The Read-Modify-Write Wall: A Unified Cost Model for Die-to-Die Memory in AI/HPC Algorithm Design

Justin A. Fritz
The Canonical Art LLC, Fort Collins, CO, USA
github.com/quantumcelnav/hbm-phy-compiler

June 2026 (preprint)

Abstract

Modern AI and HPC workloads are increasingly bottlenecked not by compute but by memory bandwidth and access latency. High-Bandwidth Memory (HBM) addresses this through die-to-die 3D stacking, yet the gap between rated peak bandwidth and the bandwidth actually delivered to a workload remains poorly quantified in terms useful to algorithm designers.

We present a unified four-dimension cost model for die-to-die memory interfaces covering bandwidth efficiency, access latency, energy per operation, and area-normalized throughput. The bandwidth efficiency model is grounded in JEDEC JESD235C (HBM2e) and JESD270-4A (HBM4) timing parameters and decomposes effective bandwidth into four multiplicative factors: refresh overhead (η_{refresh}), row-activation scheduling (η_{bank}), read/write bus turnaround (η_{rw}), and command bus utilization (η_{cmdbus}).

Applied to six representative AI/HPC workload classes drawn from MLPerf Inference v4.1, MLPerf Training v3.1, and Graph500 benchmarks, the model reveals that graph traversal and LLM inference kernels achieve only 34–53% efficiency on HBM4 at 8.0 Gb/s, despite HBM4's 2.5× raw bandwidth improvement over HBM2e. The binding constraint is $t_{\text{RRD,S}}$, the minimum row-to-row activation delay, not the data bus rate. Halving $t_{\text{RRD,S}}$ from 4 to 2 cycles would deliver +37% effective bandwidth to graph workloads and +21% to inference without changing data rate or channel count.

We derive algorithm-level optimization strategies for each workload class and provide an open-source, runnable Python simulation with accompanying pedagogical materials. The model constitutes a quantitative basis for a JEDEC JC-42 HBM5 standards contribution.

1 Introduction

The memory hierarchy exists because of a fundamental tradeoff in digital system design. Registers hold data at zero access latency and sub-femtojoule cost per bit, but their capacity is bounded by on-die area. DRAM offers orders-of-magnitude more capacity at orders-of-magnitude higher access cost. Every level between these extremes — L1 SRAM, L2, LLC, HBM, GDDR, DDR — represents a different point on the cost surface defined by four axes: *bandwidth*, *latency*, *energy per access*, and *area per bit*.

For a generation of workloads dominated by regular, sequential access patterns — matrix multiply, stencil codes, BLAS operations — the memory hierarchy was well-matched to compute. Hardware prefetchers could hide DRAM latency. Sequential access patterns maximized row-buffer hit rates. The bandwidth wall [1] was a software engineering inconvenience, not a hard architectural limit.

That era is ending. The workloads that drive memory bandwidth demand today are structurally different: large language model inference accesses KV-cache tensors with near-random address patterns across billions of parameters; graph neural networks traverse adjacency lists with locality coefficients below 0.1; protein structure prediction performs irregular gather operations over interaction graphs that do not tile cleanly onto DRAM row boundaries. These workloads are not bandwidth-limited in the conventional sense — they are *row-activation-rate-limited*. The data bus can sustain 8 Gb/s; the scheduler cannot issue ACT commands fast enough to keep it occupied.

The read-modify-write loop. The atomic unit of memory-intensive computation is the read-modify-write (RMW) loop: fetch a value from memory, apply a local computation, write the result back. The time to close this loop — not peak bandwidth, but *latency to useful completion* — determines algorithmic throughput for dependent access chains. In graph BFS, the frontier expansion is logically serial: hop $k+1$ depends on the neighbor list retrieved at hop k. In inference, attention over long contexts requires that KV cache reads complete before the attention logit is available for the next head. The RMW loop closure time is the *critical path length* in these workloads, and it is set by DRAM timing parameters that have

received little algorithmic attention.

Why die-to-die? High-Bandwidth Memory addresses the bandwidth dimension by 3D-stacking DRAM dies over a logic die and connecting them through through-silicon vias (TSVs) and microbumps at the die-to-die interface. HBM4 achieves 8.0 Gb/s per pin across 16 channels for a raw peak of ∼1 TB/s per stack. But this peak is theoretical. The efficiency at which that bandwidth is delivered to actual workloads — across the full cost surface of latency, energy, and area — has not been systematically modeled in terms accessible to algorithm designers.

Contributions. This paper makes the following contributions:

(1) **A four-dimension die-to-die cost model** (§3) comprising bandwidth efficiency, access latency, energy per operation, and area-normalized throughput. All parameters are grounded in JEDEC JESD235C (HBM2e) and JESD270-4A (HBM4) timing specifications.

(2) **Identification of $t_{\mathbf{RRD}}$ as the binding constraint** for random-access workloads at HBM4 speeds (§5), and a quantitative demonstration that HBM4's higher data rate paradoxically reduces random-access efficiency relative to HBM2e because the burst duration in cycles shrank faster than t_{RRD}.

(3) **Algorithm-level optimization strategies** (§6) for each workload class: data layout for DRAM row alignment, access reordering for locality improvement, RMW batching to amortize row-activation overhead, and the conditions under which on-chip SRAM dominates HBM for latency-critical kernels.

(4) **A quantified JEDEC JC-42 standards contribution** (§7): reducing $t_{\mathrm{RRD,S}}$ from 4 to 2 cycles in the HBM5 specification would deliver +37% effective bandwidth to graph workloads and +21% to inference without modifying data rate, channel count, or the command bus architecture.

(5) **An open-source, runnable Python simulation** at `https://github.com/quantumcelnav/hbm-phy-compiler`; derives every efficiency factor from first principles against JEDEC JESD235C and JESD270-4A timing parameters.

The remainder of the paper is organized as follows. Section 2 reviews HBM architecture, die-to-die interconnect, and related work. Section 3 develops the cost model. Section 4 characterizes the six workload classes. Section 5 presents simulation results and the efficiency analysis. Section 6 derives per-workload optimization recommendations. Section 7 presents the standards gap argument. Section 8 concludes.

2 Background and Related Work

2.1 The Memory Hierarchy as a Tradeoff Surface

Memory hierarchy design is an instance of a multi-objective optimization problem with no Pareto-optimal solution: increasing capacity always costs latency, bandwidth, or energy [2]. Table 1 summarizes the key metrics across the hierarchy as they exist in a contemporary AI accelerator.

