



# 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/](https://janusq.github.io/HPCA_2025_Tutorial/)



Liqiang Lu

[liqianglu@zju.edu.cn](mailto: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 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



## JanusQ Software 2.0

- Cluster I/O
- Quantum compilation



## Achievements published in top conferences

- MICRO & ASPLOS
- ICCAD quantum chemistry competition

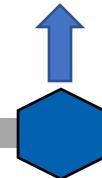


## HPCA & Janus 3.0 tutorial

- Constrained binary optimization QAOA
- Second time of Janus tutorial



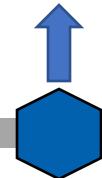
HPCA 2025



2023/02



2022/07



2023/10



2024/04



2025/03



2025/06



## Hybrid quantum-classical SAT solver

HPCA 2023

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



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

# JanusQ Quantum Cloud



浙江大學  
ZHEJIANG UNIVERSITY

## Coding interface of JanusQ

**A**

**QuCode**

Run Program QFT manipulation Export

**Program editor**

```
✓ QFT
var sender = qint.new('S')
var receiver = qint.new('R')
var ancillary = qint.new('A')

qc.write(0x0)

let label = 'genData'
qc.startLabel(label)
sender.ry(135, 0x2 | 0x4)
qc.disableDisplay(label)
qc.endLabel(label)

qc.nop()

label = 'InvQFT'
qc.startLabel(label)
sender.invQFT()
qc.endLabel(label)

qc.nop()
label = 'send'
qc.startLabel(label)
sender.exchange(receiver)
qc.endLabel(label)
qc.disableDisplay(label)
qc.nop()

label = 'QFT'
qc.startLabel(label)
receiver.QFT()
```

**A-1**

**Circuit reuse**

**A-2**

**Self-defined Gate**

**A-3**

**Console**

Freq 1 = 7  
Freq 2 = 7  
Freq 3 = 0  
Freq 4 = 7

**B**

**Quantum circuit view**

**C**

**Evolution view**

**D**

**Variable State**

Probability Magnitude Entanglement

**C-1**

**C-2**

**D-1**

**D-2**

**D-3**

**D-4**

**State view**

This section displays the JanusQ Quantum Cloud interface. It includes a QuCode editor with quantum circuit code, a Quantum circuit view showing a complex multi-qubit circuit with various gates like H, S, and CNOT, an Evolution view showing state transitions over time, a Variable State view displaying probability and magnitude distributions for qubits S, R, and A, and a State view showing the overall state of the system.

## Execution modes

**switching mode**

Choose mode

Quantum cluster  Quantum computer  python simulator  
 JavaScript simulator  analysis

cancel confirm

This modal dialog allows users to switch between different execution modes. It lists four options: Quantum cluster, Quantum computer, python simulator, and JavaScript simulator. The 'JavaScript simulator' option is currently selected. Buttons for 'cancel' and 'confirm' are at the bottom.

## Provided quantum processors

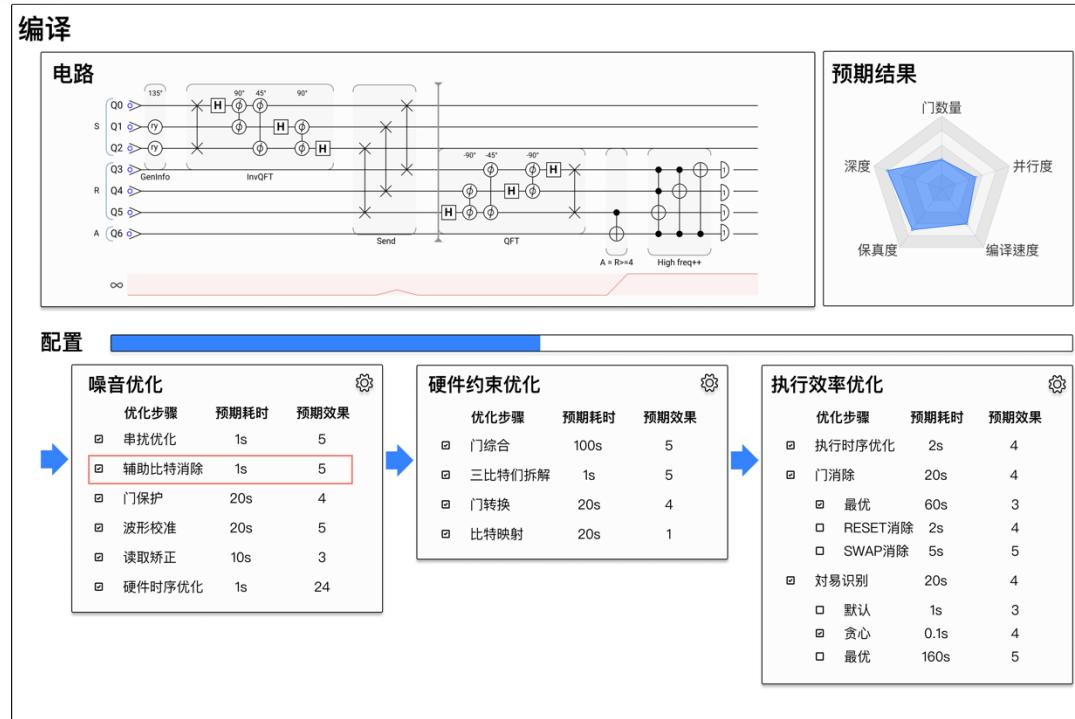
**Resources**

计算机名字 N36U19\_0  
ONLINE STATUS 10 QUBITS

计算机名字 N36U19\_1  
ONLINE STATUS 13 QUBITS

This section shows a list of provided quantum processors. It displays two entries: 'N36U19\_0' with 10 qubits and 'N36U19\_1' with 13 qubits, both marked as 'ONLINE'. Each entry includes a small grid icon representing the processor layout.

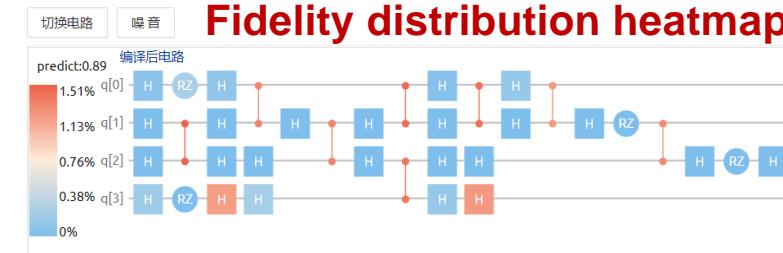
## Quantum computing compilation optimization



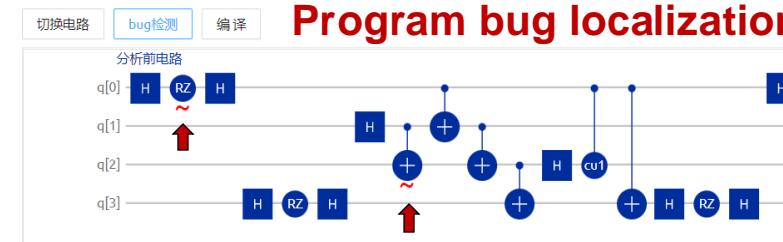
## Support selection of compilation optimization options

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

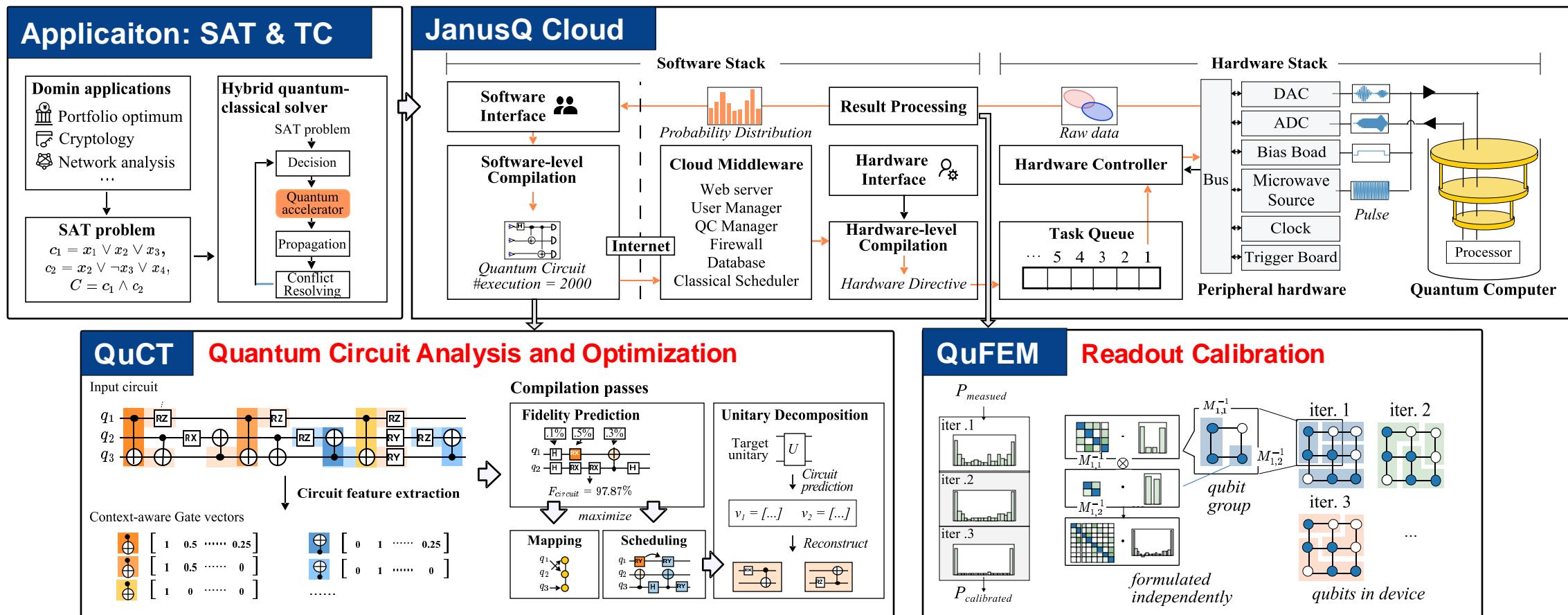
## Fidelity prediction and optimization



## Program verification and correction

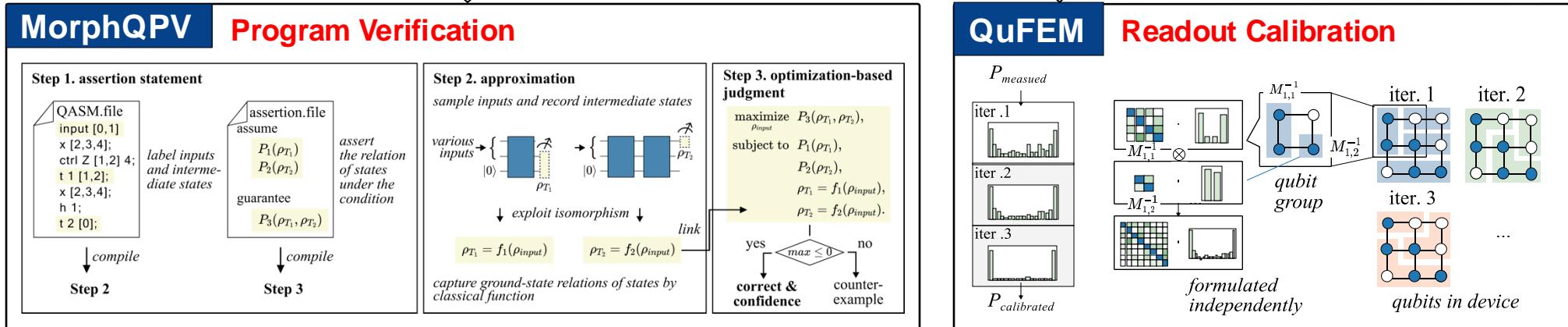
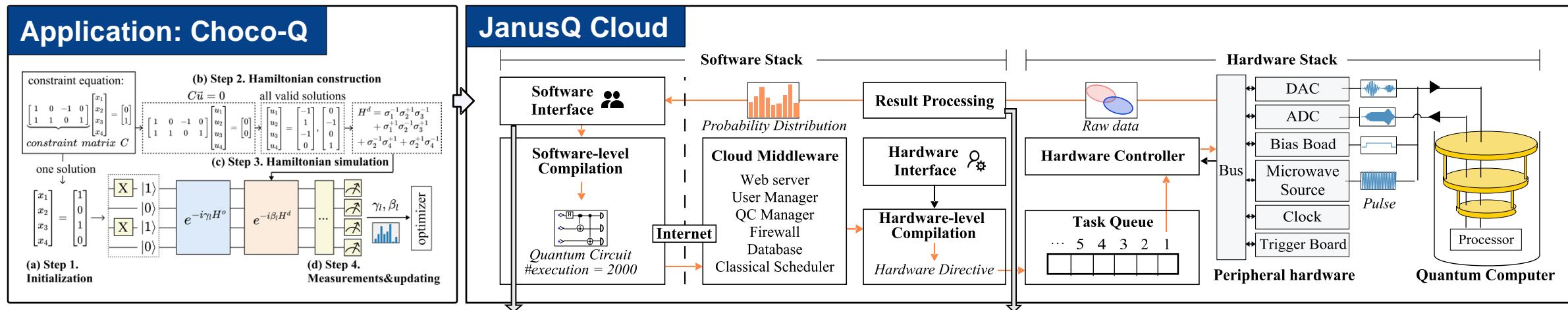


## Janus 2.0: A Software Framework for Analyzing, Optimizing and Implementing Quantum Circuit





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



# Organizers



浙江大學  
ZHEJIANG UNIVERSITY



Jianwei Yin  
Professor

[zjuyjw@cs.zju.edu.cn](mailto:zjuyjw@cs.zju.edu.cn)



Liqiang Lu  
Assistant professor

[liqianglu@zju.edu.cn](mailto:liqianglu@zju.edu.cn)



Siwei Tan  
Assistant professor

[siweitan@zju.edu.cn](mailto:siweitan@zju.edu.cn)



Tianyao Chu  
Ph.D. Student

[tianyao\\_chu@zju.edu.cn](mailto:tianyao_chu@zju.edu.cn)

