

TRRespass: Exploiting the Many Sides of Target Row Refresh

P. Frigo, E. Vannacci, H. Hassan, V. Van der Veen,
O. Mutlu, C. Giuffrida, H. Bos, K. Razavi



SAFARI

Qualcomm



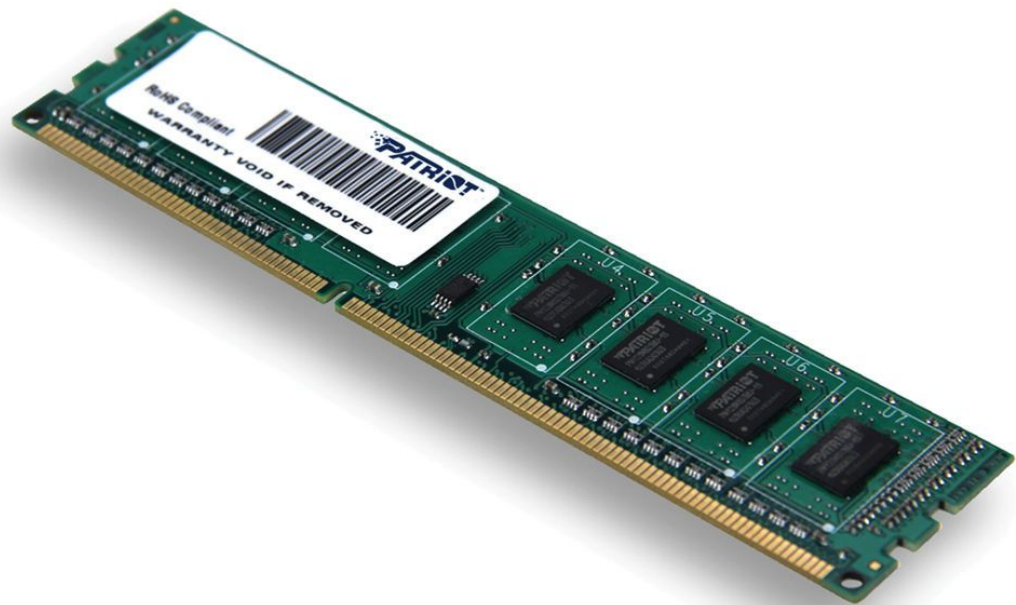
Emanuele Vannacci



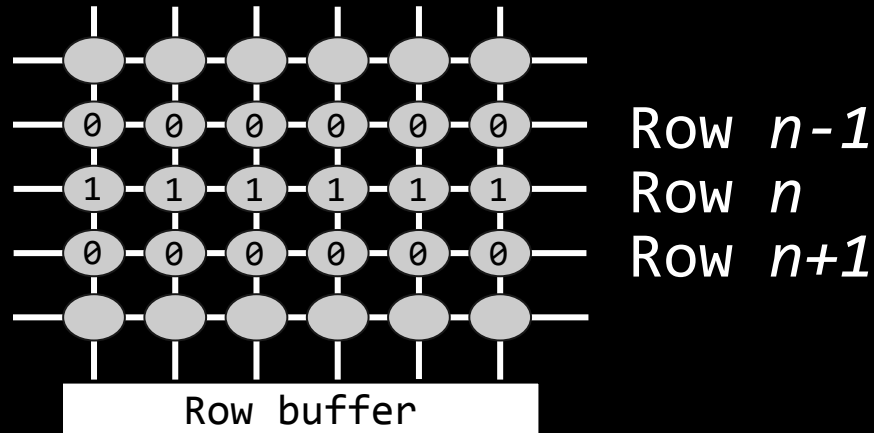
Pietro Frigo

The rise and rise and rise of

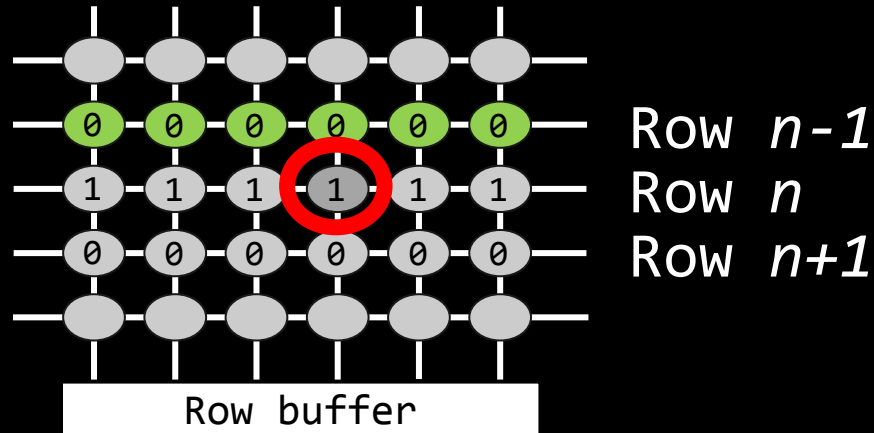
Rowhammer



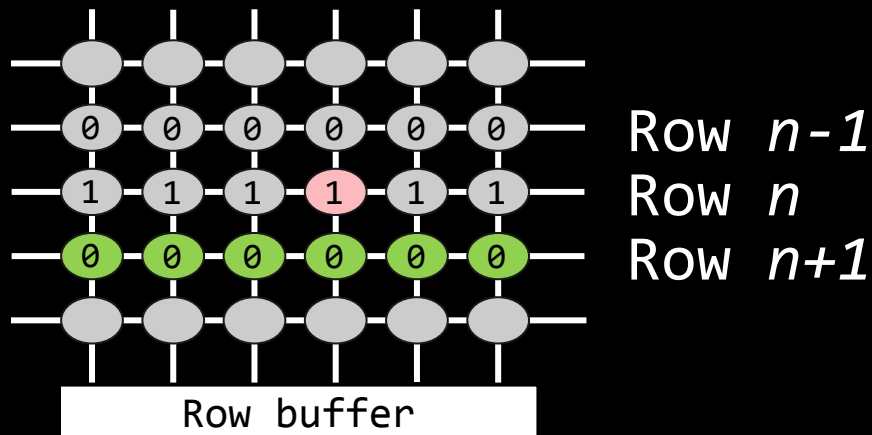
Rowhammer



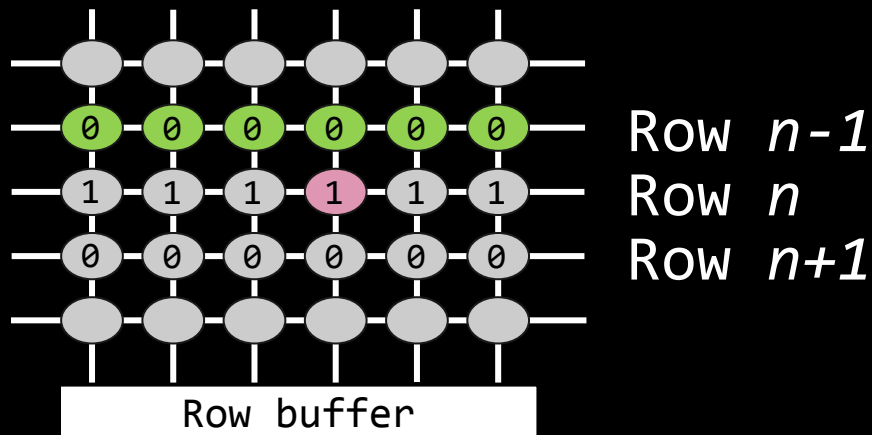
Rowhammer



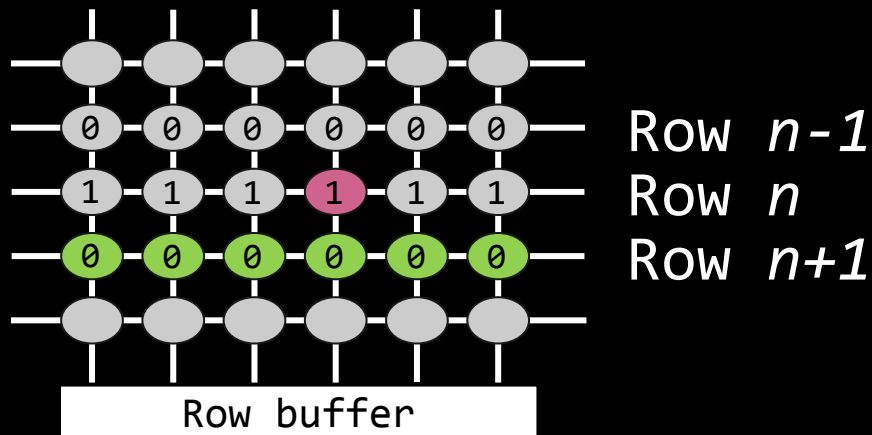