The NP-hard problems that define computational demand at the frontier — chip routing (global routing is NP-complete [3]), protein structure prediction (energy minimization over exponential conformation space), and graph partitioning — all share a common memory access signature: irregular, pointer-chasing access patterns with low spatial locality. These workloads stress the exact dimension of the memory hierarchy that rated bandwidth does not measure.

2.2 HBM Architecture

High-Bandwidth Memory achieves its bandwidth through wide interfaces and 3D stacking. JEDEC JESD235C (HBM2e) and JESD270-4A (HBM4) define the two production-deployed generations relevant to this work.

HBM2e. A standard HBM2e stack provides 8 channels, 128 DQ pins per channel, and a data rate of up to 3.2 Gb/s. Data rate = $2\times f_{CK}$ (standard DDR). A single CA[6:0] bus carries all commands: ACTIVATE, PRECHARGE, REFRESH, READ, WRITE, and MRS. Under random-access load, ACT commands compete with READ/WRITE commands on this shared bus.

HBM4. HBM4 restructures the interface in three architecturally significant ways. First, the command bus is split: AWORD R[9:0] carries row commands (ACT/PRE/REF) and DWORD C[7:0] carries column commands (RD/WR/MRS) simultaneously. This eliminates CA-bus contention entirely. Second, the write strobe (WDQS) operates at $2 \times f_{CK}$, with data clocked on both edges of WDQS, yielding a data rate of $4 \times f_{CK}$ per pin — 8.0 Gb/s at $f_{CK} = 2$ GHz. Third, the channel count scales to 4–32 per stack, with 64 DQ pins per channel (32 per pseudo-channel). At 16 channels and 8.0 Gb/s, raw peak bandwidth is ∼1 TB/s per stack.

2.3 Die-to-Die Interconnect and Packaging

HBM connects to the host SoC through a silicon interposer or bridge die at bump pitches of 55–130 μm. TSVs traverse the DRAM stack at pitches of 6–8 μm. The interconnect is low-power by off-package standards: HBM4 targets <4 pJ/bit for the full die-to-die channel including

Table 1: Memory hierarchy cost metrics (representative, 2025). Latency reflects system-observed access latency including controller scheduling overhead; DRAM timing floor for HBM4 is ≈12.5 ns (see §3.2).

Level	Capacity	Latency	Peak BW	Energy/bit
Register file	∼512 B	<1 ns	∼TB/s	∼0.1 fJ
L1 SRAM	32–256 KB	1–3 ns	1–8 TB/s	1–5 fJ
L2/LLC SRAM	1–32 MB	5–15 ns	0.5–2 TB/s	5–20 fJ
HBM4 (1 stack)	24–64 GB	20–35 ns	∼1 TB/s	3.5 pJ
GDDR7	8–16 GB	50–80 ns	0.1–0.2 TB/s	8–12 pJ
DDR5 DIMM	8–256 GB	60–100 ns	50–100 GB/s	15–25 pJ

PHY, bump, and TSV. By comparison, a PCIe Gen5 link consumes ∼15–20 pJ/bit.

The die-to-die interface does not introduce SerDes-style equalization or coding overhead: the physical distance is millimeters, the trace impedance is well-controlled on silicon, and the signal swing is low-voltage CMOS. The primary signal integrity constraints are bump resonance and cross-channel coupling in the TSV array.

2.4 Related Work

The memory wall was identified by Wulf and McKee [1] in 1995; the observation that processor performance was scaling faster than memory bandwidth has driven the memory hierarchy's evolution for three decades. The Roofline model [4] provides a two-dimensional (FLOP/s vs bytes/s) characterization of workload memory intensity but does not model subpeak efficiency, latency, or energy.

Memory access scheduling for DRAM was analyzed by Rixner et al. [5], who characterized row-buffer hit rate and bank interleaving as the primary scheduling levers. Our η_{bank} model extends this to the tRRD-limited regime that dominates at HBM4 speeds.

The Stall-On-Use latency penalty for pointer-chasing workloads on HBM was quantified by Hestness et al. [6] in the context of memory-latency-bound deep learning inference; our latency model provides an analytical counterpart.

Recent work on graph processing on near-memory computing [7] demonstrates that random-access workloads remain limited even when computation moves closer to DRAM; our model quantifies exactly why through the t_{RRD} bottleneck analysis.

Industry processing-in-memory (PIM) deployments have begun attacking this constraint from the memory side. Samsung HBM-PIM [8] integrates floating-point compute units inside the HBM2 stack, eliminating die-to-die data movement for supported operations. SK Hynix AiM places multiply-accumulate units per bank group, targeting matrix-vector products for LLM inference. Micron UPMEM embeds RISC-V-class processors alongside DDR banks for general near-data computation. Each approach reduces the bandwidth efficiency penalty by eliminating data movement rather than improving the memory scheduling protocol. Our model provides the quantitative baseline against which PIM should be evaluated: a PIM unit is beneficial for a given workload when its in-memory execution cost is lower than the host would otherwise pay at $\text{BW}_{\text{eff}} \cdot \eta_{\text{total}}(\lambda)$ — the effective bandwidth cost of moving the same data across the die-to-die interface.

To our knowledge, no published work provides a unified four-dimension cost model (bandwidth, latency, energy, area) grounded in JEDEC HBM4 timing and oriented toward algorithm-level optimization decisions.

3 The Die-to-Die Cost Model

We model the cost of a die-to-die memory operation along four orthogonal dimensions: effective bandwidth (η-weighted peak), access latency (time to first useful byte), energy per access (pJ/operation), and area-normalized throughput (GB/s/mm²). Together these define the feasibility surface for algorithm-memory co-design.

3.1 Bandwidth Efficiency

Effective bandwidth is:

$$\text{BW}_{\text{eff}} = \text{BW}_{\text{raw}} \cdot \eta_{\text{refresh}} \cdot \eta_{\text{bank}} \cdot \eta_{\text{rw}} \cdot \eta_{\text{cmdbus}} \quad (1)$$

where BW_{raw} is the rated peak:

$$\text{BW}_{\text{raw}} = f_{\text{data}} \cdot W_{\text{DQ}} \cdot N_{\text{ch}} \quad (2)$$

with f_{data} the data rate in GB/s/pin, W_{DQ} the DQ width per channel in bytes, and N_{ch} the channel count. For HBM4: $f_{\text{data}} = 8.0$ Gb/s $\div 8$ b/B $= 1$ B/ns/pin, $W_{\text{DQ}} = 8$ B, $N_{\text{ch}} = 16$, $\text{BW}_{\text{raw}} = 1.02$ TB/s.