# Outline

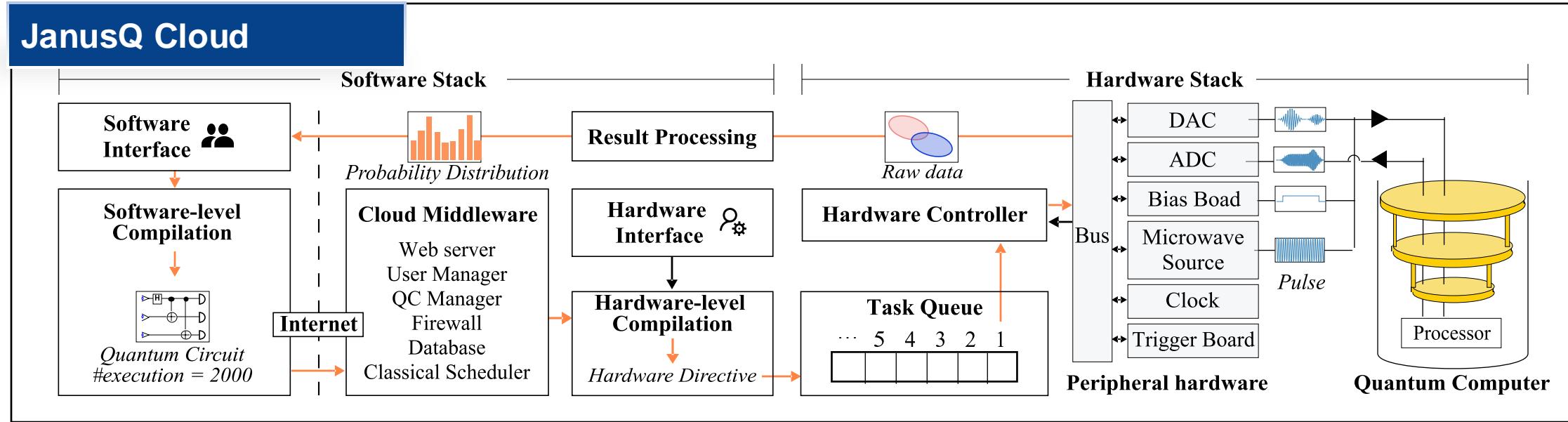


Topic	Presenter	Time
Topic-1. Introduction of Janus quantum cloud platform	Jianwei Yin & Liqiang Lu	0.5 hour
Topic-2. QuCT Details (a) Vectorization model and code examples (b) Fidelity optimization & Unitary decomposition using gate vectors (c) Extending the framework by yourself: other downstream tasks!	Tianyao Chu	(1.0 hour) 0.25 hour 0.5 hour 0.25 hour
Topic-3. MorphQPV Details (a) MorphQPV overview (b) Assertion statement (c) Implementation details of the automatic verification and repair	Siwei Tan	(1.25 hour) 0.5 hour 0.25 hour 0.5 hour

# Outline



Topic	Presenter	Time
Topic-4. QuFEM Details	Kaiwen Zhou	(0.5 hour)
(a) Characterization of readout error		0.25 hour
(b) Readout calibration using QuFEM		0.25 hour
Topic-5. Choco-Q Details	Liqiang Lu	(0.5 hour)
(a) Introduction of constained binary optimization problem		0.12 hour
(b) Choco-Q overview		0.25 hour
(c) Solve real-world problems with Choco-Q		0.12 hour
Topic-6. Q & A	All	0.25 hour
Total		~4 hours

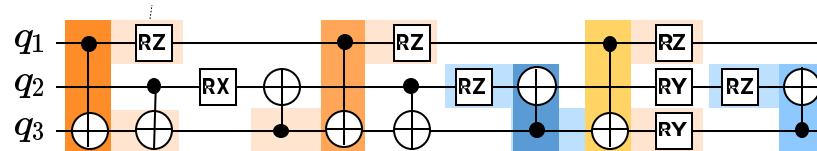


### Janus Quantum Infrastructure:

- 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 C: Contextual, T: Topological

Input circuit



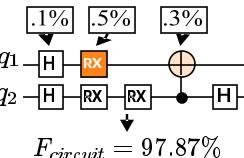
Circuit feature extraction

Context-aware Gate vectors

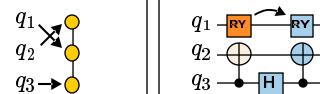
	$\begin{bmatrix} 1 & 0.5 & \dots & 0.25 \end{bmatrix}$		$\begin{bmatrix} 0 & 1 & \dots & 0.25 \end{bmatrix}$
	$\begin{bmatrix} 1 & 0.5 & \dots & 0 \end{bmatrix}$		$\begin{bmatrix} 0 & 1 & \dots & 0 \end{bmatrix}$
	$\begin{bmatrix} 1 & 0 & \dots & 0 \end{bmatrix}$	.....	

Compilation passes

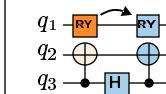
Fidelity Prediction



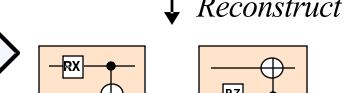
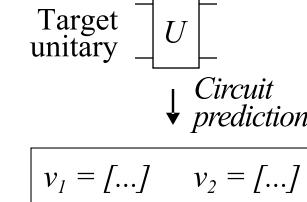
Mapping



Scheduling



Unitary Decomposition



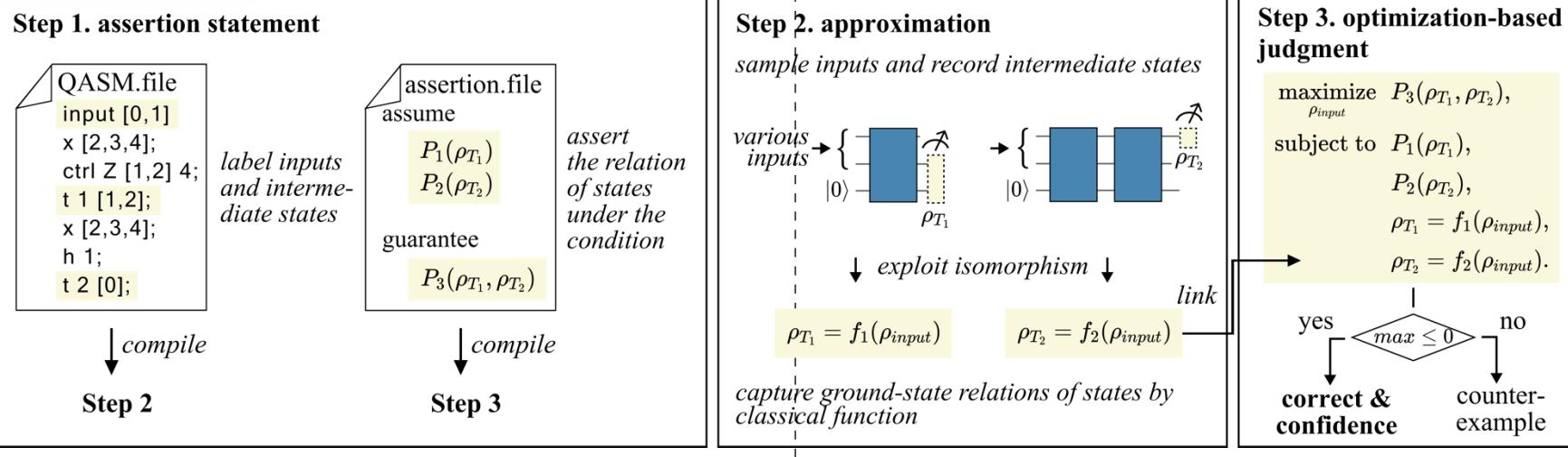
### QuCT: Topology-aware quantum circuit optimizer (MICRO 2023)

- 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

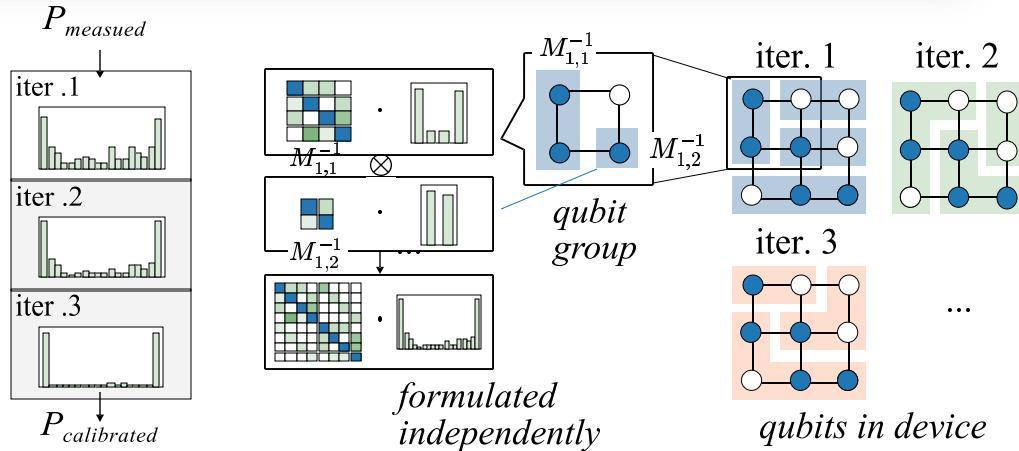


### MorphQPV: Isomorphism-based quantum program verification (ASPLOS 2024)

- What is program verification in quantum circuit analysis? Why is it difficult?
- How do we use isomorphism relationship to verify 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 FEM: Finite Element Method

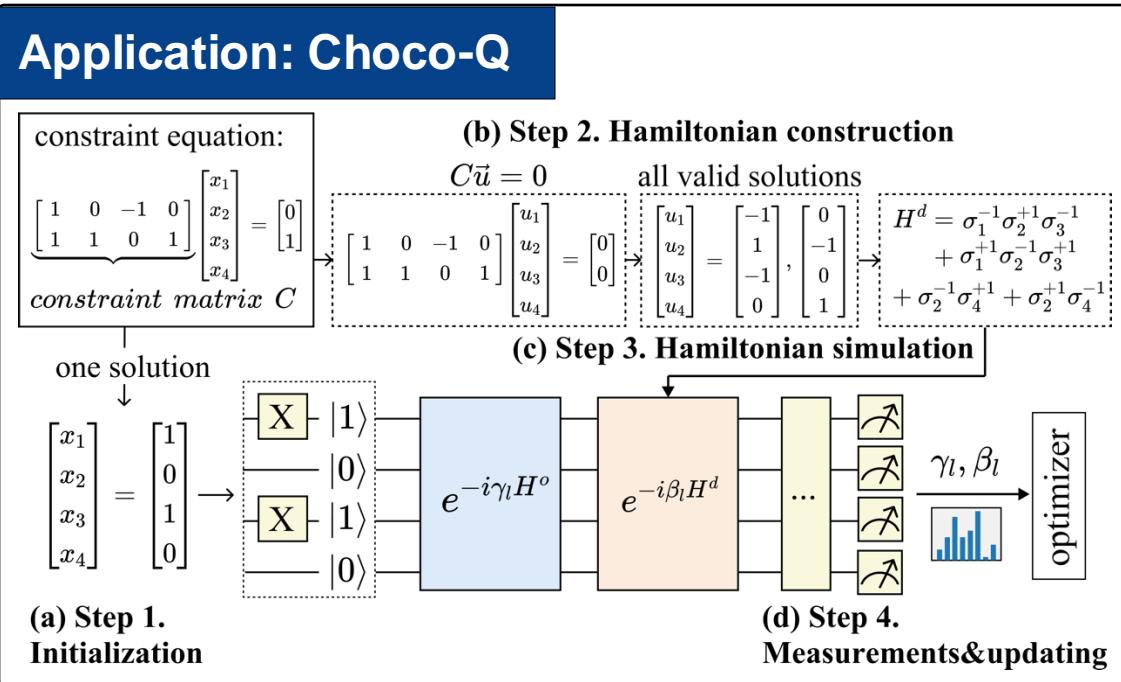


### QuFEM: Readout Calibration based on finite element method (ASPLOS 2024)

- 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]

## Application: 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

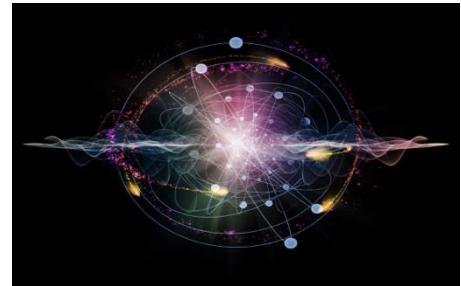


浙江大學  
ZHEJIANG UNIVERSITY

# Background Knowledge



Development of  
Classical Computing



Motivation of  
Quantum Computing



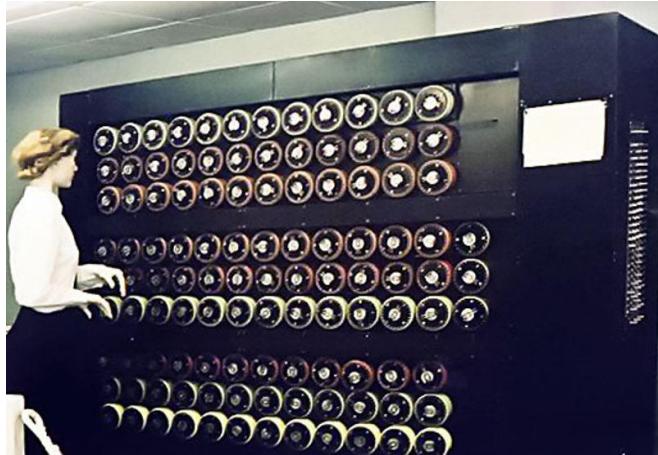
Development of  
Quantum Computing

# Development Of Classical Computing

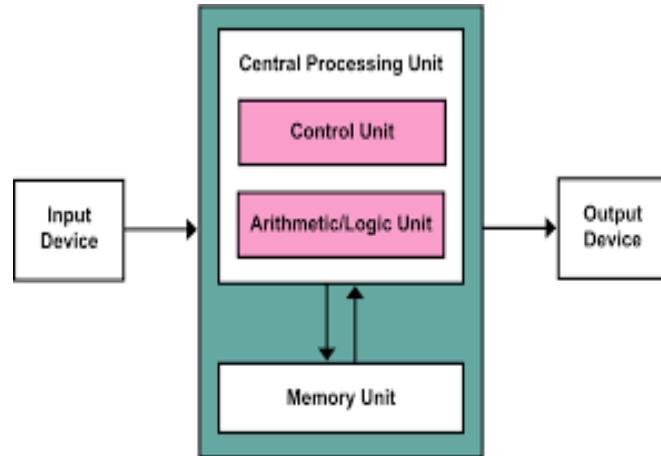


浙江大學  
ZHEJIANG UNIVERSITY

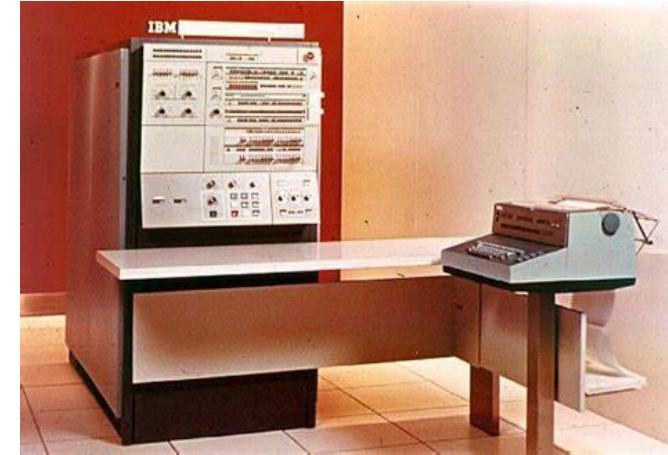
Domain-specific calculator (1939)



