2024 54th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)

Simultaneous Many-Row Activation in Off-the-Shelf DRAM Chips: Experimental Characterization and Analysis

İsmail Emir Yüksel[1] Yahya Can Tuğrul[1,2] F. Nisa Bostancı[1] Geraldo F. Oliveira[1]
A. Giray Yağlıkçı[1] Ataberk Olgun[1] Melina Soysal[1] Haocong Luo[1]
Juan Gómez-Luna[1] Mohammad Sadrosadati[1] Onur Mutlu[1]
[1]*ETH Zürich* [2]*TOBB University of Economics and Technology*

We experimentally analyze the computational capability of commercial off-the-shelf (COTS) DRAM chips and the robustness of these capabilities under various timing delays between DRAM commands, data patterns, temperature, and voltage levels. We extensively characterize 120 COTS DDR4 chips from two major manufacturers. We highlight four key results of our study. First, COTS DRAM chips are capable of 1) simultaneously activating up to 32 rows (i.e., simultaneous many-row activation), 2) executing a majority of X (`MAJX`) operation where X>3 (i.e., `MAJ5`, `MAJ7`, and `MAJ9` operations), and 3) copying a DRAM row (concurrently) to up to 31 other DRAM rows, which we call `Multi-RowCopy`. Second, storing multiple copies of `MAJX`'s input operands on all simultaneously activated rows drastically increases the success rate (i.e., the percentage of DRAM cells that correctly *perform the computation) of the `MAJX` operation. For example, `MAJ3` with 32-row activation (i.e., replicating each `MAJ3`'s input operands 10 times) has a 30.81% higher average success rate than `MAJ3` with 4-row activation (i.e., no replication). Third, data pattern affects the success rate of `MAJX` and `Multi-RowCopy` operations by 11.52% and 0.07% on average. Fourth, simultaneous many-row activation, `MAJX`, and `Multi-RowCopy` operations are highly resilient to temperature and voltage changes, with small success rate variations of at most 2.13% among all tested operations. We believe these empirical results demonstrate the promising potential of using DRAM as a computation substrate. To aid future research and development, we open-source our infrastructure at `https://github.com/CMU-SAFARI/SiMRA-DRAM`.*

1. Introduction

Modern computing systems move vast amounts of data between main memory (DRAM) and processing elements (e.g., CPU, GPU, TPU, and FPGA) [1, 2]. Unfortunately, this data movement is a major bottleneck that consumes a large fraction of execution time and energy in many modern applications [1–28]. To address this problem, Processing-In-Memory (PIM) [13, 14, 24–26, 29–121] is a promising paradigm to alleviate data movement bottlenecks [64, 65, 68, 70, 76, 79, 80, 122, 123]. There are two main approaches to PIM [2, 11]: 1) Processing-Near-Memory (PNM) [13, 14, 24–26, 29–61, 99, 108], where computation logic is added near the memory arrays (e.g., in a DRAM chip or at the logic layer of a 3D-stacked memory [124–126]); and 2) Processing-Using-Memory (PUM) [56, 62–98, 100–107, 109, 127, 128], where computation is performed by leveraging the analog operational properties of the memory circuitry.

A subset of PIM proposals devise mechanisms that enable PUM using DRAM cells for computation, including data copy and initialization [67, 72, 77, 78, 89, 104, 127], Boolean logic [56, 64–66, 68, 70, 72, 76, 79, 122, 127–129], majority-based arithmetic [64, 66, 69, 72, 91, 127, 130, 131], and lookup table based operations [82, 106, 107, 132]. We refer to DRAM-based PUM as *Processing-Using-DRAM (PUD)* and the computation performed using DRAM cells as PUD operations.

PUD benefits from the bulk data parallelism in DRAM devices to perform bulk bitwise PUD operations. Prior works show that bulk bitwise operations are used in a wide variety of important applications, including databases and web search [64, 67, 79, 130, 133–140], data analytics [64, 141–144], graph processing [56, 80, 94, 130, 145], genome analysis [60, 99, 146–149], cryptography [150, 151], set operations [56, 64], and hyperdimensional computing [152–154].

Recent works [72, 86, 89, 129, 155] experimentally demonstrate that some of these PUD operations can be realized in commercial off-the-shelf (COTS) DRAM chips by operating beyond manufacturer-recommended DRAM timing parameters. To do so, these works [72, 86, 89, 129, 155] carefully engineer a sequence of DRAM commands that allows the DRAM chip to activate (i.e., open) multiple (e.g., 2 or 4) DRAM rows *simultaneously* depending on the DRAM row addresses, via a process that we refer to as *multiple row activation*. By performing multiple row activation, prior works can 1) copy data between two DRAM rows [72, 89], 2) perform three-input majority computation [72, 129, 155], and 3) generate true-random numbers [86, 89]. To investigate the effectiveness of PUD operations in COTS DRAM chips, such works perform extensive characterization of real DDR3 and DDR4 chips to identify the appropriate timing delays between DRAM commands that lead to PUD operations.

Even though prior works show that COTS DRAM chips can perform PUD operations, there are several questions about the *effectiveness* and *robustness* of PUD operations in COTS DRAM chips that should be answered to develop a rigorous understanding of the computational capabilities of modern DRAM chips, including:

Q1. In a DRAM subarray with many DRAM rows (512–1024), is it possible to *robustly* perform simultaneous activation of more than four DRAM rows (i.e., simultaneous many-row activation)?

Q2. What other PUD operations can be realized in COTS DRAM chips by leveraging simultaneous many-row activation?

Q3. How robustly can PUD operations using simultaneous many-row activation be performed in COTS DRAM chips?

Q4. Can the robustness of PUD operations be improved?

Q5. What are the effects of various DRAM operating conditions

2158-3927/24/$31.00 ©2024 IEEE
DOI 10.1109/DSN58291.2024.00024

(i.e., voltage and temperature scaling, data pattern dependence, and different timing delays between DRAM commands) on the robustness of PUD operations?

Our *goal* is to experimentally analyze the computational capability of COTS DRAM chips and the robustness of such capability under various operating conditions by answering the five key questions above. To this end, we conduct real DRAM chip experiments on 120 COTS DDR4 chips from two major DRAM manufacturers contained within 18 DRAM modules and back up our results with circuit-level simulations. Based on our real DRAM chip experiments, we make 18 new empirical observations and share 7 key takeaway lessons that provide answers to the five key questions above.

Answering Q1. COTS DRAM chips are capable of activating up to 32 DRAM rows, which we call *simultaneous many-row activation*. We observe that carefully crafted DRAM commands simultaneously activate 2, 4, 8, 16, and 32 rows (§4). We hypothesize that the hierarchical row decoder design, present in high-performance and high-density DRAM chips, is the primary reason behind the simultaneous many-row activation phenomenon we observe in COTS DRAM chips. Based on this hypothesis and the obtained mappings between the target row address and the observed activated DRAM rows, we derive a row decoder circuitry design that allows simultaneous many-row activation (§7.1).

Answering Q2. We observe that simultaneous many-row activation allows COTS DRAM chips to execute two PUD operations that prior works do *not* cover. COTS DRAM chips are capable of 1) executing a majority-of-X (`MAJX`) operation where X>3, i.e., `MAJ5`, `MAJ7`, and `MAJ9` (§5); 2) copying one source DRAM's row content concurrently into multiple (e.g., 31) destination DRAM rows (`Multi-RowCopy`) (§6).

Answering Q3. We observe that the success rate of a PUD operation (i.e., the percentage of DRAM cells that *reliably* and *correctly* perform PUD operation) *significantly* varies across PUD operations. Our analysis shows that 1) `MAJ3`, `MAJ5`, `MAJ7`, and `MAJ9` operations achieve 99.00%, 79.64%, 33.87%, and 5.91% average success rate, respectively (§5); and 2) copying one row's content to 1, 3, 7, 15, and 31 DRAM rows have an average success rate of 99.996%, 99.989%, 99.998%, 99.999%, and 99.982% (§6).

Answering Q4. We observe that *input replication* is a promising approach to improve the success rate of PUD operations. Storing multiple copies of `MAJX`'s input operands on all simultaneously activated rows drastically increases the success rate. For example, our results show that performing `MAJ3` with 32-row activation provides a 30.81% higher success rate than performing `MAJ3` with 4-row activation (§5 and §7.2).

Answering Q5. Data pattern stored in simultaneously activated rows affects the success rate of the `MAJX` and `Multi-RowCopy` operations. We observe that data pattern affects the success rate by 11.52% and 0.07% on average for `MAJX` operations and `Multi-RowCopy` operations. Temperature and voltage scaling has a small impact on the success rate of simultaneous many-row activation, `MAJX`, and `Multi-RowCopy` operations. Our results show that success rate changes by only 2.13% (1.32%) across all tested operations when the temperature (voltage) changes from 50 °C (2.5V) to 90 °C (2.1V).

By leveraging our 18 observations and 7 takeaways from extensive experiments on real DRAM chips, we demonstrate the performance benefits of supporting `MAJX` and `Multi-RowCopy` in COTS DRAM chips on 1) seven microbenchmarks and 2) a cold boot attack prevention mechanism.

This paper makes the following key contributions:

- We demonstrate, through an extensive experimental characterization of 120 modern DRAM chips from two major manufacturers, that modern DRAM chips can *robustly* activate up to 32 DRAM rows simultaneously.
- We demonstrate a proof-of-concept that COTS DRAM chips are capable of 1) executing `MAJ5`, `MAJ7`, `MAJ9`, and `Multi-RowCopy` operations and 2) the increasing success rate of `MAJX` by replicating the input operands of `MAJX`.
- We show the effect of DRAM operating parameters (i.e., timing delays between DRAM commands, data pattern, temperature, and voltage) on simultaneous many-row activation, `MAJX`, and `Multi-RowCopy` operations.

We believe that our findings can be used as a basis for building new and robust PUD mechanisms into DRAM chips and DRAM standards in the future. We hope that changes to the DRAM interface and DRAM chips, inspired by our proof-of-concept results, can enable PUD mechanisms with lower overhead and larger performance benefits than what we demonstrate. To aid future research and development, we open-source our infrastructure at `https://github.com/CMU-SAFARI/SiMRA-DRAM`.

2. Background

We briefly describe 1) DRAM organization, operation, timings, and 2) Processing-Using-DRAM (PUD) in commercial off-the-shelf (COTS) DRAM chips.

2.1. DRAM Organization, Operation and Timing

DRAM Organization. Fig. 1 shows the organization of DRAM-based memory systems. A *memory channel* connects the processor (CPU) to a *DRAM module*, where a module consists of multiple *DRAM ranks*. A rank is formed by a set of *DRAM chips* operated in lockstep. A DRAM chip has multiple *DRAM banks*, each composed of many *DRAM subarrays*. Within a subarray, DRAM cells form a two-dimensional structure interconnected over *bitlines* and *wordlines*. The row decoder in a subarray decodes the row address and drives one wordline out of many. A row of DRAM cells on the same wordline is referred to as a DRAM *row*. The DRAM cells in the same *column* are connected to the sense amplifier via a bitline. A DRAM cell stores the binary data value in the form of electrical charge on a capacitor (V_{DD} or 0 V) and this data is accessed through an access transistor, driven by the wordline to connect the cell capacitor to the bitline.

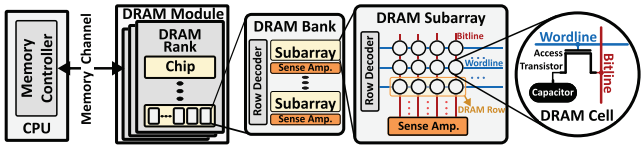


Figure 1: DRAM module, chip, bank, and subarray organization.

DRAM Operation. Data stored in a subarray is accessed at row granularity. To access a row, the memory controller issues

an ACT (ACTIVATE) command to assert the wordline and enable the sense amplifier. When the wordline is asserted, the cell capacitor connects to the bitline and shares its charge, causing a perturbation on the bitline voltage. After the sense amplifier is enabled, it senses and amplifies the voltage deviation on the bitline towards V_{DD} or 0 V. Once the data is fetched to the sense amplifiers, the memory controller may issue WR/RD commands to write to/read from the row. To access another row, the bank needs to be in the precharged state. To do so, the memory controller issues a PRE (PRECHARGE) command to disable the sense amplifiers, de-assert the wordline, and precharging the bitlines to $V_{DD}/2$. Once the bank is precharged, the memory controller can access another row.

DRAM Timing. To ensure correct operation, the memory controller must obey the DRAM timing parameters specified in the DRAM interface standards (e.g., DDR4) by the Joint Electron Device Engineering Council (JEDEC) [156]. We describe the most relevant timing constraints in the scope of this paper. The memory controller must wait for the latency of sensing the row's data and fully restoring a DRAM cell's charge (t_{RAS}) before issuing a PRE command after an ACT command. To open another row, the memory controller must wait for the latency of de-asserting a wordline and precharging the bitlines to $V_{DD}/2$ (t_{RP}) before issuing another ACT command. For more details on DRAM timing, we refer the reader to prior works [21, 157–159].

2.2. Processing Using COTS DRAM Chips

Simultaneous Many-Row Activation. Current DRAM standards do *not* officially support PUD operations. Yet, the design of COTS DRAM chips does *not* prevent users from activating multiple DRAM rows at once by issuing an ACT → PRE → ACT command sequence (called APA) with violated t_{RAS} and t_{RP} timing constraints [72, 86, 89, 129, 155, 160]. By doing so, two fundamental PUD operations can be performed in COTS DRAM chips: 1) in-DRAM majority-of-three (as in Ambit [64, 79]) and 2) in-DRAM row copy (as in RowClone [67]).

In-DRAM Majority-of-Three (MAJ3). Prior works [64, 79] introduces the concept of simultaneously activating three rows in a DRAM subarray (i.e., triple-row activation) through modifications to the DRAM circuitry. When three rows are concurrently activated, three cells connected to each bitline share charge simultaneously and contribute to the perturbation of the bitline [64, 79]. Upon sensing the perturbation of the three simultaneously activated rows, the sense amplifier amplifies the bitline voltage to V_{DD} or 0 V if at least two of the three DRAM cells are charged or discharged, respectively. As such, simultaneously activating three rows results in a Boolean majority-of-three operation (MAJ3). Prior works [72, 129, 155] demonstrate that COTS DRAM chips are capable of performing MAJ3 by simultaneously activating multiple rows in the same subarray. The state-of-the-art mechanism, FracDRAM [129], shows that a DRAM cell in COTS DDR3 chips can store fractional values (e.g., $V_{DD}/2$). FracDRAM uses fractional values to perform MAJ3 operations while simultaneously activating four DRAM rows in the same subarray.

In-DRAM Row Copy. RowClone [67] enables data movement within DRAM at row granularity without incurring the energy and execution time costs of transferring data between the DRAM and the computing units. An intra-subarray RowClone operation [67] works by issuing two back-to-back ACT commands: the first ACT copies the contents of the source row A into the sense amplifiers. The second ACT connects the DRAM cells in the destination row B to the bitlines. Because the sense amplifiers have already sensed and amplified the source data by the time row B is activated, the data in each cell of row B is overwritten by the data stored in the sense amplifiers (i.e., row A's data). Building on this observation, prior works [72, 89] experimentally demonstrate that COTS DRAM chips are capable of performing RowClone operation by enabling consecutive activation of two DRAM rows in the same subarray.

3. Methodology

We describe our methodology for two analyses. First, we experimentally characterize the computational capability of 120 commercial off-the-shelf (COTS) DRAM chips from two major manufacturers in terms of simultaneously activating many DRAM rows (§3.2), performing majority operations (§3.3), and copying one row's content (concurrently) to multiple rows, i.e., Multi-RowCopy (§3.4). Second, to back up our observations from real-device experiments, we conduct SPICE [161, 162] simulations (§3.5).

