



浙江大學
ZHEJIANG UNIVERSITY

HPCA 2025 Tutorial

Topic 1. Introduction of Janus Quantum Cloud Platform and Installing JanusQ



Speaker: Liqiang Lu

College of Computer Science and Technology
Zhejiang University (ZJU)

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



Liqiang Lu

liqianglu@zju.edu.cn

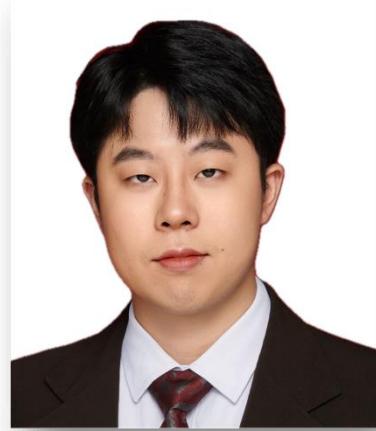
Liqiang Lu is a ZJU100 Young Professor in the College of Computer Science, Zhejiang University. His research interests include quantum computing, computer architecture, deep learning accelerator, and software-hardware codesign. He has authored more than 30 scientific publications in premier international 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”.



Siwei Tan

siweitan@zju.edu.cn

Siwei Tan received the Ph.D. degree in computer science from Zhejiang University (ZJU) in 2024 and keep staying at Zhejiang University as a Tenure-track Assistant Professor. He is interested in the quantum software, quantum hardware, and machine learning. He has published more than 20 papers in top international journals and conferences such as ASPLOS, MICRO, HPCA, DAC, ICCAD, FSE, TVCG, et al.

[FSE 2025] **Siwei Tan**, Liqiang Lu, Debin Xiang, et al. “HornBro: Homotopy-like Method for Automated Quantum Program Repair”.

[ASPLOS 2024] **Siwei Tan**, Debing Xiang, Liqiang Lu, et al. “MorphQPV: Exploiting Isomorphism in Quantum Programs to Facilitate Confident Verification”.

[HPCA 2023] **Siwei Tan**, Mingqian Yu, Liqiang Lu, et al. “HyQSAT: A Hybrid Approach for 3-SAT Problems by Integrating Quantum Annealer with CDCL”.

[MICRO 2023] **Siwei Tan**, Congliang Lang, Liang Xiang, et al. “QuCT: A Framework for Analyzing Quantum Circuit by Extracting Contextual and Topological Features”.

Presenter



Tianyao Chu is a first-year PhD student at the College of Computer Science, Zhejiang University. His research interests include multi-node quantum computing and quantum networking system. He is currently working on the quantum network-on-chip based on enhanced QST.

Tianyao Chu

tianyao_chu@zju.edu.cn

Presenter



Kaiwen Zhou is a first-year PhD student at the College of Computer Science, Zhejiang University. He is interested in quantum computer architecture and high-performance computing. He is currently working on designing a QLDPC decoding accelerator.

Kaiwen Zhou

kaiwenzhou@zju.edu.cn

Outline of Presentation



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

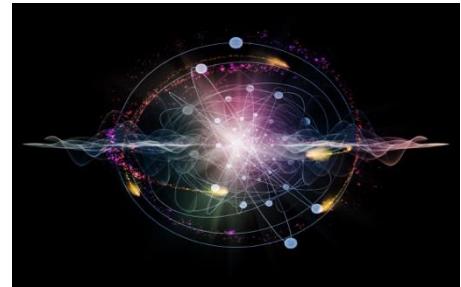


浙江大學
ZHEJIANG UNIVERSITY

Background Knowledge



Development of
Classical Computing



Motivation of
Quantum Computing



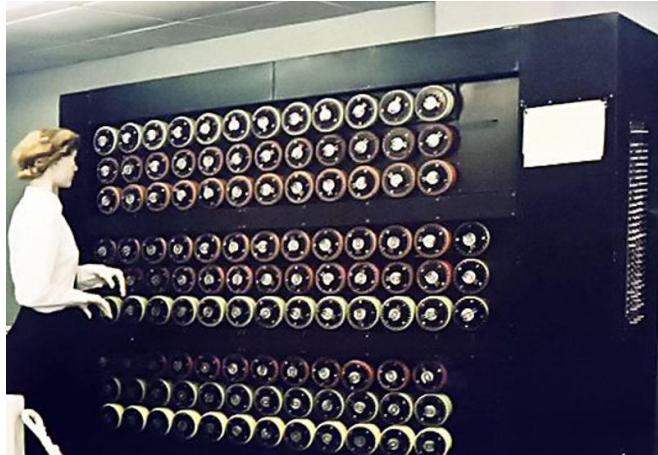
Development of
Quantum Computing

Development Of Classical Computing

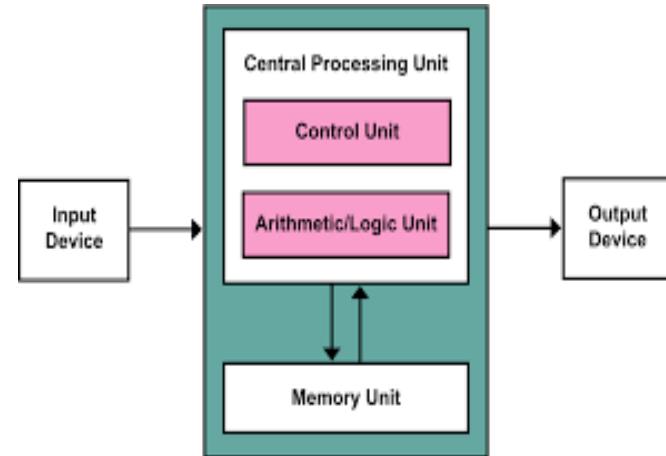


浙江大學
ZHEJIANG UNIVERSITY

Domain-specific calculator (1939)



Von Neumann architecture (1947)



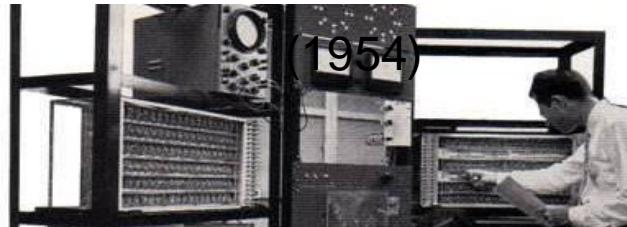
IBM360 Integrated Circuit (1964)



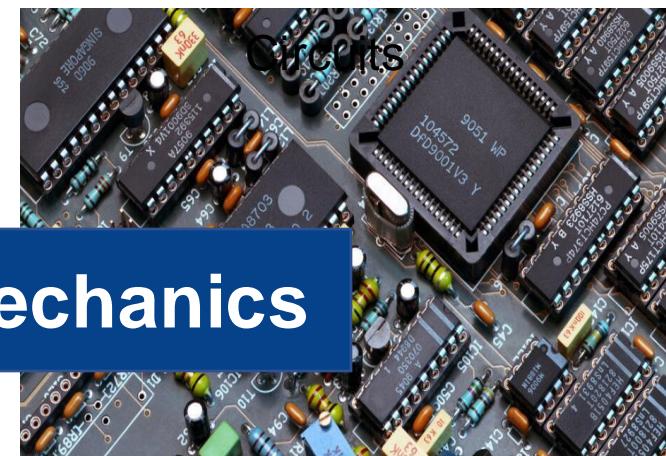
Vacuum Tube Computer ENIAC (1942)