Von Neumann architecture (1947)



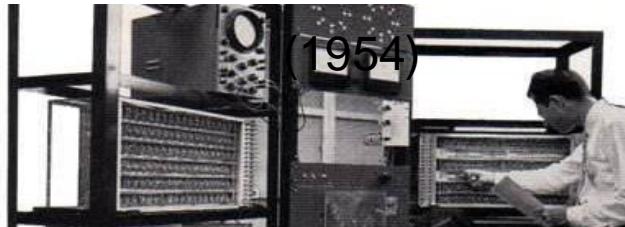
IBM360 Integrated Circuit (1964)



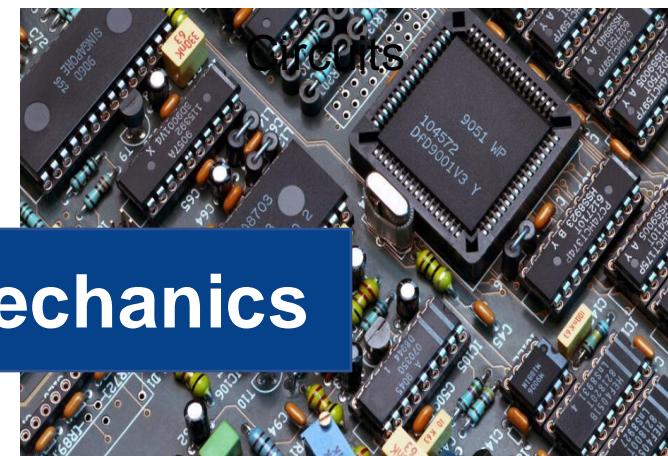
Vacuum Tube Computer ENIAC (1942)



Transistor Computer TRADIC



Large Scale Integrated Circuits

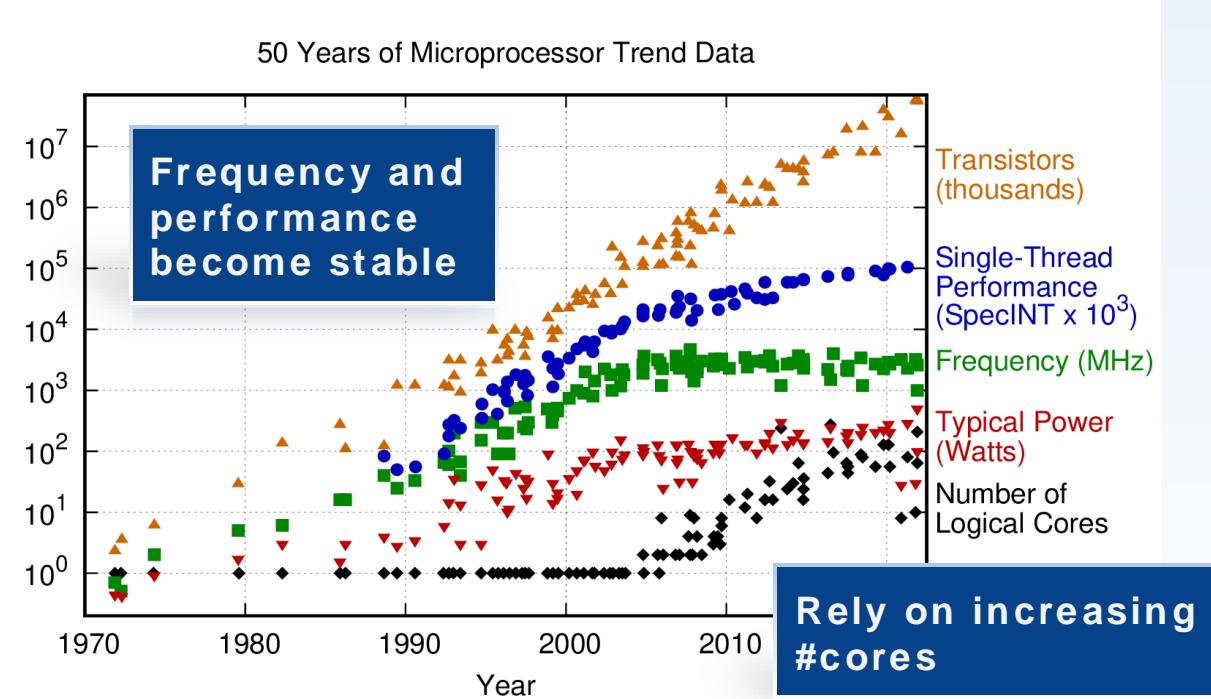


Macroscopic Effects of Quantum Mechanics

# Motivation Of Quantum Computing



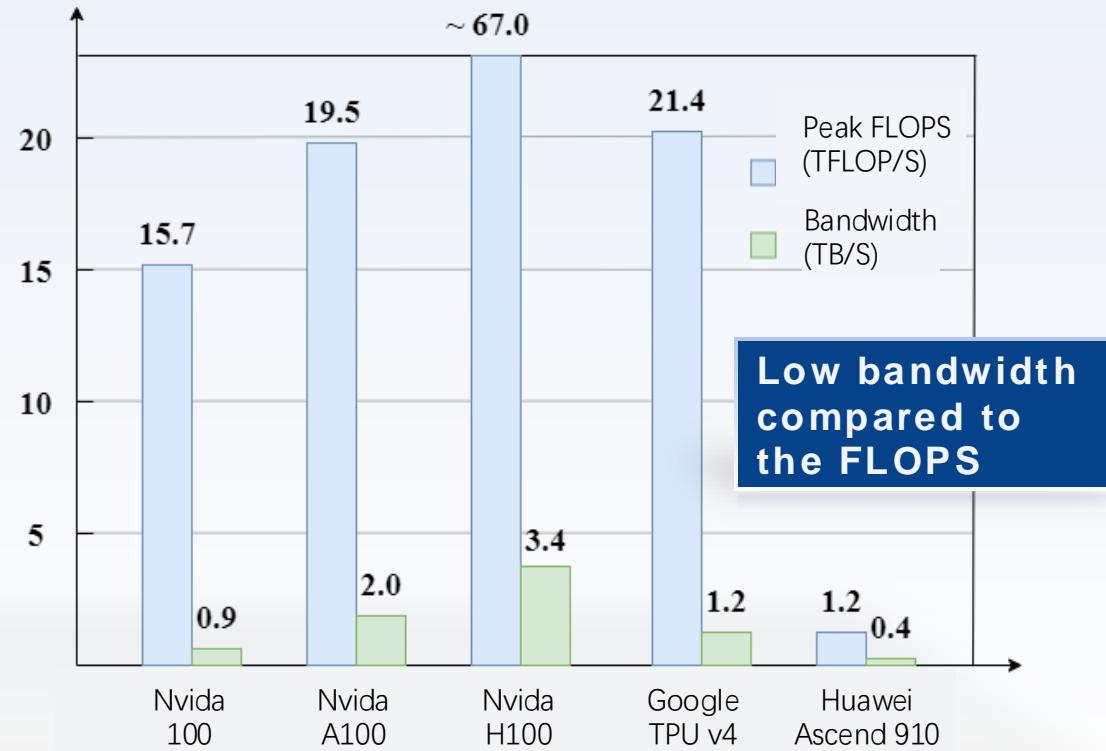
## Computation barrier



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonne, O. Shacham, K. Olukolu, L. Hammond, and C. Balen. New plot and data 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.

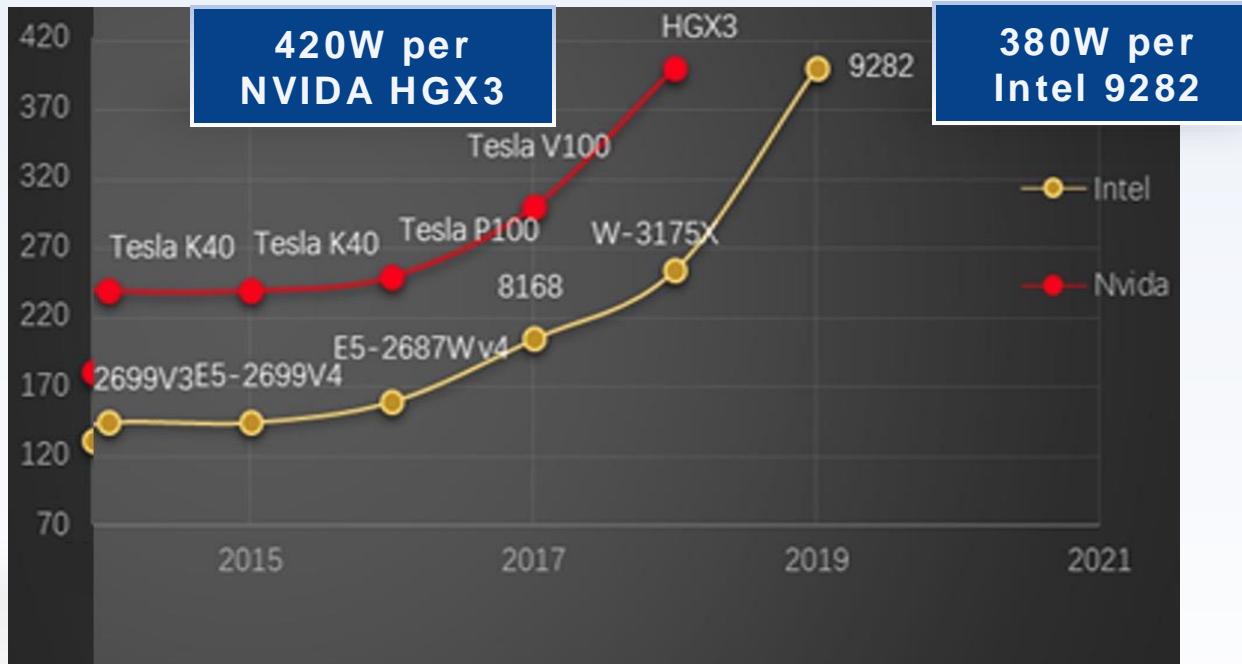
## Storage barrier



- 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.

## Power wall

Thermal design power exponentially increases.



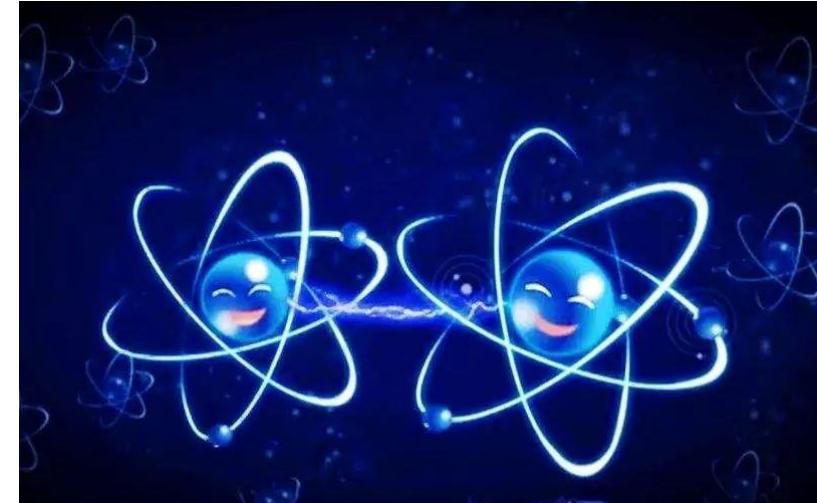
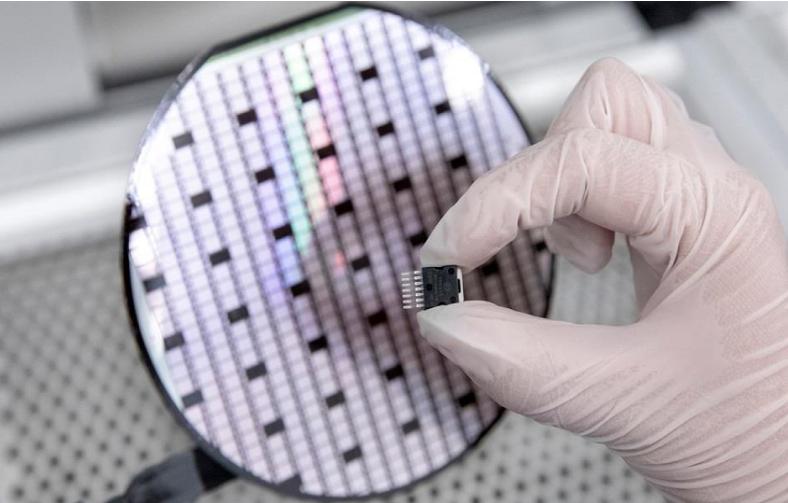
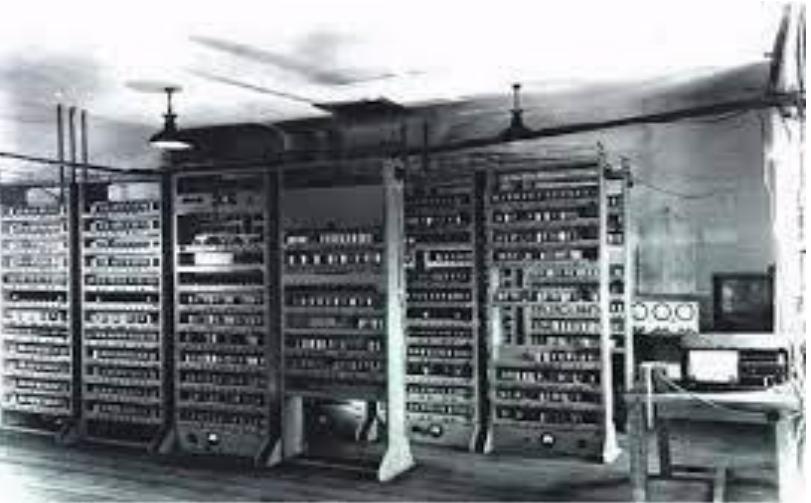
From <https://www.blueocean-china.net/faq3/241.html>

