DATIS: DRAM Architecture and Technology Integrated Simulation

Shiyu Xia
Shanghai Jiao Tong University
Shanghai, China
xiashiyu@mail.sjtu.edu.cn

Chen Zhang*
Shanghai Jiao Tong University
Shanghai, China
chenzhang.sjtu@sjtu.edu.cn
*Corresponding Author

Guangyu Sun
Peking University
Beijing, China
gsun@pku.edu.cn

Guohao Dai
Shanghai Jiao Tong University
Shanghai, China
daiguohao@sjtu.edu.cn

Runsheng Wang
Peking University
Beijing, China
rswang@pku.edu.cn

Zhigang Ji*
Shanghai Jiao Tong University
Shanghai, China
zhigangji@sjtu.edu.cn
*Corresponding Author

Ru Huang
Peking University
Beijing, China
ruhuang@pku.edu.cn

***Abstract*—Recent advances in DRAM technologies and large-dataset applications in data centers make both academic and industrial researchers eager to explore DRAM's novel usage and cross-disciplinary DTCO (design and technology co-design) spaces, as illustrated by recent studies of the PIM (Processing-In-Memory) or RowHammer effect etc. This evolving landscape has created a pressing need for systematic testing and validation of those emerging DTCO studies. However, previous DRAM simulators have lacked joint modeling of device and architecture, impeding effective simulation of these DTCO designs. To address this gap, we introduce DATIS (DRAM Architecture and Technology Integrated Simulator), a tool that effectively connects architectural design and the complexities of DRAM technology. DATIS addresses two critical challenges: abstracting technology intricacies and establishing connections between architectural activities and device-level process structures. This versatile tool empowers researchers to unlock the latent capabilities of DRAM and provides manufacturers with a platform to experiment with new processes and architecture co-design. To the best of our knowledge, DATIS is the first DRAM simulator in academia that integrates architecture and technology modeling. We build DATIS upon Ramulator, a well-known open source DRAM simulator for architecture-level modeling, and thus it can support a wide range of DRAM specifications, including DDRx, LPDDR5, GDDR6, and HBM2&3 etc. Our experiments demonstrate DATIS's efficacy and precision through three compelling case studies, addressing pivotal facets of DRAM technology, including storage, reliability, and computation.**

***Index Terms*—DRAM, Simulation, DTCO, Architecture, Technology**

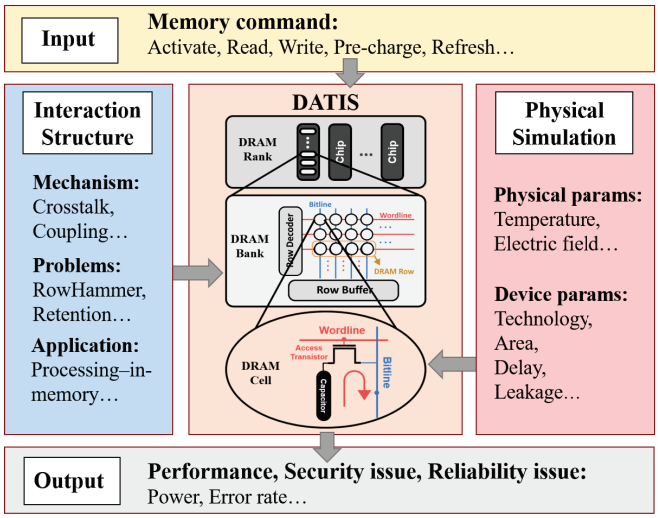


Fig. 1. Overview of the DATIS Framework.

I. Introduction

DRAM has long served as the cornerstone of main memory architecture. As DRAM's potentials and capabilities are increasingly harnessed, it is playing an ever more important role in data centers and large-dataset applications. Academic and industrial researchers have proposed various Design and Technology Co-Design (DTCO) methods aimed at optimizing system reliability [1], [2], accelerating application kernels [3]–[5], and enhancing architectural performance and energy efficiency [6]–[8].

One such innovation is Computing-in-Memory (CIM) [4], [9], which leverages the analog properties of DRAM technology to accelerate data center workloads. CIM enables the execution of logic operations like AND and OR entirely within the DRAM by utilizing charge sharing on memory bit-lines. However, due to manufacturing imperfections, variations in DRAM cells can result in non-uniform behavior across the chip [10]. These variations can lead to partial execution of operations or erroneous results. Another example is the RowHammer effect, a DRAM failure mechanism that occurs after frequent accesses to the same bit lines. This phenomenon is often attributed to substrate leakage and manufacturing defects. As DRAM technology scales and chip densities increase, these technology- and process-dependent mechanisms emerge, posing both opportunities and threats to DRAM operation.

However, DRAM simulators have struggled to keep pace with these rapid developments. While there have been several open-source DRAM simulators [11]–[14] developed for various memory architectures, they are typically built on standard constraints, which primarily model DRAM's system and

979-8-3315-3695-4/25/$31.00 ©2025 IEEE

architectural behaviors. Researchers often resort to *ad-hoc* modifications to simulate the aforementioned novel, undocumented operations or DTCO effects. Such simulations deviate significantly from real-world scenarios. This gap between architectural models and technology models hinders effective exploration and study of different aspects of the DRAM design space.

As a solution, we introduce DATIS, a DRAM Architecture and Technology Integrated Simulator, which is designed to comprehensively characterize the joint impact of architectural activities and technology features. Our aim is to provide a systematic methodology for testing and validating cross-disciplinary DTCO designs for various DRAM technologies, catering to both industrial evaluation and academic research.