3.1. DRAM Testing Infrastructure

We conduct real DRAM chip experiments using DRAM Bender [155, 163], a DDR4 testing infrastructure that provides precise control of the DRAM commands issued to a module. Fig. 2 shows our experimental setup, which consists of six main components: ❶ a Xilinx Alveo U200 FPGA board [164] programmed with DRAM Bender, ❷ a host machine that generates DRAM commands used in our tests, ❸ rubber heaters that clamp the ❹ DRAM module on both sides to avoid fluctuations in ambient temperature, ❺ a MaxWell FT200 temperature controller [165] that keeps the DRAM chips at the target temperature, and ❻ an external TTi PL068-P power supply [166], which enables us to control DRAM wordline voltage (i.e., V_{PP}) at the precision of ±1 mV.[1]

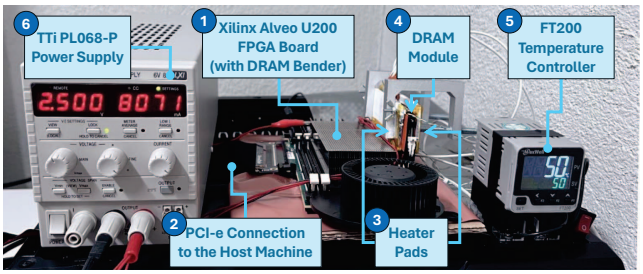


Figure 2: DDR4 DRAM Bender [155] experimental setup.

Real DDR4 DRAM Chips Tested. Table 1 shows the 120 (18) real DDR4 DRAM chips (modules) from two major DRAM manufacturers that we focus our analysis on. To demonstrate that our observations are not specific to a certain DRAM architecture/process but rather common across different designs and generations, we test a variety of DRAM chips spanning

[1] Modern DRAM chips use two separate voltage rails [156, 167–170]: 1) V_{DD}, used to operate the core DRAM array and peripheral circuitry, and 2) V_{PP}, used exclusively to assert a wordline during a DRAM row activation.

different die densities and die revisions.[2]

Table 1: Summary of DDR4 DRAM chips tested.

DRAM Mfr.	#Modules	#Chips	Die Rev.	Density	Org.	Subarray Size
SK Hynix	7	56	M	4Gb	x8	512 or 640
(Mfr. H)	5	40	A	4Gb	x8	512
Micron	4	16	E	16Gb	x16	1024
(Mfr. M)	2	8	B	16Gb	x16	1024

Metric. We define a metric to evaluate the robustness of PUD operations: *the success rate*. Success rate refers to the percentage of DRAM cells that produce correct output in *all* test trials of a PUD operation. Hence, we have two data points here. If a DRAM cell produces an incorrect result at least once, we refer to this DRAM cell as an *unstable cell* that *cannot* be used to perform PUD operations. For example, if an operation has a 25% success rate, it means that a quarter of the DRAM cells always produce correct results in the simultaneously activated rows and can be used to reliably perform that operation. We report the success rate distribution, which comes from all tested row groups in all DRAM chips.

t_{RAS} and t_{RP} Scaling. We test various reduced timing delays 1) between ACT and PRE commands, i.e., t_1, and 2) between PRE and ACT commands, i.e., t_2. All experiments are conducted at the timing delays that achieve the highest success rate for the tested PUD operations unless stated otherwise.

Temperature Scaling. We perform our experiments at five temperature levels: 50°C, 60°C, 70°C, 80°C, and 90°C. All experiments are conducted at 50°C unless stated otherwise.

V_{PP} Voltage Scaling. We test five V_{PP} levels: 2.5V, 2.4V, 2.3V, 2.2V, and 2.1V. All experiments are conducted at the nominal V_{PP} level (i.e., 2.5V) unless stated otherwise. In our analysis, we focus on the effects of V_{PP} on PUD computation. Even though both V_{PP} and V_{DD} can affect a DRAM chip's computation robustness, changing V_{DD} can negatively impact DRAM reliability in ways unrelated to computation inside a DRAM subarray (e.g., I/O circuitry instabilities) because V_{DD} supplies power to all logic elements within the DRAM chip. In contrast, V_{PP} affects only the wordline voltage. Therefore, we expect V_{PP} to influence computation inside a subarray, without adversely affecting DRAM chip components outside a subarray. V_{PP} also affects data retention [170] and access latency characteristics [172] of a DRAM chip. Such effects of V_{PP} are out of the scope of our analyses.

Data Patterns. We use two types of data patterns: random and fixed data patterns. For the random data pattern, we generate uniformly distributed random data and fill each activated row with the random data, where each activated row has a different data pattern. For fixed data patterns (i.e., 0x00/0xFF, 0xAA/0x55, 0xCC/0x33, 0x66/0x99), we fill each activated row either with 1) all 0x00 or all 0xFF, 2) all 0xAA or all 0x55, 3) all 0xCC or all 0x33, and 4) all 0x66 or all 0x99. All experiments are conducted using the random data pattern (where we observe the lowest success rate across tested data patterns) unless stated otherwise.

Number of Instances Tested. We randomly select three subarrays in each bank (a total of 16 banks) per DRAM module. Within each subarray, we randomly test 100 different groups of rows that are simultaneously activated each for 2-, 4-, 8-, 16-, and 32-row activation, which results in testing a total of 24K different groups of simultaneously activated rows per module.

Finding Subarray Boundaries. To understand the computational capability of PUD operations performed in a DRAM subarray, we follow the methodology used in prior works to identify subarray boundaries [72, 86, 89, 129, 155, 160]. We leverage the observation that it is possible to copy a row's data to another row (i.e., RowClone operation [67]) within the same subarray by leveraging the shared bitlines. We repeatedly perform RowClone across all row pairs. If we can copy a row's data to another row, we infer that these two rows are within a subarray. By conducting this experiment, we reverse engineer the subarray sizes (listed in Table 1) and subarray boundaries in all tested DRAM modules.

3.2. Simultaneous Many-Row Activation Experiments

In our experiments, we use the ACT R_F $\xrightarrow{t_1}$ PRE $\xrightarrow{t_2}$ ACT R_S (APA) DRAM command sequence, where R_F is the firstly activated row (i.e., Row_{First}), R_S is the secondly activated row (i.e., Row_{Second}), and t_1 and t_2 are timing delays between command pairs in the command sequence.

Testing Methodology. Our experiment consists of three steps. First, we initialize a whole subarray with a predefined data pattern. Second, we issue an APA command sequence to simultaneously activate multiple rows in the subarray. Third, we issue a WR command with a different data pattern from the predefined data pattern by respecting the timing parameters. This WR command causes the sense amplifiers to overdrive their bitlines and thus updates the values of the cells in all simultaneously activated DRAM rows [86, 173]. After the three-step procedure, we precharge the bank and read each row in the subarray while adhering to the nominal timing parameters. When we read all rows in the subarray, we expect that simultaneously activated rows store the exact data pattern sent with the WR command.

We test every possible R_F and R_S combination in the APA command sequence and various t_1 and t_2 timing parameters for each tested subarray. We observe that not all simultaneously activated rows fully store the exact data pattern (i.e., simultaneous many-row activation is not 100% robust.). Hence, we report the success rate of this simultaneous many-row activation as the percentage of simultaneously activated rows' cells that store the exact data pattern sent with the WR command.

3.3. Bitwise Majority Computation Experiments

Key Idea. Our key idea to perform an in-DRAM majority-of-X (MAJX) operation consists of four steps. First, we issue an ACT R_F command to assert the wordline of the R_F, connecting R_F to the bitline. Second, we issue a PRE command immediately after the first ACT while greatly violating t_{RAS}. This way, R_F does not have sufficient time to fully share its charge with the bitline. Third, we issue the second ACT command to another row by greatly violating t_{RP}. This prevents the DRAM control circuitry from de-asserting R_F and activates multiple rows. All activated rows share their charge with the bitline, which results in a perturbation in the bitline that corresponds to a majority operation across these rows. Finally, the sense amplifier amplifies the perturbation on the bitline. If the MAJX

[2]We also test Samsung chips; however, we observe no successful PUD operations. We hypothesize why we do not observe all tested PUD operations in Samsung chips in §9. We provide much more detail on all tested DRAM chips in the extended version of this paper [171].

is successful, the sense amplifier overwrites the bitline and all activated rows' content with the MAJX's result. For example, if we activate three cells at once, and they have A, B, and B values, after a successful MAJ3 operation, these three cells would have B as their value.

Replicating Inputs of MAJX. To perform MAJX operation with N-row activation (i.e., simultaneously activating N rows), we replicate each MAJX input operand (i.e., a total of X input operands) $\lfloor N/X \rfloor$ times.[3] For example, if we perform MAJ3 with 32-row activation, we replicate each input ten (i.e., $\lfloor 32/3 \rfloor = 10$) times. If the number of DRAM rows opened is not a multiple of X (i.e., $N\%X > 0$), we initialize $N\%X$ of the activated DRAM rows in a way that they do *not* contribute to bitline voltage, using a Frac [129] operation (§2.2).[4] We call these rows *neutral rows*. For example, if we perform MAJ3 with 32-row activation, we set the remaining two ($32\%3 = 2$) DRAM rows neutral.

Testing Methodology. We characterize the MAJX operation in five steps: 1) store MAJX's input operands (a total of X operands) in X rows, 2) replicate inputs of MAJX into remaining simultaneously activated rows 3) initialize neutral rows using the Frac operation, if it is needed, 4) perform the MAJX operation by issuing ACT $R_F \xrightarrow{t_1}$ PRE $\xrightarrow{t_2}$ ACT R_S command sequence, 5) wait for t_{RP}, and 6) read back the values in the row buffer.[5]

3.4. Multi-RowCopy Experiments

Key Idea. Copying one source DRAM row's content to multiple destinations DRAM rows at once (*Multi-RowCopy*) comprises of four steps in our real chip evaluation. First, we issue ACT to the source row (i.e., R_F) to assert the wordline and to connect R_F to the bitlines. Second, we issue PRE after waiting for t_{RAS}. This ensures the sense amplifier senses R_F correctly and drives bitlines to the source row's charge. Third, we issue the second ACT command by greatly violating t_{RP} (i.e., $\leq 3ns$).[6] The second ACT command interrupts the PRE command. By doing so, it 1) prevents the bitline from being precharged to $V_{DD}/2$, 2) keeps R_F and the sense amplifier enabled, and 3) simultaneously activates many rows. Finally, the sense amplifier overwrites all activated rows with the source row's data. If the Multi-RowCopy is successful, all the simultaneously activated rows with an APA command have the source row's data.

Testing Methodology. We characterize the Multi-RowCopy operation in four steps: we 1) initialize all destination rows with a predetermined data pattern, 2) initialize the source row with a different data pattern from the predetermined data pattern, 3) perform the Multi-RowCopy operation, 4) precharge the bank and individually read each destination row while adhering to the manufacturer-recommended DRAM timing parameters.

[3] In Boolean algebra, when MAJ inputs are replicated, MAJ maintains the functionality, e.g., MAJ6(A,B,C,A,B,C)=MAJ3(A,B,C).

[4] We use Frac parameters that achieve the best success rate for MAJX operation (i.e., the number of Frac operations, the timing delay, and the location of neutral rows). We refer the reader to FracDRAM [129] for more detail.

[5] For Mfr. M, Frac operation is not supported. However, we observe that the sense amplifiers of these modules are always biased to one or zero. Initializing the neutral rows with all zeros/ones enables a MAJX operation.

[6] Prior works [72, 89, 155] perform in-DRAM copy operation from one row to another using APA by waiting for more between PRE and ACT than Multi-RowCopy does (e.g., for 6ns). This results in consecutive activation of two rows.

3.5. SPICE Simulations

To provide insights into our real-chip-based experimental observations, we conduct SPICE [162] simulations to measure the bitline and cell voltage levels. We extend a DRAM model used in prior work [172] that allows us to perform multiple-row activation. We use LTspice [161] with the reference 55 nm DRAM model from Rambus [174] and scale it based on the ITRS roadmap [175, 176] to model the 22 nm technology node following the PTM transistor models [177].[7] To account for process variation, we perform Monte-Carlo simulations [178] over 10^4 iterations by randomly varying the capacitor and transistor parameters 10%, 20%, 30%, and 40% for each run.

4. Characterization of Simultaneous Many-Row Activation in COTS DRAM Chips

We provide two key analyses on simultaneous many-row activation in COTS DRAM chips. First, we characterize the robustness of simultaneous many-row activation by analyzing the effect of 1) timing delay between each command pair in the ACT $\rightarrow$ PRE $\rightarrow$ ACT (APA) sequence, 2) DRAM chip temperature, and 3) wordline voltage underscaling. Second, we analyze how the power consumption of simultaneous many-row activation measures against standard DRAM operations.

Effect of Timing Delay. Fig. 3 shows the distribution of the success rate for different numbers of simultaneously activated rows across all tested row groups in all DRAM chips in a box and whiskers plot for different combinations of t_1, the timing between ACT and PRE (rows of subplots) and t_2, the timing delay between PRE and ACT (columns of subplots).[8] We make Observations 1 and 2 from Fig. 3.

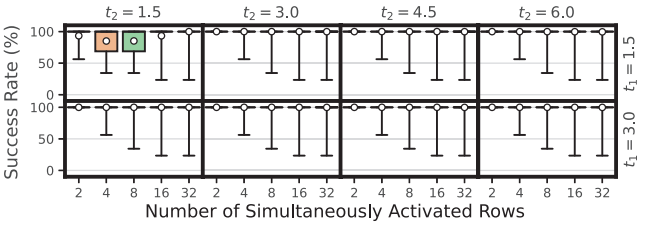


Figure 3: Effect of $t_1(ns)$ and $t_2(ns)$ on the success rate of simultaneous many-row activation.

Observation 1. COTS DRAM chips can simultaneously activate up to 32 rows with a >99.85% success rate.

When we issue an APA with the best timing delay (i.e., $t_1 = 3ns$, $t_2 = 3ns$), we can perform 2-, 4-, 8-, 16-, and 32-row activation with an average success rate of 99.99%, 99.99%, 99.99%, 99.99%, and 99.85%. We derive Takeaway 1 from Observation 1.

Takeaway 1. COTS DRAM chips are capable of simultaneously activating 2, 4, 8, 16, and 32 rows at very high success rates.

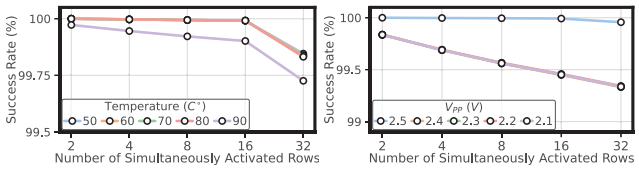
[7] We do *not* expect SPICE simulation and real-world experimental results to be identical because a circuit model cannot simulate a real DRAM chip's exact behavior without proprietary design and manufacturing information.

[8] In a box-and-whiskers plot, the box is lower-bounded by the first quartile and upper-bounded by the third quartile. The inter-quartile range (*IQR*) is the distance between the first and third quartiles (i.e., box size). Whiskers show the minimum and maximum values.

Observation 2. If t_1 or t_2 are lower than 3ns, we observe a drastic decrease in success rate.

For example, for 8-row activation, choosing t_1=t_2=1.5ns decreases the average success rate by 21.74% compared to the best timing delay configuration (i.e, t_1=1.5ns and t_2=3ns). We hypothesize that the row decoder circuitry [173] cannot fully activate all cells in the to-be-activated rows due to the very low delay (t_1+t_2=3ns) between two ACTs in APA command sequence.

