GoldCoreX Product Lineup

GDX-M1 — Flagship 1M Qubit Hybrid

- Use: Retrofittable CPU/GPU quantum accelerator
- Integration: PCIe or SoC for quantum-augmented compute clusters
- Target Market: Data centers, chip vendors, AI labs

GDX-QE1 — Quantum Entanglement Engine

- Use: Multi-bus, refresh-stable entanglement module
- Stackable: Yes
- Target Market: Quantum cloud APIs, encryption clusters, multi-user QML systems

GDX-Micro — Mobile-Grade QPU

- Use: Quantum-enhanced smartphones, wearables, and AI edge devices
- **Qubits:** 500k
- Power Profile: Ultra-low-power photonic switching
- Target Market: Samsung, Apple, Lenovo, defense mobile systems

GDX-Matrix 10 — Supercompute Array Core

- Use: 10 × QE1 matrix stack (10M qubits)
- Application: Quantum HPC, matrix-based entangled compute
- Target Market: National labs, defense agencies, quantum simulation platforms

GDX-Link I/O — Classical-Quantum Interface Layer

- Use: Classical-to-QASM logic translation
- Compatibility: Interfaces with M1, QE1, and Matrix 10
- **Features:** Integrated photon routing logic
- Target Market: Developers, hardware integrators

GDX-CoreDev Kit — Developer SDK Bundle

- Use: 100k qubit chip + SDK tools for R&D
- Software: Includes photon controller emulator and dev suite
- Target Market: University labs, quantum startups, SDK teams

GDX-Secure — Quantum Encryption Module

- Use: Embedded quantum key exchange
- Qubits: $\sim 200 \mathrm{k}$
- Application: Quantum-resistant encryption systems
- Target Market: Military, financial sector, satellite communications

GDX-SatOps — Space-Grade Quantum Controller

- Use: Radiation-hardened QPU for orbital/hostile environments
- **Qubits:** 250k
- Application: Space-based quantum networking, secure comms
- Target Market: NASA, SpaceX, defense satellites

GDX-Alpha-X — Experimental "Open Core" Platform

- Use: Customizable platform for QML and experimental work
- **Qubits:** 750k
- Features: Tunable well depth, pulse timing, and refresh cycle
- Target Market: DeepMind, OpenAI, NVIDIA, academic labs