DARA: A Data-Aware Refreshing Architecture for HBM Power Reduction in LLM Accelerators

Jiajie Fan
Shanghai Jiao Tong University
Shanghai, China
fanjj666@sjtu.edu.cn

Lebei Cui
Shanghai Jiao Tong University
Shanghai, China
lbcui_2024@sjtu.edu.cn

Anmin Liu
Peking University
Beijing, China
anminliu@stu.pku.edu.cn

Xinyao Wang
Shanghai Jiao Tong University
Shanghai, China
pyrimidine@sjtu.edu.cn

Zhutianya Gao
Shanghai Jiao Tong University
Shanghai, China
zechariarh0825@sjtu.edu.cn

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

Abstract

HBM refresh overhead is becoming a major energy bottleneck in Large Language Model (LLM) inference. In 3D-stacked HBM, high thermal density aggravates DRAM charge leakage, forcing tighter refresh intervals and significantly increasing refresh power. The root cause is a semantic mismatch: JEDEC-style uniform refresh protects all stored bits equally based on worst-case cell retention, while LLM weights exhibit strong significance heterogeneity across both weight elements and bit positions. Existing schemes still fall short not only because they are either semantics-blind, coarse-grained, or retraining-intensive, but more fundamentally, there still lacks an interface that maps semantic significance to hardware-controllable refresh domains.

We propose **DARA, a data-aware refreshing architecture** for HBM power reduction in LLM accelerators. It combines a multi-tier significance model with an analytical reliability model to translate semantic error tolerance into safe refresh relaxation, and further introduces significance-aware remapping and a hardware-assisted runtime optimization framework to fully convert the significance heterogeneity of LLM weights into hardware-executable differentiated refresh. Across multiple LLMs and data formats, DARA reduces refresh power by 32.7%–87.8% while preserving near-original inference quality with negligible runtime and hardware overheads.

Keywords

High Bandwidth Memory (HBM), DRAM Refresh, Energy Efficiency

1 Introduction

The rapid scaling of Large Language Models (LLMs) has made High Bandwidth Memory (HBM) a critical yet increasingly power-hungry component of modern AI accelerators. In 3D-stacked HBM, high thermal density aggravates charge leakage in 1T1C DRAM cells and significantly increases refresh overhead: refresh energy can account for up to 50% of total DRAM energy consumption [16]. Moreover, rising temperature further tightens the refresh interval, which in turn increases heat generation, forming a vicious positive feedback loop and finally aggravating refresh power overhead. Our study shows that tightening the refresh period from 256 ms to 8 ms raises the refresh-power fraction from 4.3% to 83.0%. Therefore, refresh overhead has become a first-order bottleneck for energy-efficient LLM inference on HBM.

This inefficiency primarily stems from *the mismatch between uniform refresh policy and the significance heterogeneity of LLM weights*. JEDEC-style refresh uniformly protects all stored bits based on worst-case cell retention [8], while LLM weights are highly non-uniform in error sensitivity: at the element level, sparse or near-zero data are more error-tolerant, whereas at the bit level, exponent bits are much more sensitive than mantissa bits. As a result, uniformly protecting all weight bits can over-protect semantically resilient data and waste substantial refresh energy on regions whose corruption would have limited impact on inference quality.

Existing refresh optimizations still fall short. Prior schemes are either data-agnostic [2, 5, 10, 16], coarse-grained [11, 14, 17, 20], or retraining-heavy [11], and therefore cannot fully exploit the significance heterogeneity of LLM weights, especially across bit-planes. More fundamentally, conventional HBM layouts store all bits of a weight element in the same physical bank row, preventing differentiated refresh across data of different significance. Therefore, these limitations expose the core missing abstraction gap in prior work: to fully exploit the semantic significance heterogeneity of LLM weights requires not only a significance metric to identify critical data, but, more importantly, *an interface that maps semantic significance to hardware-controllable refresh domains*.

To fill this gap, we present **DARA, a data-aware refreshing architecture** that tightly couples semantic significance with hardware refresh domains through a cross-layer design. First, DARA establishes a unified modeling framework consisting of a multi-tier significance model across weight elements and bit positions, and a data reliability model that quantitatively translates semantic error tolerance to safe refresh relaxation. Second, DARA introduces a significance-aware remapping strategy that reorganizes weight bit-planes into independently refresh-able bank groups, enabling hardware-executable differentiated refresh. Third, DARA provides a hardware-assisted closed-loop optimization framework spanning significance modeling, reliability calibration, physical remapping, and runtime refresh control. Across multiple LLMs and data formats, DARA reduces refresh power by 32.7%–87.8% while preserving near-original inference quality with negligible data reading latency and hardware overheads.

*Corresponding author.

Conference acronym 'XX, San Jose, CA, USA
2026. ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/10.1145/3831252.3834263

To summarize, our key contributions are:

- **Significance and reliability modeling:** A unified modeling framework that quantifies data significance and translates semantic error tolerance into safe refresh relaxation.
- **Significance-aware remapping:** A strategy that reorganizes weight data into independently refresh-able bank groups, enabling hardware-executable differentiated refresh.
- **Hardware-assisted closed-loop optimization:** A memory-controller-assisted framework that enforces differentiated refresh at runtime, achieving **32.7%-87.8%** refresh-power reduction with negligible overheads.

2 Background & Motivation

2.1 Refresh Overhead in HBM-Based LLM Inference

HBM has become the default substrate for storing LLM weights due to the high capacity density and memory bandwidth enabled by its 3D-stacked architecture. However, this tightly integrated structure also creates high thermal density, causing the capacitors in 1T1C DRAM cells to suffer from accelerated charge leakage and substantially increasing refresh overhead. Prior studies report that refresh-dominated static power can account for 20%–50% of total memory power [7, 16, 17, 21]. More critically, as temperature rises, the refresh interval must be further tightened to preserve data reliability. This higher refresh activity in turn increases power consumption and heat generation, forming a vicious positive feedback loop, which exacerbate refresh power issue. For example, our study shows that when the refresh period shrinks from 256 ms to 8 ms, the fraction of DRAM power consumed by refresh rises sharply from 4.3% to 83.0%. Therefore, reducing refresh overhead is the first-order requirement for energy-efficient LLM inference on HBM.