Developing a scalable testing methodology involves two key challenges. First, an appropriate abstraction of technology nodes and manufacturing processes is crucial, as they significantly affect DRAM characteristics. DRAM cells typically consist of a selector transistor and a capacitor (1T-1C). Designs like STAR, RCAT, SaddleFin, and Vertical Channels [15]–[18] have been developed to improve performance. Researchers often use transistor and material models with TCAD simulations to capture physical properties, but TCAD is time-consuming and direct integration with DRAM simulators is impractical. To address this, we propose a concise set of parameters and data structures to represent DRAM cell properties, avoiding costly TCAD simulations.

Second, accurate modeling is needed to link architectural activities with DRAM cell geometry and layout. Manufacturers employ various geometries and connection structures, such as folded and open BL sensing schemes [19], as technology scales. To model this, we introduce graph-based data structures for DRAM cell arrays, where 1T-1C cells and their connections are represented by nodes, edges, and attributes. Simulating dynamic activities, such as activation, precharge, and read/write operations, uses an event-driven method, with cell charging or discharging modeled as messages passing through the graph's nodes and edges.

In summary, DATIS is a highly extensible DRAM simulator that offers an accurate performance model bridging architecture design and technology characteristics. It empowers researchers to explore DRAM's novel features and undocumented capabilities. Moreover, it provides DRAM manufacturers with the means to experiment with new processes or architecture co-design using system-trace-enriched stimuli.

This paper makes the following contributions:

- We present a novel, efficient performance model that describes key features and behaviors of device and geometry structures under various manufacturers' technology nodes and processes.
- We integrate DATIS with Ramulator, unlocking the capability of jointly simulating architectural activities and technology node effects based on the above modeling.
- We demonstrate the feasibility and effectiveness of DATIS through three representative applications, covering DRAM's storage features (leakage and retention time), security (RowHammer), and novel functions (computing-in-memory).

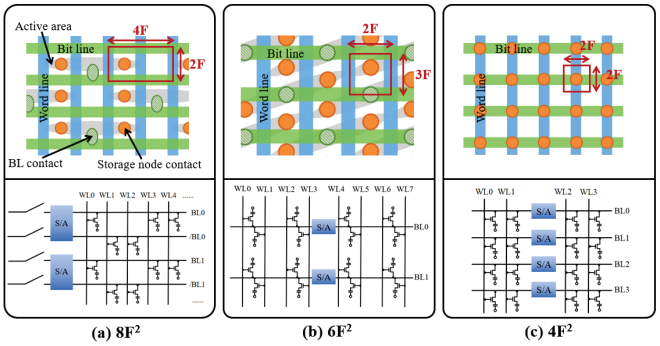


Fig. 2. DRAM process and geometry structures.

To the best of our knowledge, no existing DRAM simulator of this kind has been designed to comprehensively capture technology features and their joint interactions with architecture design.

II. Background and Motivation

In this section, we first provide a concise overview of DRAM architectures and technology advancements. We then review existing research on DRAM simulators. Finally, we present three notable cases that highlight the importance of close architecture and technology co-design.

A. DRAM Architecture and Structures

1) Architecture and Operation Basics: Modern DRAM systems feature a multi-level hierarchy. At the top level, the DRAM controller manages multiple channels, each with independent control signals. Channels consist of multiple ranks, and each rank contains several DRAM chips, which are divided into banks housing organized cell arrays. Before reading (RD) or writing (WR) to a DRAM cell, the corresponding wordline (WL) must be activated (ACT) to place data on the bitlines (BL). Precharge (PRE) is the process of preparing the memory cells for the next activation. Refresh (REF) is periodically applied on the memory row to counteract capacitor charge leakage and ensure data integrity over time.

2) Cell Layout.: DRAM cell layout has evolved to improve density and performance, as shown in Figure 2. Traditional DRAM uses an $8F^2$ folded sensing architecture, where adjacent bitlines share a sense amplifier (SA) and transistor drain [20]. With technology scaling, the $6F^2$ open bitline (BL) sensing scheme emerged, connecting one SA to two BLs [20], though shared active regions among nearby cells increase noise [21]. The latest $4F^2$ layout vertically aligns transistors and capacitors at wordline-bitline intersections [22], reducing BL capacitance and noise by eliminating shared drain nodes. These advancements aim to improve DRAM efficiency and performance.

3) Device Structures.: As technology nodes scale, DRAM cell structures have evolved, as shown in Figure 3. At the $8F^2$ node, the Recessed Channel Gate Transistor (RCAT) [16], developed from the Step-Gated Asymmetric (STAR) cell [15],

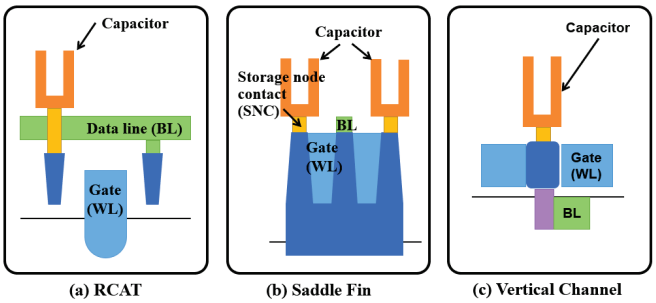


Fig. 3. Comparison of DRAM cell structures: (a) RCAT, (b) Saddle Fin, and (c) Vertical Channel.

improves both static and dynamic performance by introducing a recessed channel. The $6F^2$ node introduced the SaddleFin [17] structure, combining RCAT's recessed channel (length dimension) with FinFET's fin structure (width dimension), significantly extending retention time. Currently, the $4F^2$ node is shifting to Vertical Gate [22] cells, offering enhanced drive capabilities over earlier designs.

