SIRIUS QUANTUM

A Quantum Lab on One Workstation

Exact quantum statevector simulation on AMD Radeon 8060S and AMD Instinct MI350P with Zilver

A statevector simulator tracks a quantum computer's full state: every amplitude, updated after every gate. That makes it the reference that approximate methods and learned models are checked against. The price is memory: each added qubit doubles it. A 32-qubit state is 4.3 billion complex amplitudes, more numbers than a 1-billion-parameter model has weights, updated in full after every gate.

Summary. Zilver, our open-source statevector simulator, installs with `pip` and runs **32 qubits** with the same code on two AMD GPUs: the integrated Radeon 8060S of a workstation and one partition of an **Instinct MI350P**. The 32-qubit benchmark circuit takes 6.06 s on the Radeon and **1.84 s** on MI350P; a 541-gate, 32-qubit adder takes **2.03 s** on MI350P. The quantum Fourier transform matches its closed form to 2.7e-6 relative error on both machines. A one-qubit gate costs 0.98 memory passes on the Radeon and 1.42–1.51 on MI350P, where the kernel is not yet tuned.

1 The route in

ROCm 10.0.0 has a Windows path for gfx1151 that is pure pip. The wheels at `stable.repo.amd.com/rocm/whl-next/` carry the whole user-mode stack: HIP runtime, compiler, math libraries and the precompiled gfx1151 kernel pack. There is no HIP SDK, no MSI and no service, and everything lands in a per-user virtual environment.

The single administrator step in the entire path is the kernel-mode driver, which was already current on this machine. Nothing else required elevation.

We first tried WSL2 and found no ROCm device exposed there at all: no `/dev/kfd`, no `/dev/dri`. `torch-2.5.1+rocm6.2` installs cleanly and reports no GPU. Native Windows was the way in.

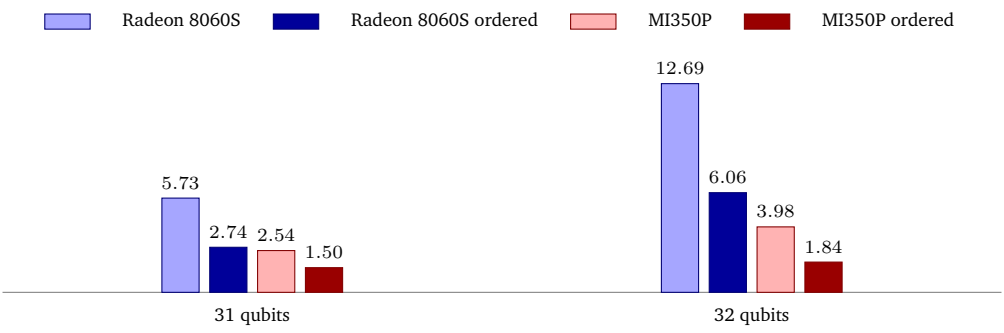
2 What it runs

A Hadamard and a Y-rotation on every qubit, then a CNOT chain (the entangling step): 95 gates at 32 qubits. Each time is end to end, including copying the full state back to the host. The norm is reported to seven places.

qubits	state	seconds	norm
31	17.18 GB	5.73	0.9999993
32	**34.36 GB**	**12.69**	**0.9999993**

Time roughly doubles with the state, which is the behaviour of a simulator limited by memory bandwidth rather than arithmetic.

Our first working run on this box reached 31 qubits in 134 s and could not allocate 32 at all. After the fused kernel replaced the framework gate path, 31 qubits took 21.99 s and 32 completed. Three later changes brought 31 qubits to 5.73 s: fusing the 95 gates into 19 passes (Section 3), staging the readback through pinned memory, and building the initial state on the device instead of copying it across.



seconds per run, end to end

Against that first 134 s run, this is **23 times faster** for the same circuit. The framework path is not in the chart

because it never ran at 32 qubits: it allocates temporaries on top of the state, and the 34.36 GB state leaves no room for them. Updating in place is what made the 32nd qubit reachable.