2.2 Uniform Refresh Is Semantically Wasteful for LLM Weights

The inefficiency of conventional refresh stems from a fundamental semantic mismatch. JEDEC-style uniform refresh is designed to satisfy the worst-case cell-retention requirement and therefore assumes that all stored bits deserve the same level of protection. LLM inference, however, does not require uniform protection across all weights. Instead, *its accuracy depends on the non-uniform semantic significance of the stored data.* At the weight-element level, sparse or near-zero weights are inherently more error-tolerant than a relatively small subset of critical weights. At the bit level, modern numerical formats such as BF16 and FP16 introduce another layer of heterogeneity: exponent bits are substantially more sensitive than mantissa bits because they determine the magnitude scale of the value

As a result, different regions of LLM weight data exhibit significantly different sensitivities of errors. Applying one rigid refresh policy to all of them therefore over-protects semantically resilient data and wastes a substantial amount of refresh energy on regions whose corruption would have limited impact on inference quality. In this sense, *the mismatch between uniform refresh and data significance heterogeneity directly leads to substantial semantical refresh-energy waste.*

2.3 Why Existing Schemes Cannot Exploit Weight Heterogeneity

To mitigate DRAM refresh overhead, many refresh optimization techniques have been proposed. However, they still fall short for several reasons.

(1) Semantic-blind: Most prior works reduce over-refreshing using cell-centric properties such as retention profiling, temperature-aware refresh, ECC-assisted relaxation, and sub-bank scheduling [2–5, 10, 16]. While effective at exploiting device-level characteristics, these methods reason only about the physical state of memory cells rather than the semantic importance of the stored data. As a result, their data-agnostic strategy cannot exploit the significance heterogeneity of LLM weights.

(2) Coarse-grained: More schemes such as EXTREME, EDEN, and ADROIT incorporate application semantics, but their management granularity is typically limited to pages, layers, or data types. This granularity is too coarse to capture the strong heterogeneity within LLM weights, especially across bit positions [11, 14, 17, 20].

(3) Computation-expensive: Some schemes rely on extensive model retraining, imposing a prohibitive computational burden that makes them practically unfeasible for massive-scale LLMs [11].

More fundamentally, even if a new significance metric could identify the fine-grained heterogeneity within LLM weights, *a core abstraction is still missing to map data-significance heterogeneity onto hardware refresh domains.* Under the conventional word-centric mapping policy in HBM, all bits of a weight element are stored in the same physical bank row, making data with different semantic significance physically inseparable for refresh control. As a result, the hardware cannot assign different refresh intervals to semantically different data. Therefore, what is missing is not merely a better significance metric or a more accurate refresh model, but **an interface that maps semantic significance to hardware-controllable refresh domains**.

2.4 Design Requirements

To realize such an interface, three architectural requirements must be met: (a) a unified modeling methodology consisting of a multi-tier significance abstraction across weight elements and bit positions, together with an analytical data-reliability model that relates refresh interval, temperature, and failure budget; (b) a remapping strategy that maps logical data significance onto physical refresh domains; and (c) a closed-loop control and optimization framework that enforces differentiated refresh at runtime.

Figure 1 shows the overall architecture of DARA and illustrates how these requirements are addressed in a cross-layer manner. At the software layer, requirement (a) is addressed by the *Significance Profiler* and the *Refresh Policy Generator*, which are driven by two coupled analytical models (Section 3) to determine failure budgets, mapping configurations, and the initial refresh intervals for all Bank Groups. At the HW-SW interface layer, requirement (b) is addressed by the *Significance-aware Remapper*, which guides the *Data Slicing* process to partition and remaps weight data according to semantic-significance profiling results (Section 4). At the hardware layer,

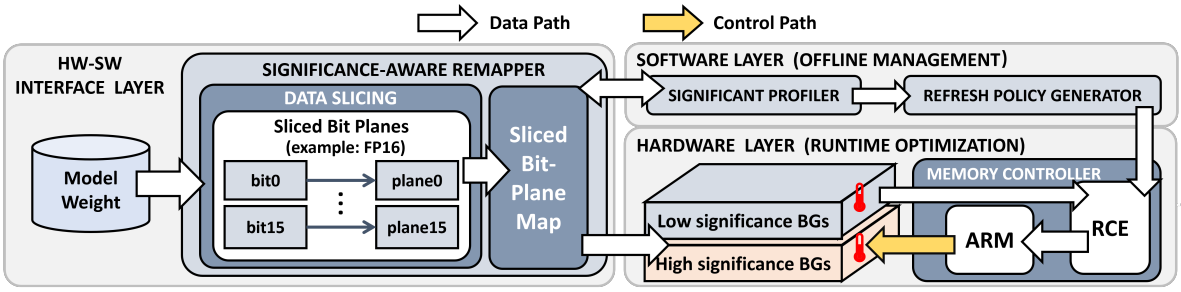

Figure 1: Overall architecture of DARA, illustrating the significance-aware cross-layer closed-loop optimization framework across the sensing, decision, and execution stages.

requirement (c) is mainly addressed by the *Real-time Calculation Engine (RCE)* and the *Adaptive Refresh Manager (ARM)* implemented in the memory controller, which compute and update the refresh interval in runtime for each Bank Group based on temperature monitoring, while preserving compatibility with conventional memory I/O protocols (Section 5). Together, these three cross-layer designs form a complete runtime closed-loop optimization framework.

3 Data Significance and Reliability Model

To quantify the significance heterogeneity of weight elements and different bits, we firstly establish two coupled analytical models. The multi-tire significance model quantifies the weight-element significance and bit sensitivity, whereas the analytical data reliability model quantifies the interplay between the refresh interval t, operating temperature T, and the resulting failure probability F, providing the physical bound for refresh relaxation.

3.1 Multi-Tier Significance Modeling

3.1.1 Weight Element-wise Significance. At the weight element level, we evaluate the criticality of individual weight elements using the Weight-and-Activation metric [22]:

$$S_{ij} = |w_{ij}| \cdot \|X_j\|_2 \tag{1}$$

where $\|X_j\|_2$ denotes the L_2-norm of the input activations. This metric identifies high-magnitude weights that coincide with large activation values as critical parameters, while sparse or low-activation weights are categorized as resilient regions.

