Year 1858: When the vast ocean symbolically shrunk into a “channel”



H.M.S. Agamemnon Laying the Atlantic Telegraph Cable in 1858 while a whale crosses the line.

Early Silicon of Raptor

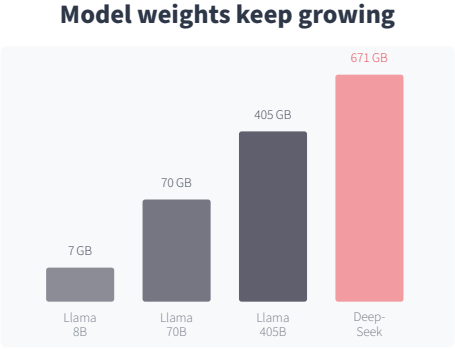
The First 3D-DRAM Accelerator for Generative Inference †

Prashant J. Nair * – d-Matrix Inc. and UBC
and d-Matrix Team

* Also affiliated as an Associate Professor at the University of British Columbia (UBC).
† Work done during sabbatical/leave from UBC.

1st July 2026

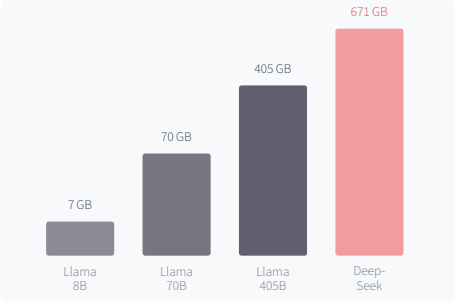
The Growing Data Problem



INT8 weights. Larger models need more cards.

The Growing Data Problem

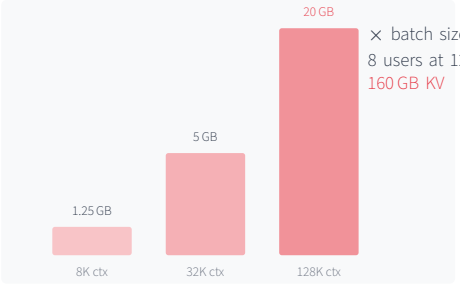
Model weights keep growing



INT8 weights. Larger models need more cards.

KV cache grows with context

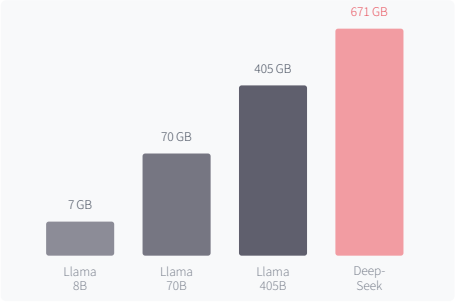
(Llama-3.1 70B, INT8, per user)



Longer context × more users ⇒ KV cache dominates.

The Growing Data Problem

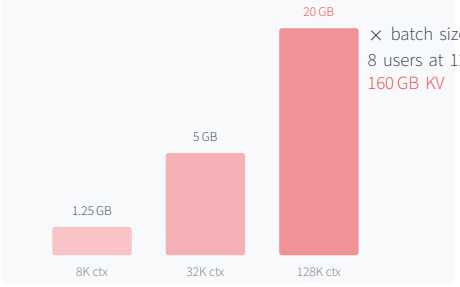
Model weights keep growing



INT8 weights. Larger models need more cards.

KV cache grows with context

(Llama-3.1 70B, INT8, per user)



Longer context × more users ⇒ KV cache dominates.

Weights and KV cache present a **capacity and bandwidth problem**. Both of these are growing.

The decode bottleneck is fundamental. Batching cannot solve it. Parallelism taxes it further.

The question becomes:

What kind of memory do we actually need?

Fast enough to feed the compute. Large enough to hold the model.

