

Public Evidence for Edge Compute on NVIDIA Jetson Thor

13 September 2026 | Technical note | Version 2.0
QDOX Research | QDOX-TR-01

Public-source review and recorded compiler evidence. No new GPU experiment.
Independent publication. Not peer reviewed by IEEE.

Abstract

Selecting a computer for an intelligent machine requires evidence at several levels: documented hardware, accepted instructions, correct executable kernels, and application behaviour. This note examines the first two levels through a publicly accessible collection of compiler probes and device-property records for NVIDIA Jetson Thor. It compares the recorded observations with official platform specifications and archived compiler documentation. A mechanical review of the published matrix identifies thirteen probe forms tested against five targets, including one malformed form that cannot support a hardware conclusion. The review also distinguishes architecture-specific from family-specific instruction targets and identifies a disagreement between a repository cache description and the manufacturer's published specification. Equations connect reported device attributes to theoretical bandwidth and thread residency, while a separate model explains why arithmetic capability does not determine application latency. None of these calculations is a throughput measurement. The result is an inspectable account of what the public artifacts establish, where their evidence stops, and how future correctness, memory-traffic, and timing experiments could extend it.

Index Terms: Compiler validation, edge computing, graphics processors, reproducibility, robotics.

I. Introduction

Thor Hacks is a public repository of compiler probes, reference outputs and technical notes [1]. NVIDIA describes Jetson Thor as a platform for robotics and physical artificial intelligence; its T5000 specification lists 128 GB of memory and a 256-bit memory interface rated at 273 GB/s [2]. Those specifications motivate evaluation. They do not establish the responsiveness of a particular model or the safety of a robot.

A central distinction is between an instruction's availability and a useful implementation of that instruction. NVIDIA's Parallel Thread Execution documentation describes a virtual instruction set and its translation to target instructions [3]. An assembler can accept a program that has never been executed with valid inputs. A kernel can execute correctly without improving a complete application. This note keeps those evidence levels separate.

The question is therefore narrow: **What can the published Thor probes support, and which further claims require different experiments?** The contribution is a public-source analysis with explicit limits, not a new robot controller, inference engine or performance result.

II. Public Sources and Review Method

The repository was accessible without authentication on 13 September 2026. Its inspected public revision is 21c08d78d5a45da6b61eb0193a3abbff24dfc0be. The compiler-acceptance harness generates a small program for each probe and tests five targets: `sm_110`, `sm_110a`, `sm_90a`, `sm_100a` and `sm_120a` [4]. Its saved output identifies CUDA 13.2, compiler version 13.2.78 [5].

Two additional artifacts provide independent checks. The migration script generates compilation tests for language versions, architecture names and selected structure fields [6], with a separate saved output [7]. The device query calls CUDA runtime property functions [8], with the returned properties published as text [9]. Fig. 1 shows their dataflow and the boundary between observations and later application validation.

