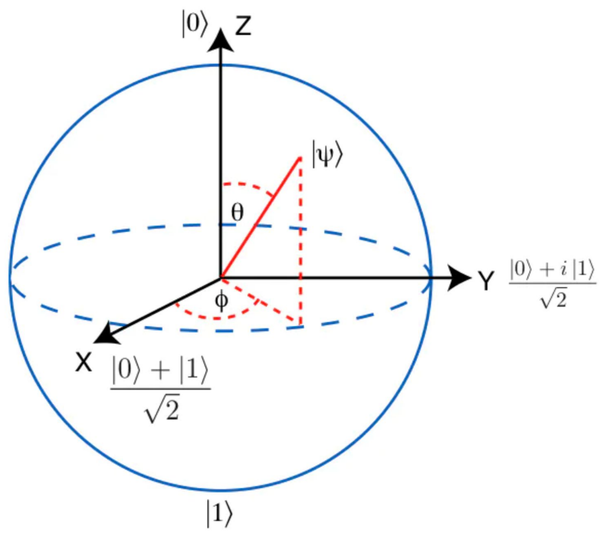
Modeling a Novel Quantum Computer Architecture

Introduction

Our clients have proposed novel ion-trap quantum computing hardware design concepts. Such novel concepts include a structure that allows for a ~1000 qubit system with ~10 nodes with ~100 qubits each, having 12 ion traps per node with ~8-10 qubits per ion trap. Traps may pass qubits between each other within a node with novel qubit handoff designs. Nodes may communicate with each-other via standard digital signals after qubits are evaluated. Our team was tasked with designing this hardware, and with developing software to schedule quantum algorithms on such hardware.

Ion Trap Quantum Computing in 100 Words

- Qubits are a quantum store of information that are more complex and powerful than bits in classical computers. Their information can be portrayed by a "Bloch Sphere"
- The electrons in Yb^+ ions can behave as qubits under the right conditions
- "Ion-Trap" computers suspend Yb^+ ions in magnetic fields and induce a superposition on the electron by using lasers to modify their state electron state
- An individual trap can hold ~10 ions
- For reliability purposes, 1 ion ≠ 1 qubit at the logical level. Many physical qubits are "combined" to increase computational accuracy
- Quantum volume is a complex metric for computational power. The current record is held by Honeywell's H1 machine, which uses a similar multi-trap architecture called a QCCD. It has a 2^{15} Q.V.

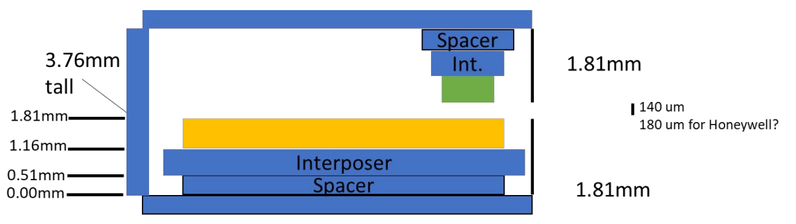


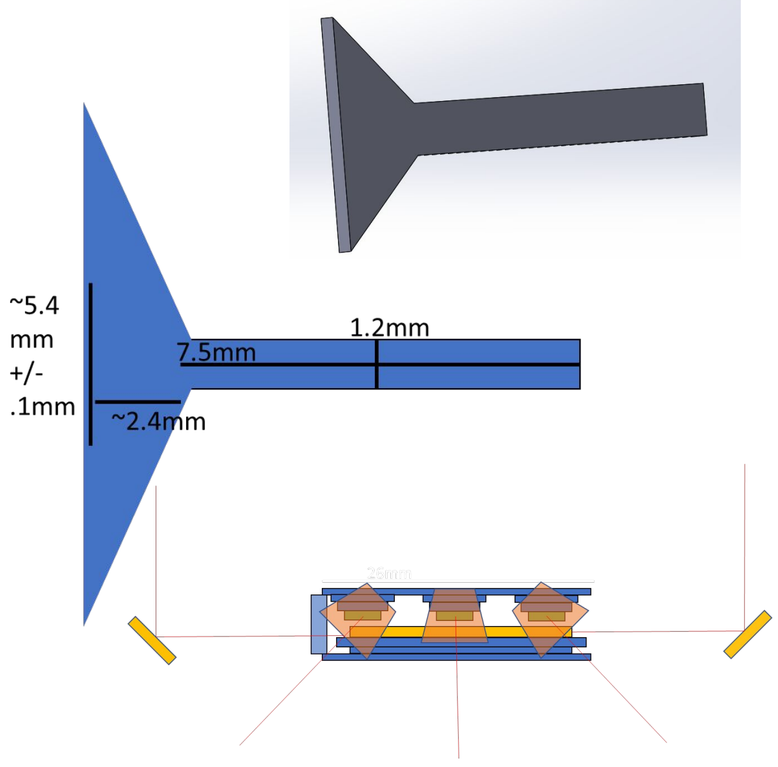
Hardware Overview

Methodology:

