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/

Presenter

Tianyao Chu

tianyao_chu@zju.edu.cn

Tianyao Chu is a first-year PhD student at the College of Computer Science, Zhejiang University. He is interested in distributed quantum computing and quantum network system. He is currently working on designing a quantum network on chip based on enhanced QST.

Presenter

Kaiwen Zhou

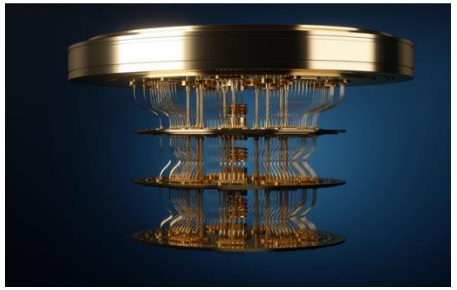
@zju.edu.cn

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.

Background Knowledge

Development of
Classical Computing

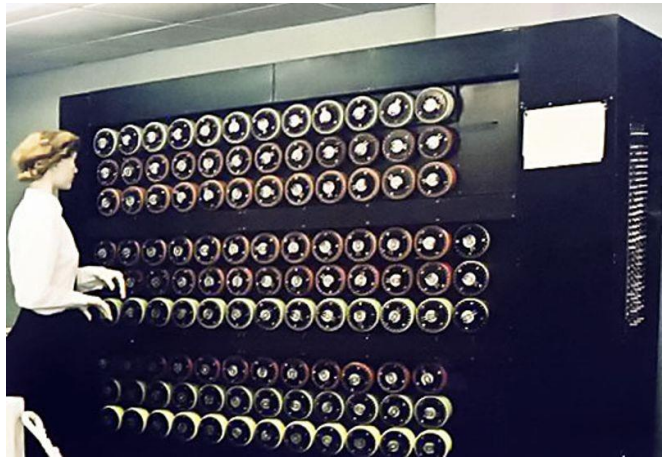
Motivation of
Quantum Computing



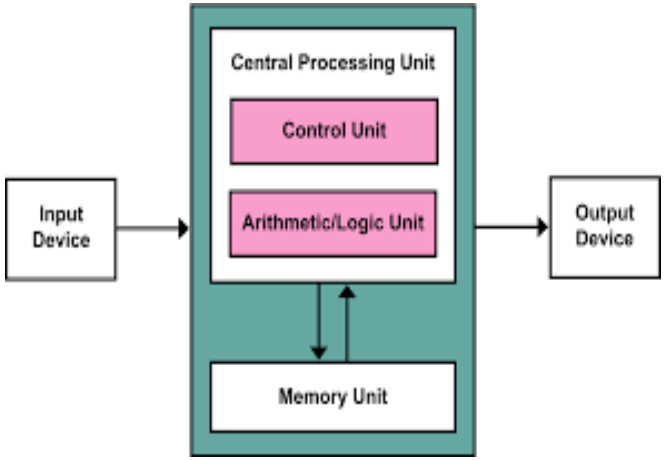
Development of
Quantum Computing

Development Of Classical Computing

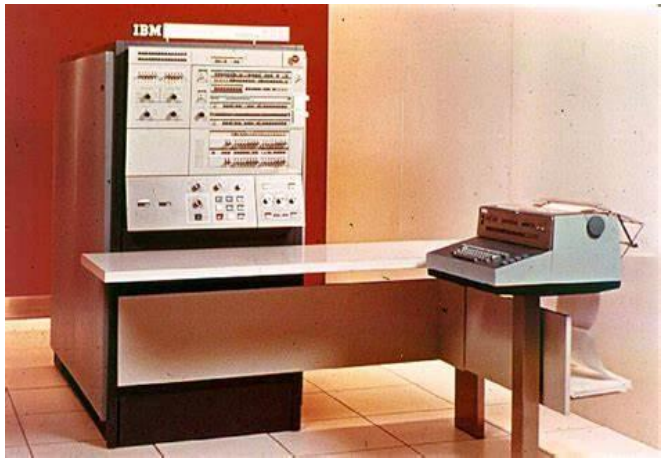
Domain-specific calculator （1939）



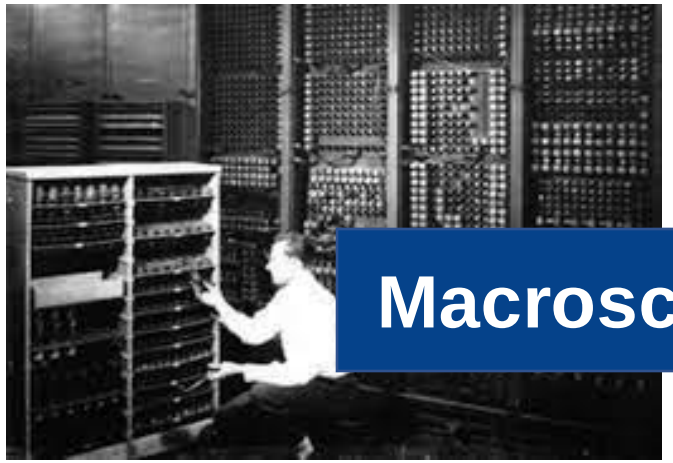
Von Neumann architecture (1947)