Rowhammer



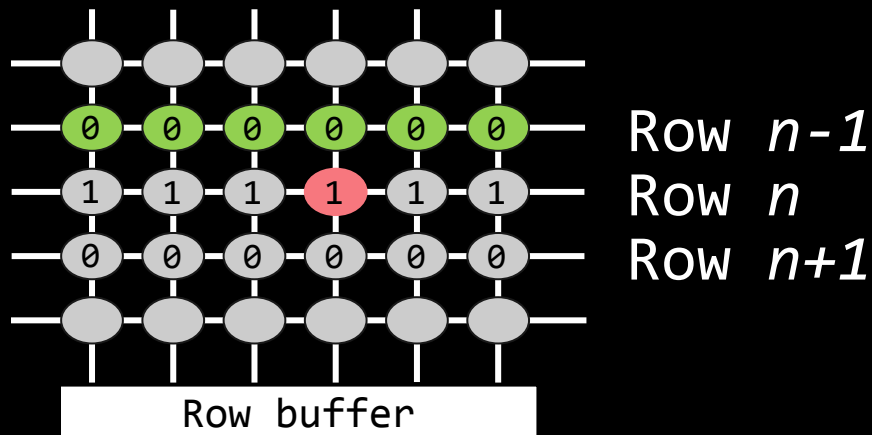
Rowhammer



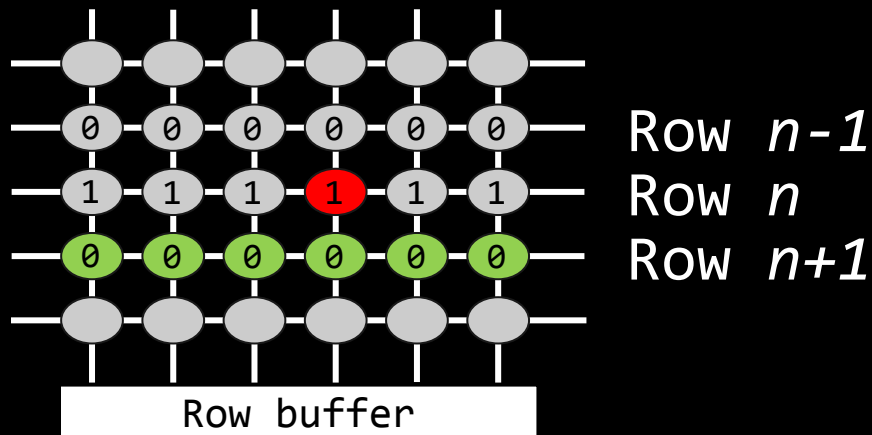
Rowhammer



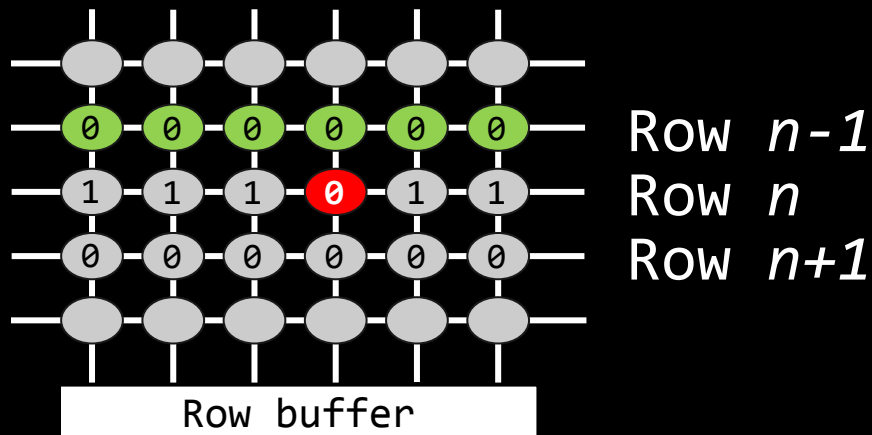
Rowhammer



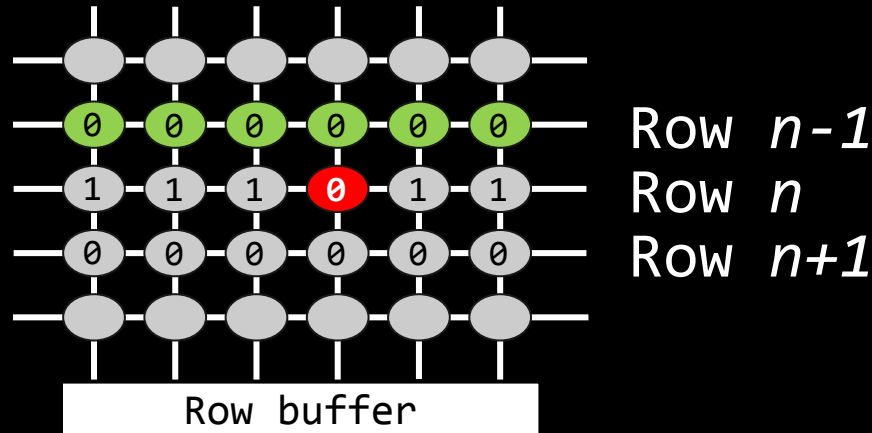
Rowhammer



Rowhammer

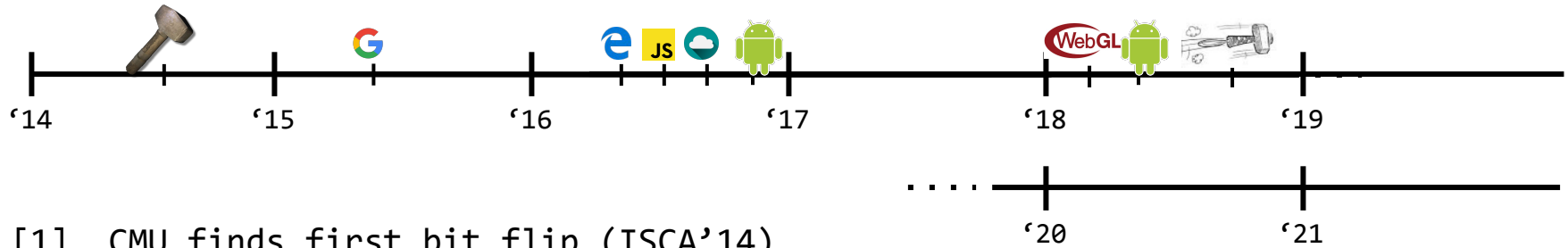


Rowhammer



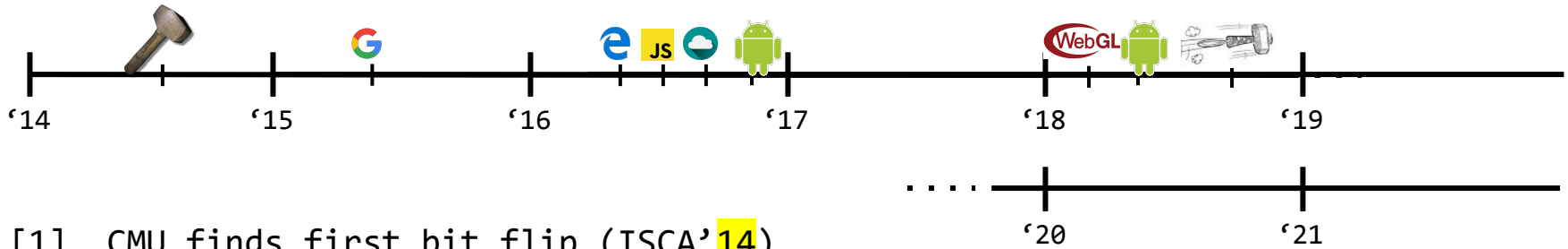
Don't know in advance which flips, but if it flips once, it will flip again

Rowhammer Evolution



- [1] CMU finds first bit flip (ISCA'14)
- [2] Google Project Zero : 1st Rowhammer root exploit (flipping PTEs)
- [3] Dedup Est Machina (S&P'16) : RH in MS Edge
- [4] Rowhammer.js (DIMVA'16) : 1st RH bit flip in JavaScript
- [5] Flip Feng Shui (USENIX SEC'16): co-located VMs in cloud
- [6] DRAMMER (CCS'16) : ARM / Smartphones
- [7] Grand Pwning Unit (S&P'18) : Attack from JavaScript on ARM via the GPU
- [8] HammerTime (RAID'18) : RH on many DIMMS + emulator/simulator
- [9] Rampage/GuardION (DIMVA'18) : DMA-based RH without ION kmalloc heap
- [10] Throwhammer (USENIX ATC'18) : RH across network