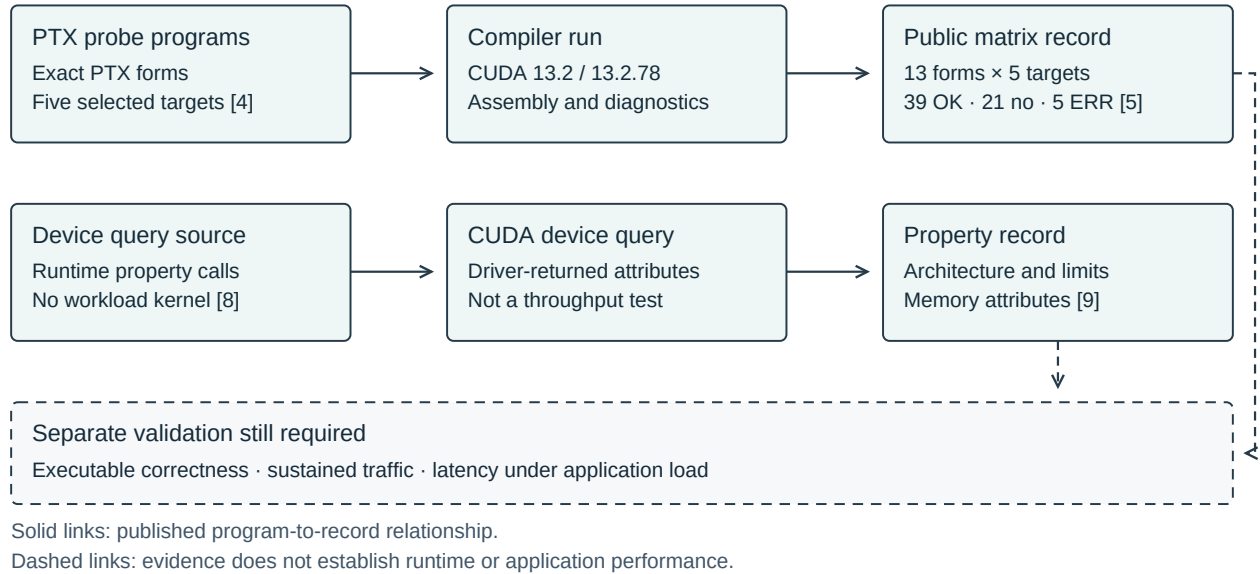


Fig. 1. Public evidence dataflow. Solid arrows connect the supplied programs to their published output records. Dashed arrows indicate reasoning or further validation; they do not represent an implemented runtime system.

This review downloaded the cited public files anonymously, compared probe names with matrix rows, counted outcomes, checked the two shell scripts' syntax, and cross-checked statements against official documentation. It did not execute a GPU kernel or reproduce the CUDA 13.2 compiler run. The matrix is a reanalysis of a published record, not a new measurement. No private source is used to substantiate a claim.

III. Recorded Compiler and Device Evidence

A. Acceptance Matrix

The source contains thirteen probe invocations, and the output contains thirteen rows with five classified outcomes each. The 65 cells comprise 39 accepted compilations, 21 explicit unsupported-target outcomes and five errors. These counts describe the supplied matrix; they are not estimates of the fraction of Thor instructions supported [4], [5].

TABLE I
Recorded compiler outcomes; assembly is not runtime validation

Probe family	Forms	sm_110	sm_110a	Evidence boundary
Dense signed 8-bit matrix control	1	Accepted	Accepted	Assembler control
Sparse warp-level matrix operations	4	Accepted	Accepted	Exact tested forms
Fifth-generation tensor allocation, load and matrix operations	7	Unsupported	Accepted	Exact tested forms and targets
Warp-level block-scaled operation	1	Error	Error	Malformed arguments; no capability result

The seven fifth-generation forms include block-scaled dense and sparse matrix instructions. The public record demonstrates their acceptance under the tested architecture-specific target. It does not demonstrate valid operand descriptors, correct tensor-memory use, synchronization, numerical accuracy or an inference benefit.

The error row is especially informative. All five targets reject the supplied warp-level block-scaled form with an argument mismatch. A failure on the control target prevents interpreting this row as evidence that a device lacks the capability. Keeping it visible is more useful than converting it into an unsupported-target result [5].

B. Target Coverage Is Incomplete by Design

The five-target matrix omits sm_110f. Archived NVIDIA documentation distinguishes ordinary, architecture-specific and family-specific targets, and lists sm_110f as a valid target [10]. Its instruction documentation permits family-specific targets for some fifth-generation operations, with restrictions that depend on the exact form [3]. Consequently, the matrix supports the narrow comparison between sm_110 and sm_110a; it cannot establish that every fifth-generation operation universally requires an a target. A future matrix should add the omitted target and retain instruction-specific controls.

C. Migration Findings

The saved migration output rejects the tested older intermediate-language versions for sm_110 while accepting version 9.0. It rejects the sm_101 architecture name but accepts compilation for sm_87 as well as the Thor targets. The latter result illustrates why build success cannot confirm that the intended platform was selected. The script does not execute that binary on Thor [6], [7].

Two selected clock fields are reported as absent from the device-property structure. The query program uses attribute calls for those values instead. The field-test harness has a limitation: it labels any compilation failure as a missing field, while its final output omits the full diagnostic. An environment error could therefore be misclassified. Diagnostic retention is needed before turning this tool into an automated compatibility gate [6], [8].

D. Recorded Properties and a Source Disagreement

The device-property record reports compute capability 11.0, 20 streaming multiprocessors, 1,536 maximum resident threads per multiprocessor, a 32 MiB GPU L2 cache and 227 KiB of opt-in shared memory per block [9]. These are driver-query observations in the published record, not current measurements of free capacity, sustained operating frequency or workload isolation.

A separate repository note infers that no shared L3 exists from the absence of a particular operating-system cache entry [11]. NVIDIA's public T5000 specification instead lists a **16 MB shared system L3 cache** [2]. The two claims conflict. This note follows the manufacturer specification for the documented hardware description and leaves the software-visibility discrepancy unresolved. Absence from one software interface is not sufficient evidence of physical absence. The GPU L2 observation and the system L3 specification refer to different cache descriptions and should not be conflated.