Temperature Scaling. Fig. 4a shows the average success rate (y-axis) of simultaneously activating various numbers of rows (x-axis) under five temperature levels: 50 °C, 60 °C, 70 °C, 80 °C, and 90 °C. We make Observation 3 from Fig. 4a.



(a) (b)

Figure 4: Average success rate of simultaneous many-row activation with varying (a) temperature and (b) wordline voltage.

Observation 3. Increasing temperature up to 90 °C has a small effect on the success rate.

We observe only a 0.07% average success rate decrease on average on simultaneous many-row activation when the temperature is increased from 50 °C to 90 °C. We hypothesize that since simultaneous many-row activation experiments use I/O peripheral circuitry to perform WR commands, temperature could affect not only the subarray elements and their operation (e.g., DRAM cell, sense amplifiers, and charge-sharing) but also peripheral circuitry and their operation (e.g., write drivers/buffers and driving bitlines with the data pattern sent with WR command [173]). We believe that fully understanding the effect of temperature on the reliability of simultaneous many-row activation begs more investigation at the circuit level, which we leave for future work.

Voltage Scaling. Fig. 4b shows the average success rate (y-axis) of simultaneously activating various numbers of rows (x-axis) under five V_{PP} levels (in different line colors): 2.5V, 2.4V, 2.3V, 2.2V, and 2.1V.[9] We make Observation 4 from Fig. 4b.

Observation 4. Reducing the wordline voltage only slightly affects the success rate of simultaneous many-row activation.

Underscaling voltage from 2.5V to 2.1V decreases the average success rate by at most 0.41%.

We derive Takeaway 2 from Observations 3 and 4.

Takeaway 2. Simultaneous many-row activation is highly resilient to temperature and wordline voltage in COTS DRAM chips.

Power Consumption. We evaluate the power consumption of simultaneous many-row activation operations and standard DRAM operations (i.e., RD, WR, ACT+PRE, and REF) using one DRAM module.[10] Fig. 5 shows the average power consumption measured for each operation. Dashed lines represent standard DRAM operations. We make Observation 5 from Fig. 5.

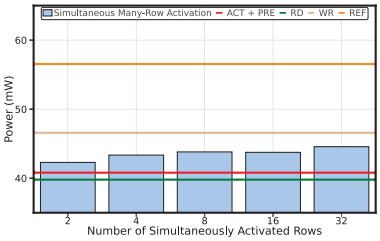


Figure 5: Power consumption of simultaneous many-row activation and standard DRAM operations.

Observation 5. Simultaneous many-row activation power draw likely meets the power budget of DDR4 chips.

The power consumption of simultaneously activating 32 rows is 21.19% smaller than the most power-consuming single DRAM operation's (i.e., REF) power consumption.

5. Characterization of Majority Operations in COTS DRAM Chips

We experimentally characterize the robustness of majority-of-X (MAJX) operations, where $X \in \{3,5,7,9\}$, under four operating parameters: 1) timing delay between each command pair in the APA sequence, 2) data pattern, 3) DRAM chip temperature, and 4) wordline voltage.

Effect of Timing Delay. Fig. 6 shows the distribution of the MAJ3's success rate for different numbers of simultaneously activated rows in a box and whiskers plot for different combinations of t_1 (i.e., the timing between ACT and PRE) and t_2 (i.e., the timing delay between PRE and ACT).[8] We replicate the MAJ3's input operands 1, 2, 5, and 10 times when performing the MAJ3 operation with 4-, 8-, 16-, and 32-row activation, respectively (§3.3). We make Observations 6 and 7 from Fig. 6.

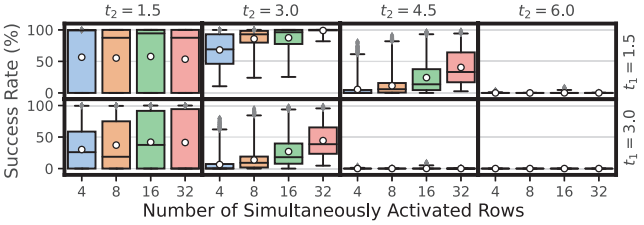


Figure 6: Effect of $t_1(ns)$, $t_2(ns)$, and the number of simultaneously activated rows on MAJ3 operation.

Observation 6. Storing multiple copies of MAJ3's input operands drastically increases the success rate of MAJ3.

MAJ3 with 32-row activation achieves 30.81% higher success rate than MAJ3 with 4-row activation. We hypothesize that MAJ3 with 4-row activation has a lower success rate than MAJ3 with 32-row activation because the deviation in a bitline's voltage is less likely to exceed the reliable sensing margin. For such a bitline, the sense amplifier fails to reliably produce correct MAJX results. Storing multiple copies of MAJ3's input operands could increase the deviation on the bitline voltage, which results in an increased success rate (the circuit-level simulations in §7.2 supports our observation and hypothesis).

[9] We test two DRAM modules due to time and infrastructure limitations.

[10] We provide a detailed description of how we conduct this experiment in the extended version of this paper [171]

Observation 7. Timing delays between commands in the APA command sequence heavily affects the success rate of MAJ3.

Choosing $t_1 = 1.5ns$ and $t_2 = 3ns$ to perform MAJ3 with 32-row activation achieves 99.00% average success rate, which is 45.50% higher than the second highest average success rate achieved by a different timing configuration (i.e., $t_1 = t_2 = 3ns$). We have two hypotheses to explain Observation 7. First, every activated row should contribute equally to the charge-sharing process to achieve the highest possible success rate in the MAJX operation. To achieve this, the timing delay between two ACT commands ($t_1 + t_2$) should be kept as low as possible since increasing this delay could cause the first activated row of the APA sequence (i.e., R_F) to share its charge more than others. Second, too small a delay between PRE and ACT may prevent (e.g., choosing $t_2 = 1.5ns$) the assertion of intermediate signals in the row decoder circuitry, leading to an inability to simultaneously activate multiple rows. Consequently, choosing $t_1 = 1.5ns$ and $t_2 = 3ns$ achieves the highest success rate.

Data Pattern. We analyze the effect of data pattern on the success rate for MAJ3, MAJ5, MAJ7, and MAJ9 operations[11] as we vary the number of simultaneously activated rows. Fig. 7 shows the success rate distribution of MAJ3, MAJ5, MAJ7, and MAJ9 operations across DRAM cells for five tested data patterns. We make Observations 8-10 from Fig. 7.

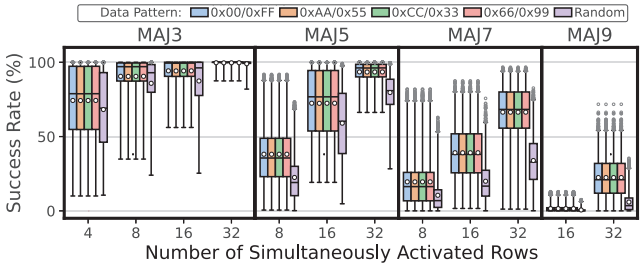


Figure 7: Success rates of MAJ3, MAJ5, MAJ7, and MAJ9 operations with different data patterns.

Observation 8. We can perform MAJ5, MAJ7, and MAJ9 operations in COTS DRAM chips.

COTS DRAM chips are capable of performing MAJ5, MAJ7, and MAJ9 operations. We observe that performing MAJ5, MAJ7, and MAJ9 operations with 32-row activation achieves average success rates (across tested groups of rows that are simultaneously activated) of 79.64%, 33.87%, and 5.91%, respectively.

We derive Takeaway 3 from Observation 8.

Takeaway 3. COTS DRAM chips are capable of performing MAJ5, MAJ7, and MAJ9 operations.

Observation 9. Data pattern significantly affects the success rate of the MAJX operation.

Fixed data patterns (i.e., 0x00/0xFF, 0xAA/0x55, 0xCC/0x33, and 0x66/0x99 data patterns) have a small and similar effect on the success rate of the MAJX operation, whereas uniformly distributed random data pattern significantly decreases the success rate of the MAJX operation. For example, when using 32-row activation, MAJ3, MAJ5, MAJ7, and MAJ9 operations with random data pattern have 0.68%, 13.85%, 32.56%, and 16.51% lower average success rates than MAJ3, MAJ5, MAJ7, and MAJ9 operations with all 0x00/0xFF data pattern, respectively.

Observation 10. Storing multiple copies of MAJX's input operands increases the success rate of not only MAJ3 but also MAJ5, MAJ7, and MAJ9 operations.

For example, storing multiple copies increases the average success rate of MAJ5, MAJ7, and MAJ9 with random data pattern by 56.27%, 35.15%, and 13.11%.

We derive Takeaway 4 from Observations 6 and 10.

Takeaway 4. Storing multiple copies of MAJX's input operands significantly increases the success rate of the MAJX operation in COTS DRAM chips.

Temperature Scaling. Fig. 8 shows the success rate distribution of the MAJX operation with various simultaneously activated rows at five different temperature levels: 50 °C, 60 °C, 70 °C, 80 °C, and 90 °C. We make Observations 11 and 12 from Fig. 8.

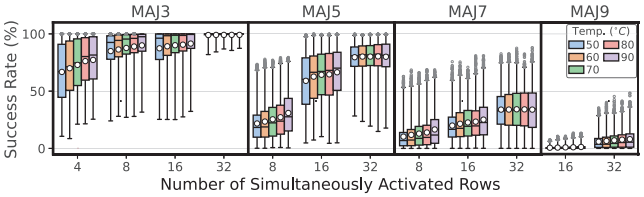


Figure 8: Success rate of MAJX at different DRAM chip temperatures.

Observation 11. Temperature slightly affects the success rate of the MAJX operation.

For example, from 50 °C to 90 °C, the average success rate varies by 4.25% on average across all the tested operations. We hypothesize that increasing the temperature could reduce the threshold voltages of cells' access transistors, and thus, the charge-sharing operation among activated rows and bitlines happens faster and stronger [179, 180]. Consequently, the MAJX operation exhibit higher success rates at higher temperatures.

Observation 12. Storing multiple copies of MAJX's input operands lowers the effect of temperature on success rate.

For example, performing MAJ3 with 32-row activation has at most 1.65% success rate variations, whereas MAJ3 with 4-row activation has at most 15.20%.

Voltage Scaling. Fig. 9 shows the success rate distribution of the MAJX operation at five different wordline voltage (V_{PP}) levels: 2.5V, 2.4V, 2.3V, 2.2V, and 2.1V.[9] We make Observation 13 from Fig. 9.

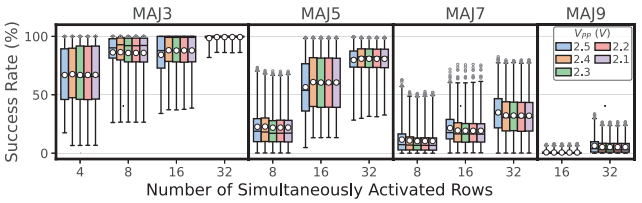


Figure 9: Effect of wordline voltage on the success rate of the MAJX operation.

[11] We omit MAJX operations that have <1% success rate at most (i.e., MAJ11+ for Mfr. H, and MAJ9+ for Mfr. M).

Observation 13. Wordline voltage slightly affects the success rate of the MAJX operation.

Success rate of the MAJX operation has a 1.10% variation on average across all the tested operations.

We derive Takeaway 5 from Observations 9, 11, and 13.

Takeaway 5. Wordline voltage and temperature only slightly affect the success rate of the MAJX operation, whereas data pattern has a significant effect on success rate.

6. Characterizing Multi-Row Initialization

We experimentally characterize the Multi-RowCopy operation where one source row's content can concurrently be copied to up to 31 destination rows. We evaluate the robustness of the Multi-RowCopy operation under four key parameters: 1) timing delay between each command pair in the APA sequence, 2) data pattern, 3) chip temperature, and 4) wordline voltage.

Effect of Timing Delay. Fig. 10 shows the distribution of the success rate for different numbers of destination rows in a box and whiskers plot[8] for different combinations of t_1, the timing between ACT and PRE and t_2, the timing delay between PRE and ACT. We make Observations 14 and 15 from Fig. 10.

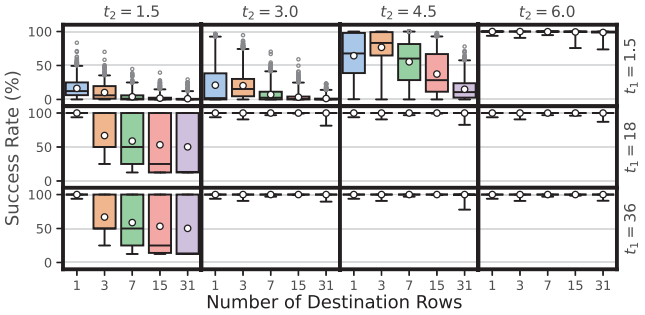


Figure 10: Effect of $t_1(ns)$, $t_2(ns)$ and the number of simultaneously activated rows on the success rate of the Multi-RowCopy operation.

Observation 14. COTS DRAM chips can copy one row's content to up to 31 rows with a >99.98% success rate.

When we use the timing delays that achieve the highest success rate (i.e., t_1=36ns and t_2=3ns), copying one source row's content to 1, 3, 7, 15, and 31 destination rows has an average success rate of 99.996%, 99.989%, 99.998%, 99.999%, and 99.982%. We hypothesize that waiting for the t_{RAS} timing parameter (i.e., t_1=36ns) before issuing a PRE in the APA command sequence ensures that the sense amplifier senses the source row correctly and drives the bitlines to the source row's charge level.

Observation 15. Low t_1 (i.e., $t_1 = 1.5ns$) has a significantly lower success rate than other timing configurations.

When we choose t_1 as 1.5ns, Multi-RowCopy achieves 49.79% lower success rate than the second worst timing configuration. We hypothesize that decreasing t_1 (e.g., to $t_1 = 1.5ns$) could cause the sense amplifiers to not fully drive the bitlines with the source row's charge, which results in low success rates.

Takeaway 6. COTS DRAM chips are capable of copying one row's data to 1, 3, 7, 15, and 31 other rows at very high success rates.

Data Pattern. Fig. 11 shows the average success rate of Multi-RowCopy (y-axis) for different numbers of destination rows (x-axis) with three data patterns (in different line colors). We make Observation 16 from Fig. 11.

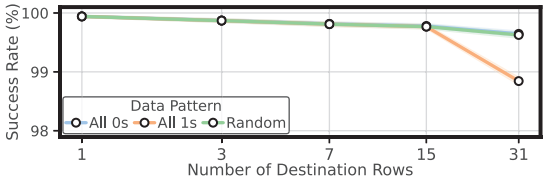
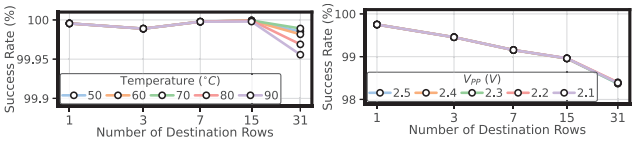


Figure 11: Data pattern dependence of Multi-RowCopy.

Observation 16. Copying all-1s to 31 rows has a slightly lower success rate than copying other data patterns.

Although copying up to 15 rows has a small (i.e., at most 0.11%) success rate difference among tested data patterns, when we copy all-1s to 31 rows, we observe a 0.79% decrease in success rate compared to all-0 and random data patterns.

Temperature Scaling. Fig. 12a shows the average success rate of Multi-RowCopy (y-axis) for different numbers of destination rows (x-axis) at five temperature levels (in different line colors): 50 °C, 60 °C, 70 °C, 80 °C, and 90 °C. We make Observation 17 from Fig. 12a.



(a) (b)

Figure 12: Average success rate of Multi-RowCopy operations with varying (a) temperature and (b) wordline voltage scaling.

Observation 17. Increasing temperature up to 90 °C has a very small effect on the success rate of all the tested Multi-RowCopy operations.