Today’s Memory Technologies

SRAM: Bandwidth Advantage

Why SRAM is fast:

- 6T cell with direct bit-line access
- On-die: no interposer, no package crossing
- Sub-nanosecond access latency

Specs (on-die, per card):

Bandwidth	**150 TB/s**
Capacity	**4 GB**
Latency	∼1 ns
I/O energy	∼0.5 pJ/bit

SRAM: Bandwidth Advantage

Why SRAM is fast:

- 6T cell with direct bit-line access
- On-die: no interposer, no package crossing
- Sub-nanosecond access latency

Why SRAM cannot scale:

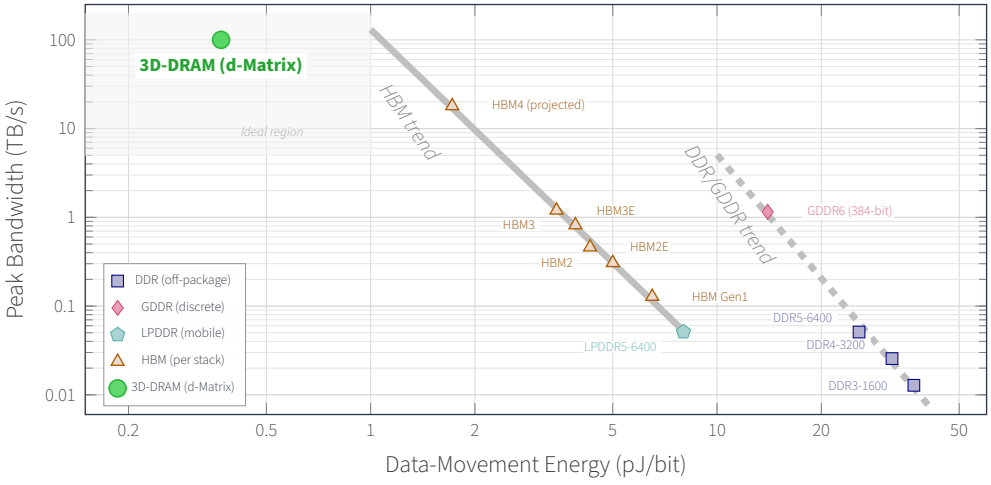
- 6T cell is **100–150× larger** than DRAM 1T1C
- Leakage: ∼1 nW/bit (significant at GB scale)
- Practical limit: ∼**4 GB** per card
- Cost: ∼$1000/GB vs. ∼$10/GB for DRAM

Specs (on-die, per card):

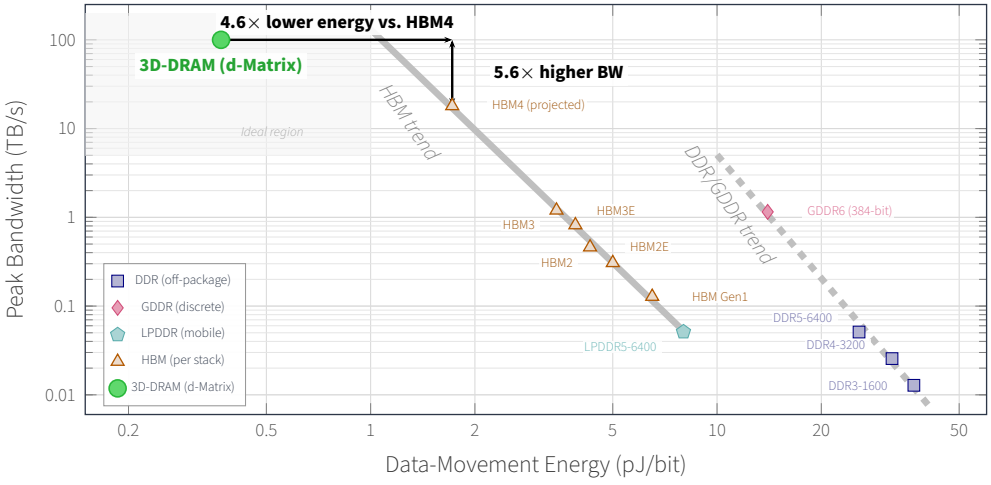
Bandwidth	**150 TB/s**
Capacity	**4 GB**
Latency	∼1 ns
I/O energy	∼0.5 pJ/bit

Very fast memory ⇒ but **4 GB** cannot hold a 70B model.