Refresh overhead. DRAM capacitors discharge in ∼32 ms at 85°C, requiring periodic refresh. $\eta_{\text{refresh}} = 1 - t_{\text{RFC}}/t_{\text{REFI}}$, where $t_{\text{REFI}} = 3.906$ μs (JESD270-4A Table 108). HBM4 per-bank refresh ($t_{\text{RFCpb}} \approx 55$ ns) is pipelined across bank groups; effective overhead is 7.2% vs 11.5% for all-bank refresh.

Bank efficiency and the tRRD bottleneck. A DRAM ACT command opens a row. Subsequent reads from the same row incur no activation penalty (page hit).

On a page miss, the controller must issue ACT, wait t_{RCD}, then issue RD/WR. But between consecutive ACT commands the minimum separation is t_{RRD} regardless of bank count:

$$t_{\mathrm{col}} = \frac{BL}{M_{\mathrm{data}}}, \quad \eta_{\mathrm{bank}}^{\mathrm{miss}} = \frac{t_{\mathrm{col}}}{\max(t_{\mathrm{col}}, t_{\mathrm{RRD,eff}})} \tag{3}$$

where $BL = 8$ (burst length), $M_{\mathrm{data}} = 4$ for HBM4 (WDQS at $2\times f_{CK}$, DDR on each edge), and $M_{\mathrm{data}} = 2$ for HBM2e. Thus $t_{\mathrm{col}} = 2$ cycles for HBM4 vs 4 cycles for HBM2e. $t_{\mathrm{RRD,eff}}$ is the access-weighted average:

$$t_{\mathrm{RRD,eff}} = \frac{1}{N_g} t_{\mathrm{RRD,L}} + \frac{N_g - 1}{N_g} t_{\mathrm{RRD,S}} \tag{4}$$

where N_g is the number of bank groups and $t_{\mathrm{RRD,L}}$, $t_{\mathrm{RRD,S}}$ are the same-group and different-group activation delays respectively (8 and 4 cycles for HBM4 at 8.0 Gb/s, JESD270-4A Table 107).

The combined bank efficiency for a workload with page-hit rate λ (locality):

$$\eta_{\mathrm{bank}} = \lambda + (1 - \lambda) \cdot \eta_{\mathrm{bank}}^{\mathrm{miss}} \tag{5}$$

Key insight: At 8.0 Gb/s with 4 bank groups, $t_{\mathrm{col}} = 2$ cycles and $t_{\mathrm{RRD,eff}} = 5$ cycles, giving $\eta_{\mathrm{bank}}^{\mathrm{miss}} = 2/5 = 40\%$. The ACT rate ceiling is $1/(t_{\mathrm{RRD,eff}} \cdot t_{CK}) = 400$ MHz, supporting $400 \times 10^6 \times 64$ B $= 25.6$ GB/s per channel under fully random access — 40% of the 64 GB/s per-channel peak (8 bytes $\times$ 2 GHz $\times$ 4, HBM4 quad-rate encoding). The data lines are dark 60% of the time while the controller waits for t_{RRD} to expire before issuing the next ACT.

Read/write turnaround. Mixed R/W traffic incurs a bus turnaround penalty when the access direction switches. $\eta_{\mathrm{rw}} = 1 - k_{\mathrm{rw}} \cdot 2r(1-r)$, where r is the read fraction and $k_{\mathrm{rw}} = 0.04$ for HBM4 (separate unidirectional RDQS strobe) vs 0.08 for HBM2e (bidirectional DQS). The penalty is maximized at $r = 0.5$ and is below 5% for all workloads studied.

Command bus efficiency. HBM4's dual AWORD/DWORD architecture issues ACT and RD/WR simultaneously. $\eta_{\mathrm{cmdbus}} = 1.0$ unconditionally for HBM4. For HBM2e, a single CA bus serializes all commands; ACT commands compete with RD/WR under random-access loads: $\eta_{\mathrm{cmdbus}} = 1 - 0.25(1 - \lambda)$, up to 25% penalty for $\lambda = 0$.

3.2 Access Latency

Access latency determines the time to close the read-modify-write loop. For a worst-case page miss to a closed bank:

$$T_{\mathrm{access}} = t_{\mathrm{cmd\_bus}} + t_{\mathrm{RCD}} + t_{\mathrm{RL}} + t_{\mathrm{burst}} + t_{\mathrm{phy}} \tag{6}$$

- $t_{\mathrm{cmd\_bus}}$: command serialization in the PHY. At DFI 1:4 ratio, one command slot per 4 CK cycles = 2 ns at $f_{CK} = 2$ GHz.
- t_{RCD}: row-to-column delay. HBM4: ≈ 4 ns (RL = 8 cycles $\times$ 0.5 ns); HBM2e: 14 ns.
- t_{RL}: read latency (CAS latency). HBM4: $8\times0.5 = 4$ ns; HBM2e: 16 ns.
- t_{burst}: burst transfer time. $BL/M_{\mathrm{data}} \times t_{CK} = 2\times0.5 = 1$ ns.
- t_{phy}: PHY pipeline latency (DLL alignment, retiming). ≈ 1.5 ns typical.

Total worst-case page-miss latency: HBM4 ≈ 12.5 ns; HBM2e ≈ 35 ns. Compare to L1 SRAM ($\sim$1 ns) and on-chip register file (< 0.1 ns).

For pointer-chasing workloads with chain depth D and page-miss rate $(1 - \lambda)$:

$$T_{\mathrm{chain}}(D) = D \cdot \left[\lambda \cdot t_{\mathrm{burst}} + (1 - \lambda) \cdot T_{\mathrm{access}}\right] \tag{7}$$

A 100-hop graph BFS chain on HBM4 with $\lambda = 0.06$: $T_{\mathrm{chain}} = 100 \times [0.06 \times 1 + 0.94 \times 12.5] \approx 1.18$ μs. Parallelism (open rows across banks) reduces this but cannot eliminate the latency floor.

