HPCA 2025 Tutorial

Janus 3.0: A Software Framework for Analyzing, Optimizing, Verifying, and Implementing Quantum Circuit

Organizers: Jianwei Yin, **Liqiang Lu**, Siwei Tan, Tianyao Chu

College of Computer Science and Technology
Zhejiang University (ZJU)

https://janusq.github.io/HPCA_2025_Tutorial/

Self-introduction

Liqiang Lu
liqianglu@zju.edu.cn

Liqiang Lu is a ZJU100 Young Professor in the College of Computer Science, Zhejiang University. His research interests include quantum computing, computer architecture, deep learning accelerator, and software-hardware codesign. He has authored more than 30 scientific publications in premier international journals and conferences in related domains, including ISCA, MICRO, HPCA, ASPLOS, FCCM, DAC, IEEE Micro, and TCAD. He also serves as a TPC member in the premier conferences in the related domain, including DAC, ICCAD, FPT, HPCC, etc.

[HPCA 2025] Debin Xiang, Qifan Jiang, **Liqiang Lu**, et al. “Choco-Q: Commute Hamiltonian-based QAOA for Constrained Binary Optimization”.
[ASPLOS 2024] Siwei Tan, **Liqiang Lu**, Hanyu Zhang, et al. “QuFEM: Fast and Accurate Quantum Readout Calibration Using the Finite Element Method”.
[MICRO 2021] Liqiang Lu, Yicheng Jin, Hangrui Bi, et al. “Sanger: A Co-Design Framework for Enabling Sparse Attention using Reconfigurable Architecture”.
[ISCA 2021] Liqiang Lu, Naiqing Guan, Yuyue Wang, et al. “TENET: A Framework for Modeling Tensor Dataflow Based on Relation-centric Notation”.

ACES-Q Research Group

ACES Lab
ADVANCED COMPUTING AND EMERGING SERVICE LAB

ACES-Q Research Group is a part of Advanced Computing and Emerging Service Lab in Zhejiang University led by Prof. Jianwei Yin. Our group focuses on **Quantum Computing** and consists of 3 faculty members and 16 students from the College of Computer Science and Technique and the School of Software Technology, Zhejiang University.

Milestone of Janus Quantum

ZHEJIANG UNIVERSITY

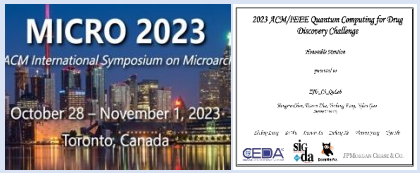
JanusQ Software 2.0

- Cluster I/O
- Quantum compilation

JanusQ Software

Achievements published in top conferences

- MICRO & ASPLOS
- ICCAD quantum chemistry competition



HPCA & Janus 3.0 tutorial

- Constrained binary optimization QAOA
- Second time of Janus tutorial

High Performance Computer Architecture

HPCA 2025

2022/07

2023/02

2023/10

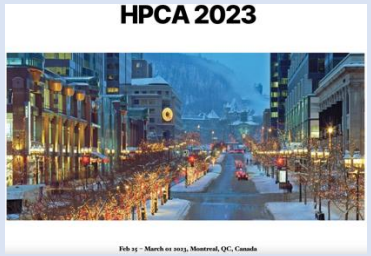
2024/04

2025/03

2025/06

Hybrid quantum-classical SAT solver

Revealed with JanusQ cloud in HPCA 2023



ASPLOS & Janus 2.0 tutorial

- Ultra-fast readout calibration
- First China quantum tutorial

Follow our work accepted by DAC

62 THE CHIPS TO SYSTEMS CONFERENCE

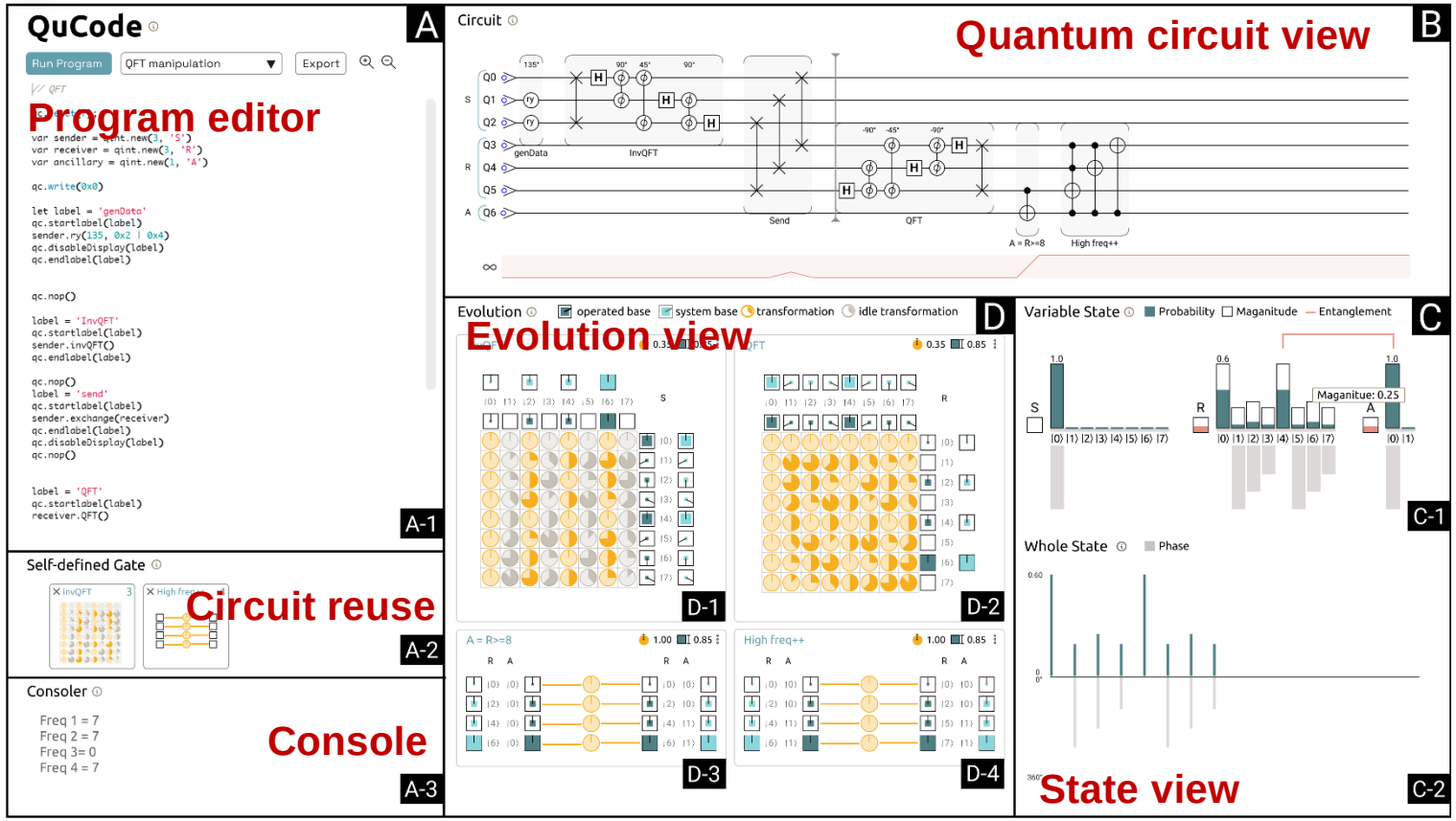
SHAPING THE NEXT GENERATION OF ELECTRONICS

Outline of Presentation

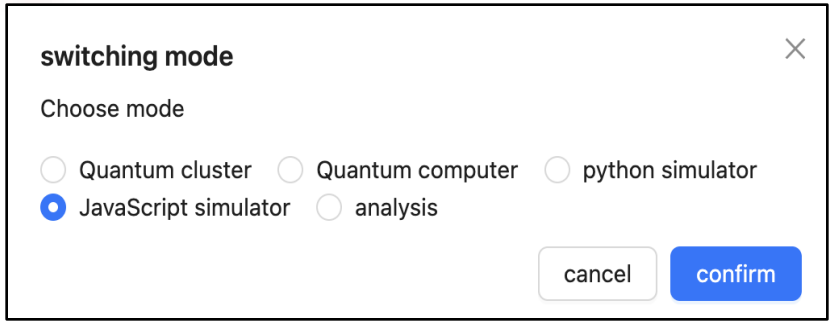
- **Tutorial Overview**
- Background Knowledge
- Mathematical Model of Quantum Computing
- Janus (Taiyuan) Quantum Cloud Platform
- Installing JanusQ

Janus Quantum Cloud

Coding interface of JanusQ



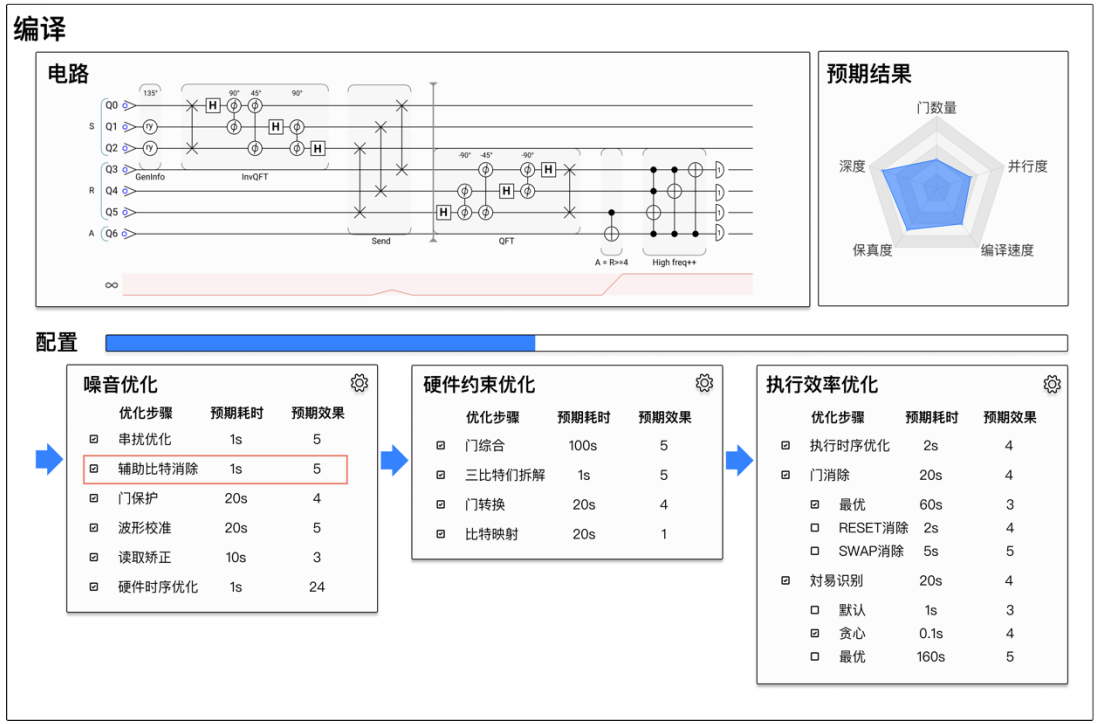
Execution modes



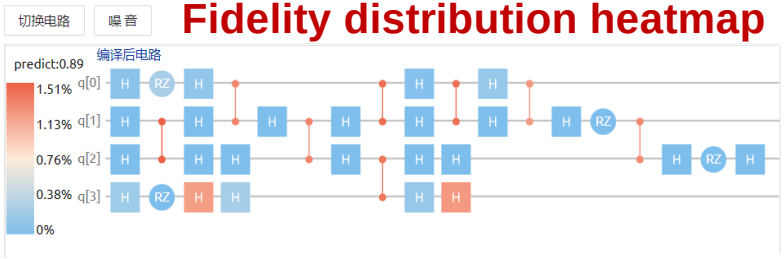
Provided quantum processors

Janus Quantum Cloud

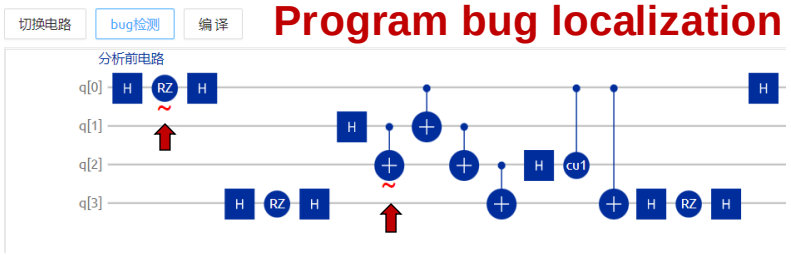
Quantum computing compilation optimization



Fidelity prediction and optimization



Program verification and correction

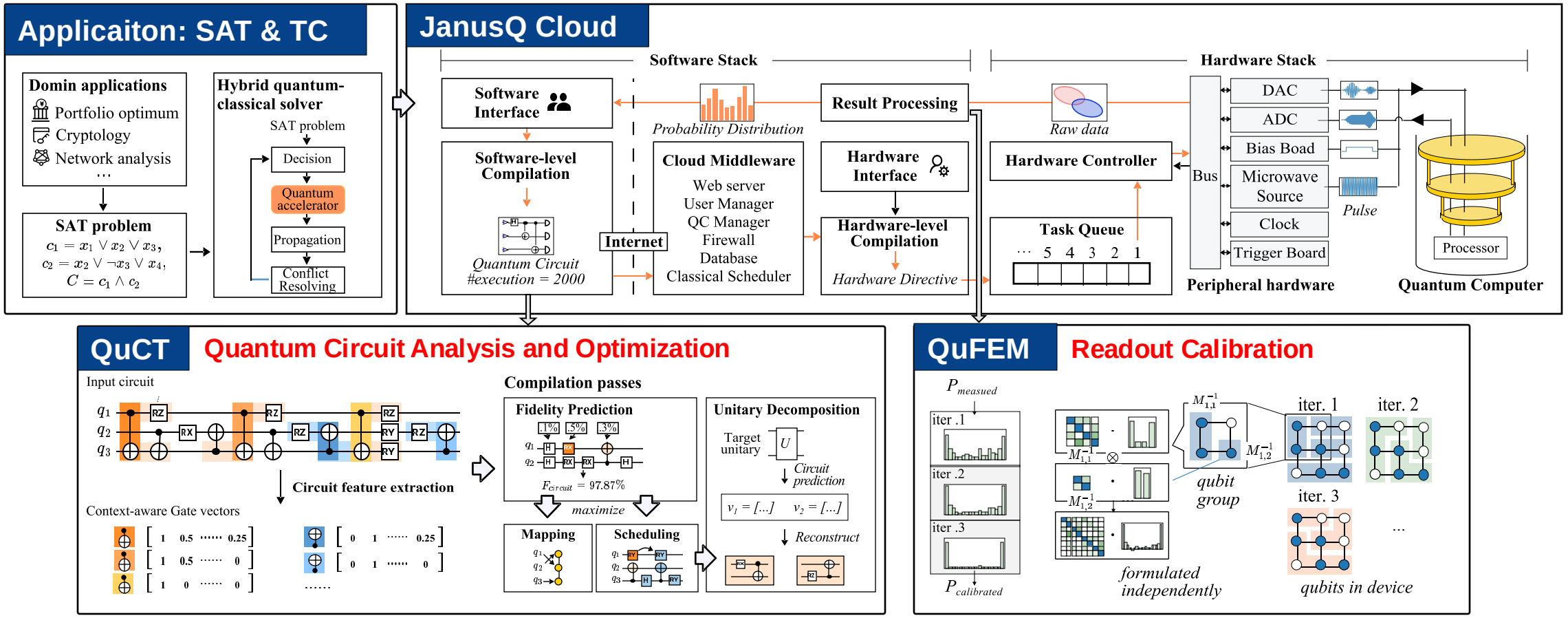


Support selection of compilation optimization options

1. **Noise optimization, reducing computation error rates**
2. **Quantum gate optimization, reducing the number of quantum gates**
3. **Compilation speed, reducing compilation time**

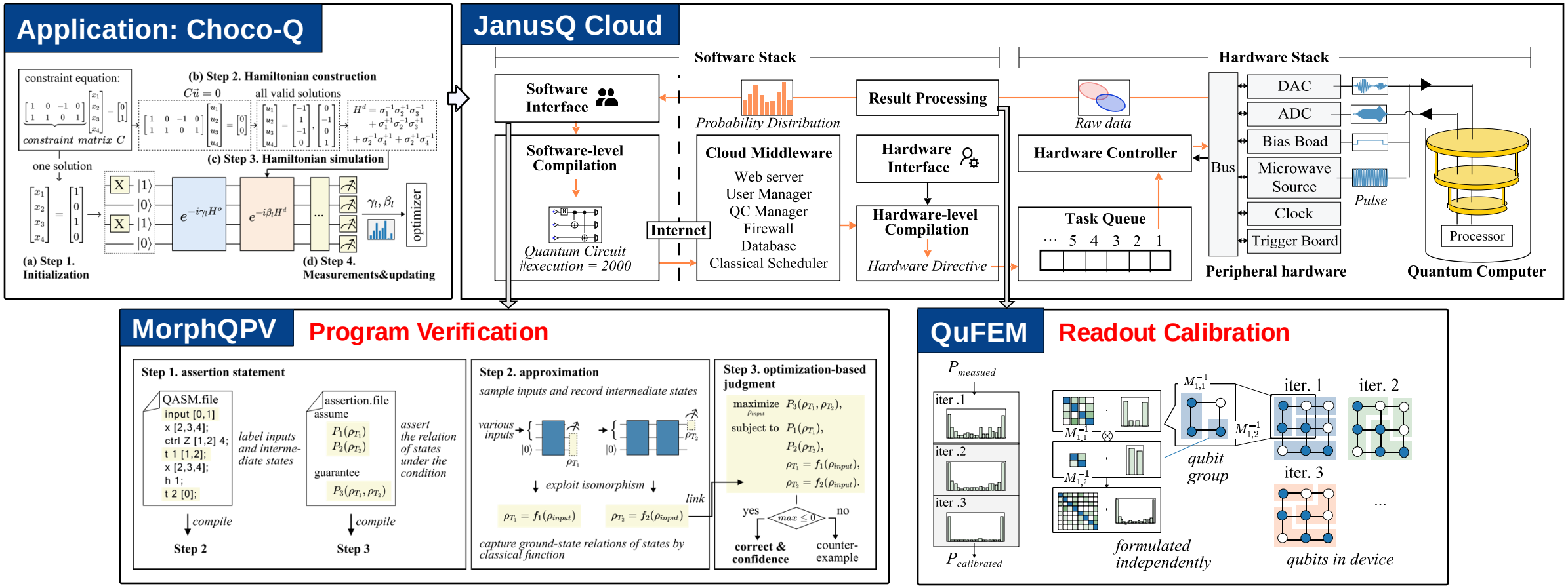
Janus 2.0 Review

Janus 2.0: A Software Framework for Analyzing, Optimizing and Implementing Quantum Circuit



Janus 3.0 Overview

Janus 3.0: A Software Framework for Analyzing, Optimizing, Verifying, and Implementing Quantum Circuit