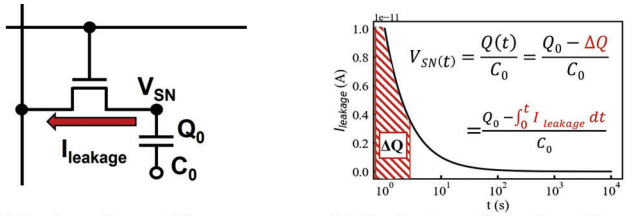
B. DRAM Simulators

Over the past decade, several DRAM simulators have been developed to evaluate main memory performance. USIMM [13] provides a basic DRAM simulation module, workloads, and metrics for scheduling algorithm development. DRAMSim2 [12] adds cycle-accurate simulation with a Verilog timing model, improving memory scheduling realism. DRAMSim3 [23] builds on this by addressing 3D packaging. Ramulator [11] emphasizes user-friendliness, supporting a wide range of standards with scalability and portability, making it a popular open-source simulator. However, as technology nodes shrink, device-level performance variations have become more significant, a factor not fully addressed by existing simulators. This gap limits optimization of current DRAM architectures and development of new storage systems.

C. DTCO Cases in DRAMs

The overall performance of DRAM, including factors like throughput and power efficiency, is often influenced by the combined effects of architectural activities and technology process. This often necessitates a co-optimization approach encompassing both architectural design and technology considerations.

1) Memory-related performance optimization: DRAM faces data corruption risks due to capacitor charge leakage, requiring periodic refresh commands. Previous research [8] shows that refreshing can reduce throughput by nearly 50% and account for over 40% of total power consumption at the 64 Gb density node. One example of DTCO with leakage is process-aware refreshing algorithm design to optimize memory access throughput and power consumption. Mutlu et al. [8] leverage retention time variations among DRAM cells, achieving significant power savings and throughput improvements with a new refresh strategy. As DRAM capacity and density increase, leakage challenges worsen, impacting reliability, operating frequency, and device design and so on.



(a) Leakage Current Diagram (b) The Leakage Curve Over Time

Mechanism	Types	Formulas
Generation-Recombination	GIDL	$I_{GIDL} = A_{GIDL} \exp\left(-\frac{E_a}{kT}\right)$
	GIJL	$I_{GIJL} = A_{GIJL} \exp\left(-\frac{E_a}{kT}\right)$
Drift-Diffusion	Sub-threshold	$I_{D\text{-}D} = A_{D\text{-}D} \exp\left(-\frac{E_a}{kT}\right)$
	P-N junction drift/diffusion	
Tunneling through oxide	Gate leakage	$I_{GATE\_LEAKAGE} = \sum_{i=1}^{n} \Delta I_{IT}(i)$
$I_{leakage} = I_{GIDL} + I_{GIJL} + I_{DD} + I_{GATE\_LEAKAGE}$		

(c) Table of Leakage Mechanisms and Corresponding Formulas

Fig. 4. Device Leakage Model.

2) RowHammer and security: RowHammer is a circuit-level interference where repeated access to memory rows within a single refresh cycle induces charge disturbances in adjacent cells, leading to bit-flip errors. Addressing RowHammer requires close DTCO collaboration, as it is tied to process technology, device structures, and circuit layouts. For instance, in the $6F^2$ geometry, two adjacent cells on the same bitline share an active region, as shown in Figure 2(b). A single sense amplifier (S/A) connects to two bitlines (BLs), and this shared configuration causes unpredictable charge fluctuations when neighboring rows are accessed, complicating bit-flip predictions.

3) Computing-in-memory: Computing-in-memory (CIM) has gained attention as a solution to memory-wall challenges, utilizing DRAM's analog properties for logic operations within main memory. However, this reliance on DRAM's analog behavior requires careful consideration of architectural and technological factors. Research shows that process variations and manufacturing imperfections can significantly impact charge levels and cell voltage. Performing CIM operations on partially charged cells during a refresh cycle can lead to unpredictable behavior or calculation failures. Therefore, close co-design of architecture and technology is essential to support advancements in CIM.

With these DTCO scenarios as motivation, we introduce our integrated simulation tool for architecture and technology validation and verification.

III. Modeling Method

In this section, we present our approach to constructing simulation models that bridge architectural activities with process and technology factors.

A. Device-level Analog Behavior

DRAM capacitors inherently experience charge loss over time, with static leakage emerging as a critical concern in

various DTCO scenarios, especially when analog characteristics, as discussed in Section II-C, become a factor. In such situations, the information stored in DRAM cells transcends the binary realm of 0s and 1s, instead existing as decimal values between 0 and 1. DRAM researchers often retrieve device performance parameters through complex TCAD simulations or semiconductor parameter analysis. However, direct integration of these methods into DRAM simulators is impractical due to unacceptable simulation time or the necessity of real chip testing [24].