DRAM Landscape: The Power-Normalized Bandwidth (PN) Inversion Law

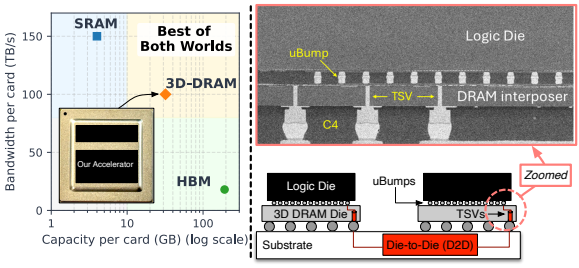


DRAM Landscape: The Power-Normalized Bandwidth (PN) Inversion Law



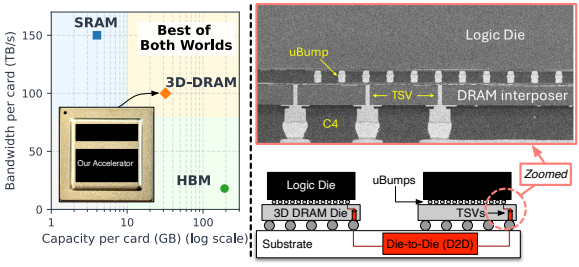
3D-DRAM: The Opportunity

Commercial 3D-DRAM-Based Accelerator for Gen-AI



Logic die bonded face-to-face to 3D-DRAM die.
Vertical interconnects replace mm-scale HBM traces.

Commercial 3D-DRAM-Based Accelerator for Gen-AI



Logic die bonded face-to-face to 3D-DRAM die.
Vertical interconnects replace mm-scale HBM traces.

	HBM	3D-DRAM
Topology	2.5D (beside)	F2F (below)
I/O energy	2–3 pJ/bit	0.37 pJ/bit
BW / card	18 TB/s	100+ TB/s
Capacity	192 GB	32 GB

5.6× higher BW

5× lower energy/bit

7× denser I/O

Integration

The 3D-DRAM Integration Landscape

Hardware	Architecture	Co-Design
TSV density vs. routing congestion trade-offs	Bank-to-channel mapping: 840 banks $\neq 256 \times 4$	Thermal: T_j=105°C $\Rightarrow$ 8× faster refresh
Power delivery through 3D stack (IR drop, decap)	Weight & KV-cache tiling across banked memory	ECC, DBI metadata, & scrub share bank rows
Signal integrity: crosstalk & SSN on wide parallel bus	Redundancy without breaking channel symmetry	Refresh scheduling vs. compute pipeline bubbles
I/O power: no burst structure $\Rightarrow$ no PHY DBI	LPDDR5X as secondary tier: data placement policy	Known-good-die testing $\rightarrow$ post-package repair flow
μBump yield & alignment at 36 μm pitch	Compiler support for stream-aware access patterns	Multi-high stacking: routing, power, thermals
Clock distribution & skew across die stack	Runtime KV-cache alloc & eviction across banks	Hybrid bonding roadmap: <2 μm pitch, 8-high

The 3D-DRAM Integration Landscape

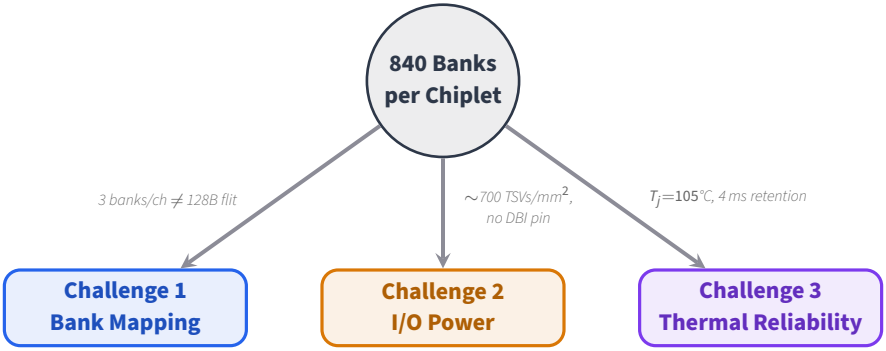
Hardware	Architecture	Co-Design
TSV density vs. routing congestion trade-offs	★ **Bank-to-channel mapping: 840 banks $\neq$ 256$\times$4**	★ **Thermal: T_j=105°C $\Rightarrow$ 8$\times$ faster refresh**
Power delivery through 3D stack (IR drop, decap)	Weight & KV-cache tiling across banked memory	ECC, DBI metadata, & scrub share bank rows
Signal integrity: crosstalk & SSN on wide parallel bus	★ **Redundancy without breaking channel symmetry**	Refresh scheduling vs. compute pipeline bubbles
★ **I/O power: no burst structure $\Rightarrow$ no PHY DBI**	LPDDR5X as secondary tier: data placement policy	Known-good-die testing $\rightarrow$ post-package repair flow
μBump yield & alignment at 36 μm pitch	Compiler support for stream-aware access patterns	Multi-high stacking: routing, power, thermals
Clock distribution & skew across die stack	Runtime KV-cache alloc & eviction across banks	Hybrid bonding roadmap: $<$2 μm pitch, 8-high