Organizers

Jianwei Yin

Professor

zjuyjw@cs.zju.edu.cn

Liqiang Lu

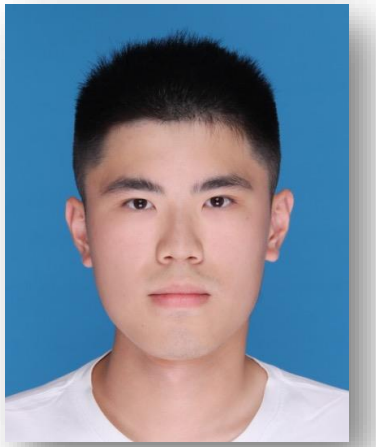
Assistant professor

liqianglu@zju.edu.cn

Siwei Tan

Assistant professor

siweitan@zju.edu.cn



Tianyao Chu

Ph.D. Student

tianyao_chu@zju.edu.cn

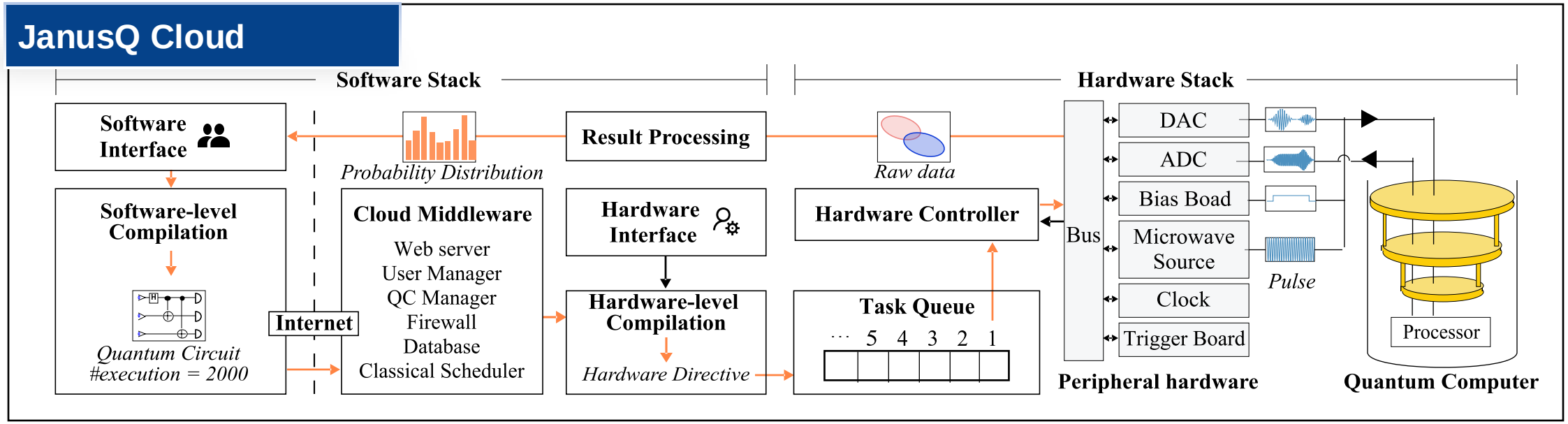
Outline

Topic	Presenter	Time
Topic-1. Introduction of Janus quantum cloud platform	Liqiang Lu	1:00pm – 1:30pm
Topic-2. QuCT Details (a) Vectorization model and code examples (b) Fidelity optimization & Unitary decomposition using gate vectors (c) Extending the framework by yourself: other downstream tasks!	Tianyao Chu	1:30pm – 2:15pm
Topic-3. MorphQPV Details (a) MorphQPV overview (b) Assertion statement (c) Implementation details of the automatic verification and repair	Siwei Tan	2:15pm – 3:00pm
Afternoon Break		3:00pm – 3:30pm

Outline

Topic	Presenter	Time
Topic-4. QuFEM Details (a) Characterization of readout error (b) Readout calibration using QuFEM	Kaiwen Zhou	3:30pm – 4:00pm
Topic-5. Choco-Q Details (a) Introduction of constained binary optimization problem (b) Choco-Q overview (c) Solve real-world problems with Choco-Q	Liqiang Lu	4:00pm – 4:45pm
Topic-6. Q & A	All	4:45pm – 5:00pm
Total		1:00pm – 5:00pm

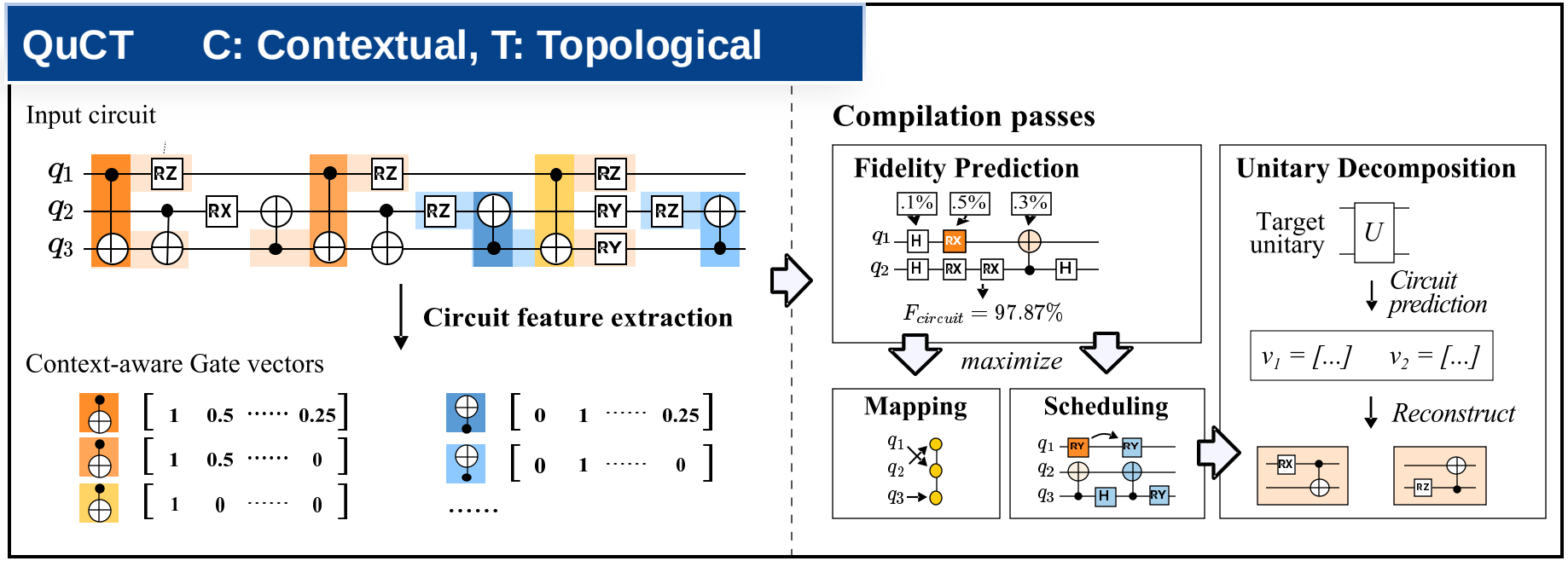
JanusQ Cloud



Janus Quantum Infrastructure:

- How do you use the code editor on the cloud platform?
- How do we submit the task to quantum hardware via API?
- How do we take advantage of JanusQ architecture?

QuCT

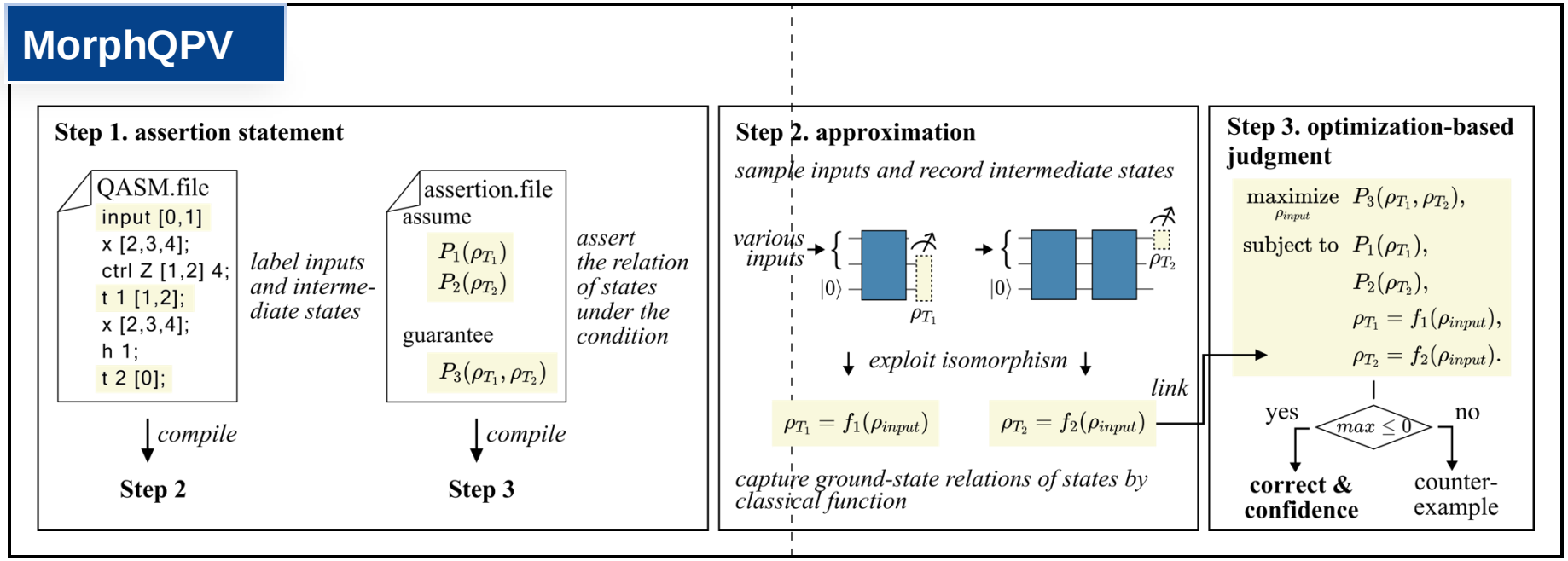


QuCT: Topology-aware quantum circuit optimizer (MICRO 2023)

- Why is topological information important in circuit analysis and optimization?
- How do we use topological information to analyze and optimize noise?
- How can we speed up the unitary decomposition with an upstream model?

Related paper: QuCT: A Framework for Analyzing Quantum Circuit by Extracting Contextual and Topological Features. **[MICRO 2023]**

MorphQPV



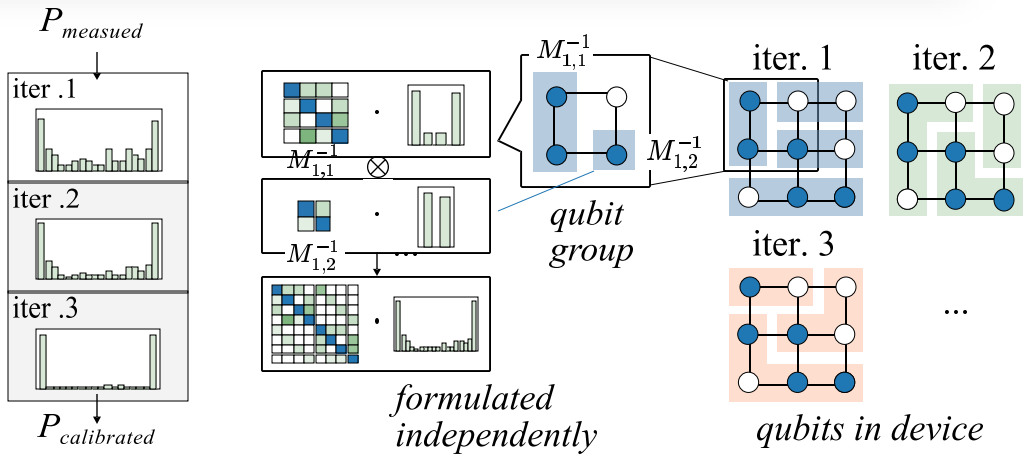
MorphQPV: Isomorphism-based quantum program verification (ASPLOS 2024)

- What is program verification in quantum circuit analysis? Why is it difficult?
- How do we use isomorphism relationship to verity quantum programs?
- How do we realize the assertion statement in quantum programs?

Related paper: MorphQPV: Exploiting Isomorphism in Quantum Programs to Facilitate Confident Verification. **[ASPLOS 2024]**

QuFEM

QuFEM FEM: Finite Element Method

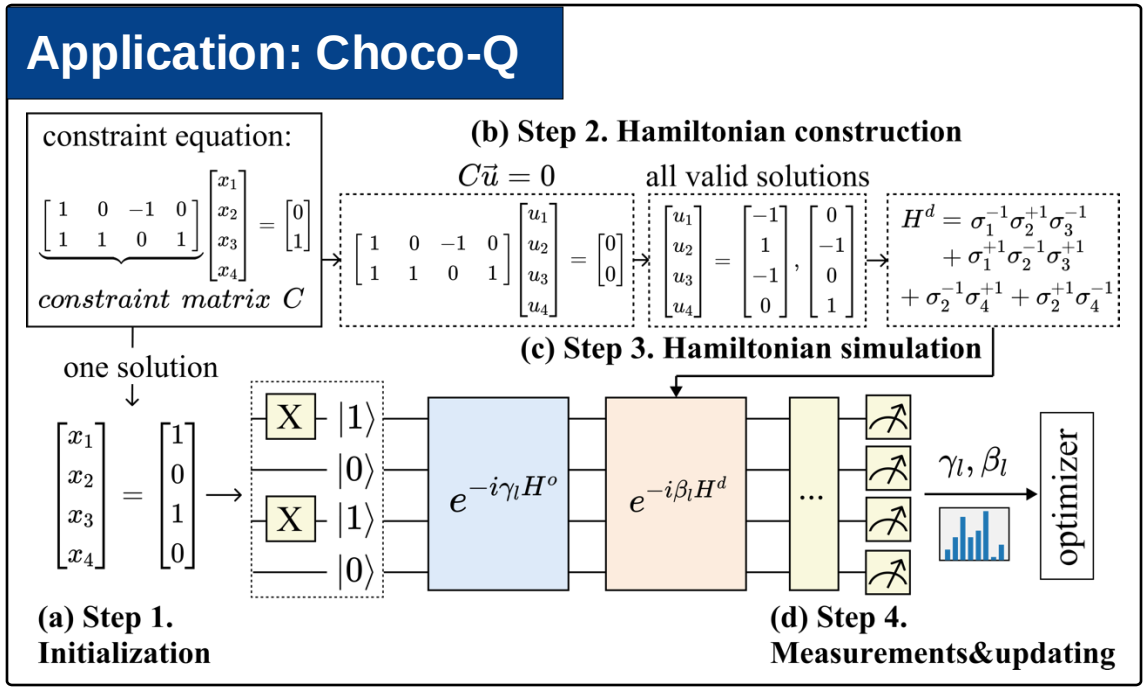


QuFEM: Readout Calibration based on finite element method (ASPLOS 2024)

- What is readout error and the difficulty of calibrating it?
- What is the finite element method?
- How do we mitigate the readout noise using the finite element method?

Related papers: QuFEM: Fast and Accurate Quantum Readout Calibration Using the Finite Element Method. **[ASPLOS 2024]**

Choco-Q



Choco-Q: Constrained Binary Optimization (HPCA 2025)

- What is constrained binary optimization?
- How do we use commute Hamiltonian to squeeze the searching space of QAOA?

Related papers:

Choco-Q: Commute Hamiltonian-based QAOA for Constrained Binary Optimization. **[HPCA 2025]**

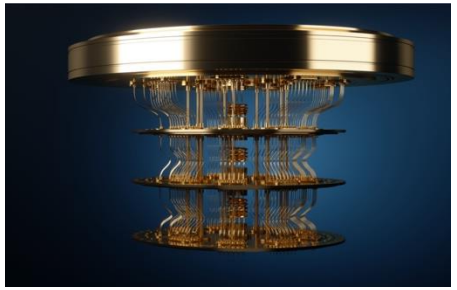
Outline of Presentation

- Tutorial Overview
- **Background Knowledge**
- Mathematical Model of Quantum Computing
- Janus (Taiyuan) Quantum Cloud Platform
- Installing JanusQ

Background Knowledge

Development of
Classical Computing

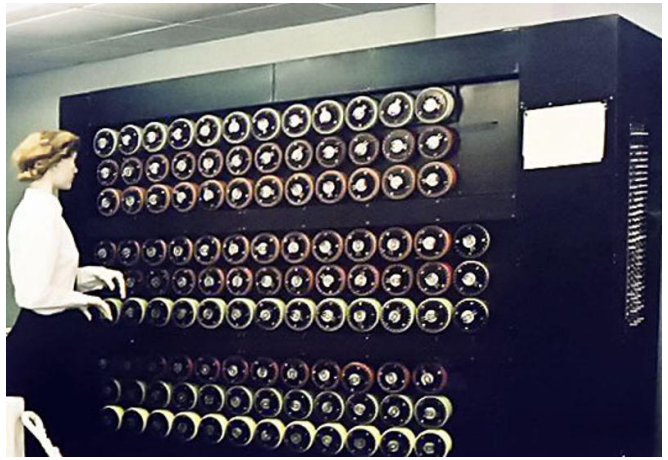
Motivation of
Quantum Computing



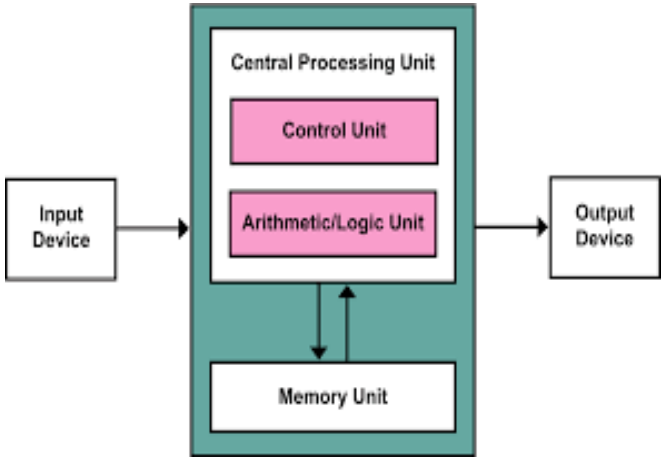
Development of
Quantum Computing

Development Of Classical Computing

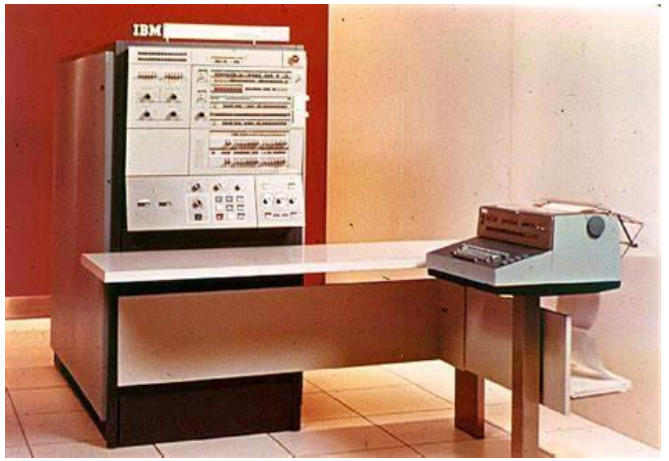
Domain-specific calculator （1939）



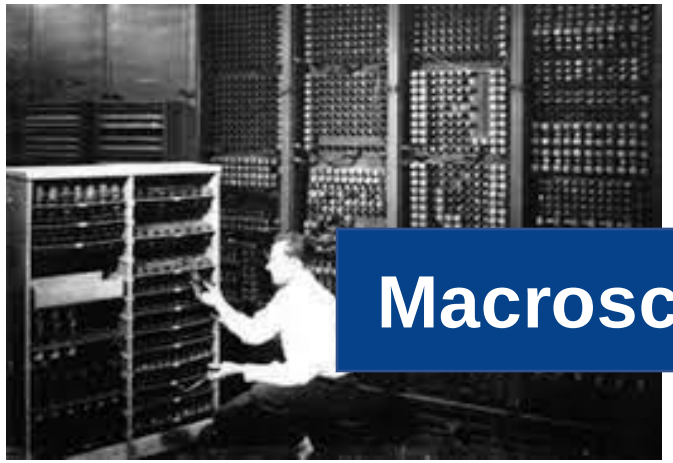
Von Neumann architecture (1947)



