# Frozen Weights, Fast ROM: Read-Only Memory as the Substrate for Frontier LLMs

*Master edition combining "Frozen Weights, Fast Silicon" and "Fast ROM at RAM Speed," with a field-tested objection analysis — Michael Snyder (Pic2Mag), August 2026*

**Abstract.** Deployed large language models read their weights but never write them, yet those immutable weights are stored in the scarcest, most expensive memory technology available (HBM/DRAM). This master edition combines two earlier papers into a single argument. First, the frozen-weights proposal: burn a model's base weights into cheap mask ROM — packaged as changeable cartridges or daughter cards — keep a small LoRA-style diff in dynamic memory, and let DRAM hold what is actually dynamic. Second, the rebuttal to the standard objection: "ROM is too slow" confuses a product category with a physical limit. Memory speed lives in the interface periphery and the packaging, not in the storage element; ROM never hit a speed wall — it was abandoned at the frontier in the 1980s, when DRAM proved able to both read and write. Built with today's interfaces, ROM runs at RAM speed by construction, and — freed of capacitors, refresh, and write paths — it is denser and lower-power per die. Objections collected from public field-testing are analyzed with responses. Related work — LoRA, Apple's flash streaming, and Taalas's hardwired-LLM ASIC (acquired by AMD in 2026) — shows each piece of the architecture already in production; this paper assembles them.

## 1. The Observation: Trained Weights Are Read-Only

Once training finishes, a frontier model's weights — including its mixture-of-experts centers — never change again until a new version is trained. Everything the model knows, from the bronze age to its training cutoff, is frozen. Yet every inference server today stores that permanent content in dynamic RAM, paying DRAM and HBM prices, power, and packaging costs for data that is written exactly once. Viewed from the memory system, a deployed LLM is a **read-only publication**, and it deserves a read-only medium. The industry's "fast memory log jam" — years of HBM scarcity and pricing pressure — is partly self-inflicted: we are spending our best memory on our least dynamic data.

Once you realize that the weights and mixture-of-experts centers are permanent memory that never changes, the question stops being "how do we get more fast memory" and becomes "why is permanent data sitting in rewriteable memory at all?" The rest of this paper is that question, taken seriously.

## 2. The Encyclopedia Model: ROM Base + Live Diff

A print encyclopedia ships as twenty permanent volumes plus a small yearly update volume. The same split applies to models: ship the base weights as ROM "volumes," and ship a small diff file that lives in dynamic VRAM and patches the ROM image on the fly — re-issue the diff whenever the model is corrected or refreshed. This is not speculative: **LoRA already is the diff mechanism** [3]. Low-rank adaptation freezes the base weight matrix W and learns a tiny low-rank delta $\Delta W = BA$ (often <1% of the parameters), which is merged or applied at inference with no added latency. A ROM+diff architecture is therefore LoRA with the base matrix moved from DRAM into silicon: the diff in DRAM stays small precisely because the ROM base is allowed to be huge and cheap.

One more property of the analogy matters: **encyclopedia sets never go out of date**. A 1990 edition remains a correct record of what humanity knew in 1990. A model cartridge is the same kind of publication — everything from the bronze age to the training cutoff, frozen correctly — and the medium itself is archival. Mask ROM stores data as physical geometry, not trapped charge: flash fights leakage for a decade or two, DRAM needs constant refresh, but a mask pattern is stable for centuries. In the same way an encyclopedia set never expires, an LLM weight ROM cartridge is good for the next thousand years — and the small yearly update volume rides in DRAM.

## 3. The Cartridge Model: The New Solution Is the Old Solution

The consumer industry solved the “permanent content, generic machine” problem decades before LLMs. From the Atari 2600 (1977) through the Game Boy and the Nintendo 64, consoles shipped permanent content on swappable mask-ROM cartridges while the hardware stayed general-purpose: the console was the platform, the cartridge was the publication. The new solution is the same as the old solution we used years ago. **Put a cartridge slot on the video card.** The GPU and its DRAM remain general-purpose, and the model ships as a cartridge you can hold in your hand — swap the cartridge, swap the model.

