Q-OS: A Universal FPGA Operating System for Quantum-Mimetic Execution

Noam Cohen and Deywe Okabe

Independent Research: Quantum-Mimetic Hardware Systems

February 16, 2026

Abstract

The rapid development of quantum algorithms is currently bottlenecked by the scarcity, cost, and extreme physical requirements of cryogenic quantum hardware. **Q-OS** is a universal operating system and bitstream architecture designed to bridge this gap. By deploying Q-OS onto any standard Field Programmable Gate Array (FPGA), users can instantly convert classical silicon into a Quantum-Mimetic Transducer. Featuring an embedded Web User Interface (UI) and native support for industry-standard frameworks like Google Cirq, Q-OS allows researchers to write, compile, and execute quantum circuits on emulated continuous-variable qubits at room temperature.

1 The Core Philosophy: Democratizing Quantum Hardware

Q-OS operates on the principle of hardware abstraction. To the end-user, the system appears as a standard quantum processing unit (QPU). The user writes a quantum circuit using standard Python libraries (e.g., Cirq or Qiskit) via the Q-OS Web UI. Under the hood, the operating system bypasses the need for physical qubits by translating these quantum gates into continuous-wave analog modulation instructions, executing them on the FPGA's deterministic mimetic fabric.

2 Universal System Architecture

To ensure compatibility across a wide range of hardware (from Alinx development boards to enterprise-grade datacenter FPGAs), Q-OS is structured as a System-on-Chip (SoC) stack, divided into three fundamental layers:

2.1 Layer 1: The Application and Framework Layer (The Web UI)

Hosted on the embedded processing cores of the FPGA, this layer runs a lightweight web server.

- **Web UI:** A browser-based interface allowing users to upload Python scripts, visually construct quantum circuits, and view execution results.
- **Quantum Execution Engine:** An embedded Python environment that ingests circuits from Google Cirq. It acts as a compiler, interpreting quantum gates (like Hadamard or CNOT) and mapping them to their corresponding SPHY-wave modulation parameters.

2.2 Layer 2: The Hardware Abstraction Layer (HAL)

This is the communication bridge between the software and the silicon. The parsed quantum instructions are sent from the processor to the FPGA fabric via high-speed AXI buses. This layer ensures that regardless of the specific FPGA brand, the memory-mapped instructions are delivered uniformly.

2.3 Layer 3: The Mimetic Bitstream (Programmable Logic)

This is the core proprietary hardware configuration uploaded to the FPGA fabric.

- **State Preparation:** Translates the instructions from the HAL into exact phase and frequency parameters for the HPHYSPI and SPYSPI waves.
- **The Mimetic Loop:** Routes the interference patterns through the ADC/DAC loop, projecting the probability amplitudes as a continuous analog voltage gradient, effectively running the quantum algorithm on a physically deterministic substrate.

3 The Quantum Execution Engine: Compiling Cirq to SPHY Waves

The most critical component of Q-OS is the Quantum Execution Engine. Hosted on the FPGA's processing system (PS), this engine acts as a Just-In-Time (JIT) compiler. It translates quantum matrix operations directly into analog modulation parameters for the FPGA's Programmable Logic (PL).

3.1 The Mathematical Challenge: The Hadamard Gate

In a standard quantum framework, applying a Hadamard gate to a qubit in the $|0\rangle$ state creates an equal superposition. Mathematically, the Hadamard operator is defined as:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{1}$$

When applied to the basis state $|0\rangle$, the resulting state vector is:

$$H|0\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \tag{2}$$

Q-OS bypasses the cryogenic requirements of physical rotation by mapping this probability matrix to a deterministic analog wave state.

3.2 Translation to the Mimetic Domain

When the Q-OS Execution Engine receives the command `cirq.H(qubit_0)` from the Web UI, it maps the desired probability amplitudes ($\frac{1}{\sqrt{2}}$) to the interference pattern of the SPHY waves. The quantum gradient equation of our Mimetic Bridge relies on the interference intensity I between the primary harmonic stabilizer ($\mathcal{H}$) and the secondary phase-alignment wave ($\mathcal{S}$):

$$I(\mathcal{H}, \mathcal{S}) = \mathcal{H}^2 + \mathcal{S}^2 + 2\mathcal{H}\mathcal{S}\cos(\Delta\phi) \tag{3}$$

To emulate the Hadamard superposition, Q-OS dynamically recalculates the phase difference ($\Delta\phi$) required to perfectly mirror this probability distribution.

3.3 AXI-Bus Parameter Injection

The computed phase adjustments are injected via the AXI memory bus into the Continuous-Wave (CW) generators.

- **Classical State ($|0\rangle$):** Fully constructive interference, pulling the DAC output to the logical-zero threshold.
- **Superposition State ($H|0\rangle$):** Q-OS injects a deterministic phase shift (e.g., $\Delta\phi = \pi/2$) into the SPYSPI wave generator.

The DAC outputs a modulated continuous analog voltage (V_{out}) actively stabilized by the SPHY waves (acting as the simulated Gravity Hamiltonian, $\hat{H}_\Phi$), remaining locked in a mimetic superposition until the measurement collapse.

4 Q-OS Web UI and User Experience

The complex translation of SPHY waves and analog DAC/ADC loops is completely abstracted away from the end-user. Interaction is handled through the **Q-OS Quantum Dashboard**, a lightweight web application hosted directly on the FPGA.

4.1 The Browser-Based IDE and Visualization

Users can import standard frameworks directly in the browser. As the user types their Python code, the UI dynamically renders the corresponding wire-and-block quantum circuit diagram, providing immediate visual feedback.

4.2 Mimetic Hardware Telemetry

Q-OS provides real-time telemetry of the local FPGA fabric:

- **Waveform Monitors:** Live visualizations of HPHYSPI and SPYSPI generation.
- **DAC/ADC Loop Status:** Real-time readout of the analog voltage gradients hosting the mimetic superposition.
- **Thermodynamic Coherence Indicator:** Verifies that the artificial $\hat{H}_\Phi$ reservoir is active, mathematically guaranteeing the deterministic coherence state ($\Delta S = 0$).

5 Deployment and Bitstream Architecture

Deploying Q-OS is designed to be a frictionless experience, circumventing standard FPGA development toolchains by packaging the stack into a single boot image.

5.1 The Unified Boot Image (`BOOT.bin`)

For System-on-Chip FPGAs like the Alinx Zynq series, Q-OS distributes a unified `BOOT.bin` file containing the First Stage Bootloader (FSBL), the Q-OS Linux Kernel, and the Mimetic Bitstream (`.bit` file).

5.2 Plug-and-Play Initialization

The user simply flashes the `BOOT.bin` onto a MicroSD card, inserts it into the FPGA, and connects it to their local network. As the ARM cores boot, Q-OS automatically flashes the `.bit` file directly into the FPGA fabric. Once booted, the device acquires a local IP address, and the user simply navigates to that address in their browser to access the Quantum Dashboard.