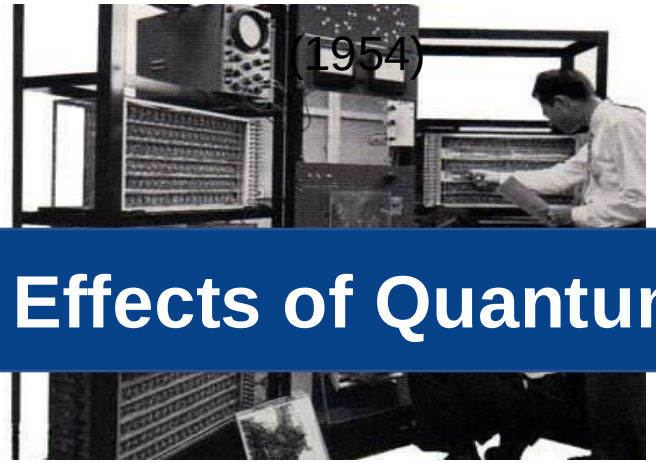
IBM360 Integrated Circuit (1964)



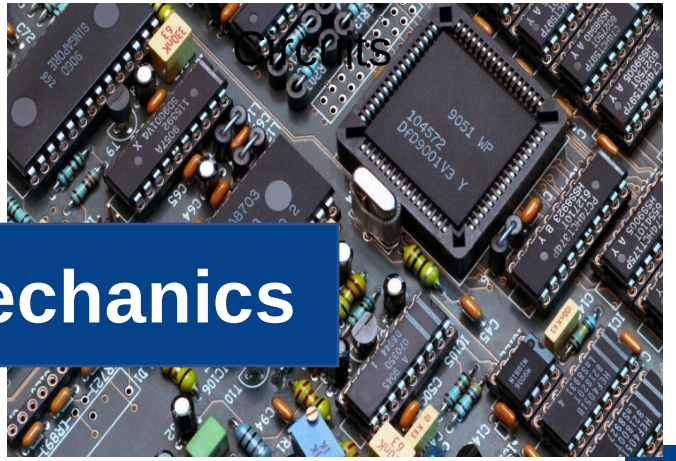
Vacuum Tube Computer ENIAC (1942)



Transistor Computer TRADIC (1954)



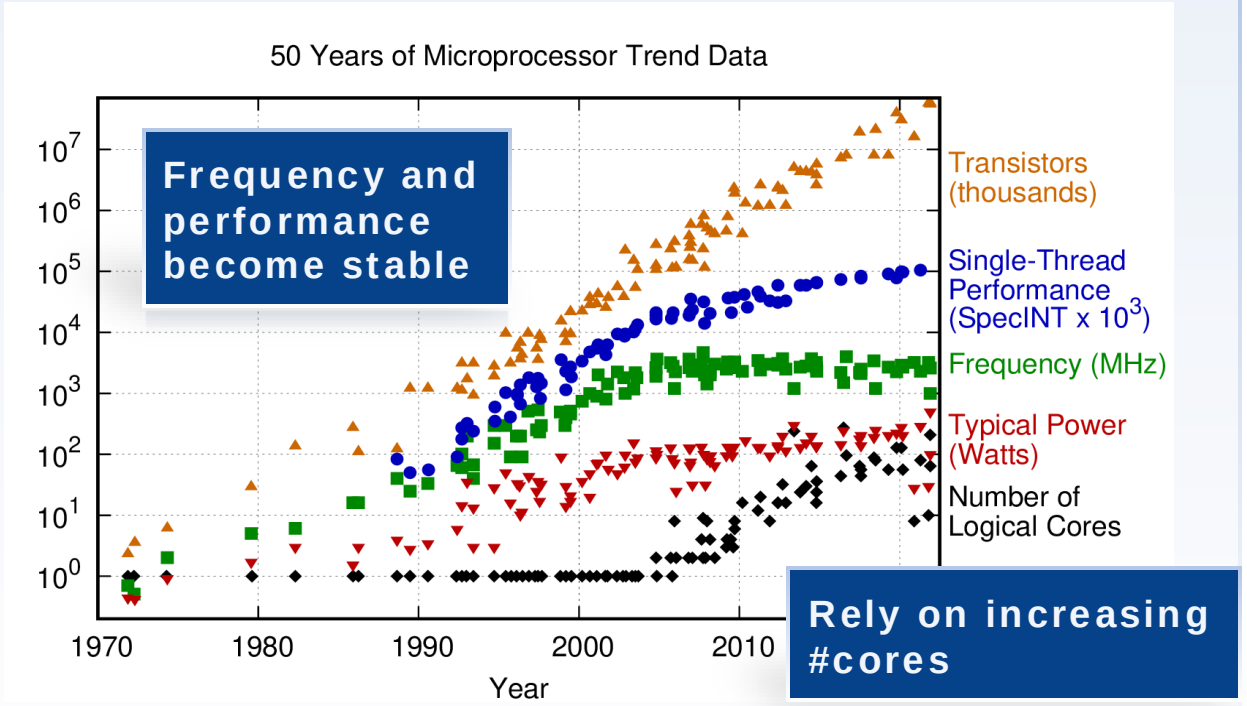
Large Scale Integrated Circuits



Macroscopic Effects of Quantum Mechanics

Motivation Of Quantum Computing

Computation barrier

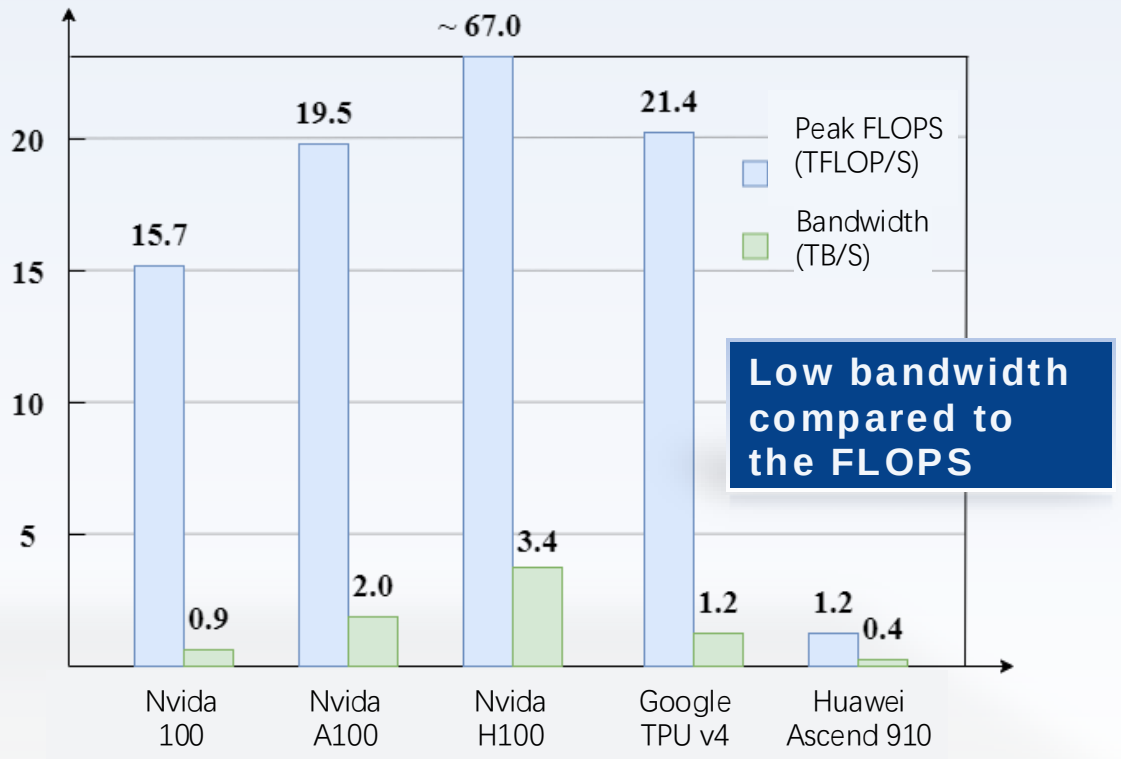


Original dalta up to the year 2010 colecled and plolted by M. Horowitz, F. Labonle, 0. Shacham, K. Olukolun, L. Hammond, and c. BalienNew plot and dala collected for 2010-2021 by K.Rupp

- The research costs and cycles of advanced chip processes are continuously increasing, Moore's Law is approaching obsolescence.
- The computing systems cannot rely solely on the development of traditional single chips. Instead, it requires new chip design methods and computing principles.

Motivation Of Quantum Computing

Storage barrier

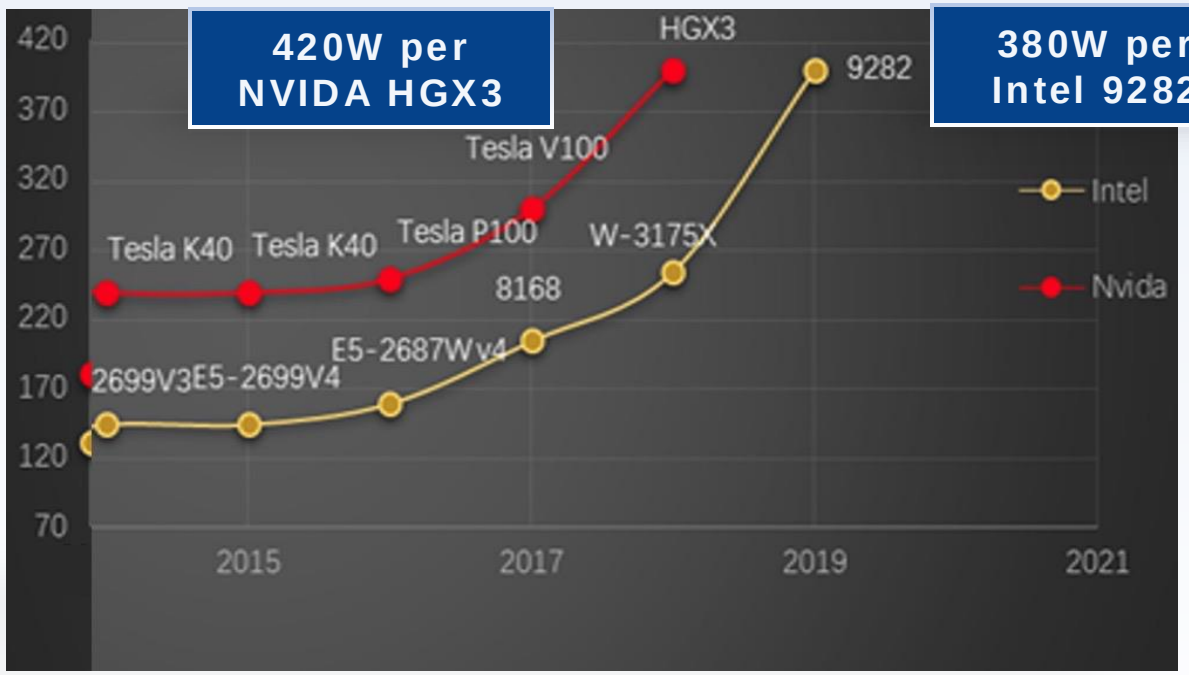


- Computation power and bandwidth is not matched.
- Latency of memory access is high, limiting CPU performance.
- Quantum computing is in memory.
- Non-von Neumann architectures.

Motivation Of Quantum Computing

Power wall

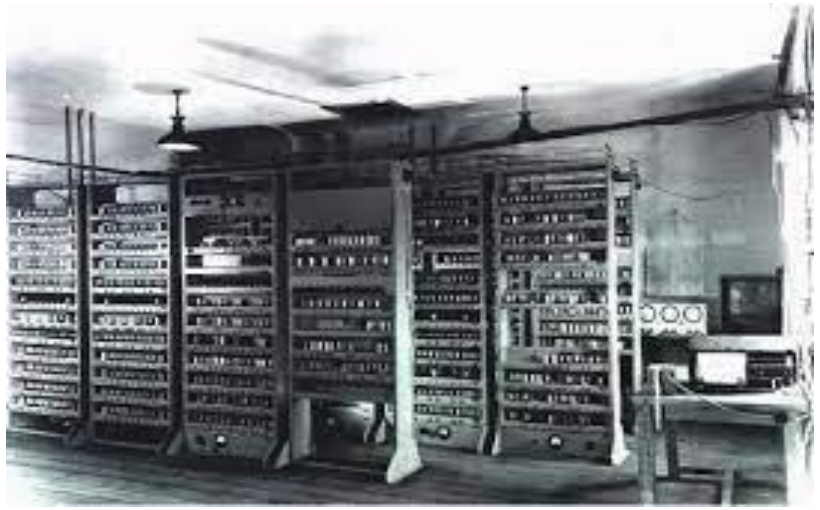
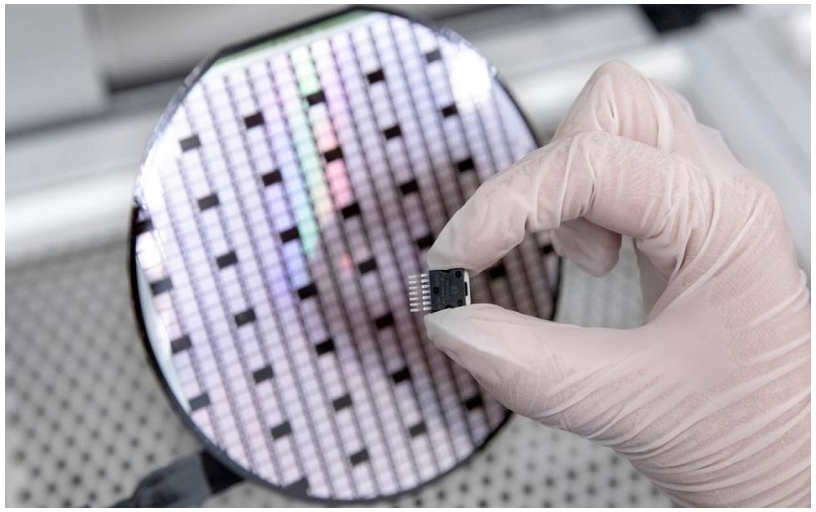
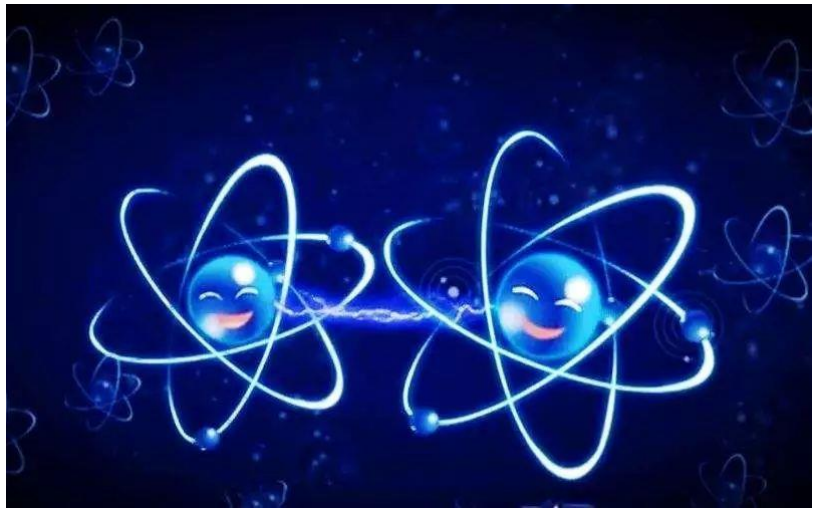
Thermal design power exponentially increases.



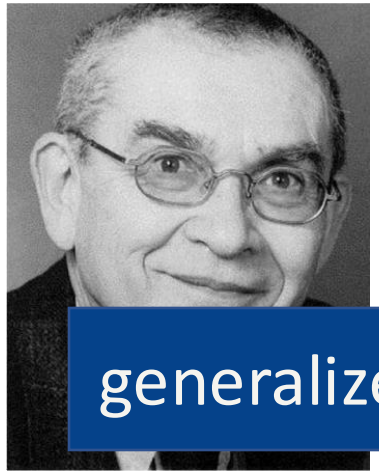
From https://www.blueocean-china.net/faq3/241.html

- The power consumption increases exponentially with computing power.
- Energy consumption restricts computing power.
- Systematical resource scheduling at the architectural level.

Proposal of Quantum Computing

- Classical physics is no longer able to fully describe the underlying physical mechanisms at its core.