3.1.2 Bit-wise Numerical Sensitivity. At the fine granularity, the numerical significance of each bit is dictated by its position within the data format. For floating-point formats (e.g., BF16 and FP16), the relationship between binary encodings and numerical values is defined by the IEEE 754 standard:

$$W = (-1)^s \cdot 2^{E-\text{bias}} \cdot \left(1 + \sum_{i=1}^{M} b_i \cdot 2^{-i}\right) \tag{2}$$

where s, E, and b_i represent the sign bit, exponent, and i-th mantissa bit, respectively. Any bit-flip in the exponent results in an exponential magnitude shift, whereas flips in the mantissa contribute only linearly, making the former far more critical. For integer formats

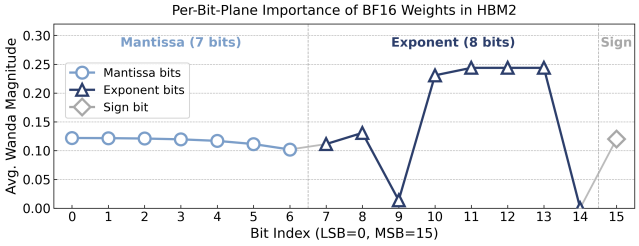

Figure 2: Bit-level significance profiling using Average Weight-and-Activation Magnitude (e.g., Qwen2.5-3B). Exponent bits (7–14) exhibit sharp significance heterogeneity compared to mantissa bits (0–6)

(e.g., INT8/4), the significance follows a linear weighted sum:

$$W = \text{scale} \cdot \left(-b_{N-1} \cdot 2^{N-1} + \sum_{i=0}^{N-2} b_i \cdot 2^i\right) \tag{3}$$

establishing a distinct hierarchy between Most Significant Bits (MSBs) and Least Significant Bits (LSBs).

We profile a pre-trained Qwen2.5-3B model using the Weight-and-Activation magnitude metric to confirm this heterogeneity. As shown in Fig. 2, mantissa bits (0–6) exhibit a flat, low importance profile, confirming their resilience. In contrast, exponent bits (7–14) show highly non-uniform importance. Notably, Bit 14 (E[7]) shows near-zero significance because weight magnitudes in Qwen2.5 are largely clustered within $|W| < 2.0$, keeping this bit consistently zero. The sign bit (15) remains critical as it dictates weight polarity. Ultimately, a memory region's importance is the product of its element-level criticality and its bit-level sensitivity, providing the theoretical foundation for DARA's isolated refresh control.

3.2 Analytical Data Reliability Modeling

We utilize the Weibull distribution to model the cumulative failure probability F of a DRAM cell due to charge leakage over time t:

$$F(t; k, \lambda) = 1 - \exp\left[-\left(\frac{t}{\lambda}\right)^k\right] \tag{4}$$

where k is the shape parameter reflecting the retention time distribution, and λ is the characteristic life.

The Arrhenius equation defines how λ scales with absolute temperature T:

$$\lambda(T) = \lambda_{\text{ref}} \cdot \exp\left[\frac{E_a}{k_B}\left(\frac{1}{T} - \frac{1}{T_{\text{ref}}}\right)\right] \tag{5}$$

By substituting Eq. 5 into Eq. 4, we derive the governing equation for the optimal refresh interval:

$$t(T,F) = \left[\lambda_{\text{ref}} \exp\left(\frac{E_a}{k_B}\left(\frac{1}{T} - \frac{1}{T_{\text{ref}}}\right)\right)\right] \cdot [-\ln(1-F)]^{1/k} \tag{6}$$

Our evaluation employs conservative physical constants to ensure robustness against Variable Retention Time (VRT) and process variations. Under identical conditions, our model is calibrated to predict a bit error rate (BER) 10^2 to $10^4\times$ higher than field measurements reported in the literature [12, 15]. Despite this relatively conservative settings, remarkable results were observed in experiments, which indicates parameter settings conforming to actual measurements will leads to more significant results.

4 Significance-aware Remapping Strategy

The analytical models in Section 3 provide the theoretical foundation for differentiated refresh. However, a key interface gap still remains between logical data significance and physical refresh domains. To bridge this gap, we propose a significance-aware data remapping strategy that translates logical error budgets into physical storage domains.

We first describe the *Data Layout and Physical Mapping* details, which answers what data regions should be remapped to differentiated refresh domains and how this remapping can be realized while preserving compatibility with multiple memory addressing policies. We then present the *Remapping configuration Flow*, which answers how to set the refresh interval t based on the operating temperature T and failure probability F.

4.1 Data Layout and Physical Mapping

To bridge the gap between semantic importance and physical control, DARA decouples bits with identical significance to form bit-planes $P_j = \langle b_{j,0}, \ldots, b_{j,K-1} \rangle$. These logical planes are mapped to HBM hardware during the HBM data-write phase.

The HBM2 organization considered in this work consists of eight physical channels, each divided into two pseudo-channels, resulting in 16 pseudo-channels in total. As illustrated in Figure 3, the standard storage model(left) utilizes word-centric interleaving where each channel burst delivers complete, multi-bit weights. In contrast, our bit-slicing model(right) clusters bits of the same significance index from multiple weights into dedicated channels. For example, a burst across 16 channels in HBM2, which traditionally provides 16 complete FP16 weights per cycle, is reorganized such that each channel carries a specific bit-plane for a larger set of 256 weights. It details this transition within a specific channel (e.g., CH0), which lays at the bottom of the figure and has four bank groups. The numbers represent the location from `CH0.BG0.BK0.ROW0.COL0` to `CH0.BG3.BK0.ROW0.COL1` and the order of the numbers represents the reading order in this channel. In the standard model, the data in the location `CH0.BG0.BK0.ROW0.COL0` contains a sequence of complete weights (a_1 to a_{16}). Through our remapping logic, this same physical location is transformed to store only a specific bit-plane

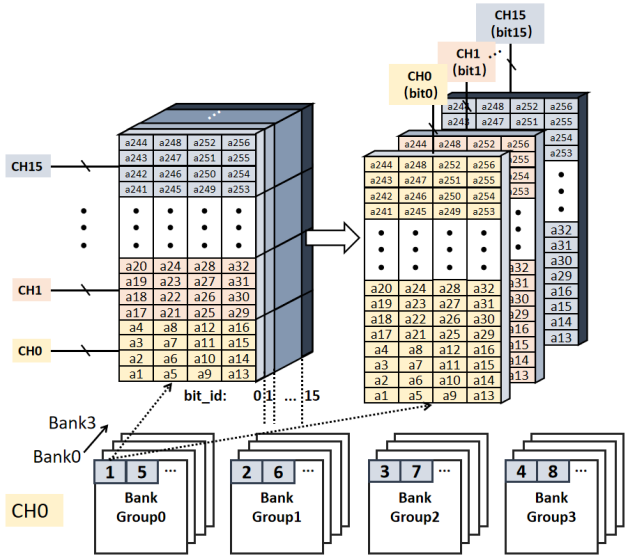

Figure 3: Transformation of data organization from standard word-centric interleaving (left) to significance-uniform domains (right) in HBM2.