Transistor Computer TRADIC



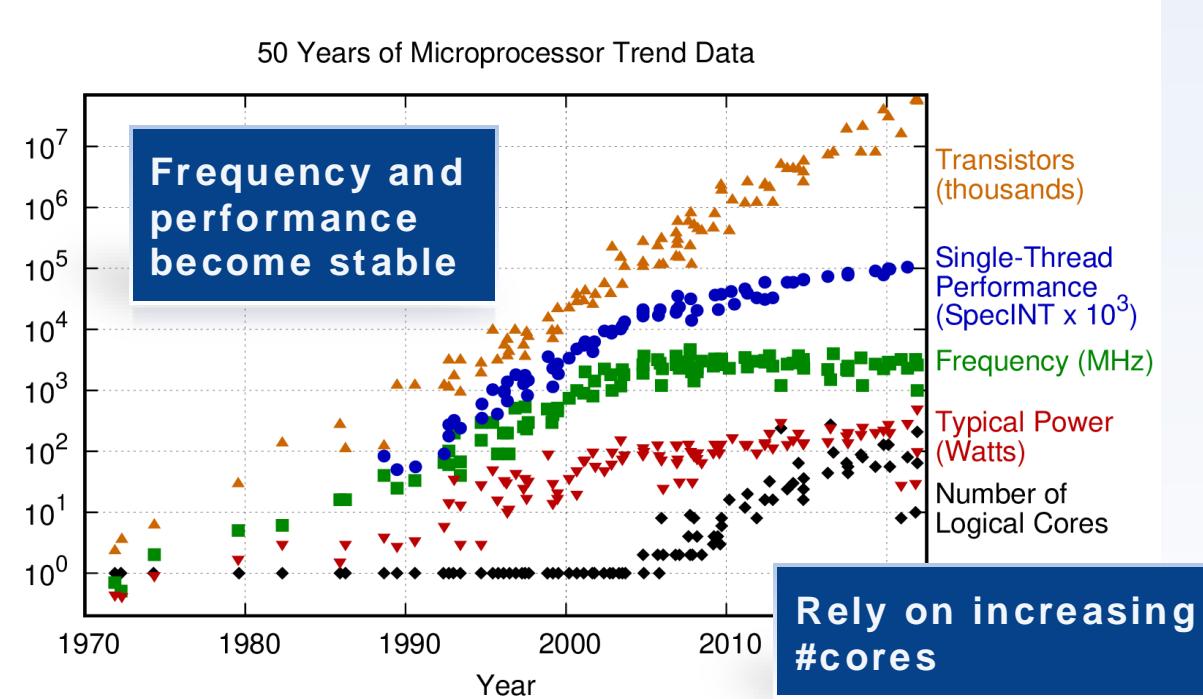
Large Scale Integrated Circuits



Macroscopic Effects of Quantum Mechanics

Motivation Of Quantum Computing

Computation barrier

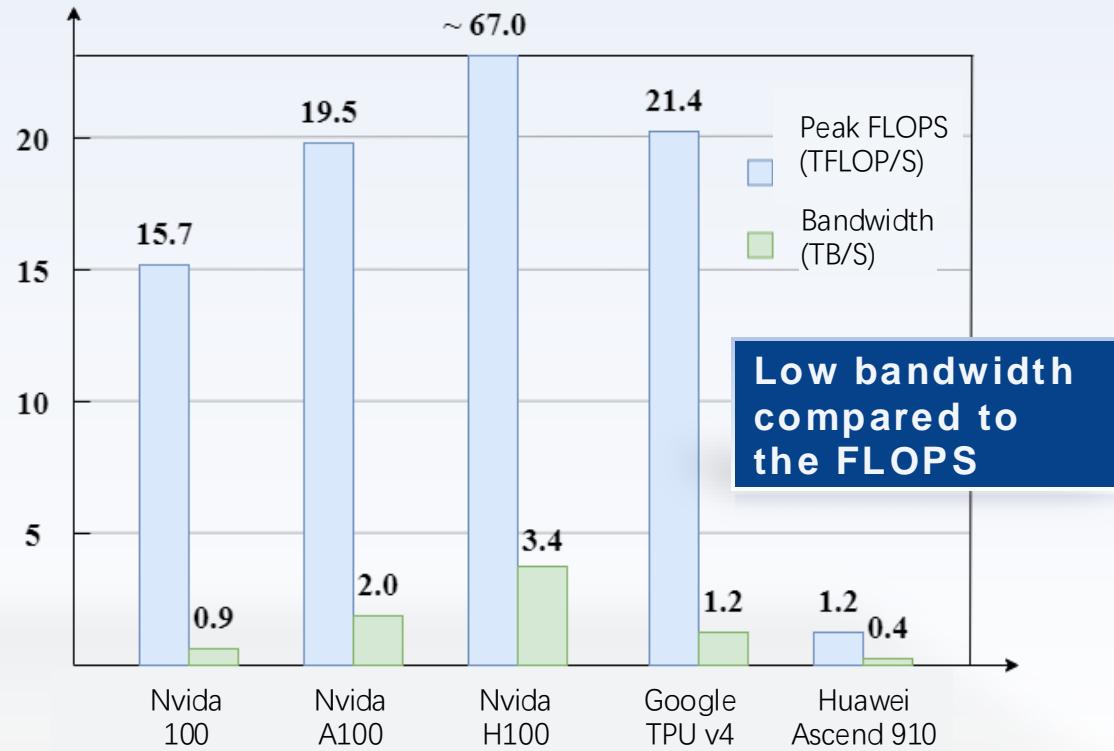


Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonne, O. Shacham, K. Olukolu, L. Hammond, and C. Balen. New plot and data collected for 2010-2021 by K. Rupp.

- 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.

Motivation Of Quantum Computing

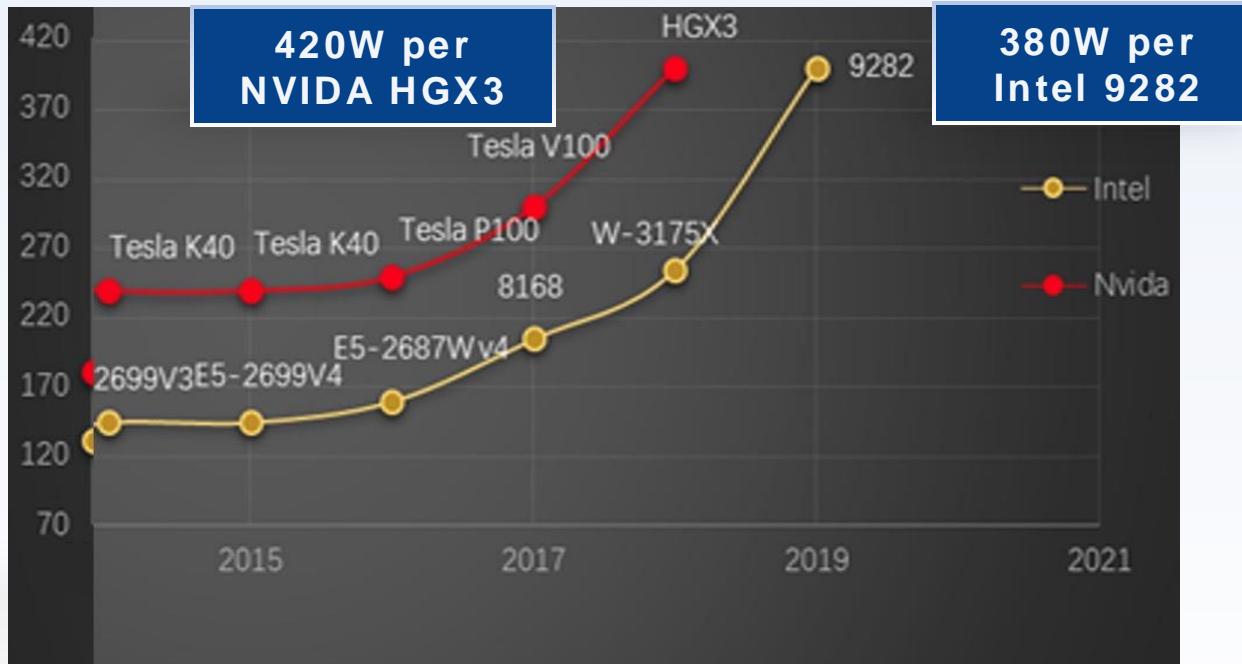
Storage barrier



- 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.

Power wall

Thermal design power exponentially increases.



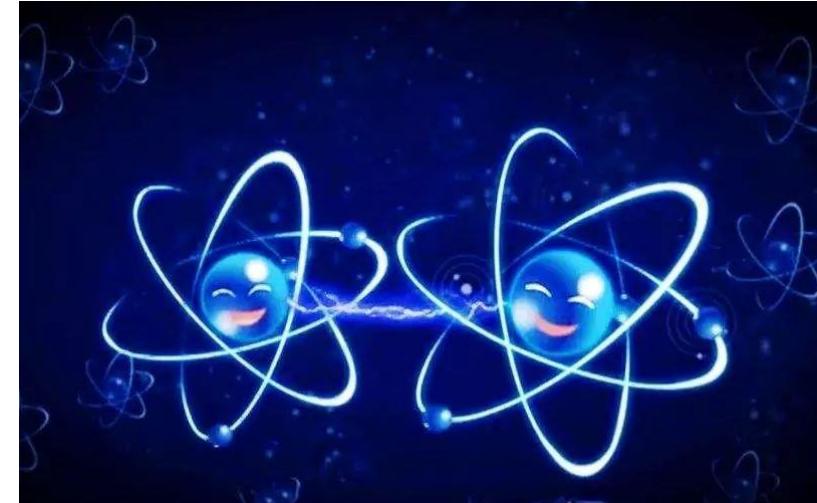
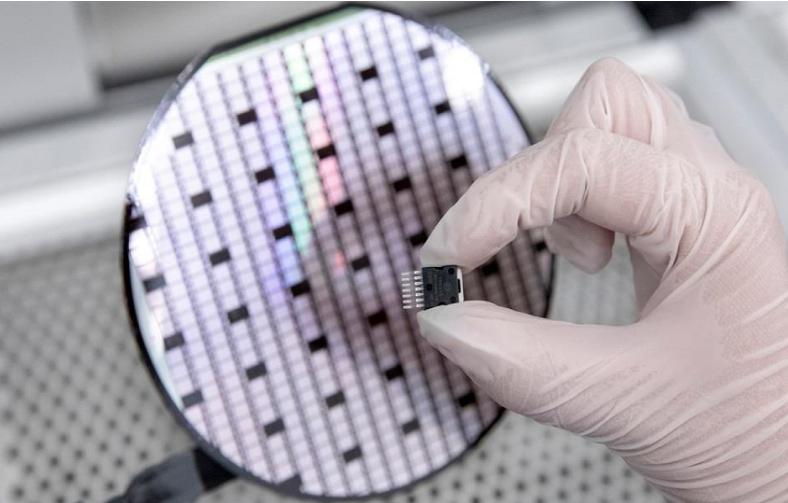
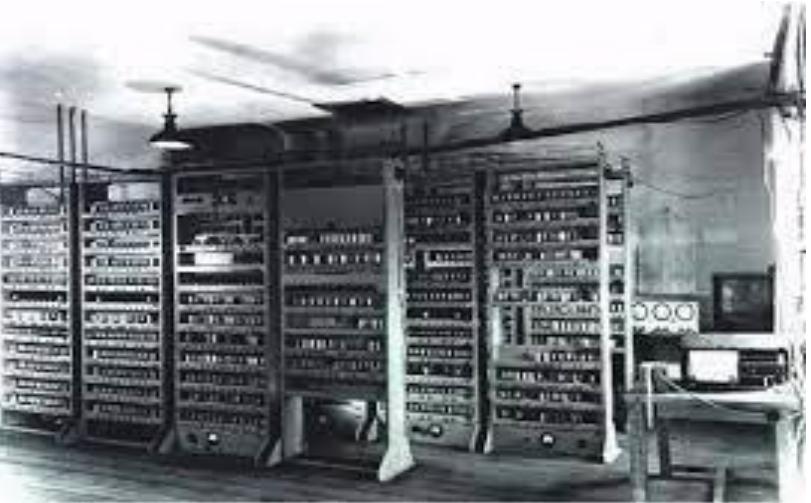
From <https://www.blueocean-china.net/faq3/241.html>