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

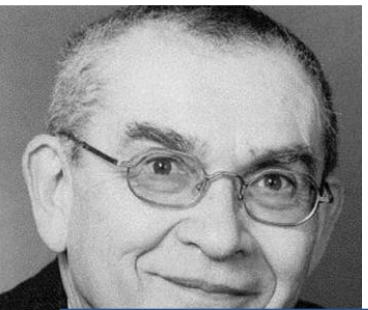
# Proposal of Quantum Computing



浙江大學  
ZHEJIANG UNIVERSITY



- 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

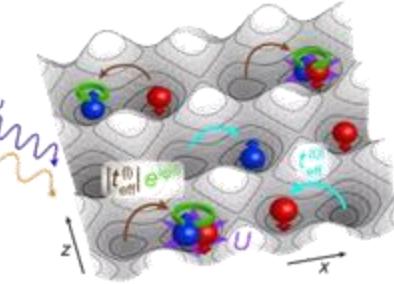


**Richard Feynman**

- 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

# Basic Concepts of Quantum Computing

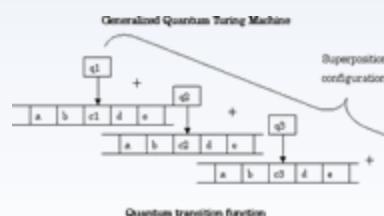
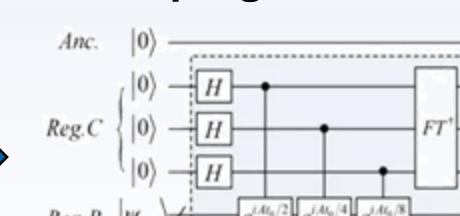
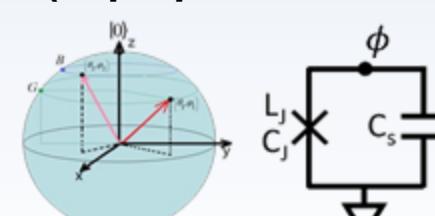


Physical  
Simulation

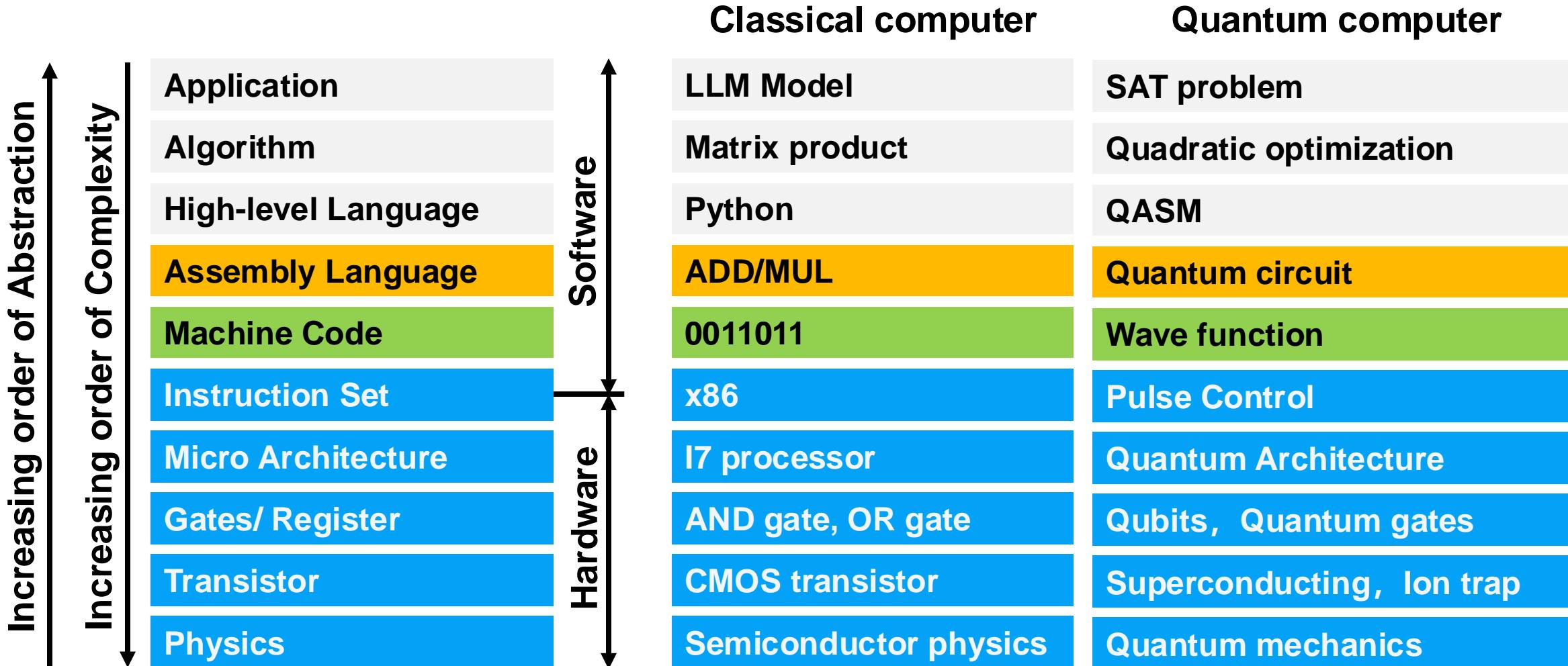


Factor

Application

		Computing theory	Encoding	Physical implementation
Classical program		Classical Turing machine	Program based on binary encoding	CMOS (0/1)
Quantum program	Quantum	Quantum Turing machine	Circuit-based quantum program	Superconducting, photon (superposition state)
	lead to high computational advantages via parallelism.	 $\Lambda : \mathfrak{S}(\mathcal{H}) \rightarrow \mathfrak{S}(\mathcal{H})$ $\mathcal{H} = \mathcal{H}_0 \otimes \mathcal{H}_E \otimes \mathcal{H}_G$	 $ 0\rangle$ $ 0\rangle$ $ 0\rangle$ $Reg.C$ $Reg.B  \psi_B\rangle$ $U_{PE}$	 Josephson junction
		suffer from complexity due to quantum collapse and entanglement		has low energy consumption due to reversibility and superconductivity.

# Quantum Computing Architecture



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

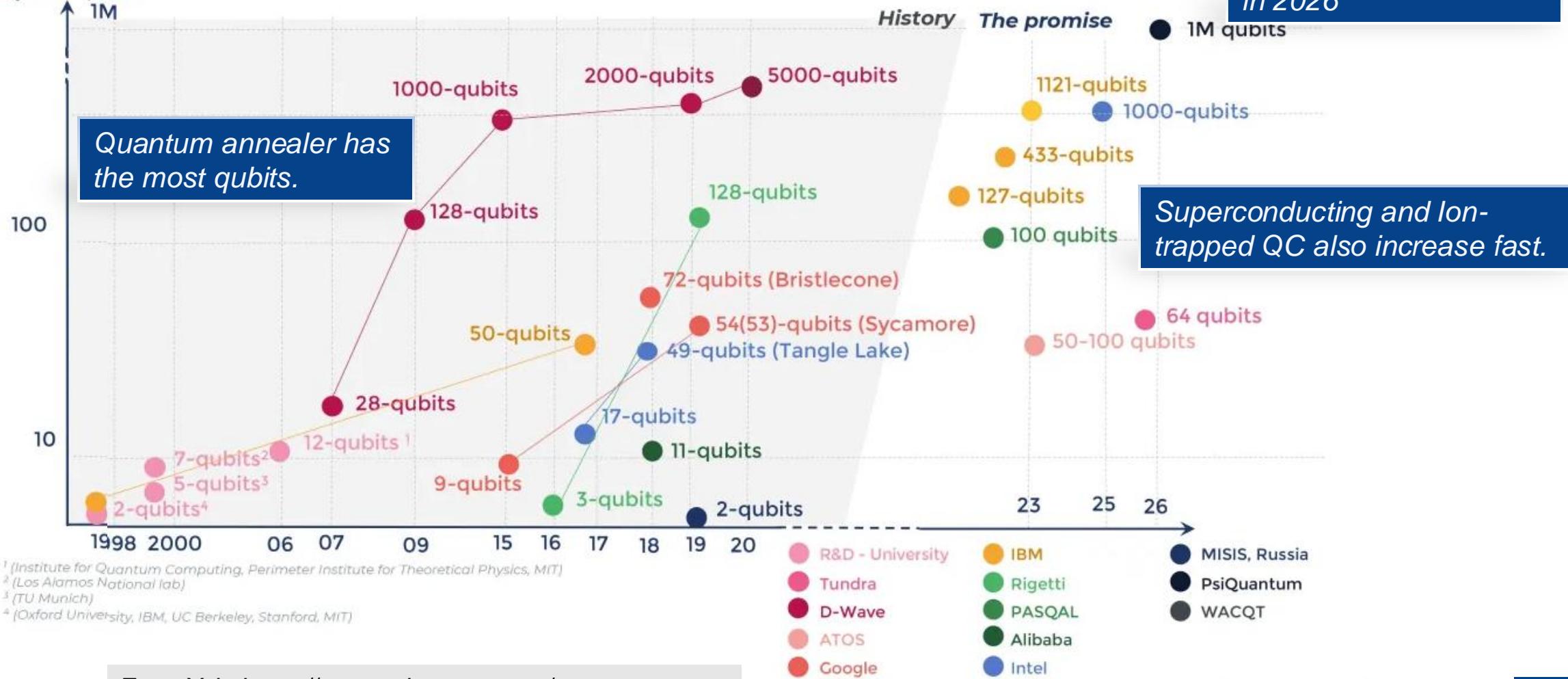
*Classical and quantum computing has similar architecture*

# **Development of Quantum Computer**



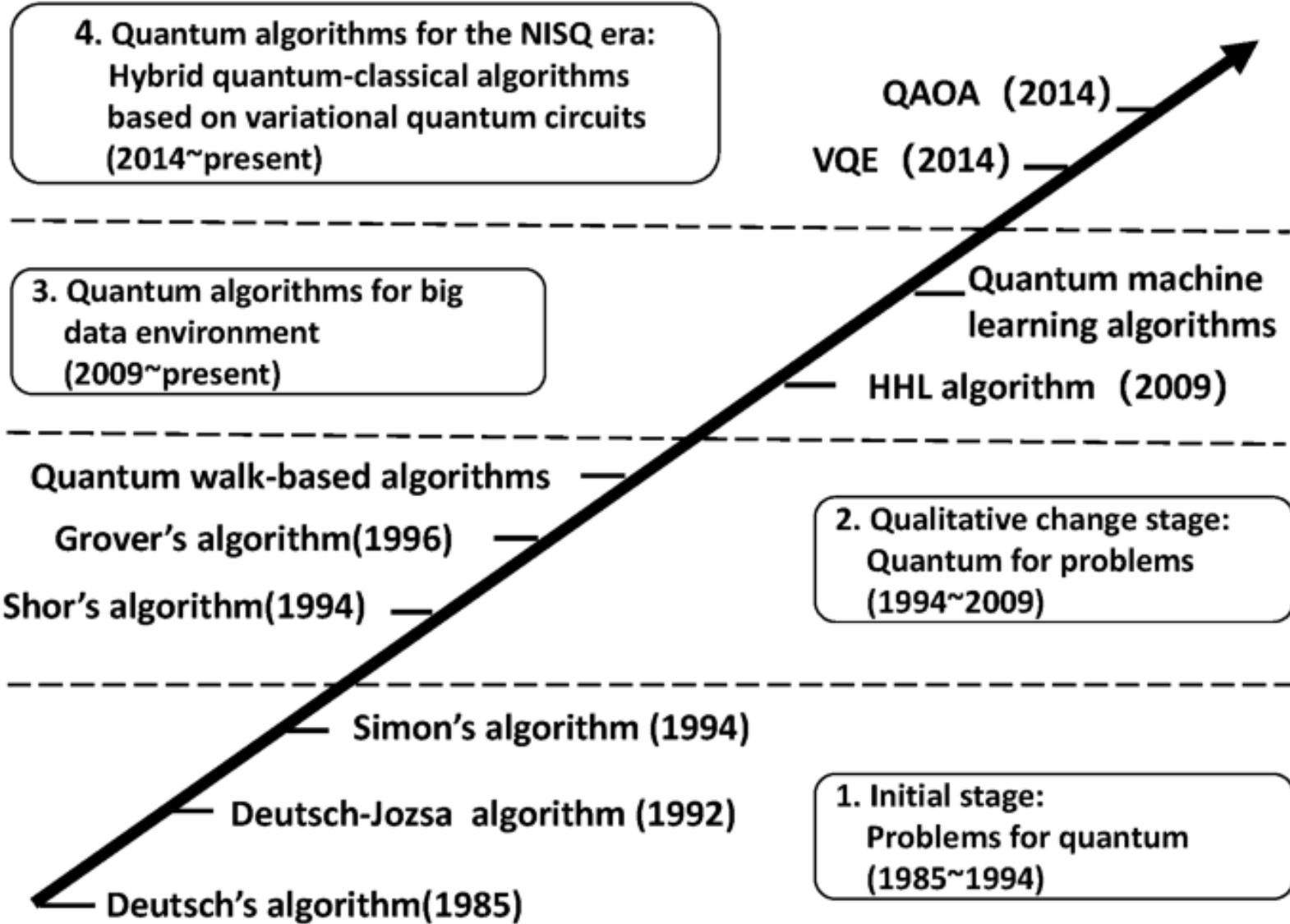
浙江大学  
ZHEJIANG UNIVERSITY

Graph below shows physical qubit roadmap (Note: for a quantum computer, 50 logical qubits minimum are required → it means 50 000 physical qubits)



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

---

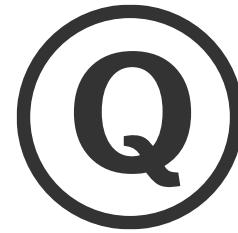


- Tutorial Overview
- Background Knowledge
- **Mathematical Model of Quantum Computing**
- Janus (Taiyuan) Quantum Cloud Platform
- Installing JanusQ



浙江大學  
ZHEJIANG UNIVERSITY

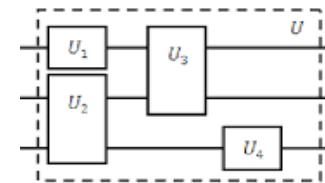
# Mathematical Model of Quantum Computing



Qubits



Quantum Evolution



Quantum Circuit



## 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**  $|\varphi\rangle$ .

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

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

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

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

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

# Quantum Bit (Qubit)



浙江大學  
ZHEJIANG UNIVERSITY

## 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**  $|\varphi\rangle$ .

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

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

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

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

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

## Multiple qubits

**$N$  qubits** has  $2^N$  bases  $|00\cdots 0\rangle, |00\cdots 1\rangle, \dots, |11\cdots 1\rangle$ . The information stored in their **superposition state** is represented as a  **$2^N$ -dimension state vector**.