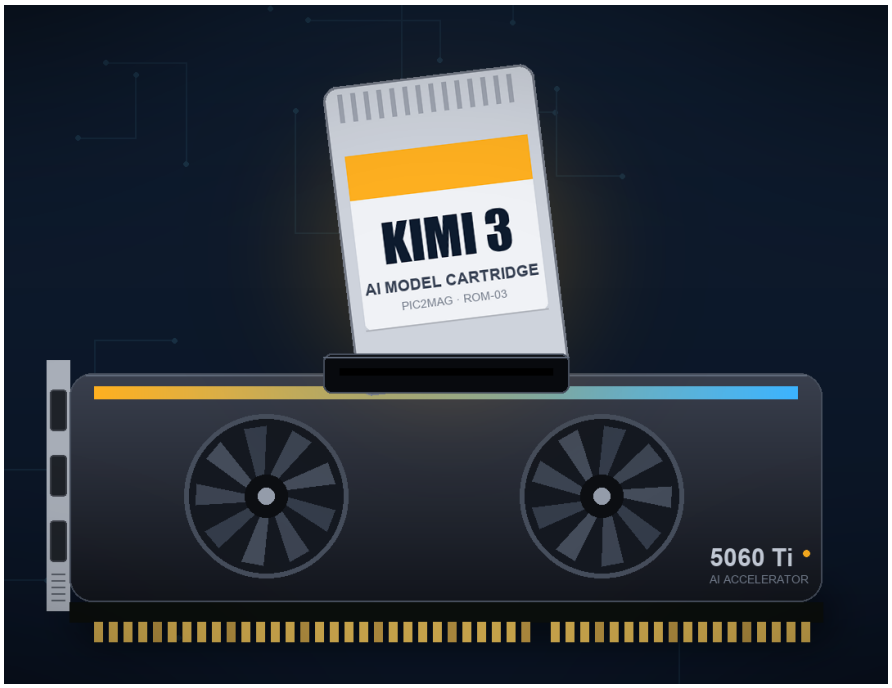


*Figure 1. The cartridge concept: a 5060 Ti-class video card with a changeable top-edge ROM cartridge (“KIMI 3”). The GPU and its DRAM stay generic; the model is the cartridge.*

The connector, notably, is not a research problem. Removable high-bandwidth buses have been standardized many times over: Thunderbolt 5 moves 80 Gbps bidirectionally (up to 120 Gbps in one direction) over an ordinary USB-C plug [9]; OCuLink (SFF-8611) breaks four lanes of PCI Express out to external devices; the enterprise SFF-8639 (U.2) connector has hot-swapped NVMe storage at full PCIe speed for a decade [10] — to say nothing of the PCI Express card edge itself. A cartridge interface is a choice among existing standards, not an invention.

## 4. Hardware Paths

**Diode-logic mask ROM.** A diode-matrix ROM is the oldest and simplest ROM: each stored bit is the presence or absence of a diode at a grid crossing. It uses very few transistors and is compatible with essentially every CMOS process of the last four decades, so production can run on mature, fully-depreciated fabs instead of competing with GPUs and HBM for leading-edge capacity. A key practical property of mask ROM is that the data pattern lives only in the upper metal/via layers: base wafers can be fabricated in advance and warehoused, then a new model release is committed with a small number of final masks — turnaround of weeks, not the months a full custom chip needs.

**Photonic logic** is the more glamorous alternative, but it is cutting-edge and would compete directly with fast-memory manufacturing for advanced capacity — the opposite of the goal.

**Mixture-of-experts tie-in.** MoE models activate only a few experts per token. Keep the hot experts (and the router) in DRAM and let cold experts live in ROM, swapped in on demand. Apple’s “LLM in a flash” showed that activation patterns are predictable enough to make this “windowing” work even over slow flash [2]; ROM with far better latency makes it easier still.

## 5. The Speed Question: A Wall That Never Was

The most common objection to this proposal deserves its own section. “ROM is too slow to feed a GPU” is repeated with great confidence and very little arithmetic. It is worth taking apart carefully.

### 5.1 The Category Error

“ROM is too slow.” Which ROM? The parts people picture are SPI NOR flash chips — a hundred to a few hundred megabytes per second — designed, on purpose, to be the cheapest possible place to park a BIOS image. Judging read-only memory by those parts is like judging storage by SD cards. Bandwidth is interface width × pin rate; latency is set by the peripheral pipeline and the distance to the consumer. The storage element barely enters the equation. A mask-ROM cell drives a bitline about as fast as a DRAM cell does, and it does so without precharging a capacitor first. No vendor has ever shipped a bandwidth-optimized ROM — not because it was impossible, but because until now there was no customer for one. “Too slow” is a description of a product line that was optimized for cost, not a property of the physics.