To facilitate efficient simulation of static device models, we propose a set of streamlined abstractions designed to capture the analog attributes of DRAM cells. We denote the voltage across the capacitor as V_{SN}, which is determined by the amount of charge currently retained in the capacitor $Q(t)$ and its capacitance C_0. The V_{SN} decreases as the charge leaks out, a process governed by the leakage current I_{leakage}. Hence, this charge $Q(t)$ is calculated by subtracting the charge leaked over time from the initial charge Q_0. The leaked charge is obtained by integrating the leakage current I_{leakage} over time t,as illustrated in Figure 4(b). Many previous researches have revealed that the relationship between I_{leakage} and retention time t is very complex and usually related to vendor processes. For example, in a recent study [25], these mechanisms have been classified according to their origins: 1) Generation-Recombination (G-R) leakage arises from carriers generation within the depletion region; 2) drift–diffusion (D–D) leakage results from carriers movement in the non-depletion region; and 3) tunneling-induced leakage occurs as carriers traverse the dielectric layer.

In the proposed DATIS simulator, we provide an API for researchers to fill in their own formula of I_{leakage}, and we will do the integral calculation automatically.

To simplify and abstract the complex relationship between leakage current I_{leakage} and retention time t in simulations, we provide a general representation, as shown in Equation (1). In this representation, I_{leakage} consists of the sum of four components: I_{GIDL}, I_{GIJL}, I_{DD}, and $I_{\text{GATE\_LEAKAGE}}$. Each of these factors represents different leakage mechanisms: I_{GIDL} (Gate-Induced Drain Leakage), I_{GIJL} (Gate-Induced Junction Leakage), I_{DD} (Drift-Diffusion Leakage), and $I_{\text{GATE\_LEAKAGE}}$ (Gate Leakage). Users can customize the formulas for these four parameters respectively. For instance, some previous work [26], [27] proposed much more simplified representations, e.g. inversely proportional to the retention time t, as shown in Equation (2). While some other work [25] presents a more complicated formulation, as is defined in equation (1) and Figure 4(c). These different representations of I_{leakage} can all be represented and simulated in DATIS, which are customizable by user APIs.

$$I_{\text{leakage}} = I_{\text{GIDL}} + I_{\text{GIJL}} + I_{\text{DD}} + I_{\text{GATE\_LEAKAGE}} \quad (1)$$

$$I_{\text{leakage}} = \frac{\alpha}{t} \quad (2)$$

Moreover, process variations typically influence parameters such as C_0, E_a, and α, among others. Previous research reveals that these variations usually follow Gaussian distributions [28]. To assist users in simulating process variations, we support APIs that allow users to add customized process-dependent distributions to these parameters.

B. Geometry-level Interaction Model

There are mainly two types of interactions among DRAM cells, (1) charge sharing among capacitors and (2) crosstalks during circuit activities.

Charge sharing occurs when cells are activated for operations such as "row copy," where data is transferred between DRAM rows, or during the "read" operation, where data is moved to the row buffer. In CIM, logic operations are performed by activating multiple rows simultaneously, distributing charges among cells. Our simulation model is based on the capacitance equation $Q = C \times U$, where charge sharing causes the charge to distribute evenly, resulting in all capacitors sharing the same voltage, as shown in Figure 5.

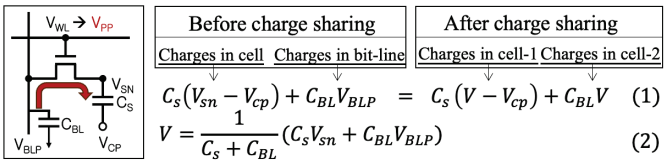


Fig. 5. Charge sharing models.

Inter-cell crosstalk shows complex leakage patterns influenced by activation modes, voltage distributions, and geometric structures, as illustrated in Figure 6. Architecturally, it is classified into single-sided and double-sided crosstalk. Geometrically, adjacent cells may share a common or distinct source node. At the device level, leakage behavior varies based on whether the cell stores a 1 or 0. Additionally, temperature and supply voltage also affect inter-cell crosstalk.

To model crosstalk between adjacent cells, we propose a graph-based intermediate representation (GIR), composed of nodes and edges, $G = \{N,\ E\}$, where each node ($n \subset N$) represents a DRAM cell and each edge ($e \subset E$) represents connections between neighboring cells, indicating potential crosstalk.

With the GIR abstraction, users can define DRAM processes in DATIS by specifying cell characteristics and interactions. For instance, Michael et al. [19] use a transmission line model to calculate wordline and bitline crosstalks, while Andrew et al. [29] propose an equation linking RowHammer leakage with crosstalk noise and temperature. Users can apply these functions to the GIR with appropriate parameters to model these effects. DATIS simulates cell array activities in three steps per clock cycle: (1) apply DRAM command traces on wordlines (e.g., ACT, READ), (2) scan the GIR to accumulate crosstalks around each cell, and (3) update cell statuses.

C. Aging model

Aging of the DRAM peripheral circuits poses an inevitable reliability threat to DRAM operations, such as the variable delays. These delays can lead to deviations in the timing issues, ultimately resulting in errors during DRAM read and

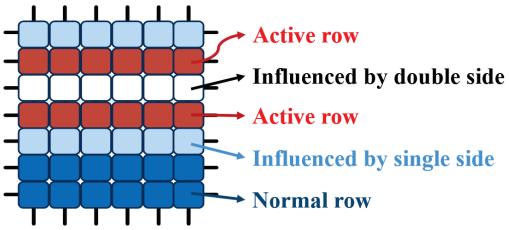


Fig. 6. Different crosstalk patterns.