Yuri Manin

- Algebraic Geometry
- Discrete Geometry (Diophantine Geometry)
- Manin (1980) and Feynman

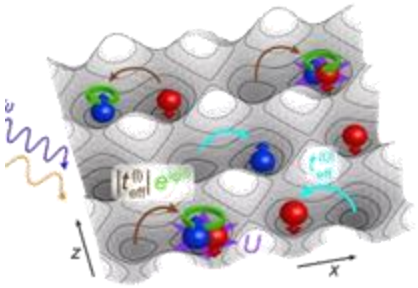
Richard Feynman

- Feynman Path Integral, Feynman Diagram, Feynman Parton Model
- Quantum Electrodynamics, Nobel Prize in Physics

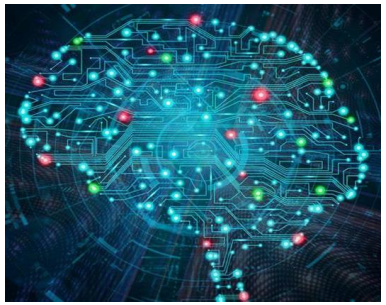
generalize to other domains such as the database search and cryptography

vely

Basic Concepts of Quantum Computing

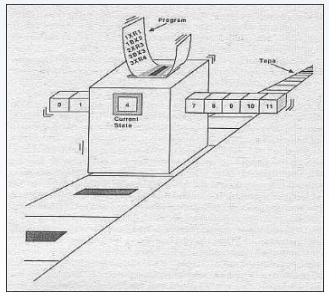
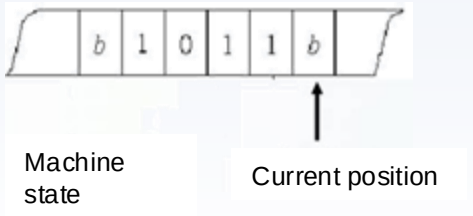
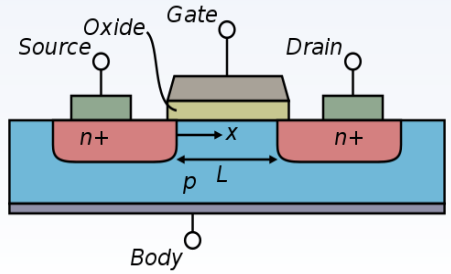
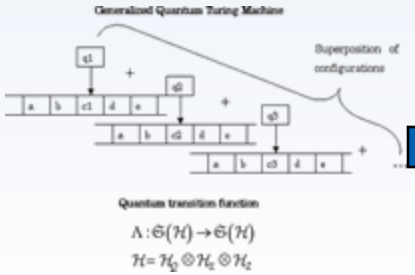
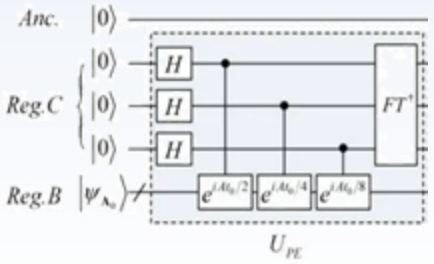
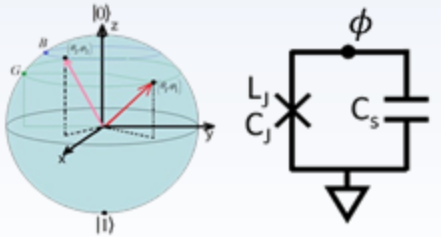


Physical Simulation

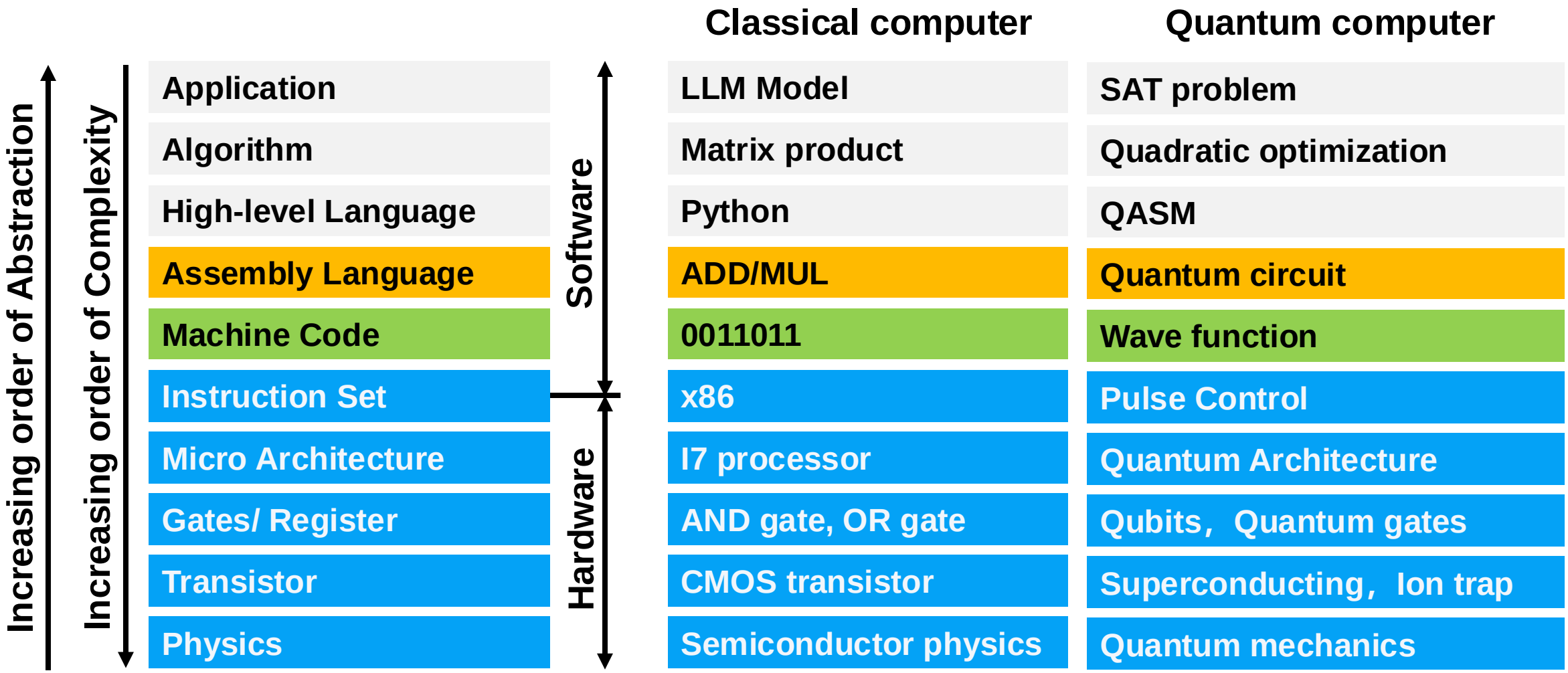


Factor

Application

	Computing theory	Encoding	Physical implementation
Classical program	**Classical Turing machine** 	**Program based on binary encoding** 	**CMOS (0/1)** 
Quantum program	**Quantum Turing machine** 	**Circuit-based quantum program** 	**Superconducting, photon (superposition state)**  Josephson junction
	lead to high computational advantages via parallelism.	*suffer from complexity due to quantum collapse and entanglement*	*has low energy consumption due to reversibility and superconductivity.*

Quantum Computing Architecture

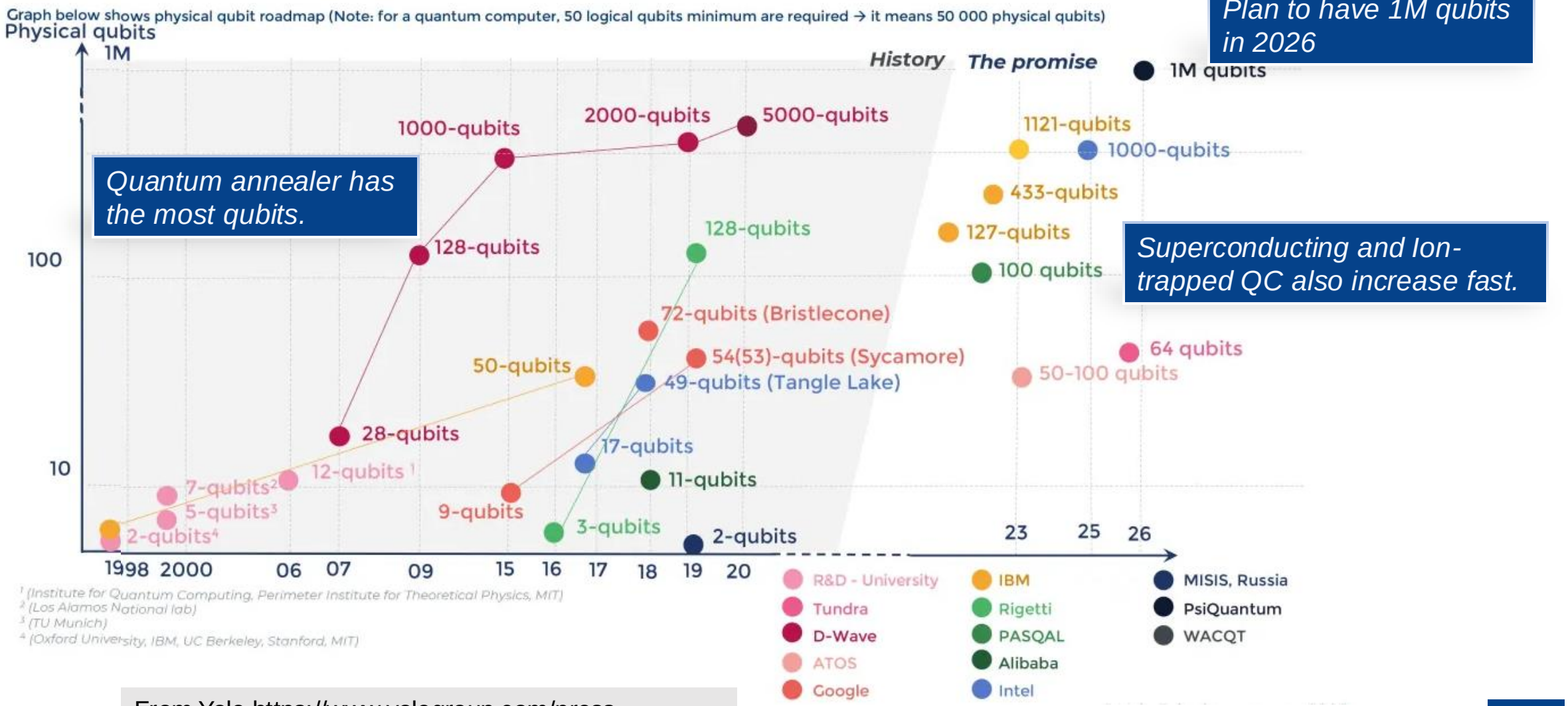


	Classical computer	Quantum computer
Application	LLM Model	SAT problem
Algorithm	Matrix product	Quadratic optimization
High-level Language	Python	QASM
Assembly Language	ADD/MUL	Quantum circuit
Machine Code	0011011	Wave function
Instruction Set	x86	Pulse Control
Micro Architecture	I7 processor	Quantum Architecture
Gates/ Register	AND gate, OR gate	Qubits，Quantum gates
Transistor	CMOS transistor	Superconducting，Ion trap
Physics	Semiconductor physics	Quantum mechanics

From https://www.secplicity.org/2018/09/19/understanding-the-layers-of-a-computer-system/

Classical and quantum computing has similar architecture

Development of Quantum Computer



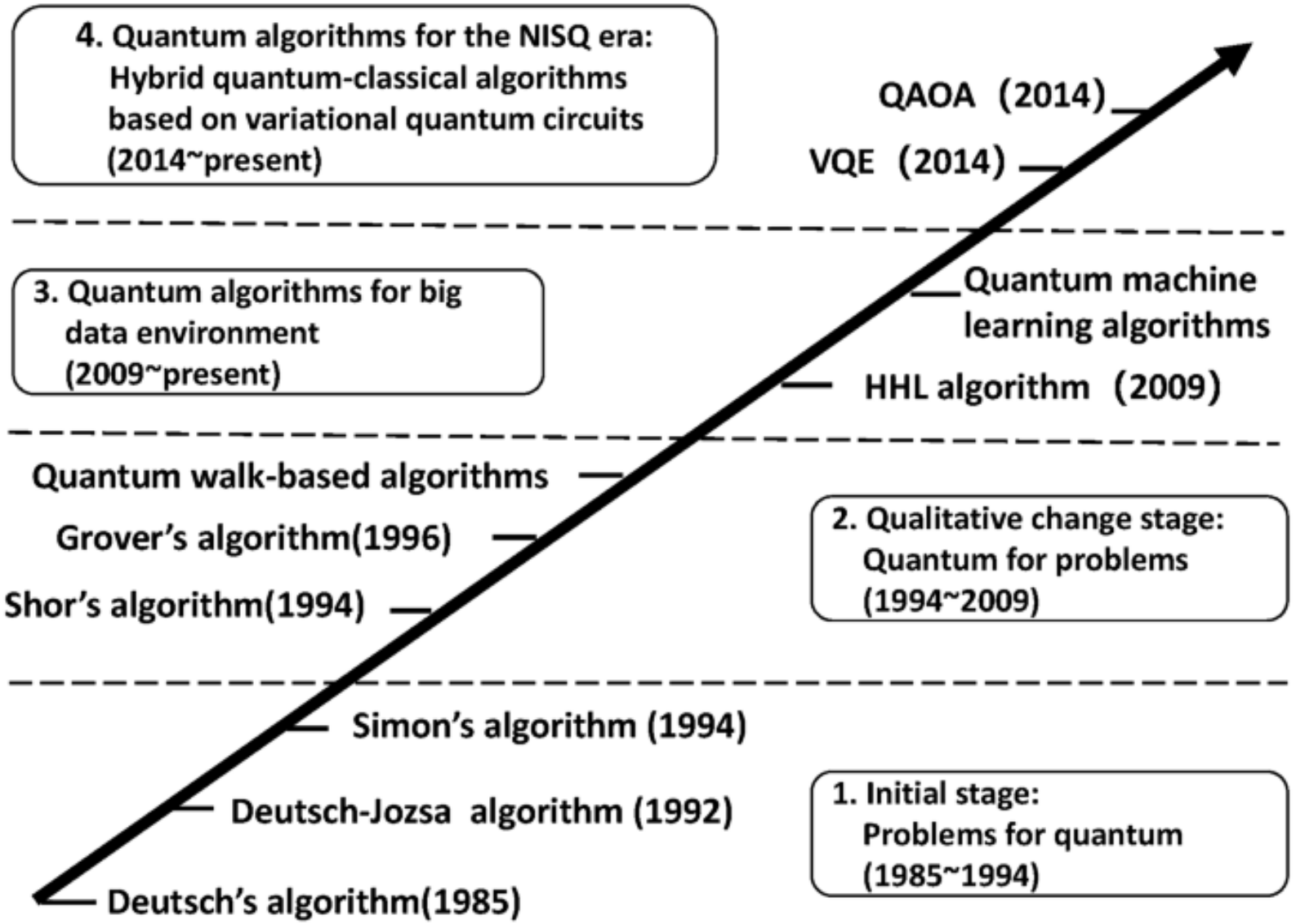
Plan to have 1M qubits in 2026

Quantum annealer has the most qubits.

Superconducting and Ion-trapped QC also increase fast.

From Yole https://www.yolegroup.com/press-release/quantum-technologies-the-market-is-growing/

Development of Quantum Algorithm



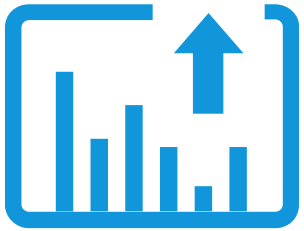
Zhang, S., Li, L. A brief introduction to quantum algorithms. *CCF Trans. HPC* **4**, 2022

Outline of Presentation

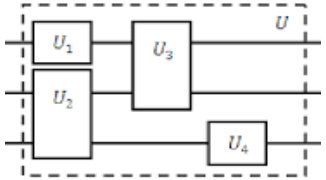
- Tutorial Overview
- Background Knowledge
- **Mathematical Model of Quantum Computing**
- Janus (Taiyuan) Quantum Cloud Platform
- Installing JanusQ

Mathematical Model of Quantum Computing

Qubits



Quantum Evolution



Quantum Circuit

Quantum Bit (Qubit)

Single qubit

A **qubit** has two bases $|0\rangle$ and $|1\rangle$. The information stored in its **superposition state** is represented as a **2-dimension state vector** $|\varphi\rangle$.

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle$$

$$|\varphi\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$$

subject to $|\alpha|^2 + |\beta|^2 = 1$

Quantum Bit (Qubit)

Single qubit

A **qubit** has two bases $|0\rangle$ and $|1\rangle$. The information stored in its **superposition state** is represented as a **2-dimension state vector** $|\varphi\rangle$.

$$|0\rangle = \begin{bmatrix}1\\0\end{bmatrix}$$

$$|1\rangle = \begin{bmatrix}0\\1\end{bmatrix}$$

$$|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle$$

$$|\varphi\rangle = \begin{bmatrix}\alpha\\\beta\end{bmatrix}$$

subject to $|\alpha|^2 + |\beta|^2 = 1$

Multiple qubits

***N* qubits** has 2^N bases $|00\cdots0\rangle$, $|00\cdots1\rangle\cdots$, $|11\cdots1\rangle$. The information stored in their **superposition state** is represented as a 2^N**-dimension state vector**.

$$|00\cdots0\rangle = \begin{bmatrix}1\\0\\\vdots\\0\end{bmatrix} \quad |00\cdots1\rangle = \begin{bmatrix}0\\1\\\vdots\\0\end{bmatrix} \quad \cdots \quad |11\cdots1\rangle = \begin{bmatrix}0\\0\\\vdots\\1\end{bmatrix}$$

$$|\varphi\rangle = \alpha_0|00\cdots0\rangle + \alpha_1|00\cdots1\rangle + \cdots + \alpha_{2^N}|11\cdots1\rangle$$

$$|\varphi\rangle = \begin{bmatrix}\alpha_0\\\alpha_1\\\vdots\\\alpha_{2^N}\end{bmatrix}$$

subject to $|\alpha_0|^2 + |\alpha_1|^2 + \cdots |\alpha_{2^N}|^2 = 1$

Quantum Evolution

Unitary matrix

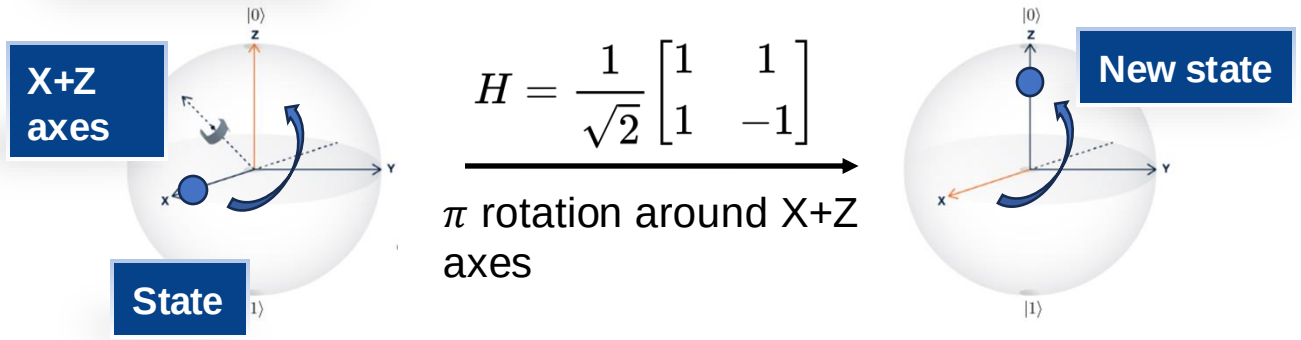
A quantum evolution caused by **quantum gates** is represented as a **unitary matrix (unitary),** which is a square matrix whose conjugate transpose is its inverse.

$$UU^{\dagger} = I$$

The evolution of qubit state $|\varphi\rangle$ is represented as:

$$|\varphi'\rangle = U|\varphi\rangle$$

Bloch sphere



Quantum Evolution

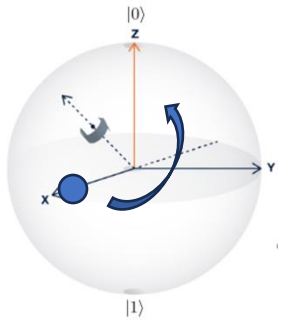
Unitary matrix

A quantum evolution caused by **quantum gates** is represented as a **unitary matrix (unitary),** which is a square matrix whose conjugate transpose is its inverse.

$$UU^{\dagger} = I$$

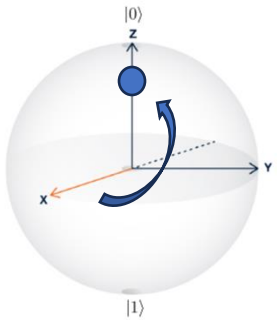
The evolution of qubit state $|\varphi\rangle$ is represented as:

$$|\varphi'\rangle = U|\varphi\rangle$$



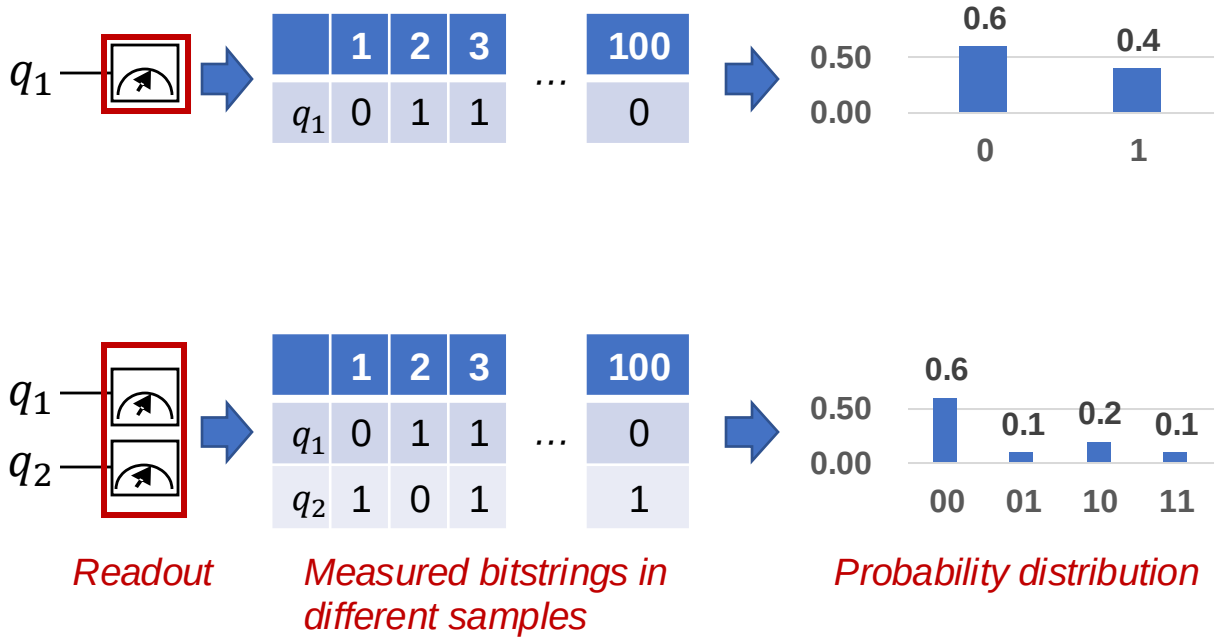
$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

π rotation around X+Z axis: Exchanges X and Z



Quantum readout

A sampling of a quantum state is a bitstring. Multiple sampling of this state composes a probability distribution of measuring different bitstrings.