Increasing the DRAM chip temperature from 50 °C to 90 °C has 0.04% success rate variation on average of all the tested Multi-RowCopy operations.

Voltage Scaling. Fig. 12b shows the average success rate of Multi-RowCopy (y-axis) for different numbers of destination rows (x-axis) at five voltage levels (in different line colors): 2.5V, 2.4V, 2.3V, 2.2V, and 2.1V.[9] We make Observation 18 from Fig. 12b.

Observation 18. Reducing the wordline voltage has a small impact on the success rate.

Underscaling the V_{PP} by 0.4V decreases the success rate by 1.32% at most across all the tested Multi-RowCopy operations.

We derive Takeaway 7 from Observations 16-18.

Takeaway 7. Multi-RowCopy in COTS DRAM chips is highly resilient to changes in data pattern, temperature, and wordline voltage.

7. PUD Operations in COTS DRAM Chips: Hypothesis & Underlying Mechanisms

We have experimentally demonstrated that COTS DRAM chips are capable of performing PUD operations by violating DRAM

timing parameters and leveraging simultaneous many-row activation. To fundamentally understand the reasons of this capability, one would have to obtain access to the micro architectural design of the DRAM modules we test, which, unfortunately, is *not* publicly available. However, based on the known literature on modern DRAM chips [72, 86, 129, 155, 173, 181–183], open-sourced SPICE models of DRAM arrays [159, 184, 185], and our observations (§4-6), we draw two hypotheses to shine light on our experimental observations. First, we hypothesize that simultaneous many-row activation is possible in real DRAM chips due to the *hierarchical DRAM row decoder* modern DRAM devices employ (§7.1). Second, we hypothesize that input replication improves the success rate of `MAJX` operation by increasing bitline voltage perturbation toward a margin that the DRAM sense amplifier can produce correct `MAJX` result (§7.2).

7.1. Hypothetical Row Decoder Design

We explain how a hierarchical DRAM row decoder circuitry [181], which is used to construct high-performance and high-density DRAM chips, could allow for simultaneous activation of many DRAM rows. The row decoder circuitry in a DRAM bank decodes the n-bit row address (RA) and asserts a wordline out of 2^n wordlines. Modern DRAM chips have multiple tiers of decoding stages to reduce latency, area, and power consumption [181–183]. We analyze the row decoder circuitry of a COTS DRAM chip [186], which has 2^{16} rows in a bank. We observe that in this chip, each subarray consists of 2^9 rows and the number of subarrays in a bank is 2^7. We present a hypothesis regarding the row decoder circuitry that allows simultaneous many-row activation and the sequence of operations that occur in the row decoder when consecutive `ACT` and `PRE` commands are issued.

Row Address Indexing. We observe that the lower-order 9 bits of the RA are used to index a row in a subarray, while the higher-order 7 bits are used to index a subarray in a bank. To do so, we carefully reuse the DRAM row adjacency reverse engineering methodology described in prior works [160, 187].

Detailed Row Decoder Design. Fig. 13 illustrates a hypothetical row decoder circuitry in a bank that consists of two components: 1) Global Wordline Decoder (GWLD) (❶) and 2) Local Wordline Decoder (LWLD) (❷). When an `ACT` command is issued, three operations occur in the following order. First, GWLD decodes the higher-order 7 bits of the RA (RA[9:15]) and drives the Global Wordline (*GWL*) that enables the LWLD of a subarray (e.g., GWL0 enables LWLD of SA0 in Fig. 13). Second, Stage 1 of LWLD predecodes the lower-order 9 bits of the RA (RA[0:8]) in five tiers of predecoders (Predecoder A, B, C, D, and E; ❸) and latches the predecoded address bits ($P_{A0}, P_{A1}, ..., P_{E3}$). Third, Stage 2 of LWLD decodes the predecoded *P* signals to assert the corresponding Local Wordline (*LWL*) in Stage 2 (❹). When a `PRE` command is issued with standard DRAM timing parameters, the latched predecoded *P* signals are correctly de-asserted the corresponding *LWL*.

Activating Multiple Rows: A Walk-Through. Reducing the latency between `PRE` and the second `ACT` commands (i.e., t_{RP}) allows the predecoders to latch the next RA without deasserting the RA targeted by the first `ACT` command. Hence, after the second `ACT` command, depending on the target addresses of `APA` sequence, multiple latches of each predecoder in LWLD

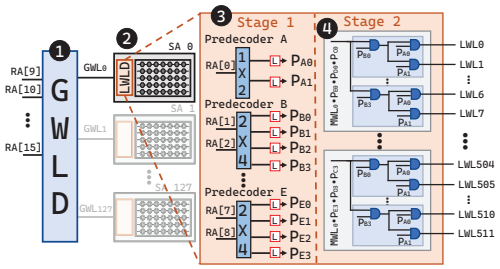


Figure 13: Hypothetical row decoder design.

can be set. By changing the row addresses targeted by two `ACT` commands, we can control the number and addresses of the simultaneously activated rows in a subarray.

Fig. 14 demonstrates an example of simultaneously activating four rows in the same subarray when the `ACT` 0 → `PRE` → `ACT` 7 is issued with violated timings.The memory controller issues each command (shown in orange boxes below time axis) at the corresponding tick mark, and asserted signals are highlighted in red. The bank is initially precharged and no signals in the row decoder circuitry are asserted (❶).

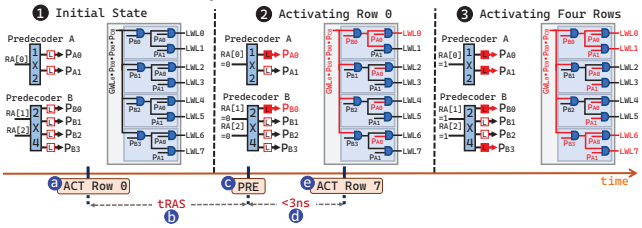


Figure 14: Example of activating multiple rows in hypothetical row decoder design. Red represents asserted signals.

We simultaneously activate four rows in X steps. First, we issue an `ACT` command to `Row 0` (ⓐ), and wait for the manufacturer-recommended t_{RAS} (ⓑ). This results in the predecoders asserting P_{A0} and P_{B0} signals, which drive LWL_0 and activate `Row 0` (❷). Second, we issue a `PRE` command (ⓒ) with violated t_{RP} timing, e.g., <3ns (ⓓ), and we issue another `ACT` command to `Row 7` (ⓔ). Issuing back-to-back `PRE`→`ACT` asserts P_{A1} and P_{B3} signals without de-asserting P_{A0} and P_{B0} (❸). As a result, P_{A0}, P_{A1}, P_{B0}, and P_{B3} signals are set simultaneously, and thus the decoder tree asserts LWL_0, LWL_1, LWL_6, and LWL_7 wordlines, thereby simultaneously activating rows 0, 1, 6, and 7.

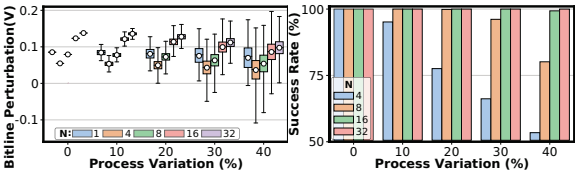
We formulate our observation as follows: to activate 2^N rows, N different predecoders have to latch two predecoded P signals. For instance, as illustrated in Fig. 14, to activate 4 rows, we issue `APA` command sequence that targets the rows that only latch the two different predecoders' (i.e., Predecoders A and B) two outputs (i.e., $\{P_{A0}, P_{A1}\}$ and $\{P_{B0}, P_{B3}\}$). Hence, to activate 32 rows, an `APA` sequence needs to target such rows that make all predecoders latch two outputs (e.g., `ACT` 127 → `PRE` → `ACT` 128). We hypothesize that the upper bound for the number of rows that are simultaneously activated depends on the number of predecoders. The examined module likely has five predecoders, and thus, we can activate up to 2^5 rows.

7.2. Increasing Success Rate by Leveraging Input Replication

Current DRAM standards do not officially support the `MAJX` operation. Yet, by leveraging the fundamental design and opera-

tional principles of COTS DRAM chips, it is possible to perform MAJX by issuing APA with violated timing parameters: t_{RAS} and t_{RP} (§5). Doing so can reduce the bitline voltage perturbation compared to a standard DRAM activation operation (i.e., single row activation), as multiple cells are simultaneously connected to the same bitline. As a result, the reduced bitline voltage perturbation is less likely to exceed reliable sensing margin, and the sense amplifier fails to reliably produce correct MAJX results. We hypothesize that by storing multiple copies of MAJX input operands on all simultaneously activated rows (which we call input replication), the bitline voltage can be increased and perturbed towards a safer margin and, thus, potentially increase the success rate of MAJX operations.

To test our hypothesis, we conduct SPICE simulations and analyze the effect of input replication on the success rate of one example MAJX operation, MAJ3(1,1,0). Fig. 15 shows the effect of process variation on the sensing operation for MAJ3(1,1,0) with N-row activation, where $N \in \{1, 4, 8, 16, 32\}$. Fig. 15a depicts the bitline perturbation distribution right before the sensing operation (y-axis) across 1000 different sets of N DRAM cell(s) (e.g., for 4-row activation, we test 1000 different sets of four cells) for different process variation percentages (x-axis). Each $N = 1$ box represents the bitline bitline perturbation distribution for a single-row activation. Boxes for other N values show the bitline perturbation distribution for 4-, 8-, 16-, and 32-row activation. Fig. 15b shows the success rate of MAJ3(1,1,0) with N-row activation, where $N \in \{4, 8, 16, 32\}$.



(a) (b)

Figure 15: (a) Effect of input replication on the bitline deviation (b) and the success rate of MAJ3 for N-row activation across different process variations using SPICE simulations

We make three key observations based on Fig. 15. First, increasing the number of simultaneously activated rows increases the bitline perturbation in every process variation percentage. On average, performing MAJ3 with 32-row activation (i.e., ten copies for each input operand) has 159.05% higher bitline voltage perturbation than performing MAJ3 with 4-row activation on average. Second, activating more than eight rows always results in a higher bitline perturbation than single-row activation on average for every process variation percentage. Third, input replication results in a higher success rate under all process variation percentages. The success rate of MAJ3 with 4-row activation reduces by 46.58% when process variation increases from 0% to 40%. In contrast, the success rate of MAJ3 with 32-row activation reduces only by 0.01%. We conclude that higher success rates observed with replicated inputs in COTS DRAM chips (§5) due to bitline voltage is perturbed towards a more reliable sensing margin than no replication.

8. Case Studies

We study the potential performance benefits of enabling the new PUD operations (i.e., MAJ5, MAJ7, MAJ9, and Multi-RowCopy) we demonstrated in COTS DRAM chips. We analyze the potential performance gain using new PUD operations in 1) seven majority-based arithmetic & logic operations and 2) content destruction operations for cold-boot attack prevention.

8.1. Majority-Based Computation

Majority-Based Boolean and Arithmetic Operations. Majority operations can be used to implement 1) logic operations such as AND/OR [64, 65, 72, 76, 79, 90, 130]) and XOR [188], and 2) full addition [72, 90, 127, 130]. These operations can be used as basic building blocks for more complex PUD computation (e.g., multiplication) [72, 80, 90, 131].

Experimental Methodology. We evaluate the execution time of seven arithmetic & logic microbenchmarks implemented using MAJX operations (MAJ3, MAJ5, and MAJ7 for Mfr. M and MAJ3, MAJ5, MAJ7, and MAJ9 for Mfr. H). We perform 32-bit logic (AND, OR, and XOR) and arithmetic (addition, subtraction, multiplication, and division) computations on 8KB elements.

We use DRAM Bender [155] to tightly schedule the DRAM commands to perform MAJX, Multi-RowCopy, and RowClone operations and measure their latency. To measure the latency of MAJX operations with N-row activation, we perform RowClone to copy the MAJX inputs into X number of rows and replicate the input operations into N rows using Multi-RowCopy operations. We use the empirical success rates (obtained from DRAM chips) and calculate the throughput for each MAJX operation. We then choose the group of rows that are simultaneously activated, which produces the highest throughput across all 18 tested DRAM modules for each MAJX operation. We analytically model the execution time of the arithmetic and logic microbenchmarks using the highest throughput values. For the baseline, we use the highest throughput of MAJ3 with 4-row activation and RowClone operations, which is the state-of-the-art for PUD operations.

Performance Evaluation. Fig. 16 shows the execution time speedup of seven microbenchmarks that use MAJX operations in DRAM chips from two manufacturers normalized to the baseline (i.e., MAJ3 with 4-row activation), the blue dashed line. We make three key observations. First, new MAJX operations (i.e., MAJ5, MAJ7, and MAJ9) improve performance over MAJ3 in all microbenchmarks. On average, new MAJX operations provide 121.61% (46.54%) higher performance over using only MAJ3 in Mfr. M (Mfr. H). Second, increasing the number of input operands in MAJX improves performance. MAJ7 provides 62.10% (31.71%) higher performance than MAJ5 in Mfr. M (Mfr. H). Third, in Mfr. H, MAJ9 leads to performance degradation of 114.12%. This is because MAJ9 has a poor success rate (shown in Fig. 7), which requires repeatedly performing the MAJ9, resulting in higher execution time. We conclude that enabling new MAJX operations has great potential to improve the performance of majority-based computation.

Majority-based Error Correction Operations. Majority operations could be useful for many systems that experience errors frequently, such as systems in space environments [189–191]. One common reliability technique for mitigating these errors is triple modular redundancy (TMR) [189, 192]. TMR stores three copies of the original data and performs majority voting to decide the correct output. A single bit error in TMR does

Figure 16: Speedup of using `MAJ5`, `MAJ7`, and `MAJ9` over the state-of-the-art (`MAJ3`) in seven arithmetic & logic microbenchmarks.

not produce any error as the majority operation would calculate the correct result based on the other two correct copies of data [189, 192].

`MAJ3` operations can be leveraged to perform majority voting and thus could be useful for such systems. Since we observe that off-the-shelf DRAM chips can perform up to seven input majority operations (i.e., `MAJ9`), `MAJX` operations can be used to correct not only a single fault but up to three faults in the systems. Due to space limitations, we leave the exploration of this use case to future work.

8.2. Content Destruction-Based Cold-Boot Attack Prevention

Cold-Boot Attack Prevention Mechanism. A cold boot attack is a physical attack on DRAM that involves hot-swapping a DRAM chip and reading out the contents of the DRAM chip [193–202]. Cold boot attacks are possible because the data stored in DRAM is not immediately lost when the chip is powered off. This is due to the capacitive nature of DRAM cells that can hold their data up to several minutes [193, 195, 203–206]. A practical and secure way to mitigate cold boot attacks is to destroy the DRAM content rapidly during power-off/on [92, 207]. Off-the-shelf PUD operations (i.e., RowClone [67, 72, 89], `Frac` [129], and `Multi-RowCopy` (§6)) can be used to rapidly destroy the DRAM content.

Experimental Methodology. We schedule DRAM command sequences to perform all content destruction operations (i.e., RowClone, `Frac`, and `Multi-RowCopy`) and measure the latency of each operation using DRAM Bender [155]. We then use an analytical model to calculate the total time to overwrite all data in a DRAM bank. The three PUD operation-based content destruction operations (i.e., RowClone-based, `Frac`-based, and `Multi-RowCopy`-based) can be implemented as follows: 1) **RowClone-based content destruction** is a two-step process. First, we issue a `WR` command to write predetermined data to an arbitrary row. Second, we perform RowClone from that row to all other DRAM rows, rewriting all original content. 2) **`Frac`-based content destruction** is implemented to repeatedly send the `Frac` operation to every row to place all DRAM rows into a neutral state, making them store $V_{DD}/2$. 3) **`Multi-RowCopy`-based content destruction** is implemented with varying numbers of rows that are simultaneously activated, from 2 to 32, in two steps. First, we issue a `WR` command to write predetermined data to an arbitrary row. Second, we perform up to 32-row activation with `Multi-RowCopy` operation to rapidly destroy the data in all rows.