Rowhammer Evolution



- [1] CMU finds first bit flip (ISCA'14)
- [2] Google Project Zero : 1st Rowhammer root exploit (flipping PTEs)
- [3] Dedup Est Machina (S&P'16) : RH in MS Edge
- [4] Rowhammer.js (DIMVA'16) : 1st RH bit flip in JavaScript
- [5] Flip Feng Shui (USENIX SEC'16): co-located VMs in cloud
- [6] DRAMMER (CCS'16) : ARM / Smartphones
- [7] Grand Pwning Unit (S&P'18) : Attack from JavaScript on ARM via the GPU
- [8] HammerTime (RAID'18) : RH on many DIMMS + emulator/simulator
- [9] Rampage/GuardION (DIMVA'18) : DMA-based RH without ION kmalloc heap
- [10] Throwhammer (USENIX ATC'18) : RH across network

Memory integrity is **dead**



I am not worried

server memory is
much better!

ECC?

Challenge Accepted



Lucian Cojocar

I flip bits in
ECC memory!

I win awards!





I am still not
worried

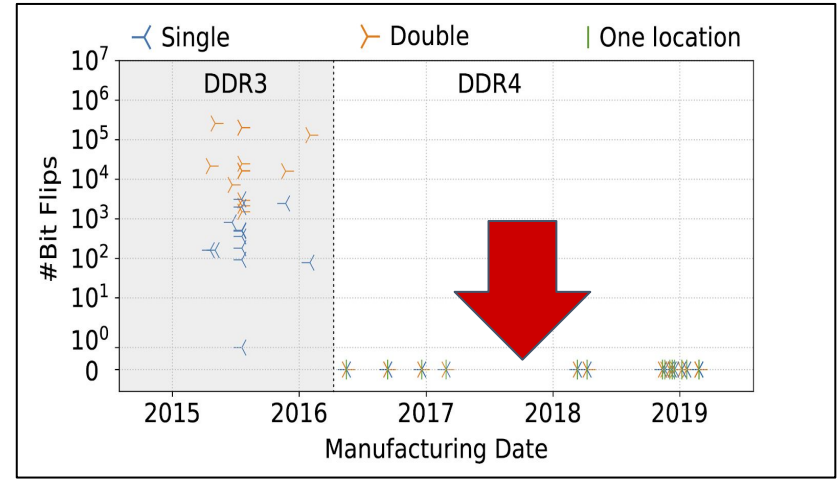
new memory is
much better still!

Target Row Refresh (TRR)

Track row activations: too many? → refresh victims
Many possible implementations → unknown

DDR4 that is Rowhammer-free





pTRR DDR3

Intel reports pTRR on DDR3 server systems

In-DRAM TRR

Earliest manufacturing date of RH-free DRAM modules

'12

'13

'14

'15

'16

'17

'18

'19

pTRR DDR4

First DDR4 generation is pTRR protected

DIMMs we focused on

Reverse engineering

Analysis from CPU not possible

FPGA-based memory controller

SoftMC: A Flexible and Practical Open-Source Infrastructure for Enabling Experimental DRAM Studies

Hasan Hassan^{1,2,3} Nandita Vijaykumar³ Samira Khan^{4,3} Saugata Ghose³ Kevin Chang³
Gennady Pekhimenko^{5,3} Donghyuk Lee^{6,3} Oguz Ergin² Our Mullu^{1,3}
¹ETH Zurich ²TORUS University of Economics & Technology ³Carnegie Mellon University
⁴University of Virginia ⁵Microsoft Research ⁶NVIDIA Research

DRAM is the primary technology used for main memory in modern systems. Unfortunately, as DRAM scales down to smaller technology nodes, it faces key challenges in both data integrity and latency, which strongly affects overall system reliability and performance. To develop reliable and high-performance DRAM-based main memory in future systems, it is critical to characterize, understand, and analyze various aspects (e.g., reliability, latency) of existing DRAM chips. To enable this, there is a strong need for a publicly-available DRAM testing infrastructure that can flexibly and efficiently test DRAM chips in a manner accessible to both software and hardware developers.

This paper develops the first such infrastructure, SoftMC (Soft Memory Controller), an FPGA-based testing platform that can control and test memory modules designed for the commonly-used DDR (Double Data Rate) interface. SoftMC has two key properties: (i) it provides flexibility to thoroughly control memory commands and (ii) it is easy to use as it provides a simple and high-level programming interface for users, completely hiding the low-level details of the FPGA.

We demonstrate the capability, flexibility, and programming ease of SoftMC with two example use cases. First, we implement a test that characterizes the retention time of DRAM cells. Experimental results we obtain using SoftMC are consistent with findings of prior studies on retention time in modern DRAM, which serves as a validation of our infrastructure. Second, we identify two recently-proposed mechanisms, which rely on accessing recently-refreshed or recently-accessed DRAM cells faster than other DRAM cells. Using our infrastructure, we show that the expected latency reduction effect of these mechanisms is not observable in existing DRAM chips, which demonstrates the sufficiency to characterize emerging non-volatile memory modules. We discuss several other use cases of SoftMC, including the ability to characterize emerging non-volatile memory modules that follow the DDR standard. We hope that our open-source infrastructure fills a gap in the space of publicly-available research and methodologies in memory system design.