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

$$|\varphi\rangle = \alpha_0|00\cdots 0\rangle + \alpha_1|00\cdots 1\rangle + \dots + \alpha_{2^N}|11\cdots 1\rangle$$

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

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



## Unitary matrix

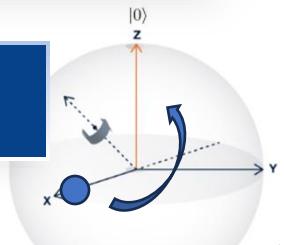
A quantum evolution caused by **quantum gates** is represented as a **unitary matrix (unitary)**, which is a square matrix whose conjugate transpose is its inverse.

$$UU^\dagger = I$$

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

Bloch sphere

X+Z axes

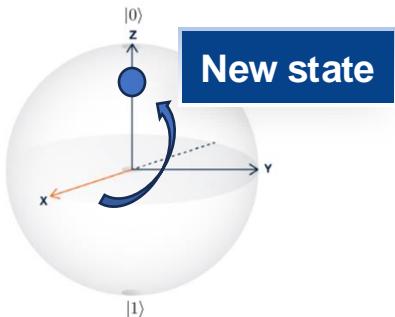


State

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

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

$\xrightarrow{\pi \text{ rotation around X+Z axes}}$



New state

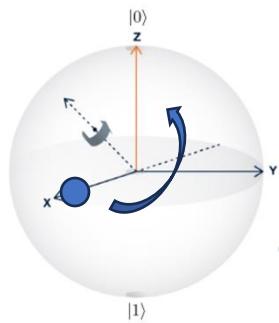
## Unitary matrix

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.

$$UU^\dagger = I$$

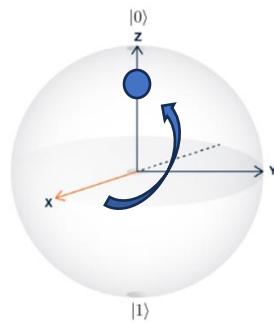
The evolution of qubit state  $|\varphi\rangle$  is represented as:

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



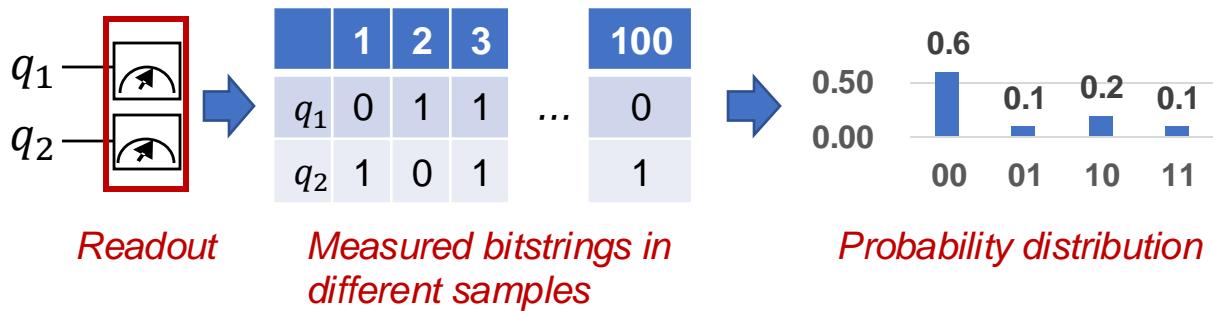
$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$\pi$  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



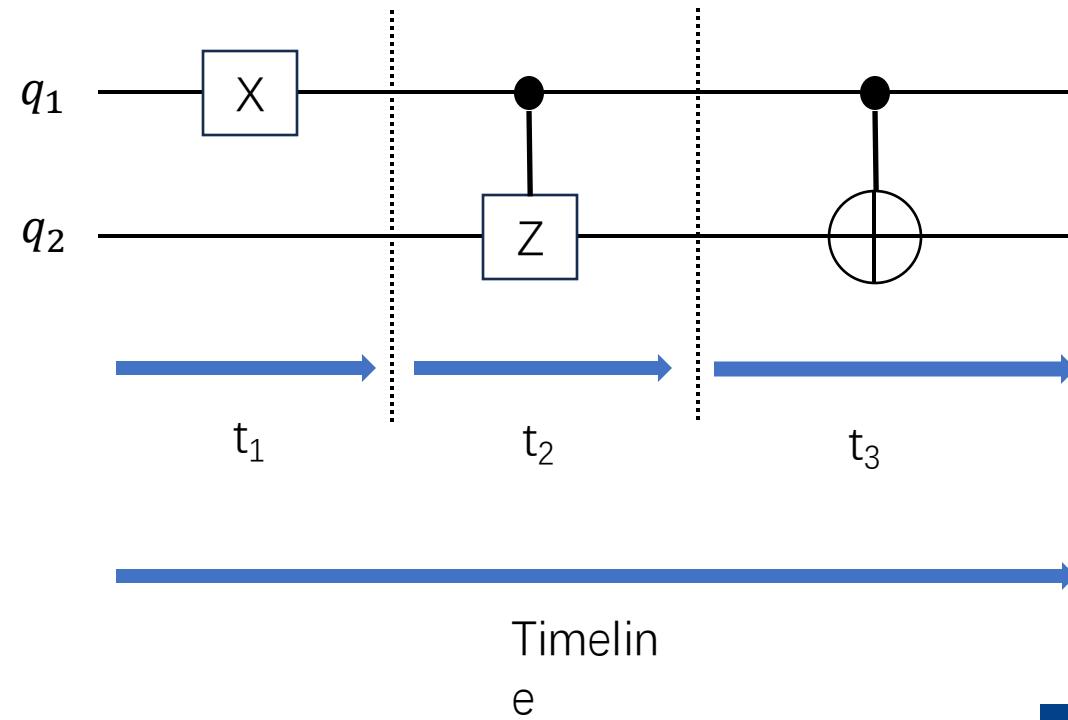
## Quantum gates

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)		$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli-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)		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$
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)		$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$

## Qubit timeline

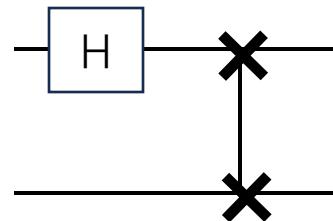
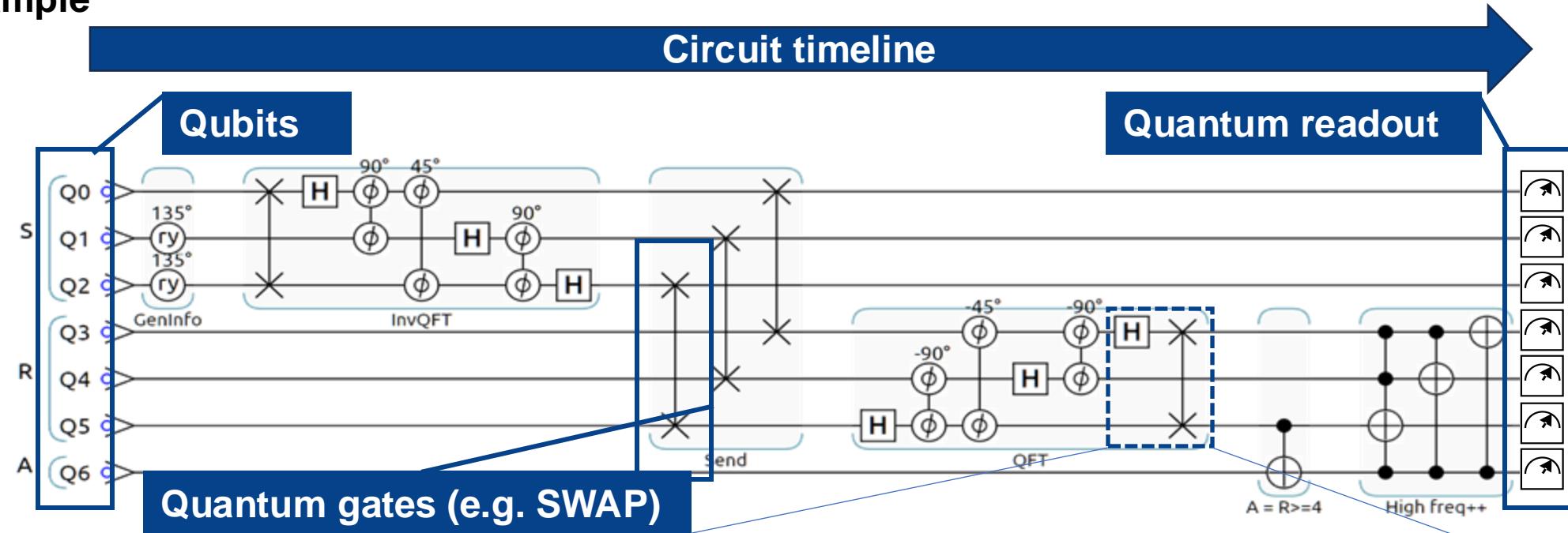
Each line in a quantum circuit represents a **qubit**. The quantum gate in a line is applied on the same qubits **from left to right in time direction**.



# 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

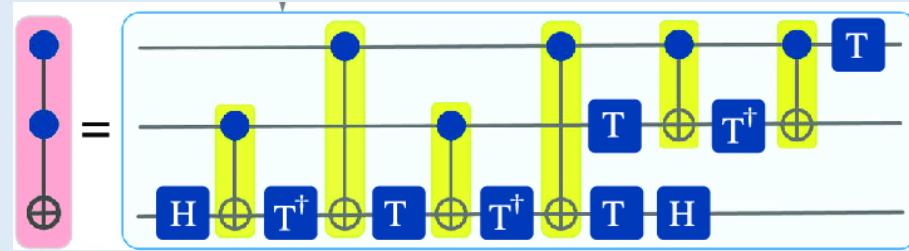
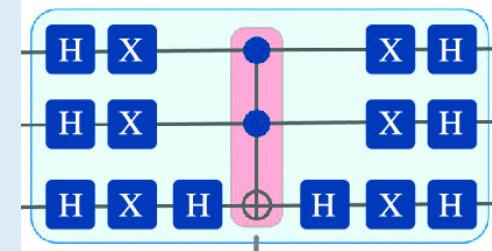