Quantum Circuit

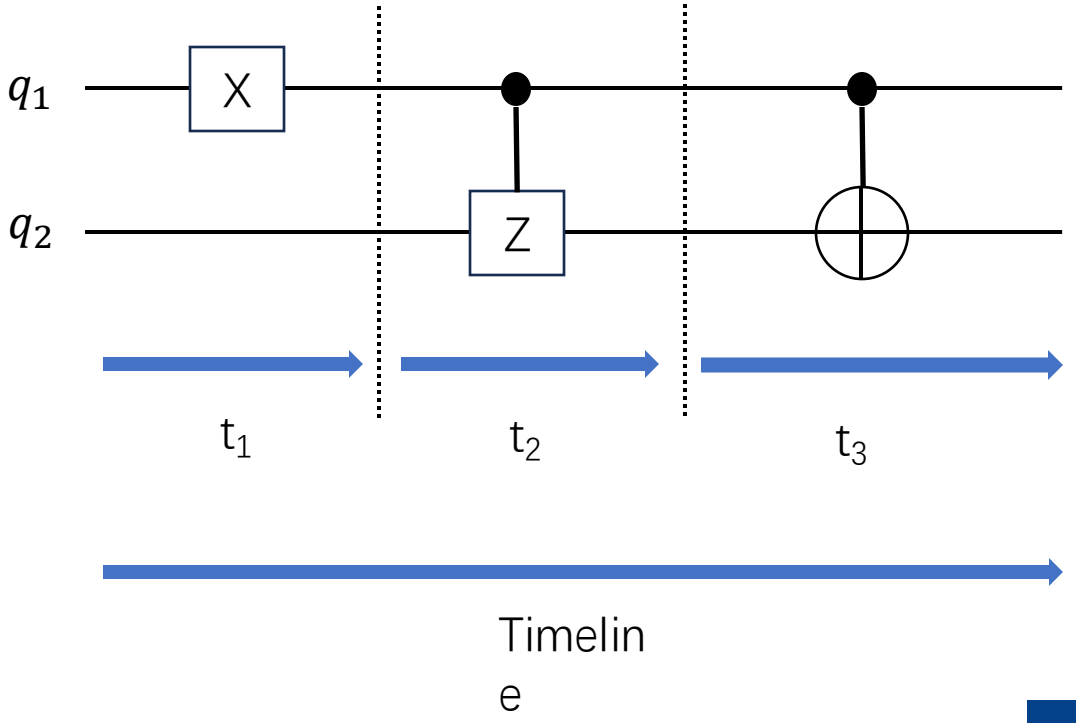
Quantum gates

In the quantum circuit model of computation, a **quantum gate** is a **basic quantum circuit operating** on a small number of qubits.

Operator	Gate(s)	Matrix
Pauli-X (X)	X	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli-Y (Y)	Y	$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
Controlled Not (CNOT, CX)		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
Controlled Z (CZ)	Z	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$
SWAP		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Toffoli (CCNOT, CCX, TOFF)		$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$

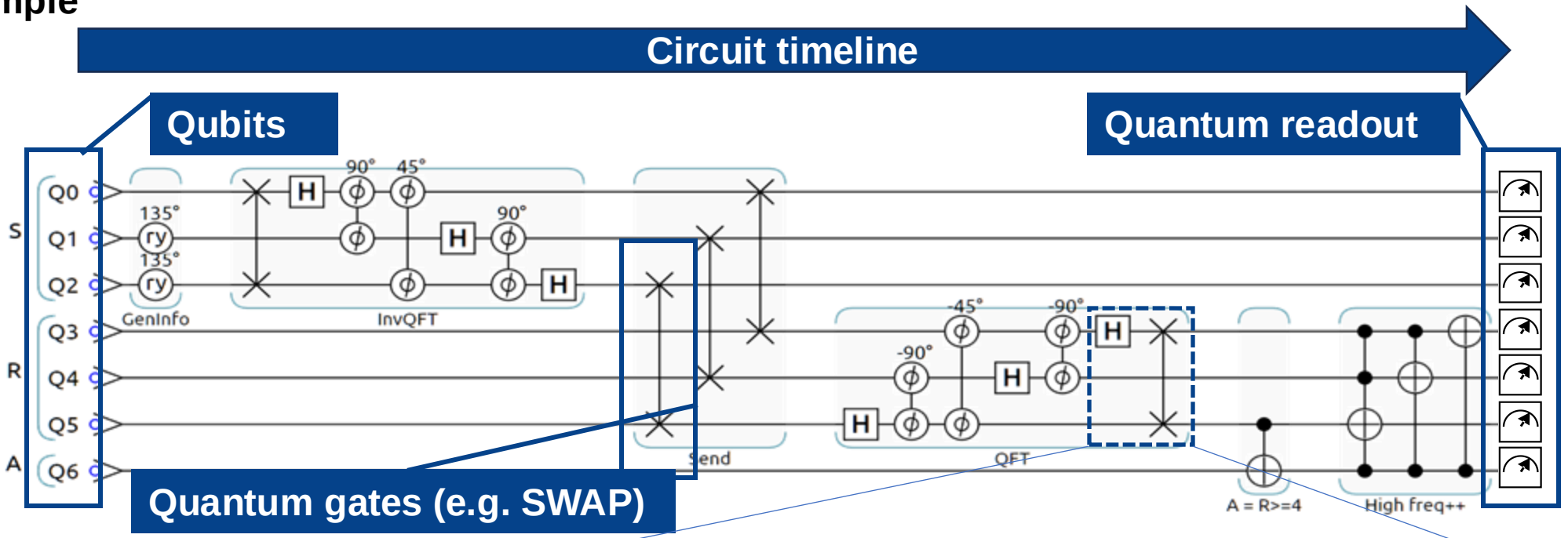
Qubit timeline

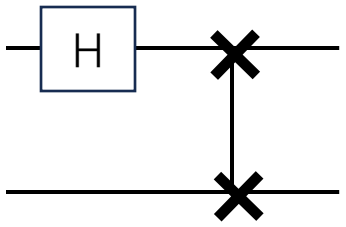
Each line in a quantum circuit represents **a qubit**. The quantum gate in a line is applied on the same qubits **from left to right in time direction.**



Quantum Circuit

For example





$$=$$

$$SWAP \cdot (H \otimes I) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & -1 \end{bmatrix}$$

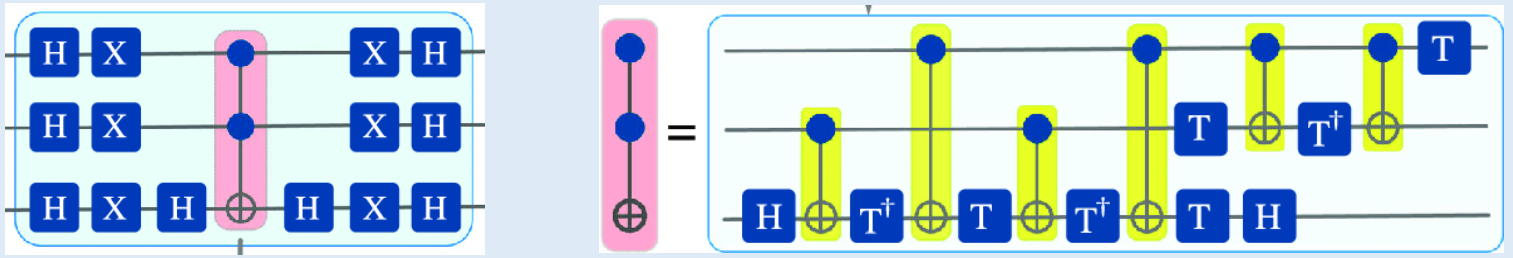
Implementation of Quantum Circuit

On superconducting quantum computer

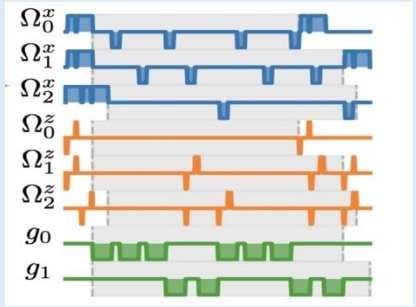
Step 1. Circuit statement

```
OPENQASM 2.0;
qreg qubits[3];
H qubits[1];H qubits[2];
H qubits[3];
X qubits[1];X qubits[2];
X qubits[3];
H qubits[3];
Toffoli qubits[1], qubits[2],
qubits[3];
H qubits[3];
X qubits[1];X qubits[2];
X qubits[3];
H qubits[1];H qubits[2];
H qubits[3];
```

Step 2. Circuit compilation



Step3. Circuit execution



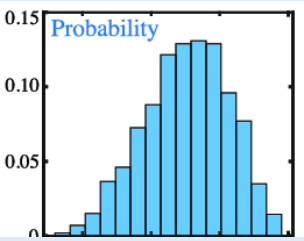
Pulse generation

Quantum device

Step4. Result processing

Error mitigation

Visualization

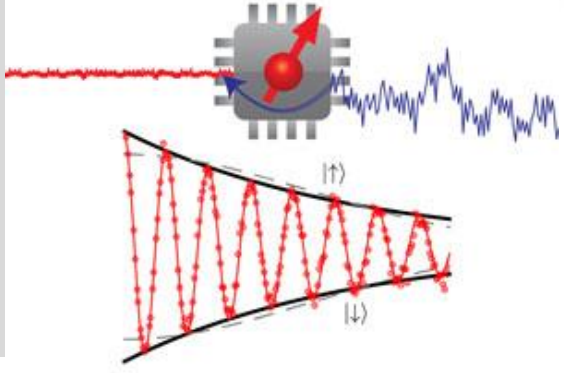


Probability distribution

Challenge in Quantum Computing

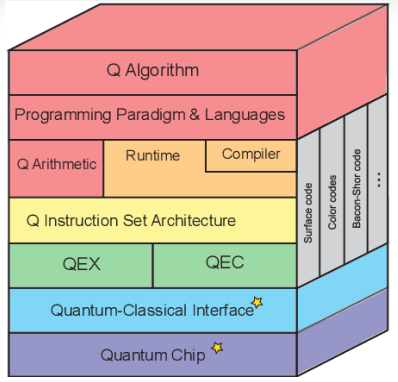
Noise

- Coherence error
- Gate error
- State preparation error
- Readout error
- Crosstalk



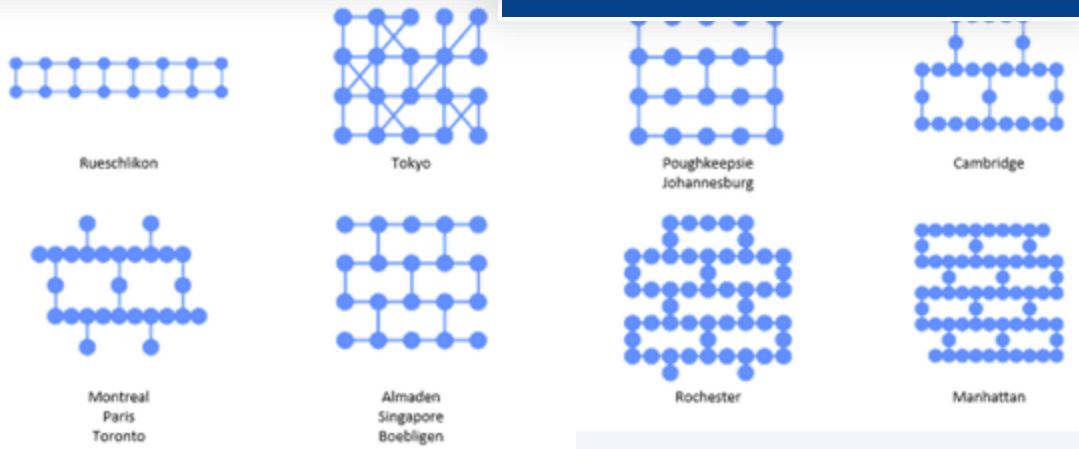
Instructions

- Multi-level compilation
- Micro-architecture instructions
- Quantum-classical communication



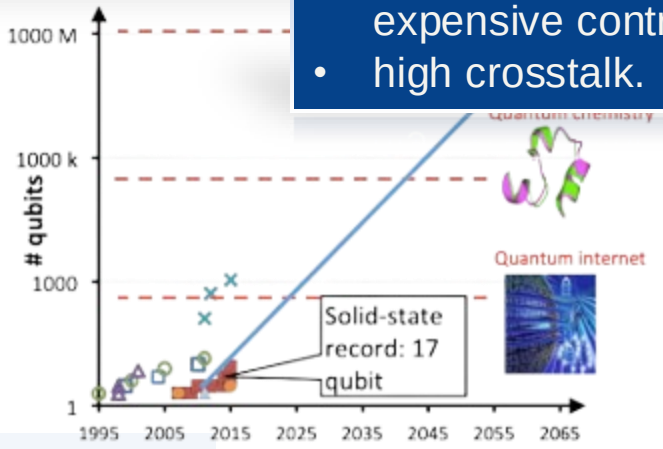
Topology

Locality in communication



Scalability

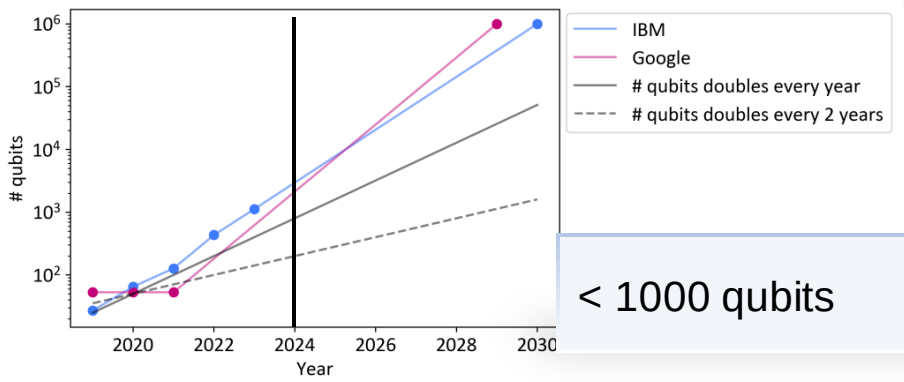
- poor fabrication techniques expensive control systems
- high crosstalk.



Constrained by physical implementation

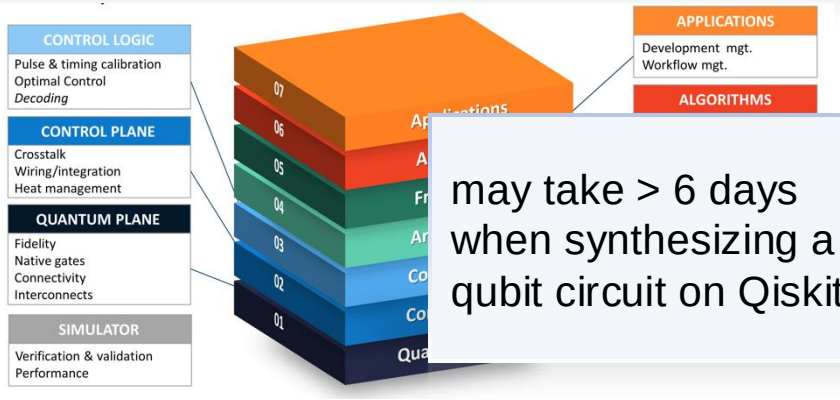
Results of Challenges

Limited Hardware Resource



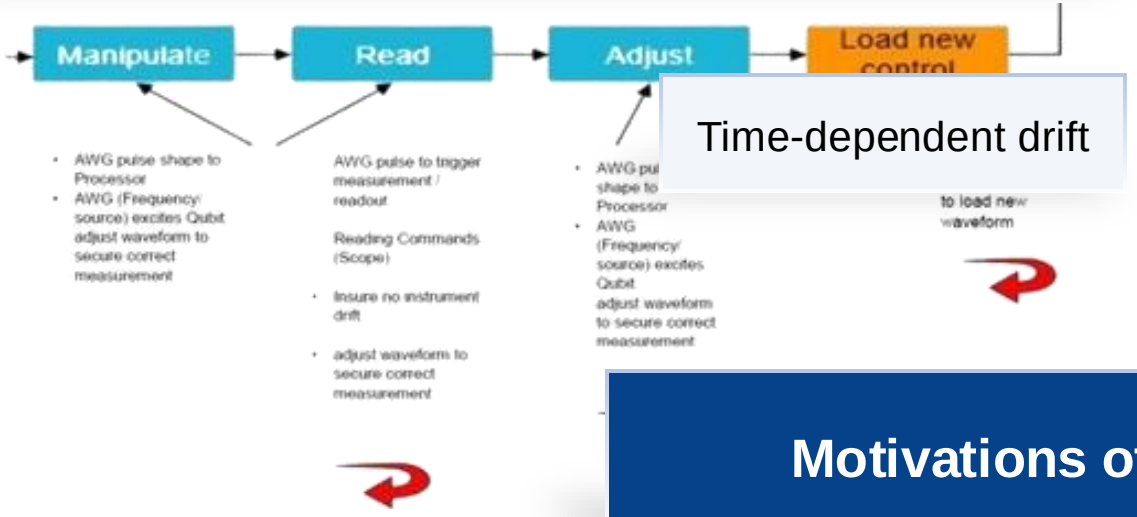
< 1000 qubits

Compilation Complexity



may take > 6 days
when synthesizing a 100-qubit circuit on Qiskit

Difficulty Of Hardware Calibration

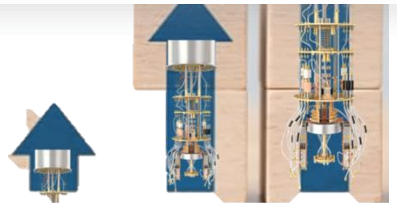


Time-dependent drift

Rare quantum advantage

- Limited qubits resources
- High error rate
- Hard to ensure quantum advantage in real applications