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

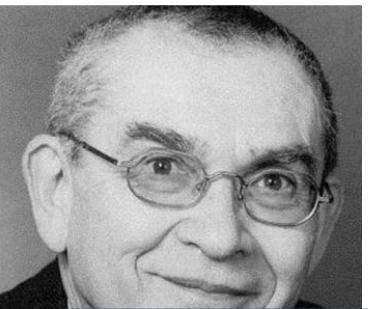
Proposal of Quantum Computing



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

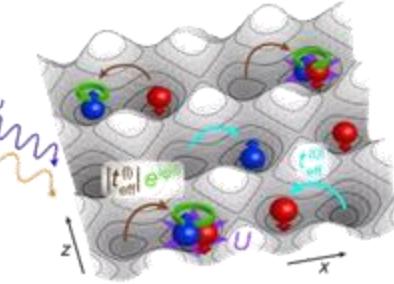


Richard Feynman

- 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

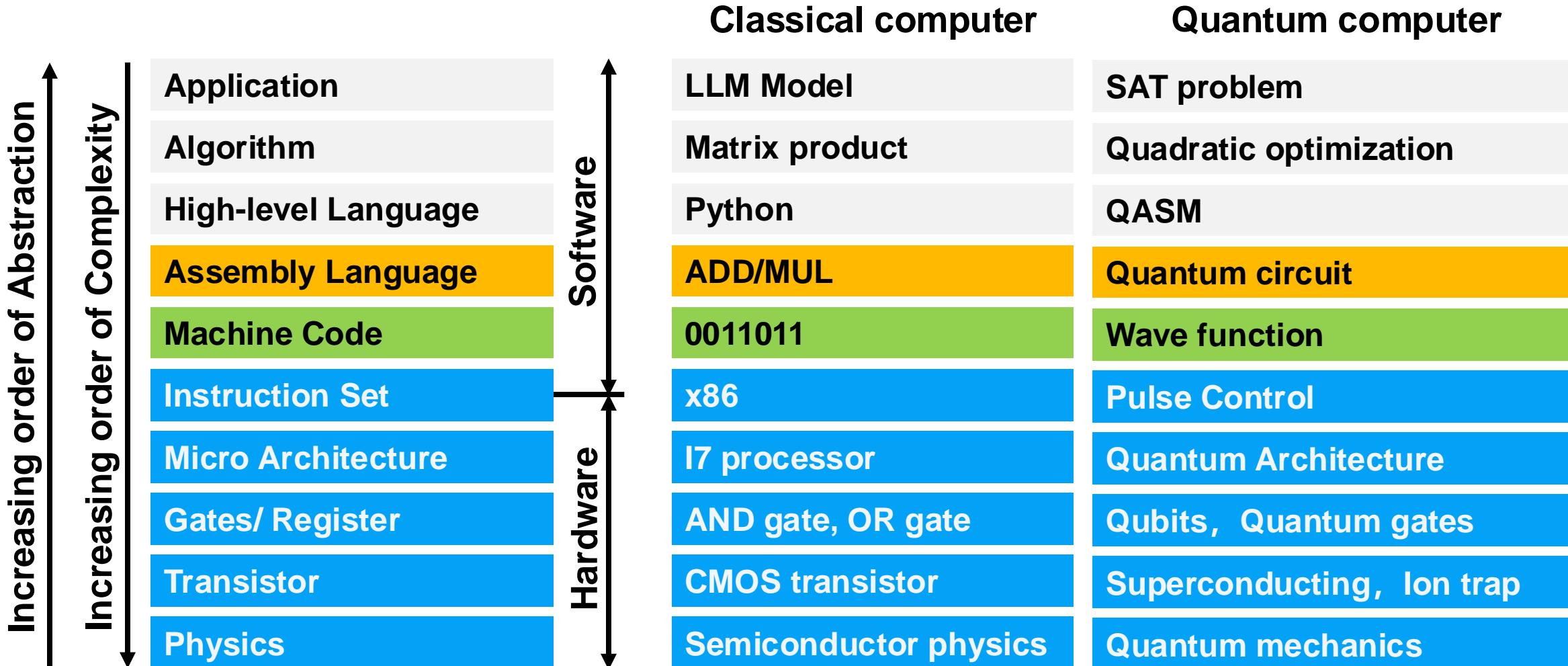


Factor

Application

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

Quantum Computing Architecture



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

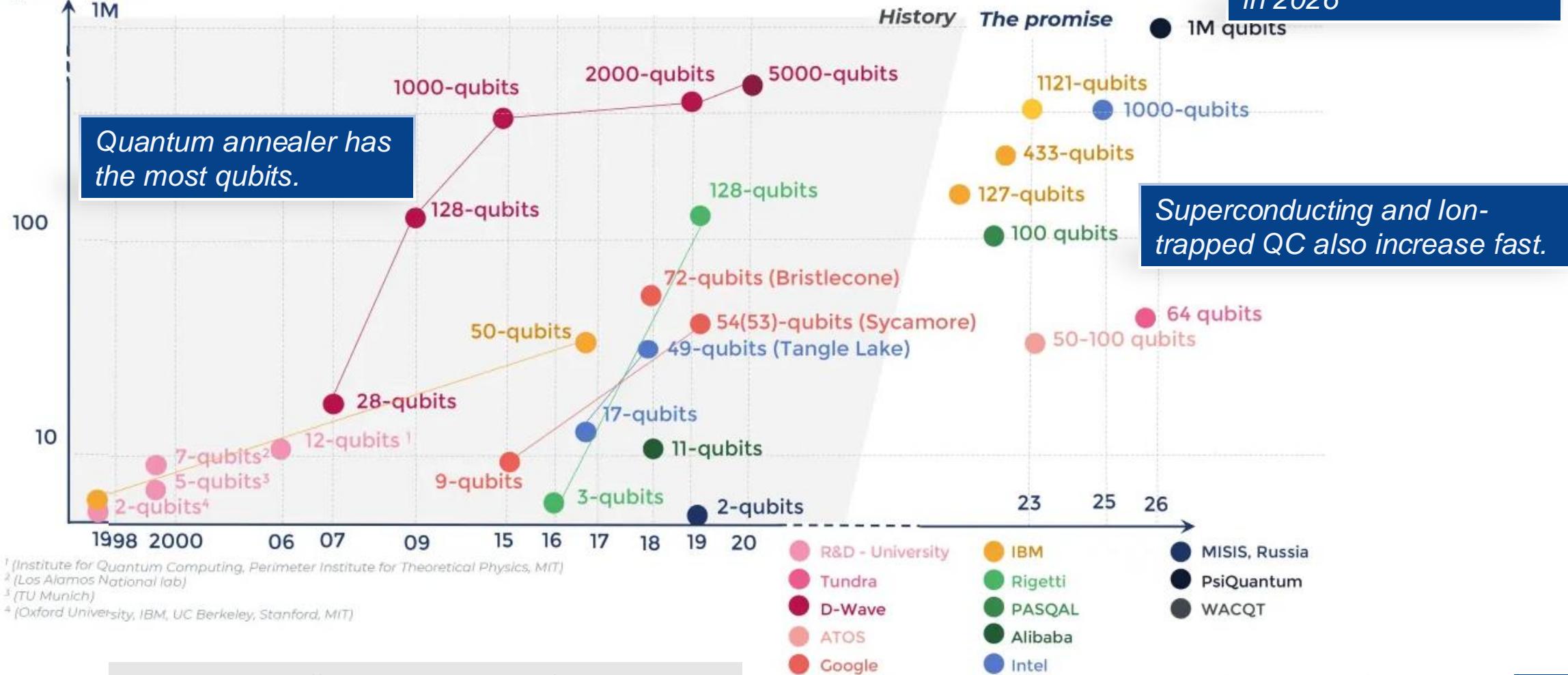
Classical and quantum computing has similar architecture

Development of Quantum Computer



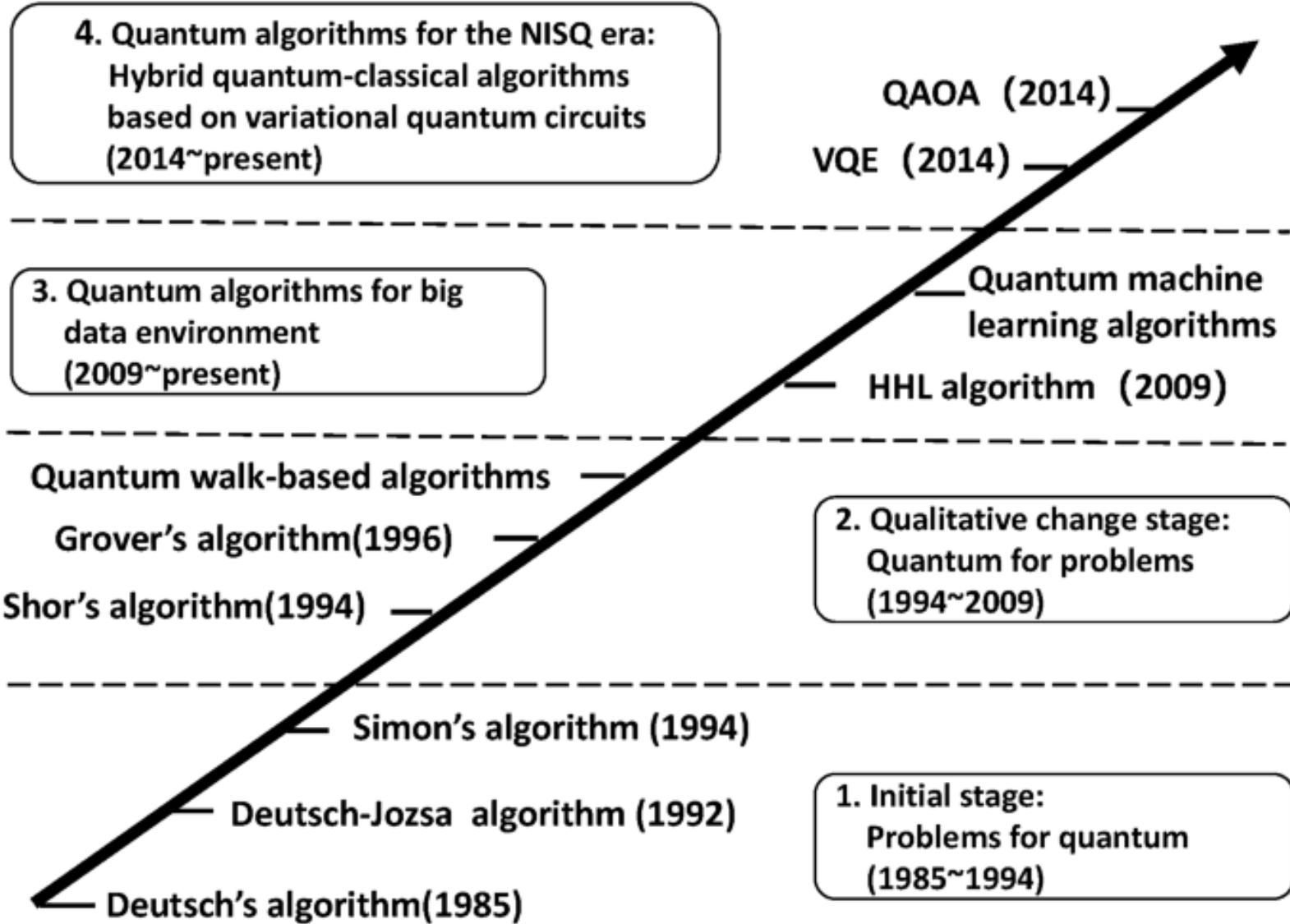
浙江大学
ZHEJIANG UNIVERSITY

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

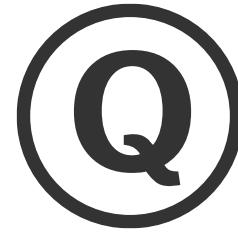


- Background Knowledge
- **Mathematical Model of Quantum Computing**
- Janus (Taiyuan) Quantum Cloud Platform
- Installing JanusQ



浙江大學
ZHEJIANG UNIVERSITY

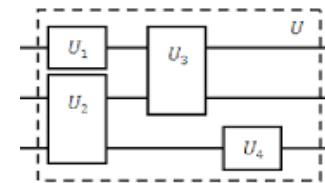
Mathematical Model of Quantum Computing



Qubits



Quantum Evolution



Quantum Circuit

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** $|\varphi\rangle$.

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

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

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

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

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

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** $|\varphi\rangle$.