The ordered bars use the qubit ordering described below; the MI350P bars are from Section 6.

3 Fusing and ordering the circuit

A gate costs one pass over memory whatever it does, so the quantity to minimise is passes, not gates. Two changes cut them.

The first fuses gates by the **union of the qubits they touch** rather than by counting gates. A pass spanning four qubits can carry four one-qubit gates, two CNOTs, or three chained CNOTs of an entangling ladder. The gates need not commute, only fit. That takes the 95-gate circuit from 95 passes to 19.

The second uses a property of the starting state. A run begins at $|0 \ldots 0\rangle$, and a qubit no gate has touched yet is still exactly $|0\rangle$, unentangled from the rest. The state therefore factorises: it is a dense block over the touched qubits and zero everywhere else. Ordering the gates so that block grows one qubit at a time, and choosing which bit each qubit occupies so the live amplitudes stay contiguous, lets the early passes read and write a fraction of the state instead of all of it. The 19 passes then touch as much memory as 1.57 full ones.

That brings 31 qubits to 2.74 s and 32 qubits to 6.06 s, both plotted above, with the norm unchanged at 0.9999993.

At 32 qubits the 95 gates themselves take **0.59 s**. The remaining 5.5 s is allocating the 34.36 GB state and copying it back to the host. The simulation is no longer the expensive part of the run.

Two things to note with it. This numbers the qubits to suit the ordering, which changes no physics: the same circuit with the qubits named in the other direction is the 12.69 s above. And the saving is largest on short circuits: most of the gate time at 95 gates, a small fraction of it for a circuit thousands of gates deep.

4 The kernel

A statevector gate is a memory problem, not a compute problem. It must touch every amplitude once; the only question is how many extra times the software touches it.

A single-qubit gate is a permutation plus a 2x2 multiply across the whole state. By 24 qubits the state (134 MB) is far larger than any cache on this part, so the gate has to read it and write it back once. That is the floor, and we measure it on the state itself rather than on a proxy: an in-place stream over the full 34.36 GB buffer at 32 qubits runs at 214.5 GB/s, and that is 1.00 passes by definition. We use one buffer, not two: a copy between two buffers needs twice the memory and degrades with width on this part, from 207 GB/s at 24 qubits to 119 GB/s at 31, so it would flatter any kernel measured against it.

Against that floor the fused kernel costs **one pass**: measured at **0.98** at 32 qubits, which is the floor to within measurement noise. The framework tensor path, measured the same way at 24 qubits, costs 4.4 to 7.2 passes per gate, rising as the stride tightens. The difference is temporaries, and removing them is the whole optimisation.

The kernel is written for gfx1151 and compiled at runtime through hipRTC (AMD's runtime compiler) in 0.3 s. There is no build toolchain in the loop: the source is a string in the Python process. It is verified against a planted Walsh-Hadamard state where every amplitude is known in closed form, to a maximum absolute error of 1.9e-9 at 16 and 20 qubits.

At the full 32 qubits we check it against the quantum Fourier transform of a basis state, whose amplitudes are also known in closed form: 18 runs from 20 to 32 qubits, with 65,537 amplitudes read back at 32, and a largest error of 2.7e-6 relative to the exact amplitude size. Moving one gate to the wrong qubit makes the same check fail (error 1.41), and a repeated run is bit-identical.

5 The memory ceiling

A 33-qubit state needs **64 GiB (68.7 GB)**. That is the whole of the GPU-addressable window as this machine is configured, with 64 GB of its 128 GB assigned to graphics. We reserved a block that size and it failed on the first write to it, so nothing is left for the runtime and 32 qubits is the ceiling in this configuration. AMD's Variable Graphics Memory allows up to 96 GB on Windows, enough for a 33-qubit state. The ceiling is set by memory capacity, the natural limit for a statevector, and not by compute or software.

Worth stating the other way round. Each qubit doubles the state, and the 32-qubit state is 34.36 GB, a little over half the 64 GiB window. Reaching 32 qubits was never a question of having a large machine, it was a question of needing only one copy of the state: a path that peaks at two to three times the state wants 69 to 103 GB for the same width and does not fit in the window at all.