Performance Evaluation. Fig. 17 shows the speedup in execution time for content destruction normalized to the RowClone-based content destruction's execution time.

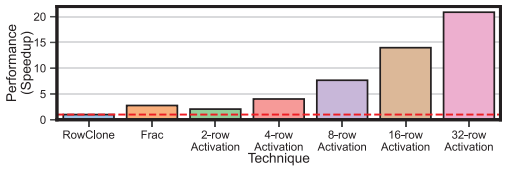


Figure 17: Speedup over RowClone-based content destruction

We make two key observations based on Fig. 17. First, `Multi-RowCopy`-based content destruction with 4-, 8-, 16- and 32-row activation outperforms both RowClone-based and `Frac`-based content destruction by up to 20.87× and 7.55×, respectively. Second, increasing the number of simultaneously activated rows increases the speedup of the `Multi-RowCopy`-based technique, as increasing the number of operands in `Multi-RowCopy` decreases the total number of operations. We conclude that `Multi-RowCopy`-based content destruction has a great potential to enable rapid destruction of DRAM content.

9. Limitations of Tested COTS DRAM Chips

We identify three key limitations of COTS DRAM chips in performing PUD operations.[12]

Limitation 1. Some COTS DRAM chips do *not* support all PUD operations. While we test COTS DRAM chips from all three major manufacturers (i.e., SK Hynix, Samsung, and Micron), we report major positive results and detailed evaluations in DRAM chips from Micron (Mfr. M) and SK Hynix (Mfr. H), which share more than half of the DRAM market (i.e., 55.8%) [208]. we can *simultaneously* activate many rows in a subarray, and thus, we can perform *all* the tested PUD operations. In the tested Samsung chips[13] (a total of 64 DRAM chips), we do not observe simultaneous activation of more than one row in a subarray. Hence, we do not observe any tested PUD operations in Samsung chips. We hypothesize that these DRAM chips have internal circuitry that ignores the `PRE` command or the second `ACT` command when the timing parameters (t_{RP} and t_{RAS}) are greatly violated, which is in line with the hypotheses provided by prior work [160]. If such a limitation were not imposed, we believe these DRAM chips are also fundamentally capable of performing the operations we examine in this work.

Limitation 2. Tested COTS DRAM chips support only consecutive activation of two rows and simultaneous activation of 2, 4, 8, 16, and 32 rows. In our experiments, we observe that we cannot control how many rows that DRAM chips simultaneously activate: either we can consecutively activate two rows or simultaneously activate 2, 4, 8, 16, and 32 rows. We hypothesize that this is due to our current infrastructure limitations, where we can issue DRAM commands at intervals of only 1.5ns. Controlling the number of activated rows may require finer-grained timing control between DRAM commands (e.g., 0.1ns). Having fine-grained control would allow us to deassert/assert desired intermediate signals in the row decoder circuitry, which could result in different numbers of simultaneously activated rows than 2, 4, 8, 16, and 32.

[12] We provide more details and limitations that we identify in the extended version [171].

[13] We provide more details on every tested chip (including Samsung chips) in the extended version of this paper [171].

Limitation 3. Performing PUD operations potentially have an effect on transient errors in DRAM chips. We perform each test 10000 times for each simultaneously activated row group and check for bitflips in the whole DRAM bank. We do not observe any errors in rows outside of the simultaneously activated row group across any of the tested DRAM chips. We believe that investigating all potential effects (e.g., on transient errors requires a much more extensive exploration of various aspects, which warrants its own study.

Our goal is to understand the undocumented computational capability of DRAM chips and the reliability of such computation capability. Despite the limitations mentioned above, our results can inspire future works to make PUD operations more reliable and/or leverage these operations in new ways in different applications with different requirements. We hope that future work builds on our study and that the resulting body of work enables DRAM manufacturers to reliably support operations that we demonstrate in future DRAM standards.

10. Related Work

To our knowledge, this is the first work that 1) rigorously characterizes the robustness of simultaneously activating up to 32 rows 2) demonstrates `MAJX` operations where X>3 3) demonstrated concurrently copying one row's content to up to 31 other rows (i.e., `Multi-RowCopy`), and 4) shows the success rate of `MAJX` can be improved by replicating the input operands of `MAJX` in COTS DDR4 DRAM chips. We discuss related works in two synergistic directions: 1) Processing-Using-DRAM (PUD) in COTS DRAM chips and 2) PUD in modified DRAM chips.

10.1. PUD in COTS DRAM Chips

Multiple Row Activation-based PUD Operations. Several prior works demonstrate bulk bitwise and data copy operations in COTS DRAM devices using multiple row activation [72, 86, 89, 128, 129, 155, 160]. ComputeDRAM [72] 1) activates three rows simultaneously (i.e., triple-row activation [64, 65, 68, 76, 77, 79]) to perform three-input majority and two-input AND and OR operations [64, 65, 68, 76, 77, 79] and 2) demonstrates copying one row's content to another row (i.e., the RowClone [67] operation) in COTS DDR3 chips. FracDRAM [129] shows that a DRAM cell in COTS DDR3 chips can store fractional values (e.g., VDD/2). FracDRAM uses fractional values to perform MAJ3 operations while simultaneously activating four DRAM rows in the same subarray. DRAM Bender [155, 163] demonstrates two-input AND and OR operations in COTS DDR4 chips. PiDRAM [89, 209] provides a flexible end-to-end FPGA-based framework that enables real system studies of PuD techniques, such as RowClone [67, 72]. A concurrent work [128] demonstrates that COTS DRAM chips are capable of 1) performing NOT and up to 16-input AND, NAND, OR, and NOR operations and 2) simultaneously activating up to 48 DRAM rows in two neighboring subarrays. None of these works 1) characterize the *robustness* of simultaneous many-row activation, 2) demonstrate `MAJX`, where X>3, 3) demonstrate copying one row's content to up to 31 other rows (`Multi-RowCopy`), and 4) provide a detailed hypothetical row decoder design that explains how and why many rows can be simultaneously activated.

Other works enable different functionalities using simultaneous many-row activation. QUAC-TRNG [86] introduces quadruple row activation and exploits this phenomenon to generate true random numbers in off-the-shelf DRAM chips. Our observations (e.g., `Multi-RowCopy` and simultaneously activating up to 32 DRAM rows) could also be leveraged to generate true random numbers. HiRA [160] demonstrates that real DRAM chips are capable of activating two rows (in quick succession) in electrically isolated (i.e., *not* physically adjacent) subarrays (called *hidden row activation*). This work uses hidden row activation to parallelize a DRAM row's refresh operation with the refresh or activation of other rows in the same bank.

Security Primitives. Prior works propose DRAM-based mechanisms to implement true random number generators (TRNGs) and physical unclonable functions (PUFs). DRAM-based TRNGs generate true random numbers by violating timing parameters [210, 211], using data retention failures [212, 213] and using startup values [214, 215]. DRAM-based PUFs generate signatures by using retention-based failures [212, 213, 216], violating timing parameters [87], exploiting write access latencies [217], and using startup values [218].

10.2. PuD in Modified DRAM Chips

Prior works [64–68, 70, 71, 76, 78–80, 110, 122, 219–239] enable bulk operations in DRAM chips by *modifying* the DRAM circuitry. We demonstrate that COTS DRAM chips are capable of performing PUD operations *without* any modification to DRAM chips or interface. We believe truly and robustly supporting operations that we demonstrate in DRAM requires proper modifications to DRAM circuitry and standards. Yet, our demonstration shows that existing COTS DRAM chips are already quite capable of computation, and such proper modifications to DRAM are very promising and likely to be very fruitful.

11. Conclusion

We presented our extensive characterization study on the computational capability of COTS DRAM chips and the robustness of these capabilities in 120 COTS DDR4 DRAM chips. Our study leads to 18 new empirical observations and shares 7 key takeaway lessons, which demonstrate that COTS DRAM chips are capable of performing `MAJ5`, `MAJ7`, `MAJ9`, and copying one row's content to up to 31 DRAM rows. We believe that our rigorous empirical results demonstrate the potential of using DRAM as a powerful computation substrate. We hope future works build upon our comprehensive study to better characterize, understand, and enhance the computational capability of DRAM chips.

Acknowledgements

We thank the anonymous reviewers of DSN 2024 for their encouraging feedback. We thank the SAFARI Research Group members for providing a stimulating intellectual environment. We acknowledge the generous gifts from our industrial partners, including Google, Huawei, Intel, and Microsoft. This work is supported in part by the Semiconductor Research Corporation (SRC), the ETH Future Computing Laboratory (EFCL), and the AI Chip Center for Emerging Smart Systems (ACCESS).

References

[1] O. Mutlu, S. Ghose, J. Gómez-Luna, and R. Ausavarungnirun, "Processing Data Where It Makes Sense: Enabling In-Memory Computation," *MICPRO*, 2019.

[2] O. Mutlu, S. Ghose, J. Gómez-Luna, and R. Ausavarungnirun, "A Modern Primer on Processing in Memory," in *Emerging Computing: From Devices to Systems — Looking Beyond Moore and Von Neumann*. Springer, 2021. [Online]. Available: https://arxiv.org/abs/2012.03112

[3] O. Mutlu, "Memory Scaling: A Systems Architecture Perspective," in *IMW*, 2013.

[4] O. Mutlu and L. Subramanian, "Research Problems and Opportunities in Memory Systems," *SUPERFRI*, 2014.

[5] J. Dean and L. A. Barroso, "The Tail at Scale," *CACM*, 2013.

[6] S. Kanev, J. P. Darago, K. Hazelwood, P. Ranganathan, T. Moseley, G.-Y. Wei, and D. Brooks, "Profiling a Warehouse-Scale Computer," in *ISCA*, 2015.

[7] M. Ferdman, A. Adileh, O. Kocberber, S. Volos, M. Alisafaee, D. Jevdjic, C. Kaynak, A. D. Popescu, A. Ailamaki, and B. Falsafi, "Clearing the Clouds: A Study of Emerging Scale-Out Workloads on Modern Hardware," in *ASPLOS*, 2012.

[8] L. Wang, J. Zhan, C. Luo, Y. Zhu, Q. Yang, Y. He, W. Gao, Z. Jia, Y. Shi, S. Zhang *et al.*, "BigDataBench: A Big Data Benchmark Suite from Internet Services," in *HPCA*, 2014.

[9] O. Mutlu, S. Ghose, J. Gómez-Luna, and R. Ausavarungnirun, "Enabling Practical Processing in and Near Memory For Data-Intensive Computing," in *DAC*, 2019.

[10] O. Mutlu, "Intelligent Architectures for Intelligent Machines," in *VLSI-DAT*, 2020.

[11] S. Ghose, A. Boroumand, J. S. Kim, J. Gómez-Luna, and O. Mutlu, "Processing-in-Memory: A Workload-Driven Perspective," *IBM JRD*, 2019.

[12] G. F. Oliveira, J. Gómez-Luna, L. Orosa, S. Ghose, N. Vijaykumar, I. Fernandez, M. Sadrosadati, and O. Mutlu, "DAMOV: A New Methodology and Benchmark Suite for Evaluating Data Movement Bottlenecks," *IEEE Access*, 2021.

[13] A. Boroumand, S. Ghose, Y. Kim, R. Ausavarungnirun, E. Shiu, R. Thakur, D. Kim, A. Kuusela, A. Knies, P. Ranganathan, and O. Mutlu, "Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks," in *ASPLOS*, 2018.

[14] A. Boroumand, S. Ghose, B. Akin, R. Narayanaswami, G. F. Oliveira, X. Ma, E. Shiu, and O. Mutlu, "Google Neural Network Models for Edge Devices: Analyzing and Mitigating Machine Learning Inference Bottlenecks," in *PACT*, 2021.

[15] S. Wang and E. Ipek, "Reducing Data Movement Energy via Online Data Clustering and Encoding," in *MICRO*, 2016.

[16] D. Pandiyan and C.-J. Wu, "Quantifying the Energy Cost of Data Movement for Emerging Smart Phone Workloads on Mobile Platforms," in *IISWC*, 2014.

[17] S. Koppula, L. Orosa, A. G. Yağlıkçı, R. Azizi, T. Shahroodi, K. Kanellopoulos, and O. Mutlu, "EDEN: Enabling Energy-Efficient, High-Performance Deep Neural Network Inference Using Approximate DRAM," in *MICRO*, 2019.

[18] U. Kang, H.-S. Yu, C. Park, H. Zheng, J. Halbert, K. Bains, S. Jang, and J. S. Choi, "Co-Architecting Controllers and DRAM to Enhance DRAM Process Scaling," in *The Memory Forum*, 2014.

[19] S. A. McKee *et al.*, "Reflections on the Memory Wall." *CF*, 2004.

[20] M. V. Wilkes, "The Memory Gap and the Future of High Performance Memories," *CAN*, 2001.

[21] Y. Kim, V. Seshadri, D. Lee, J. Liu, and O. Mutlu, "A Case for Exploiting Subarray-Level Parallelism (SALP) in DRAM," in *ISCA*, 2012.

[22] W. A. Wulf and S. A. McKee, "Hitting the Memory Wall: Implications of the Obvious," *CAN*, 1995.

[23] S. Ghose, T. Li, N. Hajinazar, D. S. Cali, and O. Mutlu, "Demystifying Complex Workload–DRAM Interactions: An Experimental Study," in *SIGMETRICS*, 2020.

[24] J. Ahn, S. Hong, S. Yoo, O. Mutlu, and K. Choi, "A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing," in *ISCA*, 2015.

[25] J. Ahn, S. Yoo, O. Mutlu, and K. Choi, "PIM-Enabled Instructions: A Low-Overhead, Locality-Aware Processing-in-Memory Architecture," in *ISCA*, 2015.

[26] K. Hsieh, E. Ebrahimi, G. Kim, N. Chatterjee, M. O'Connor, N. Vijaykumar, O. Mutlu, and S. W. Keckler, "Transparent Offloading and Mapping (TOM): Enabling Programmer-Transparent Near-Data Processing in GPU Systems," in *ISCA*, 2016.

[27] Y. Wang, L. Orosa, X. Peng, Y. Guo, S. Ghose, M. Patel, J. S. Kim, J. G. Luna, M. Sadrosadati, N. M. Ghiasi *et al.*, "FIGARO: Improving System Performance via Fine-Grained In-DRAM Data Relocation and Caching," in *MICRO*, 2020.

[28] R. Sites, "It's the Memory, Stupid!" *MPR*, 1996.

[29] F. Devaux, "The True Processing in Memory Accelerator," in *Hot Chips*, 2019.

[30] N. M. Ghiasi, J. Park, H. Mustafa, J. Kim, A. Olgun, A. Gollwitzer, D. S. Cali, C. Firtina, H. Mao, N. A. Alserr *et al.*, "GenStore: A High-Performance and Energy-Efficient In-Storage Computing System for Genome Sequence Analysis," in *ASPLOS*, 2022.

[31] J. Gómez-Luna, I. El Hajj, I. Fernandez, C. Giannoula, G. F. Oliveira, and O. Mutlu, "Benchmarking Memory-Centric Computing Systems: Analysis of Real Processing-in-Memory Hardware," in *CUT*, 2021.

[32] J. Gómez-Luna, I. E. Hajj, I. Fernández, C. Giannoula, G. F. Oliveira, and O. Mutlu, "Benchmarking a New Paradigm: An Experimental Analysis of a Real Processing-in-Memory Architecture," arXiv:2105.03814 [cs.AR], 2021.

