provides background on rowhammer attacks. Section III describes related prior work. Section V describes the target threat model. Section VI presents the proposed countermeasure and the benign memory-access policy for the FPGA-to-microprocessor SDRAM interface in FPGA-SoCs. Section VII describes the simulation results, hardware demo, and overhead for the FPGA-to-microprocessor SDRAM security monitor. Finally, we conclude this article in Section IX.

II. Preliminaries

A. FPGA-to-Microprocessor SDRAM Interconnects

In FPGA-SoCs, FPGAs and the SDRAM are connected using cacheable and noncacheable interconnects. We focus on the noncacheable interconnect, which we refer to as FPGA-to-microprocessor SDRAM interface. In Xilinx SoCs, it is called the programmable logic (PL)-to-PS interface. In Intel SoCs, it is called FPGA-to-HPS SDRAM interface. This interface is used when high-throughput and low-latency SDRAM accesses from the FPGA are required. It is configured as an advanced extensible interface (AXI) interface in Xilinx SoCs and as AXI or Avalon memory-mapped (MM) interface in Intel SoCs.

1) AXI Protocol: The AXI interfaces allow read and write operations to occur simultaneously; therefore, two command ports are used at a time, while Avalon-MM interfaces issue either a read or write operation at-a-time. The same proposed countermeasure applies to Avalon-MM interfaces. The basic AXI control, command, and data signals that are used in this article and their descriptions are listed in Table I. The AXI operations during read and write memory transactions are as follows [17].

When an IP reads data from memory, it performs the following: 1) assigns address read valid signal (ARVALID) to "1" and *ARADDR* to the desired address; 2) *ARVALID* remains set to "1" till SDRAM asserts *ARREADY* to "1"; 3) assigns *RREADY* to "1" when it is ready to receive data; and 4) SDRAM assigns *RVALID* to "1" to send *RDATA*. When an IP writes data to memory, it performs the following: 1) assigns *AWVALID* to "1" and *AWADDR* to the desired address; 2) *AWVALID* remains set to "1" till SDRAM assigns *AWREADY* to "1"; 3) assigns *WVALID* to "1" and *WDATA* to the target data; and 4) *WVALID* remains set to "1" till SDRAM assigns *WREADY* to "1."

TABLE I
Relevant AXI Bus Signals and Their Definitions

Signal Name	Signal Definition (Sender)
ARVALID (AWVALID)	Read (Write) address valid (IP)
ARADDR (AWADDR)	Target read (Write) address (IP)
ARREADY (AWREADY)	Ready for read (write) address (SDRAM)
RREADY	IP ready to read data (IP)
RVALID	Sending valid read data (SDRAM)
RDATA (WDATA)	Read data (SDRAM) (Write data (IP))
WREADY	Ready to receive write data (SDRAM)
WVALID	IP sending valid write data (IP)

TABLE II
Relevant Avalon-MM Bus Signals and Their Definitions

Signal Name	Signal Definition (Sender)
read	Indicate a read transfer (IP)
write	Indicate a write transfer (IP)
address	Target address shared by read and write (IP)
readdata	Data for read transfers (SDRAM)
writedata	Data for write transfers (IP)
waitrequest	Indicate that SDRAM cannot respond to commands (SDRAM)

2) Avalon-MM Protocol: From a security perspective, the biggest difference between the AXI and Avalon-MM interfaces is that Avalon-MM interfaces do not allow read and write operations to be issued simultaneously. The Avalon-MM signals that are relevant to the monitor design in this article are summarized in Table II. Read and write operations share a common *address* port. They are controlled by two signals, *read* and *write*, which must be asserted separately. As an example, the master component can set the correct read address and assert the *read* signal at cycle 1. If read operations have a fixed latency of 1 cycle, then, at cycle 2, the master component can obtain the data read from the *readdata* port. At cycle 3, the master component deasserts the *read* signal and asserts the *write* signal, while changing the address to the intended write address and setting the *writedata* port. After waiting for a few cycles, the response for the previous write transfer can be read [18]. The SDRAM can set waitrequest signal to "1" when it cannot respond to incoming read or write commands. When waitrequest signal is set to "1," the master IP halts all the transaction to the SDRAM till waitrequest is set to "0" by the SDRAM [18].

B. DRAM

Each DRAM chip is composed of multiple banks and each bank is composed of a 2-D array of memory cells that are arranged in rows and columns. Each row is connected to a wordline and has multiple memory cells. A typical 1T1C cell consists of a pass transistor and a capacitor. In current DRAM implementations, a true cell (anti-cell) stores a bit value of logic "1" ("0") when the capacitor is charged and a bit value of logic “0” (“1”) when discharged [16], [19]. During a read or write operation, the wordline connected to the target row is asserted and the value of the memory cells in the row is transferred to the row buffer. Since DRAM cells are composed of capacitors that leak their charge over time, they need to be continuously refreshed to ensure that they retain their values. The time interval between the charging cycles of the capacitors of DRAM cells is called “refresh interval” [16].

C. Rowhammer Attack

The rowhammer attack is caused by the repetitive activation of specific DRAM rows during read or write operations within the DRAM memory refresh interval. When aggressors are activated and deactivated multiple times during the refresh interval, they leak charge to the cells in neighboring memory rows (victims), which leads to the modification of the values of their stored bits.

Rowhammer attacks can be single sided or double sided. In single-sided attacks, one arbitrary memory address is repetitively accessed. In double-sided rowhammer, two addresses adjacent to a specific memory location are repetitively accessed. The values stored in the activated row needs to be transferred to the row buffer several times to induce bit flips. The experiments in [16] show that continuous accesses to the same row do not induce errors as the value will be transferred to the row buffer once and read from there in subsequent activation. Therefore, the rowhammer attack is effective only if different rows in the same bank are repetitively accessed in alternating patterns.

Rowhammer attacks compromise remote systems [20], [21] and are used in root privilege escalation [22], [23], and denial-of-service attacks on Intel Software Guard Extensions-based systems [24]. Recent work [25] demonstrates a rowhammer attack that remotely targets heterogeneous Intel FPGA-SoCs.

III. Related Prior Work

Countermeasures for rowhammer include increasing the memory refresh rate to reduce bit flips [26]–[28]. However, these countermeasures increase the system power consumption and reduces the memory throughput. Therefore, they are not so desirable in power-constrained systems and also when memory aging is a concern. Another problem is that this defense is bypassed by the double-sided rowhammer attack, when neighbors of the victim rows are repeatedly accessed [10].

Other proposed solutions identify anomalous behavior in performance counters that indicates the presence of rowhammer attacks. After a rowhammer attack is detected, victim rows are refreshed to mitigate the effects of the attack. The work in [10] monitors the miss rate of the last level of cache to identify rows from the same bank that are accessed with high frequency. Payer [9] detected rowhammer if the number of cache misses constitutes greater than 70% of the overall number of cache accesses. These approaches are not suitable for the detection of rowhammer that are launched from FPGA because caches are not involved in the FPGA-to-microprocessor SDRAM transactions.

Another countermeasure adds silicon overhead by integrating memory-row counters in the memory controllers or DRAM chips to count the frequency of activating specific rows. The victim rows are refreshed when a prespecified threshold on memory access frequency is reached. Kim *et al.* [29] implemented a counter for each memory row. To reduce the area overhead, Seyedzadeh *et al.* [11] divided each memory bank into groups of memory rows and assigned a counter to each group. Lee *et al.* [13] implemented a counter for only the active rows that are accessed with a frequency above a prespecified threshold. Vig *et al.* [14] integrated a module in the memory controller to monitor the memory transactions. Memory rows that are accessed more than once within a sliding window of ten addresses are regarded as a source of rowhammer attacks. Thus, the hash of their neighboring rows, which are considered as victim rows, are stored in a hash tree. The entries in the hash tree are compared with the actual values of the memory rows to identify bit flips.

These counter-based countermeasures target microprocessors. Therefore, they cannot identify which malicious IPs on the FPGA are launching the attack. Our proposed approach leverages the reconfiguration ability of the FPGA to implement an immediate solution to the attacks that are launched from the FPGA IPs.

IV. Motivation

Methods to secure memory in ASIC chips using counter-based countermeasures have been previously proposed [11], [13], [14], [29]. However, securing SoCs with FPGAs against rowhammer attacks using our proposed security monitor is particularly important due to the following reasons.

1) *Identify the Source of Attack:* The methods in [11], [13], and [29] monitor memory access requests. When a certain threshold on the number of memory access requests to a specific row is reached, the victim rows are refreshed. However, these methods do not identify the source of the attack. Thus, since the source of the attack is not identified, the attack persists in the system. Moreover, these countermeasures can reduce the performance of the overall system due to the repetitive refresh cycles applied to victim rows. Current memory modules waste 1.4%–35% of their operation time on memory refreshes [16]. The proposed approach avoids performance degradation due to persistent attacks by identifying the attackers.
2) *Ability to Detect Double-Sided Rowhammer Attacks:* Unlike counter-based measures [11], [29], the proposed approach does not keep an exclusive counter for each row. We monitor if any row is invoked more than twice within a fixed window. All repetitive accesses to all the rows are captured within a specific sliding window. Therefore, our approach can detect both single-sided and double-sided rowhammer attacks. A single-sided rowhammer attack requires repetitive access to the same specific row. A double-sided rowhammer attack requires repetitive access to the same two neighboring rows. In

our proposed approach, any repetitive access to any row within a specified window is flagged as a rowhammer attack. Thus, a double-sided rowhammer attack can be detected.