## On superconducting quantum computer

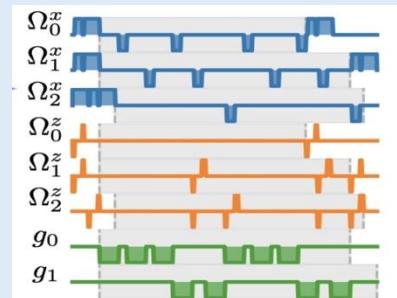
### Step 1. Circuit statement

```
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



### Step 3. Circuit execution



Pulse generation

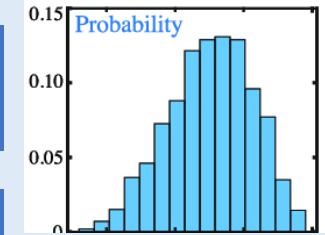


Quantum device

### Step 4. Result processing

Error mitigation

Visualization



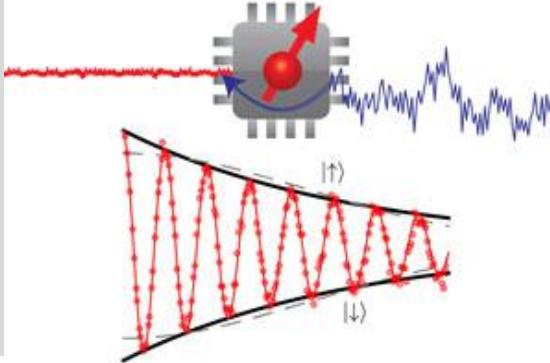
Probability distribution

# Challenge in Quantum Computing



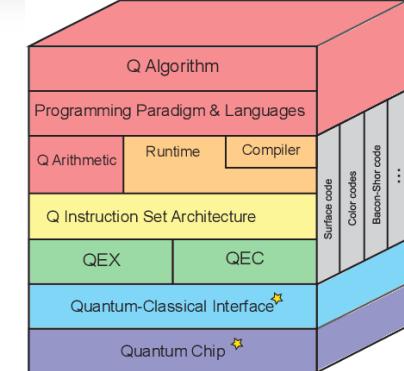
## Noise

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

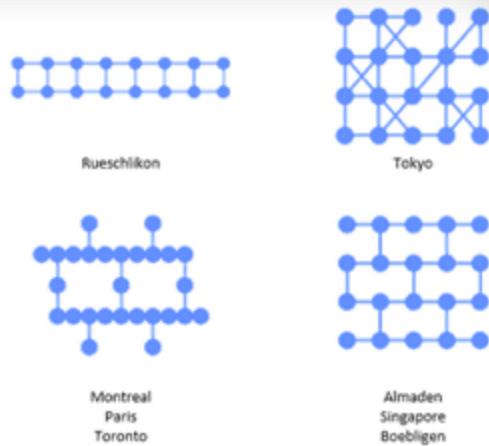


## Instructions

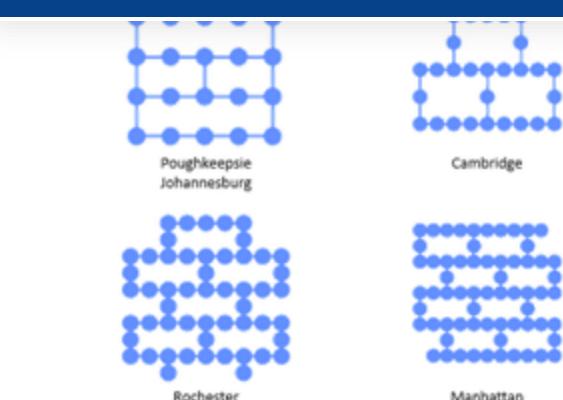
- Multi-level compilation
- Micro-architecture instructions
- Quantum-classical communication



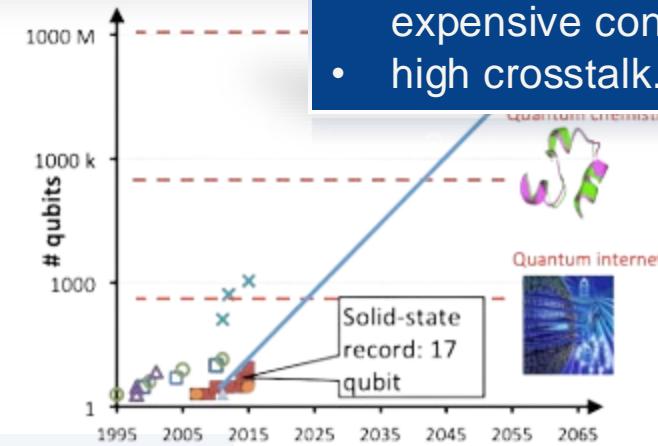
## Topology



## Locality in communication



## Scalability



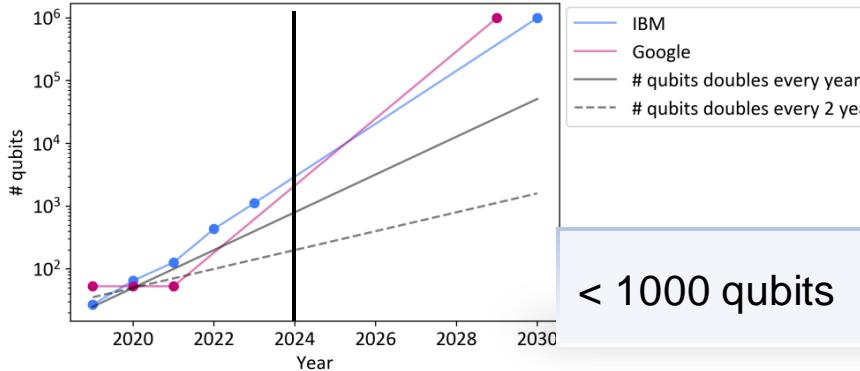
- poor fabrication techniques
- expensive control systems
- high crosstalk.

Constrained by physical implementation

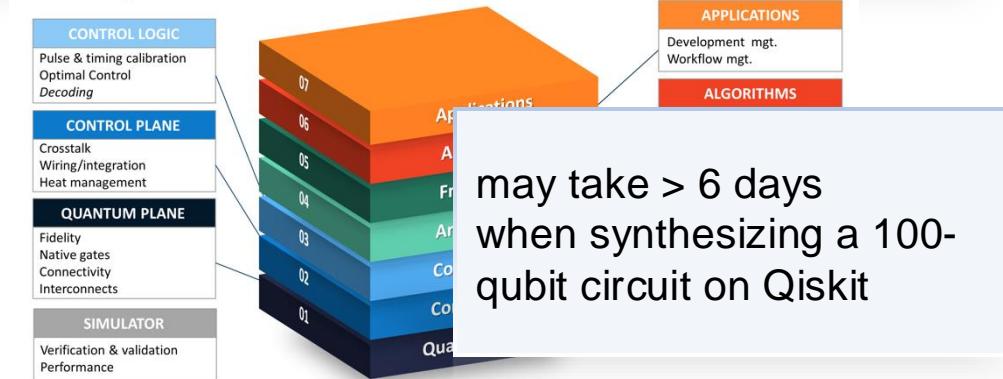
# Results of Challenges



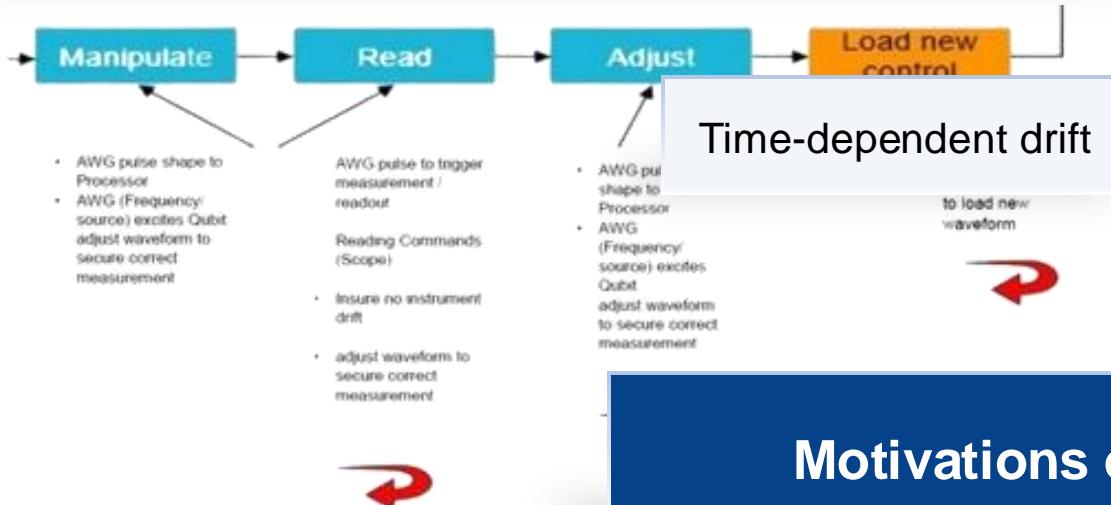
## Limited Hardware Resource



## Compilation Complexity



## Difficulty Of Hardware Calibration



## 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





**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.

# Development History of Janus Cloud



浙江大學  
ZHEJIANG UNIVERSITY

## Four updates since July 2023

Taiyuan Quantum  
Provide real-time quantum computing services  
Get Start

**July 2023**

1. Redesign the cloud interface.
2. Adding the support of English.

Document

### Qucode API

#### 1. Circuit Commands

Preparation work before performing quantum operations, applying for and initializing the quantum circuit.

`new(qubit_number, name)`

Apply for a variable for the quantum circuit, which will contain `qubit_number` quantum bits.

Parameters

`qubit_number` – the number of requested quantum bits.  
`name` – quantum circuit variable name.

Example