[33] J. Gómez-Luna, I. El Hajj, I. Fernandez, C. Giannoula, G. F. Oliveira, and O. Mutlu, "Benchmarking a New Paradigm: Experimental Analysis and Characterization of a Real Processing-in-Memory System," *IEEE Access*, 2022.

[34] C. Giannoula, N. Vijaykumar, N. Papadopoulou, V. Karakostas, I. Fernandez, J. Gómez-Luna, L. Orosa, N. Koziris, G. Goumas, and O. Mutlu, "SynCron: Efficient Synchronization Support for Near-Data-Processing Architectures," in *HPCA*, 2021.

[35] G. Singh, D. Diamantopoulos, C. Hagleitner, J. Gomez-Luna, S. Stuijk, O. Mutlu, and H. Corporaal, "NERO: A Near High-Bandwidth Memory Stencil Accelerator for Weather Prediction Modeling," in *FPL*, 2020.

[36] S. Lee, K. Kim, S. Oh, J. Park, G. Hong, D. Ka, K. Hwang, J. Park, K. Kang, J. Kim, J. Jeon, N. Kim, Y. Kwon, K. Vladimir, W. Shin, J. Won, M. Lee, H. Joo *et al.*, "A 1ynm 1.25V 8Gb, 16Gb/s/pin GDDR6-Based Accelerator-in-Memory Supporting 1TFLOPS MAC Operation and Various Activation Functions for Deep-Learning Applications," in *ISSCC*, 2022.

[37] L. Ke, X. Zhang, J. So, J.-G. Lee, S.-H. Kang, S. Lee, S. Han, Y. Cho, J. H. Kim, Y. Kwon *et al.*, "Near-Memory Processing in Action: Accelerating Personalized Recommendation with AxDIMM," *IEEE Micro*, 2021.

[38] C. Giannoula, I. Fernandez, J. Gómez-Luna, N. Koziris, G. Goumas, and O. Mutlu, "SparseP: Towards Efficient Sparse Matrix Vector Multiplication on Real Processing-in-Memory Systems," in *SIGMETRICS*, 2022.

[39] A. Denzler, R. Bera, N. Hajinazar, G. Singh, G. F. Oliveira, J. Gómez-Luna, and O. Mutlu, "Casper: Accelerating Stencil Computation using Near-Cache Processing," *IEEE Access*, 2023.

[40] D. Patterson, T. Anderson, N. Cardwell *et al.*, "A Case for Intelligent RAM," *IEEE Micro*, 1997.

[41] D. G. Elliott, M. Stumm, W. M. Snelgrove *et al.*, "Computational RAM: Implementing Processors in Memory," *Design and Test of Computers*, 1999.

[42] M. Gokhale, B. Holmes, and K. Iobst, "Processing in Memory: The Terasys Massively Parallel PIM Array," in *Computer*, 1995.

[43] M. Hall, P. Kogge, J. Koller, P. Diniz, J. Chame, J. Draper, J. LaCoss, J. Granacki, J. Brockman, A. Srivastava *et al.*, "Mapping Irregular Applications to DIVA, a PIM-Based Data-Intensive Architecture," in *SC*, 1999.

[44] A. Boroumand, S. Ghose, M. Patel, H. Hassan, B. Lucia, K. Hsieh, K. T. Malladi, H. Zheng, and O. Mutlu, "LazyPIM: An Efficient Cache Coherence Mechanism for Processing-in-Memory," in *CAL*, 2016.

[45] A. Boroumand, S. Ghose, M. Patel, H. Hassan, B. Lucia, N. Hajinazar, K. Hsieh, K. T. Malladi, H. Zheng, and O. Mutlu, "LazyPIM: Efficient Support for Cache Coherence in Processing-in-Memory Architectures," arXiv:1706.03162 [cs.AR], 2017.

[46] D. Zhang, N. Jayasena, A. Lyashevsky, J. L. Greathouse, L. Xu, and M. Ignatowski, "TOP-PIM: Throughput-Oriented Programmable Processing in Memory," in *HPDC*, 2014.

[47] J. S. Kim, D. S. Cali, H. Xin, D. Lee, S. Ghose, M. Alser, H. Hassan, O. Ergin, C. Alkan, and O. Mutlu, "GRIM-Filter: Fast Seed Location Filtering in DNA Read Mapping using Processing-in-Memory Technologies," in *APBC*, 2018.

[48] P. C. Santos, G. F. Oliveira, D. G. Tomé, M. A. Z. Alves, E. C. Almeida, and L. Carro, "Operand Size Reconfiguration for Big Data Processing in Memory," in *DATE*, 2017.

[49] G. F. Oliveira, P. C. Santos, M. A. Alves, and L. Carro, "NIM: An HMC-Based Machine for Neuron Computation," in *ARC*, 2017.

[50] D. Kim, J. Kung, S. Chai, S. Yalamanchili, and S. Mukhopadhyay, "Neurocube: A Programmable Digital Neuromorphic Architecture with High-Density 3D Memory," in *ISCA*, 2016.

[51] A. Boroumand, S. Ghose, M. Patel, H. Hassan, B. Lucia, R. Ausavarungnirun, K. Hsieh, N. Hajinazar, K. T. Malladi, H. Zheng *et al.*, "CoNDA: Efficient Cache Coherence Support for Near-Data Accelerators," in *ISCA*, 2019.

[52] K. Hsieh, S. Khan, N. Vijaykumar, K. K. Chang, A. Boroumand, S. Ghose, and O. Mutlu, "Accelerating Pointer Chasing in 3D-Stacked Memory: Challenges, Mechanisms, Evaluation," in *ICCD*, 2016.

[53] A. Boroumand, S. Ghose, B. Akin, R. Narayanaswami, G. F. Oliveira, X. Ma, E. Shiu, and O. Mutlu, "Mitigating Edge Machine Learning Inference Bottlenecks: An Empirical Study on Accelerating Google Edge Models," arXiv:2103.00768 [cs.AR], 2021.

[54] A. Boroumand, S. Ghose, G. F. Oliveira, and O. Mutlu, "Polynesia: Enabling High-Performance and Energy-Efficient Hybrid Transactional/Analytical Databases with Hardware/Software Co-Design," in *ICDE*, 2022.

[55] A. Boroumand, S. Ghose, G. F. Oliveira, and O. Mutlu, "Polynesia: Enabling Effective Hybrid Transactional/Analytical Databases with Specialized Hardware/Software Co-Design," arXiv:2103.00798 [cs.AR], 2021.

[56] M. Besta, R. Kanakagiri, G. Kwasniewski, R. Ausavarungnirun, J. Beránek, K. Kanellopoulos, K. Janda, Z. Vonarburg-Shmaria, L. Gianinazzi, I. Stefan *et al.*, "SISA: Set-Centric Instruction Set Architecture for Graph Mining on Processing-in-Memory Systems," in *MICRO*, 2021.

[57] I. Fernandez, R. Quislant, C. Giannoula, M. Alser, J. Gomez-Luna, E. Gutierrez, O. Plata, and O. Mutlu, "NATSA: A Near-Data Processing Accelerator for Time Series Analysis," in *ICCD*, 2020.

[58] G. Singh, J. Gómez-Luna, G. Mariani, G. F. Oliveira, S. Corda, S. Stuijk, O. Mutlu, and H. Corporaal, "NAPEL: Near-Memory Computing Application Performance Prediction via Ensemble Learning," in *DAC*, 2019.

[59] S. Lee, S.-h. Kang, J. Lee, H. Kim, E. Lee, S. Seo, H. Yoon, S. Lee, K. Lim, H. Shin *et al.*, "Hardware Architecture and Software Stack for PIM Based on Commercial DRAM Technology: Industrial Product," in *ISCA*, 2021.

[60] J. S. Kim, D. S. Cali, H. Xin, D. Lee, S. Ghose, M. Alser, H. Hassan, O. Ergin, C. Alkan, and O. Mutlu, "GRIM-Filter: Fast Seed Location Filtering in DNA Read Mapping using Processing-in-Memory Technologies," in *APBC*, 2018.

[61] P. C. Santos, G. F. Oliveira, J. P. Lima, M. A. Alves, L. Carro, and A. C. Beck, "Processing in 3D Memories to Speed Up Operations on Complex Data Structures," in *DATE*, 2018.

[62] P. Chi, S. Li, C. Xu, T. Zhang, J. Zhao, Y. Liu, Y. Wang, and Y. Xie, "PRIME: A Novel Processing-in-Memory Architecture for Neural Network Computation in ReRAM-Based Main Memory," in *ISCA*, 2016.

[63] A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, and V. Srikumar, "ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars," in *ISCA*, 2016.

[64] V. Seshadri, D. Lee, T. Mullins, H. Hassan, A. Boroumand, J. Kim, M. A. Kozuch, O. Mutlu, P. B. Gibbons, and T. C. Mowry, "Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology," in *MICRO*, 2017.

[65] V. Seshadri and O. Mutlu, "In-DRAM Bulk Bitwise Execution Engine," arXiv:1905.09822 [cs.AR], 2019.

[66] S. Li, D. Niu, K. T. Malladi, H. Zheng, B. Brennan, and Y. Xie, "DRISA: A DRAM-Based Reconfigurable In-Situ Accelerator," in *MICRO*, 2017.

[67] V. Seshadri, Y. Kim, C. Fallin, D. Lee, R. Ausavarungnirun, G. Pekhimenko, Y. Luo, O. Mutlu, P. B. Gibbons, M. A. Kozuch, and T. Mowry, "RowClone: Fast and Energy-Efficient In-DRAM Bulk Data Copy and Initialization," in *MICRO*, 2013.

[68] V. Seshadri and O. Mutlu, "The Processing Using Memory Paradigm: In-DRAM Bulk Copy, Initialization, Bitwise AND and OR," arXiv:1610.09603 [cs.AR], 2016.

[69] Q. Deng, L. Jiang, Y. Zhang, M. Zhang, and J. Yang, "DrAcc: A DRAM Based Accelerator for Accurate CNN Inference," in *DAC*, 2018.

[70] X. Xin, Y. Zhang, and J. Yang, "ELP2IM: Efficient and Low Power Bitwise Operation Processing in DRAM," in *HPCA*, 2020.

[71] L. Song, X. Qian, H. Li, and Y. Chen, "PipeLayer: A Pipelined ReRAM-Based Accelerator for Deep Learning," in *HPCA*, 2017.

[72] F. Gao, G. Tziantzioulis, and D. Wentzlaff, "ComputeDRAM: In-Memory Compute Using Off-the-Shelf DRAMs," in *MICRO*, 2019.

[73] C. Eckert, X. Wang, J. Wang, A. Subramaniyan, R. Iyer, D. Sylvester, D. Blaauw, and R. Das, "Neural Cache: Bit-Serial In-Cache Acceleration of Deep Neural Networks," in *ISCA*, 2018.

[74] S. Aga, S. Jeloka, A. Subramaniyan, S. Narayanasamy, D. Blaauw, and R. Das, "Compute Caches," in *HPCA*, 2017.

[75] D. Fujiki, S. Mahlke, and R. Das, "Duality Cache for Data Parallel Acceleration," in *ISCA*, 2019.

[76] V. Seshadri, D. Lee, T. Mullins, H. Hassan, A. Boroumand, J. Kim, M. A. Kozuch, O. Mutlu, P. B. Gibbons, and T. C. Mowry, "Buddy-RAM: Improving the Performance and Efficiency of Bulk Bitwise Operations Using DRAM," arXiv:1611.09988 [cs.AR], 2016.

[77] V. Seshadri and O. Mutlu, "Simple Operations in Memory to Reduce Data Movement," in *Advances in Computers, Volume 106*, 2017.

[78] V. Seshadri, Y. Kim, C. Fallin, D. Lee, R. Ausavarungnirun, G. Pekhimenko, Y. Luo, O. Mutlu, P. B. Gibbons, M. A. Kozuch, and T. C. Mowry, "RowClone: Accelerating Data Movement and Initialization Using DRAM," arXiv:1805.03502 [cs.AR], 2018.

[79] V. Seshadri, K. Hsieh, A. Boroumand, D. Lee, M. A. Kozuch†, O. Mutlu, P. B. Gibbons, and T. C. Mowry, "Fast Bulk Bitwise AND and OR in DRAM," in *CAL*, 2015.

[80] S. Li, C. Xu, Q. Zou, J. Zhao, Y. Lu, and Y. Xie, "Pinatubo: A Processing-in-Memory Architecture for Bulk Bitwise Operations in Emerging Non-Volatile Memories," in *DAC*, 2016.

[81] J. D. Ferreira, G. Falcao, J. Gómez-Luna, M. Alser, L. Orosa, M. Sadrosadati, J. S. Kim, G. F. Oliveira, T. Shahroodi, A. Nori *et al.*, "pLUTo: In-DRAM Lookup Tables to Enable Massively Parallel General-Purpose Computation," arXiv:2104.07699 [cs.AR], 2021.

[82] J. D. Ferreira, G. Falcao, J. Gómez-Luna, M. Alser, L. Orosa, M. Sadrosadati, J. S. Kim, G. F. Oliveira, T. Shahroodi, A. Nori *et al.*, "pLUTo: Enabling Massively Parallel Computation in DRAM via Lookup Tables," in *MICRO*, 2022.

[83] J. Park, R. Azizi, G. F. Oliveira, M. Sadrosadati, R. Nadig, D. Novo, J. Gómez-Luna, M. Kim, and O. Mutlu, "Flash-Cosmos: In-Flash Bulk Bitwise Operations Using Inherent Computation Capability of NAND Flash Memory," in *MICRO*, 2022.

[84] M. S. Truong, L. Shen, A. Glass, A. Hoffmann, L. R. Carley, J. A. Bain, and S. Ghose, "Adapting the RACER Architecture to Integrate Improved In-ReRAM Logic Primitives," *JETCAS*, 2022.

[85] M. S. Truong, E. Chen, D. Su, L. Shen, A. Glass, L. R. Carley, J. A. Bain, and S. Ghose, "RACER: Bit-Pipelined Processing Using Resistive Memory," in *MICRO*, 2021.

[86] A. Olgun, M. Patel, A. G. Yağlıkçı, H. Luo, J. S. Kim, N. Bostanci, N. Vijaykumar, O. Ergin, and O. Mutlu, "QUAC-TRNG: High-Throughput True Random Number Generation Using Quadruple Row Activation in Commodity DRAM Chips," in *ISCA*, 2021.

[87] J. S. Kim, M. Patel, H. Hassan, and O. Mutlu, "The DRAM Latency PUF: Quickly Evaluating Physical Unclonable Functions by Exploiting the Latency–Reliability Tradeoff in Modern Commodity DRAM Devices," in *HPCA*, 2018.

[88] F. N. Bostancı, A. Olgun, L. Orosa, A. G. Yağlıkçı, J. S. Kim, H. Hassan, O. Ergin, and O. Mutlu, "DR-STRaNGe: End-to-End System Design for DRAM-Based True Random Number Generators," in *HPCA*, 2022.

[89] A. Olgun, J. G. Luna, K. Kanellopoulos, B. Salami, H. Hassan, O. Ergin, and O. Mutlu, "PiDRAM: A Holistic End-to-End FPGA-Based Framework for Processing-In-DRAM," *TACO*, 2022.

[90] M. F. Ali, A. Jaiswal, and K. Roy, "In-Memory Low-Cost Bit-Serial Addition Using Commodity DRAM Technology," in *TCAS I*, 2019.

[91] S. Li, A. O. Glova, X. Hu, P. Gu, D. Niu, K. T. Malladi, H. Zheng, B. Brennan, and Y. Xie, "SCOPE: A Stochastic Computing Engine for DRAM-Based In-Situ Accelerator," in *MICRO*, 2018.