3) *Monitor Noncacheable Transactions:* The connection between the microprocessor and the SDRAM is cacheable. Thus, the detection of rowhammer attacks launched by the microprocessor can be detected by monitoring the cache miss patterns [9], [10]. However, some transactions that are issued from IPs to the SDRAM controller, and implemented on FPGAs, are noncacheable. Thus, there is a need for monitors implemented on FPGAs to exclusively monitor these noncacheable transactions.

A common practice to secure SDRAM memory used in FPGASoCs is to use an error-correcting code (ECC). Such ECC-based solutions were previously believed to secure SDRAM against rowhammer attacks [16]. However, the work in [30] shows that ECC can be bypassed by the Rowhammwer attackers.

V. Threat Model

In an FPGA-SoC, FPGAs can be configured with malicious 3rd party IPs (3PIPs) that can be one or more of the following types.

1) Implemented by untrusted vendors.
2) Altered during data transfer between the vendor and the FPGA, even if bitstreams are secured with encryption and authentication mechanisms. Bitstream encryption and authentication can be compromised by side-channel analysis [31]–[33]. In addition, side-channel analysis attacks can be remotely launched when the FPGA is shared among multiple tenants [34].
3) Injected with hardware Trojans that skip detection during the verification phase to launch rowhammer attacks [35].

We assume that the integration of the IP within the system is performed using trusted design-automation tools, and that the FPGA fabric is trusted. Therefore, the FPGA-to-microprocessor SDRAM interface and the security monitors are trusted. They cannot mask the rowhammer behavior of the malicious IPs. We assume that the 3PIPs are provided as bitstreams. Hence, the system integration tools do not have access to their HDL code. Thus, their code cannot be inspected for rowhammer activity. Also, these IPs do not require root privileges to launch rowhammer attacks. They can cause bit flips by repetitively requesting to read memory addresses that lie within their legitimate memory space in user mode [23].

VI. Proposed Countermeasure

The proposed security monitor, implemented on the FPGA fabric, observes the FPGA-to-microprocessor SDRAM memory-access transactions that are sent from IPs to each bank of the microprocessor SDRAM (see Fig. 1). SDRAM can include multiple banks to allow simultaneous memory access. The monitor determines which chip and bank each

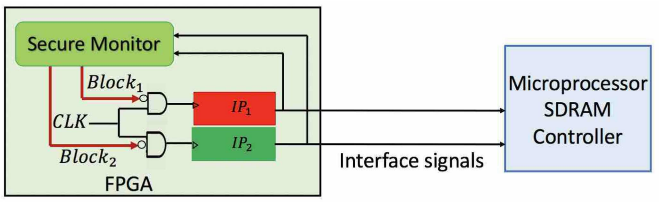


Fig. 1. Integration of our proposed security monitor in FPGA-SoC.

input address belongs to, and assigns the address to a specific sliding window W that corresponds to a specific chip and bank. The sliding window W for each SDRAM bank is a buffer that stores the row-address requests to each bank. It has a size of ψ. If a row address in the same memory bank is repeated for a prespecified threshold of N_{th} times among the ψ addresses, the monitor raises a rowhammer alarm and blocks the IPs that requested access to the repeated row address. The monitor blocks these malicious IPs by asserting a block signal to "1." This block signal is used to clock-gate the clock that synchronizes the operations of the malicious IPs. Note that given this setup, since the security monitor only listens to the signals associated with the FPGA-to-SDRAM interface, it does not make assumptions about the internal design of the IP.

In addition, in FPGAs that support partial reconfiguration, this block signal can control the FPGA configuration manager to remove the configuration of the malicious IPs from the FPGA. Partial reconfiguration is a feature of current-generation FPGAs that allows only a partition of the FPGA to be configured without interrupting the activity of the IPs configured on the rest of the FPGA [36]. Thus, the operations of the malicious IPs are halted when the IP block signal is asserted.

We select the values of ψ and N_{th} for single-sided rowhammer attacks according to the criteria proposed in [14] as follows. Kim *et al.* [16] investigated the effect of SDRAM activation interval (AI) at a refresh interval of 64 ms, which represents the time interval between the subsequent activations of the same row, and the number of induced bit flips. The results in [16] show that in the case of a single-sided rowhammer attack, when AI is equal to 500 ns, no bit flips are induced. This observation can be justified as follows; when a target row is activated once within 500 ns, the maximum number of row activations of this target memory address per the refresh interval of 64 ms is equal to 64 ms/500 ns = 128 K. According to the results of SDRAM characterization in [16], the minimum number of memory activations per row required to launch a successful single-sided rowhammer attack is 139 K activations per the refresh interval. Thus, the 128 K activations per the target memory row are smaller than the minimum number of activations per memory row that is required to cause bit flips in the SDRAM chip [16]. Therefore, we select $N_{th} = 2$.

We determine the value of ψ as follows: $\psi = AI/t_{rc}$, where t_{rc} is called row cycle time, which is the time taken by the memory to activate a row. In this work, we consider $t_{rc} = 50$ ns [14], [37]. Therefore, within a 500-ns interval, a maximum of ten row activations are possible. Thus, we select the minimum value of $\psi = 10$.

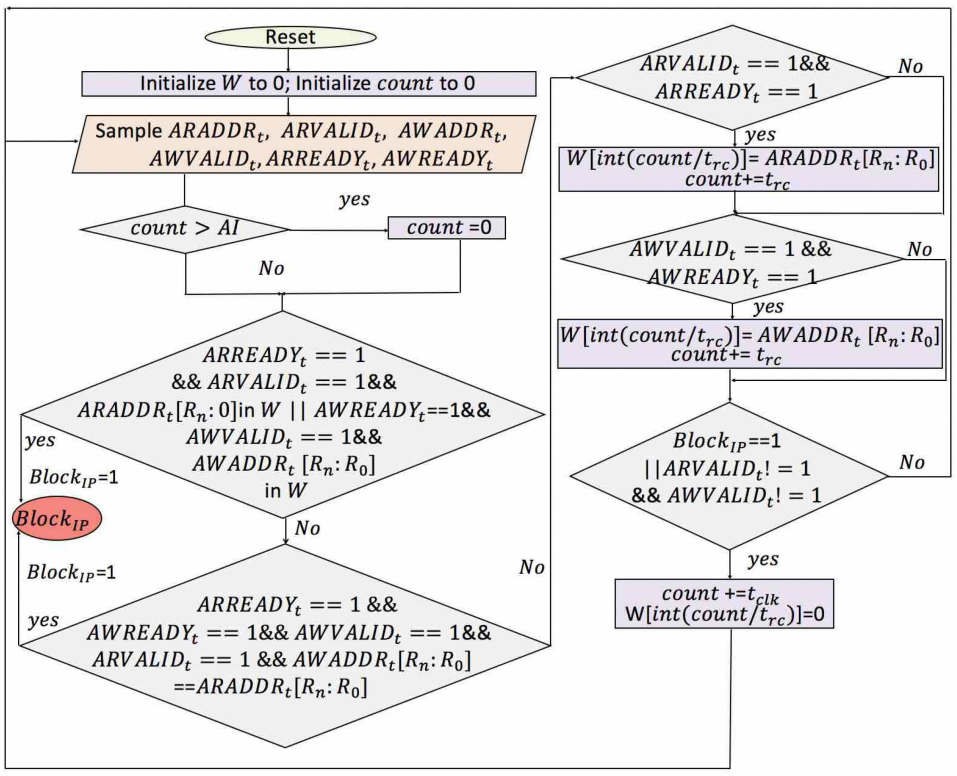

Reset
Initialize W to 0; Initialize $count$ to 0
Sample $ARADDR_t$, $ARVALID_t$, $AWADDR_t$, $AWVALID_t$, $ARREADY_t$, $AWREADY_t$
$count > AI$
yes
$count$ =0
No
$ARREADY_t == 1$ && $ARVALID_t == 1$&& $ARADDR_t[R_n:0]$in W || $AWREADY_t$==1&& $AWVALID_t == 1$&& $AWADDR_t\ [R_n:R_0]$ in W
yes
$Block_{IP}$=1
$Block_{IP}$
No
$Block_{IP}$=1
yes
$ARREADY_t == 1$ && $AWREADY_t == 1$&& $AWVALID_t == 1$&& $ARVALID_t == 1$ && $AWADDR_t[R_n:R_0]$ ==$ARADDR_t[R_n:R_0]$
No
$ARVALID_t == 1$&& $AREADY_t == 1$
No
yes
$W[int(count/t_{rc})]= ARADDR_t[R_n:R_0]$ count+=t_{rc}
$AWVALID_t == 1$ && $AWREADY_t == 1$
No
yes
$W[int(count/t_{rc})]= AWADDR_t\ [R_n:R_0]$ count+= t_{rc}
$Block_{IP}$==1 ||$ARVALID_t! = 1$ && $AWVALID_t! = 1$
No
yes
count +=t_{clk} W[$int(count/t_{rc})$]=0

Fig. 2. Flowchart for the operations of the proposed security monitor observing an IP with an AXI interface. R_n: R_0 corresponds to the indices of the row address within the physical address, where $n + 1$ is the number of row address bits.