### 5.2 A Short, Correct History

Read-only memory was not always the slow sibling. Mask ROM was the program store of the 1970s — cartridges, BIOS, microcode — and it was as fast as anything else on the board. The divergence was economic, not physical. Intel’s 1103 (1970) made DRAM the commodity working memory because it could read *and* write; the 1702 EPROM (1971) began ROM’s long slide toward writability and cost — EPROM, then EEPROM, then Masuoka’s flash at Toshiba in the early 1980s [15][16]. Every rung of that ladder optimized cost-per-bit and convenience. Not one rung optimized bandwidth, because DRAM already had the bandwidth job. By the 1990s “memory” meant DRAM and “ROM” meant cheap code storage, and the roadmap for fast ROM simply stopped. Meanwhile, four decades of interface engineering — wide parallel buses, gigabit SerDes, 2.5D packaging — kept advancing in the service of RAM alone. Nothing hit a wall; everybody left the room. The tools that make HBM fast work exactly as well behind a ROM array, because they were never about the cells.

It is worth remembering how fast the old parts actually were. A Nintendo Entertainment System cartridge held mask ROMs that delivered a byte every CPU cycle, all day, for decades — and the console had no faster memory anywhere in the design. Arcade boards, synthesizers, and every PC BIOS through the 1980s ran their ROMs at full bus speed. The parts were never the bottleneck; they simply had no reason to get faster once DRAM took the performance seat.

### 5.3 Where Speed Actually Lives

Open an HBM stack and look for the speed. It is not in the DRAM cells: a cell read is an ordinary tens-of-nanoseconds affair, and a destructive one at that. The speed is in the periphery — a 1024-bit interface on HBM3 [12], doubled to 2048 bits on HBM4 [13], running at roughly 8 Gbps per pin across a silicon interposer a few millimeters from the GPU. Bandwidth = width × rate × proximity. None of those three factors cares whether the bit behind the wire is stored as charge on a capacitor or as the physical presence of a via. Put a ROM array behind the same periphery and you get the same bandwidth, full stop. And the door is open: the HBM interface is a public JEDEC standard, and the HBM4 base logic die is already fabricated by TSMC on a foundry logic process [14] — the industry has begun separating the memory stack from the memory maker. A read-only stack is the logical next step.

**A worked calculation.** Take the conservative end: a mask-ROM macro on a mature 28nm logic process, modestly clocked, behind a 64-bit DDR interface per chip — call it 4–8 GB/s per chip with no heroics. A daughter card carrying 64 such chips behind a simple aggregation switch presents 256–512 GB/s to its card-edge link, and 64 chips × 8 GB is half a terabyte of frozen weights. Now scale the interface, not the cell: wider buses and faster SerDes put the same card in NVLink territory. Nothing in this arithmetic touches the storage element. Every lever — width, rate, proximity, chip count — is interface engineering, and interface engineering is precisely the field that never stopped advancing.

### 5.4 The Delete-Gates Dividend

The rebuttal is not merely that ROM can match DRAM. Delete what ROM does not need and it wins on density and power:

- **No storage capacitor.** A DRAM cell is one transistor plus a capacitor, and that capacitor — a tall, high-aspect-ratio structure — is the hardest module in DRAM manufacturing. Mask ROM stores the bit as

geometry: a via present or absent. The cell is smaller, and it needs no exotic process. In fact, leading-edge logic nodes cannot build DRAM capacitors at all — foundries do not offer the module — so ROM is effectively the *only* dense memory you can put on a 3nm-class logic process.

• **No refresh.** DRAM leaks, so HBM spends logic, power, and several percent of its bus time on refresh, plus the rowhammer mitigation machinery modern parts require. All of it disappears in ROM.

• **Non-destructive reads.** A DRAM read destroys the stored charge, and the sense amplifiers must write it back every cycle. A ROM read disturbs nothing: simpler timing, less energy per bit, no restore path.

• **No write path.** Write drivers, write datapath, and most of the training and calibration logic are deleted outright. The periphery shrinks; the freed area becomes more array.

The net result: a read-only die holds **more** data than the fully functional RAM die of the same area — precisely because it does less — at lower power per bit read, at the same interface speed.