1. Introduction

DRAM (Dynamic Random Access Memory) is the predominant technology used to build main memory systems of modern computers. The continuous scaling of DRAM process technology has enabled tremendous growth in DRAM process technology over the last few decades, leading to higher capacity main memories. Unfortunately, as the process technology node scales down to the sub-20 nm feature size range, DRAM technology faces key challenges that critically impact its reliability and performance. The fundamental challenge with scaling DRAM cells into smaller technology nodes arises from the way DRAM stores data in cells. A DRAM cell consists of a transistor and a capacitor that store its data permanently as the capacitor leaks its charge gradually over time. To maintain correct data in DRAM, each cell is periodically refreshed to replenish the charge in the capacitor. At smaller technology nodes, it is becoming increasingly difficult to store and retain enough charge in a cell, causing various reliability and performance issues [60, 61]. Ensuring reliable operation of the DRAM cells is a key challenge in future technology nodes [38, 45, 60, 61, 65, 68, 71].

The fundamental problem of retaining data with less charge in smaller cells directly impacts the reliability and performance of DRAM cells. First, smaller cells placed in close proximity make cells more susceptible to various types of interference. This potentially disrupts DRAM operation by flipping bits in DRAM, resulting in major reliability issues [46, 66, 74, 83, 84, 90], which can lead to system failure [66, 84] or security breaches [25, 46, 82, 85, 86, 95, 98]. Second, it takes longer time to access a cell with less charge [27, 58], and write latency increases as the access transistor size reduces [8]. Thus, smaller cells directly impact DRAM latency, as DRAM access time is determined by the worst-case (i.e., slowest) cell in the technology scaling in the past decade [6, 36, 57, 71], and in fact, some latencies are expected to increase [38], making memory latency an increasingly critical system performance bottleneck.

As such, there is a significant need for new mechanisms that improve the reliability and performance of DRAM-based main memory systems. In order to determine if DRAM-based mechanisms such as these are viable, it is important to accurately characterize, analyze, and understand DRAM access latency to be accurate, and it is important to accurately analyze behavior based on the experimental studies of real DRAM chips, since a large number of factors (e.g., inter- and intra-cell-to-cell interference [46, 74, 83], inter- and intra-process variation [17, 18, 36, 58], random errors [29, 61, 61, 62], stored data patterns [59, 67], internal organization [50, 61, 103], altered data patterns [43, 44, 61]) concurrently impact the reliability and latency of cells. Many of these phenomena and their interactions cannot be properly modeled (e.g., in simulation) or using analytical methods) without rigorous experiments for such experimental characterization and analysis. The need for such experimental characterization and analysis, in simulation or in hardware, is to understand the underlying mechanisms of reliability and performance of future DRAM-based main memory systems at various levels (both software and hardware) that can be used as a publicly-available DRAM testing infrastructure that can enable system users and designers to characterize real DRAM chips.

Two key features are desirable from such an experimental memory testing infrastructure. First, the infrastructure should be flexible enough to test any DRAM operation (supported by the commonly-used DRAM interfaces, e.g., the standard Double Data Rate, or DDR, interface) to characterize cell behavior and validate the impact of a mechanism (e.g., adopting different refresh rates for different cells [42, 44, 60, 79, 90]) on real DRAM chips. Second, the infrastructure should be easy to use, such

Hassan et al., HPCA'17

Reverse engineering

Analysis from CPU not possible

FPGA-based memory controller

Discovered 3 things

- sampler : keeps set of rows
- inhibitor : refreshes victims
- regret : about life choices

SoftMC: A Flexible and Practical Open-Source Infrastructure for Enabling Experimental DRAM Studies

Hasan Hassan^{1,2,3} Nandita Vijaykumar³ Samira Khan^{4,3} Saugata Ghose³ Kevin Chang³
Gennady Pekhimenko^{5,3} Donghyuk Lee^{6,3} Oguz Ergin² Onur Mutlu^{1,3}
¹ETH Zurich ²TORUS University of Economics & Technology ³Carnegie Mellon University
⁴University of Virginia ⁵Microsoft Research ⁶NVIDIA Research

DRAM is the primary technology used for main memory in modern systems. Unfortunately, as DRAM scales down to smaller technology nodes, it faces key challenges in both data integrity and latency, which strongly affects overall system reliability and performance. To develop reliable and high-performance DRAM-based main memory in future systems, it is critical to characterize, understand, and analyze various aspects (e.g., reliability, latency) of existing DRAM chips. To enable this, there is a strong need for a publicly-available DRAM testing infrastructure that can be used by both software and hardware developers. This paper develops the first such infrastructure, which can control and test memory modules designed for the commonly-used DDR (Double Data Rate) interface. SoftMC has two key properties: (i) it provides flexibility to thoroughly control memory commands, and (ii) it is easy to use as it provides a simple and intuitive high-level programming interface for users, completely hiding the low-level details of the FPGA.

We demonstrate the capability, flexibility, and programming ease of SoftMC with two example use cases. First, we implement a test that characterizes the retention time of DRAM cells. The findings of prior studies on retention time of DRAM cells, which serve as a validation of our infrastructure. Second, we identify two recently-proposed mechanisms, which rely on accessing recently-refreshed or recently-accessed DRAM cells faster than other DRAM cells. Using our infrastructure, we show that the expected latency reduction effect of these mechanisms is not obeyed in existing DRAM chips, which demonstrates the sufficiency to characterize emerging non-volatile memory modules. We discuss several other use cases of SoftMC, including the ability to characterize emerging non-volatile memory modules that do not follow the DDR standard. We hope that our open-source release of SoftMC fills a gap in the space of publicly-available research and methodologies in memory system design.

1. Introduction

DRAM (Dynamic Random Access Memory) is the predominant technology used to build main memory systems of modern computers. The continuous scaling of DRAM process technology has enabled tremendous growth in DRAM density in the last few decades, leading to higher capacity main memory in the sub-20 nm feature size range. DRAM technology faces challenges that critically impact its reliability and performance in the smaller technology nodes. A DRAM cell consists of a transistor and a capacitor. Data is stored as charge in the capacitor. A DRAM cell cannot retain its data permanently as the capacitor leaks its charge gradually over time. To maintain correct data in DRAM, each cell is periodically refreshed to replenish the charge in the capacitor. At smaller technology nodes, it is becoming increasingly difficult to store and retain enough charge in a cell, causing various reliability and performance issues [60, 61]. In future technology nodes DRAM cells is a key challenge.