```
var a = new(3,'a')
```

Request a variable 'a' with 3 qubits.

`reset(qubit_number)`

Initialize `qubit_number` quantum bits to the |0⟩ state for the quantum circuit. This function should be called before adding a quantum operation.

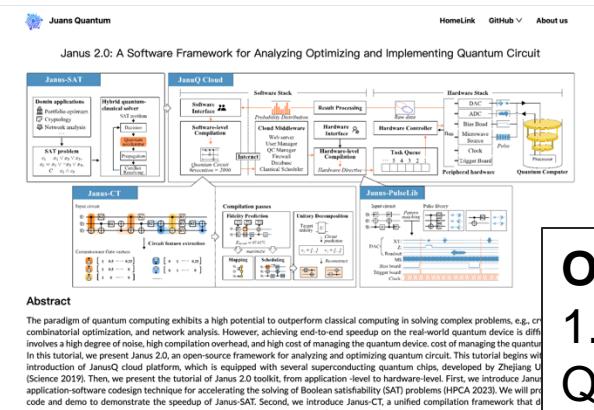
**August 2023**

1. Update the document

serial number	project name	task number	creation time	operation
1	336	0	2023-11-09 16:11:46	<a href="#">edit item</a> <a href="#">view details</a> <a href="#">delete</a>
2	8899	0	2023-11-08 16:11:5	<a href="#">edit item</a> <a href="#">view details</a> <a href="#">delete</a>
3	test20231027	0	2023-10-27 17:10:8	<a href="#">edit item</a> <a href="#">view details</a> <a href="#">delete</a>
4	20230923	0	2023-09-23 20:09	<a href="#">edit item</a> <a href="#">view details</a> <a href="#">delete</a>
5	0923-1	0	2023-09-23 20:09	<a href="#">edit item</a> <a href="#">view details</a> <a href="#">delete</a>
6	23	0	2023-09-23 20:09	<a href="#">edit item</a> <a href="#">view details</a> <a href="#">delete</a>
7	通信	0	2023-09-16 09:09	<a href="#">edit item</a> <a href="#">view details</a> <a href="#">delete</a>

**September 2023**

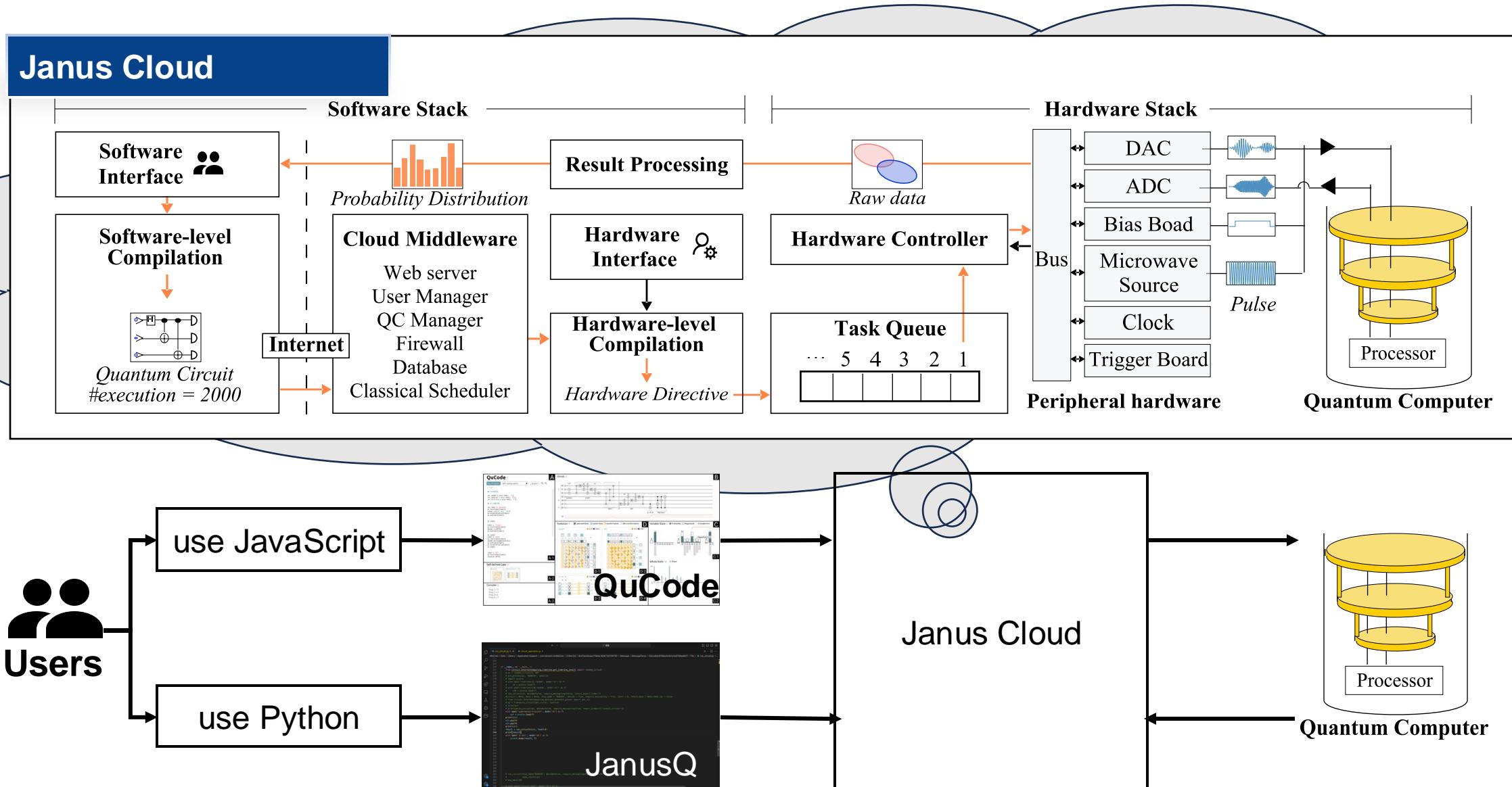
1. Added multi-level project management.



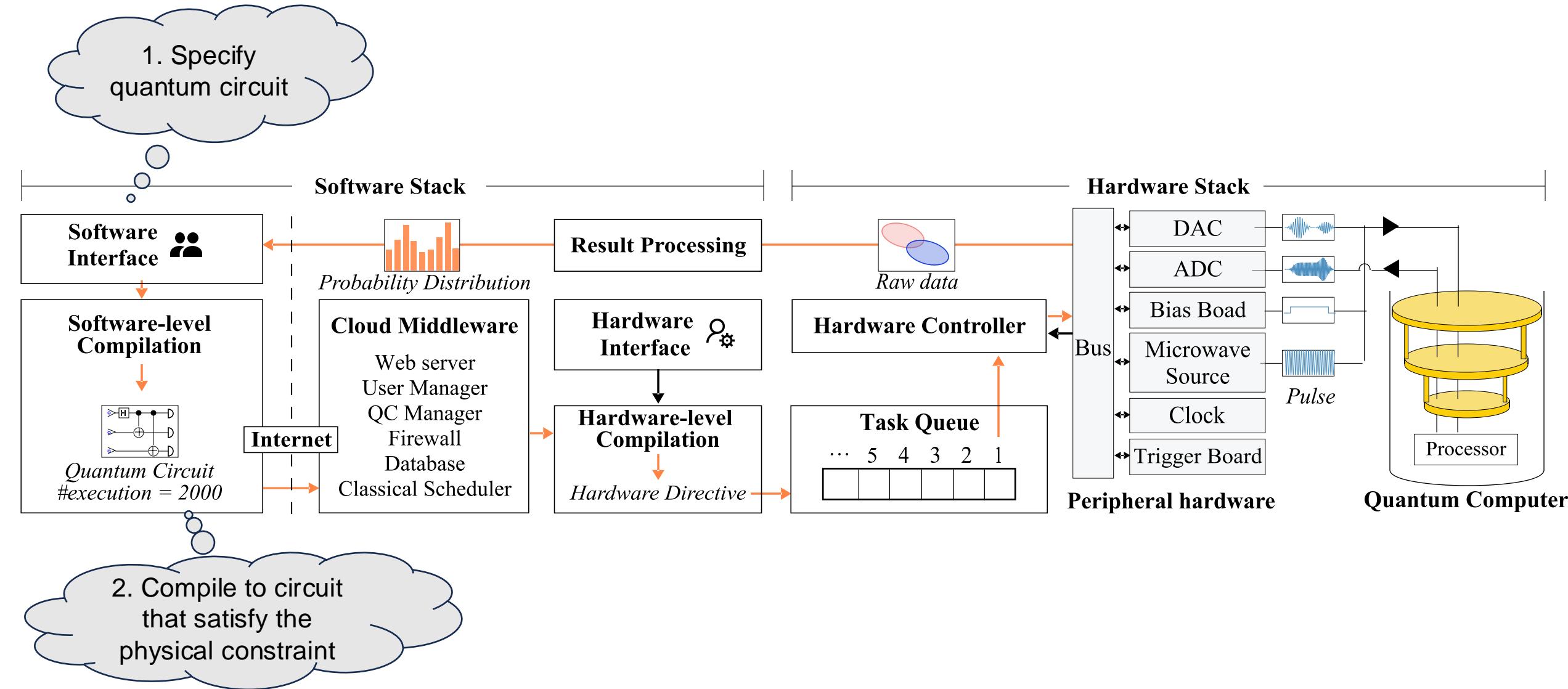
**October 2023**

1. Enable the online QuCT and HyQSAT

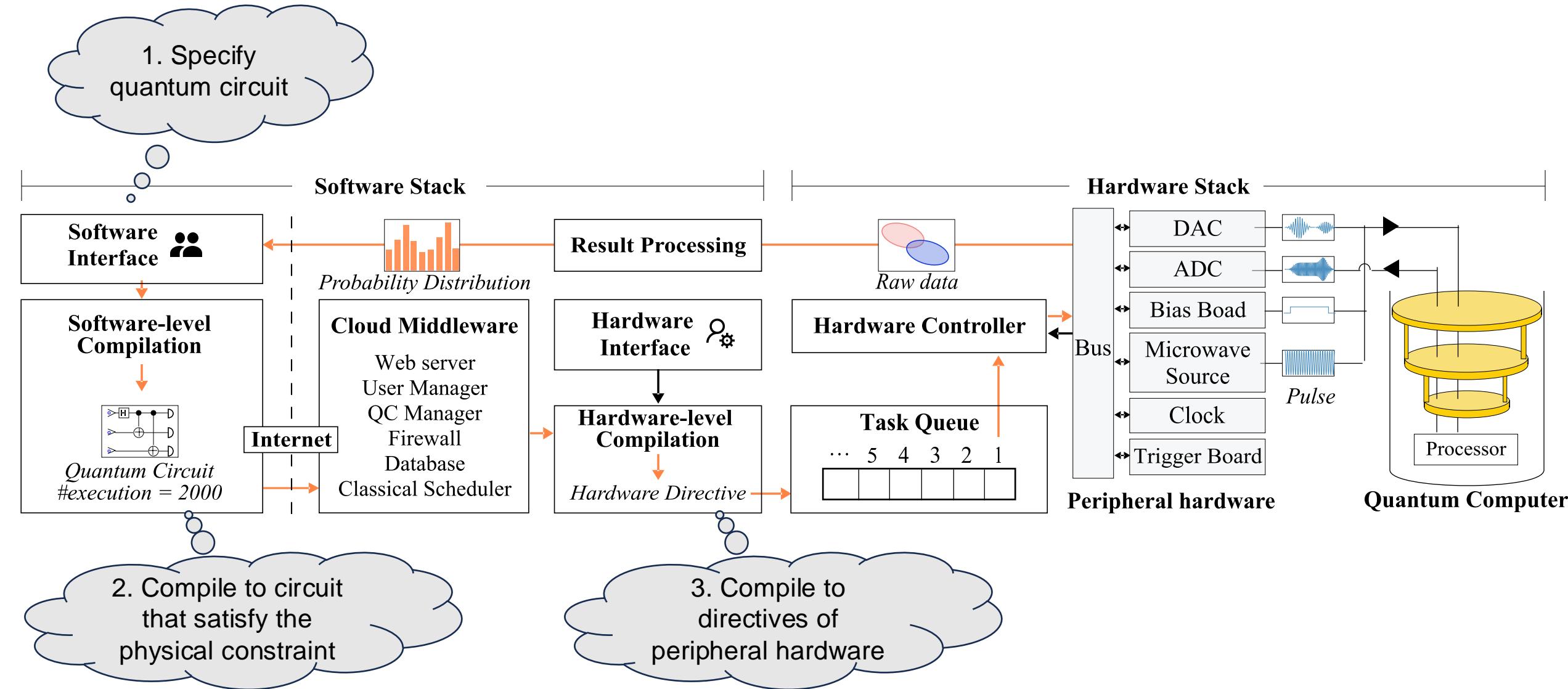
# What we can do on Janus Platform



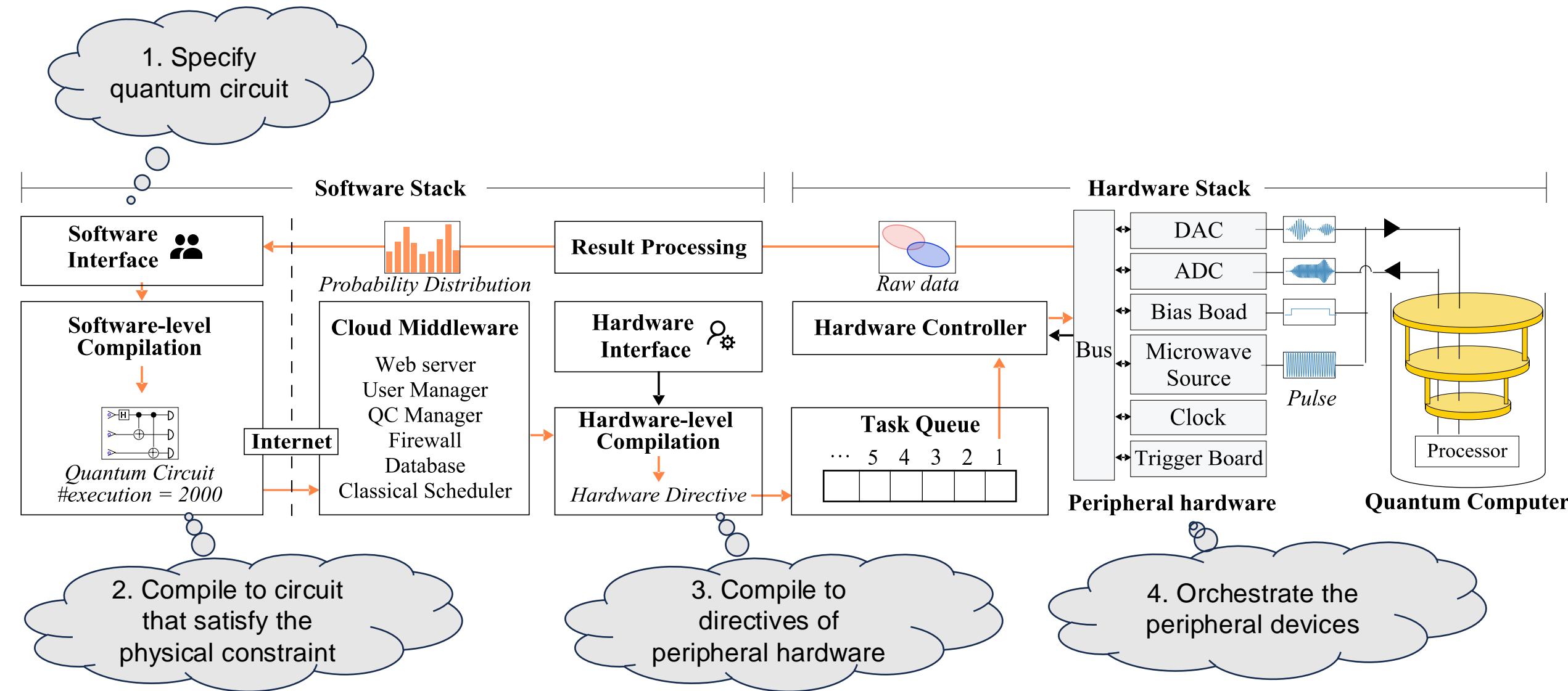
# What we can do on Janus Platform



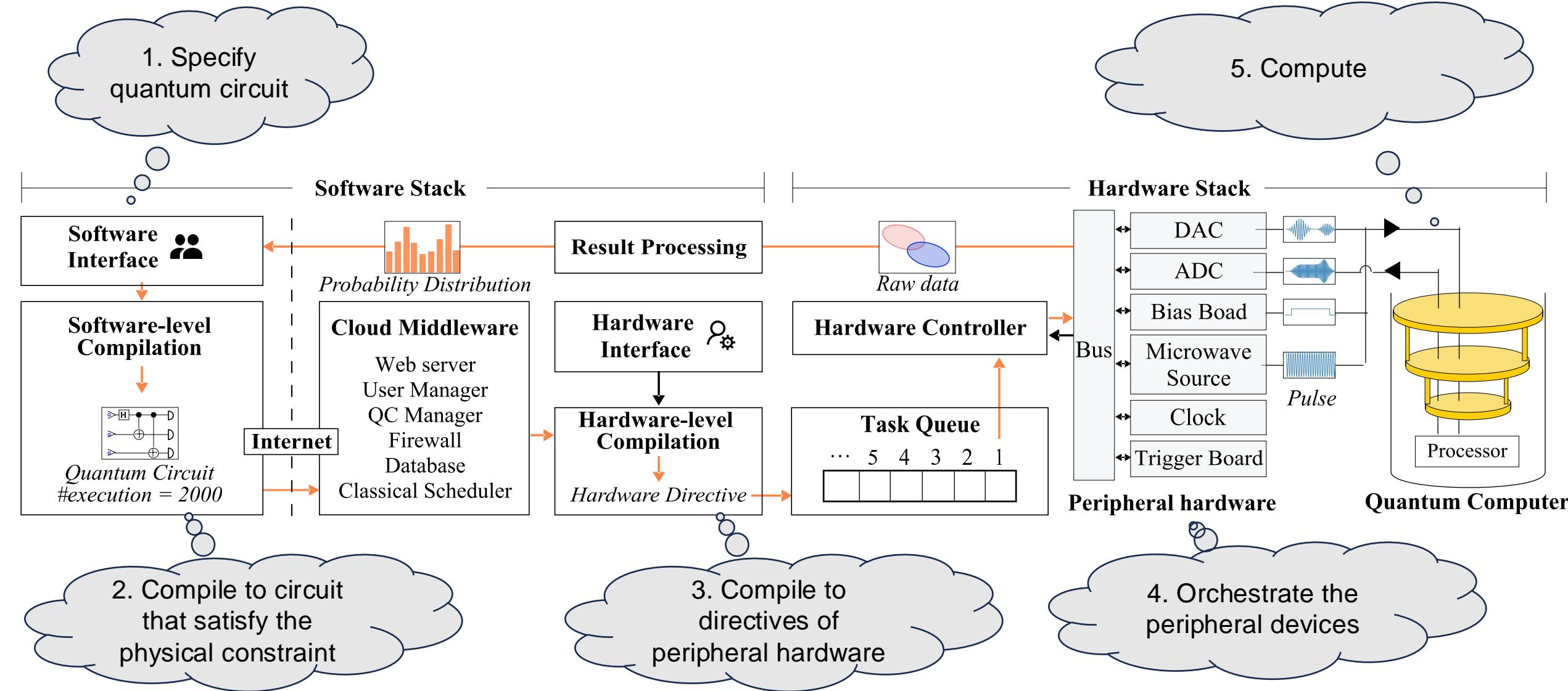
# What we can do on Janus Platform



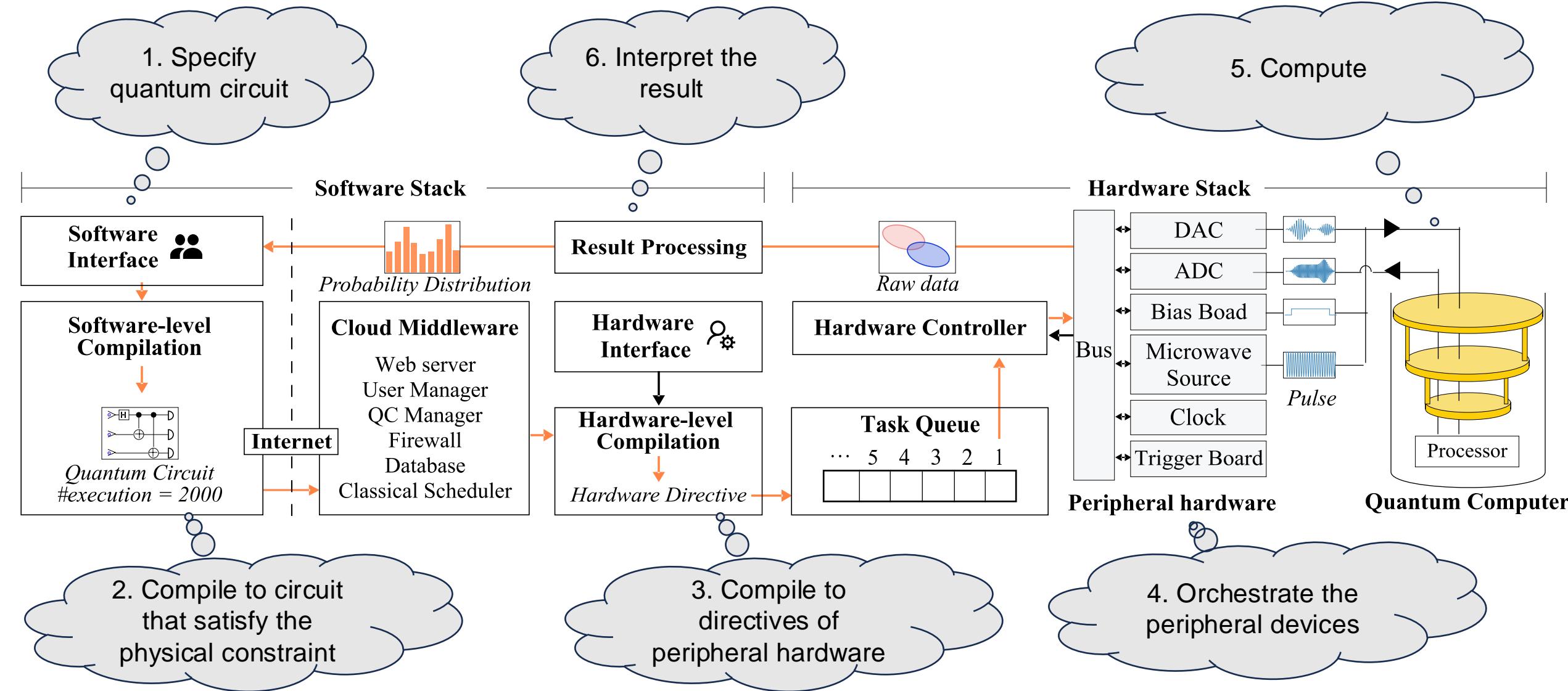
# What we can do on Janus Platform



# What we can do on Janus Platform



# What we can do on Janus Platform



# Creating an Account



浙江大學  
ZHEJIANG UNIVERSITY

Go to Janus Cloud

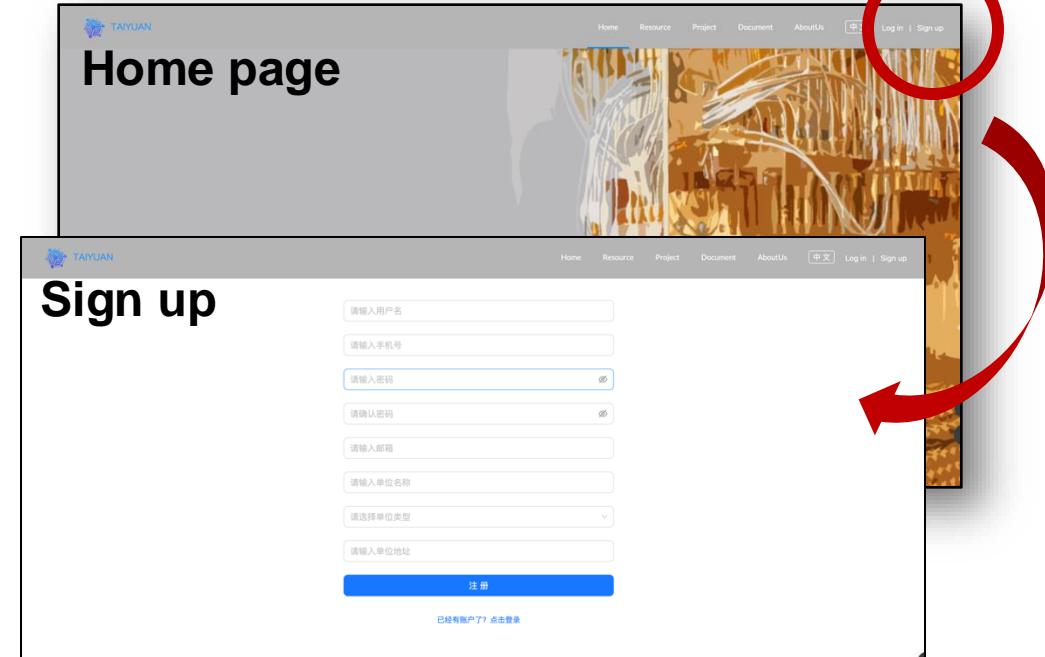
Open <http://janusq.zju.edu.cn/home>



QR code

Sign up

Click sign up button



- After signing up, you will get an API key for task submission.
- **The quantum computer can be accessed on the website during the tutorial !!**

# Running Program On the Website



## Submitting on website

Open <http://janusq.zju.edu.cn/home>

TAIYUAN

### Home page

Taiyuan Quantum

Provide real-time quantum computing service

Get Start

Click

## Programming on the Code editor

QuCode

A

Circuit

B

D

C

C-1

C-2

D-1

D-2

D-3

D-4

A-1

A-2

A-3

InvQFT

QFT

Variable State

Whole State

Probability

Magnitude

Entanglement

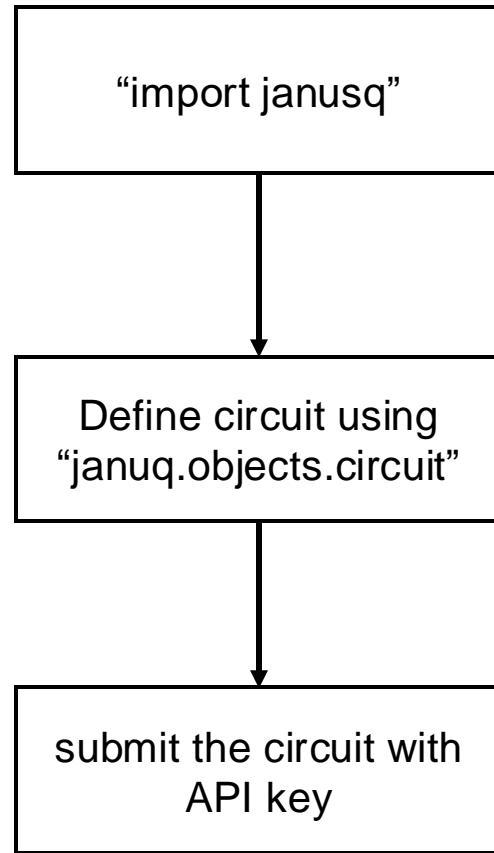
Phase

- The quantum computer can be accessed on the website during the tutorial !!

# 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



浙江大學  
ZHEJIANG UNIVERSITY

**QuCode**

Run Program
QFT manipulation
Export

*VV QFT*

### Program editor