- Utilize the High-Optical Access (HOA) Trap 2.0 created by Sandia National Laboratory as a "standard" for trap hardware
- Discuss design iterations with client and make adjustments as needed

Implementation - Diagrams:





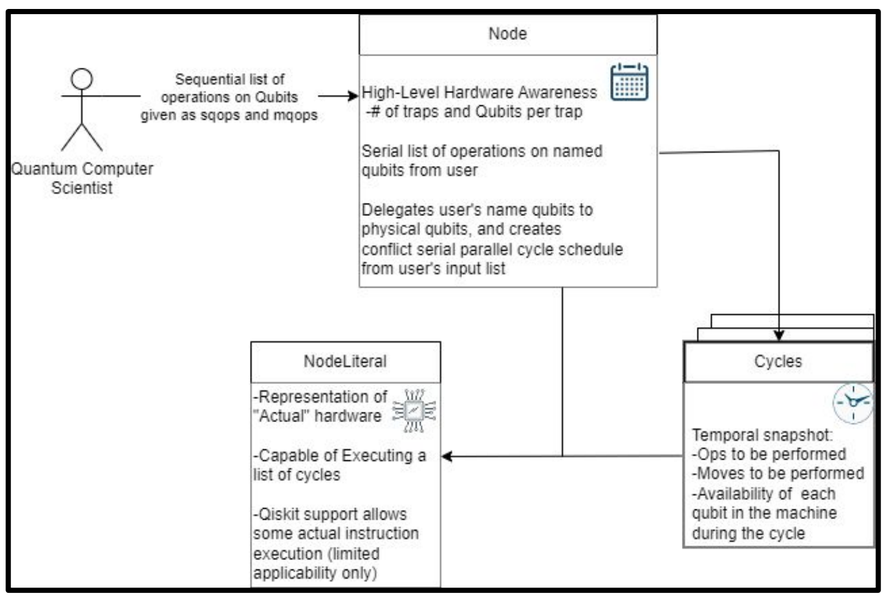
Results/Accomplishments:

- Measurements for HOA-like ion trap and related components
- Visual layout of quantum computer and necessary ancillary hardware
- Solidworks schematic for ion trap and chips

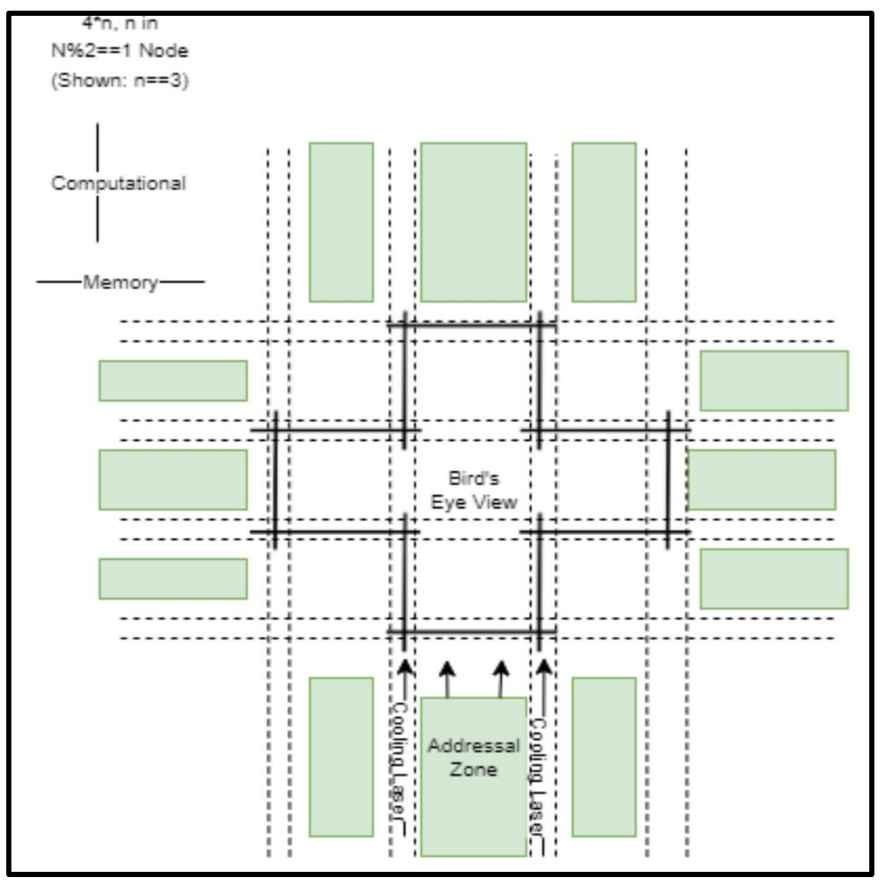
Impact:

- Contributed to a novel QC architecture project that is the 1st of its kind at ISU
- Helped to address architecture viability without fabrication

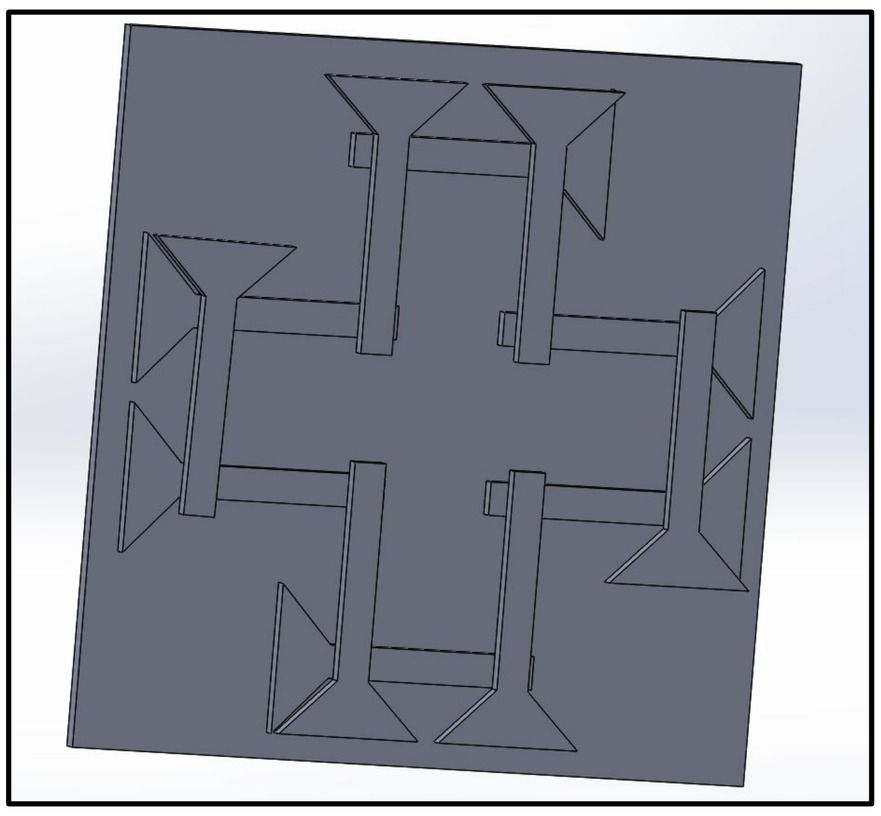
Design Overview



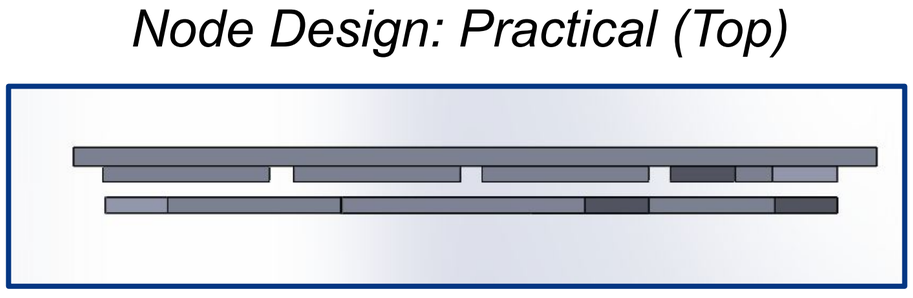
Software Design



Node Design: Concept



Node Design: Practical (Top)



Node Design: Practical (Side)

Software Overview

Methodology:

- Acquire client's requirements and conduct research
- Implement requirements, guided by research. Focus on expansion and maintainability.
- Advisory review
- Iterate

Implementation - Three Classes:

- Node: Easy to use Interface
- NodeLiteral: Hardware Standard Interface
- Cycle: Models a machine cycle

Results/Accomplishments:

- We created software that is capable of scheduling quantum operations
- Our software is abstracted and documented in a way that allows future teams to expand upon our work

Impact:

- Contributed to a novel QC architecture project that is the 1st of its kind at ISU
- Helped to address architecture viability without fabrication
- Extensible for very general hardware implementations

```
# Make A Node
n = Node(8)
n.sop(0,'x')
n.sop(1,'y')
n.sop(0,'z')
n.sop(2,'x')
n.sop(2,'z')
n.mop(0,3)
n.sop(0,'p',.5)

# Print Description To Validate
print("NODE DESCRIPTION")
n.describe()

# Generate NodeLiteral
l = n.getLiteral()

# Print Description To Validate
print("NODELITERAL DESCRIPTION")
l.describe()

# Generate Circuit To Validate
circ = l.getcirc()
print(circ.draw('text'))

# Generate And Execute A Circuit For Validation
backend = BasicAer.get_backend('qasm_simulator')
r = l.execute(backend)
print(r.get_counts())
print("END PROGRAM")
```

User Code Example