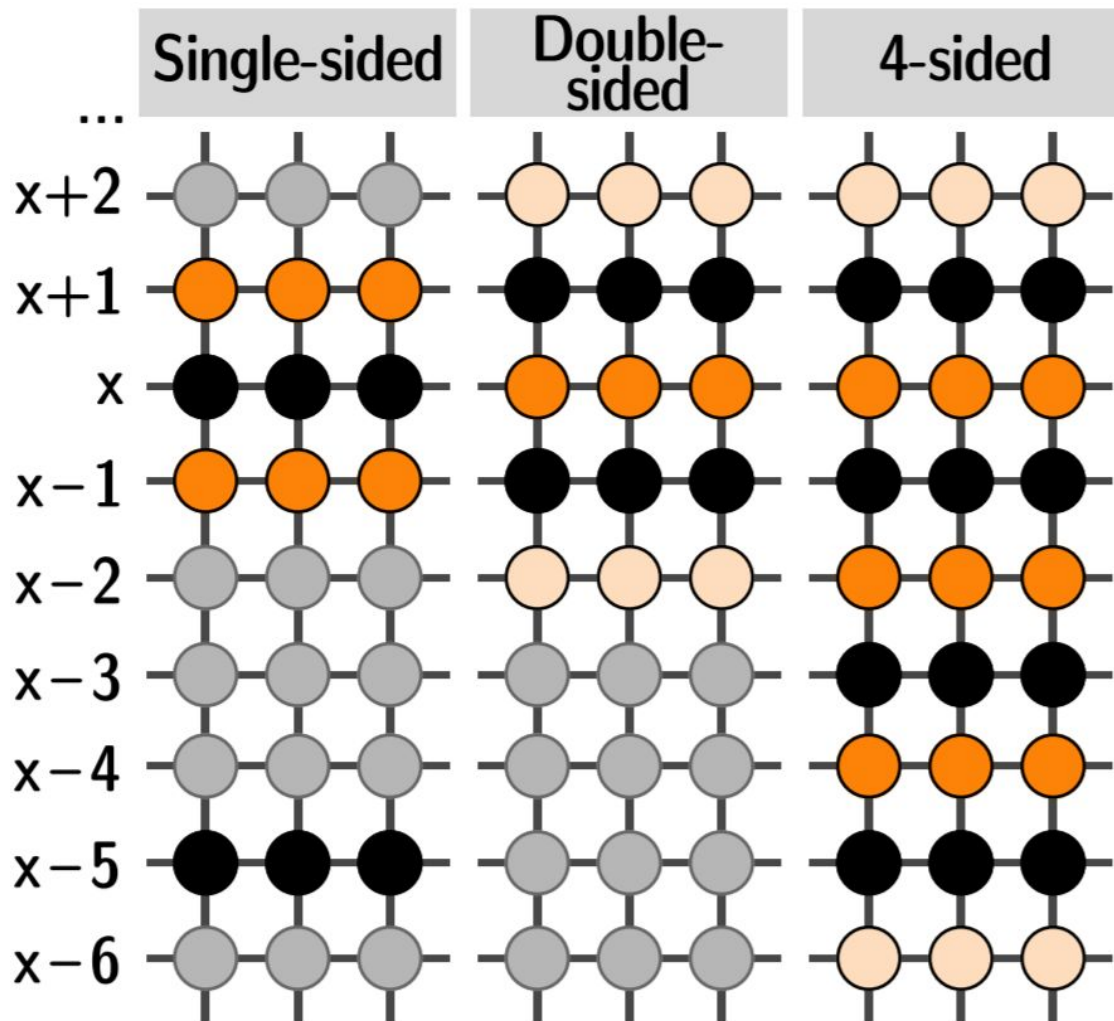
The fundamental problem of retaining data with less charge in smaller cells directly impacts the reliability and performance of DRAM cells. First, smaller cells placed in close proximity make cells more susceptible to various types of interference. This potentially disrupts DRAM operation by flipping bits in DRAM, resulting in major reliability issues [46, 66, 74, 83, 84, 90], which can lead to system failure [66, 84] or security breaches [25, 46, 82, 85, 86, 95, 98]. Second, it takes longer time to access a cell with less charge [27, 56], and write latency increases as the access transistor size reduces [8]. Thus, smaller cells directly impact DRAM latency, as DRAM access time is determined by the worst-case (i.e., slowest) cell in the technology scaling in the past decade [6, 36, 57, 71], and in fact, some latencies are expected to increase [38], making memory latency an increasingly critical system performance bottleneck. As such, there is a significant need for new mechanisms that improve the reliability and performance of DRAM-based main memory systems. In order to design and evaluate DRAM-based mechanisms, it is important to accurately and comprehensively analyze and understand DRAM access behavior to be accurate, and understand DRAM cell behavior. This analysis be based on the experimental studies of real DRAM chips, since a large number of factors (e.g., characterization variation [17, 18, 56, 58], inter- and intra-process and operating conditions [59, 61, 62], random effects [29, 61, 103], stored data patterns [43, 44, 61]) concurrently impact the reliability and latency of cells. Many of these phenomena and their interactions cannot be properly modeled (e.g., in simulation or using analytical methods) without rigorous experiments for such experimental and analysis of real DRAM chips. The first goal of building the understanding and analysis of the reliability and performance of future DRAM-based main memory is at various levels (both software and hardware) to improve the need for a publicly-available DRAM testing infrastructure that can ease the system users and designers to characterize real DRAM chips.

