Fast ROM at RAM Speed: The Speed Wall That Never Was

A rebuttal to "ROM is too slow" — the case for ROM daughter cards and read-only HBM in the AI inference era.

Michael Snyder (Pic2Mag), August 2026

Abstract. Critics of the frozen-weights-in-ROM proposal make one confident claim: "ROM is too slow to feed a GPU." The claim confuses a product category with a physical limit. The speed of every modern memory — HBM included — is set by its interface periphery and its packaging, not by its storage element. ROM never hit a speed wall: the industry simply stopped developing high-speed read-only memory in the 1980s, because DRAM could read *and* write, and one part was easier to buy than two. Nothing physical happened to ROM; it was left behind by a business decision. This paper shows that a ROM built with today's interface and packaging technology runs at RAM speed by construction — and is denser and lower-power than DRAM per die precisely because of what you delete: capacitors, refresh, and the write path. It then proposes two buildable products: a read-only HBM stack, and a mask-ROM daughter card with an NVLink-class link sitting beside a standard VRAM-filled GPU.

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.

2. The Wall That Never Was: 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 [6]. 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.

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 [2], doubled to 2048 bits on HBM4 [3], 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 [4] — 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.

4. Better Than Equal: 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.

5. Two Buildable Products

5.1 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.

5.2 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 1). Point-to-point accelerator links in this class run at roughly 0.9–1.8 TB/s aggregate [7], an open alternative exists in UALink [8], 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.

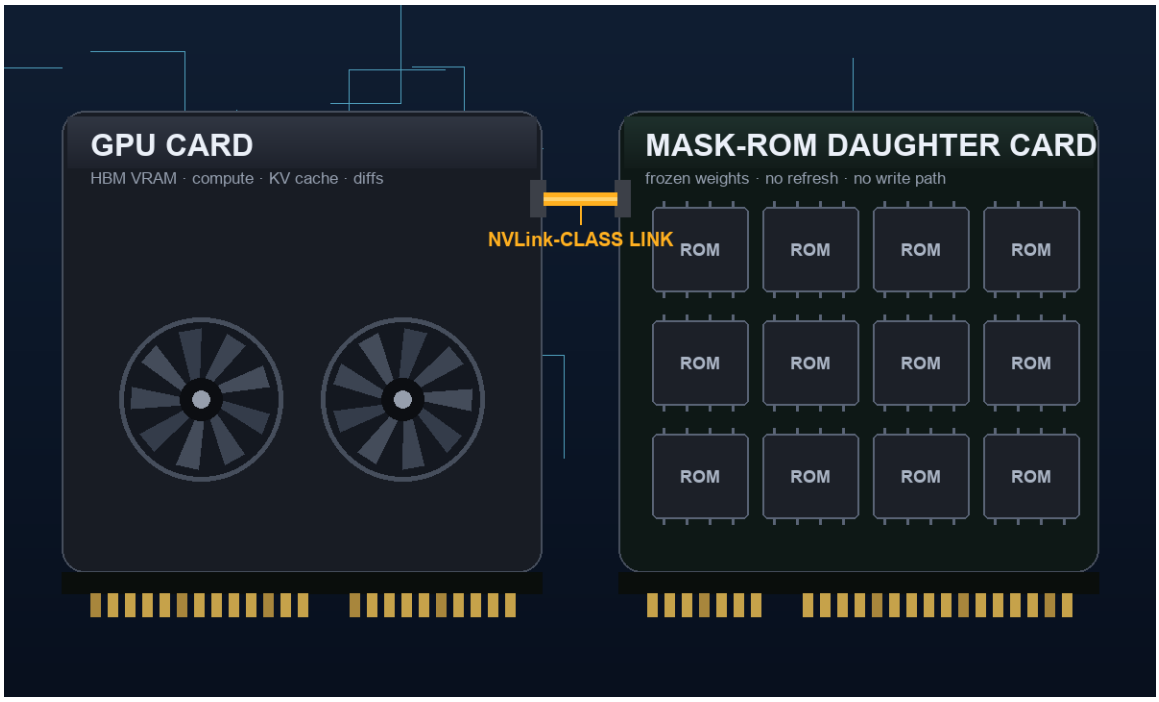

Figure 1. The workhorse configuration: a mask-ROM daughter card beside a standard VRAM-filled GPU, joined by an NVLink-class link. The GPU keeps every writable byte (KV cache, activations, diffs); the ROM card feeds frozen weights at link speed.

One multiplier deserves mention: the weight histogram of a burned model is known in advance, so an entropy code (Huffman or ANS) with its decode table on the card squeezes the stream losslessly — effective bandwidth becomes link rate × compression ratio, at zero cost to model quality [9].

6. The Honest Objections

“In-package still wins.” Correct — for latency and peak bandwidth. That is what §5.1 is for. The daughter card trades a slice of bandwidth for replaceability and freedom from advanced-packaging queues; the two tiers coexist.

“Models churn; frozen silicon strands.” A daughter card is not an etched ASIC. Swapping a card is a minutes-long maintenance event, not a fab cycle — the churn objection applies to Taalas-style hardwiring [10], not to a replaceable card.

“ROM costs more per GB than DRAM.” Today it does — because today’s ROM volumes are trivial and the parts are specialty items. Mask ROM on mature, fully depreciated fabs is structurally cheap: the wafer is plain CMOS, the cell needs no capacitor module, and the data pattern lives in the last few masks, so base wafers can be warehoused and committed to a model with a handful of masks. The cost curve is a volume curve — and volume is the one thing the AI era supplies in abundance.

“If it’s so easy, why hasn’t anyone done it?” Because until large language models, there was no mass market for hundreds of gigabytes of permanent data that must move at RAM speed. The physics has been sitting on the shelf for forty years. The market arrived in 2026.

7. Conclusion

ROM did not hit a speed wall in the 1980s; the industry turned a corner, and ROM stayed on the old road. That road turns out to lead directly to the AI era’s central memory problem: enormous, permanent, read-only data served at full bandwidth. The historic thinking that says “ROM is slow” is arguing with a product line that stopped developing in 1989, not with anything physics built. We have a new problem that is really an old one — and the old tools, reapplied, solve it. Build the daughter card. Build the read-only HBM. The wall never existed.

References

[1] M. Snyder, “Frozen Weights, Fast Silicon: Read-Only Memory as the Substrate for Frontier LLMs,” companion whitepaper, Pic2Mag, 2026.

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

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

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

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

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

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

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

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

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