The 3D-DRAM Integration Landscape

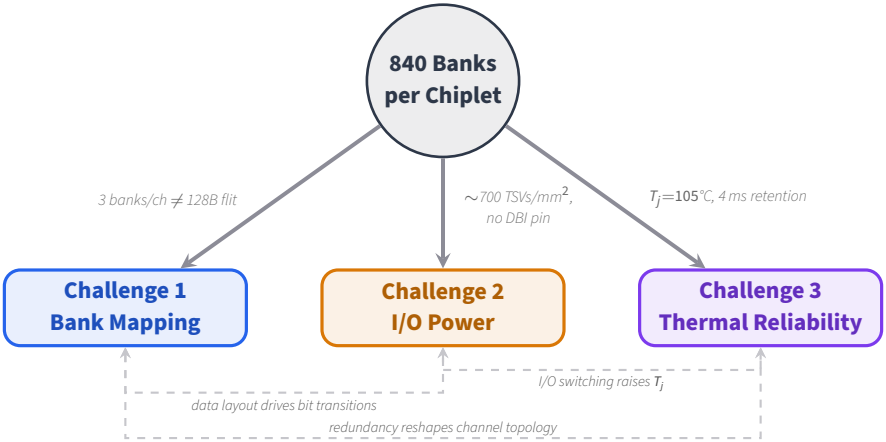
Hardware	Architecture	Co-Design
TSV density vs. routing congestion trade-offs	★ **Bank-to-channel mapping: 840 banks $\neq$ 256$\times$4**	★ **Thermal: T_j=105°C $\Rightarrow$ 8$\times$ faster refresh**
Power delivery through 3D stack (IR drop, decap)	Weight & KV-cache tiling across banked memory	ECC, DBI metadata, & scrub share bank rows
Signal integrity: crosstalk & SSN on wide parallel bus	★ **Redundancy without breaking channel symmetry**	Refresh scheduling vs. compute pipeline bubbles
★ **I/O power: no burst structure $\Rightarrow$ no PHY DBI**	LPDDR5X as secondary tier: data placement policy	Known-good-die testing $\rightarrow$ post-package repair flow
μBump yield & alignment at 36 μm pitch	Compiler support for stream-aware access patterns	Multi-high stacking: routing, power, thermals
Clock distribution & skew across die stack	Runtime KV-cache alloc & eviction across banks	Hybrid bonding roadmap: $<$2 μm pitch, 8-high

★ The **challenges** we discuss in this talk

These Challenges Are Entangled



These Challenges Are Entangled



A solution to any one challenge **constrains the design space of the other two.**

Challenge 1: Bank Mapping

Challenge 1: The Bank-to-Channel Mapping Problem



The setup

- Each TE needs a **128 B flit** per access
- 16 ch × 16 TE-Groups = **256 ch/chiplet**
- Each bank delivers **32 B** per column access

Challenge 1: The Bank-to-Channel Mapping Problem



The setup

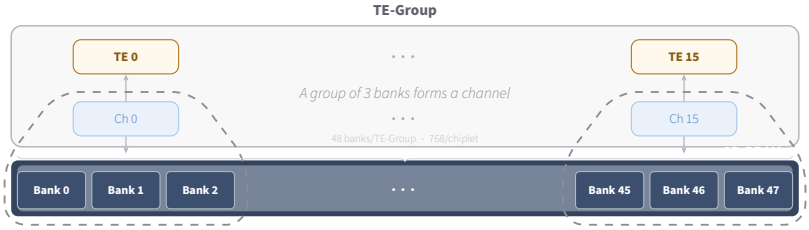
- Each TE needs a **128 B flit** per access
- 16 ch × 16 TE-Groups = **256 ch/chiplet**
- Each bank delivers **32 B** per column access

The arithmetic

- 128 B ÷ 32 B = **4** banks/ch *needed*
- DRAM die: **840 banks** (768 after 72 spares)
- 768 ÷ 256 = **3** banks/ch *available*