(e.g., bit0) for the expanded weight set a_1 to a_{256}. The data stored in the same location but different channel_id is the different bits of the same weight.

In HBM systems, address mapping schemes define how logical addresses are translated into physical coordinates. The reading order in CH0, which is represented by eight numbers, implies a given address mapping. Assuming that the weights in the next parallel reading are complete a257-a512. In the bit-slicing storage model, the block marked with the number two should store the bit0 of a257-a512. While the data of this block may stored in different locations (such as `CH0.BG0.BK1.ROW0.COL0`) in other address mappings according to their different reading orders. It indicates that under different address mapping schemes, discrepancies in the internal traversal sequence within each channel cause the same physical bank group to store distinct data content, even though the data retrieved in a single parallel burst remains identical and the required logic transformations are correctly applied.

The transformation from standard storage model to bit slicing storage model is implemented purely through modifications to the address mapping logic and is adaptable to any address mapping schemes, requiring no structural alterations to the HBM's native I/O interface protocols, such as bit-width, burst length, or physical storage architecture. By isolating bit-planes into separate Bank Groups (BGs), DARA establishes the physical foundation for independent, significance-aware refresh control while maintaining strict data correctness and satisfying inference latency requirements.

4.2 Remapping Configuration Flow

After bit-slicing address remapping, we use a two-phase searching algorithm to establish the refresh intervals for each bank group. It follows a coarse-to-fine search strategy, as detailed in Algorithm 1, to maximize power savings while strictly adhering to a global perplexity (PPL) target π_{tgt}.

Algorithm 1: Adaptive HBM2 Per-BG Refresh Interval Search

Input: Model $\mathcal{M}$, PPL target π_{tgt}, refresh range $[t_{\min}, t_{\max}]$
Output: Per-BG refresh intervals $\{t_k\}$

$\{s_k\} \leftarrow$ Weight-and-Activation Score($\mathcal{M}$) ;
// Importance metric for each Bank Group
Sort BGs $g_1, \ldots, g_K$ by s_k ascending ; // Relax least-sensitive BGs first
// Phase 1: Bit-plane Level Coarse Search
foreach *bit-plane* b **do**
  $t_b \leftarrow$ BinarySearch($t_{\min}, t_{\max}$);
  Inject errors at rate $\epsilon(t)$ on bit b of **all** BGs;
  Find max t s.t. PPL($\mathcal{M}$) $\le \pi_{\text{tgt}}$;
// Phase 2: Per-BG Fine-grained Relaxation
$t_k \leftarrow t_{b(k)}$ for all k ; // Initialize with Phase 1 results as lower bound
for $k = 1$ **to** K **do**
  $t_k \leftarrow$ BinarySearch($t_k, t_{\max}$);
  Inject errors into $g_1, \ldots, g_k$ simultaneously using current $\{t_1, \ldots, t_k\}$;
  Find max t_k s.t. PPL($\mathcal{M}$) $\le \pi_{\text{tgt}}$;
return $\{t_k\}$;

Phase 1: Bit-plane Level Coarse Search. Phase 1 determines the maximum error-tolerance budget (F) for each bit-plane. Guided by the reliability model, we inject uniform retention errors into bit-planes to empirically link hardware-level failures to software-level accuracy. Then, for each bit-index b, the algorithm performs a binary search to identify the maximum refresh interval t_b that maintains the PPL within π_{tgt}. The resulting intervals $\{t_b\}$ provide a baseline reliability floor for the significance-uniform **partitions** established by our mapping strategy, effectively translating the abstract importance of bit-planes into concrete failure budgets for real-time hardware execution.

Phase 2: Per-BG Fine-grained Relaxation. Phase 2 refines the refresh policy by exploiting the non-uniform importance of weights within the same bit-plane. We first calculate the Weight-and-Activation Score ($S = |W| \cdot \|X\|_2$) to quantify the contribution of each weight tensor to the output. BGs are then ranked in ascending order of their mean Weight-and-Activation scores, allowing the algorithm to prioritize the relaxation of less critical physical regions.

Starting from the Phase 1 bounds ($t_k = t_{b(k)}$), we iteratively relax the refresh interval for each BG. Crucially, we employ cumulative injection: for each candidate t_k, we formulate data retention errors through data reliability model, and simultaneously inject them into the current BG and all previously optimized BGs ($g_1, \ldots, g_{k-1}$). This ensures that the aggregated error floor of the entire HBM stack remains below the accuracy threshold, effectively squeezing the maximum power reduction out of non-critical weights.

4.3 Deployment and Online Execution Flow

During the software-driven initialization phase, the model's significance profile is translated into a set of failure budgets F_k assigned to individual Bank Groups (BGs). These budgets represent the error-tolerance floor of each data domain and are deployed onto the internal registers of the memory controller. Once configured, the budget of each BG remains fixed throughout the model's operational lifetime to ensure predictable inference accuracy.

During real-time inference, these budgets are further leveraged by the hardware-assisted closed-loop optimization framework together with runtime thermal monitoring, enabling the system to fully exploit semantic heterogeneity in real HBM hardware.

5 Hardware-assisted closed-loop optimization

To leverage the mentioned remapping strategy, we further propose a memory-controller-assisted closed-loop optimization to alter the refresh interval of each bank group according to the real-time temperature. A data access and reconstruction process is also proposed to preserve compatibility with conventional memory I/O protocols.

5.1 Significance-Aware Memory Controller

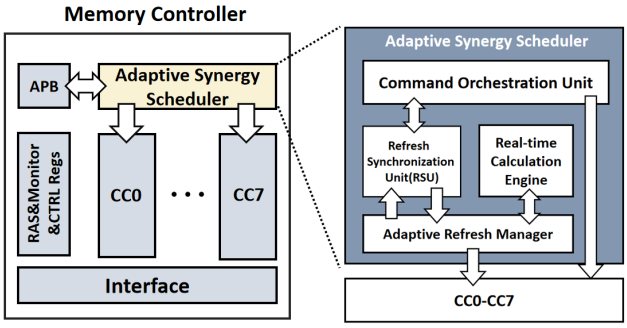

Figure 4: Architecture of the SAMC in HBM2. Eight physical-channel controllers(CC0–CC7) each serve two pseudo-channels, while the ASS coordinates differentiated refresh and transparent bit-plane mapping across all 16 pseudo-channels.