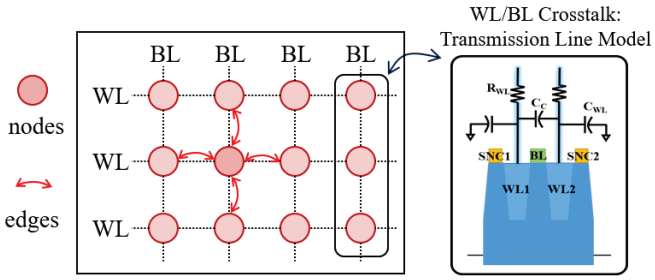


Fig. 7. Graph IR to model interaction crosstalks.

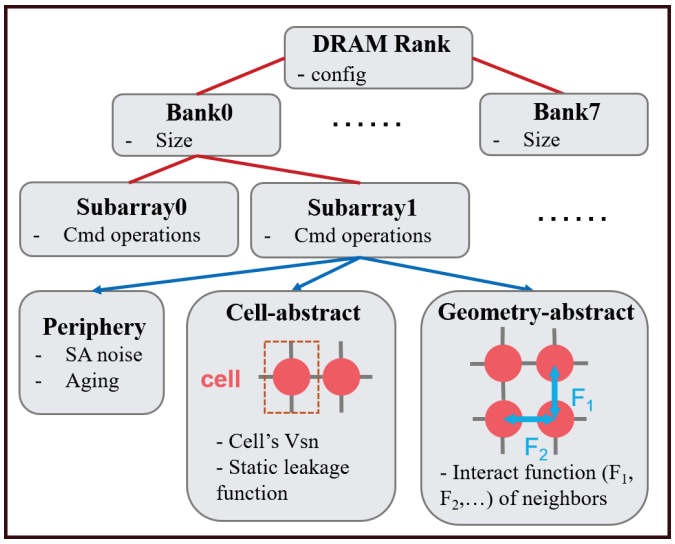


Fig. 8. Hierarchical Structure of DATIS.

write operations. To account for aging, the digital circuits are functionally modelled with its timing delay as aging-aware parameters. The analog circuits, such as the sensitive amplifier are modelled with its matching and noise. The aging of these circuits can be obtained through the circuit simulation [30]. Encompassing all these information into DATIS, we can accurately model and address the impact of aging on DRAM performance.

IV. Implementation Details

DATIS is built upon Ramulator [31], a popular open-source simulator that supports various DRAM standards, including DDRx, LPDDR5, GDDR6, and HBM2&3 etc. Likewise, DATIS follows a hierarchical structure mirroring the organization of DRAM, as is shown in Figure 8. User-configurable parameters are transmitted to their respective levels while the rank and bank layers construct the virtual structure of DRAM based on these parameters.

The subarray layer encompasses crucial command processing functions responsible for handling memory traces. These functions encompass prevalent DRAM commands, including PRE, ACT, RD, WR, REFA, REFB, and REFPR (targeted row refreshing). Notably, within the subarray layer, three models outlined in Section III are integrated: 1) cell-abstract, which records various static properties of array cells, including node voltage, static leakage, capacitors etc.; 2) geometry-abstract, representing interaction functions between individual cells and neighboring cells through equivalence; and 3) periphery, integrating non-ideal circuit-level models such as aging and noise.

V. Case Study

In this section, we demonstrate the practicality and effectiveness of the DATIS with three DTCO cases mentioned in the previous Section II-C.

A. Experimental Setup

Due to the confidentiality of DRAM device parameters, we encounter challenges in accessing device parameters used in previous papers or provided by manufacturers for our simulations. In this section, we utilize data from TCAD simulations, and we compare our simulation results with the phenomena reported in previous research papers. We conduct experiments using sub-20 nm node-sized Buried-Channel-Array-Transistor (BCAT) devices for the DRAM array transistors. The transfer characteristics (I_d-V_g) of these transistors were calibrated using testing data from real sub-20 nm process DRAM chips. The geometry is structured with a $6F^2$ layout, and the cell capacitance is set to 10 fF.

B. Static leakage and retention time

The static leakage model is complex due to the presence of multiple mechanisms. We adopt the three leakage current mechanisms and formulas mentioned in Figure 4. The crucial parameters required for these formulas were obtained through TCAD simulations and experimental data, consistent with [25]. We can draw two observations from Figure 9. First,at the cumulative probability distribution (CDF) of 0.5, the retention times corresponding to temperatures of 325K, 375K, and 425K are 4×10^5 ms, 8×10^3 ms, and 2×10^2 ms, respectively, which align with [25]. Second, it demonstrates the correlation between temperature and retention distribution. As temperature increases, retention time decreases, aligning with findings from previous studies [32].

C. RowHammer and DTCO optimizations

To mitigate storage failures caused by RowHammer effects, numerous studies [1], [10] have explored the relationships between refreshing intervals and RowHammer errors, and have proposed new refreshing algorithms [33], [34] to protect vulnerable cells. To assess DATIS's adaptivity to the RowHammer phenomenon, we conducted simulations of DRAM read and write operations while varying refresh intervals and monitoring DRAM security performance.