Need 4 banks/channel ⇒ have only 3 ⇒ **the 128 B flit does not divide evenly**

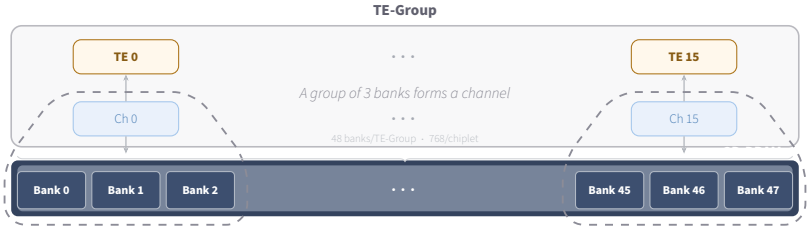
Challenge 1: The Overfetch Dilemma



With 3 banks per channel:

- One channel access returns **96 B** (3 × 32 B)
- 128 B flit ⇒ two accesses ⇒ 192 B fetched
- **33% overfetch ≈ 33 TB/s wasted**

Challenge 1: The Overfetch Dilemma



With 3 banks per channel:

- One channel access returns **96 B** (3 × 32 B)
- 128 B flit ⇒ two accesses ⇒ 192 B fetched
- **33% overfetch ≈ 33 TB/s wasted**

Naïve fix — column staggering:

- Stagger column indices to pack one flit per pair
- Requires a **192 B shifting buffer**
- Complicates timing closure and verification

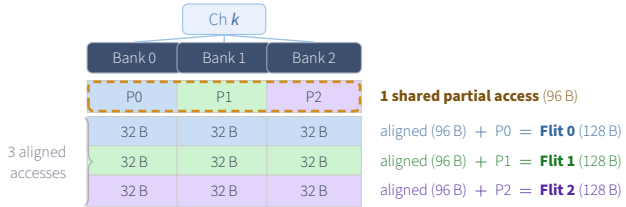
How to deliver 128 B from 3 × 32 B banks **without wasting bandwidth or adding complex datapaths?**

Challenge 1 ✓ Solution: Stream Blocking

Challenge. A 128 B flit = **4×32 B**, but each channel has only **3 banks** — a naïve fetch reads 192 B per 128 B flit (⇒ **33% overfetch**).

Challenge 1 ✓ Solution: Stream Blocking

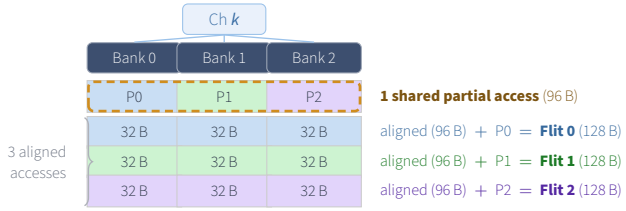
Challenge. A 128 B flit = **4×32 B**, but each channel has only **3 banks** — a naïve fetch reads 192 B per 128 B flit (⇒ **33% overfetch**).



Puts the 4th 32 B (the *partial*) into one access *shared* by 3 flits — $4 \times 96 = 384$ B in $= 3 \times 128 = 384$ B out ⇒ **0% overfetch**.

Challenge 1 ✓ Solution: Stream Blocking

Challenge. A 128 B flit = **4×32 B**, but each channel has only **3 banks** — a naïve fetch reads 192 B per 128 B flit (⇒ **33% overfetch**).



Puts the 4th 32 B (the *partial*) into one access *shared* by 3 flits — $4 \times 96 = 384$ B in $= 3 \times 128 = 384$ B out ⇒ **0% overfetch**.

Every column access is fully utilised — the **33 TB/s** lost to overfetch is reclaimed, with **no shifting network**.

Challenge 2: I/O Power at Scale

Challenge 2: The I/O Power Wall

The power cost of 100 TB/s:

Measured I/O Power

$P_{I/O} = 100\ \text{TB/s} \times 0.37\ \text{pJ/bit} = \mathbf{296}\ \text{W}$

Why conventional DBI cannot help:

- HBM uses multi-cycle bursts (BL 4–16)
- PHY sees full burst ⇒ decides on inversion
- DBI pin per byte lane signals choice

Our 3D-DRAM has **none of these**: single-cycle transfer, no burst, no sideband pin.