IV. Quantitative Interpretation

A. Theoretical and Effective Bandwidth

NVIDIA distinguishes a bandwidth calculation from a timed transfer measurement [12]. Let f_m be the memory-clock attribute in hertz and w the interface width in bits. Applying the double-data-rate calculation used by the public query gives

$$B_{\text{theory}} = 2f_m \frac{w}{8}. \quad (1)$$

Using the recorded values, $f_m = 4.266 \times 10^9$ Hz and $w = 256$, yields

$$B_{\text{theory}} = 273.024 \times 10^9 \text{ bytes/s} \approx 273.0 \text{ GB/s}. \quad (2)$$

This arithmetic is consistent with the manufacturer's rounded bandwidth specification [2], [9]. It supplies no measured sustained rate. For a timed kernel, the effective bandwidth convention is

$$B_{\text{effective}} = \frac{D_r + D_w}{t}, \quad (3)$$

where D_r and D_w are bytes read and written, and t is the measured elapsed time [12]. The byte count must identify what is being measured: requested kernel traffic and actual off-chip transactions can differ because of caches and access patterns. Use one unit convention consistently; this note uses decimal GB/s and binary MiB/KiB where explicitly stated.

B. A Thread-Residency Bound

Let T_{max} be the reported resident-thread limit per multiprocessor and T_b the threads in a block. A bound from thread count alone is

$$n_{\text{blocks}} \leq \left\lfloor \frac{T_{\text{max}}}{T_b} \right\rfloor. \quad (4)$$

For $T_{\text{max}} = 1536$ and $T_b = 1024$, no more than one such block fits by thread count. Its thread occupancy would be $1024/1536 = 2/3$, subject to other resource constraints [9]. Registers and shared memory may further constrain residency [3]. This calculation does not reserve multiprocessors for another process or guarantee real-time scheduling.

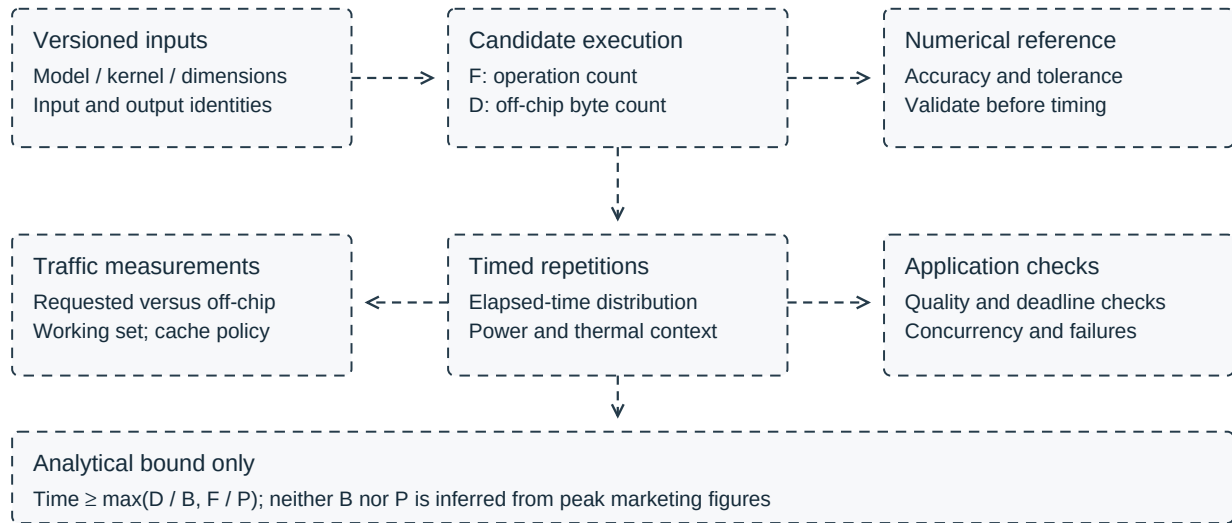
C. Application Latency Needs More Than Arithmetic Peak

Consider an idealized work unit, such as one generated token. Let D be its off-chip byte traffic and F its arithmetic work. With applicable sustained memory and compute ceilings B and P , an optimistic bound is

$$t_{\text{unit}} \geq \max\left(\frac{D}{B}, \frac{F}{P}\right), \quad r_{\text{unit}} \leq \min\left(\frac{B}{D}, \frac{P}{F}\right). \quad (5)$$

The bound assumes ideal overlap and omits additional delays. It is an analytical model, not a fitted Thor performance claim. Arithmetic intensity $I = F/D$ identifies the balance point at $I = P/B$:

increasing compute capability alone cannot improve the memory-bound side of this model. Fig. 2 maps the quantities to a proposed measurement path.