To support the runtime optimization strategy, we modified memory controller in HBM2 (denoted as SAMC), incorporating the Adaptive Synergy Scheduler (ASS) to bridge the gap between software-level significance profiling and hardware-level refresh execution.

As shown in Figure 4, the ASS is integrated into the control path of the memory controller, interacting with the host interface and the 8 parallel physical channel controllers (CC0–CC7). To ensure that significance-aware decisions are executed with minimal impact on memory bandwidth, the ASS is organized into four primary functional units:

- **Command Orchestration Unit (COU):** This unit manages the top-level command flow and host request queuing. It features a Weight Slicing/Reassembly Unit (WS/RU) that slice complete weights during the data-write phase in HBM and reassemble bit-fragments from the parallel channels. Furthermore, it orchestrates the broadcasting of commands across all parallel channels, while effectively differentiating and scheduling standard memory requests alongside those in bit-slicing mode.
- **Real-time Calculation Engine (RCE):** The RCE serves as the computational core that executes the thermal-scaling component

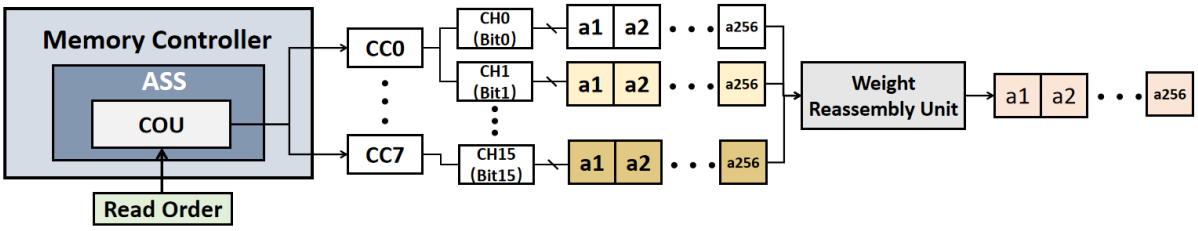

Figure 5: The data path of DARA in HBM2. During reads, the hardware reassembles bit fragments into weights in their original format (BF/FP16 shown), maintaining functional transparency.

of the analytical reliability model. Instead of solving full Weibull-based equations in real time, it applies sensed thermal gradients to pre-loaded baseline refresh intervals (*c_bg*) derived during offline profiling. By leveraging a high-performance 11-stage Arrhenius calculation pipeline, the RCE dynamically scales these intervals to provide precise, per-BG refresh updates with minimal logic overhead.

- **Refresh Synchronization Unit (RSU):** Implements synchronization window logic to monitor refresh events in critical MSB channels and orchestrate aligned refresh operations for resilient LSB domains. This ensures strict adherence to JEDEC timing constraints while maximizing power savings through coordinated execution.
- **Adaptive Refresh Manager (ARM):** The ARM serves as the execution bridge to the channel controllers. It utilizes a coordinated command orchestration and a priority encoder to manage an array of per-BG programmable counters. By translating RCE decisions into localized refresh signals, the ARM ensures that critical bits are protected while aggressively extending the intervals for error-tolerant memory regions.

DARA primarily modifies the memory controller through the RCE, ARM, RSU, and COU. The assumed HBM-stack support is limited to TAR-style thermal sensing and region-level differentiated-refresh execution, following prior work [6].

5.2 Closed-loop Optimization Framework

5.2.1 Adaptive Refresh Policy Determination. The runtime refresh-interval decision-making is the core process, which is primarily centered on the Real-time Calculation Engine (RCE). This unit bridges the gap between high-level data semantics and low-level physical constraints by integrating real-time thermal sensing data with offline-derived significance configurations.

The first step is real-time thermal sensing at the Bank-Group granularity. Specifically a digital temperature sensor, reported in prior work [6], is integrated at the geometric center of each physical bank group to provide real-time thermal monitoring for the HBM2 architecture. These sensors generate 7-bit thermal readings that cover a range of $0 - 105^{\circ}C$ with $1^{\circ}C$ resolution. At the second step, the RCE aggregates temperature zone indicators directly from the BG-level sensors and applies them to the pre-programmed baseline refresh intervals (*c_bg*). Instead of implementing a full, computationally expensive solver for the Weibull reliability equations on-chip, the RCE executes a high-performance 11-stage Arrhenius calculation pipeline to determine the thermal scaling factor in real time. At the final step, this factor is then multiplied by the baseline intervals to dynamically compute the optimal t_{REFI} for each Bank Group.

This architecture design ensures that refresh frequencies are precisely calibrated to current thermal gradients while maintaining a low hardware footprint by offloading complex significance budgeting to the offline profiling stage.

5.2.2 Refresh Execution. With specific refresh strategies, the physical assignment and execution of differentiated refresh commands is required, which is orchestrated by the Adaptive Refresh Manager (ARM).

To facilitate this granular control, the standard global refresh counter is replaced with an array of per-BG programmable counters. The ARM continuously translates timing decisions from the RCE into hardware thresholds, utilizing a priority encoder and a sticky-request handshake mechanism to handle concurrent refresh demands across different bank groups. By employing a signed difference logic for time comparison, the ARM ensures robust operation even across counter wrap-around events. When a BG-specific counter reaches its programmed threshold, a localized refresh operation is triggered independently and remains pending until a hardware acknowledgment is received from the channel controller. DARA changes only the selected refresh region and the programmed interval; each issued refresh still follows the standard channel-controller timing and arbitration constraints.

This mechanism is the key to minimizing HBM static power, as it allows the hardware to maintain high-frequency protection for critical exponent bit-planes while aggressively extending the intervals for error-tolerant mantissa regions.

5.2.3 Data Access and Reconstruction. The bit-slicing storage format fundamentally alters the conventional memory access pattern by isolating weight bits into significance-uniform domains across physical channels. While a standard HBM burst delivers complete weights from a single channel, DARA requires each channel to retrieve a sequence of bit fragments belonging to a specific bit-plane. To preserve compatibility with conventional memory I/O protocols, the hardware must perform real-time bit reorganization to reconstruct the original weight values before they reach the host.