Two key features are desirable from such an experimental memory testing infrastructure. First, the infrastructure should be flexible enough to test any DRAM operation (supported by the Data Rate, or DDR, interfaces) e.g., the standard Double Data Rate (DDR) interfaces, e.g., the standard DDR2, DDR3, or refresh rates for different cells [42, 44, 60, 79, 90]) on real DRAM chips. Second, the infrastructure should be easy to use, such

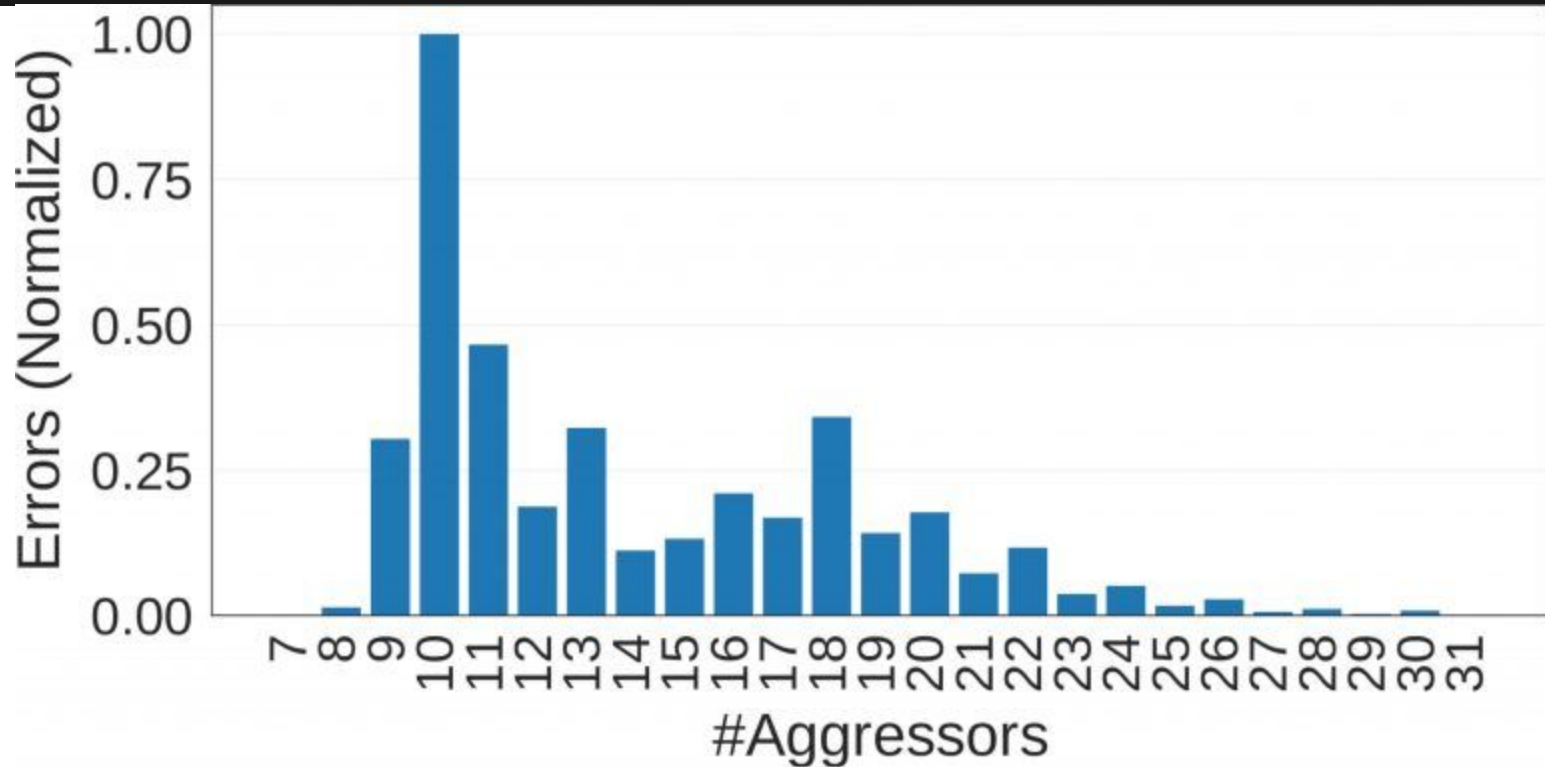
Hassan et al., HPCA'17

We need different patterns!

Need to trick TRR into tracking wrong rows!