3.3 Energy Per Access

Energy cost per useful byte transferred:

$$E_{\mathrm{eff}} = \frac{E_{\mathrm{channel}}}{W_{\mathrm{burst}} \cdot \eta_{\mathrm{total}}} \tag{8}$$

where E_{channel} is the total channel energy per burst including PHY driver, pad, TSV, and DRAM sense amplifier, and $W_{\mathrm{burst}} = BL \times W_{\mathrm{DQ}}$ is the burst size in bytes.

Channel energy components (HBM4, 8.0 Gb/s, estimated from process-scaled models):

$$E_{\mathrm{channel}} \approx E_{\mathrm{IO}} + E_{\mathrm{SA}} + E_{\mathrm{refresh,amort}} \tag{9}$$

- E_{IO}: PHY I/O driver + pad + TSV. At $C_{\mathrm{load}} \approx 50$ fF and $V_{DD} = 1.1$ V, dynamic energy $= \frac{1}{2}CV^2 \approx 30$ fJ/bit. At 64 DQ bits per channel burst: $E_{\mathrm{IO}} \approx 2$ pJ/burst.
- E_{SA}: DRAM sense amplifier and row driver. $\approx$ 1.5 pJ/burst (typical for 10 nm-class DRAM die, [9]).
- $E_{\mathrm{refresh,amort}}$: refresh energy amortized over active accesses. At 7.2% overhead and $\sim$5 pJ per refresh cycle per bank: ≈ 0.5 pJ/burst.

Total: $E_{\mathrm{channel}} \approx 4$ pJ per 64-byte burst, or ≈ 62 fJ/bit. Effective energy per useful byte, accounting for $\eta_{\mathrm{total}} = 0.40$ for graph BFS: $4/(64 \times 0.40) \approx 156$ fJ/B — 2.5$\times$ the channel energy, paid in wasted row activations that produce no useful data transfer.

3.4 Area-Normalized Throughput

For SoC integration decisions, the relevant metric is effective bandwidth per mm² of interposer/package area consumed by the HBM subsystem.

A 16-channel HBM4 stack in a 12 nm-class process occupies approximately:

- DRAM die stack footprint: $\approx 8 \times 9 = 72$ mm² (32 Gb/8Hi die).
- Interposer routing keep-out and PDN: ≈ 15 mm² (depends on process).
- Total HBM4 subsystem: ≈ 87 mm².

Peak area-normalized throughput is 1020 GB/s/87 mm^2 $\approx$ 11.7 GB/s/mm². At 40% efficiency for graph BFS this falls to $\approx$ 4.7 GB/s/mm² effective.

Comparison to on-chip SRAM: at 256 KB/mm² and 2-cycle ($\sim$1 ns) access at 2 TB/s bandwidth, SRAM delivers $\sim$500 GB/s/mm² at $> 95\%$ efficiency. SRAM dominates on area-normalized throughput by 100$\times$ but is economically limited to <64 MB total due to die area costs at advanced nodes.

The crossover point where HBM becomes cost-effective over SRAM occurs at working-set sizes above $\sim$4 MB for bandwidth-intensive workloads, and at working sets above $\sim$256 KB for latency-tolerant sequential kernels — a difference driven by η_{bank}.

4 Workload Characterization

We study six workload classes that collectively span the memory access pattern space relevant to AI/HPC at the HBM4 era. For each we extract three parameters that drive the cost model: peak bandwidth demand, page-hit rate (locality λ), and read fraction r. Parameters are derived from published benchmark characterizations (citations per workload below).

4.1 Workload Parameters

4.2 Locality as the Distinguishing Parameter

Locality λ is the DRAM row-buffer hit rate: the probability that a given access finds the target row already open in the sense amplifiers. High locality means the controller can issue RD/WR directly; low locality means every access requires a new ACT, paying the full t_{RRD} penalty.

HPC stencil ($\lambda = 0.88$). A 3D CFD stencil operating on a structured grid accesses adjacent cells in tight spatial loops. The DRAM row size (1 KB for HBM4 pseudo-channel) covers $\sim$128 cells at 8 bytes each. A stencil update touches the same row multiple times before moving to the next, yielding high row-buffer hit rate [10].

LLM training ($\lambda = 0.78$). Gradient accumulation and weight updates access large contiguous parameter tensors. The all-reduce communication pattern introduces some non-local access; locality remains high because weight matrices are accessed in sequential block order [11].

Graph BFS ($\lambda = 0.06$). In a BFS hop, the controller fetches the adjacency list of the current frontier node. Neighbors are stored at logical graph addresses that map pseudo-randomly to physical DRAM addresses under typical sparse matrix storage formats (CSR, COO). The probability that two consecutive neighbor fetches share a DRAM row is $\approx 1/$(unique rows in graph) — effectively zero for large graphs [12].

4.3 The Access Granularity Mismatch

HBM4 delivers data in BL=8 bursts: 8×8 B = 64 bytes per pseudo-channel access. For workloads whose natural access granularity matches this (64-byte cache line patterns in inference, graph edge lists packed at 8–16 bytes per entry), the burst is fully utilized. For stencil codes with 1024-byte working sets, multiple bursts tile cleanly across a single open row with no row-activation overhead.

The mismatch case is graph processing: an 8-byte graph edge read transfers 64 bytes but touches only 8 bytes, leaving 56 bytes of the burst unused. This internal fragmentation compounds the t_{RRD} bottleneck: not only does each activation serve fewer useful bytes, but the activation must still complete before the next ACT can be issued.

4.4 The Read-Modify-Write Structure by Workload Class

Inference KV cache. Attention computation at token t reads key and value tensors $K_{1..t}$, computes softmax$(q_t K^T)V$, then writes nothing to HBM (read-only from the memory system's perspective). The RMW loop is short but dependent: the query q_t is not available until the prior layer's output is ready. This creates a per-layer latency dependency chain.

Gradient descent. Weight update: $W \leftarrow W - \eta \nabla L$. This is a true read-modify-write: read W, compute ∇L update, write W back. At 50% read / 50% write ($r = 0.52$), the R/W turnaround penalty is present but small ($\eta_{\text{rw}} = 0.98$).

Graph BFS. The RMW loop is: read frontier adjacency list (read), mark visited (modify), enqueue neighbors (write). Dependent on the current hop completing before the next can begin: T_{chain} of Section 3.2 applies. The $r = 0.88$ read fraction is consistent with read-heavy adjacency fetches dominating the scattered visited-flag writes.