As illustrated in Figure 5, when a read order is dispatched, the Command Orchestration Unit (COU) coordinates parallel data retrieval across all 16 pseudo-channels. The Weight Slicing/Reassembly Unit (WS/RU) then utilizes a combinational wiring matrix to aggregate these fragments back into the original BF/FP16 format through

a deterministic five-stage synchronous pipeline. This reconstruction process, integrated with address-based routing logic, ensures that the significance-aware memory organization remains entirely transparent to the LLM accelerator while maintaining compatibility with standard storage patterns at a negligible latency overhead. During writes, the hardware performs the inverse operation by slicing BF/FP16 weights into bit fragments and distributing them across the corresponding pseudo-channels.

6 Evaluation

In this section, we present a comprehensive evaluation of DARA to validate its effectiveness across three primary dimensions: refresh energy efficiency, inference performance, and hardware implementation cost. Our assessment aims to demonstrate that by leveraging the significance heterogeneity of LLM weights, DARA can achieve substantial power savings with negligible impact on model accuracy and system latency.

We benchmark our proposed architecture against standard JEDEC-compliant HBM2 baselines using representative LLM workloads to verify its practical viability in production-scale AI inference scenarios. To ensure data reliability, the maximum refresh interval is capped at 5000 ms, following prior work [15].

6.1 Refresh Energy Efficiency and Reliability Analysis

6.1.1 Experimental Methodology. We evaluate DARA using DRAMSim3 [13] configured with HBM2 4-Hi specifications at 85°C and 90°C. The workloads include Qwen2.5-3B [24], Phi-3-mini [1], and Gemma-2-2B [23] in BF16, FP16, INT8, and INT4 formats. At 85°C, the BF16,FP16,INT8,INT4 baseline powers are 249.2, 249.2, 124.6, 83.1mW for Qwen, 249.2, 249.2, 142.4, 71.2mW for Phi-3, and 249.2, 249.2, 124.6, 62.3mW for Gemma-2. The 90°C baselines use the temperature-dependent refresh setting in Fig. 6. We additionally evaluate Qwen2.5-3B under three address mappings: RoRaBgBaChCo, RoBaBgRaChCo, and ChRoBaBgRaCo, denoted AMS1–AMS3.

6.1.2 Results and Analysis. DARA reduces refresh energy by 32.7%–87.8% compared to the JEDEC standard baseline, as shown in Fig. 6.

Fig. 6(a) illustrates that at 90°C, Qwen2.5-3B achieves peak savings of 87.8% in FP16 and 86.7% in BF16, while Gemma-2-2B reaches 85.0%. Phi-3-mini achieves approximately 56.0% savings in BF16, FP16, and INT8; its similar results across formats arise from consistently high numerical sensitivity and limited inter-BG variation.

Fig. 6 (b) presents the corresponding energy efficiency across three representative address mapping schemes. These results confirm that the proposed refresh relaxations translate into substantial energy reductions, with the `rorabgbachco` mapping achieving a peak saving of 87.8% under 90°C. These experimental results underscore that our proposed scheme remains consistently effective across various address mapping configurations.

To ensure the data reliability, we set initial refresh intervals under the constraint of a maximum 0.1 increase in perplexity (PPL). We also employ conservative physical constants to ensure robustness against variable retention time and process variations, which lead to a predicted bit error rate (BER) 10^2 to $10^4\times$ higher than field measurements [12, 15]. It indicates that more remarkable results will be observed under parameter settings which conform to actual measurements. We also evaluate robustness under input variation. DARA derives the refresh-budget table from profiling prompts and then applies the same fixed error pattern to unseen prompts without re-optimizing the budgets. On WikiText2, PPL changes from 11.6096 to 11.6255 (+0.0159); on MMLU-Pro, PPL changes from 4.8390 to 4.9620. For OpenBookQA, where PPL is not the task metric, multiple-choice accuracy changes from 32.01% to 31.59%. These results suggest that fixed significance budgets remain stable across held-out inputs, while deployment mismatch can be mitigated by conservative intervals or periodic re-profiling.

Table 1: Weight-block read performance of DARA and the baseline at 85 °C. Mixed Qwen sweeps vary independent 64-B normal reads. Latency is in HBM2 cycles (t_{CK} = 0.5 ns), and bandwidth includes both payload types.

Model-trace block latency			
Model	Average (DARA/Baseline)	P95 (DARA/Baseline)	P99 (DARA/Baseline)
Qwen2.5-3B	**25.13**/31.45	**47**/**47**	**53**/62
Gemma2-2B	**28.42**/28.62	55/**41**	136/**52**
Phi3Mini-3.8B	**26.80**/28.62	47/**41**	57/**52**

Qwen mixed-request average block latency				
Independent 64 B normal reads	0	14K	42K	84K
Baseline	29.28	32.15	32.57	32.58
DARA	**24.29**	**25.37**	**26.45**	**26.67**

Qwen mixed-request effective bandwidth (B/cycle)				
Independent 64 B normal reads	0	14K	42K	84K
Baseline	207.25	205.08	200.13	194.00
DARA	**248.11**	**244.29**	**236.21**	**225.66**

6.2 Read Overhead Analysis

6.2.1 Experimental Methodology. We evaluate DARA against a conventional BF16 baseline using Ramulator2 [18], extended with custom memory-system plugins for an HBM2 device with 16 pseudo-channels. PyTorch forward hooks generate 20 M-request traces from Qwen2.5-3B, Gemma-2-2B, and Phi-3-mini. We report average, P95, and P99 weight-block latency and effective bandwidth. Each logical 1024 B weight block contains sixteen consecutive 64 B cachelines, issued as conventional word-centric reads by the baseline and coordinated as bit-sliced accesses across all 16 pseudo-channels by DARA. For mixed traffic, the latency and bandwidth sweeps fix 200K and 119,616 weight-block reads, respectively.