Many-sided Rowhammer







Lawyers

Proxies

National Cyber Security Center

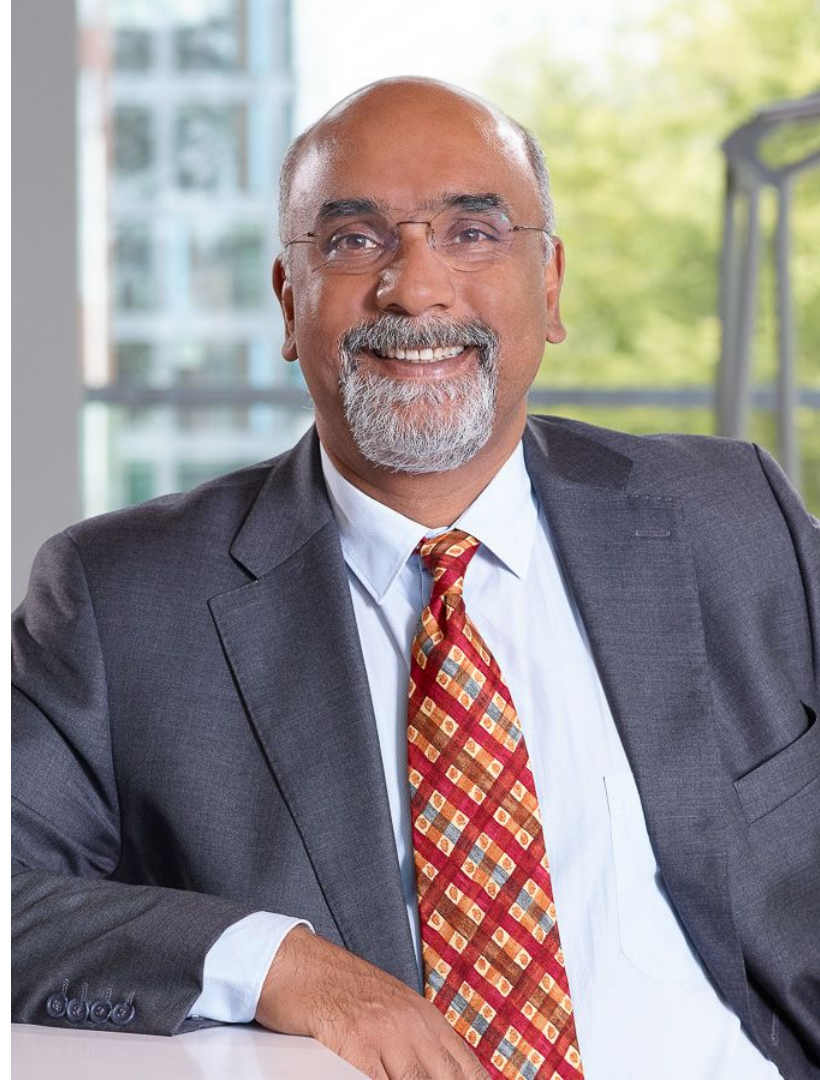
Anonymous email

Rector

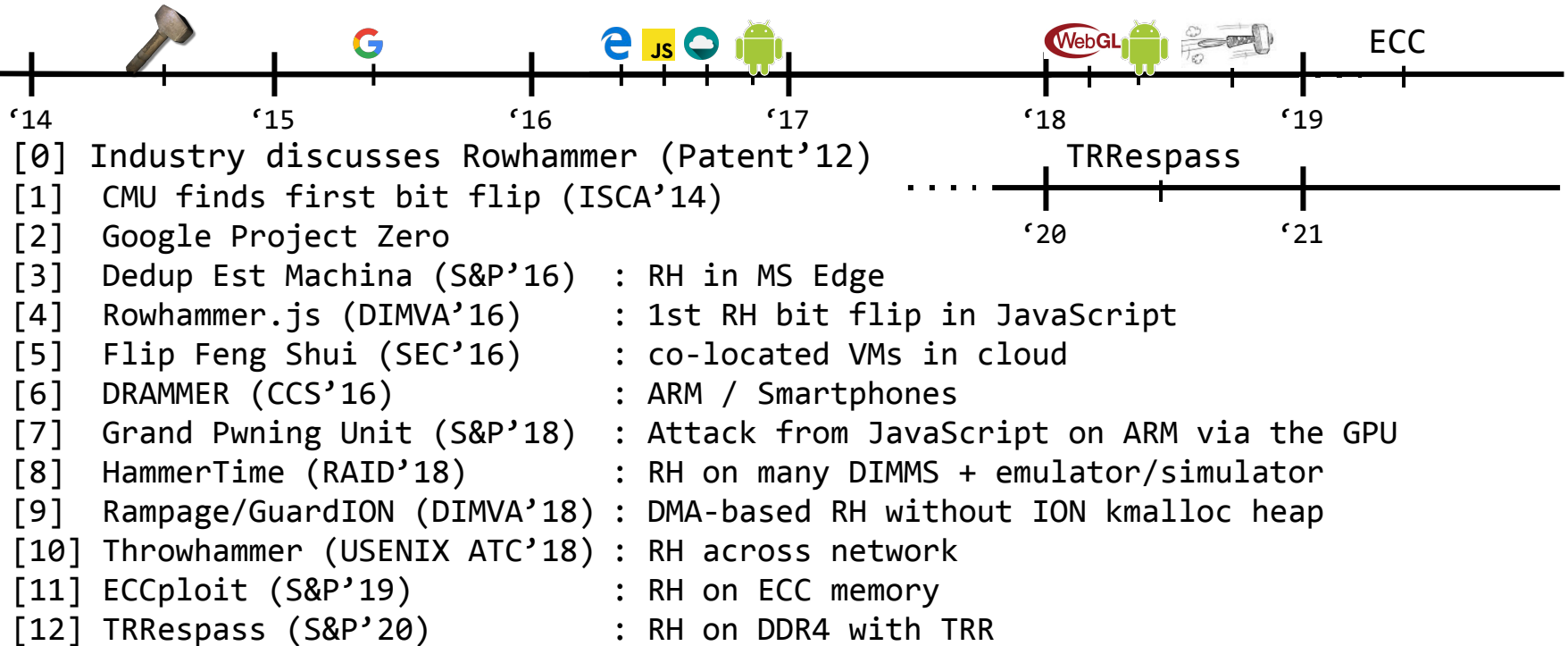
Vinod Subramaniam (rector)

Awesome

Read our paper. Twice.



Rowhammer Evolution



the
PWNIE
AWARDS

2020 PWNIE AWARDS

TRRespass!

Reverse Engineering



Best Paper
Award

IEEE Security &
Privacy 2020



TRRespass effective on **13** out of **42** tested DIMMs

The DDR4 substrate is much more vulnerable!

Bit flips with less activations compared to RH on DDR3 devices

TRR is not secure

TRR can track a limited number of rows at the same time.

Thus, victim rows are not refreshed



DDR4 devices are even more vulnerable than previous versions

All major vendors are affected

90% of the market

We can use fuzzing on DRAM!

After 10 years RH is still a problem

No immediate mitigation