```
var sender = qint.new(3, 'S')
var receiver = qint.new(3, 'R')
var ancillary = qint.new(1, 'A')

qc.write(0x0)

let label = 'genData'
qc.startlabel(label)
sender.ry(135, 0x2 | 0x4)
qc.disableDisplay(label)
qc.endlabel(label)

qc.nop()

label = 'InvQFT'
qc.startlabel(label)
sender.invQFT()
qc.endlabel(label)

qc.nop()
label = 'send'
qc.startlabel(label)
sender.exchange(receiver)
qc.endlabel(label)
qc.disableDisplay(label)
qc.nop()

label = 'QFT'
qc.startlabel(label)
receiver.QFT()
```

### Self-defined Gate

X InvQFT 3	X High freq++ 4
------------	-----------------

### Console

```
Freq 1 = 7
Freq 2 = 7
Freq 3 = 0
Freq 4 = 7
```

**A**
**B**

**Circuit**

### Quantum circuit view

**D**
**C**

**Evolution**

InvQFT
0.35
0.85

S	R

**Variable State**

S
R
A

**D-1**
**D-2**
**C-1**
**C-2**

**QFT**

0.35
0.85

S	R

**High freq++**

1.00
0.85

R	A

**D-3**
**D-4**
**State view**
**C-3**

**A = R>=8**

R
A

S	R

**High freq++**

R
A

R	A

**D-1**
**D-2**
**C-4**
**C-5**

**Whole State**

Phase

**State view**

Probability
Magnitude
Entanglement

HPCA 2025

53

# QuCode: Online Quantum Program Editor



浙江大學  
ZHEJIANG UNIVERSITY

Program editor

Run Program
QFT manipulation ▾
Export
Q
Q

```
VV QFT

qc.reset(7);

var sender = qint.new(3, 'S')
var receiver = qint.new(3, 'R')
var ancillary = qint.new(1, 'A')

qc.write(0x0)

let label = 'genData'
qc.startlabel(label)
sender.ry(135, 0x2 | 0x4)
qc.disableDisplay(label)
qc.endlabel(label)

qc.nop()

label = 'InvQFT'
qc.startlabel(label)
sender.invQFT()
qc.endlabel(label)

qc.nop()
label = 'send'
qc.startlabel(label)
sender.exchange(receiver)
qc.endlabel(label)
qc.disableDisplay(label)
qc.nop()

label = 'QFT'
qc.startlabel(label)
receiver.QFT()
```

**A** Circuit

**B** Evolution

**C** Variable State

**D** Self-defined Gate

**A-1** Consoler
 

```
Freq 1 = 7
Freq 2 = 7
Freq 3 = 0
Freq 4 = 7
```

**A-2** Self-defined Gate

**A-3** Evolution

**D-1** InvQFT

**D-2** QFT

**D-3** A = R >= 8

**D-4** High freq++

**C-1** Variable State

**C-2** Whole State

HPCA 2025

54

# QuCode: Online Quantum Program Editor



浙江大學  
ZHEJIANG UNIVERSITY

Program editor A Circuit B

Provide 14 examples:

1. QFT
2. teleportation
3. Grover
- ...

qc.reset();  
var sender = qint.new(3, 'S');  
var receiver = qint.new(3, 'R');  
var ancillary = qint.new(1, 'A');  
qc.write(0x0);  
  
let label = 'genData';  
qc.startlabel(label);  
sender.ry(135, 0x2 | 0x4);  
qc.disableDisplay(label);  
qc.endlabel(label);  
  
qc.nop();  
  
label = 'InvQFT';  
qc.startlabel(label);  
sender.invQFT();  
qc.endlabel(label);  
  
qc.nop();  
label = 'send';  
qc.startlabel(label);  
sender.exchange(receiver);  
qc.endlabel(label);  
qc.disableDisplay(label);  
qc.nop();  
  
label = 'QFT';  
qc.startlabel(label);  
receiver.QFT();

A-1 A-2 A-3

D D-1 D-2 D-3 D-4

C C-1 C-2

Self-defined Gate  
InvQFT 3 High freq++ 4

Consolier  
Freq 1=7  
Freq 2=7  
Freq 3=0  
Freq 4=7

Variable State  
Probability Magnitude Entanglement

QFT S R A

High freq++ S R A

Whole State Phase

# QuCode: Online Quantum Program Editor



浙江大學  
ZHEJIANG UNIVERSITY

**QuCode** ⓘ

Run Program   QFT manipulation   Export   🔍   🌐

```

// QFT
qc.reset(7);

var sender = qint.new(3, 'S')
var receiver = qint.new(3, 'R')
var ancillary = qint.new(1, 'A')

qc.write(0x0)

let label = 'genData'
qc.startlabel(label)
sender.ry(135, 0x2 | 0x4)
qc.disableDisplay(label)
qc.endlabel(label)

qc.nop()

label = 'InvQFT'
qc.startlabel(label)
sender.invQFT()
qc.endlabel(label)

qc.nop()
label = 'send'
qc.startlabel(label)
sender.exchange(receiver)
qc.endlabel(label)
qc.disableDisplay(label)
qc.nop()

label = 'QFT'
qc.startlabel(label)
receiver.QFT()
        
```

### Circuit View

Grouping

Purity

**A-1**

**D-1**

**C-1**

**A-2**

**D-2**

**C-2**

**A-3**  
**Consoler** ⓘ  
 Freq 1 = 7  
 Freq 2 = 7  
 Freq 3 = 0  
 Freq 4 = 7

**D-3**

**D-4**

# QuCode: Online Quantum Program Editor



浙江大學  
ZHEJIANG UNIVERSITY

**QuCode**

Run Program   QFT manipulation   Export   🔍   🔍

```

// QFT
qc.reset(7);

var sender = qint.new(3, 'S')
var receiver = qint.new(3, 'R')
var ancillary = qint.new(1, 'A')

qc.write(0x0)

let label = 'genData'
qc.startlabel(label)
sender.ry(135, 0x2 | 0x4)
qc.disableDisplay(label)
qc.endlabel(label)

qc.nop()

label = 'InvQFT'
qc.startlabel(label)
sender.invQFT()
qc.endlabel(label)

qc.nop()
label = 'send'
qc.startlabel(label)
sender.exchange(receiver)
qc.endlabel(label)
qc.disableDisplay(label)
qc.nop()

label = 'QFT'
qc.startlabel(label)
receiver.QFT()
      
```

**A**

**Circuit**

**Evolution**

InvQFT

0.35   0.85

QFT

0.35   0.85

A = R>=8

1.00   0.85

High freq++

1.00   0.85

**B**

**A-1**

**D-1**

**A-2**

**D-2**

**A-3**

**D-3**

**D-4**

**C**

**State View**

partial trace

The whole state

**C-1**

**C-2**

# QuCode: Online Quantum Program Editor



浙江大學  
ZHEJIANG UNIVERSITY

**QuCode**

Run Program   QFT manipulation ▾   Export   🔍   🔍

```

V// QFT

qc.reset(7);

var sender = qint.new(3, 'S')
var receiver = qint.new(3, 'R')
var ancillary = qint.new(1, 'A')

qc.write(0x0)

let label = 'genData'
qc.startlabel(label)
sender.ry(135, 0x2 | 0x4)
qc.disableDisplay(label)
qc.endlabel(label)

qc.nop()

label = 'InvQFT'
qc.startlabel(label)
sender.invQFT()
qc.endlabel(label)

qc.nop()
label = 'send'
qc.startlabel(label)
sender.exchange(receiver)
qc.endlabel(label)
qc.disableDisplay(label)
qc.nop()

label = 'QFT'
qc.startlabel(label)
receiver.QFT()
      
```

A
B

**Circuit**

A-1
D-1
C-1

A-2
D-2
C-2

A-3
D-3
D-4

**Evolution View**

D
C

1. Visualize the unitary

2. Visualize the change of qubit states

D-1
D-2

# Outline of Presentation

---



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



浙江大學  
ZHEJIANG UNIVERSITY

# Installing JanusQ



<https://github.com/JanusQ/JanusQ>



**Language:** Python 3, C++

**Available platforms:** Linux, Mac, Windows

*HyQSAT has not been tested on Windows.*

**Hardware requirements:**

- Classical computer:  $\geq 16$  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.

**Installation:**

**From Docker (recommend)**

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

**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
```

Docker page of JanusQ:

<https://hub.docker.com/r/janusq/janusq3>

## Step 5. Or use Jupyter Lab to connect the docker

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

*The password of the docker is “root”.*

# Install from Source Code



浙江大學  
ZHEJIANG UNIVERSITY

## Step 1. clone the source code

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

## Step 2. install requirements (virtual environment is preferred)

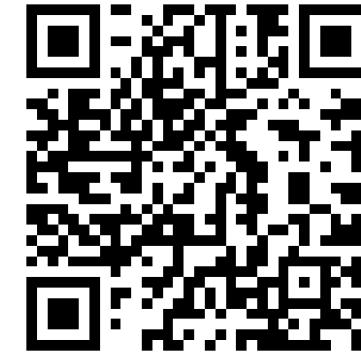
```
cd ./JanusQ  
conda create -n janusq python=3.10  
conda activate janusq  
pip install -r requirements.txt
```

## Step 3. compile HyQSAT

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

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

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



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

*implemented based on C++.*

# 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 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 QuFEM

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

protocol = EnumeratedProtocol(4)
```

## 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](https://janusq.github.io/HPCA_2025_Tutorial/demo)

- Getting Started
  - └ Install JanusQ
  - └ Submit to cloud
- QuCT
  - └ Vectorization model of QuCT
  - └ Fidelity prediction of QuCT on quantum simulators
  - └ Fidelity prediction of QuCT on real quantum devices
  - └ Fidelity optimization based on QuCT
  - └ Unitary decomposition based on QuCT
  - └ Extending framework for bug identification
- MorphQPV
  - └ Verify Quantum Program
- QuFEM
  - └ Readout calibration of QuFEM on quantum simulators
  - └ Readout calibration of QuFEM on real quantum devices
- Choco-Q
  - └ Constrained binary optimization with QAOA

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 develop 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 uses a table comparison to generate the gate vector.



浙江大學  
ZHEJIANG UNIVERSITY

# Thanks for listening!