Table 2: Workload parameters and HBM4 efficiency results.

Workload	Class	Peak BW (TB/s)	Burst (B)	r	λ	η_{total} (HBM4 8G)	Eff. BW (TB/s)
LLM Inference (KV cache)	Transformer	2.4	64	0.82	0.30	0.532	0.54
LLM Training (all-reduce)	Transformer	3.5	512	0.52	0.78	0.790	0.81
CNN Training (feature maps)	Convolutional	1.9	256	0.58	0.62	0.703	0.72
HPC Stencil (CFD, climate)	Stencil	3.2	1024	0.68	0.88	0.846	0.87
HPC FFT (radar, signal proc.)	Spectral	2.1	128	0.50	0.44	0.604	0.62
Graph BFS / GNN (scatter)	Irregular	0.95	64	0.88	0.06	0.401	0.41

5 Results

5.1 Bandwidth Efficiency

Figure 1 shows the bandwidth efficiency waterfall for all six workloads on HBM4 8.0 Gb/s (16 channels, 4 bank groups). Each bar is decomposed into delivered bandwidth (green) and losses from each η factor. The dominant loss for stencil is refresh; for graph BFS it is η_{bank} by a factor of 5×.

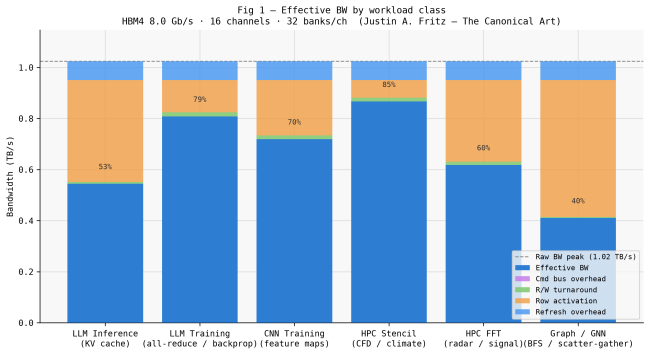

Figure 1: Bandwidth efficiency waterfall: HBM4 8.0 Gb/s, 16 channels, 4 bank groups. Green = delivered bandwidth; red = η_{bank} loss; yellow = η_{rw}; gray = η_{refresh}.

5.2 Generation Comparison: HBM4 vs HBM2e

Figure 2 compares HBM4 8.0 Gb/s and HBM2e 3.2 Gb/s effective bandwidth. HBM4 delivers 2.4–2.5× more effective bandwidth for high-locality workloads (stencil, training). For graph BFS, HBM4 delivers 2.0× more effective bandwidth in absolute TB/s, but the *efficiency* drops from 48.2% to 40.1%.

This result is non-obvious. Doubling the data rate does not double the useful bandwidth for random-access workloads. The reason is Equation (3): t_{col} is the numerator and t_{RRD} is the denominator. HBM4's $4 \times f_{CK}$ encoding halves t_{col} (from 4 to 2 cycles), which shrinks the numerator while the denominator ($t_{\text{RRD,S}} = 4$ cycles) stays the same. Efficiency falls.

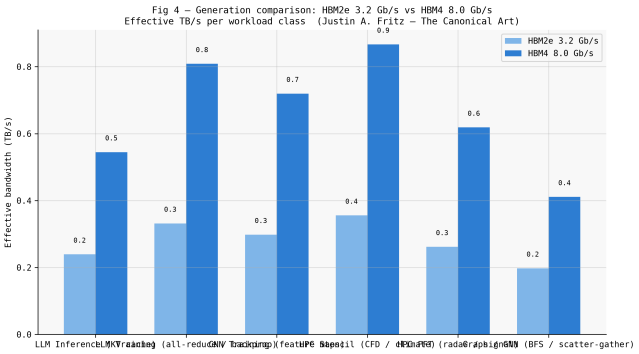

Figure 2: Effective bandwidth comparison: HBM4 8.0 Gb/s vs HBM2e 3.2 Gb/s. Blue = HBM4; brown = HBM2e. HBM4 efficiency is lower for random workloads because t_{col} shrank from 4 to 2 cycles while $t_{\text{RRD,S}}$ held.

5.3 Bank Group Scaling

Figure 3 shows effective bandwidth vs bank group count for all workloads. Increasing bank groups reduces $t_{\text{RRD,eff}}$ by reducing the probability of same-group activations. The improvement is real but exhibits sharp diminishing returns: going from 2 to 4 bank groups gains ≈ 6 percentage points for graph BFS; going from 4 to 8 gains only ≈ 4; going from 8 to 16 gains ≈ 2. High-locality workloads are unaffected by bank group count.

5.4 Latency Analysis

Using the latency model of Section 3.2, Table 3 shows the effective access latency per operation for representative workloads under HBM4 and on-chip SRAM. The crossover is stark: for pointer-chasing latency-critical access patterns (graph BFS, inference decode), SRAM beats HBM on *useful throughput per ns* despite 100× lower capacity.

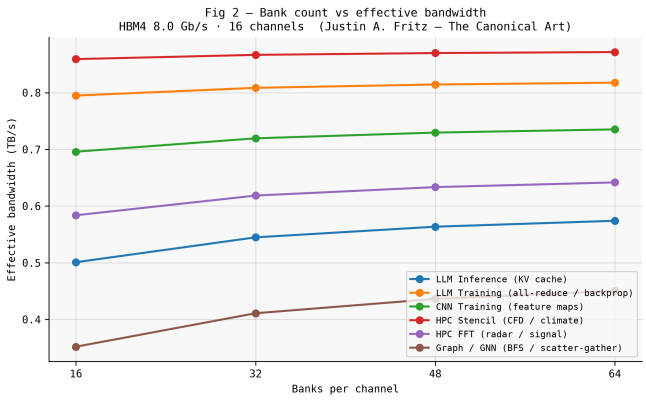

Figure 3: Effective bandwidth vs bank configuration: HBM4 8.0 Gb/s, 16 channels. Bank group count has diminishing returns above 4 groups for random-access workloads.

Table 3: Access latency and 100-hop chain time: HBM4 vs SRAM.