## 6. Two Form Factors

**6.1 Read-only HBM (the flagship).** Stack ROM dies on a logic base die speaking the JEDEC HBM protocol and place it in-package beside the GPU: identical bandwidth to HBM, zero refresh power, denser dies. Tens of gigabytes per stack puts an 8B–70B-class model at 4-bit precision in one or two stacks. The honest cost: this tier plays in advanced packaging, with its interposers and mask NRE — it is the premium product, not the cheap one.

**6.2 The ROM daughter card (the workhorse).** Fill a daughter card with mask-ROM chips and sit it beside a high-power GPU card filled with VRAM, joined by an NVLink-class link (Figure 2). Point-to-point accelerator links in this class run at roughly 0.9–1.8 TB/s aggregate [17], an open alternative exists in UALink [18], and humbler connectors — OCuLink, SFF-8639, Thunderbolt 5 — already standardize the slower end. The division of labor is clean: VRAM keeps what must be written (KV cache, activations, the small diff file); the ROM card feeds what is frozen, at link speed. No interposer, board-level integration, and when the model revises, the card slides out and a new one slides in — the whole card is the cartridge.

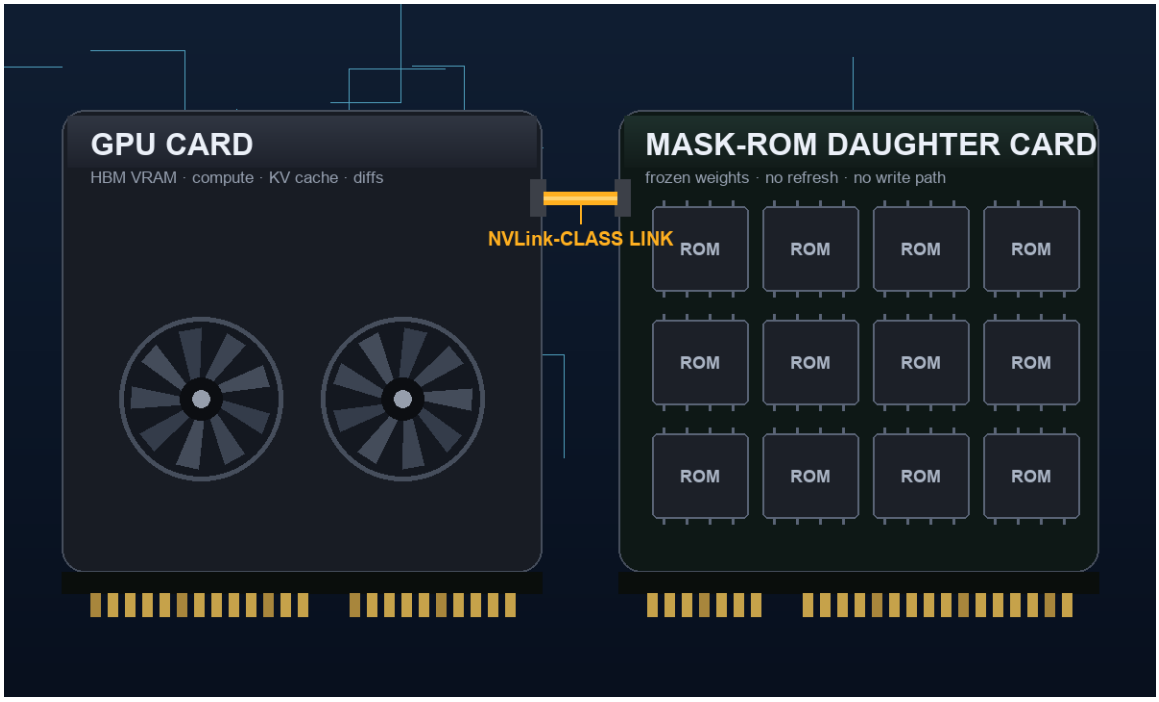


*Figure 2. The workhorse configuration: a mask-ROM daughter card beside a standard VRAM-filled GPU, joined by an NVLink-class link. The GPU keeps every writable byte (KV cache, activations, diffs); the ROM card feeds frozen weights at link speed.*

## 7. Compression Multiplies the Bus

There is a free lunch hiding in the interface math. Huffman coding (1952) [11] — and its modern relatives, arithmetic coding and ANS — can compress any known set of values when the relative frequency and data type of the values are known ahead of time. A burned model is exactly that case: the complete weight tensor and its histogram are known at mask time, so a near-optimal prefix code can be computed offline and the tiny decode table burned into the cartridge controller alongside the data. Quantized weight distributions are sharply bell-shaped, not uniform — a 4-bit weight spends its sixteen symbols very unevenly — so lossless entropy coding squeezes the stream well below its nominal width. Every bit of that squeeze is free bandwidth: the effective throughput of the cartridge link becomes bus rate × compression ratio, at zero cost to model quality, because the coding is lossless. Compression does not just shrink the cartridge — it speeds the bus up. Combined with a 4-bit ROM base and residual precision in the DRAM diff (Section 10), a modest connector starts to look a lot less modest.