For double-sided rowhammer attacks, a fewer number of memory activations within the activation interval is required to launch the attack. According to [13], 69 K activations are required in 64 ms. Therefore, to support the detection of double-sided rowhammer attacks, we consider $AI = 1000$ ns, and we double the size of the sliding window ψ, to ensure that the number of activation per memory row does not exceed 69 K. The maximum number of allowed activations per row under this policy is 64 ms/1000 ns = 64 K, which is lower than 69 K activations required per row to successfully launch double-sided rowhammer attacks.

Memory-access requests during both read and write operations can contribute to rowhammer attacks [38]. Therefore, we monitor the memory transactions during both read and write operations. The inputs to the monitor are as follows: 1) clock and reset signals that drive the FPGA-to-microprocessor SDRAM interface; 2) addresses requested by the configured IPs during the read and write operations that are sampled at the positive edge of the FPGA-to-microprocessor SDRAM interface clock ($ARADDR_t$ and $AWADDR_t$, respectively); 3) signals that represent the validity of the sampled addresses requested during the read and write operations, $ARVALID_t$ and $AWVALID_t$, respectively; and 4) signals that indicate that the SDRAM is ready to accept read and write addresses, *ARREADY* and *AWREADY*, respectively. The output of the security monitor is a signal called "$Block_{IP}$," which blocks the attacking IP.

The configured IPs might not request a read or write address every clock cycle (i.e., $ARVALID_t = 0$ and $AWVALID_t = 0$). In addition, there is likely to be a mismatch between the IP's operating frequency and the frequency of SDRAM row activation. For example, the IP and FPGA-to-microprocessor SDRAM clocks can have a time period (t_{clk}) of 5 ns and t_{rc} of 50 ns. Therefore, the sliding window W that keeps track of the memory transactions should not increment its indices by t_{rc} when an address is not received, in order to ensure that the delay between the first address and the last address in the sliding window is at least 500 ns. We achieve this in the monitor's design by updating the indices of the sliding window W based on a timer *count* that accumulates the time that has elapsed (in ns) from the capture of the first address of the sliding window. *Count* is incremented every clock cycle by t_{rc} only when a valid memory address is requested; otherwise, it is incremented by the value of the clock period. We increment based on the faster clock, if different IPs have different operating clock frequencies. A flowchart that describes the operations of the proposed monitor is shown in Fig. 2.

It is important to note that we propose a reconfigurable monitor that observes the memory transactions to identify possible rowhammer attack attempts. Therefore, it can be used with any integrated circuit that accesses DRAM memories. In this work, we particularly focus on securing FPGA-SoCs against rowhammer attacks that are launched from FPGAs because countermeasures against these attacks are lacking. Compared with performance counter-based countermeasures, our proposed method is particularly useful to detect rowhammer attacks that are launched by noncacheable memory transactions. These memory transactions do not involve caches.

Therefore, the proposed method can be used when rowhammer attacks cannot be detected by monitoring performance counter metrics such as cache-miss rate.

A. *Effect of Changing the IP Operating Frequency*

The memory access rate is bounded by the speed of memory row activation. For example, if an IP has a higher frequency than the memory controller, multiple memory addresses will be queued. If an IP has lower frequency and requests memory access at intervals of more than AI, a rowhammer attack cannot be launched. Designs that run at higher frequency (i.e., can issue more than two similar memory requests within AI) can launch rowhammer attacks.

B. *Detection of Stealthy Colluding IPs*

We consider the scenario where multiple IPs request access to the same memory bank at the same time in the presence of multiport memory controllers [7]. The memory controller will schedule the memory accesses one after the other. An example of colluding IPs is as follows. We consider that 3PIPs from different manufacturers can collude through malicious software controlling the different IPs. For example, if 3PIPs accept the target addresses as inputs, the malicious software can instruct the 3PIPs to access repetitive memory addresses.

The proposed monitor can be implemented independently for each IP on the FPGA. However, independent monitors cannot capture the operation of stealthy IPs that collude to launch a rowhammer attack. As shown in Fig. 3, an independent monitor for IP1 and IP2 will consider each IP to be benign because within the sliding window of ten addresses for each IP, no row address is repeated twice. However, a centralized monitor can detect that some addresses are repeated twice as they are accessed by each IP once. Therefore, a combined attack launched by IP1 and IP2 can only be detected in the presence of a centralized monitor. We design the monitor to give IPs priority according to numbers provided by the system integrator. For example, in Fig. 3, we assume that the memory serves one request from each IP starting from IP1.

We identify the attacking IP by assigning a block signal to each attacker. The block signal is assigned a value of "1," when an IP attempts to access a row address more than once within a sliding window of addresses of size ψ.

To detect colluding attackers, we consider the scenario that only the IP that requests the second request to the memory address is considered as an attacker. For example, in Fig. 3, only IP2 is considered as an attacker by the proposed security monitor. We adopt this scenario because if we alternatively consider both IP1 and IP2 as attackers, IP2 can deliberately access addresses that have been previously requested by IP1 to trigger the blocking of IP1 by the security monitor, and thereby, cause denial of service.

C. *Memory-Access Policy for Benign IPs*

The proposed security monitor blocks an IP if it attempts to access a row address more than once in a sliding window with ψ consecutive memory transactions. Therefore, we propose the

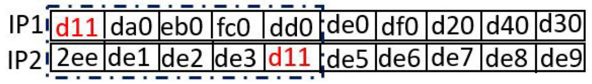


Fig. 3. Example of stealthy IPs. If IP1 and IP2 are monitored independently, they will be regarded as benign IPs, even though they are colluding to launch a rowhammer attack on the SDRAM.

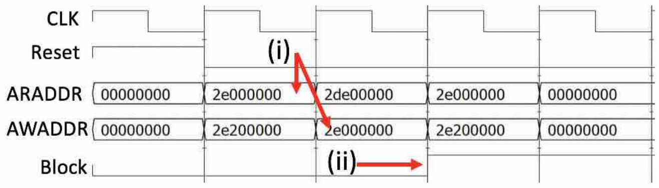


Fig. 4. Simulation of the operations of the monitor. (i) IP requests the access to the address "2e000000" during the read operation and sends a write request to the same address in the next clock cycle. (ii) Block signal is asserted to "1."

following memory-access policy to be utilized by benign IP vendors to avoid the blocking of their IPs.

1) No row address can be accessed by a benign IP more than once during a sliding window of size ψ. Section VI explains this choice of ψ.
2) Multiple IPs that belong to the same vendor cannot access the same row address more than once during a sliding window of size ψ.

With this memory-access policy, we can ensure that: 1) the number of false positives is zero because only malicious IPs will attempt to access the same row addresses more than once within the specified window and 2) the number of false negatives is zero because no IP can issue rowhammer access patterns without detection. Note that if a malicious IP adopts the policy for benign IPs, its memory-access behavior will not cause bit flips. Thus, it will be considered to be secure as far as rowhammer attacks are concerned.

VII. Experimental Setup and Results

A. *Experimental Platform*

We use Mentor Graphics ModelSim to simulate the behavior of the proposed security FPGA-to-microprocessor SDRAM monitor in the presence of malicious FPGA IPs. We evaluate the FPGA utilization of the proposed monitor using Intel Quartus II. We configure the synthesis tool in Quartus II to optimize the design for area. We demonstrate the effectiveness of the proposed monitor to detect malicious IPs on the SDRAM using an Intel DE1SoC board with a Cyclone V FPGA. The physical address is mapped in sequence into chip select, row address, bank address, and column address. We consider the presence of one-bit chip select, 15-bit row address, 3-bit bank address, and 10-bit column address [7].

B. *Detection of Malicious IPs That Connect to the SDRAM Using AXI Interface*

We simulate the operations of the malicious IP proposed in [8], which requests a read operation from addresses "2e000000" followed by write operation to the same address. As shown in Fig. 4, when the monitor observes the repeated

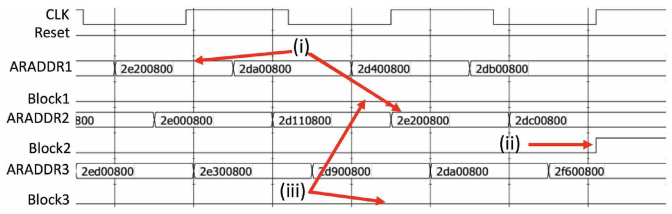


Fig. 5. Simulation of the localization of attacking IPs and the detection of colluding attackers. (i) IP1 requests the access to memory address “2e200800” and IP2 requests access to memory address “2e200800” one clock cycle after IP1’s request. (ii) IP2 is blocked. (iii) IP1 and IP3 continue their operations normally.