[92] L. Orosa, Y. Wang, M. Sadrosadati, J. S. Kim, M. Patel, I. Puddu, H. Luo, K. Razavi, J. Gómez-Luna, H. Hassan, N. Mansouri-Ghiasi, S. Ghose, and O. Mutlu, "CODIC: A Low-Cost Substrate for Enabling Custom In-DRAM Functionalities and Optimizations," in *ISCA*, 2021.

[93] M. Sharad, D. Fan, and K. Roy, "Ultra Low Power Associative Computing with Spin Neurons and Resistive Crossbar Memory," in *ADAC*, 2013.

[94] C. Gao, X. Xin, Y. Lu, Y. Zhang, J. Yang, and J. Shu, "ParaBit: Processing Parallel Bitwise Operations in NAND Flash Memory Based SSDs," in *MICRO*, 2021.

[95] W. H. Choi, P.-F. Chiu, W. Ma, G. Hemink, T. T. Hoang, M. Lueker-Boden, and Z. Bandic, "An In-Flash Binary Neural Network Accelerator with SLC NAND Flash Array," in *ISCAS*, 2020.

[96] R. Han, P. Huang, Y. Xiang, C. Liu, Z. Dong, Z. Su, Y. Liu, L. Liu, X. Liu, and J. Kang, "A Novel Convolution Computing Paradigm Based on NOR Flash Array with High Computing Speed and Energy Efficiency," *TCAS-I*, 2019.

[97] F. Merrikh-Bayat, X. Guo, M. Klachko, M. Prezioso, K. K. Likharev, and D. B. Strukov, "High-Performance Mixed-Signal Neurocomputing with Nanoscale Floating-Gate Memory Cell Arrays," *TNNLS*, 2017.

[98] P. Wang, F. Xu, B. Wang, B. Gao, H. Wu, H. Qian, and S. Yu, "Three-Dimensional NAND Flash for Vector–Matrix Multiplication," *TVLSI*, 2018.

[99] D. S. Cali, G. S. Kalsi, Z. Bingöl, C. Firtina, L. Subramanian, J. S. Kim, R. Ausavarungnirun, M. Alser, J. Gomez-Luna, A. Boroumand *et al.*, "GenASM: A High-Performance, Low-Power Approximate String Matching Acceleration Framework for Genome Sequence Analysis," in *MICRO*, 2020.

[100] A. Nag, C. Ramachandra, R. Balasubramonian, R. Stutsman, E. Giacomin, H. Kambalasubramanyam, and P.-E. Gaillardon, "GenCache: Leveraging In-Cache Operators for Efficient Sequence Alignment," in *MICRO*, 2019.

[101] M. Kang, M.-S. Keel, N. R. Shanbhag, S. Eilert, and K. Curewitz, "An Energy-Efficient VLSI Architecture for Pattern Recognition via Deep Embedding of Computation in SRAM," in *ICASSP*, 2014.

[102] Z. Wang, C. Liu, A. Arora, L. John, and T. Nowatzki, "Infinity Stream: Portable and Programmer-Friendly In-/Near-Memory Fusion," in *ASPLOS*, 2023.

[103] M. Kang, E. P. Kim, M.-s. Keel, and N. R. Shanbhag, "Energy-Efficient and High Throughput Sparse Distributed Memory Architecture," in *ISCAS*, 2015.

[104] K. K. Chang, P. J. Nair, D. Lee, S. Ghose, M. K. Qureshi, and O. Mutlu, "Low-Cost Inter-Linked Subarrays (LISA): Enabling Fast Inter-Subarray Data Movement in DRAM," in *HPCA*, 2016.

[105] N. Hajinazar, G. F. Oliveira, S. Gregorio, J. D. Ferreira, N. M. Ghiasi, M. Patel, M. Alser, S. Ghose, J. Gómez-Luna, and O. Mutlu, "SIMDRAM: A Framework for Bit-Serial SIMD Processing Using DRAM," in *ASPLOS*, 2021.

[106] P. R. Sutradhar, S. Bavikadi, M. Connolly, S. Prajapati, M. A. Indovina, S. M. P. Dinakarrao, and A. Ganguly, "Look-Up-Table Based Processing-in-Memory Architecture with Programmable Precision-Scaling for Deep Learning Applications," *TPDS*, 2021.

[107] P. R. Sutradhar, M. Connolly, S. Bavikadi, S. M. P. Dinakarrao, M. A. Indovina, and A. Ganguly, "pPIM: A Programmable Processor-in-Memory Architecture with Precision-Scaling For Deep Learning," in *CAL*, 2020.

[108] M. Lenjani, P. Gonzalez, E. Sadredini, S. Li, Y. Xie, A. Akel, S. Eilert, M. R. Stan, and K. Skadron, "Fulcrum: A Simplified Control and Access Mechanism Toward Flexible and Practical In-Situ Accelerators," in *HPCA*, 2020.

[109] X. Peng, Y. Wang, and M.-C. Yang, "CHOPPER: A Compiler Infrastructure for Programmable Bit-Serial SIMD Processing Using Memory In DRAM," in *HPCA*, 2023.

[110] G. F. Oliveira, J. Gómez-Luna, S. Ghose, A. Boroumand, and O. Mutlu, "Accelerating Neural Network Inference with Processing-in-DRAM: From the Edge to the Cloud," *IEEE Micro*, 2022.

[111] G. Singh, M. Alser, D. S. Cali, D. Diamantopoulos, J. Gómez-Luna, H. Corporaal, and O. Mutlu, "FPGA-Based Near-Memory Acceleration of Modern Data-Intensive Applications," *IEEE Micro*, 2021.

[112] G. F. Oliveira, A. Kohli, D. Novo, J. Gómez-Luna, and O. Mutlu, "DaPPA: A Data-Parallel Framework for Processing-in-Memory Architectures," arXiv:2310.10168 [cs.AR], 2023.

[113] G. F. Oliveira, J. Gómez-Luna, S. Ghose, and O. Mutlu, "Methodologies, Workloads, and Tools for Processing-in-Memory: Enabling the Adoption of Data-Centric Architectures," in *ISVLSI*, 2022.

[114] G. F. Oliveira, A. Boroumand, S. Ghose, J. Gómez-Luna, and O. Mutlu, "Heterogeneous Data-Centric Architectures for Modern Data-Intensive Applications: Case Studies in Machine Learning and Databases," in *ISVLSI*, 2022.

[115] J. Chen, J. Gómez-Luna, I. E. Hajj, Y. Guo, and O. Mutlu, "SimplePIM: A Software Framework for Productive and Efficient Processing-In-Memory," in *PACT*, 2023.

[116] H. Gupta, M. Kabra, J. Gómez-Luna, K. Kanellopoulos, and O. Mutlu, "Evaluating Homomorphic Operations on a Real-World Processing-In-Memory System," in *IISWC*, 2023.

[117] J. Gómez-Luna, Y. Guo, S. Brocard, J. Legriel, R. Cimadomo, G. F. Oliveira, G. Singh, and O. Mutlu, "Evaluating Machine LearningWorkloads on Memory-Centric Computing Systems," in *ISPASS*, 2023.

[118] M. Item, J. Gómez-Luna, Y. Guo, G. F. Oliveira, M. Sadrosadati, and O. Mutlu, "TransPimLib: Efficient Transcendental Functions for Processing-in-Memory Systems," in *ISPASS*, 2023.

[119] S. Diab, A. Nassereldine, M. Alser, J. Gómez Luna, O. Mutlu, and I. El Hajj, "A Framework for High-Throughput Sequence Alignment Using Real Processing-In-Memory Systems," *Bioinformatics*, 2023.

[120] H. Mao, M. Alser, M. Sadrosadati, C. Firtina, A. Baranwal, D. S. Cali, A. Manglik, N. A. Alserr, and O. Mutlu, "GenPIP: In-Memory Acceleration of Genome Analysis via Tight Integration of Basecalling and Read Mapping," in *MICRO*, 2022.

[121] G. Singh, D. Diamantopoulos, J. Gómez-Luna, C. Hagleitner, S. Stuijk, H. Corporaal, and O. Mutlu, "Accelerating Weather Prediction Using Near-Memory Reconfigurable Fabric," *TRETS*, 2022.

[122] V. Seshadri and O. Mutlu, "Simple Operations in Memory to Reduce Data Movement," in *Adv. Comput.*, 2017.

[123] S. Angizi and D. Fan, "ReDRAM: A Reconfigurable Processing-in-DRAM Platform for Accelerating Bulk Bit-Wise Operations," in *ICCAD*, 2019.

[124] Hybrid Memory Cube Consortium, "Hybrid Memory Cube Specification Rev. 2.0," http://www.hybridmemorycube.org/.

[125] JEDEC, "JESD235 High Bandwidth Memory (HBM) DRAM," 2013.

[126] D. Lee, S. Ghose, G. Pekhimenko, S. Khan, and O. Mutlu, "Simultaneous Multi-Layer Access: Improving 3D-Stacked Memory Bandwidth at Low Cost," *ACM TACO*, 2016.

[127] G. F. Oliveira, A. Olgun, A. G. G. Yaglikçi, N. Bostanci, J. Gómez-Luna, S. Ghose, and O. Mutlu, "MIMDRAM: An End-to-End Processing-Using-DRAM System for High-Throughput, Energy-Efficient and Programmer-Transparent Multiple-Instruction Multiple-Data Processing," HPCA, 2024.

[128] I. E. Yuksel, Y. C. Tugrul, A. Olgun, F. N. Bostanci, A. G. Yaglikci, G. F. de Oliveira, H. Luo, J. G. Luna, M. Sadrosadati, and O. Mutlu, "Functionally-Complete Boolean Logic in Real DRAM Chips: Experimental Characterization and Analysis," in