## 8. Objections, Field-Tested

Earlier editions of this argument were posted publicly in August 2026 to collect objections [19]. The recurring criticisms are reproduced here in strengthened form, with responses. Several were answered in-thread by other readers before the author could reply — usually a sign that an objection is answerable from first principles.

- **"ROM chips are slow and tiny — the fastest is the AT27C1024 at 400 MB/s and 1 Mbit."** The strongest concrete objection received — and it cites a 1980s EPROM product line. It is the cleanest possible example of the historic-thinking error analyzed in §5.1–5.2. The claim is not that 1989's ROM is fast; it is that forty years of interface engineering, applied to a ROM array, yields RAM-class bandwidth (§5.3).
- **"Just swap a hard drive or SSD."** Confuses capacity with bandwidth. Inference touches every active weight once per token; SSD-class links deliver 5–14 GB/s, roughly two orders of magnitude below the requirement — and this tier of the argument was answered in-thread by other readers, who noted that Taalas's advantage is integration and bandwidth, not storage media.
- **"Frontier models are 500+ GB."** True for frontier-class dense models; the deployed open-weight mainstream is far smaller. A 27B-parameter model at 4-bit precision is under 18 GB. The form factors in §6 map to size classes; nothing in this architecture requires a trillion-parameter cartridge.
- **"ROM costs more per GB, and is physically bigger than RAM."** Backwards for mask ROM: no capacitor, no refresh periphery, smaller cell (§5.4). Today's high ROM prices reflect specialty volumes, not structure; the cost curve is a volume curve, and volume is the one thing the AI era supplies in abundance.
- **"Models rev every few months; a cartridge is obsolete before it ships."** The churn objection is real for etched ASICs [4] and is exactly why the proposed form factors are replaceable: the card slides out, and the DRAM diff bridges versions between editions (§2, §6.2).
- **"Different models' weights are unrelated; convergence talk is meaningless."** The strong version of the convergence claim is set aside. The observable version suffices: the industry already consolidates on a few open base models per generation (the Llama, Qwen, and DeepSeek families), with thousands of fine-tunes distributed as small deltas — the ROM+diff structure of §2, already in commercial use.
- **"Mixture-of-experts already solves this."** Agreed — and incorporated (§4). The most technically careful objection received in public independently arrived at the hot/cold expert split proposed here.
- **"If it were better, faster, cheaper, it would already exist."** The market for permanently frozen, bandwidth-hungry data is new. The physics has been on the shelf for forty years; the customer arrived in 2026 (§5.2).

## 9. Related Work: Others Have Had This Idea

The proposal sits squarely in a line of work that has moved from papers to products:

- **Taalas (existence proof, 2026).** The startup Taalas hardwires model weights directly into a model-specific ASIC. Its HC1 chip bakes a quantized Llama 3.1 8B into TSMC 6nm silicon — notably a mature, low-cost node, exactly the economics this paper argues for — and claims ~17,000 tokens/s; AMD acquired the company in August 2026 [4][5][6].

• **Apple, “LLM in a flash” (2023)** [2]: run models far larger than DRAM by keeping parameters in flash and streaming them on demand — the software analog of ROM+diff.

• **LoRA (2021)** [3]: frozen base + small delta is the industry’s standard update format, as argued in §2.

• **ROM compute-in-memory research.** Academic work already implements neural networks in ROM-based compute-in-memory macros (e.g., ROM-SRAM hybrid CiM for edge AI [7]; surveys in [8]) — the diode-ROM idea has a research pedigree, not just a product one.

## 10. Extensions

• **Put the multiply in the ROM.** The bottleneck in any ROM scheme is moving weights across a board. ROM compute-in-memory (multiply where the bits live) removes the interconnect problem entirely [7][8].