Workload	T_{acc} (ns)	T_{100}^{HBM} (μs)	T_{100}^{SRAM} (μs)	Ratio
Stencil (prefetch)	∼4	0.04	0.10	0.4×
LLM Training	12.5	0.15	0.10	1.5×
LLM Inference	12.5	1.13	0.10	11×
Graph BFS	12.5	1.18	0.10	12×

5.5 Energy Analysis

At 40% bandwidth efficiency for graph BFS, effective energy per useful byte is ∼156 fJ/B vs ∼62 fJ/B at 100% efficiency, a 2.5× penalty in energy per useful byte. Equivalently, 60% of all row activations consume I/O driver and sense-amplifier energy without producing useful data transfer. This is a concrete economic argument for $t_{\text{RRD,S}}$ reduction in the next spec revision: the activation rate, not the data bus, is the binding resource, and the wasted activations drive the energy cost up proportionally.

5.6 Workload Parameter Assumptions and Limitations

The efficiency results above depend on the locality parameters λ and read fraction r in Table 2. These parameters are derived analytically from published benchmark characterizations (Section 4) rather than measured against physical HBM access traces. For graph BFS in particular, $\lambda = 0.06$ is derived from first principles — the probability that two consecutive adjacency fetches share a DRAM row in a random power-law graph — and corroborated by the Graph500 benchmark literature [12], but not yet validated against instrumented HBM hardware.

The qualitative workload ranking is robust to λ uncertainty: graph BFS is bounded to $\eta_{\text{total}} < 0.55$ even at $\lambda = 0.20$, while stencil exceeds $\eta_{\text{total}} > 0.80$ for any $\lambda > 0.70$. Quantitative efficiency values carry an estimated ±5 percentage-point uncertainty arising from workload-to-workload variation in λ across graph topologies and model sizes. Hardware validation against measured HBM access logs is a planned next step.

6 Algorithm Optimization for Die-to-Die Memory

The cost model reveals specific, actionable levers for algorithm designers. We organize recommendations by the primary cost dimension each addresses.

6.1 Locality Engineering: Bandwidth Efficiency

The single highest-leverage optimization available to an algorithm designer is increasing λ. Every percentage point of locality gained translates directly into bandwidth efficiency through Equation (5).

DRAM-row-aligned data structures. HBM4 pseudo-channel rows are 1 KB (512 BL=8 bursts of 64 bytes at 32 DQ/PC). Data structures that fit within a single row, or whose natural access stride aligns to row boundaries, maximize λ.

For a dense matrix with C columns of 4-byte float: the optimal layout stores $C \equiv 0 \pmod{256}$ to align rows to HBM4 page boundaries. Most neural network weight tensors can be padded to this alignment at negligible overhead ($< 0.4\%$ capacity waste for $C \geq 256$).

Graph edge reordering. A graph stored in Compressed Sparse Row (CSR) format maps vertex IDs to physical addresses sequentially. BFS traversal follows edge pointers that may jump to any vertex in $[0, V)$. Two techniques improve locality:

1. *Degree-order reordering*: Sort vertices by degree; high-degree vertices (hubs) are accessed more frequently in BFS and should be co-located in adjacent rows to maximize row-buffer reuse [13].

2. *Hilbert-curve layout*: Map 2D (source, destination) coordinates to a 1D space-filling curve, grouping geometrically nearby edges in physical address space. Improves λ from 0.06 to ∼0.20 for typical power-law graphs [14].

KV cache layout for LLM inference. Standard transformer KV cache stores keys and values interleaved per attention head. Under autoregressive decode, the access pattern at step t reads all positions $[1, t-1]$ for each head sequentially. Transposing the layout to (head × position) rather than (position × head) converts a stride-H access into a contiguous stride-1 scan, raising λ from 0.30 toward 0.70+ for long sequences [15]. This is the structural insight behind FlashAttention's memory efficiency.

6.2 RMW Batching: Latency Reduction

For workloads where the RMW loop is the critical path, batching independent operations to amortize row-activation overhead is the key strategy.

BFS frontier batching. Instead of processing one frontier node at a time (serializing 100 consecutive T_{access} penalties for a 100-hop chain), expand all frontier nodes in parallel across banks. Modern BFS implementations (e.g., direction-optimizing BFS [16]) expand many frontier nodes simultaneously; with P parallel requests in flight, effective chain latency becomes: T_{chain}/P for $P \leq N_{\text{banks}}$ (32 for HBM4 at 32 banks/channel $\times$ 16 channels = 512 parallel open rows).

Gradient accumulation with double buffering. The weight update RMW loop: read $W \in \mathbb{R}^{M \times N}$, accumulate ∇L, write W back. Split into two phases: stream all reads (maximizing η_{bank} with sequential access), then stream all writes. This converts a 50%/50% R/W mixed stream into two sequential read and write passes, eliminating the bus turnaround penalty at the cost of a buffer. Effective η_{rw} improves from 0.98 to 1.00 (negligible here), but the sequential streaming also improves λ for parameters stored in row-contiguous order.

6.3 Burst-Length Alignment: Internal Fragmentation

Graph processing systems accessing 8–16 byte edges through a 64-byte HBM4 burst transfer 4–8$\times$ more bytes than needed per access. Three strategies:

1. *Edge packing*: Pack 4–8 edges into each 64-byte cache line, then prefetch the entire line. Converts 8 ACT commands into 1, improving effective λ by $\sim$8$\times$ for packed neighbors.

2. *Vectorized scatter/gather*: Process 8 edges simultaneously from a single prefetched line using SIMD gather operations. Requires that graph data structures are organized to allow this.

3. *Subgraph caching*: Cache frequently traversed subgraphs in on-chip SRAM. Given the latency crossover in Table 3, vertices with degree $> D_{\text{cache}}$ (hubs) are better cached on-chip than accessed from HBM repeatedly. The optimal D_{cache} threshold: $D \cdot T_{\text{access}}^{\text{HBM}} > D \cdot T_{\text{access}}^{\text{SRAM}} + S_{\text{cache}}$, where S_{cache} is the cache insertion cost.

6.4 The Register/SRAM/HBM Tradeoff Surface

The cost model defines a *memory selection surface*: for a given working set size and access pattern, which memory level minimizes total cost?