Challenge 2: The I/O Power Wall

The power cost of 100 TB/s:

Measured I/O Power

$P_{I/O} = 100$ TB/s $\times$ 0.37 pJ/bit $=$ **296** W

Why conventional DBI cannot help:

- HBM uses multi-cycle bursts (BL 4–16)
- PHY sees full burst ⇒ decides on inversion
- DBI pin per byte lane signals choice

Our 3D-DRAM has **none of these**: single-cycle transfer, no burst, no sideband pin.

Interface comparison:

	DDR / HBM
Transfer	Multi-cycle burst
Lookahead	Full burst visible
DBI	Dedicated pin/lane

	Our 3D-DRAM
Transfer	Single-cycle 256-bit
Lookahead	**None**
DBI	**No pin. No sideband.**

DBI would save 20%

Need to rethink interface.

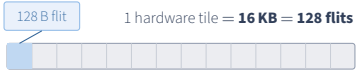
How to achieve DBI **architecturally?**

Challenge 2 ✓ Solution: Stream Flipping (Pinless DBI)

Challenge. I/O dominates: **100** TB/s × **0.37** pJ/bit = **296 W**, and conventional DBI needs a **sideband pin** the single-cycle 3D-DRAM link doesn't have.

Challenge 2 ✓ Solution: Stream Flipping (Pinless DBI)

Challenge. I/O dominates: **100** TB/s × **0.37** pJ/bit = **296 W**, and conventional DBI needs a **sideband pin** the single-cycle 3D-DRAM link doesn't have.



⇒ **long** per-channel flit streams; each flit compared to the *previous* one.

(a) Without DBI

flit $i-1$: 0 0 · · · 0
flit i: 1 1 · · · 1
flit $i+1$: 0 0 · · · 0

→ every bit toggles twice

(b) Stream flipping: invert flit i

0 0 · · · 0	**0**
0 0 · · · 0 (was 1 · · · 1)	**1**
0 0 · · · 0	**0**

meta

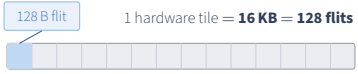
→ ≈0 toggles + 1 metadata bit / flit

Architectural DBI, no pin:

- **Write:** compare each flit to the previous → invert.
- **Read:** fetch the 1-bit tag; invert if set
- Tag co-located with ECC ⇒ **0.8%** overhead, **no PHY change → 20% savings.**

Challenge 2 ✓ Solution: Stream Flipping (Pinless DBI)

Challenge. I/O dominates: **100** TB/s × **0.37** pJ/bit = **296 W**, and conventional DBI needs a **sideband pin** the single-cycle 3D-DRAM link doesn't have.



⇒ **long** per-channel flit streams; each flit compared to the *previous* one.

(a) Without DBI

flit $i-1$: 0 0 · · · 0
flit i: 1 1 · · · 1
flit $i+1$: 0 0 · · · 0

→ every bit toggles twice

(b) Stream flipping: invert flit i

0 0 · · · 0	**0**
0 0 · · · 0 (was 1 · · · 1)	**1**
0 0 · · · 0	**0**

meta

→ ≈0 toggles + 1 metadata bit / flit

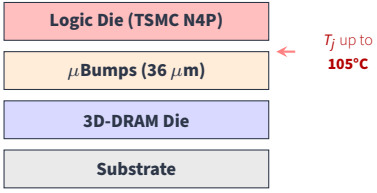
Architectural DBI, no pin:

- **Write:** compare each flit to the previous → invert.
- **Read:** fetch the 1-bit tag; invert if set
- Tag co-located with ECC ⇒ **0.8%** overhead, **no PHY change → 20% savings.**

The full **20%** I/O saving with **no pin and no PHY change**.

Challenge 3: Reliability Under Thermal Stress

Challenge 3: DRAM Reliability at 105°C



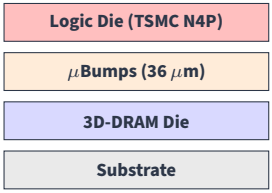
Retention consequence

Standard: 32 ms at 85°C

At 105°C: drops to **4 ms**

⇒ **8**× more frequent refresh

Challenge 3: DRAM Reliability at 105°C



Retention consequence

Standard: 32 ms at 85°C

At 105°C: drops to **4 ms**

⇒ **8**× more frequent refresh

Three compounding problems:

1. Yield — 840 banks/die. Even a 1% fault rate risks losing entire channels. Discarding bonded dies is uneconomical.

2. Symmetry — Disabling a faulty bank narrows its channel. Tensor engines expect *uniform* channel widths—one weak channel throttles the entire slice.