Long calibration time
(e.g., readout calibration)



Motivations of JanusQ

Outline of Presentation

- Tutorial Overview
- Background Knowledge
- Mathematical Model of Quantum Computing
- **Janus (Taiyuan) Quantum Cloud Platform**
- Installing JanusQ

Janus (Taiyuan) Quantum Cloud Platform

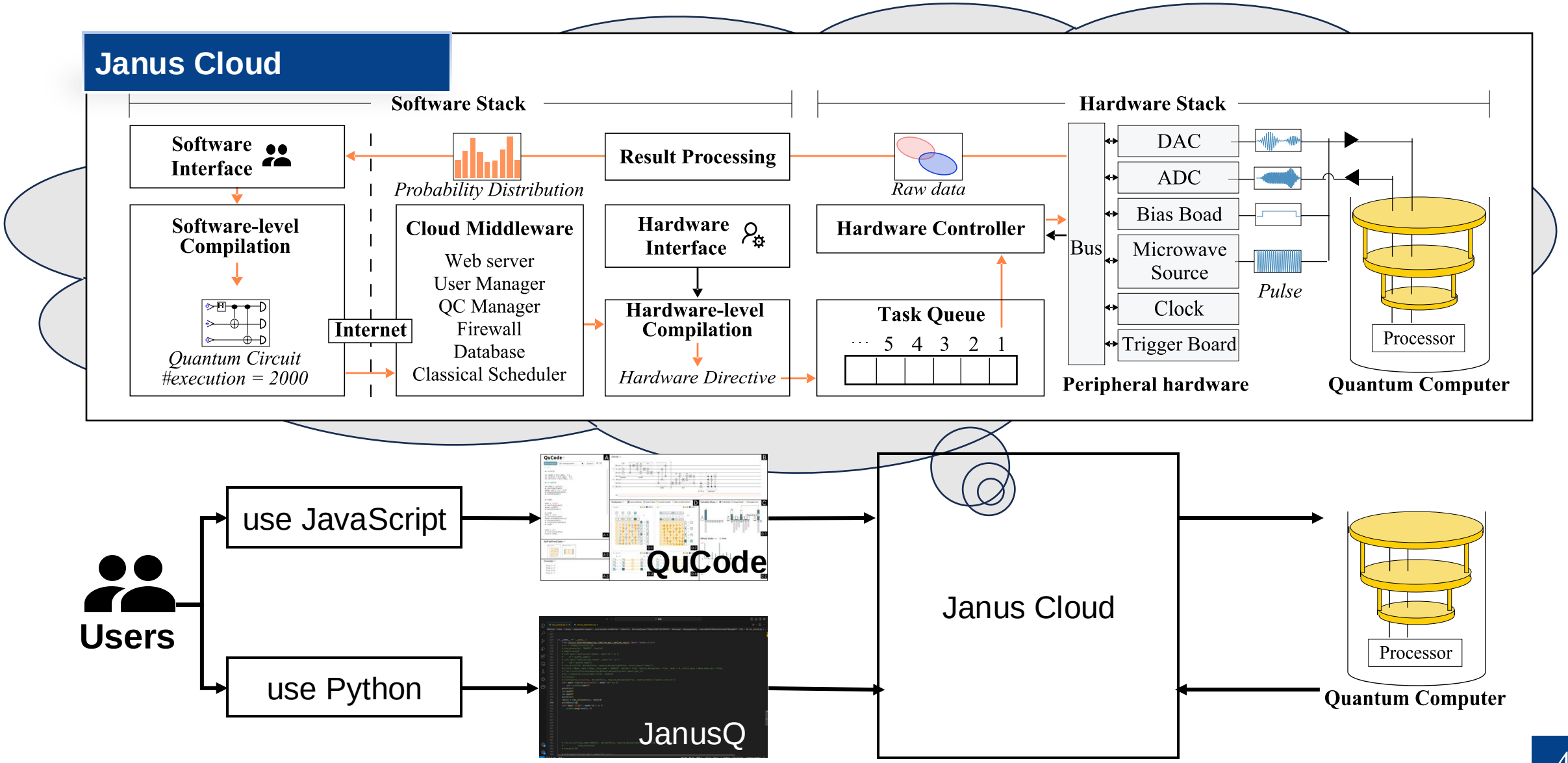
Taiyuan (太元) & Janus

Yin and Yang

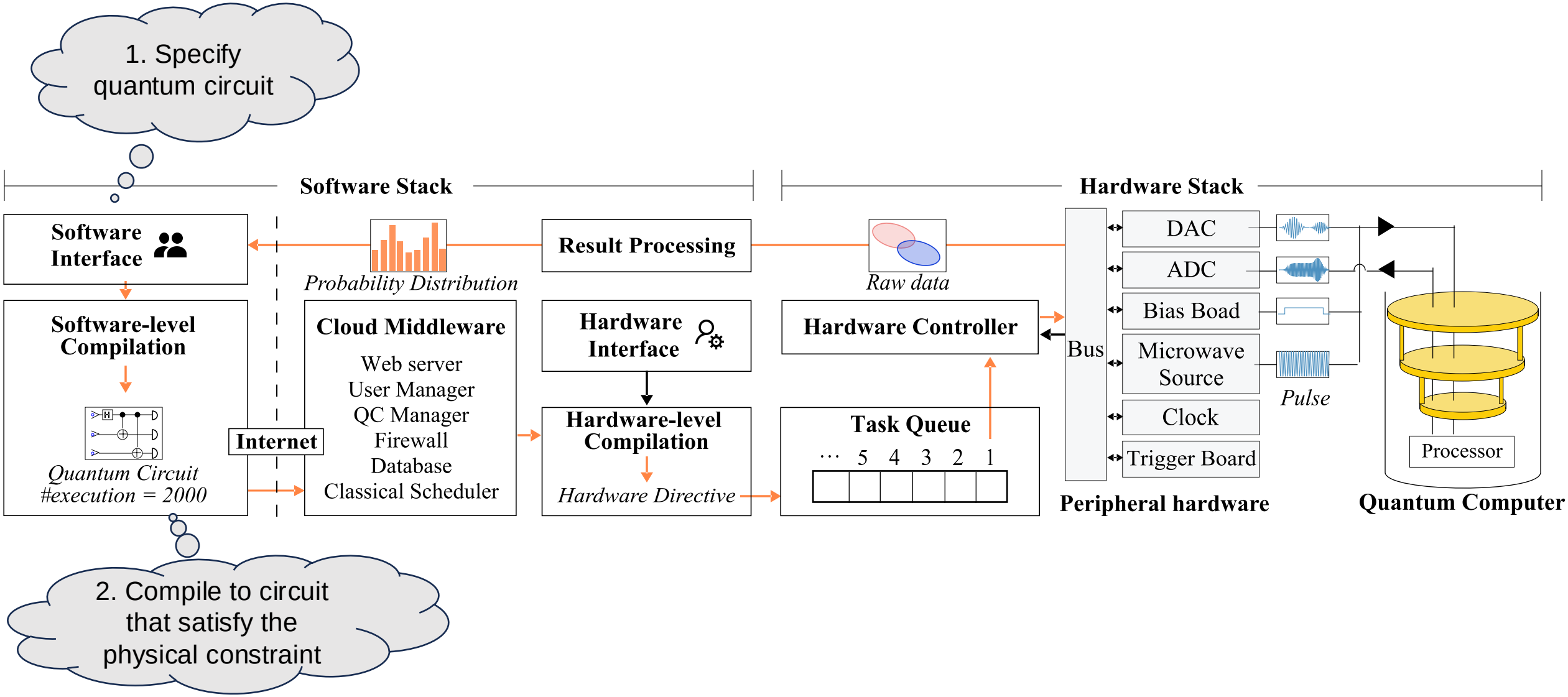
Tai Yuan is a Chinese gold who is the wife of Pan Gu (the god that created the world). She was delivered of Yin and Yang, which is entangled like a quantum superposition state

Janus is a two-faced god of the ancient Romans. He is the god of beginning, transition, time, change, dualism, and end. He has two faces simultaneously.