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

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

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

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

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

Multiple qubits

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

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

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

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

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



Unitary matrix

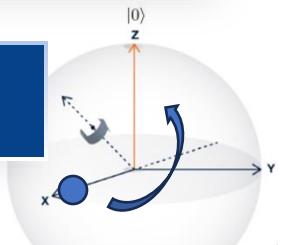
A quantum evolution caused by **quantum gates** is represented as a **unitary matrix (unitary)**, which is a square matrix whose conjugate transpose is its inverse.

$$UU^\dagger = I$$

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

Bloch sphere

X+Z axes

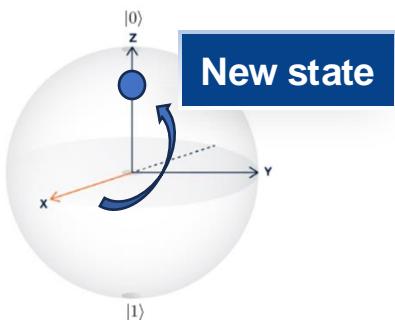


State

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

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

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



New state

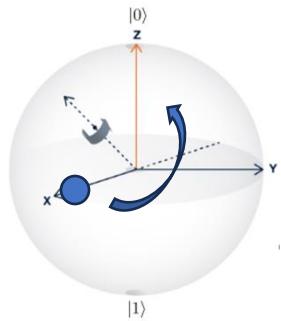
Unitary matrix

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.

$$UU^\dagger = I$$

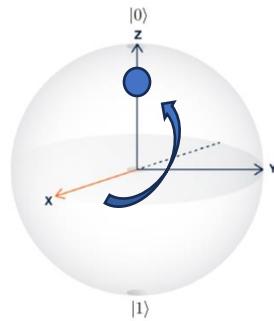
The evolution of qubit state $|\varphi\rangle$ is represented as:

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



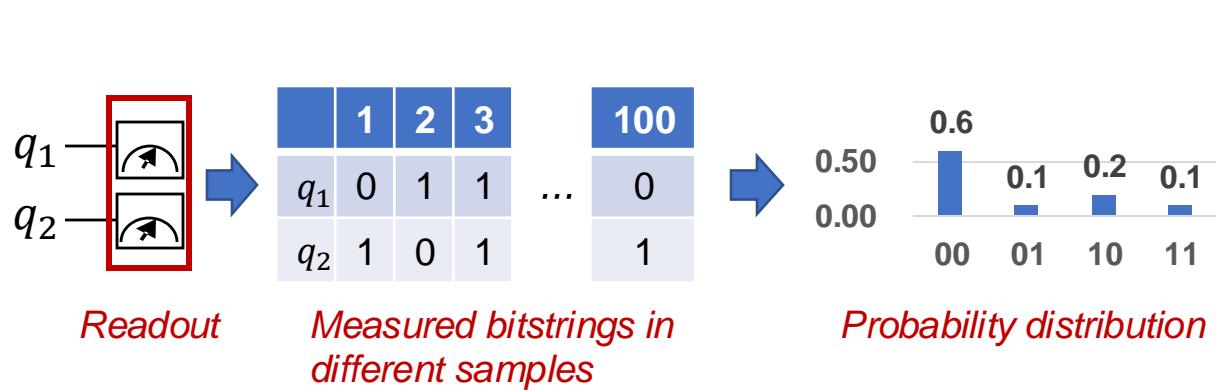
$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

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.





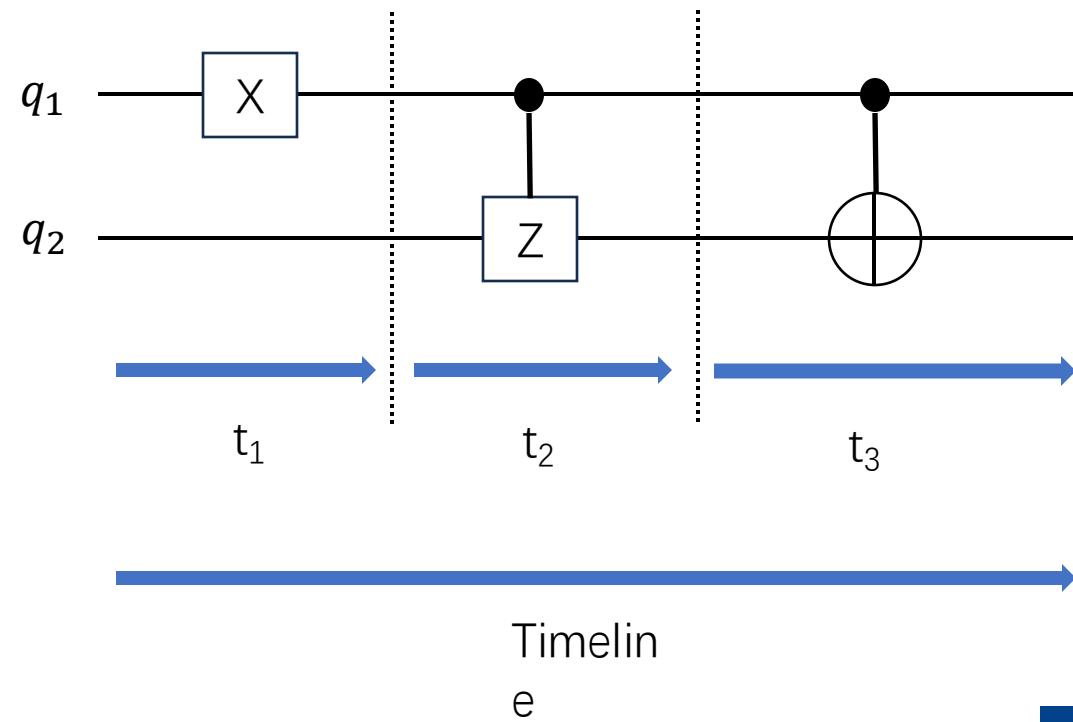
Quantum gates

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

Qubit timeline

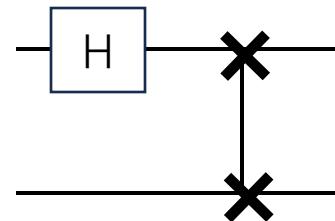
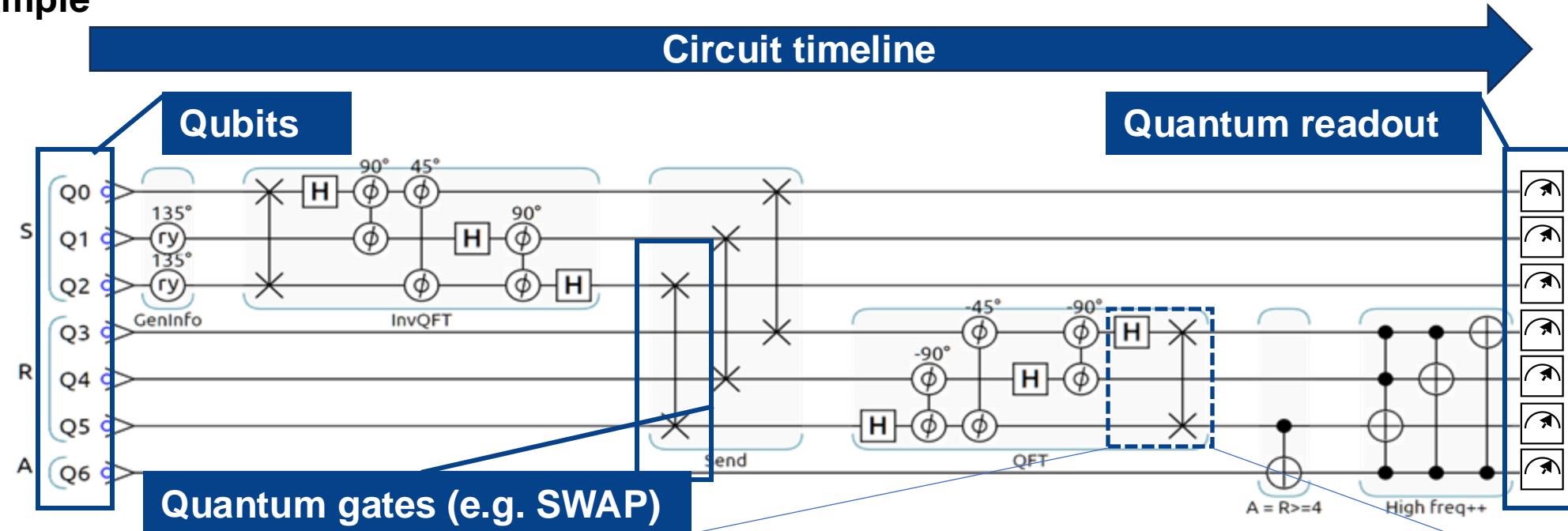
Each line in a quantum circuit represents a **qubit**. The quantum gate in a line is applied on the same qubits **from left to right in time direction**.



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

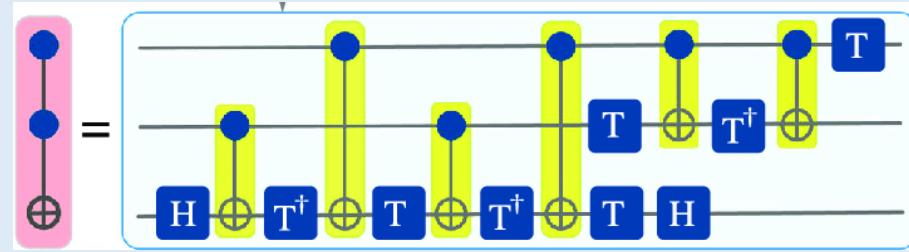
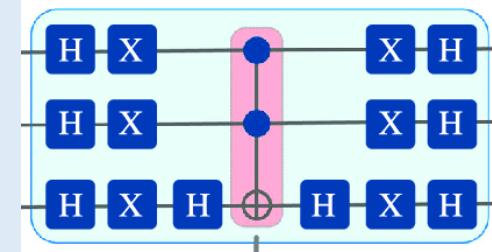