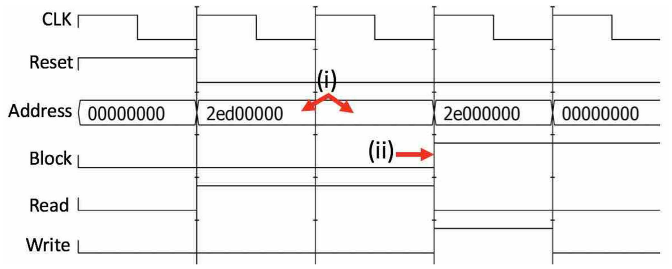


Fig. 6. Simulation of the proposed monitor successfully blocking a single rowhammer attack in the Avalon-MM interface. (i) IP repetitively requests read from the same row address. (ii) IP is blocked.

row address requested by the malicious IP, a block signal is asserted that blocks the operations of the malicious IP.

C. Identification of the Attackers That Connect to the SDRAM Using AXI Interface

We simulate the behavior of three IPs and identify the attacking IP, as shown in Fig. 5. In Fig. 5(i), IP1 first requests the access to memory address “2e200800”; then IP2 requests access to memory address “2e200800” one clock cycle after IP1’s request. Therefore, IP2 is blocked.

D. Identification of the Attackers That Connect to the SDRAM Using Avalon-MM Interface

The simulation result for single rowhammer attacks launched in the Avalon-MM interface is shown in Fig. 6. The attack is launched by requesting two continuous read operations on the same address 2ed00000. After the second request on 2ed00000 is detected, the monitor immediately asserted the block signal, disabling this potentially malicious IP.

E. Identification of the Attackers That Connect to the SDRAM Using Avalon-MM Interface

We simulate the behavior of six IPs and identify the attacking IP, as shown in Fig. 7. In Fig. 7(i), similar to Fig. 5, IP1 first requests the access to memory address “2e200800”; then IP2 requests access to memory address “2e200800” one clock cycle after IP1’s request. Therefore, IP2 is blocked.

F. Hardware Demo

We demonstrate the effectiveness of the proposed countermeasure with a hardware demo on DE1SoC board with Cyclone V (5CSEMA5F31C6) FPGA. The demo can be accessed in [39]. The proposed approach can be implemented in any Intel or Xilinx SoC board, for example, Xilinx Zynq-7000 SoC [5] and Intel Arria V Cyclone V [4] and Stratix 10 [40]. As shown in Fig. 8, the sliding window of the monitor is implemented as a buffer. The presence of repeated row addresses is determined by the presence of XNOR gates that compare the memory requests at each clock cycle to each entry in the buffer. The system integrator connects the outputs of each IP that is connected to the FPGA-to-microprocessor SDRAM interface to the inputs of the monitor. At each clock cycle, the *ARADDR*, *AWADDR*, *ARVALID*, *AWVALID*, *ARREADY*, and *AWREADY* of each IP are inputs to the monitor.

We configure the malicious IP implemented in [8] on the FPGA. This malicious IP attempts to launch rowhammer attacks on the shared SDRAM from the FPGA. The malicious IP is connected to the shared SDRAM controller through the FPGA-to-microprocessor SDRAM interface. This connection is configured using Intel Quartus Qsys as shown in Fig. 9. We observe the number of bit-flips in the SDRAM in the absence and presence of the proposed monitor as shown in Fig. 10(a) and (b), respectively. Fig. 10(a) shows one bitflip at memory address “00d6ab7” in the absence of the proposed monitor. Fig. 10(b) shows that when the proposed monitor is configured on the FPGA and activated, the rowhammer attack is detected and the malicious IP is blocked before it succeeds in causing any bit-flips. In order to observe the operations of the proposed monitor, we use the SignalTap II Logic Analyzer tool in Intel Quartus to observe the values of the requested memory addresses and the value of the block signal. The details of the monitor’s operations are shown in Fig. 11.

G. Evaluation of the Area Overhead of the Proposed Monitor

We investigate the area overhead of the proposed security monitor in terms of the number of the FPGA adaptive logic modules (ALMs). The ALM is the basic building block of Intel Cyclone V FPGAs, which includes full adders, a carry chain, a register chain, and a 64-bit LUT. In Intel Cyclone V, which is used in our experiments, the FPGA-to-microprocessor SDRAM interface has six command ports and four read and write ports. We consider as a proof of concept, master IPs that send a single data transfer per transaction (i.e., nonbursting transfers) on AXI and Avalon-MM interfaces.

1) AXI Interface: The AXI interface can simultaneously handle read and write commands. Therefore, using the AXI interface implementation, a maximum of three IPs can be configured to access the SDRAM in Cyclone V FPGAs. Hence, we investigate the area overhead of the monitor that observes the read and write memory transaction to a memory bank when one and three IPs are configured, respectively. The Quartus II utilization report shows that the monitor that observes one (three) IP (IPs) to detect single-sided rowhammer attacks utilizes 383 (1320) ALMs, respectively, which constitutes an ALM utilization of only 1.2% (4.1%) for one memory

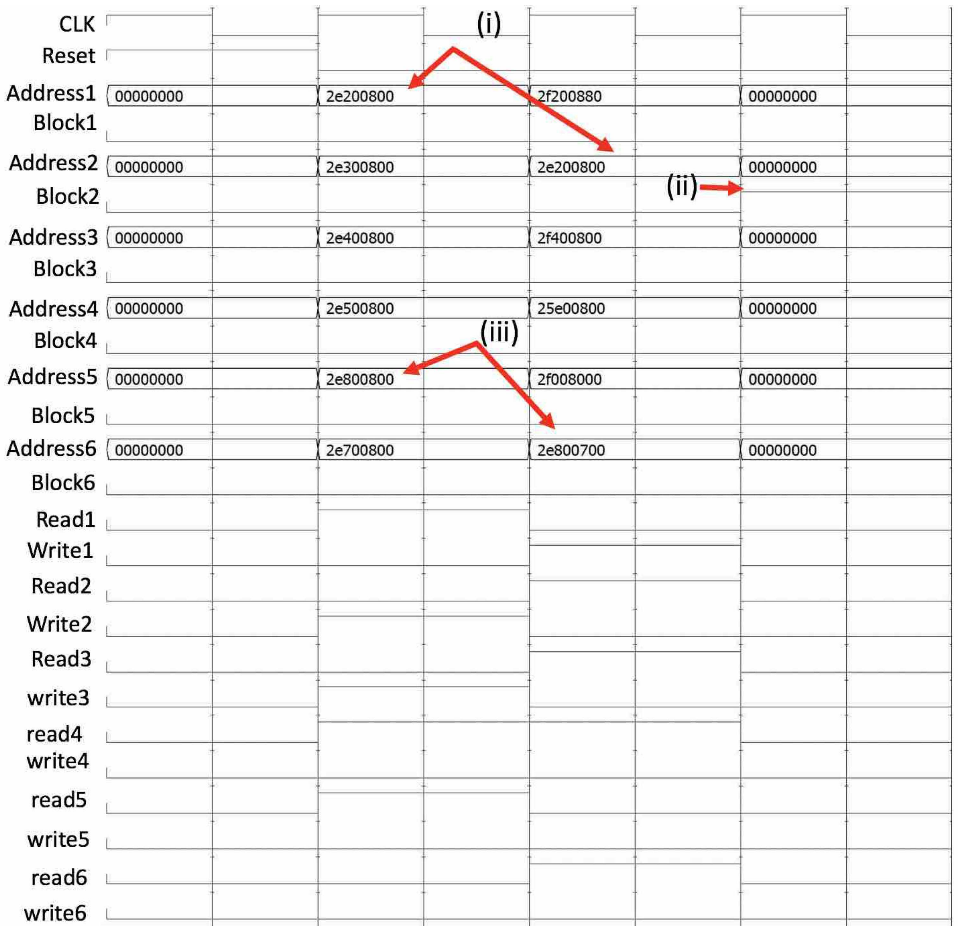


Fig. 7. Simulation of the proposed monitor successfully blocking a rowhammer attack in the Avalon-MM interface. (i) IP repetitively requests read from the same row address. (ii) IP2 is blocked. (iii) IP6 is not blocked even though it sends a read request to the same row address as IP5 because the requested row address is located in two different memory banks.

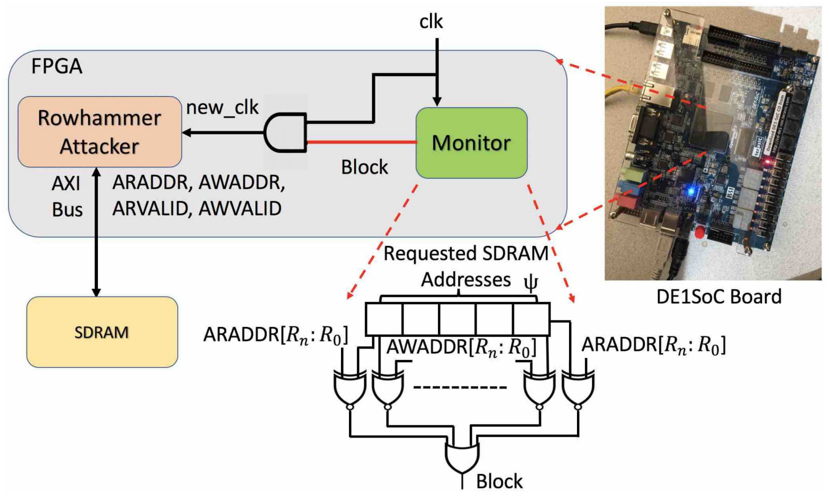


Fig. 8. Setup of the hardware demo for the proposed countermeasure.

bank. The ALM utilization for the detection of double-sided rowhammer attack for one (three) IP (IPs) is 585 (2013), which constitutes 1.8% (6.3%) overhead for one memory bank.