All boxes are a proposed test topology; no deployment or measured performance is represented.

Fig. 2. Proposed measurement topology for (3)-(5). The diagram defines experiments still required: public input identity, numerical reference, measured traffic, timing and application checks. It contains no measured throughput or latency values.

The repository's bandwidth note includes numerical performance claims, but the inspected public collection does not supply the corresponding benchmark harnesses and raw logs needed to reconstruct those rates [13]. This note therefore does not use those figures to calibrate (5), advertise a speedup or predict a robot's control frequency.

V. Limitations and Reproduction Plan

The evidence is limited to the exact public revision, programs and recorded toolchain. Compiler acceptance does not establish runtime correctness. A device query is not a stress test. The public result records do not provide repeated trials, complete power and thermal conditions or an application benchmark. The probe scripts are diagnostic programs, not comprehensive pass/fail qualification suites.

A reproducible extension should proceed in stages:

1. Check out the cited public revision, record compiler and driver versions, and rerun the compiler matrix with its known controls. Add family-specific targets without silently changing the old result.
2. Correct the malformed form and preserve both the original error and corrected output. Retain complete diagnostics and distinguish tool failures from unsupported instructions.
3. Add executable kernels with public inputs, valid descriptors and explicit synchronization. Compare outputs against a numerical reference using declared absolute and relative tolerances.
4. Publish timed memory experiments with working-set size, warm-up policy, cache assumptions, repetitions and operating conditions. Distinguish requested bytes from off-chip traffic.

5. Evaluate complete applications with documented models, input lengths, quality checks, memory use and latency distributions. Test concurrent workloads before making any timing-isolation claim.

These are proposed experiments, not completed results. Robotics, aerial systems and personal memory applications impose additional requirements that compiler probes cannot resolve: physical interaction, scheduling deadlines, communication failure, consent and access control. No implemented capability in those areas is claimed here.

VI. Conclusion

The public Thor artifacts provide an inspectable foundation for reasoning about compiler compatibility and device properties. Their most useful result is precise scope: accepted forms, rejected targets and malformed tests are different kinds of evidence. Comparison with official documentation prevents overgeneralizing the architecture-target result and exposes an unresolved cache-description discrepancy. The next scientific contribution should be reproducible runtime correctness and application measurements, with enough public detail for another researcher to challenge the result.

Preparation Note

OpenAI Codex assisted with public-source review, drafting, equations and original vector diagrams. This document records no newly executed GPU experiment. Its structure borrows conventions from IEEE author guidance; no IEEE submission, acceptance, endorsement or peer review is represented.

References

- [1] M. Disodia, “[Thor Hacks](#),” public source repository, revision 21c08d78d5a45da6b61eb0193a3abbff24dfc0be, Aug. 17, 2026. Accessed: Sep. 13, 2026.
- [2] NVIDIA, “[Jetson Thor: series modules and developer kit specifications](#).” Accessed: Sep. 13, 2026.
- [3] NVIDIA, “[Parallel Thread Execution ISA 9.2](#),” CUDA Toolkit 13.2 documentation, secs. 3.1, 9.7.16 and 13. Accessed: Sep. 13, 2026.
- [4] M. Disodia, “[Sparse and fifth-generation tensor-instruction probe](#),” Thor Hacks, cited revision, lines 13-142.
- [5] M. Disodia, “[Published compiler-acceptance output](#),” Thor Hacks, Aug. 17, 2026, lines 1-37.
- [6] M. Disodia, “[CUDA migration probe](#),” Thor Hacks, cited revision, lines 19-81.
- [7] M. Disodia, “[Published migration output](#),” Thor Hacks, Aug. 17, 2026, lines 1-34.
- [8] M. Disodia, “[Device-property query](#),” Thor Hacks, cited revision, lines 10-60.
- [9] M. Disodia, “[Published device-property output](#),” Thor Hacks, Aug. 17, 2026, lines 1-31.
- [10] NVIDIA, “[PTX Compiler API 13.2: compilation options](#).” Accessed: Sep. 13, 2026.
- [11] M. Disodia, “[CPU ISA note](#),” Thor Hacks, cited revision, lines 21-34.
- [12] NVIDIA, “[CUDA C++ Best Practices Guide 13.2](#),” sec. 9.2. Accessed: Sep. 13, 2026.
- [13] M. Disodia, “[Bandwidth and roofline note](#),” Thor Hacks, cited revision, lines 13-49.