6.2.2 Results and Analysis. Table 1 summarizes the simulation results.

Block latency is the decisive metric for compute performance. DARA outperforms the baseline under both single and mixed request streams. DARA reduces it by compressing the controller-side request-issue window. Specifically, the baseline issues conventional cacheline reads over 15 ticks. Under the standard word-centric layout, the baseline cannot use the same fanout semantics: broadcasting one baseline address would fetch unrelated cachelines rather

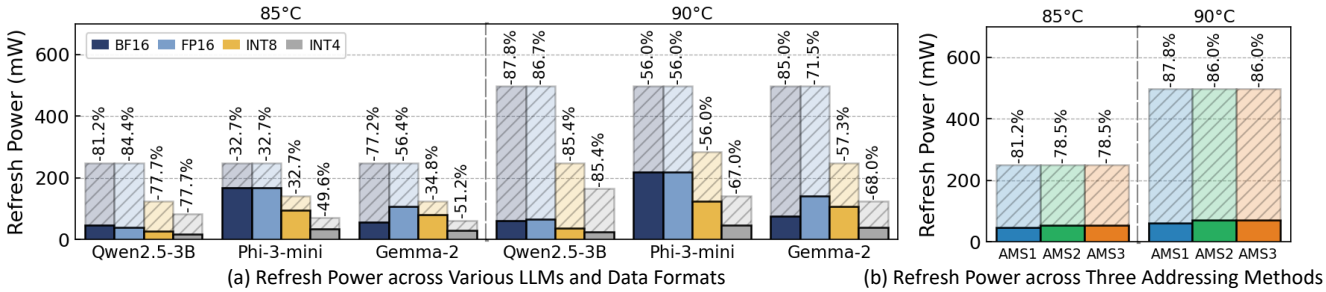

Figure 6: (a) Refresh power consumption of DARA across different LLM models, data formats (85°C,90°C) (b) Refresh power consumption of DARA across three address mapping methods (85°C,90°C)

Table 2: ASS area breakdown under NanGate45.

Block / structure	Cells	Area (μm^2)	Share of ASS
RCE (incl. TCE)	14,158	20,648	42.70%
ARM	9,945	14,504	30.00%
RSU	2,777	4,050	8.38%
COU	6,272	9,147	18.92%
ASS total	33,152	48,349	100%
Fine-grained structures			
Budget/interval registers	1,964	5,568	11.52%
COU request/pending queues	897	2,204	4.56%
Bit-slicing/reassembly + write swizzling	7,024	7,534	15.58%
Weight-region detection/fanout	22	26	0.05%

than the bit-planes of the same weight tile. In contrast, DARA internally fans out one bit-sliced weight-region read into sixteen pseudo-channel reads, completing the 16-channel broadcast in 2 ticks. DARA exhibits higher P99 latency in two of the three workloads because the DARA chain must wait for the slowest of 16 channels, making it sensitive to occasional long refresh stalls that do not block independent requests in the baseline.

Bandwidth is another important metric. Although each DARA logical-block read completes only after all 16 pseudo-channel responses return, the shorter controller-side issue window provides a net effective-bandwidth improvement in all evaluated mixed-request configurations.

6.3 Hardware Overhead Analysis

6.3.1 Experimental Methodology. The Adaptive Synergy Scheduler (ASS) was synthesized using Yosys [19] 0.63 targeting the NanGate45 45nm library. For area comparison, we use LiteDRAM [9] controller as baseline, which is synthesized by the same tool chain and library. Power is estimated at 1 GHz from a 276K-cycle Qwen VCD trace using Liberty internal-power and leakage data, with an estimated accuracy of ±50%.

6.3.2 Results and Analysis. As shown in Table 2, the ASS occupies **48,349 µm²**, which is equivalent to 6.6% of the synthesized 16-controller LiteDRAM baseline.

The LiteDRAM instance serves as a representative single-channel controller baseline. Because HBM pseudo-channels exhibit functional behaviors analogous to standalone DDR channels, we calculate the total HBM baseline area as 16× the area of a standalone

Table 3: ASS power overhead relative to adaptive refresh power savings (1.126 mW at 1 GHz).

Model	BF16	FP16	INT8	INT4
Qwen2.5-3B	0.56%	0.54%	1.16%	1.74%
Phi-3-mini	1.38%	1.38%	2.42%	3.19%
Gemma-2	0.59%	0.80%	2.60%	3.53%

LiteDRAM unit. The ASS is shared across all 16 pseudo-channels, avoiding the need to replicate DARA-specific scheduling and remapping logic for each pseudo-channel. Since both ASS and LiteDRAM are synthesized under identical technology assumptions, the relative area ratio (6.6%) represents an intrinsic architectural advantage that remains robust across process nodes.

Power consumption was estimated by extracting switching activity from 276 K-cycle Qwen2.5-3B VCD traces. At 1 GHz, the ASS consumes **1,126 µW**, with leakage dominating (89%) due to the low refresh-triggered duty cycle (≈ 0.24%).

Table 3 compares this overhead against refresh power savings across 3 models and 4 data formats. For Qwen2.5-3B FP16, DARA saves 210.3 mW, yielding a net benefit ratio of 187×. At more advanced process nodes, lower supply voltages and improved leakage control may further reduce ASS power.

7 Conclusion

This paper presents **DARA**, a data-aware refreshing architecture for reducing HBM refresh power in LLM accelerators. DARA is motivated by the observation of mismatch between conventional uniform refresh and the significance heterogeneity of LLM weights. To bridge this gap, DARA combines data significance and reliability modeling, significance-aware remapping, and hardware-assisted runtime closed-loop optimization into a unified cross-layer design, successfully achieves substantial refresh-power reduction with negligible overheads.

Acknowledgments

We sincerely thank the anonymous reviewers for their valuable suggestions that improved the paper. This work is supported by National Natural Science Foundation of China (NSFC) grant 62502305, Natural Science Foundation of Shanghai (NSFS) grant 25ZR1402275.

References

[1] Marah Abdin, Jyoti Aneja, Hany Awadalla, et al. 2024. Phi-3 Technical Report: A Highly Capable Language Model Locally on Your Phone. arXiv:2404.14219 [cs.CL] https://arxiv.org/abs/2404.14219

[2] Seungjae Baek, Sangyeun Cho, and Rami Melhem. 2014. Refresh Now and Then. *IEEE Trans. Comput.* 63, 12 (2014), 3114–3126. doi:10.1109/TC.2013.164

[3] Kevin Kai-Wei Chang, Donghyuk Lee, Zeshan Chishti, Alaa R. Alameldeen, Chris Wilkerson, Yoongu Kim, and Onur Mutlu. 2014. Improving DRAM performance by parallelizing refreshes with accesses. In *2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)*. 356–367. doi:10.1109/HPCA.2014.6835946

[4] Kevin Kai-Wei Chang, Donghyuk Lee, Zeshan Chishti, Alaa R. Alameldeen, Chris Wilkerson, Yoongu Kim, and Onur Mutlu. 2016. Reducing Performance Impact of DRAM Refresh by Parallelizing Refreshes with Accesses. arXiv:1601.06352 [cs.AR] https://arxiv.org/abs/1601.06352