HPCA, 2024.
[129] F. Gao, G. Tziantzioulis, and D. Wentzlaff, "FracDRAM: Fractional Values in Off-the-Shelf DRAM," in *MICRO*, 2022.
[130] N. Hajinazar, G. F. Oliveira, S. Gregorio, J. D. Ferreira, N. M. Ghiasi, M. Patel, M. Alser, S. Ghose, J. Gómez-Luna, and O. Mutlu, "SIMDRAM: A Framework for Bit-Serial SIMD Processing Using DRAM," in *ASPLOS*, 2021.
[131] S. Angizi and D. Fan, "GraphiDe: A Graph Processing Accelerator Leveraging In-DRAM-Computing," in *GLSVLSI*, 2019.
[132] Q. Deng, Y. Zhang, M. Zhang, and J. Yang, "LAcc: Exploiting Lookup Table-Based Fast and Accurate Vector Multiplication in DRAM-Based CNN Accelerator," in *DAC*, 2019.
[133] C.-Y. Chan and Y. E. Ioannidis, "Bitmap Index Design and Evaluation," in *SIGMOD*, 1998.
[134] E. O'Neil, P. O'Neil, and K. Wu, "Bitmap Index Design Choices and Their Performance Implications," in *IDEAS*, 2007.
[135] Y. Li and J. M. Patel, "WideTable: An Accelerator for Analytical Data Processing," *VLDB*, 2014.
[136] Y. Li and J. M. Patel, "BitWeaving: Fast Scans for Main Memory Data Processing," in *SIGMOD*, 2013.
[137] B. Goodwin, M. Hopcroft, D. Luu, A. Clemmer, M. Curmei, S. Elnikety, and Y. He, "BitFunnel: Revisiting Signatures for Search," in *SIGIR*, 2017.
[138] K. Wu, "FastBit: An Efficient Indexing Technology For Accelerating Data-Intensive Science," in *Journal of Physics*, 2005.
[139] M.-C. Wu and A. P. Buchmann, "Encoded Bitmap Indexing For Data Warehouses," in *ICDE*, 1998.
[140] Redis, "Redis bitmaps," https://redis.io/docs/data-types/bitmaps/.
[141] B. Perach, R. Ronen, B. Kimelfeld, and S. Kvatinsky, "Understanding Bulk-Bitwise Processing In-Memory Through Database Analytics," *ETC*, 2023.
[142] S.-W. Jun, M. Liu, S. Lee, J. Hicks, J. Ankcorn, M. King, and S. Xu, "Bluedbm: An Appliance For Big Data Analytics," *SIGARCH*, 2015.
[143] M. Torabzadehkashi, S. Rezaei, A. Heydarigorji, H. Bobarshad, V. Alves, and N. Bagherzadeh, "Catalina: In-Storage Processing Acceleration For Scalable Big Data Analytics," in *PDP*, 2019.
[144] J. H. Lee, H. Zhang, V. Lagrange, P. Krishnamoorthy, X. Zhao, and Y. S. Ki, "SmartSSD: FPGA Accelerated Near-Storage Data Analytics on SSD," *CAL*, 2020.
[145] S. Beamer, K. Asanovic, and D. Patterson, "Direction-Optimizing Breadth-First Search," in *SC*, 2012.
[146] M. Alser, H. Hassan, H. Xin, O. Ergin, O. Mutlu, and C. Alkan, "Gatekeeper: A New Hardware Architecture For Accelerating Pre-Alignment In DNA Short Read Mapping," in *Bioinformatics*, 2017.
[147] J. Loving, Y. Hernandez, and G. Benson, "BitPAl: A Bit-Parallel, General Integer-Scoring Sequence Alignment Algorithm," *Bioinformatics*, 2014.
[148] H. Xin, J. Greth, J. Emmons, G. Pekhimenko, C. Kingsford, C. Alkan, and O. Mutlu, "Shifted Hamming Distance: A Fast and Accurate SIMD-Friendly Filter to Accelerate Alignment Verification in Read Mapping," *Bioinformatics*, 2015.
[149] G. Myers, "A Fast Bit-Vector Algorithm for Approximate String Matching Based on Dynamic Programming," *JACM*, 1999.
[150] J. Han, C.-S. Park, D.-H. Ryu, and E.-S. Kim, "Optical Image Encryption Based on XOR Operations," *Optical Engineering*, 1999.
[151] P. Tuyls, H. D. Hollmann, J. V. Lint, and L. Tolhuizen, "XOR-based Visual Cryptography Schemes," *Des. Codes, Cryptogr.*, 2005.
[152] P. Kanerva, "Sparse Distributed Memory and Related Models," Tech. Rep., 1992.
[153] P. Kanerva, "Hyperdimensional Computing: An Introduction to Computing in Distributed Representation with High-Dimensional Random Vectors," *Cognitive Computation*, 2009.
[154] G. Karunaratne, M. Le Gallo, G. Cherubini, L. Benini, A. Rahimi, and A. Sebastian, "In-memory Hyperdimensional Computing," *Nature Electronics*, 2020.
[155] A. Olgun, H. Hassan, A. G. Yağlıkçı, Y. C. Tuğrul, L. Orosa, H. Luo, M. Patel, O. Ergin, and O. Mutlu, "DRAM Bender: An Extensible and Versatile FPGA-based Infrastructure to Easily Test State-of-the-art DRAM Chips," *TCAD*, 2023.
[156] JEDEC, *JESD79-4C: DDR4 SDRAM Standard*, 2020.
[157] D. Lee, Y. Kim, G. Pekhimenko, S. Khan, V. Seshadri, K. Chang, and O. Mutlu, "Adaptive-Latency DRAM: Optimizing DRAM Timing for the Common-Case," in *HPCA*, 2015.
[158] K. K. Chang, P. J. Nair, D. Lee, S. Ghose, M. K. Qureshi, and O. Mutlu, "Low-Cost Inter-Linked Subarrays (LISA): Enabling Fast Inter-Subarray Data Movement in DRAM," in *HPCA*, 2016.
[159] K. K. Chang, A. G. Yağlıkçı, S. Ghose, A. Agrawal, N. Chatterjee, A. Kashyap, D. Lee, M. O'Connor, H. Hassan, and O. Mutlu, "Understanding Reduced-Voltage Operation in Modern DRAM Devices: Experimental Characterization, Analysis, and Mechanisms," in *SIGMETRICS*, 2017.
[160] A. G. Yağlikci, A. Olgun, M. Patel, H. Luo, H. Hassan, L. Orosa, O. Ergin, and O. Mutlu, "HiRA: Hidden Row Activation for Reducing Refresh Latency of Off-the-Shelf DRAM Chips," in *MICRO*, 2022.
[161] Linear Technology Corp., "LTspice IV," https://t.ly/mAlfJ.
[162] A. Vladimirescu, *The SPICE Book*. Wiley New York, 1994.
[163] SAFARI Research Group, "DRAM Bender — GitHub Repository," https://github.com/CMU-SAFARI/DRAM-Bender, 2022.
[164] Xilinx Inc., "Xilinx Alveo U200 FPGA Board," https://www.xilinx.com/products/boards-and-kits/alveo/u200.html.
[165] Maxwell, "FT20X User Manual," https://t.ly/zn6wP.
[166] TTi, "PL & PL-P Series DC Power Supplies Data Sheet - Issue 5," https://t.ly/TgIH4.
[167] JEDEC, *JESD79-5: DDR5 SDRAM Standard*, 2020.
[168] JEDEC, *JESD232A: Graphics Double Data Rate (GDDR5X) Standard*, 2016.
[169] JEDEC, *JESD250C: Graphics Double Data Rate 6 (GDDR6) Standard*, 2021.
[170] A. G. Yağlıkcı, H. Luo, G. F. De Oliviera, A. Olgun, M. Patel, J. Park, H. Hassan, J. S. Kim, L. Orosa, and O. Mutlu, "Understanding RowHammer Under Reduced Wordline Voltage: An Experimental Study Using Real DRAM Devices," in *DSN*, 2022.
[171] I. E. Yuksel, Y. C. Tugrul, F. N. Bostanci, G. F. de Oliveira, A. G. Yaglikci, A. Olgun, M. Soysal, H. Luo, J. G. Luna, M. Sadrosadati, and O. Mutlu, "Simultaneous Many-Row Activation in Off-the-Shelf DRAM Chips: Experimental Characterization and Analysis," in *arXiv*, 2024.
[172] K. Chang *et al.*, "Understanding Reduced-Voltage Operation in Modern DRAM Devices: Experimental Characterization, Analysis, and Mechanisms," in *SIGMETRICS*, 2017.
[173] B. Keeth *et al.*, *DRAM Circuit Design. Fundamental and High-Speed Topics*. Wiley-IEEE Press, 2007.
[174] Rambus Inc., "Rambus Power Model," https://www.rambus.com/energy/.
[175] International Technology Roadmap for Semiconductors, "ITRS Reports," http://www.itrs2.net/itrs-reports.html, 2015.
[176] T. Vogelsang, "Understanding the Energy Consumption of Dynamic Random Access Memories," in *MICRO*, 2010.
[177] Nanoscale Integration and Modeling (NIMO) Group, ASU, "Predictive Technology Model (PTM)," http://ptm.asu.edu/, 2012.
[178] F. James, "Monte Carlo Theory and Practice," *Rep Prog Phys.*, 1980.
[179] T. Sakata, K. Itoh, M. Horiguchi, and M. Aoki, "Subthreshold-Current Reduction Circuits for Multi-Gigabit DRAM's," *JSSC*, 1994.
[180] J. Kao, S. Narendra, and A. Chandrakasan, "Subthreshold Leakage Modeling and Reduction Techniques," in *ICCAD*, 2002.
[181] N. H. Weste and D. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*. Pearson Education India, 2015.
[182] F. Bai, S. Wang, X. Jia, Y. Guo, B. Yu, H. Wang, C. Lai, Q. Ren, and H. Sun, "A Low-Cost Reduced-Latency DRAM Architecture with Dynamic Reconfiguration of Row Decoder," *TVLSI*, 2022.
[183] M. A. Turi and J. G. Delgado-Frias, "High-Performance Low-Power Selective Precharge Schemes for Address Decoders," *TCAS-II*, 2008.
[184] H. Hassan, G. Pekhimenko, N. Vijaykumar, V. Seshadri, D. Lee, O. Ergin, and O. Mutlu, "ChargeCache: Reducing DRAM Latency by Exploiting Row Access Locality," in *HPCA*, 2016.
[185] H. Luo, T. Shahroodi, H. Hassan, M. Patel, A. G. Yaglikci, L. Orosa, J. Park, and O. Mutlu, "CLR-DRAM: A Low-Cost DRAM Architecture Enabling Dynamic Capacity-Latency Trade-Off," in *ISCA*, 2020.
[186] *4Gb DDR4 SDRAM Component: H5AN4G8NMFR-TFC*, SK Hynix, Mar. 2013, https://www.datasheets.com/part-details/h5an4g8nmfr-tfc-sk-hynix-inc-63813153.
[187] L. Orosa, A. G. Yağlıkçı, H. Luo, A. Olgun, J. Park, H. Hassan, M. Patel, J. S. Kim, and O. Mutlu, "A Deeper Look into RowHammer's Sensitivities: Experimental Analysis of Real DRAM Chips and Implications on Future Attacks and Defenses," in *MICRO*, 2021.
[188] E. Alkaldy, K. Navi, and F. Sharifi, "A Novel Design Approach for Multi-Input XOR Gate Using Multi-Input Majority Function," *AJSE*, 2014.
[189] M. J. Wirthlin, A. M. Keller, C. McCloskey, P. Ridd, D. Lee, and J. Draper, "SEU Mitigation and Validation of the LEON3 Soft Processor Using Triple Modular Redundancy for Space Processing," in *FPGA*, 2016.
[190] A. Jacobs, G. Cieslewski, A. D. George, A. Gordon-Ross, and H. Lam, "Reconfigurable Fault Tolerance: A Comprehensive Framework for Reliable and Adaptive FPGA-based Space Computing," *TRETS*, 2012.
[191] F. Siegle, T. Vladimirova, J. Ilstad, and O. Emam, "Mitigation of Radiation Effects in SRAM-based FPGAs for Space Applications," *CSUR*, 2015.
[192] R. E. Lyons and W. Vanderkulk, "The Use of Triple-Modular Redundancy to Improve Computer Reliability," *IBM Journal of Research and Development*, 1962.
[193] J. Bauer, M. Gruhn, and F. C. Freiling, "Lest We Forget: Cold-Boot Attacks on Scrambled DDR3 Memory," *Digital Investigation*, 2016.
[194] M. Gruhn and T. Müller, "On the Practicability of Cold Boot Attacks," in *ARES*, 2013.
[195] J. A. Halderman, S. D. Schoen, N. Heninger, W. Clarkson, W. Paul, J. A. Calandrino, A. J. Feldman, J. Appelbaum, and E. W. Felten, "Lest We Remember: Cold-Boot Attacks on Encryption Keys," *CACM*, 2009.
[196] C. Hilgers, H. Macht, T. Müller, and M. Spreitzenbarth, "Post-Mortem Memory Analysis of Cold-Booted Android Devices," in *IMF*, 2014.
[197] H. T. Lee, H. Kim, Y.-J. Baek, and J. H. Cheon, "Correcting Errors in Private Keys Obtained from Cold Boot Attacks," in *ICISC*, 2012.
[198] S. Lindenlauf, H. Höfken, and M. Schuba, "Cold Boot Attacks on DDR2 and DDR3 SDRAM," in *ARES*, 2015.
[199] T. Müller, A. Dewald, and F. Freiling, "AESSE: A Cold-Boot Resistant Implementation of AES," in *EUROSEC*, 2010.
[200] P. Simmons, "Security through Amnesia: A Software-Based Solution to the Cold Boot Attack on Disk Encryption," in *ACSAC*, 2011.
[201] R. Villanueva-Polanco, "Cold Boot Attacks on Bliss," in *LATINCRYPT*, 2019.
[202] S. F. Yitbarek, M. T. Aga, R. Das, and T. Austin, "Cold Boot Attacks are Still Hot: Security Analysis of Memory Scramblers in Modern Processors," in *HPCA*, 2017.
[203] S. Khan, D. Lee, Y. Kim, A. R. Alameldeen, C. Wilkerson, and O. Mutlu, "The Efficacy of Error Mitigation Techniques for DRAM Retention Failures: A Comparative Experimental Study," in *SIGMETRICS*, 2014.
[204] J. Liu, B. Jaiyen, Y. Kim, C. Wilkerson, O. Mutlu, J. Liu, B. Jaiyen, Y. Kim, C. Wilkerson, and O. Mutlu, "An Experimental Study of Data Retention Behavior in Modern DRAM Devices," in *ISCA*, 2013.
[205] J. Liu, B. Jaiyen, R. Veras, and O. Mutlu, "RAIDR: Retention-Aware Intelligent DRAM Refresh," in *ISCA*, 2012.
[206] M. Patel, J. S. Kim, and O. Mutlu, "The Reach Profiler (REAPER): Enabling the Mitigation of DRAM Retention Failures via Profiling at Aggressive Conditions," in *ISCA*, 2017.
[207] Trusted Computing Group, "TCG Platform Reset Attack Mitigation Specification," *TCG*, 2008.
[208] TrendForce, "DRAM Manufacturers Revenue Share Worldwide From 2011

to 2023, by the Third Quarter," https://www.trendforce.com/presscenter/news/20231204-11942.html, 2023.
[209] SAFARI Research Group, "PiDRAM Source Code," https://github.com/CMU-SAFARI/PiDRAM, 2022.
[210] J. S. Kim, M. Patel, H. Hassan, L. Orosa, and O. Mutlu, "D-RaNGe: Using Commodity DRAM Devices to Generate True Random Numbers with Low Latency and High Throughput," in *HPCA*, 2019.
[211] B. B. Talukder, J. Kerns, B. Ray, T. Morris, and M. T. Rahman, "Exploiting DRAM Latency Variations for Generating True Random Numbers," in *ICCE*, 2019.
[212] C. Keller, F. Gurkaynak, H. Kaeslin, and N. Felber, "Dynamic Memory-based Physically Unclonable Function for the Generation of Unique Identifiers and True Random Numbers," in *ISCAS*, 2014.
[213] S. Sutar, A. Raha, and V. Raghunathan, "D-PUF: An Intrinsically Reconfigurable DRAM PUF for Device Authentication in Embedded Systems," in *CASES*, 2016.
[214] C. Eckert, F. Tehranipoor, and J. A. Chandy, "DRNG: DRAM-Based Random Number Generation Using Its Startup Value Behavior," in *MWSCAS*, 2017.
[215] F. Tehranipoor, W. Yan, and J. A. Chandy, "Robust Hardware True Random Number Generators Using DRAM Remanence Effects," in *HOST*. IEEE, 2016.
[216] W. Xiong, A. Schaller, N. A. Anagnostopoulos, M. U. Saleem, S. Gabmeyer, S. Katzenbeisser, and J. Szefer, "Run-time Accessible DRAM PUFs in Commodity Devices," in *CHES*, 2016.
[217] M. S. Hashemian, B. Singh, F. Wolff, D. Weyer, S. Clay, and C. Papachristou, "A Robust Authentication Methodology Using Physically Unclonable Functions in DRAM Arrays," in *DATE*, 2015.
[218] F. Tehranipoor, N. Karimian, W. Yan, and J. A. Chandy, "DRAM-Based Intrinsic Physically Unclonable Functions for System-Level Security and Authentication," *VLSI*, 2016.
[219] S. Li, A. O. Glova, X. Hu, P. Gu, D. Niu, K. T. Malladi, H. Zheng, B. Brennan, and Y. Xie, "SCOPE: A Stochastic Computing Engine for DRAM-Based In-Situ Accelerator," in *MICRO*, 2018.
[220] T. A. Manning, "Apparatuses and Methods for Comparing Data Patterns in Memory," 2018, US Patent 9,934,856.
[221] F. Parveen, Z. He, S. Angizi, and D. Fan, "Hybrid Polymorphic Logic Gate with 5-Terminal Magnetic Domain Wall Motion Device," in *ISVLSI*, 2017.
[222] F. Parveen, S. Angizi, Z. He, and D. Fan, "Low Power In-Memory Computing Based on Dual-Mode SOT-MRAM," in *ISLPED*, 2017.
[223] F. Parveen, Z. He, S. Angizi, and D. Fan, "HielM: Highly Flexible In-Memory Computing using STT MRAM," in *ASP-DAC*, 2018.
[224] F. Parveen, S. Angizi, Z. He, and D. Fan, "IMCS2: Novel Device-to-Architecture Co-Design For Low-Power In-Memory Computing Platform using Coterminous Spin Switch," in *IEEE Trans. Magn.*, 2018.
[225] A. S. Rakin, S. Angizi, Z. He, and D. Fan, "PIM-TGAN: A Processing-in-Memory Accelerator for Ternary Generative Adversarial Networks," in *ICCD*, 2018.
[226] A. K. Ramanathan, G. S. Kalsi, S. Srinivasa, T. M. Chandran, K. R. Pillai, O. J. Omer, V. Narayanan, and S. Subramoney, "Look-Up Table Based Energy Efficient Processing in Cache Support for Neural Network Acceleration," in *MICRO*, 2020.
[227] S. H. S. Rezaei, M. Modarressi, R. Ausavarungnirun, M. Sadrosadati, O. Mutlu, and M. Daneshtalab, "NoM: Network-on-Memory for Inter-Bank Data Transfer in Highly-Banked Memories," in *CAL*, 2020.
[228] V. Seshadri, T. Mullins, A. Boroumand, O. Mutlu, P. B. Gibbons, M. A. Kozuch, and T. C. Mowry, "Gather-Scatter DRAM: In-DRAM Address Translation to Improve the Spatial Locality of Non-Unit Strided Accesses," in *MICRO*, 2015.
[229] A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, and V. Srikumar, "ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars," in *ISCA*, 2016.
[230] L. Song, Y. Zhuo, X. Qian, H. Li, and Y. Chen, "GraphR: Accelerating Graph Processing Using ReRAM," in *HPCA*, 2018.
[231] Y. Tian, T. Wang, Q. Zhang, and Q. Xu, "ApproxLUT: A Novel Approximate Lookup Table-Based Accelerator," in *ICCAD*, 2017.
[232] L. Wu, R. Sharifi, A. Venkat, and K. Skadron, "DRAM-CAM: General-Purpose Bit-Serial Exact Pattern Matching," in *CAL*, 2022.
[233] L. Xie, H. A. Du Nguyen, M. Taouil, S. Hamdioui, and K. Bertels, "Fast Boolean Logic Mapped on Memristor Crossbar," in *ICCD*, 2015.
[234] X. Xin, Y. Zhang, and J. Yang, "ROC: DRAM-Based Processing with Reduced Operation Cycles," in *DAC*, 2019.
[235] L. Yang, S. Angizi, and D. Fan, "A Flexible Processing-in-Memory Accelerator for Dynamic Channel-Adaptive Deep Neural Networks," in *ASP-DAC*, 2020.
[236] J. Yu, H. A. Du Nguyen, L. Xie, M. Taouil, and S. Hamdioui, "Memristive Devices for Computation-in-Memory," in *DATE*, 2018.
[237] J. T. Zawodny and G. E. Hush, "Apparatuses and Methods to Reverse Data Stored in Memory," 2018, US Patent 9,959,923.
[238] Y. Zha and J. Li, "Hyper-AP: Enhancing Associative Processing through a Full-Stack Optimization," in *ISCA*, 2020.
[239] H. Zhao, A. Goda, K. K. Parat, A. G. Mauri, H. Liu, T. Tanzawa, S. Yamada, and K. Sakui, "Apparatuses and Methods to Control Body Potential in Memory Operations," 2017, US Patent 9,536,618.