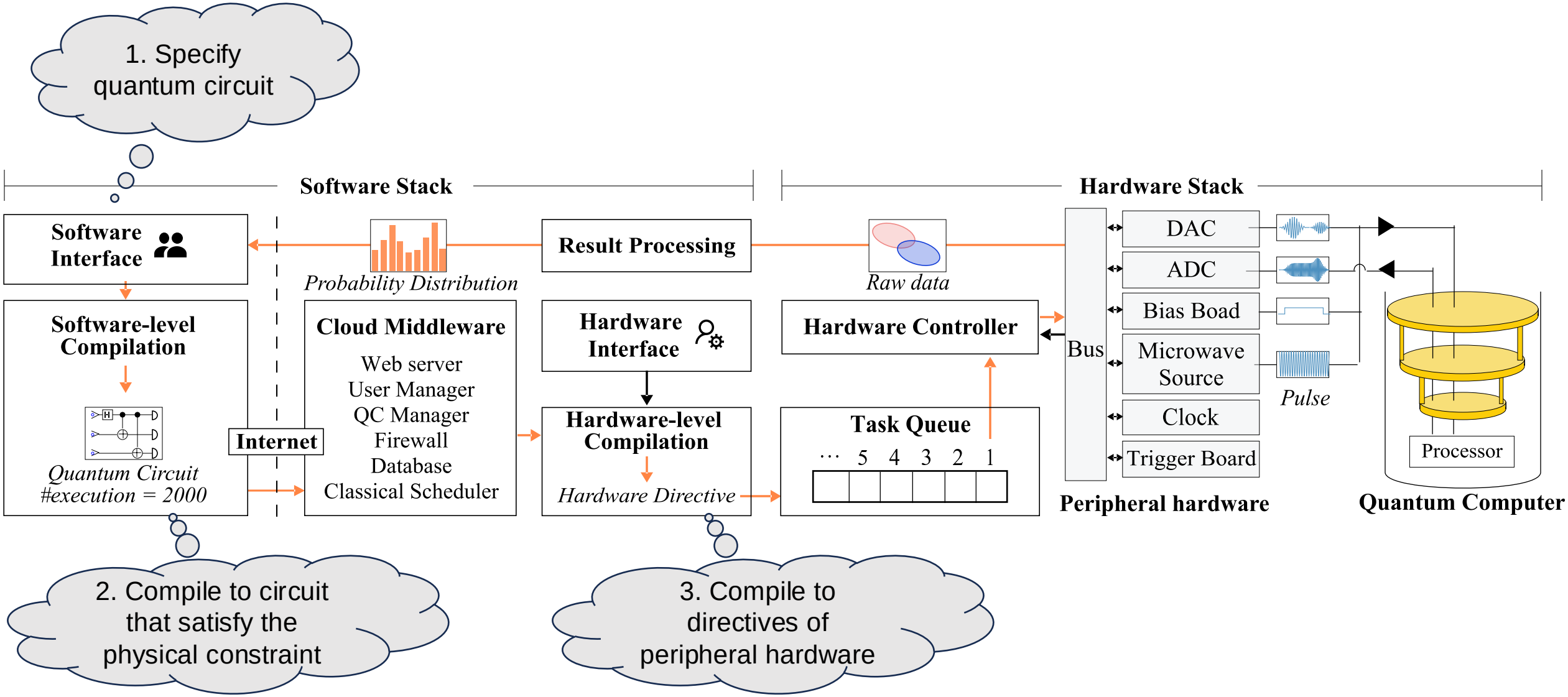
What we can do on Janus Platform



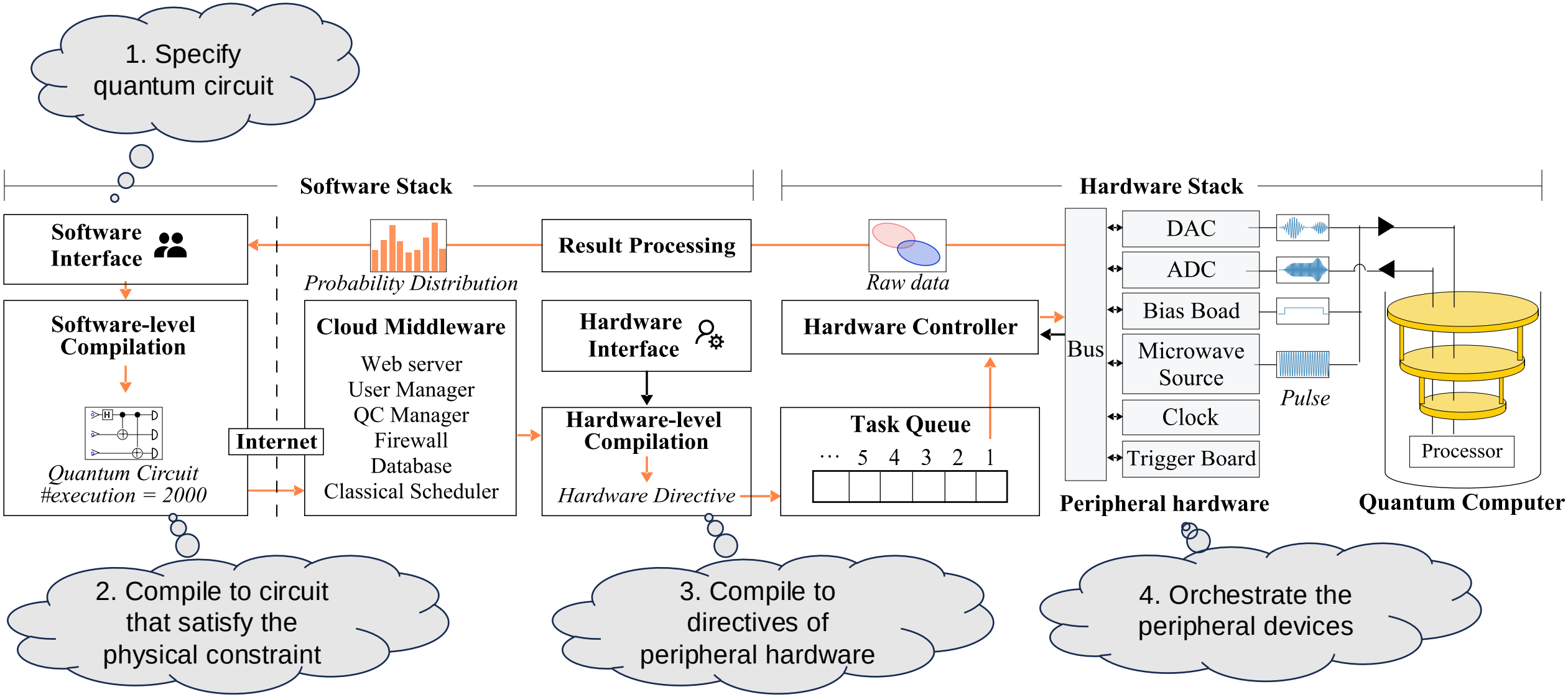
What we can do on Janus Platform



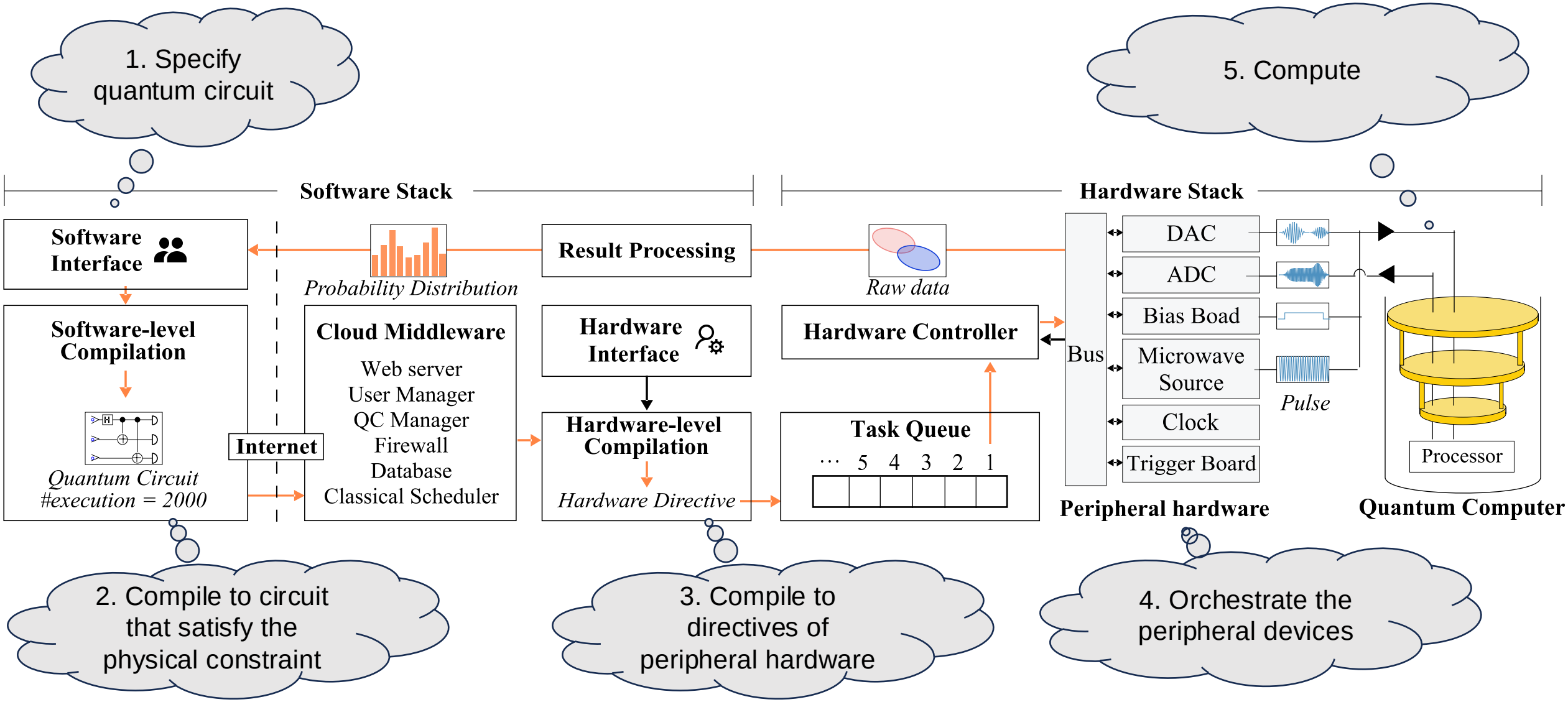
What we can do on Janus Platform



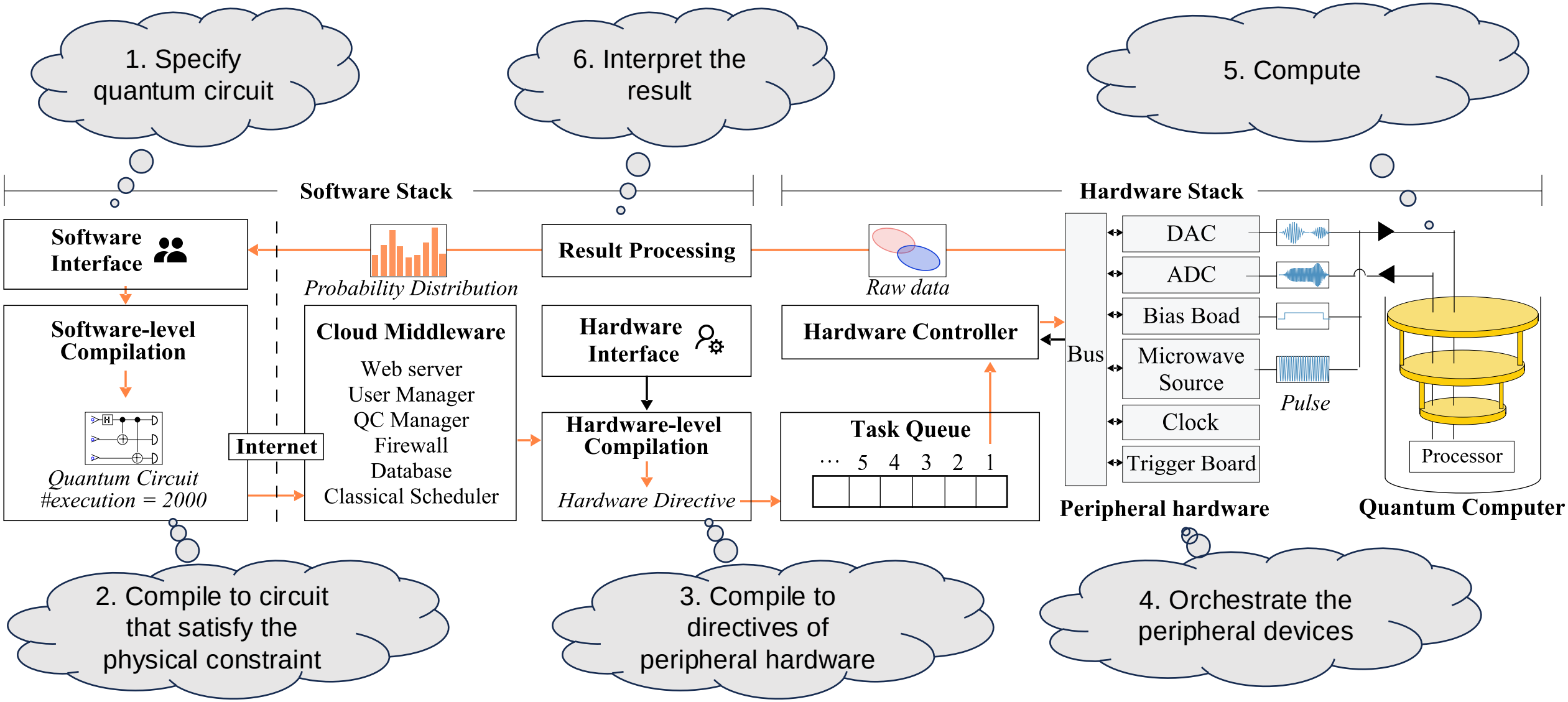
What we can do on Janus Platform



What we can do on Janus Platform



What we can do on Janus Platform

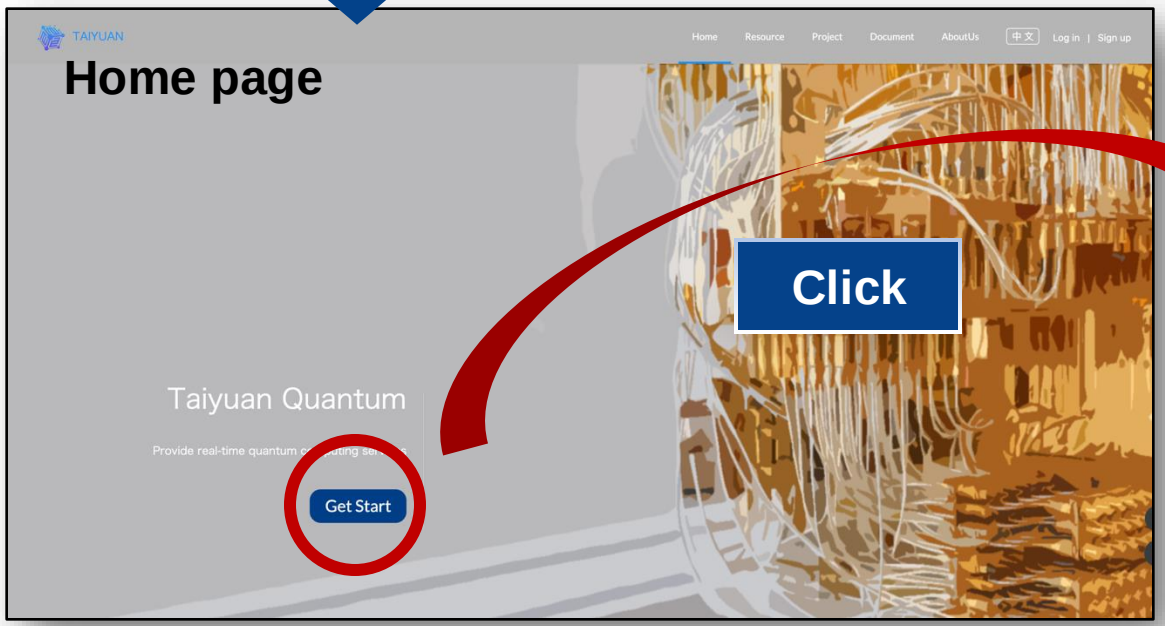


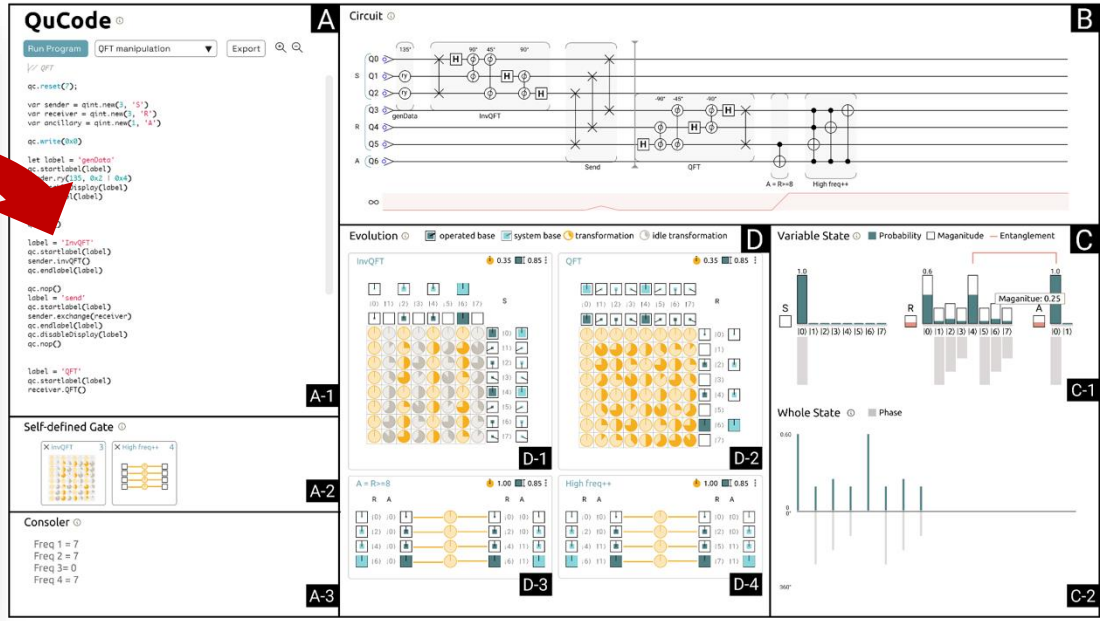
Running Program On the Website

Submitting on website

Programming on the Code editor

Open http://janusq.zju.edu.cn/home or http://193.112.219.143:10211/home

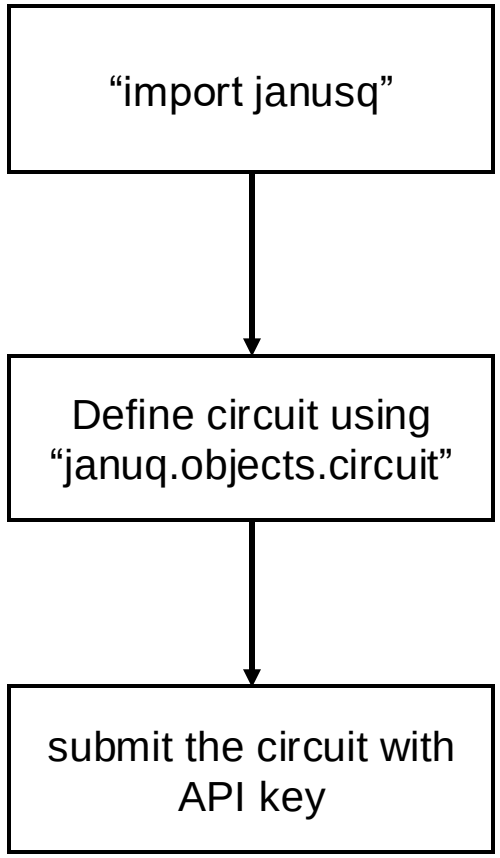




- Click the “Get Start” button to experience JanusQ.

Running Program By Python API

Submitting by Python



Import package

```
import matplotlib.pyplot as plt
from janusq.cloud_interface import submit, get_result
from janusq.objects.circuit import Circuit
```

Define a circuit

```
qc = Circuit([], n_qubits = 4)
qc.h(0, 0)
qc.cx(0, 1, 1)
qc.cx(1, 2, 2)
qc.cx(2, 3, 3)
print(qc)
```

Submit circuit

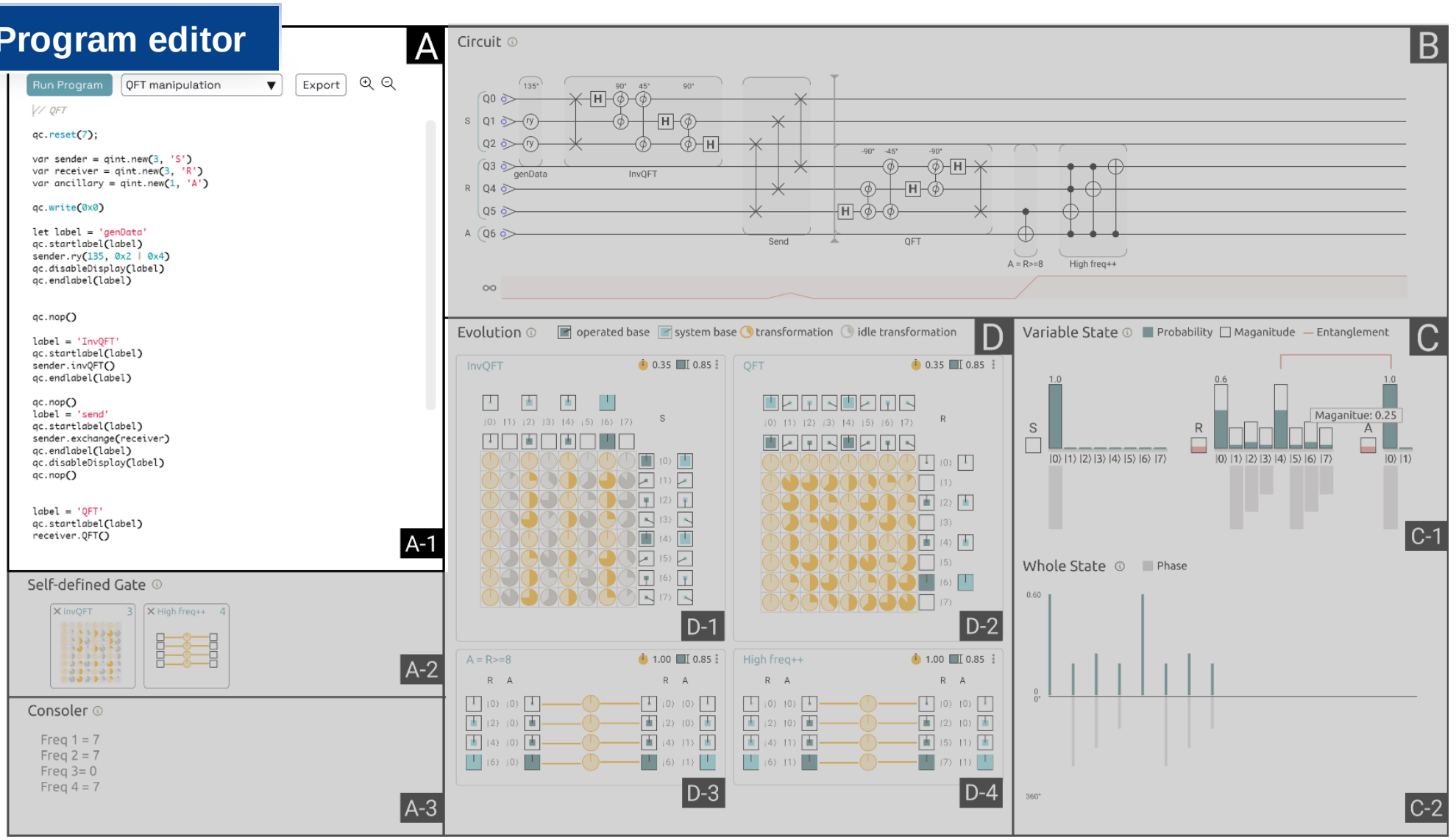
```
result = submit(circuit=qc, label= 'GHZ', shots= 3000,
run_type='simulator', API_TOKEN='')
```

Get result

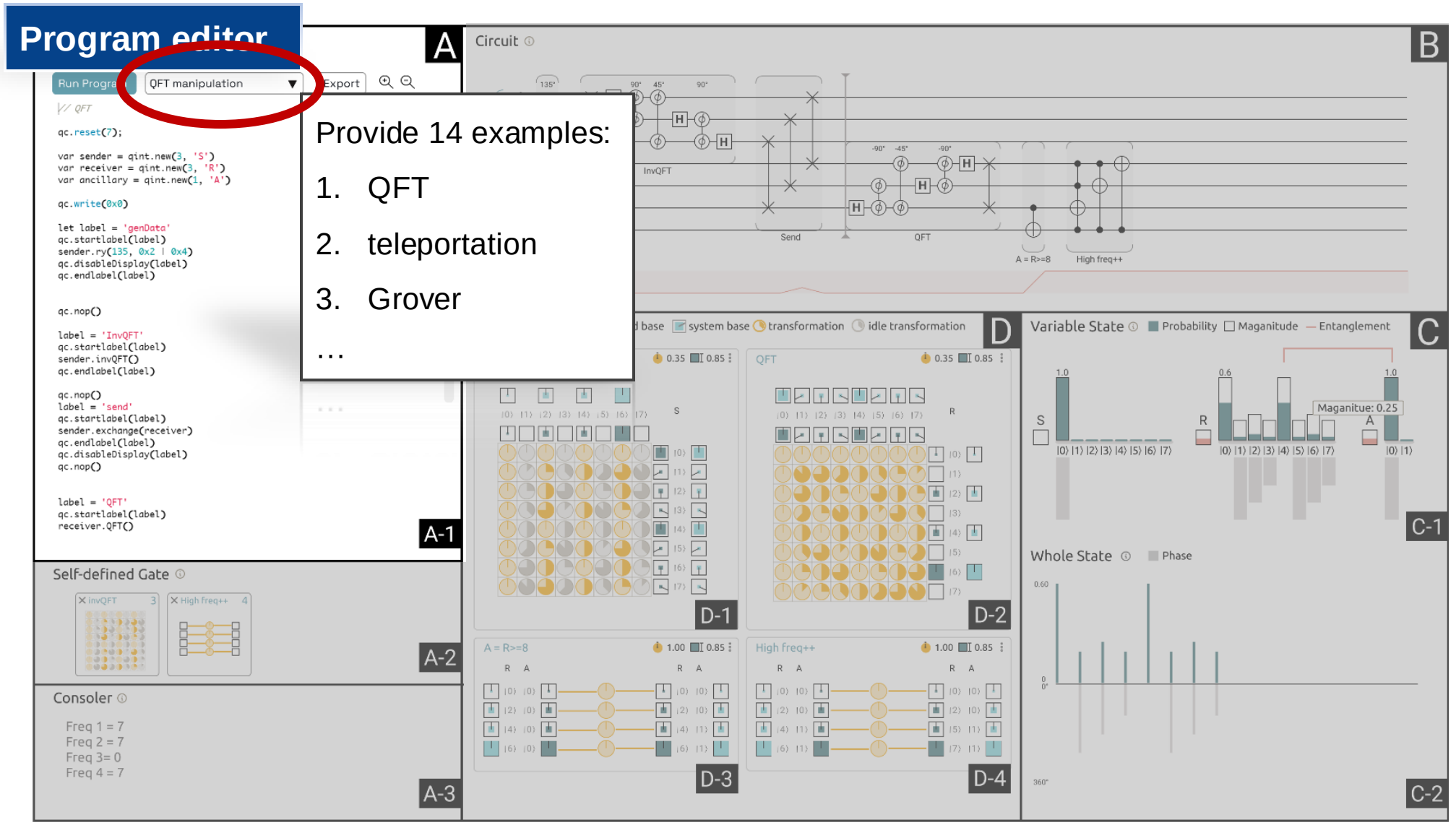
```
result = get_result(result['data']['result_id'],
run_type='simulator', result_format='probs')
print(result)
```