On superconducting quantum computer

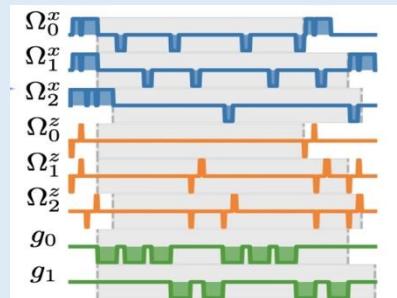
Step 1. Circuit statement

```
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

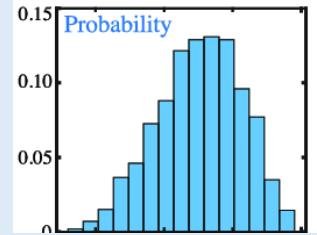


Quantum device

Step 4. Result processing

Error mitigation

Visualization



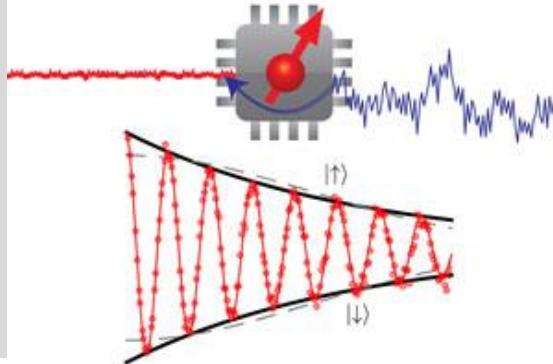
Probability distribution

Challenge in Quantum Computing



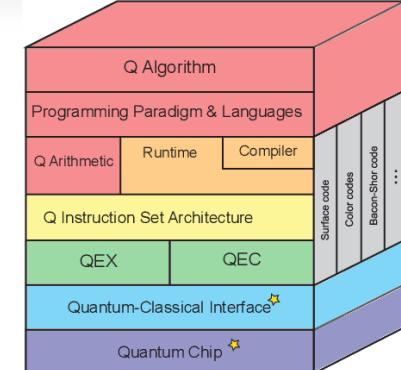
Noise

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

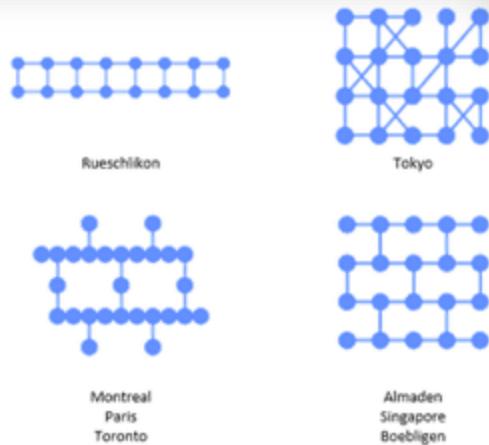


Instructions

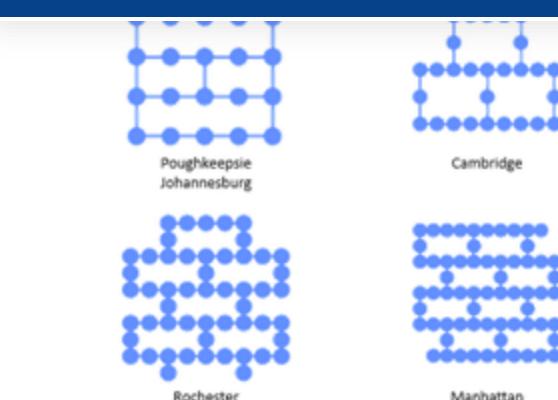
- Multi-level compilation
- Micro-architecture instructions
- Quantum-classical communication



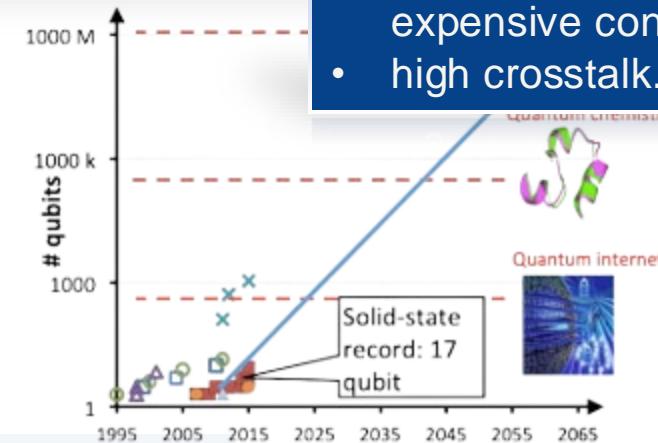
Topology



Locality in communication



Scalability



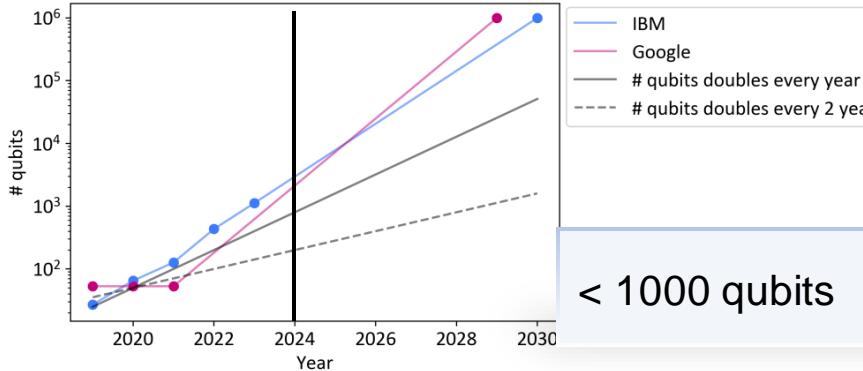
- poor fabrication techniques
- expensive control systems
- high crosstalk.

Constrained by physical implementation

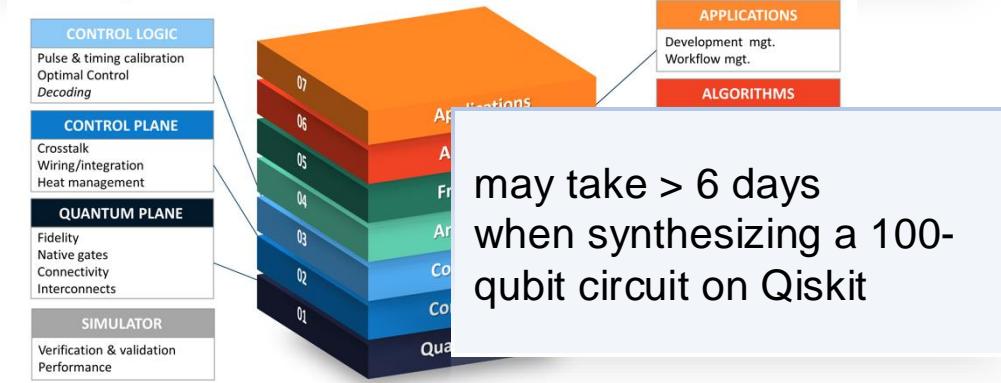
Results of Challenges



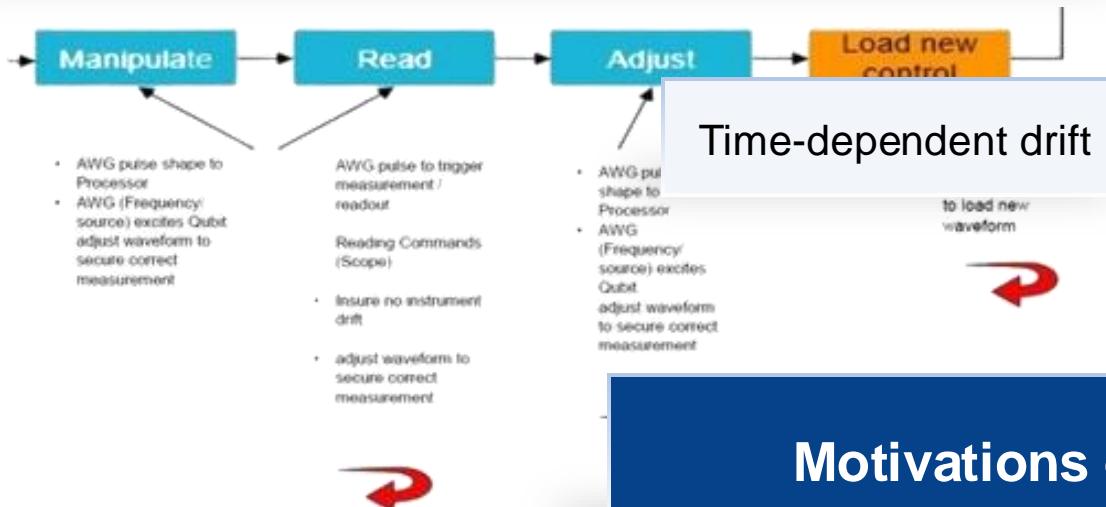
Limited Hardware Resource



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



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

Janus (Taiyuan) Quantum Cloud Platform





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.

Development History of Janus Cloud



浙江大學
ZHEJIANG UNIVERSITY

Four updates since July 2023

The image shows a website for "Taiyuan Quantum". The header features a logo with the word "TAIYUAN" and a blue circular icon. The navigation menu includes "Home", "Resource", "Project", "Document", "AboutUs", and a search bar with a magnifying glass icon. Below the header, there is a large, abstract image of a quantum circuit or network. On the left side of the page, there is a sidebar with the text "Taiyuan Quantum" and "Provide real-time quantum computing services", along with a blue "Get Start" button. The main content area contains the following text:

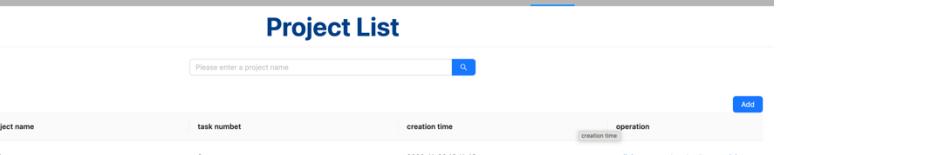
July 2023

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

The screenshot shows the Taiyuan website's "Document" page. The top navigation bar includes links for Home, Resource, Project, Document (which is underlined), and AboutUs, along with language selection (Chinese) and a user icon. A sidebar on the left contains links for Quicode API, User guide, and Overview. The main content area features a section titled "Quicode API" with a sub-section "1. Circuit Commands". It describes preparation work before performing quantum operations, applying for and initializing quantum circuits. Below this, there are two examples: "new(qubit_number, name)" and "reset(qubit_number)". A large, semi-transparent rectangular box is overlaid on the right side of the page, containing the text "August 2023" in large bold letters and "1. Update the document" below it.

August 2023

1. Update the document

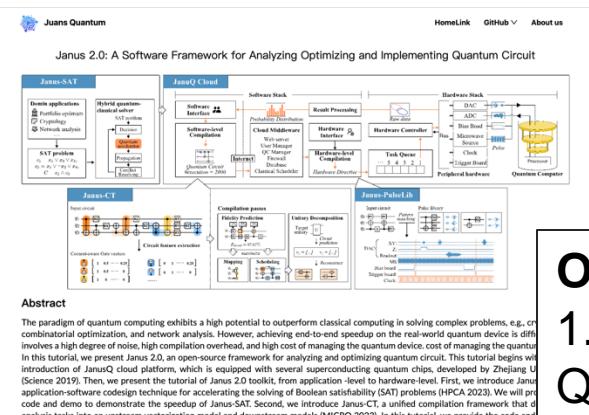


The screenshot shows a project management application interface. At the top left is the logo 'TAYUAN'. The top navigation bar includes links for Home, Resource, Project (which is underlined), Document, AboutUs, and a user icon. Below the navigation is a search bar with placeholder text 'Please enter a project name' and a blue search button. A large blue 'Add' button is positioned on the right side of the header. The main content area is titled 'Project List' and contains a table with 7 rows of data. The columns are: serial number, project name, task number, creation time, and operation. The 'operation' column includes links for edit item, view details, and delete. To the right of the table, a large box highlights the addition of 'multi-level project management' in September 2023.