2) Avalon-MM Interface: Since the Avalon-MM interface does not allow simultaneous read and write commands, a maximum of six IPs can be configured in the Cyclone V FPGAs. Thus, we evaluate the area overhead of the monitor for listening to one and six IPs. The monitor that listens to one (six) IP(s) to detect single-sided rowhammer attacks utilizes 119 (1246) ALMs, respectively, which constitutes an ALM utilization of only 0.37% (3.9%) for one memory bank. The ALM utilization for the detection of double-sided rowhammer attack

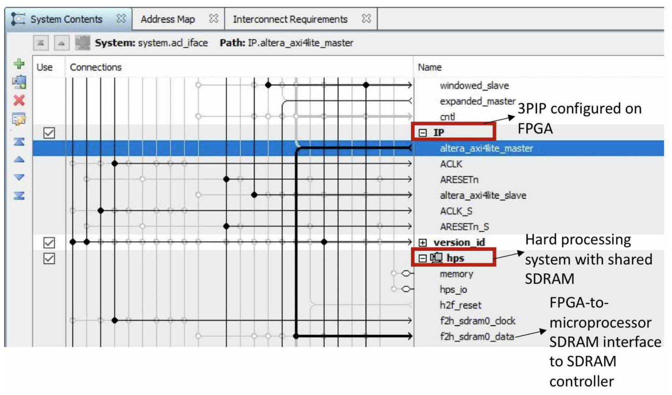
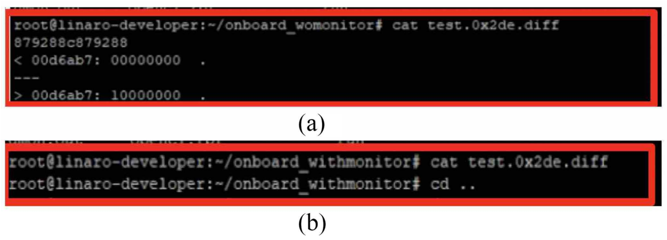


Fig. 9. Illustration of how the IP is connected to the SDRAM through the FPGA-to-microprocessor SDRAM interface in Intel Quartus Qsys.



(a)

```
root@linaro-developer:~/onboard_womonitor# cat test.0x2de.diff
879288c879288
< 00d6ab7: 00000000  .
---
> 00d6ab7: 10000000  .
```

(b)

```
root@linaro-developer:~/onboard_withmonitor# cat test.0x2de.diff
root@linaro-developer:~/onboard_withmonitor# cd ..
```

Fig. 10. Memory bitflips in the (a) absence and (b) presence of the proposed countermeasure.

for one (six) IP(s) is 181 (1918), which constitutes 0.56% (6.0%) overhead for one memory bank.

When a single IP communicates with the SDRAM through Avalon-MM interface, the secure monitor utilizes fewer ALMs compared to when an AXI interface is used. This observation can be attributed to the fact that the Avalon-MM interface has only one address signal that is shared between the read and write operations. In contrast, the AXI interface has two address signals. One address signal is dedicated for the read operations and the other one is dedicated for the write operations. Therefore, the monitor for the Avalon-MM interface compares only one address to the saved addresses, while the AXI monitor compares two.

Since DRAM chips have multiple banks, e.g., four or eight banks [41], the estimated area overhead for monitoring all the banks might not be practical for FPGAs with limited resources. Therefore, we propose to restrict IPs implemented on the FPGA to access a limited number of banks. The number of DRAM chip banks that the IPs are allowed to access depends on the logic utilization for each IP, so that the FPGA can fit the IPs and the monitors at the same time.

H. Power Overhead Estimation

In addition to adding a small area overhead, the proposed monitor also consumes additional power. We estimate the power overhead using the PowerPlay Power Analyzer in the Intel Quartus toolchain. The default toggle rate for I/O signals is set as 12.5%.

1) AXI Interface: The monitor that listens to one (three) IP(s) to detect single-sided rowhammer attacks consumes 424.34 (433.97) mW. The power consumption for the detection of double-sided rowhammer attack for one (three) IP(s) is 424.33 (433.91) mW. To further understand the data reported, we estimate the power of a single inverter with the same settings as 419.30 mW. We treat this as the baseline power, i.e., the power consumption of an empty FPGA fabric without any programmed logic, to maximize the estimated percentage overhead. Then, the single-sided rowhammer monitor that listens to one (three) IP(s) consumes 5.04 (14.67) mW additional power and 1.20% (3.50) of the baseline, while the double-sided rowhammer monitor that listens to one (three) IP(s) consumes 5.03 (14.61) mW additional power and 1.20% (3.48) of the baseline.

2) Avalon-MM Interface: The monitor that listens to one (six) IP(s) to detect single-sided rowhammer attacks consumes 421.92 (435) mW. Compared to the baseline power of the FPGA fabric, it consumes 2.62 (15.7) mW additional power when listening to one (six) IPs and 0.62 (3.74) % of the baseline. The power consumption for the detection of double-sided rowhammer attack for one (six) IP(s) is 421.91 (434.98) mW. Compared to the baseline power of the FPGA fabric, it consumes 2.61 (15.68) mW additional power when listening to one (six) IPs, 0.62% (3.74) of the baseline.

I. Comparison With Prior Work

Prior counter-based rowhammer detection approaches report their area overhead using different metrics. For example, [11] reports the area overhead as the percentage of the DRAM chip die area, while [14] reports only the number of bits that are required to store the hash tree used to determine the number of bit flips; it neglects the memory controller module that implements the sliding window approach to identify vulnerable memory rows. As a result, it is difficult to conclusively determine which method has the lowest area overhead. In Table III, we compare our proposed approach to prior work. The proposed approach is the only counter-based countermeasure that targets the security of SoCs against attacks launched from the FPGA. It can be immediately implemented in FPGA-SoCs available in the market. It is also the only method that utilizes FPGAs to provide security; thus, the added overhead is temporary and depends on the presence of IPs that access the SDRAM through FPGA-to-microprocessor SDRAM. Moreover, the proposed security monitor is the only approach that can identify the attacker IP on the FPGA and detect the attack early enough such that the victim rows do not require refreshing.

J. Evaluation of Overhead Due to Memory-Access Policy

In Section VI-C, we proposed a memory-access policy for benign IPs to eliminate potential false positives for our monitor. For some IPs, the policy can impose additional overhead in memory operations. In this section, we evaluate this overhead by investigating the memory access patterns of three designs that are representative of typical IPs accelerated on FPGAs: 1) machine learning [42]; 2) cryptography [43]; and 3) communications [44]. All three IPs can directly access SDRAM through the FPGA-to-SDRAM interface. In this article, the

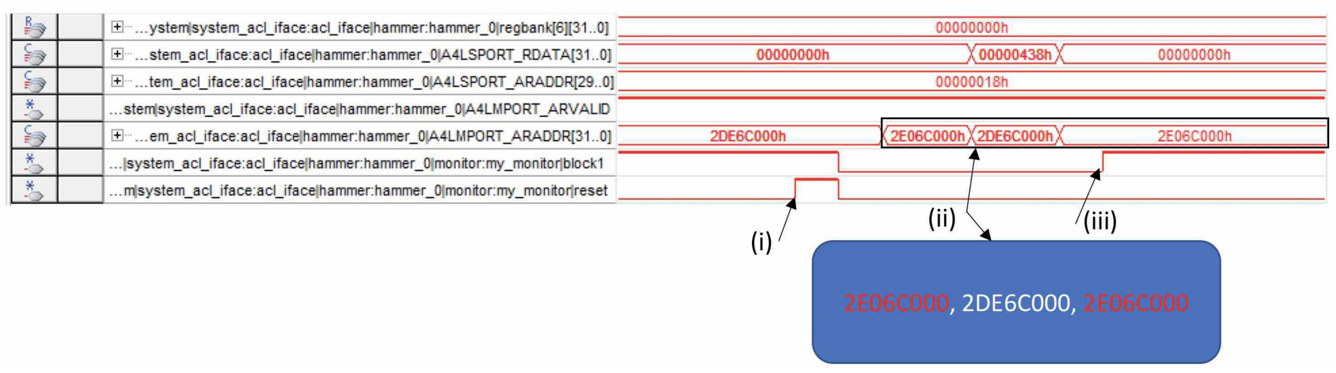

...ystem|system_acl_iface:acl_iface|hammer:hammer_0|regbank[6][31..0]
...stem_acl_iface:acl_iface|hammer:hammer_0|A4LSPORT_RDATA[31..0]
...tem_acl_iface:acl_iface|hammer:hammer_0|A4LSPORT_ARADDR[29..0]
...stem|system_acl_iface:acl_iface|hammer:hammer_0|A4LMPORT_ARVALID
...em_acl_iface:acl_iface|hammer:hammer_0|A4LMPORT_ARADDR[31..0]
...|system_acl_iface:acl_iface|hammer:hammer_0|monitor:my_monitor|block1
...m|system_acl_iface:acl_iface|hammer:hammer_0|monitor:my_monitor|reset
00000000h
00000000h
00000438h
00000000h
00000018h
2DE6C000h
2E06C000h
2DE6C000h
2E06C000h
(i)
(ii)
(iii)
2E06C000, 2DE6C000, 2E06C000

Fig. 11. Operations of the proposed monitor observed in the SignalTap II logic analyzer. (i) Monitor reset signal is asserted. (ii) Monitor reset signal is deasserted and memory requests are recorded in the sliding window. (iii) Block signal is asserted when repetitive access to the same row address is detected.