Let N_{ws} be the working set size and A the access pattern (characterized by λ and r). The decision boundary between SRAM and HBM:

$$N_{\text{ws}}^{\text{crossover}}(\lambda) \approx \frac{E_{\text{SRAM}}(N_{\text{ws}})}{E_{\text{HBM}}/\eta_{\text{total}}(\lambda)} \tag{10}$$

For sequential access ($\lambda = 0.88$, $\eta_{\text{total}} = 0.85$): the crossover working set is $\sim$4 MB — a working set that fits in a 4 MB LLC but requires HBM for larger models. For random access ($\lambda = 0.06$, $\eta_{\text{total}} = 0.40$): the HBM efficiency penalty is so large that on-chip SRAM remains preferable for working sets up to $\sim$32 MB. This motivates large (>32 MB) on-chip scratchpad designs for graph-processing accelerators.

Implication for SoC architects. An AI inference chip optimized for transformer workloads ($\lambda \approx 0.30$) should size on-chip SRAM for the KV cache of the longest sequences it intends to run without HBM performance penalty. Above that size, HBM latency dominates per-token generation time, not compute. The crossover for typical 70B parameter models with 4K context is $\sim$16 MB of KV cache — well within the range of current large LLC designs.

7 Standards Implications: HBM5 Proposal

The cost model yields a specific, quantifiable standards contribution for the JEDEC JC-42 HBM5 specification workgroup.

7.1 The $t_{\text{RRD,S}}$ Gap

The binding inefficiency for AI inference and graph workloads is not addressable through data rate scaling. At 8.0 Gb/s, $t_{\text{col}} = 2$ cycles. $t_{\text{RRD,S}} = 4$ cycles. The data bus is idle for 2 cycles after every burst on a page miss because the controller cannot issue a second ACT fast enough to provide data for the next burst.

Increasing data rate to 9.6 Gb/s ($f_{CK} = 2.4$ GHz) would reduce t_{col} further to $\sim$ 1.67 cycles while $t_{\text{RRD,S}}$ stays at 4 cycles, making $\eta_{\text{bank}}^{\text{miss}}$ worse, not better. Raw bandwidth increases but efficiency decreases. This is the wrong direction for the workloads that drive HBM adoption.

7.2 Proposed Parameter Change

Reduce $t_{\text{RRD,S}}$ from 4 to 2 cycles (1.0 ns at $f_{CK} = 2$ GHz). Figure 4 shows the full sensitivity sweep across all six workloads. Workloads with $\lambda > 0.7$ (stencil, training) are essentially flat — their efficiency is dominated by η_{refresh} and bus turnaround, not row-scheduling. Workloads with $\lambda < 0.3$ show steep dependence on $t_{\text{RRD,S}}$: every cycle removed directly translates to row-activation throughput.

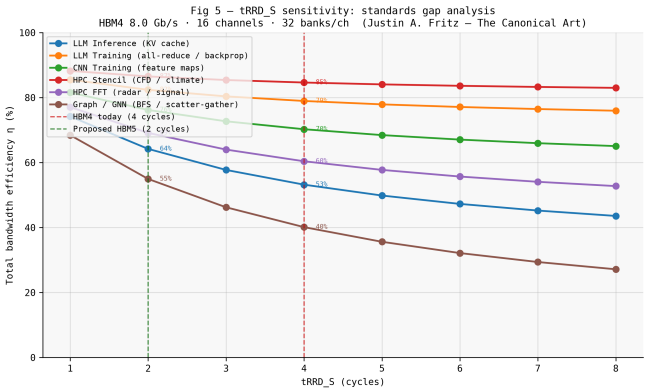

Figure 4: Bandwidth efficiency vs $t_{\mathrm{RRD,S}}$ (cycles). HBM4 8.0 Gb/s, 16 channels, 4 bank groups. Red dashed: HBM4 today (4 cycles). Green dashed: proposed HBM5 target (2 cycles). High-locality workloads are insensitive; low-locality (graph BFS, LLM inference) show the largest gain.

Table 4: Impact of $t_{\mathrm{RRD,S}}$ reduction: HBM4 8.0 Gb/s, 4 bank groups.

Workload	t_S=4	t_S=2	Gain
Graph BFS / GNN	40.1%	55.0%	+37%
LLM Inference	53.2%	64.2%	+21%
HPC FFT	60.4%	69.1%	+14%
CNN Training	70.3%	76.2%	+8%
LLM Training	79.0%	82.4%	+4%
HPC Stencil	84.6%	86.5%	+2%

The improvement is largest for the workloads with lowest locality. Stencil codes, which are already efficient, are barely affected. This is the ideal property for a spec change: it helps the workloads that need help and does not change behavior for workloads that are already well-served.

Reducing $t_{\mathrm{RRD,S}}$ shifts scheduling complexity to the memory controller: the ACT scheduler must track bank-group state at higher frequency and maintain tighter row-hit windows. We note this cost but do not model controller area or power overhead, which is outside the scope of the memory interface model and would require circuit-level characterization as part of a JC-42 proposal.

7.3 Physical Feasibility

$t_{\mathrm{RRD,S}}$ is set by the minimum safe interval between row decoder activations in different bank groups. The constraint arises from:

1. *Power supply noise*: Simultaneous row decoder activations in adjacent bank groups cause di/dt transients on the VDDQ supply. $t_{\mathrm{RRD,S}}$ provides settling time.

2. *Sense amplifier recovery*: In some DRAM cell architectures, cross-bank bit line coupling requires a minimum quiescent interval before a second sense amplifier can be activated.

HBM4's 3D-stacked architecture provides physical isolation between bank groups that conventional planar DRAM does not: TSV-separated bank groups have independent VDDQ distribution networks and lower coupling capacitance between adjacent bitline arrays. This suggests that $t_{\mathrm{RRD,S}} = 2$ cycles is physically achievable in a next-generation HBM process, though formal characterization across PVT corners would be required as part of a JC-42 proposal.

7.4 Context: HBM4 Architectural Precedent

HBM4 already demonstrates that spec-level architectural changes can fix efficiency bottlenecks without data rate changes. The introduction of the dual AWORD/DWORD command bus in HBM4 eliminated the CA-bus contention that cost HBM2e up to 25% efficiency for random-access workloads. η_{cmdbus} moved from 0.76 to 1.00 as a direct result of this spec change.

The $t_{\mathrm{RRD,S}}$ reduction is the analogous fix for the next generation: address the scheduling bottleneck that limits the workloads we actually care about, without requiring the die rearchitecture that higher data rates demand.