serial number	project name	task number	creation time	operation
1	336	0	2023-11-08 16:11:46	edit item view details delete
2	8899	0	2023-11-08 16:11:51	
3	test20231027	0	2023-10-27 17:10:51	
4	20230923	0	2023-09-23 20:09	
5	0923-1	0	2023-09-23 20:09	
6	23	0	2023-09-23 20:09	
7	通信	0	2023-09-16 14:09	

September 2023

1. Added multi-level project management.



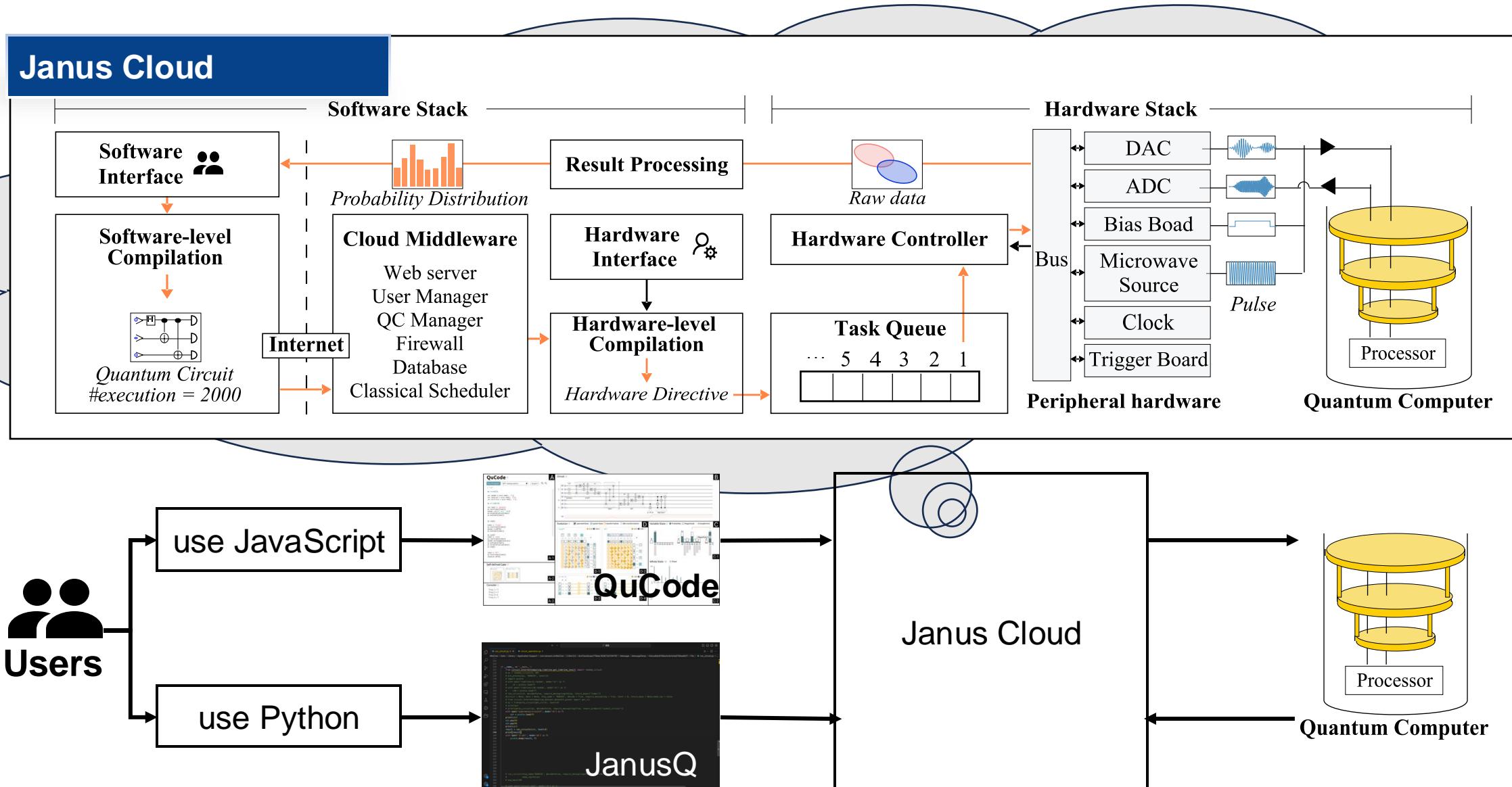
September 2023

1. Added multi-level project management.

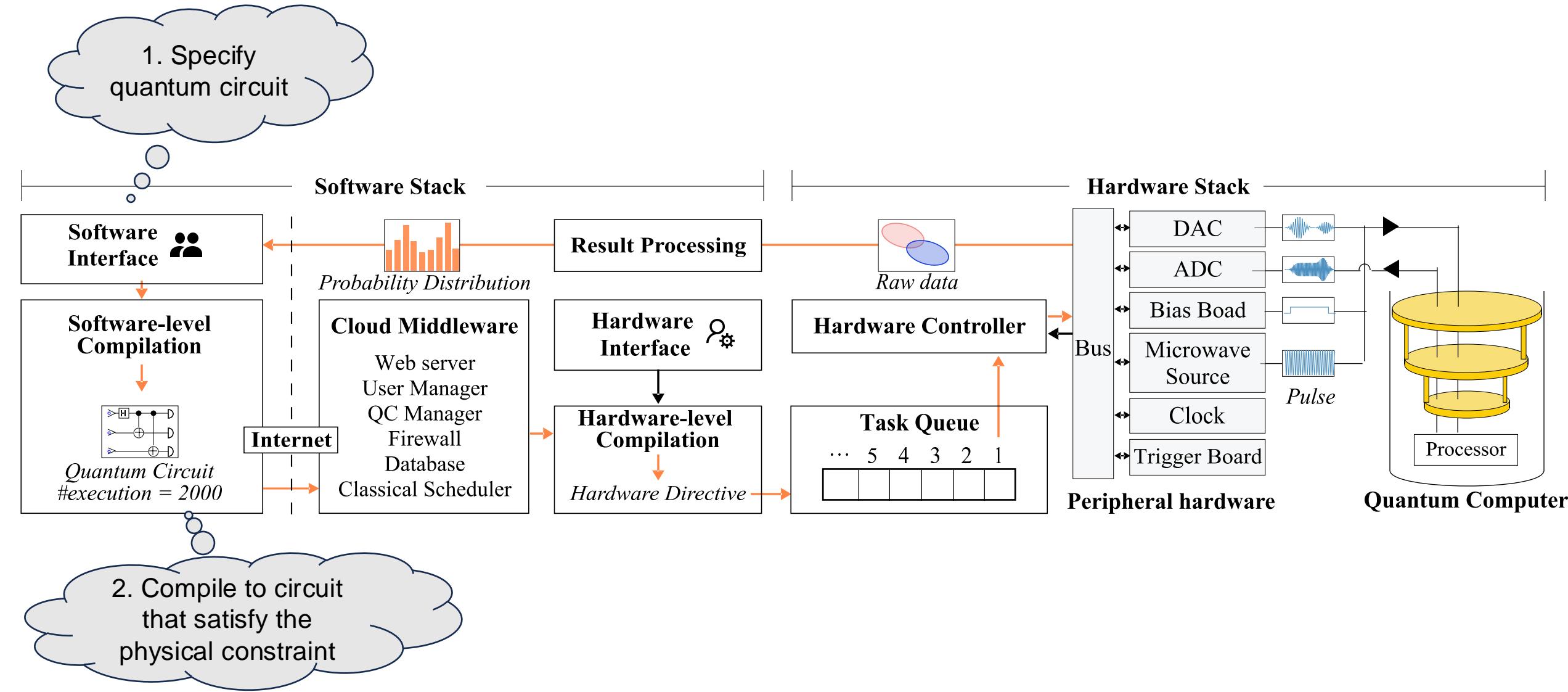
October 2023

1. Enable the online QuCT and HyQSAT

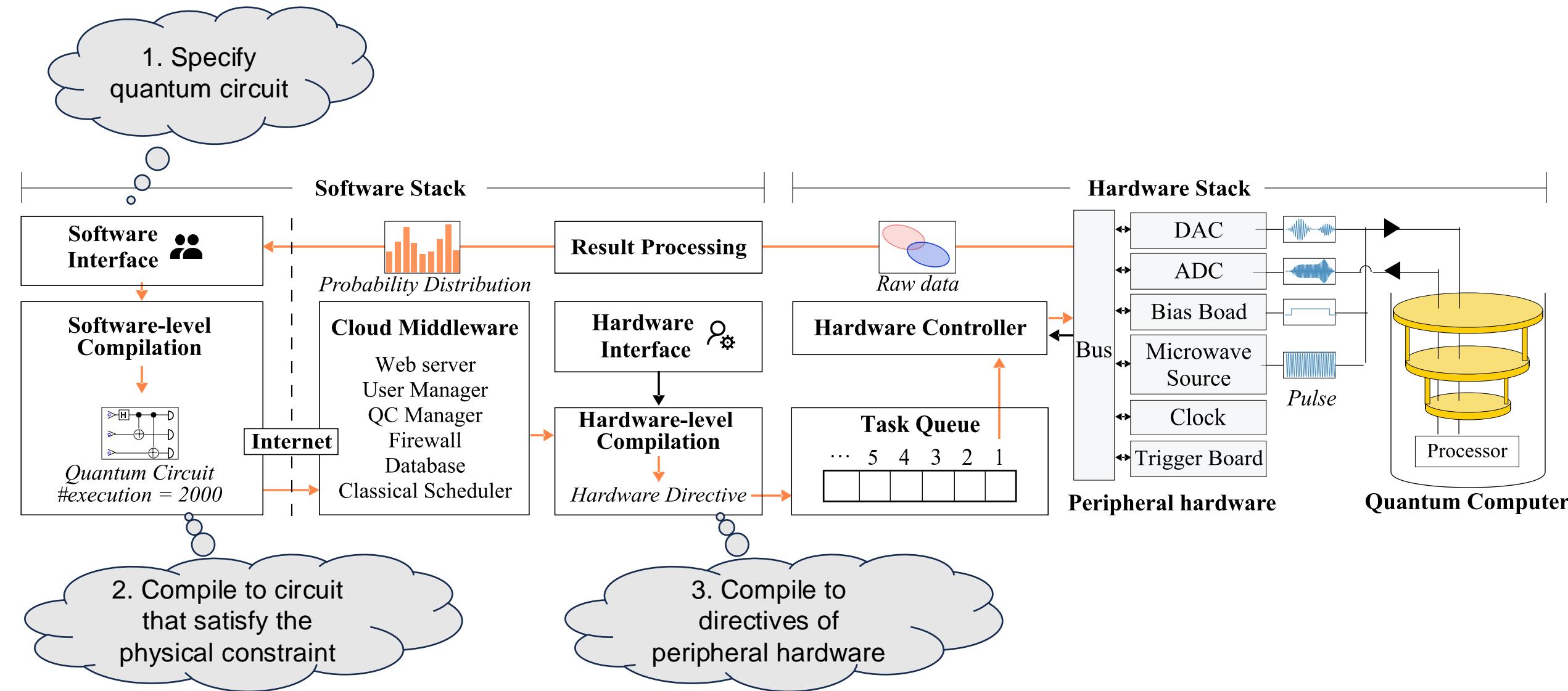
What we can do on Janus Platform



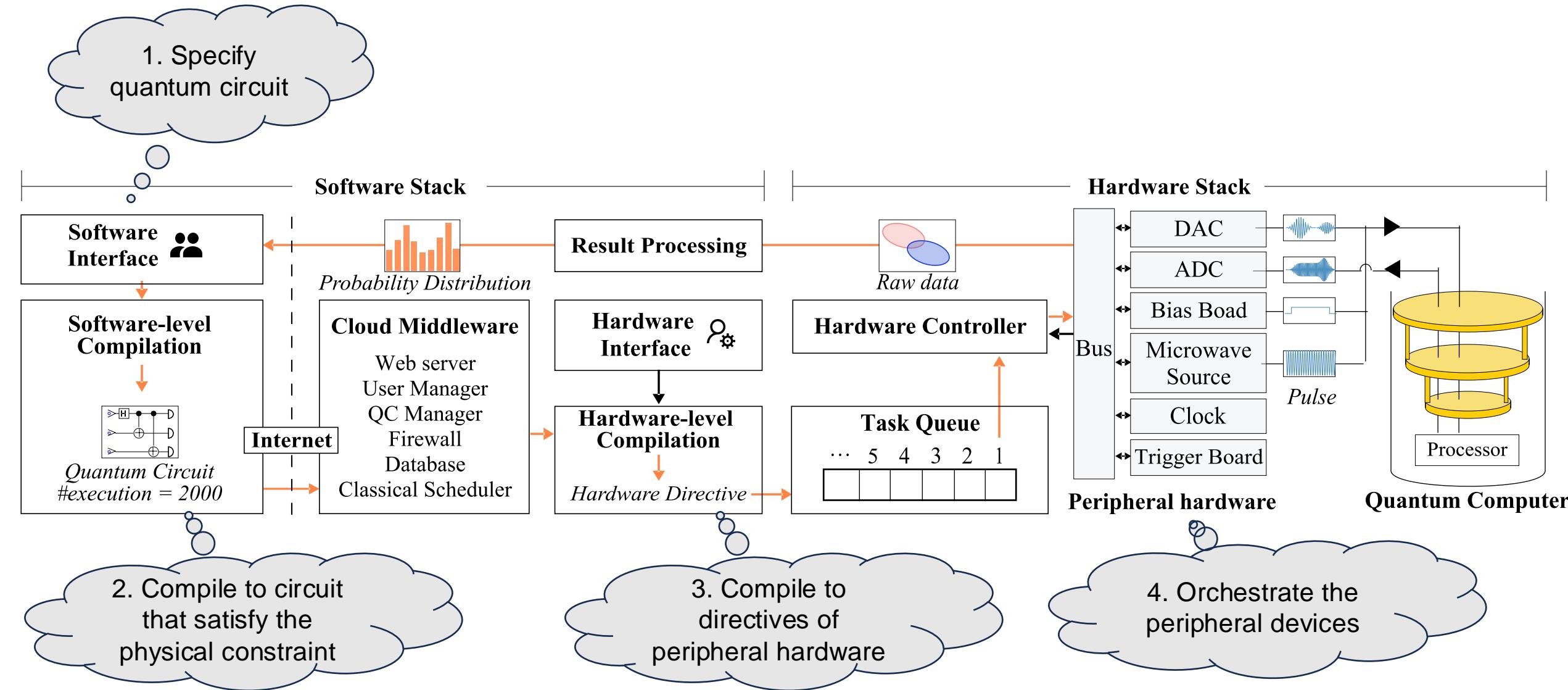
What we can do on Janus Platform



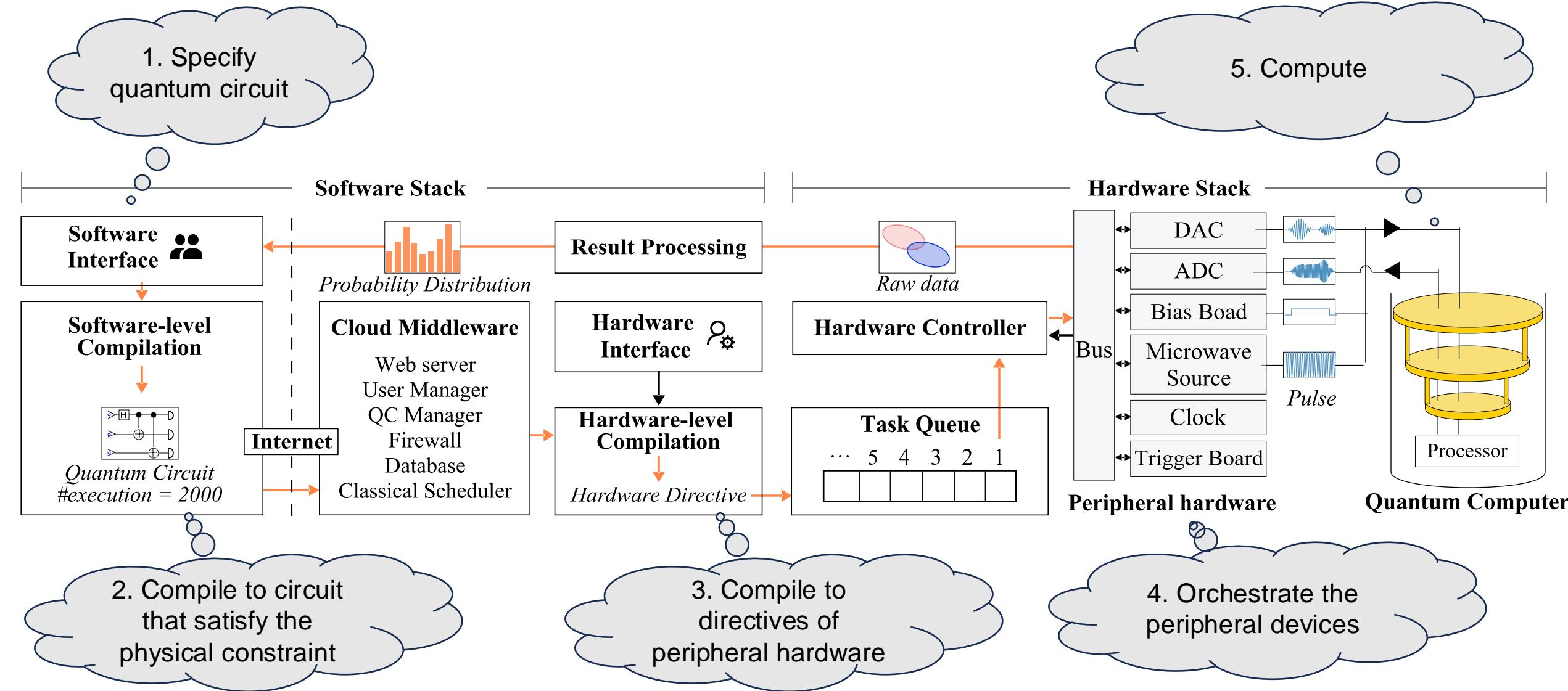
What we can do on Janus Platform



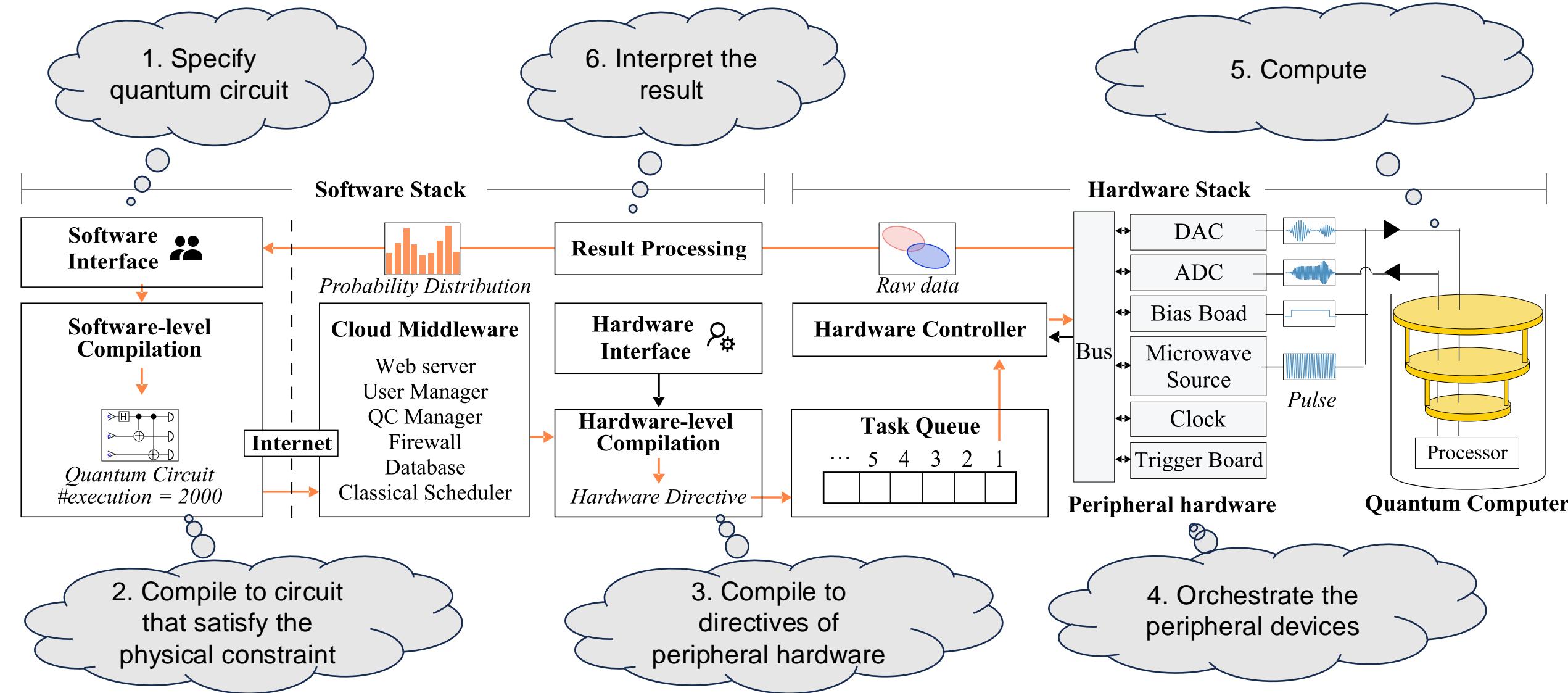
What we can do on Janus Platform



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

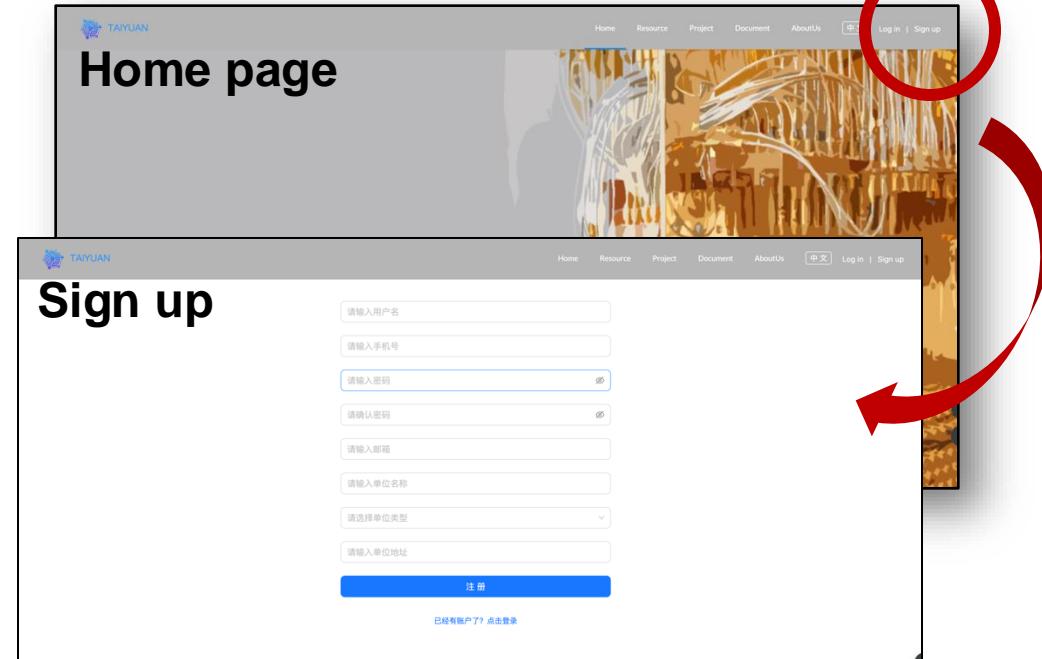
Open <http://janusq.zju.edu.cn/home>



QR code

Sign up

Click sign up button



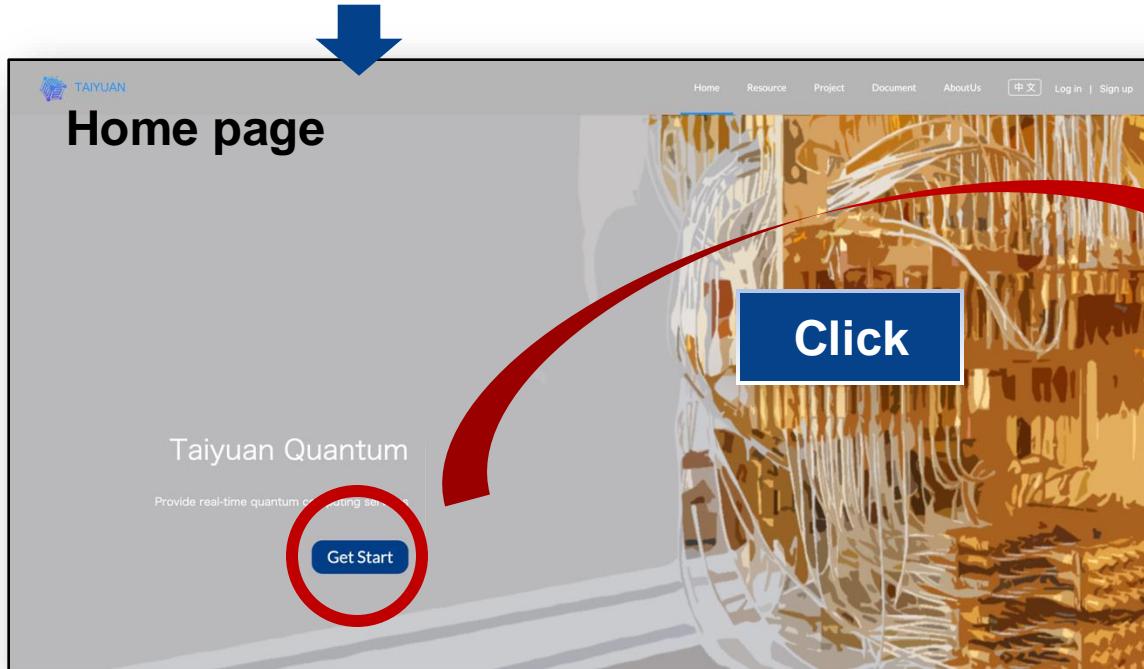
- After signing up, you will get an API key for task submission.
- **The quantum computer can be accessed on the website during the tutorial !!**

Running Program On the Website

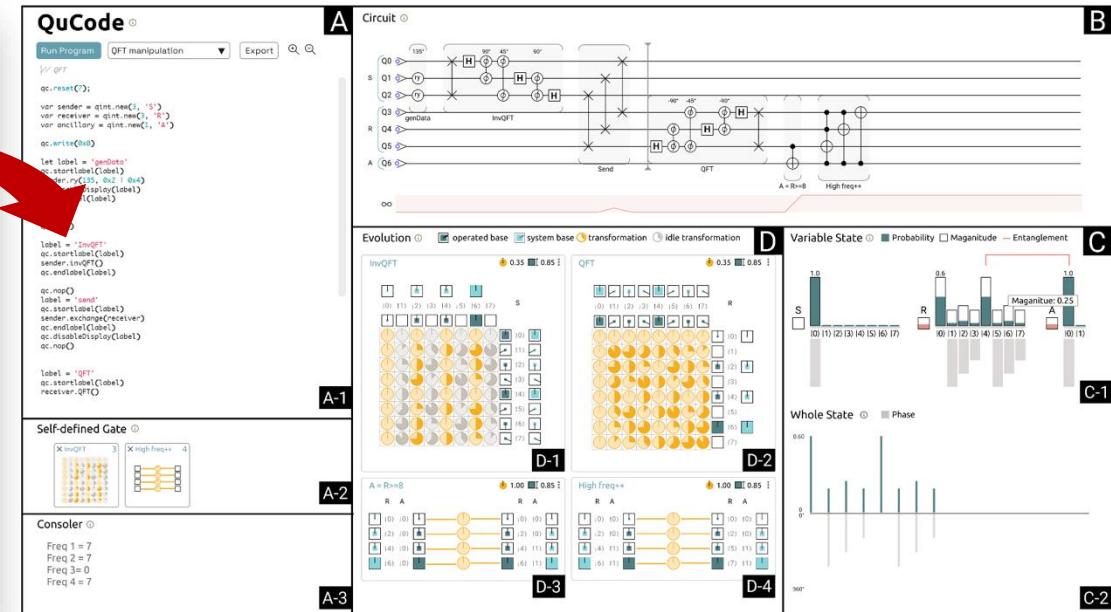


Submitting on website

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



Programming on the Code editor

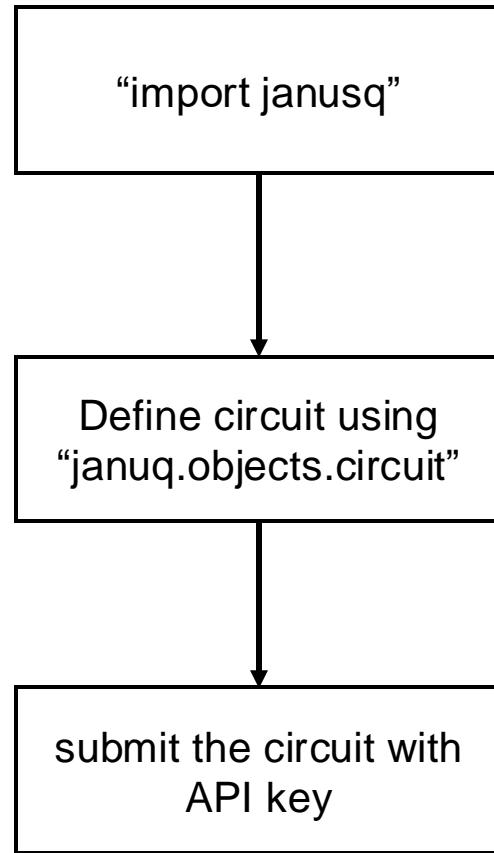


- The quantum computer can be accessed on the website during the tutorial !!

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

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()
```