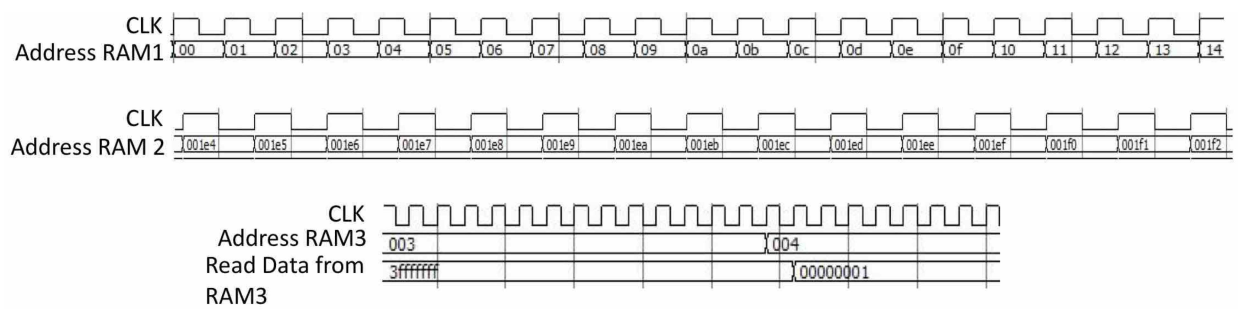

CLK
Address RAM1
00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14
CLK
Address RAM 2
001e4 001e5 001e6 001e7 001e8 001e9 001ea 001eb 001ec 001ed 001ee 001ef 001f0 001f1 001f2
CLK
Address RAM3
003
004
Read Data from RAM3
3fffffff
00000001

Fig. 12. Memory-access patterns during the operation of the SVM accelerator.

TABLE III
COMPARISON WITH PREVIOUS COUNTERMEASURES

	[13]	[11]	[14]	[29]	Proposed method
Location of Counter	Register Clock driver	DRAM chip	Memory Controller and DRAM chip	Memory Controller and DRAM Chip	FPGA
Attacker Identification	✗	✗	✗	✗	✓
Overhead	2.71KB per 1GB (DRAM bank)	1.29%-5.2% (DRAM die area)	1820-8184 bits on DRAM chip + non-evaluated checker overhead	0.0375% of main memory	6.3% FPGA utilization
Permanent/Temporary Overhead	Permanent	Permanent	Permanent	Permanent	Temporary
Victim Row Refresh Required ?	✓	✓	✓	✓	✗

designs studied are implemented by different developers to avoid any bias in the evaluation.

We set up the experiment as follows: in the worst case, each IP requires access to the SDRAM every clock cycle. A possible example of this scenario would be an IP that reads a value from a sensor and writes to the same memory address. For example, if it reads a new value every 50 ns into the same address, we suggest that it alternates between different locations. Thus, a maximum of 20 row addresses per IP need to be reserved to protect against double-sided rowhammer. These addresses constitute negligible overhead compared to the number of row addresses per memory bank, which is in the order of 8K–64K row addresses per bank, depending on the memory size.

If multiple IPs share the same memory location, it is more efficient to communicate through shared on-chip SRAM—which is not vulnerable to rowhammer—instead of SDRAM. For example, in the scenario when multiple IPs need to poll a specific memory address, a top module can be inserted to poll that memory address once and load its value inside an SRAM cell where all the other IPs can access without the risk of rowhammer attacks.

1) SVM Accelerator: Since machine-learning acceleration on FPGAs is becoming increasingly popular, we also investigate the memory access patterns of a support vector machine (SVM) accelerator [45]. We simulate the SVM accelerator in ModelSim and monitor the memory access patterns. The accelerator accesses three RAM memories during its operation. Two RAM memories hold the values of the input image to be classified and the support vectors. As shown in Fig. 12, the accelerator does not access repeated addresses. Therefore, if the SVM accelerator accesses SDRAM, enacting the proposed policy will not add any additional overhead. The accelerator also accesses a specific address in the third RAM for one cycle and keeps the address signal assigned to the same value for multiple consecutive clock cycles. During these multiple consecutive clock cycles, the returned data from RAM3 are constant. In this case, if this RAM is implemented in SDRAM, the address will be read once from the SDRAM and then the ARVALID will be deasserted and the read data will be

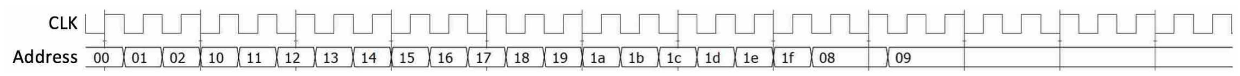


Fig. 13. Memory-access patterns during the operation of the SHA-1 hash core.

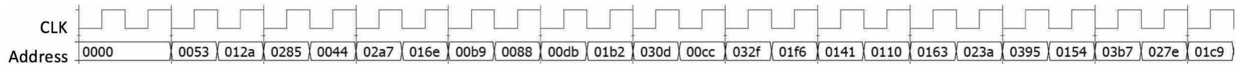


Fig. 14. Memory-access patterns during the operation of the turbo-code interleaver core.

stored in an on-chip SRAM for repetitive access during the rest of the clock cycles. This strategy will require us to reserve only one memory location on the on-chip SRAM to store the data that is read from the SDRAM. Thus, the proposed memory-access policy incurs negligible overhead in practical scenarios. Also, even if the SVM accelerator developers do not implement the proposed memory-access policy, the proposed monitor will block rowhammer attacks with 0% false positive rate.

2) SHA-1 Hash Core: In addition, we simulated the memory access behavior of a SHA-1 hash core [46] to determine its overhead under the proposed policy. A SHA-1 hash core implements a widely used hash algorithm to authenticate transferred data. It is usually connected as a peripheral component in an SoC via a bus interface. Based on the simulated memory-access pattern, we determine if the proposed policy imposes any overhead.

The implementation of SHA-1 core in [46] constitutes a wrapper that implements an on-chip SRAM memory array to store the block data to be hashed. This SRAM memory array is populated with data from a testbench. Next, the SHA-1 core reads the block data from the SRAM memory locations. The testbench waits for the SHA-1 core to finish its processing before it starts reading the output of the SHA-1 core.

In our evaluation, we consider the testbench as a process running on the microprocessor that writes the block data to different location in the SDRAM memory. We also consider that the SHA-1 core is implemented on FPGA and acceses the SDRAM through FPGA-to-microprocessor SDRAM interconnect. We are interested in the addresses that the testbench accesses. In practical scenarios, these addresses are equivalent to the SDRAM memory addresses that the SHA-1 core accesses to read the data to be hashed.

Fig. 13 shows that the testbench writes to a sequence of different SRAM memory locations (from 0×00 to 0×1f) with no repetition, and then waits for a ready signal to be asserted. During the phase when it is waiting for the ready signal, it accesses the same location multiple times. In practical scenarios, the read valid signal will be deasserted during this waiting phase. Thus, this behavior will not alert the monitor at the FPGA side to block the SHA-1 core; the SHA-1 core only accesses nonrepetitive memory addresses to read the data to be hashed. After the SHA-1 core finishes its operations, it writes the ready signal to a specific memory location and starts writing the output of the hash operations to nonrepetitive memory locations in the SDRAM memory. Since memory addresses are not repetitively accessed during the operations of the SHA-1 core, the proposed monitor incurs 0% false positive rate even if the SHA-1 developers do not follow the proposed benign memory-access policy.

3) Turbo-Code Interleaver: The third IP that we evaluate is a custom-designed interleaver component in a turbo encoder [47]. The turbo encoder is an essential component in modern communication systems. The interleaver is a subcomponent in the encoder that carries out encryption functions. The interleaver implementation performs encryption by accessing precomputed values stored in two RAMs. In theory, these RAMs can be local SRAMs. However, we consider the case that a high throughput is needed [4]; therefore, SDRAM is used.

Upon boot, the interleaver first writes precomputed patterns to two RAMs, and then performs encoding by reading from these RAMs. The signals of interest in our experiment are the memory addresses of the two RAMs. We evaluate their memory-access behavior in the encoding of a code block. In the simulation, the interleaver first writes to a sequence of different memory locations addressed by a counter that increments by 1 every cycle, and then reads from different locations to perform encryption. The addresses associated with reading are related to the input from the previous subcomponent. After analyzing the sequence of memory locations in Fig. 14, we conclude that there is no overhead associated with the proposed policy for the turbo interleaver because it does not access the same memory location. In addition, in the scenario, when the turbo interleaver developer does not implement the proposed memory-access policy, the false positive rate will be 0%.

To verify that the experiments we performed in simulation match the real scenario onboard, we demonstrate further that when the turbo-code interleaver IP is programmed on the DE1-SoC together with our proposed monitor, it can perform normal operations. We show our test setup in Fig. 15 and the SignalTap capture of internal signals in Fig. 16.