3. Error accumulation — Higher T_j drives more soft errors. ECC, scrub, and refresh must all coexist without stalling throughput.

How to maintain full-bandwidth, symmetric channels when banks fail and refresh is **8**× nominal?

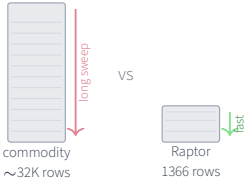
Challenge 3 ✓ Solution: Thermal-Aware Refresh & ECC

Challenge. At **105°C**, retention falls 32 ms → 4 ms (**8**× more refresh), and bits still need correcting — all *without losing significant bandwidth*.

Challenge 3 ✓ Solution: Thermal-Aware Refresh & ECC

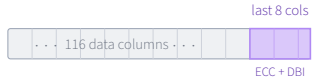
Challenge. At **105°C**, retention falls 32 ms → 4 ms (**8**× more refresh), and bits still need correcting — all *without losing significant bandwidth.*

Deep banking makes refresh cheap



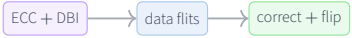
16–32× fewer rows ⇒ 4 ms refresh (8× more frequent) costs only **1.37%** → **∼103 TB/s**.

Interleaved ECC, co-located with DBI



Last **8 of 124** columns: paired **[132,128]** Reed–Solomon + DBI bits, interleaved with the subarray.

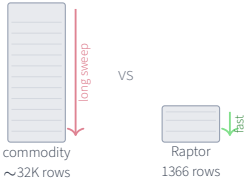
Read order:



Challenge 3 ✓ Solution: Thermal-Aware Refresh & ECC

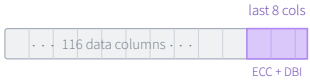
Challenge. At **105°C**, retention falls 32 ms → 4 ms (**8**× more refresh), and bits still need correcting — all *without losing significant bandwidth.*

Deep banking makes refresh cheap



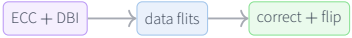
16–32× fewer rows ⇒ 4 ms refresh (8× more frequent) costs only **1.37%** → **~103 TB/s**.

Interleaved ECC, co-located with DBI



Last **8 of 124** columns: paired **[132,128]** Reed–Solomon + DBI bits, interleaved with the subarray.

Read order:



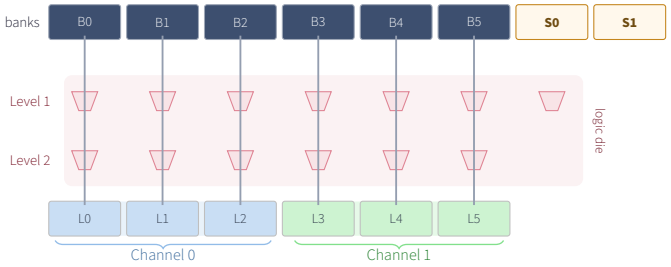
Deep banking turns an 8× refresh penalty into only a **<1.4% bandwidth loss**

Challenge 3 ✓ Solution: Bank Chaining

Challenge. The **72 spare banks** must cover faults *anywhere* on the die — without making channels asymmetric or adding long edge wires.

Challenge 3 ✓ Solution: Bank Chaining

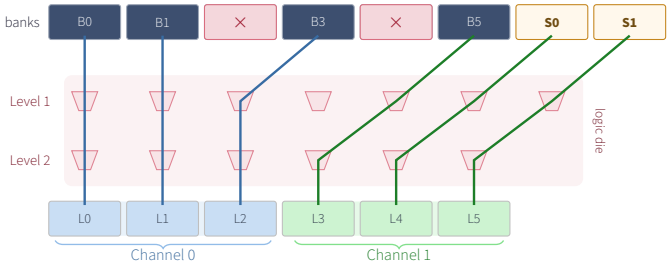
Challenge. The **72 spare banks** must cover faults *anywhere* on the die — without making channels asymmetric or adding long edge wires.



No faults — both mux levels pass each bank straight to its slot.