6 The same code on AMD Instinct MI350P

One compute partition (gfx950, 38.65 GB), Linux, PyTorch 2.13.0 for ROCm 7.1. The kernel compiled at run time in 0.3 s; no source line changed.

Correctness. The QFT check passes at 20, 24, 28, 30, 31 and 32 qubits, three seeds each: largest error 2.7e-6 relative to the exact amplitude size. Moving one controlled phase fails it (1.41). A repeat returns 65,537 sampled amplitudes bit-identical.

Speed. The benchmark circuit, end to end, `python -m zilver.gpu`; norm 0.9999993 in every run. The times are in the chart in Section 2, beside the Radeon 8060S: 3.3 times faster at 32 qubits ordered (1.84 s against 6.06 s), 3.2 times not ordered (3.98 s against 12.69 s). Allocating the 34.36 GB state takes 0.04 s.

Memory traffic. Against an in-place stream over the state (1,318 GB/s at 32 qubits), a one-qubit gate costs 1.42–1.51 passes and a controlled phase 2.14–2.18; the Radeon measured 0.98. The kernel is tuned for gfx1151, not gfx950.

Adder. Cuccaro ripple-carry adder on a uniform superposition of inputs; each Toffoli is 6 CNOT, 7 T and 2 H. Exact output: amplitude 2^{-b} on every $|a,\ a+b,\ \mathrm{carry}\rangle$, zero elsewhere; 2,000 amplitudes checked. Same code, fusion on (default, up to four qubits per pass) and off.

bits	qubits	gates	fused	one gate per pass	largest error
14	30	505	34 passes, 0.97 s	477 passes, 3.71 s	4.2e-6
15	32	541	37 passes, 2.03 s	511 passes, 15.12 s	4.4e-6

A fused pass at 32 qubits takes 55 ms; an in-place stream over the state takes 52 ms. Fused times include first-use kernel compilation. Removing one CNOT gives relative error 1.0.

Ceiling. 32 qubits (34.36 GB) is the largest state on one 38.65 GB partition. A node has two 144 GiB cards, each split into four such partitions, 309 GB in all: enough for 35 qubits (275 GB) with the state split across devices, which Zilver does not do yet. One card run unpartitioned would hold 34 qubits (137 GB); not run.

7 What this opens

Every quantum algorithm, error-mitigation scheme and learned quantum model is judged against the exact state, and at 32 qubits that state has needed a large discrete GPU. The machines now being bought for AI already hold it: a local AI box, the 128 GB Ryzen AI Max+ workstation built to run large language models at a desk, and the Instinct MI350P in an AI data centre. Zilver runs exact 32-qubit simulation on both with the same open-source code and two commands, matching the closed-form quantum Fourier transform to 2.7e-6. On the MI350P a run takes 1.84 s, about 2,000 exact 32-qubit states an hour from one partition, so the exact answer stops being an occasional reference and becomes a routine test and a training signal. Quantum simulation no longer needs its own hardware: it runs on the AI hardware researchers, developers and model builders already have, from the local box to the centre.

8 Configuration

Machines. AMD Ryzen AI Max+ 395 (16 cores, 32 threads) with Radeon 8060S integrated graphics and 128 GB of unified memory; Windows 11. AMD Instinct MI350P, one compute partition (gfx950, 38.65 GB); Linux.

Software. Zilver (open source, Apache 2.0) on PyTorch 2.13.0 for ROCm, from AMD's ROCm 10.0.0 preview wheels on Windows and the ROCm 7.1 build on Linux, and Python 3.12. The fused kernel is `src/zilver/hip/fused_gate.hip`; `python -m zilver.gpu` runs every gate through it.

Run it, with PyTorch for ROCm installed:

```
pip install zilver
python -m zilver.gpu                    # ordered
ZILVER_SHRINK=0 python -m zilver.gpu    # not ordered
```

© 2026 Sirius Quantum Solutions Ltd.