The IP under test is an automated test setup that reads inputs from a test-input memory, feeds the inputs to the turbo-code interleaver system, and then compares the outputs with a set of golden outputs stored in another memory. We add the monitor similar to the simulation setup: two monitor instances listen to the memory transactions of two internal memory blocks of the turbo-code interleaver. The reset signal and the start-test control signal of the IP are programmed to switches onboard and the block signals of the two monitors are NORed and mapped to a LED onboard, so that if any of the block signals become high, the LED will switch off. During the entire testing sequence, we do not observe the LED switching off. This shows that the benign IP memory accesses are not blocked.

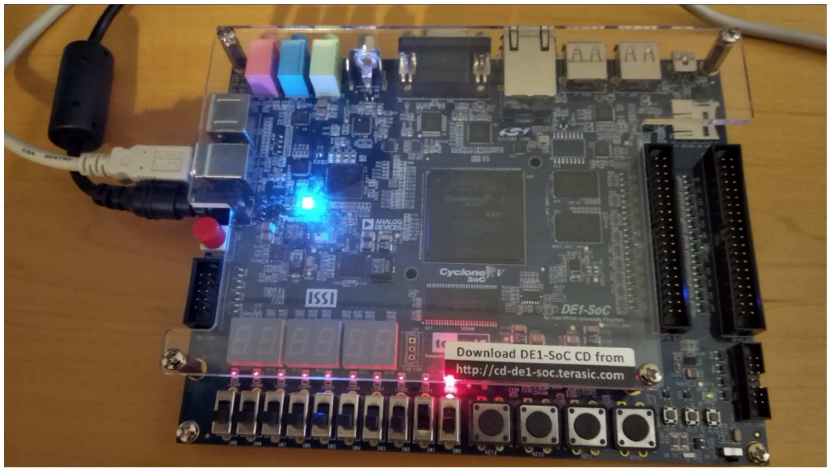


Fig. 15. Onboard test setup.

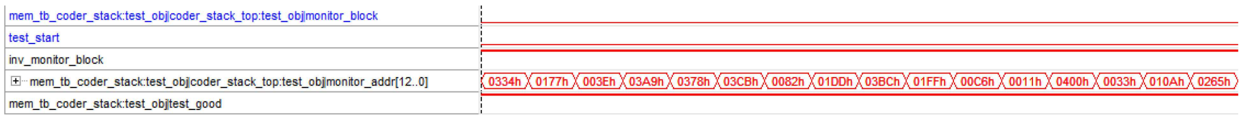

mem_tb_coder_stack:test_obj|coder_stack_top:test_obj|monitor_block
test_start
inv_monitor_block
mem_tb_coder_stack:test_obj|coder_stack_top:test_obj|monitor_addr[12..0]
0334h 0177h 003Eh 03A9h 0378h 03CBh 0082h 01DDh 03BCh 01FFh 00C6h 0011h 0400h 0033h 010Ah 0265h
mem_tb_coder_stack:test_obj|test_good

Fig. 16. SignalTap results of the turbo-code interleaver IP programmed onboard with monitors.

We further probe the internal signals of the IP using SignalTap. As shown in Fig. 16, the test_good signal of the automatic testing system is always held high, and the blocksignal always held low. We also show the address access patterns. From this experiment, we conclude that the monitors' presence do not interfere with the normal operations of the turbo-code interleaver. Therefore, we have tested the proposed monitor on an actual board not only in simulations. To control the effect of temperature in real-life systems, we suggest that the system administrators control the die temperature to lie within a fixed range. The die temperature can be monitored through on-chip temperature sensors. There are built-in sensors in almost all FPGAs from different vendors and they can be used to control the board fan speed. In this way, when our monitor is integrated, we can make sure we have control over the die temperature.

It is important to note that the benign memory-access policy should be enforced even if the proposed monitor detects rowhammer attacks with false positive rate for the designs studied in this section. Developers can implement the same IPs and get FPR that ranges from 0% to 100%. It depends on how the IP design is implemented to request access to the SDRAM. Therefore, we propose that the IP developers enforce the benign memory-access policy to ensure that their benign IPs are not blocked during their normal operations.

VIII. Discussion

A. Extension to Other Memory Interfaces

In this article, we have shown that our proposed monitor can be adapted to both the AXI and Avalon-MM interfaces, two widely used memory interfaces in FPGA-SoCs. Here, we describe how the monitor can also be extended to other interfaces that may become popular in the future.

In essence, the proposed monitor requires information on: 1) address of a memory transaction; 2) whether the transaction is a read or a write; and 3) validity of the memory transaction. Since these pieces of information are essential to define a memory transaction, they should be available in all memory interfaces. Therefore, following the examples on AXI and Avalon-MM, we can easily adapt the proposed monitor by mapping the corresponding signals given the three categories listed above. Whether the memory interface allows simultaneous read and write transactions is another consideration that we also demonstrated in the comparison between AXI and Avalon-MM interfaces.

B. DDR3 Versus DDR4

While our monitor is developed based on prior work on DDR3 memory, we believe that it can adapt to new attacks on more recent DDR4 memory. Although DDR4 was initially thought to be secure against Rowhammer attacks, a recent attack, TRRespass [48] successfully launches Rowhammer attacks on DDR4 using more than two aggressor rows. Nevertheless, the strategy that our monitor uses, i.e., checking for repeated accesses on a single row, will still be applicable to this new attack with a few adaptations in the sliding window size and detection logic. However, this adaptation will increase the area and power consumption overhead of the proposed monitor. In addition, the false positive rate might increase. We will address these issues in details as part of future work.

C. Complete Halting of Malicious IPs Versus TRR

In some scenarios, the system administrators can choose between complete halting of malicious IPs and the launch of TRR cycles. Our proposed approach presents a conservative method to prevent increased power consumption that results from the repetitive launching of TRR cycles, especially in power-restricted platforms. However, the launching of TRR cycles can be a valid solution in systems with relaxed restrictions over power consumption. Thus, we suggest an adaptive countermeasure policy. System administrators can configure the security monitor to block suspicious IPs if there are severe limits on power consumption. For example, in systems that run on batteries, if batteries are less than 40% charged, the security monitor should block any attacking IPs. Otherwise, the security monitor can force TRR cycles.

IX. Conclusion

We have presented an effective and efficient countermeasure against rowhammer attacks in FPGA-SoCs. This approach is based on an FPGA-to-microprocessor SDRAM security monitor that guarantees the detection of the rowhammer-attack attempts launched from the FPGA before the rowhammer effect is propagated to the DRAM chips. The proposed monitor utilizes only 6.3% of the Cyclone V FPGA ALMs for one memory bank when either AXI or Avalon-MM interfaces are used with nonbursting transfers. It can also be immediately integrated in current FPGA-SoCs.

Acknowledgment

The authors would like to thank Jonti Talukdar at Duke University for his help with the on-board experiment of the turbo-code interleaver.

References

[1] (2018). *Xilinx Launches the World's Fastest Data Center and AI Accelerator Cards*. [Online]. Available: https://bit.ly/2ltjtgZ

[2] *FPGA for Automotive Applications*. Accessed: Jul. 14, 2020. [Online]. Available: https://bit.ly/2NjxPtK

[3] *Architecture Brief: What Is an SoC FPGA?* Accessed: Jul. 14, 2020. [Online]. Available: https://intel.ly/2ko0LHA

[4] *Memory Inerconnects in Intel FPGA-SoCs*. Accessed: Jul. 14, 2020. [Online]. Available: https://intel.ly/2jYySWa

[5] (2021). *Zynq-7000 SoC Technical Reference Manual*. [Online]. Available: https://bit.ly/2Zfgyup

[6] (2018). *Xilinx Design Flow for Intel FPGA and SoC Users*. [Online]. Available: https://bit.ly/2jY27bL

[7] (2019). *External Memory Interface Handbook Volume 3: Reference Material*. [Online]. Available: https://intel.ly/35irS98

[8] S. Chaudhuri, "A security vulnerability analysis of SoCFPGA architectures," in *Proc. ACM 55th Annu. Design Autom. Conf. (DAC)*, 2018, pp. 1–6.

[9] M. Payer, "HexPADS: A platform to detect 'stealth' attacks," in *Proc. Springer 8th Int. Symp. Eng. Secure Softw. Syst. (ESSoS)*, 2016, pp. 138–154.

[10] Z. B. Aweke *et al.*, "ANVIL: Software-based protection against next-generation Rowhammer attacks," *ACM SIGARCH Comput. Architect. News*, vol. 44, no. 2, pp. 743–755, 2016.

[11] S. M. Seyedzadeh, A. K. Jones, and R. Melhem, "Counter-based tree structure for row hammering mitigation in dram," *IEEE Comput. Archit. Lett.*, vol. 16, no. 1, pp. 18–21, Jan.–Jun. 2017.

[12] M. Son, H. Park, J. Ahn, and S. Yoo, "Making DRAM stronger against row hammering," in *Proc. 54th ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Austin, TX, USA, 2017, pp. 1–6.