• **OTP/eFlash middle ground.** One-time-programmable or embedded-flash variants trade some density for field-updatability — a “slow ROM” for deployments that cannot wait for a new mask set.

• **Editioned releases.** Sell ROM sets like encyclopedia editions (“Llama 5, 2027 edition”) with a trade-in program; the DRAM diff carries owners between editions. The cartridge form factor (§3) makes the trade-in literal.

• **Quantize the base, refine with the diff.** Store a 4-bit base in ROM (as Taalas does [4]) and let the DRAM diff carry residual precision where it matters — then let entropy coding (§7) multiply the cartridge bandwidth on top.

## 11. Conclusion

Deployed large language models are read-only publications stored in rewriteable memory at rewriteable prices. The fix is an old pattern: permanent content on permanent media, with updates in a small live diff — encyclopedias, then game cartridges, now model weights. The speed objection dissolves under inspection: memory speed lives in the periphery and the packaging, and ROM — freed of capacitors, refresh, and write paths — matches DRAM per interface and beats it per die and per watt. Build the cartridge. Build the daughter card. Build the read-only HBM. The wall never existed.

## References


[1] M. Snyder, “Using Fixed Logic Gates to Mimic Fast Read Only Memory to Augment Frontier LLMs,” concept note (docx), Pic2Mag, 2026.

[2] K. Alizadeh et al. (Apple), “LLM in a flash: Efficient Large Language Model Inference with Limited Memory,” arXiv:2312.11514, 2023. https://arxiv.org/abs/2312.11514

[3] E. J. Hu et al., “LoRA: Low-Rank Adaptation of Large Language Models,” arXiv:2106.09685, 2021. https://arxiv.org/abs/2106.09685

[4] Wavect, “Taalas HC1 Review: Hardwired LLM ASIC,” 2026. https://wavect.io/blog/taalas-hc1-llm-asic-review/

[5] ComputeLeap, “AMD Buys Taalas: The Chip That Bakes Weights Into Silicon,” Aug. 2026. https://www.computeleap.com/blog/amd-buys-taalas-weights-in-silicon/

[6] ExplainX, “AMD Acquires Taalas: The Chip That Etches Model Weights Into Silicon,” Aug. 2026. https://explainx.ai/blog/amd-taalas-acquisition-etched-silicon-chip-august-2026

[7] “ROM-SRAM Hybrid Compute-in-Memory for Edge AI,” Research Square preprint. https://assets-eu.researchsquare.com/files/rs-7749677/v1/55cc6b41cf5df14523b8aa32.pdf

[8] L. M. de Souza et al., “Towards Efficient In-memory Computing Hardware for Quantized Neural Networks: State-of-the-Art, Open Challenges and Perspectives,” arXiv:2307.03936, 2023. https://arxiv.org/abs/2307.03936

[9] Intel Corp., “Thunderbolt 5 Technology Brief — 80 Gbps bi-directional, up to 120 Gbps,” 2023. https://www.intel.com/content/www/us/en/products/docs/io/thunderbolt/thunderbolt-5.html

[10] SNIA SFF Technology Affiliate, SFF-8639 (“U.2”) and SFF-8611 (OCuLink) connector specifications. https://www.snia.org/sff/specifications

[11] D. A. Huffman, “A Method for the Construction of Minimum-Redundancy Codes,” Proceedings of the IRE, vol. 40, no. 9, pp. 1098–1101, 1952.

[12] JEDEC, “High Bandwidth Memory (HBM3) DRAM,” JESD238, 2022. https://www.jedec.org/standards-documents/docs/jesd238

[13] JEDEC, “HBM4: 2048-bit interface High Bandwidth Memory,” JESD270-4, 2025. https://www.jedec.org

[14] SK hynix–TSMC collaboration on HBM4 base logic die (foundry-built base die), announced 2024.

[15] Intel Corp., 1103 DRAM (1970) and 1702 EPROM (1971) — historical product datasheets.

[16] F. Masuoka et al. (Toshiba), flash memory, IEDM, 1984.

[17] NVIDIA Corp., NVLink / NVLink-C2C interconnect specifications (Hopper generation ~900 GB/s; Blackwell generation ~1.8 TB/s aggregate), 2022–2024.

[18] UALink Consortium, “UALink 1.0 Specification,” 2025. https://www.ualinkconsortium.org

[19] Public discussion threads: r/pcmasterrace and r/woahdude, “How to Fix AI Tech Prices,” August 2026.