[5] Young-Ho Gong and Sung Woo Chung. 2016. Exploiting Refresh Effect of DRAM Read Operations: A Practical Approach to Low-Power Refresh. *IEEE Trans. Comput.* 65, 5 (2016), 1507–1517. doi:10.1109/TC.2015.2448079

[6] Menglong Guan and Lei Wang. 2015. Temperature aware refresh for DRAM performance improvement in 3D ICs. In *Sixteenth International Symposium on Quality Electronic Design*. 207–211. doi:10.1109/ISQED.2015.7085426

[7] Seongyon Hong, Wooyoung Jo, Sangjin Kim, Sangyeob Kim, Soyeon Um, Kyomin Sohn, and Hoi-Jun Yoo. 2025. Dyamond: Compact and Efficient 1T1C DRAM IMC Accelerator With Bit Column Addition for Memory-Intensive AI. *IEEE Journal of Solid-State Circuits* 60, 4 (2025), 1299–1310. doi:10.1109/JSSC.2025.3538899

[8] JEDEC. 2021. *JESD235D: High Bandwidth Memory (HBM) DRAM*. JEDEC Solid State Technology Association. https://www.jedec.org/standards-documents/docs/jesd235a

[9] Florent Kermarrec, Sébastien Bourdeauducq, Jean-Christophe Le Lann, and Hannah Badier. 2020. LiteX: an open-source SoC builder and library based on Migen Python DSL. arXiv:2005.02506 [cs.AR] https://arxiv.org/abs/2005.02506

[10] Seikwon Kim, Wonsang Kwak, Changdae Kim, Daehyeon Baek, and Jaehyuk Huh. 2020. Charge-Aware DRAM Refresh Reduction with Value Transformation. In *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 663–676. doi:10.1109/HPCA47549.2020.00060

[11] Skanda Koppula, Lois Orosa, A. Giray Yağlıkçı, Roknoddin Azizi, Taha Shahroodi, Konstantinos Kanellopoulos, and Onur Mutlu. 2019. EDEN: Enabling Energy-Efficient, High-Performance Deep Neural Network Inference Using Approximate DRAM. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture* (Columbus, OH, USA) *(MICRO-52)*. Association for Computing Machinery, New York, NY, USA, 166–181. doi:10.1145/3352460.3358280

[12] Junhyeong Kwon, Shi-Jie Wen, Rita Fung, and Sanghyeon Baeg. 2023. Temperature Estimation of HBM2 Channels with Tail Distribution of Retention Errors in FPGA-HBM2 Platform. *Electronics* 12, 1 (2023). doi:10.3390/electronics12010032

[13] Shang Li, Zhiyuan Yang, Dhiraj Reddy, Ankur Srivastava, and Bruce Jacob. 2020. DRAMsim3: A Cycle-Accurate, Thermal-Capable DRAM Simulator. *IEEE Computer Architecture Letters* 19, 2 (2020), 106–109. doi:10.1109/LCA.2020.2973991

[14] Xinhan Lin, Liang Sun, Fengbin Tu, Leibo Liu, Xiangyu Li, Shaojun Wei, and Shouyi Yin. 2021. ADROIT: An Adaptive Dynamic Refresh Optimization Framework for DRAM Energy Saving In DNN Training. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*. 751–756. doi:10.1109/DAC18074.2021.9586265

[15] Jamie Liu, Ben Jaiyen, Yoongu Kim, Chris Wilkerson, and Onur Mutlu. 2013. An experimental study of data retention behavior in modern DRAM devices: implications for retention time profiling mechanisms. In *Proceedings of the 40th Annual International Symposium on Computer Architecture* (Tel-Aviv, Israel) *(ISCA '13)*. Association for Computing Machinery, New York, NY, USA, 60–71. doi:10.1145/2485922.2485928

[16] Jamie Liu, Ben Jaiyen, Richard Veras, and Onur Mutlu. 2012. RAIDR: Retention-aware intelligent DRAM refresh. In *2012 39th Annual International Symposium on Computer Architecture (ISCA)*. 1–12. doi:10.1109/ISCA.2012.6237001

[17] Song Liu, Karthik Pattabiraman, Thomas Moscibroda, and Benjamin G. Zorn. 2011. Flikker: saving DRAM refresh-power through critical data partitioning. In *Proceedings of the Sixteenth International Conference on Architectural Support for Programming Languages and Operating Systems* (Newport Beach, California, USA) *(ASPLOS XVI)*. Association for Computing Machinery, New York, NY, USA, 213–224. doi:10.1145/1950365.1950391

[18] Haocong Luo, Yahya Can Tuğrul, F. Nisa Bostancı, Ataberk Olgun, A. Giray Yağlıkçı, and Onur Mutlu. 2024. Ramulator 2.0: A Modern, Modular, and Extensible DRAM Simulator. *IEEE Computer Architecture Letters* 23, 1 (2024), 112–116. doi:10.1109/LCA.2023.3333759

[19] David Shah, Eddie Hung, Clifford Wolf, Serge Bazanski, Dan Gisselquist, and Miodrag Milanovic. 2019. Yosys+nextpnr: An Open Source Framework from Verilog to Bitstream for Commercial FPGAs. In *2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. 1–4. doi:10.1109/FCCM.2019.00010

[20] Ho Hyun Shin, Young Min Park, Duheon Choi, Byoung Jin Kim, Dae-Hyung Cho, and Eui-Young Chung. 2018. EXTREME: Exploiting Page Table for Reducing Refresh Power of 3D-Stacked DRAM Memory. *IEEE Trans. Comput.* 67, 1 (2018), 32–44. doi:10.1109/TC.2017.2723392

[21] Lokesh Siddhu and Preeti Ranjan Panda. 2019. FastCool: Leakage Aware Dynamic Thermal Management of 3D Memories. In *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*. 272–275. doi:10.23919/DATE.2019.8715091

[22] Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. 2024. A Simple and Effective Pruning Approach for Large Language Models. arXiv:2306.11695 [cs.CL] https://arxiv.org/abs/2306.11695

[23] Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, et al. 2024. Gemma 2: Improving Open Language Models at a Practical Size. arXiv:2408.00118 [cs.CL] https://arxiv.org/abs/2408.00118

[24] An Yang, Baosong Yang, Beichen Zhang, et al. 2025. Qwen2.5 Technical Report. arXiv:2412.15115 [cs.CL] https://arxiv.org/abs/2412.15115