QuCode: Online Quantum Program Editor

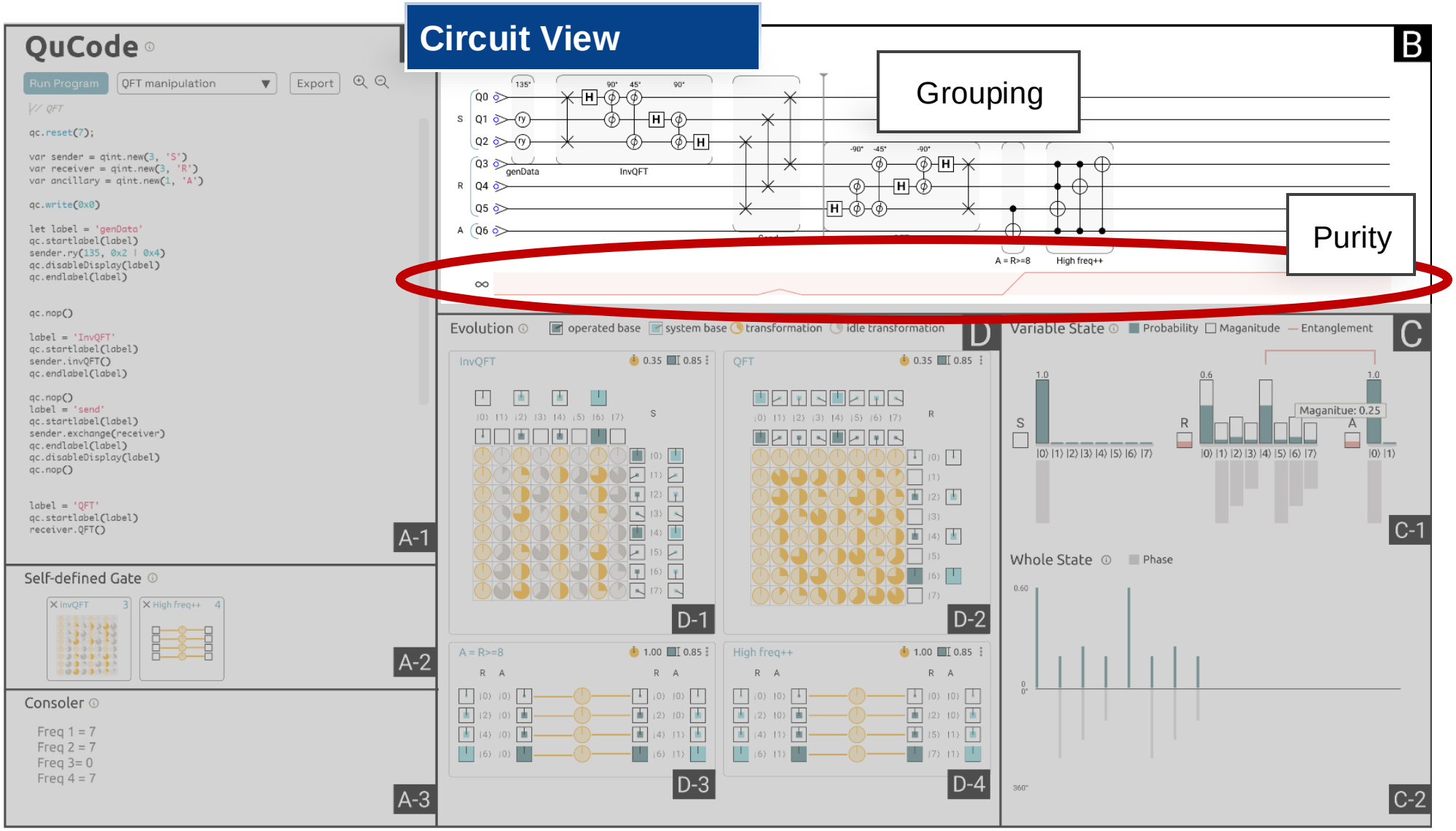
QuCode: Online Quantum Program Editor



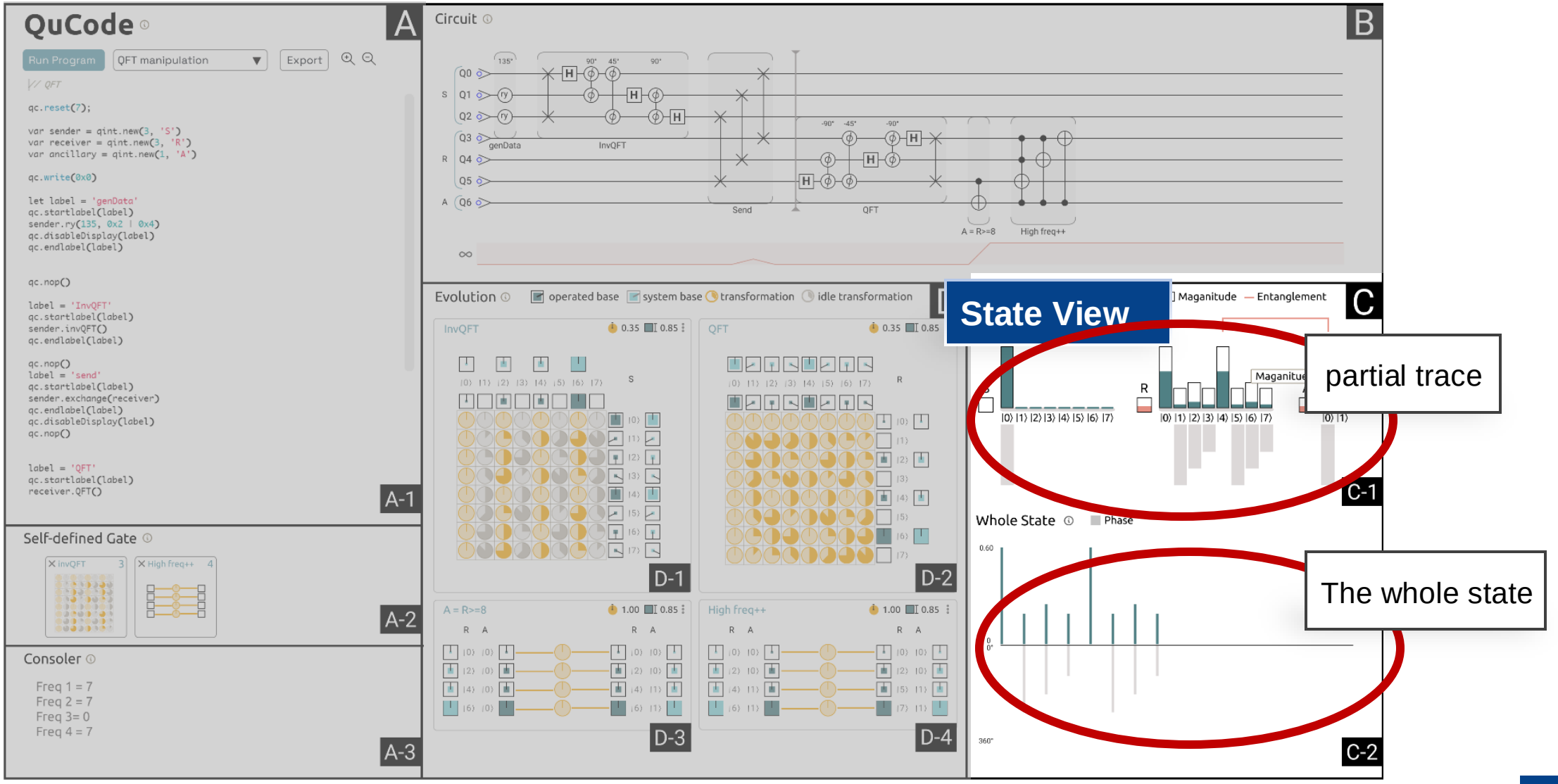
QuCode: Online Quantum Program Editor



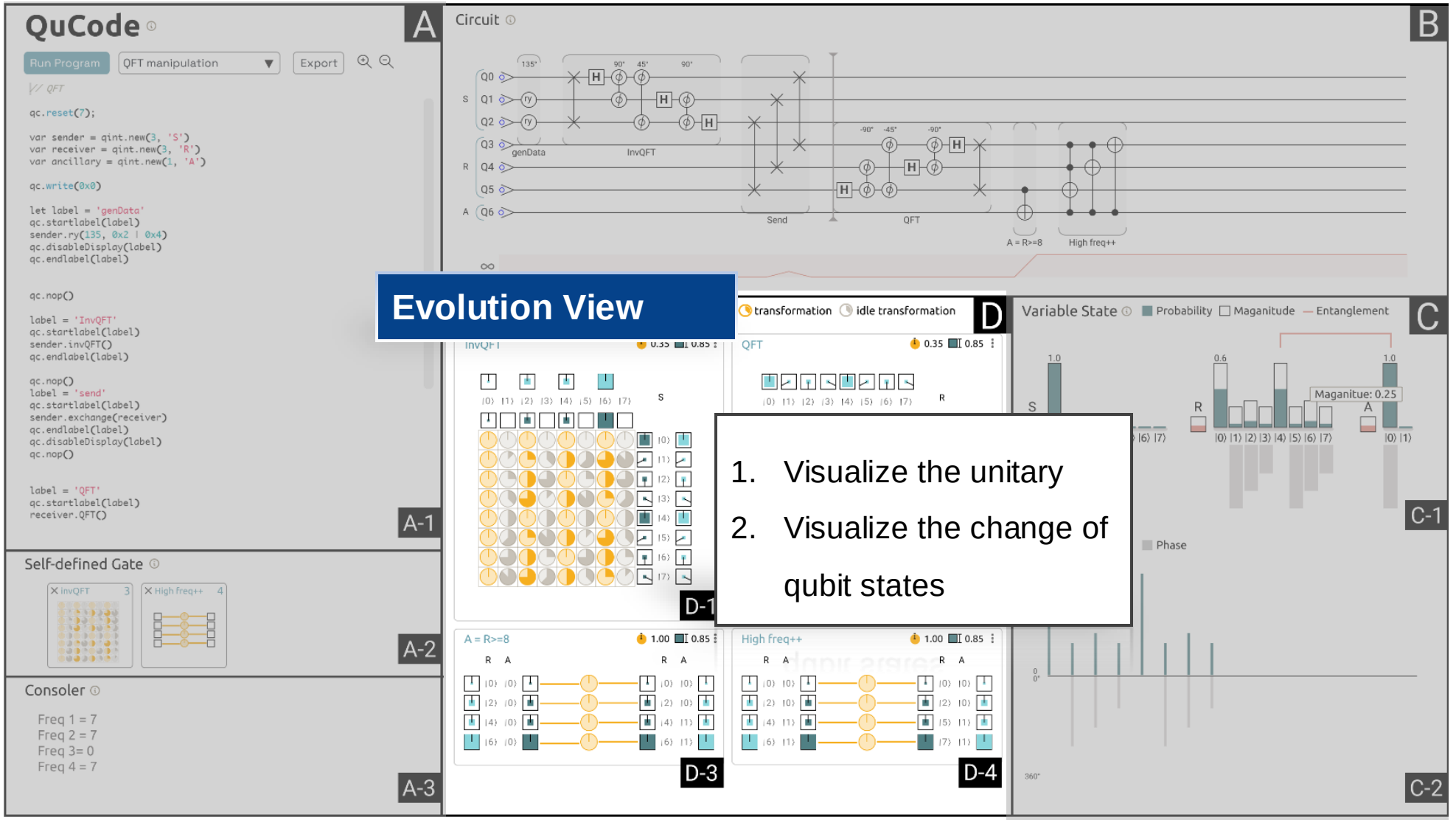
QuCode: Online Quantum Program Editor



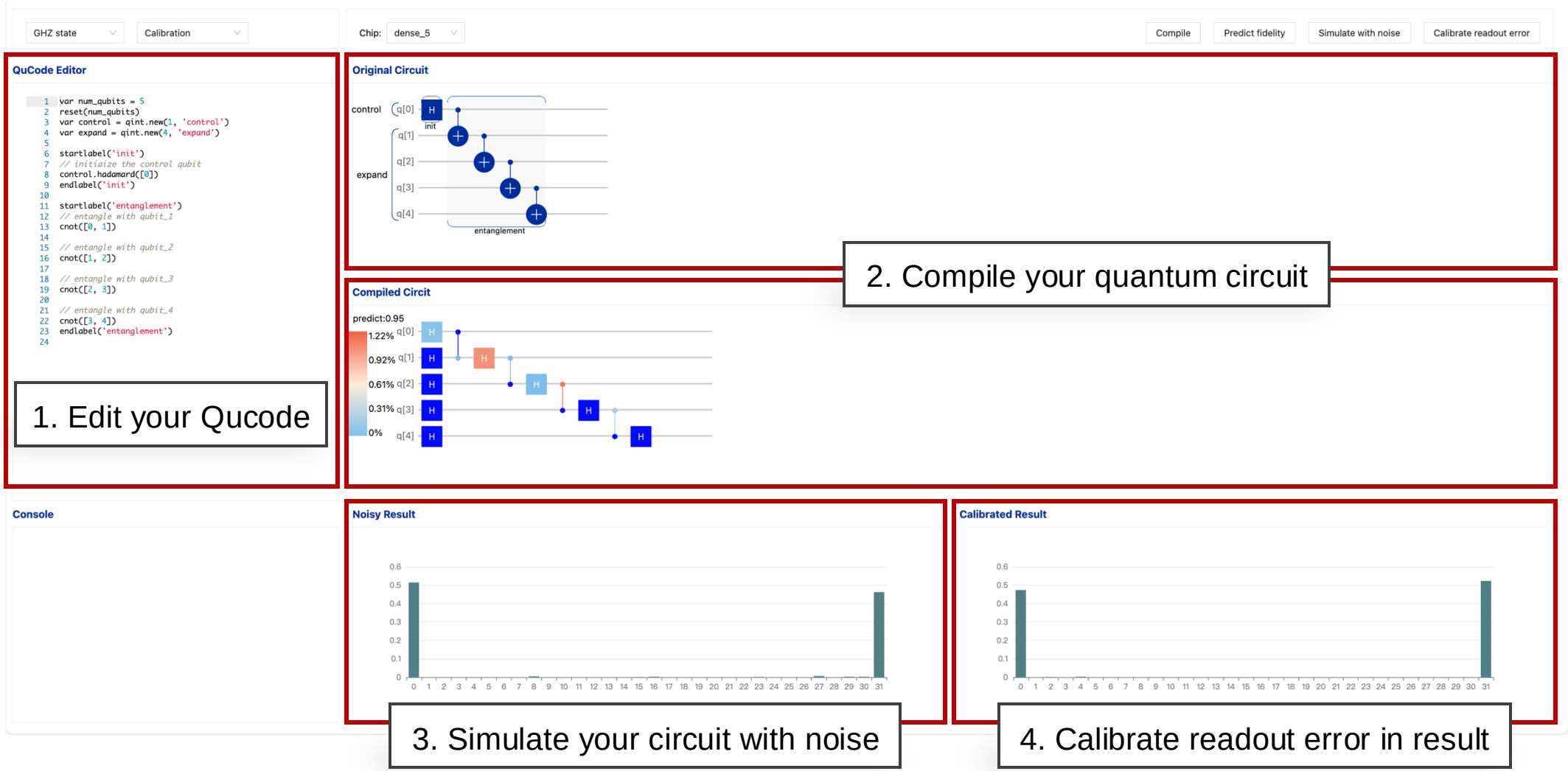
QuCode: Online Quantum Program Editor



QuCode: Online Quantum Program Editor



Simulate your program with JanusQ



Outline of Presentation

- Tutorial Overview
- Background Knowledge
- Mathematical Model of Quantum Computing
- Janus (Taiyuan) Quantum Cloud Platform
- **Installing JanusQ**

Installing JanusQ

https://github.com/JanusQ/JanusQ

Installing Methods

Language: Python 3, C++

Available platforms: Linux, Mac, Windows

HyQSAT has not been tested on Windows.

Hardware requirements:

- Classical computer: ≥ 16 GB Memory, Intel i5 10 Gen

Hardware used in the experiment

- A server with two AMD EPYC 2.25GHz 64-core CPUs and 1.6TB DDR 5 memory is preferable.
- Superconducting, ion-trapped quantum computer and D-Wave quantum annealer.

Data collected from the quantum hardware are put into the framework.

Installation:

From Docker (recommend)

From Source code

Wheel is provided in https://github.com/JanusQ/JanusQ/tree/main/dist. But HyQSAT is disabled.

Install from Docker (Preferred)

Step 1. download docker-desktop

https://www.docker.com/products/docker-desktop/

Windows Subsystem for Linux (WSL) are required for windows.

Step 2. pull JanusQ image

```
docker pull janusq/janusq3:latest
```

Step 3. start image

```
docker run -itd -p 8888:22 -p 9999:23 --name tutorial janusq/janusq3
```

Step 4. Use SSH/VSCode/PyCharm to connect the docker

```
ssh root@localhost -p 8888
```

The password of the docker is "root".

Step 5. Or use Jupyter Lab to connect the docker

```
http://localhost:9999/lab
```

Docker page of JanusQ:
https://hub.docker.com/r/janusq/janusq3

Install from Source Code

Step 1. clone the source code

```
git clone git@github.com:JanusQ/JanusQ.git
```

Step 2. install requirements (virtual environment is preferred)

```
cd ./JanusQ
conda create -n janusq python=3.10
conda activate janusq
pip install -r requirements.txt
```

Step 3. compile HyQSAT

```
cd ./janusq/application/hyqsat/solver
cmake .
make install
```

Implemented based on C++.

Step 3. install requirements for Choco-Q (Linux with CPU)

```
cd ./janusq/application/chocoq
conda env create -f environment_linux_cpu.yml
```

Change the dependency file according to your platform

Page of github:
https://github.com/JanusQ/JanusQ

Testing JanusQ

On “./JanusQ/examples/ipynb/1_1_install_janusq.ipynb”

Test QuCT

```
from janusq.analysis.vectorization import *

vec_model = RandomwalkModel()
vec_model.train()
```

Test QuFEM

```
from janusq.calibration.readout_mitigation.qufem
import EnumeratedProtocol

protocol = EnumeratedProtocol(4)
```

Test MorphQPV

```
from janusq.verification.morphqpv import
MorphQC,Config

myconfig = Config()
myconfig.solver = 'sgd'
with MorphQC(config=myconfig) as morphQC:
    morphQC.add_tracepoint(0,1)
    morphQC.assume(0,IsPure())
    …
```

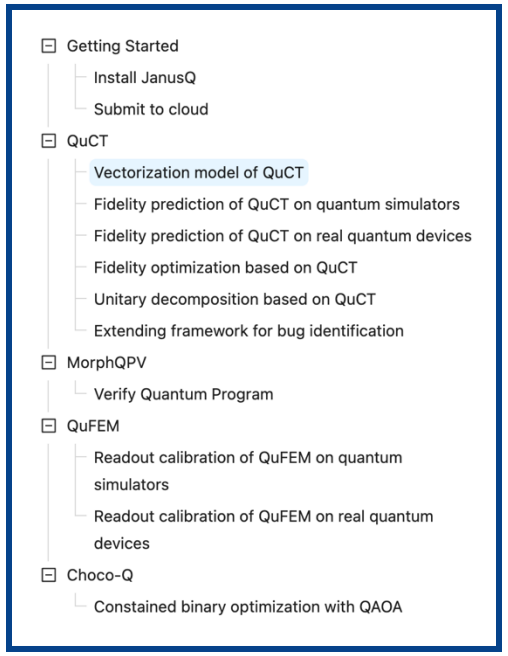
Test Choco-Q

```
from janusq.application.chocoq.chocoq.model import
LinearConstrainedBinaryOptimization as LcboModel

m = LcboModel()
x = m.addVars(5, name="x")
m.setObjective((x[0] + x[1])* x[3] + x[2], "max")
m.addConstr(x[0] + x[1] - x[2] == 0)
m.addConstr(x[2] + x[3] - x[4] == 1)
optimize = m.optimize()
```

Document and Example

On: **JanusQ/examples** or **https://janusq.github.io/HPCA_2025_Tutorial/demo**



Include 12 examples

Vectorization Model of QuCT

Author: Siwei Tan

Date: 7/4/2024

Based on paper "QuCT: A Framework for Analyzing Quantum Circuit by Extracting Contextual and Topological Features" (MICRO 2023)

In the current Noisy Intermediate-Scale Quantum era, quantum circuit analysis is an essential technique for designing high-performance quantum programs. Current analysis methods exhibit either accuracy limitations or high computational complexity for obtaining precise results. To reduce this tradeoff, we propose QuCT, a unified framework for extracting, analyzing, and optimizing quantum circuits. The main innovation of QuCT is to vectorize each gate with each element, quantitatively describing the degree of the interaction with neighboring gates. Extending from the vectorization model, we can develope multiple downstream models for fidelity prediction and unitary decomposition, etc. In this tutorial, we introduce the APIs of the vectorization model in QuCT.

In [1]:

```
%matplotlib inline

import os
os.chdir("../..")
import logging
logging.basicConfig(level=logging.WARN)

import random
import numpy as np

from janusq.analysis.vectorization import RandomwalkModel, extract_device
from janusq.objects.random_circuit import random_circuits, random_circuit
from janusq.objects.backend import GridBackend
```

Vectorization Flow

Below is the workflow to vectorize a gate in the quantum circuit. The gate is vectorized by two steps. The first step runs random walks to extract circuit features in the neighbor of the gates. the second step use a table comparison to generate the gate vector.

浙江大學
ZHEJIANG UNIVERSITY

Thanks for listening!