[13] E. Lee, I. Kang, S. Lee, G. E. Suh, and J. H. Ahn, "TWiCe: Preventing row-hammering by exploiting time window counters," in *Proc. 46th Int. Symp. Comput. Archit.*, Phoenix, AZ, USA, 2019, pp. 385–396.

[14] S. Vig, S. Bhattacharya, D. Mukhopadhyay, and S.-K. Lam, "Rapid detection of Rowhammer attacks using dynamic skewed hash tree," in *Proc. ACM 7th Int. Workshop Hardw. Archit. Support Security Privacy (HASP)*, 2018, p. 7.

[15] *FPGAs for Avionics Applications*. Accessed: Jul. 14, 2020. [Online]. Available: https://www.xilinx.com/applications/aerospace-and-defense/avionics-uav.htmll

[16] Y. Kim *et al.*, "Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors," *ACM SIGARCH Comput. Archit. News*, vol. 42, no. 3, pp. 361–372, 2014.

[17] *AXI Interface Protocol*. Accessed: Jul. 14, 2020. [Online]. Available: https://bit.ly/2POfRSd

[18] (2021). *Avalon® Interface Specifications*. [Online]. Available: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/manual/mnl_avalon_spec.pdf

[19] X.-C. Wu, T. Sherwood, F. T. Chong, and Y. Li, "Protecting page tables from RowHammer attacks using monotonic pointers in DRAM true-cells," in *Proc. ACM 24th Int. Conf. Archit. Support Program. Lang. Oper. Syst.*, 2019, pp. 645–657.

[20] M. Lipp *et al.*, "Nethammer: Inducing Rowhammer faults through network requests," 2018. [Online]. Available: arXiv:1805.04956.

[21] D. Gruss, C. Maurice, and S. Mangard, "Rowhammer. js: A remote software-induced fault attack in javascript," in *Proc. Int. Conf. Detect. Intrusions Malware Vulnerability Assessment*, 2016, pp. 300–321.

[22] M. Seaborn and T. Dullien, *Exploiting the DRAM Rowhammer Bug to Gain Kernel Privileges*, Black Hat USA, Las Vegas, NV, USA, 2016.

[23] V. D. Veen *et al.*, "Drammer: Deterministic Rowhammer attacks on mobile platforms," in *Proc. ACM SIGSAC Comput. Commun. Security Conf. (CCS)*, 2016, pp. 1675–1689.

[24] Y. Jang, J. Lee, S. Lee, and T. Kim, "SGX-Bomb: Locking down the processor via Rowhammer attack," in *Proc. ACM 2nd Workshop Syst. Softw. Trust. Execution (SysTEX)*, 2017, p. 5.

[25] Z. Weissman, T. Tiemann, D. Moghimi, E. Custodio, T. Eisenbarth, and B. Sunar, "JackHammer: efficient Rowhammer on heterogeneous FPGA-CPU platforms," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2020, no. 3, pp. 169–195, Jun. 2020.

[26] (2017). *About the Security Content of Mac EFI Security Update 2015-001*. [Online]. Available: https://support.apple.com/en-us/HT204934

[27] *Lenovo Rowhammer Mitigation*. Accessed: Jul. 14, 2020. [Online]. Available: https://support.lenovo.com/us/en/product_security/row_hammer

[28] (2015). *HP Moonshot Component Pack 2015.05.0Release Notes*. [Online]. Available: https://bit.ly/2lRZHvD

[29] D.-H. Kim, P. J. Nair, and M. K. Qureshi, "Architectural support for mitigating row hammering in DRAM memories," *IEEE Comput. Archit. Lett.*, vol. 14, no. 1, pp. 9–12, Jan.–Jun. 2015.

[30] L. Cojocar, K. Razavi, C. Giuffrida, and H. Bos, "Exploiting correcting codes: On the effectiveness of ECC memory against Rowhammer attacks," in *Proc. IEEE Symp. Security Privacy (SP)*, 2019, pp. 55–71.

[31] A. Moradi, D. Oswald, C. Paar, and P. Swierczynski, "Side-channel attacks on the bitstream encryption mechanism of Altera Stratix II: Facilitating black-box analysis using software reverse-engineering," in *Proc. ACM/SIGDA Int. Symp. Field Program. Gate Arrays*, 2013, pp. 91–100.

[32] A. Moradi, A. Barenghi, T. Kasper, and C. Paar, "On the vulnerability of FPGA bitstream encryption against power analysis attacks: Extracting keys from Xilinx Virtex-II FPGAs," in *Proc. 18th ACM Conf. Comput. Commun. Security (CCS)*, 2011, pp. 111–124.

[33] T. W. Kim, T. H. Kim, and S. Hong, *Breaking Korea Transit Card With Side-Channel Analysis Attack-Unauthorized Recharging*, Black Hat USA, Las Vegas, NV, USA, 2017.

[34] F. Schellenberg, D. R. E. Gnad, A. Moradi, and M. B. Tahoori, "An inside job: Remote power analysis attacks on FPGAs," in *Proc. Design Autom. Test Eur. Conf. Exhibit. (DATE)*, Dresden, Germany, 2018, pp. 1111–1116.

[35] K. Xiao, D. Forte, Y. Jin, R. Karri, S. Bhunia, and M. Tehranipoor, "Hardware trojans: Lessons learned after one decade of research," *ACM Trans. Design Autom. Electron. Syst.*, vol. 22, no. 1, p. 6, 2016.

[36] (May 2011). *Cyclone V Device Family Advance Information Brief*. [Online]. Available: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/cyclone-v/cyv_51001.pdf

[37] (2018). *512Mx8, 256Mx16 4Gb DDR3 SDRAM*. [Online]. Available: https://bit.ly/2RldCF2

[38] O. Mutlu and J. S. Kim, "Rowhammer: A retrospective," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 8, pp. 1555–1571, Aug. 2020.

[39] *FPGA Demo*. Accessed: Jul. 14, 2020. [Online]. Available: https://bit.ly/2EdsGQH

[40] (2020). *AN 802: Intel® Stratix® 10 SoC Device Design Guidelines*. [Online]. Available: https://intel.ly/2jY2ByB

[41] *Using SDRAM in DE1SoC*. Accessed: Jul. 14, 2020. [Online]. Available: https://intel.ly/2s9gdL4

[42] M. Papadonikolakis and C.-S. Bouganis, "Novel cascade FPGA accelerator for support vector machines classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 7, pp. 1040–1052, Jul. 2012.

[43] C. Xiao-Hui and D. Jian-Zhi, "Design of SHA-1 algorithm based on FPGA," in *Proc. 2nd Int. Conf. Netw. Security Wireless Commun. Trust. Comput.*, vol. 1. Wuhan, China, 2010, pp. 532–534.

[44] *Xilinx 3GPP LTE Turbo Encoder*. Accessed: Jul. 14, 2020. [Online]. Available: https://www.xilinx.com/products/intellectual-property/do-di-tccenc-lte.html

[45] *SVM Gaussian Classifier*. Accessed: Jul. 14, 2020. [Online]. Available: https://bit.ly/38ALuXW

[46] *SHA-1 Hash Core*. Accessed: Jul. 14, 2020. [Online]. Available: https://github.com/secworks/sha1

[47] *3GPP LTE Channel Coder*. Accessed: Jul. 14, 2020. [Online]. Available: https://github.com/SiyuanChen-Nanjing/coder-stack

[48] P. Frigo *et al.*, "TRRespass: Exploiting the many sides of target row refresh," in *Proc. IEEE Symp. Security Privacy (SP)*, San Francisco, CA, USA, May 2020, pp. 747–762.

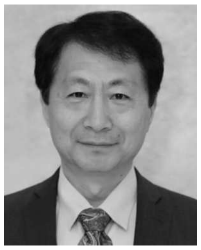
Rana Elnaggar (Member, IEEE) received the Ph.D. degree in electrical and computer engineering from Duke University, Durham, NC, USA, in 2020.

Her research interests include machine learning and hardware security.

Dr. Elnaggar thesis ranked third in the semi-finals of the 2021 TTTC's E. J. McCluskey Doctoral Thesis competition. She serves as a reviewer for multiple IEEE and ACM journals.

Siyuan Chen received the B.S.E. degree from Duke University, Durham, NC, USA, in 2020. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, USA.

His research interest includes VLSI design.



Peilin Song (Senior Member, IEEE) received the Ph.D. degree in electrical engineering from the University of Rhode Island, Kingston, RI, USA, in 1997.

He is a Principal Research Staff Member with the IBM Thomas J. Watson Research Center, Yorktown Heights, NY, USA, where he manages the Circuit Diagnostics and Testing Technology Department. He joined IBM in 1997 and has since worked in the area of design for testability, fault diagnostics, and recently hardware security and reliability.

Krishnendu Chakrabarty (Fellow, IEEE) received the B.Tech. degree from the Indian Institute of Technology Kharagpur, Kharagpur, India, in 1990, and the M.S.E. and Ph.D. degrees from the University of Michigan at Ann Arbor, Ann Arbor, MI, USA, in 1992 and 1995, respectively.

He is currently the John Cocke Distinguished Professor and the Department Chair of Electrical and Computer Engineering with Duke University, Durham, NC, USA.

Dr. Chakrabarty is a Fellow of ACM and AAAS, and a Golden Core Member of the IEEE Computer Society.