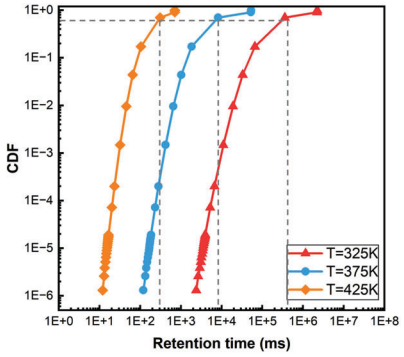


Fig. 9. Cumulative distribution function (CDF) of retention time under different temperatures.

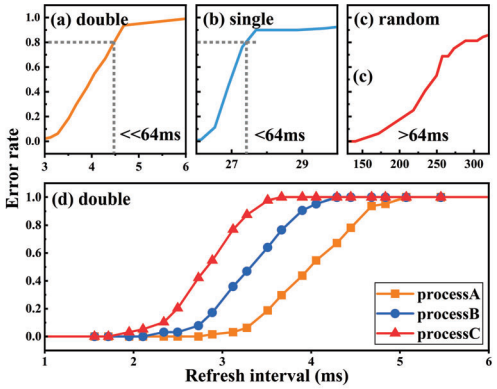


Fig. 10. Error rate with respect to varied access patterns(a-c) and processes(d).

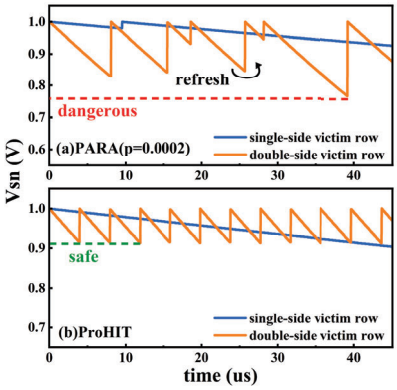


Fig. 11. Simulation of two refreshing algorithms for RowHammer.

We conduct simulations with three different patterns: (a) malicious attacks with single-side hammering, (b) malicious attacks with double-side hammering, and (c) a random pattern. As depicted in Figure 10(c), we observed a significant increase in the proportion of erroneous memory cells as refresh intervals extended. For the random pattern, a typical refresh interval of 64ms effectively maintained storage information within a secure range. However, in the case of extreme malicious attacks as shown in Figure 10(a)(b), ensuring that the DRAM error rate remains within an acceptable range necessitates significantly shorter maximum refresh intervals than 64ms. Notably, our simulations indicate that the retention time under double-side hammering is reduced by approximately 83% compared to single-side hammering, which is consistent with the previously reported reduction of 77% in another study [35]. This consistency further validates the reliability of our simulator's results depicted in Figure 10(a)(b). Furthermore, we conducted experiments with refreshing algorithms designed to mitigate RowHammer issues, showcasing how DATIS can facilitate DTCO studies. The PARA algorithm suggests refreshing neighboring rows after each row's activation based on a probability parameter. In contrast, ProHIT introduces a hot-cold table to track rows prone to RowHammer failures. As indicated by the experimental results in Figure 11 , it becomes evident that ProHIT significantly enhances DRAM security compared to PARA, particularly when facing severe attacks. This observation aligns with the findings in the referenced paper [34], which demonstrated that PRoHIT's combination of access history and probabilistic management effectively prevents row hammering by maintaining victim row voltages above the danger threshold, whereas PARA occasionally fails to do so.

D. Computing-in-memory

Computing-in-memory (CIM) in DRAM involves the simultaneous activation of multiple lines, utilizing capacitor charge sharing, and employing a sensor to enhance bit operations [4]. However, due to various leakage mechanisms, DRAM cells are not constantly maintained at full charge until the next refreshing cycle. Conducting CIM operations on cells that are not fully charged may lead to unexpected behavior or errors.

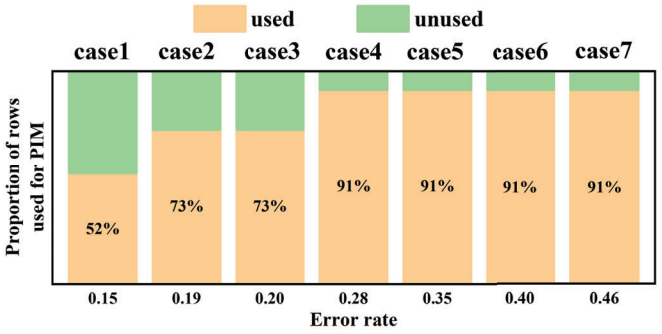


Fig. 12. Error rate under different cases.

Building upon the charge-sharing model, DATIS can simulate CIM operations and their failure mechanisms. DATIS supports a suite of CIM instructions, such as the AAP (ACT-ACT-PRE) and AP instructions. In our experiments, we conducted seven synthetic test cases involving millions of computational operations (ranging from 192M to 4992M) on a 128x128 DRAM array, using the same settings as in RowHammer experiments. These cases included: (1) one convolution, (2) one convolution and one multiplication, (3) two convolutions, (4) one convolution and two multiplications, (5) two convolutions and one multiplication, (6) two convolutions and two multiplications, and (7) two convolutions and three multiplications. Each case included intervals between operations to simulate storage time.

As shown in Figure 12, error rates rise from 0.15 to 0.46 as the computational complexity increases from case1 to case7. We observed that with the increase in computational complexity, both the error rate and the number of DRAM rows used also increase. This observation is consistent with the findings in a DDR4 chip [36], which show that the number of segments that support AND/OR operations significantly increases as BER(Bit Error Rate) increases, ranging from 0.03 to 0.5. This alignment underscores the reliability of our simulator, DATIS, in modeling CIM operations and validates the simulation results presented in our study.