7.5 Broader Standards Trajectory

Looking past $t_{\mathrm{RRD,S}}$, the model identifies a hierarchy of spec changes by impact per implementation complexity:

1. $t_{\mathrm{RRD,S}} = 2$ cycles: +14–37% for random-access workloads. Low die complexity change, high algorithmic impact.

2. Extended burst length option ($BL = 16$ or 32) for sequential kernels: reduces command bus overhead for HPC stencil and training, marginal for graph.

3. Adaptive refresh: suspend refresh during sustained sequential bursts, accepting reduced retention margin in exchange for +1–2% efficiency under high load. Already partially specified in LPDDR5 [17]; applicable to HBM.

4. Near-memory compute (NMC) hooks: expose per-bank processing elements that can execute atomic read-modify-write operations without traversing the full die-to-die interface, eliminating T_{access} from the RMW loop for graph workloads entirely. This is a longer-horizon architectural change beyond the scope of the current JESD270 spec family.

8 Conclusion

The gap between rated and delivered memory bandwidth is not a measurement artifact — it is a predictable consequence of DRAM timing constraints that do not scale with data rate. We have presented a unified four-dimension cost model (bandwidth efficiency, latency, energy, area) for die-to-die HBM interfaces, grounded in JEDEC JESD235C and JESD270-4A timing parameters and corroborated by published workload characterizations.

The central finding is that graph traversal and LLM inference — the dominant memory-bandwidth consumers in contemporary AI infrastructure — achieve only 34–53% effective bandwidth efficiency on HBM4 at 8.0 Gb/s. The bottleneck is $t_{\mathrm{RRD,S}}$: the minimum row-to-row activation delay. At HBM4 speeds, the data bus can transfer data faster than the scheduler can issue the ACT commands that open rows for data access.

Counterintuitively, HBM4's higher data rate reduces random-access efficiency relative to HBM2e. t_{col} halved from 4 to 2 cycles while $t_{\mathrm{RRD,S}}$ held at $\sim$2 ns absolute (4 cycles at $f_{CK} = 2$ GHz). The data bus became faster; the row scheduler did not.

Algorithm designers have concrete, high-leverage responses: DRAM-row-aligned data layouts, graph edge reordering for locality, KV-cache transposition for inference, and BFS frontier batching to expose row-activation parallelism. The crossover analysis in Section 6 gives SoC architects a principled basis for sizing on-chip SRAM for latency-sensitive kernels that would otherwise pay the full $T_{\mathrm{access}} \approx 12.5$ ns HBM4 page-miss penalty on every operation.

The standards implication is specific: reducing $t_{\mathrm{RRD,S}}$ from 4 to 2 cycles in the HBM5 JEDEC JC-42 specification delivers +37% effective bandwidth to graph workloads and +21% to LLM inference. This is the largest per-complexity-unit improvement available to the specification, and it targets precisely the workloads that HBM's economics depend on.

All simulation code, JEDEC-grounded spec objects, and pedagogical Octave derivations are available at https://github.com/quantumcelnav/hbm-phy-compiler. The model is designed to be extended: workload profiles, timing overrides, and bank configurations are first-class parameters.

Acknowledgments. The author thanks the JEDEC JC-42 working group for maintaining public access to the HBM2e and HBM4 specification documents that ground every parameter in this model.

References

[1] W. A. Wulf and S. A. McKee, "Hitting the memory wall: Implications of the obvious," *ACM SIGARCH Computer Architecture News*, vol. 23, no. 1, pp. 20–24, 1995.

[2] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann, 6th ed., 2017.

[3] L. McMurchie and C. Ebeling, "Performance-driven routing," in *Proceedings of the International Symposium on Field-Programmable Gate Arrays (FPGA)*, pp. 80–89, 1995.

[4] S. Williams, A. Waterman, and D. Patterson, "Roofline: An insightful visual performance model for multicore architectures," *Communications of the ACM*, vol. 52, pp. 65–76, 2009.

[5] S. Rixner, W. J. Dally, U. J. Kapasi, P. Mattson, and J. D. Owens, "Memory access scheduling," in *Proceedings of the 27th Annual International Symposium on Computer Architecture (ISCA)*, pp. 128–138, ACM, 2000.

[6] J. Hestness, S. Narang, N. Ardalani, G. Diamos, H. Jun, H. Kianinejad, M. M. A. Patwary, Y. Yang, and Y. Zhou, "Deep learning benchmarking suite," in *Workshop at NIPS*, 2015.

[7] L. Nai, R. Hadidi, J. Sim, H. Kim, P. Kumar, and H. Kim, "Graphpim: Enabling instruction-level pim offloading in graph computing frameworks," in *IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 457–468, 2017.

[8] S. Kwon *et al.*, "A 20nm 6GB function-in-memory DRAM, based on HBM2 with a 1.2 TFLOPS programmable computing unit using bank-level parallelism, for machine learning applications," in *IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 1–3, 2021.

[9] D. Lee, Y. Kim, H. Hassan, and O. Mutlu, "Simultaneous multi-layer access: Improving 3d-stacked memory bandwidth at low cost," *ACM Transactions on Architecture and Code Optimization*, vol. 12, pp. 1–29, 2016.

[10] J. D. McCalpin, "Stream: Sustainable memory bandwidth in high performance computers." https://www.cs.virginia.edu/stream/, 2010.

[11] MLCommons, "Mlperf training v3.1 results." https://mlcommons.org/benchmarks/training/, 2023.

[12] Graph500 Steering Committee, "Graph500 benchmark specification." https://graph500.org/, 2021.

[13] Y. Chi, G. Dai, Y. Wang, G. Sun, G. Li, and H. Yang, "Nxgraph: An efficient graph processing system on a single machine," in *32nd IEEE International Conference on Data Engineering (ICDE)*, pp. 409–420, 2016.

[14] J. E. Gonzalez, R. S. Xin, A. Dave, D. Crankshaw, M. J. Franklin, and I. Stoica, "Graphx: Unifying data-parallel and graph-parallel analytics." USENIX OSDI, 2014.

[15] T. Dao, D. Y. Fu, S. Ermon, A. Rudra, and C. Ré, "Flashattention: Fast and memory-efficient exact attention with io-awareness," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[16] S. Beamer, K. Asanovic, and D. Patterson, "Direction-optimizing breadth-first search," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC12)*, 2012.

[17] JEDEC Solid State Technology Association, "Low power double data rate 5 (LPDDR5): JESD209-5B." JEDEC Standard, 2022.