Circuit reuse

Console

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

A
B

Quantum circuit view

Evolution view

Evolution

- operated base
- system base
- transformation
- idle transformation

	InvQFT	QFT
0.35	0.85	0.35
0.85	0.35	0.85

D-1

D-2

State view

Variable State

- Probability
- Magnitude
- Entanglement

C-1

C-2

HPCA 2025

41

QuCode: Online Quantum Program Editor



浙江大學
ZHEJIANG UNIVERSITY

Program editor

A

Circuit

The circuit diagram shows a sequence of quantum operations. It starts with a `qc.reset()`, followed by `qc.write(0x0)`. Then there are two `qc.nop()` operations. The next part is labeled `InvQFT`, which consists of several gates: `ry(135)`, `H`, `phase(90)`, `phase(45)`, and `H`. This is followed by a `Send` operation, another `QFT`, and a condition `A = R >= 8` leading to `High freq++`.

B

Evolution

operated base

system base

transformation

idle transformation

D

InvQFT

0.35

0.85

S

(0) |0> (1) |1> (2) |2> (3) |3> (4) |4> (5) |5> (6) |6> (7) |7>

QFT

0.35

0.85

R

(0) |0> (1) |1> (2) |2> (3) |3> (4) |4> (5) |5> (6) |6> (7) |7>

Variable State

Probability

Maganitude

Entanglement

C

S

R

A

C-1

A = R >= 8

1.00

0.85

R A

(0) |0> (1) |0> (2) |0> (3) |0> (4) |0> (5) |0> (6) |0> (7) |0>

High freq++

1.00

0.85

R A

(0) |0> (1) |0> (2) |0> (3) |0> (4) |0> (5) |0> (6) |0> (7) |0>

Whole State

Phase

C-2

A-1

A-2

A-3

A-4

D-1

D-2

D-3

D-4

C-1

C-2

D-1

<div data-bbox="325 635 33

QuCode: Online Quantum Program Editor



浙江大學
ZHEJIANG UNIVERSITY

Program editor

A

Circuit ①

Run Program QFT manipulation Export 🔍 🔍

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()
```

B

Provide 14 examples:

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

A-1

A-2

A-3

D

Variable State ①

Probability Magnitude Entanglement

QFT

0.35 0.85

S R A

Whole State ① Phase

0.60 0.00

0° 360°

D-1

D-2

D-3

D-4

C

C-1

C-2

D-1: S state visualization showing probability distribution across 8 qubits.

D-2: R state visualization showing probability distribution across 8 qubits.

D-3: A state visualization showing probability distribution across 8 qubits.

D-4: High freq++ state visualization showing probability distribution across 8 qubits.

B: Quantum circuit diagram illustrating a sequence of operations: InvQFT, Send, QFT, and High freq++. The circuit involves multiple qubits and various quantum gates like H, P, and CNOT.

QuCode: Online Quantum Program Editor



浙江大學
ZHEJIANG UNIVERSITY

Circuit View

The circuit diagram illustrates a sequence of operations on four qubits (Q0 to Q3). It includes an initial state preparation section (Q0 to Q3), followed by an Inverse Quantum Fourier Transform (InvQFT) on Q0 to Q3, and a final section involving Hadamard gates and controlled operations. A red oval highlights a grouping of operations, and a red box labeled "Purity" indicates a measurement or analysis step.

QuCode

```
// 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-1

Self-defined Gate

A-2

A-3

B

C

D

D-1

D-2

D-3

D-4

C-1

C-2

Evolution

Variable State

Whole State

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

B

Evolution

InvQFT

QFT

State View

R

S

Whole State

A-1

D-1

A-2

D-2

A-3

D-3

A-4

D-4

partial trace

The whole state

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

Evolution View

D
C

INVQFT

QFT

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

C-1
C-2

1. Visualize the unitary
2. Visualize the change of qubit states

Outline of Presentation



- 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: ≥ 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 janusq3



Step 3. start image

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

Step 5. Or use Jupyter Lab to connect the docker

<http://localhost:9999/lab>

The password of the docker is "" (empty).

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)

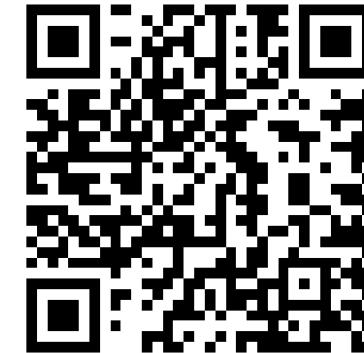
```
cd ./JanusQ  
conda create -n janusq python=3.10  
conda activate janusq  
pip install -r requirements.txt
```

Step 3. compile HyQSAT

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

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

- 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!