VI. Conclusion

In summary, the pivotal role of DRAM in main memory architecture has inspired a wide range of DTCO investigations. To provide a user-friendly and systematic validation tool for these burgeoning DTCO studies in DRAM, we present DATIS, a DRAM Architecture and Technology Integrated Simulator. DATIS serves as a bridge between architectural design and technological characteristics, facilitating research into DRAM's innovative features and capabilities. Moreover, it aids manufacturers in the collaborative design of processes and architectures. We demonstrate the practicality and efficiency of DATIS through three key applications, encompassing DRAM's storage aspects (leakage and retention time), security considerations (RowHammer), and novel functionalities (computing-in-memory).

References

[1] O. Mutlu, "The rowhammer problem and other issues we may face as memory becomes denser," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*. IEEE, 2017, pp. 1116–1121.

[2] L. Cojocar, J. Kim, M. Patel, L. Tsai, S. Saroiu, A. Wolman, and O. Mutlu, "Are we susceptible to rowhammer? an end-to-end methodology for cloud providers," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 712–728.

[3] J. Ahn, S. Yoo, O. Mutlu, and K. Choi, "Pim-enabled instructions: A low-overhead, locality-aware processing-in-memory architecture," *ACM SIGARCH Computer Architecture News*, vol. 43, no. 3S, pp. 336–348, 2015.

[4] 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 *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, 2017, pp. 273–287.

[5] O. Mutlu, S. Ghose, J. Gómez-Luna, and R. Ausavarungnirun, "Processing data where it makes sense: Enabling in-memory computation," *Microprocessors and Microsystems*, vol. 67, pp. 28–41, 2019.

[6] H. Hassan, M. Patel, J. S. Kim, A. G. Yaglikci, N. Vijaykumar, N. M. Ghiasi, S. Ghose, and O. Mutlu, "Crow: A low-cost substrate for improving dram performance, energy efficiency, and reliability," in *Proceedings of the 46th International Symposium on Computer Architecture*, 2019, pp. 129–142.

[7] K. K.-W. Chang, D. Lee, Z. Chishti, A. R. Alameldeen, C. Wilkerson, Y. Kim, and O. Mutlu, "Improving dram performance by parallelizing refreshes with accesses," in *2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2014, pp. 356–367.

[8] J. Liu, B. Jaiyen, R. Veras, and O. Mutlu, "Raidr: Retention-aware intelligent dram refresh," *ACM SIGARCH Computer Architecture News*, vol. 40, no. 3, pp. 1–12, 2012.

[9] F. Gao, G. Tziantzioulis, and D. Wentzlaff, "Computedram: In-memory compute using off-the-shelf drams," in *Proceedings of the 52nd annual IEEE/ACM international symposium on microarchitecture*, 2019, pp. 100–113.

[10] 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," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2023.

[11] Y. Kim, W. Yang, and O. Mutlu, "Ramulator: A fast and extensible dram simulator," *IEEE Computer architecture letters*, vol. 15, no. 1, pp. 45–49, 2015.

[12] P. Rosenfeld, E. Cooper-Balis, and B. Jacob, "Dramsim2: A cycle accurate memory system simulator," *IEEE computer architecture letters*, vol. 10, no. 1, pp. 16–19, 2011.

[13] N. Chatterjee, R. Balasubramonian, M. Shevgoor, S. Pugsley, A. Udipi, A. Shafiee, K. Sudan, M. Awasthi, and Z. Chishti, "Usimm: the utah simulated memory module," *University of Utah, Tech. Rep*, pp. 1–24, 2012.

[14] M. K. Jeong, D. H. Yoon, and M. Erez, "Drsim: A platform for flexible dram system research," *Accessed in: http://lph. ece. utexas. edu/public/DrSim*, 2012.

[15] M. Jang, M. Seo, Y. Kim, S. Cha, Y. Kim, S. Kim, J. Rhee, J. Cheong, T. Jung, S. Pyi *et al.*, "Enhancement of data retention time in dram using step gated asymmetric (star) cell transistors," in *Proceedings of 35th European Solid-State Device Research Conference, 2005. ESSDERC 2005.* IEEE, 2005, pp. 189–192.

[16] J.-Y. Kim, C. Lee, S. Kim, I. Chung, Y. Choi, B. Park, J. Lee, D. Kim, Y. Hwang, D. Hwang *et al.*, "The breakthrough in data retention time of dram using recess-channel-array transistor (rcat) for 88 nm feature size and beyond," in *2003 Symposium on VLSI Technology. Digest of Technical Papers (IEEE Cat. No. 03CH37407)*. IEEE, 2003, pp. 11–12.

[17] S.-W. Park, S.-J. Hong, J.-W. Kim, J.-G. Jeong, K.-D. Yoo, S.-C. Moon, H.-C. Sohn, N.-J. Kwak, Y.-S. Cho, S.-J. Baek *et al.*, "Highly scalable saddle-fin (s-fin) transistor for sub-50nm dram technology," in *2006 Symposium on VLSI Technology, 2006. Digest of Technical Papers.* IEEE, 2006, pp. 32–33.

[18] S. Gautam, S. Maheshwaram, S. K. Manhas, A. Kumar, S. Sherman, and S. H. Jo, "Reduction of gidl using dual work-function metal gate in dram," in *2016 IEEE 8th International Memory Workshop (IMW)*. IEEE, 2016, pp. 1–4.