Challenge 3 ✓ Solution: Bank Chaining

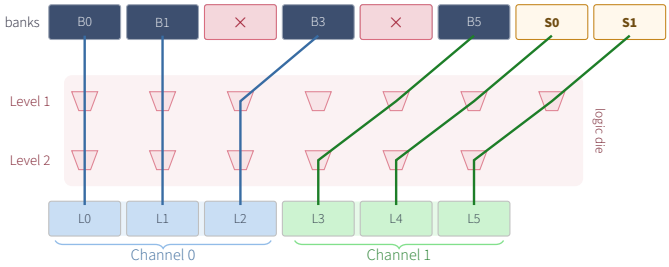
Challenge. The **72 spare banks** must cover faults *anywhere* on the die — without making channels asymmetric or adding long edge wires.



*Level 1 skips the first fault (B2); Level 2 skips the second (B4) — **M=2 physical mux levels tolerate 2 faults anywhere**, spares backfill, channels stay symmetric.*

Challenge 3 ✓ Solution: Bank Chaining

Challenge. The **72 spare banks** must cover faults *anywhere* on the die — without making channels asymmetric or adding long edge wires.



*Level 1 skips the first fault (B2); Level 2 skips the second (B4) — **M=2 physical mux levels tolerate 2 faults anywhere**, spares backfill, channels stay symmetric.*

The chiplet's **72 in-line redundant macros** absorb arbitrary faults — symmetric channels, negligible routing cost.

Performance Results

Experimental Setup

Hardware configurations:

	BW	Capacity
SRAM	150 TB/s	4 GB
HBM	18 TB/s	192 GB
3D-DRAM	100 TB/s	32 GB

All configurations: 10 PFLOP/s tensor compute

Experimental Setup

Hardware configurations:

	BW	Capacity
SRAM	150 TB/s	4 GB
HBM	18 TB/s	192 GB
3D-DRAM	100 TB/s	32 GB

All configurations: 10 PFLOP/s tensor compute

Workloads and Deployment:

- **Dense LLMs:** Llama-3.1 70B
- **MoE:** DeepSeek-V3 (671B), GPT-OSS (20B, 120B), Kimi K2
- **Speech:** Whisper, Canary-1B
- **Context Lengths:** 4K and 64K

Headline Results

4.71×

higher TPS/card
vs HBM

Headline Results

4.71×
higher TPS/card
vs HBM

2.44×
higher TPS/card
vs SRAM

Headline Results

4.71×
higher TPS/card
vs HBM

2.44×
higher TPS/card
vs SRAM

9.96×
lower TPOT
vs HBM

Headline Results

4.71×
higher TPS/card
vs HBM

2.44×
higher TPS/card
vs SRAM

9.96×
lower TPOT
vs HBM

Averaged across all workloads
At 4K context, batch size 32, 0.5 μs network latency, 1 TB/s network bandwidth.

Conclusions

Our contribution

3D-DRAM puts bandwidth *and* capacity in one package → three co-designed solutions.

Raptor 3D-DRAM vs. HBM4 & Rubin R200 silicon-area basis

Metric	Raptor 3D-DRAM	HBM4 24 Gb	HBM4 32 Gb	Rubin R200
Capacity (MB/mm²)	**11.4**	21.9	26.3	21.9
Bandwidth (GB/s/mm²)	**32.6**	1.67	1.51	1.39
Power per GB/s (mW/GB/s)	**2.96**	40.0	40.0	40.0

Effective-BW basis (Raptor 83%, others 100% util.). Power per GB/s: lower is better. Rubin R200 = 8× HBM4 24 Gb SoC.

Conclusions

Our contribution

3D-DRAM puts bandwidth *and* capacity in one package → three co-designed solutions.

Raptor 3D-DRAM vs. HBM4 & Rubin R200 silicon-area basis

Metric	Raptor 3D-DRAM	HBM4 24 Gb	HBM4 32 Gb	Rubin R200
Capacity (MB/mm²)	**11.4**	21.9	26.3	21.9
Bandwidth (GB/s/mm²)	**32.6**	1.67	1.51	1.39
Power per GB/s (mW/GB/s)	**2.96**	40.0	40.0	40.0

Effective-BW basis (Raptor 83%, others 100% util.). Power per GB/s: lower is better. Rubin R200 = 8× HBM4 24 Gb SoC.

≈20× the bandwidth per mm² and 13.5× better power per GB/s.

The memory wall is a technology problem. **3D-DRAM is the solution.**

Early Silicon of Raptor

The First 3D-DRAM Accelerator for Generative Inference

Thank You