[19] M. Redeker, B. F. Cockburn, and D. G. Elliott, "An investigation into crosstalk noise in dram structures," in *Proceedings of the 2002 IEEE International Workshop on Memory Technology, Design and Testing (MTDT2002)*. IEEE, 2002, pp. 123–129.

[20] A. Spessot and H. Oh, "1t-1c dynamic random access memory status, challenges, and prospects," *IEEE Transactions on Electron Devices*, vol. 67, no. 4, pp. 1382–1393, 2020.

[21] T. Schloesser, F. Jakubowski, J. v. Kluge, A. Graham, S. Slesazeck, M. Popp, P. Baars, K. Muemmler, P. Moll, K. Wilson, A. Buerke, D. Koehler, J. Radecker, E. Erben, U. Zimmermann, T. Vorrath, B. Fischer, G. Aichmayr, R. Agaiby, W. Pamler, T. Schuster, W. Bergner, and W. Mueller, "6f2 buried wordline dram cell for 40nm and beyond," in *2008 IEEE International Electron Devices Meeting*, 2008, pp. 1–4.

[22] H. Chung, H. Kim, H. Kim, K. Kim, S. Kim, K.-W. Song, J. Kim, Y. C. Oh, Y. Hwang, H. Hong, G.-Y. Jin, and C. Chung, "Novel 4f2 dram cell with vertical pillar transistor(vpt)," in *2011 Proceedings of the European Solid-State Device Research Conference (ESSDERC)*, 2011, pp. 211–214.

[23] S. Li, Z. Yang, D. Reddy, A. Srivastava, and B. Jacob, "Dramsim3: A cycle-accurate, thermal-capable dram simulator," *IEEE Computer Architecture Letters*, vol. 19, no. 2, pp. 106–109, 2020.

[24] S. Jin, J.-H. Yi, J. H. Choi, D. G. Kang, Y. J. Park, and H. S. Min, "Prediction of data retention time distribution of dram by physics-based statistical simulation," *IEEE Transactions on Electron Devices*, vol. 52, no. 11, pp. 2422–2429, 2005.

[25] Y. Liu, D. Wang, P. Ren, J. Li, Z. Qiao, M. Wu, Y. Wen, L. Zhou, Z. Sun, Z. Wang, Q. Han, B. Wu, K. Cao, R. Wang, Z. Ji, and R. Huang, "Understanding retention time distribution in buried-channel-array-transistors (bcat) under sub-20-nm dram node—part i: Defect-based statistical compact model," *IEEE Transactions on Electron Devices*, pp. 1–7, 2024.

[26] D. Ha, C. Cho, D. Shin, G.-H. Koh, T.-Y. Chung, and K. Kim, "Anomalous junction leakage current induced by sti dislocations and its impact on dynamic random access memory devices," *IEEE Transactions on Electron Devices*, vol. 46, no. 5, pp. 940–946, 1999.

[27] A. Weber, A. Birner, and W. Krautschneider, "Retention tail improvement for gbit drams through trap passivation confirmed by activation energy analysis," in *2006 European Solid-State Device Research Conference*. IEEE, 2006, pp. 250–253.

[28] Y. Li, H. Schneider, F. Schnabel, R. Thewes, and D. Schmitt-Landsiedel, "Dram yield analysis and optimization by a statistical design approach," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 58, no. 12, pp. 2906–2918, 2011.

[29] A. J. Walker, S. Lee, and D. Beery, "On dram rowhammer and the physics of insecurity," *IEEE Transactions on Electron Devices*, vol. 68, no. 4, pp. 1400–1410, 2021.

[30] K. Kim, I. Chung, D. Sun, S. Rhe, I. Kim, H. Hwang, K. Cho, and G. Jin, "Study on off-state hot carrier degradation and recovery of nmosfet in swd circuits of dram," in *2016 IEEE International Integrated Reliability Workshop (IIRW)*, 2016, pp. 91–94.

[31] H. Luo, Y. C. Tuğrul, F. N. Bostancı, A. Olgun, A. G. Yağlıkçı, , and O. Mutlu, "Ramulator 2.0: A Modern, Modular, and Extensible DRAM Simulator," 2023.

[32] K. Y. Kim, K. K. Min, and B.-G. Park, "Trap-induced data-retention-time degradation of dram and improvement using dual work-function metal gate," *IEEE Electron Device Letters*, vol. 42, no. 1, pp. 38–41, 2020.

[33] M. Son, H. Park, J. Ahn, and S. Yoo, "Making dram stronger against row hammering," in *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2017, pp. 1–6.

[34] Y. Kim, R. Daly, J. Kim, C. Fallin, J. H. Lee, D. Lee, C. Wilkerson, K. Lai, and O. Mutlu, "Flipping bits in memory without accessing them: An experimental study of dram disturbance errors," in *2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*, 2014, pp. 361–372.

[35] L. Zhou, J. Li, Z. Qiao, P. Ren, Z. Sun, J. Wang, B. Wu, Z. Ji, R. Wang, K. Cao, and R. Huang, "Double-sided row hammer effect in sub-20 nm dram: Physical mechanism, key features and mitigation," in *2023 IEEE International Reliability Physics Symposium (IRPS)*, 2023, pp. 1–10.

[36] 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," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 12, pp. 5098–5112, 2023.