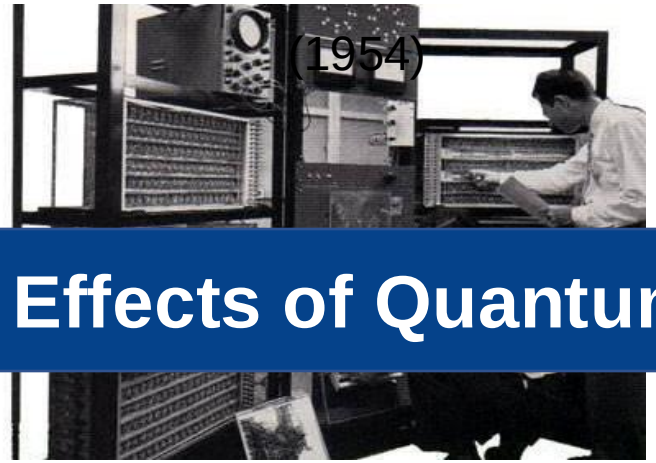
IBM360 Integrated Circuit (1964)



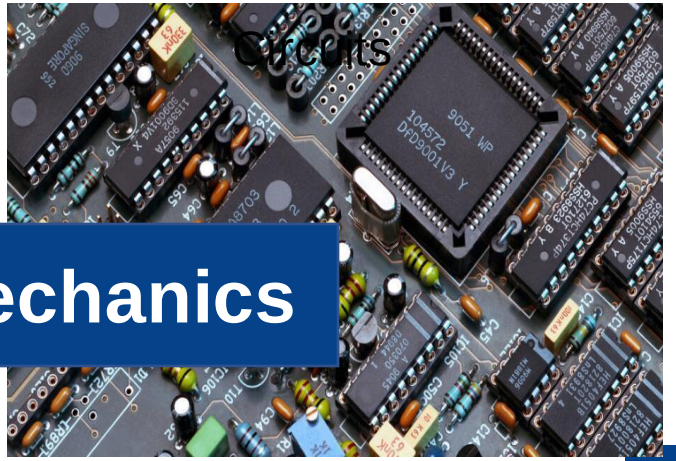
Vacuum Tube Computer ENIAC (1942)



Transistor Computer TRADIC (1954)



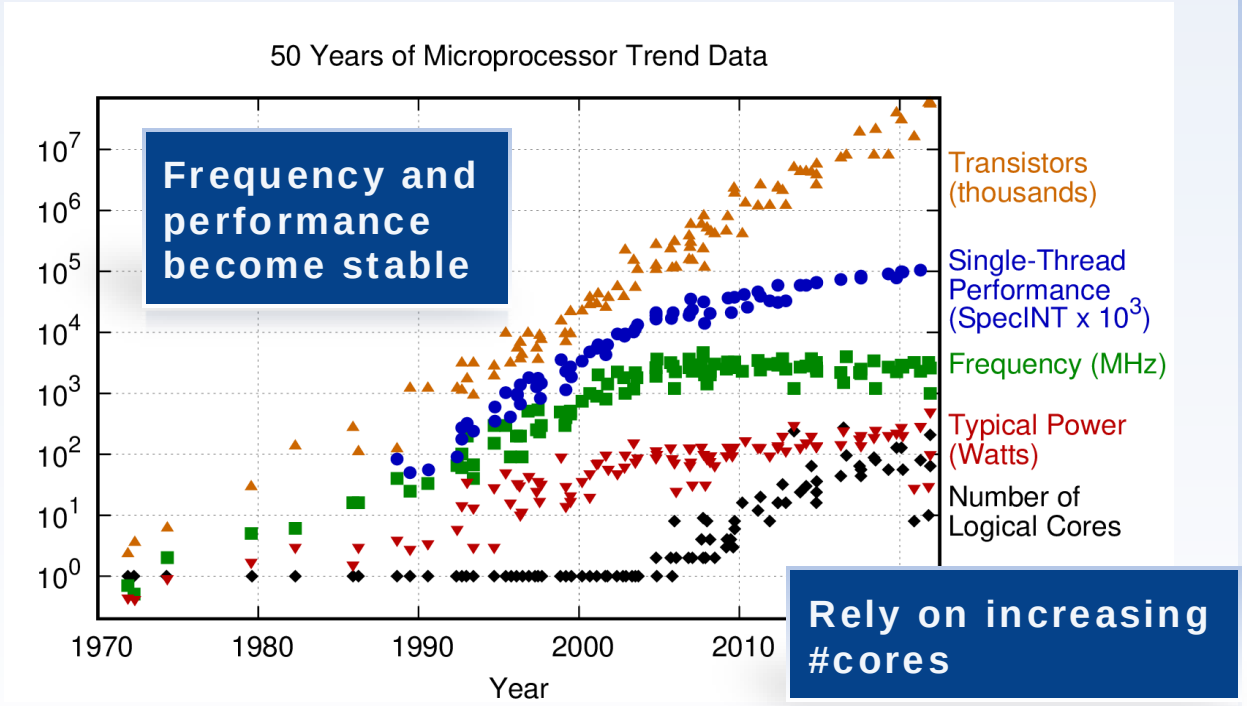
Large Scale Integrated Circuits



Macroscopic Effects of Quantum Mechanics

Motivation Of Quantum Computing

Computation barrier

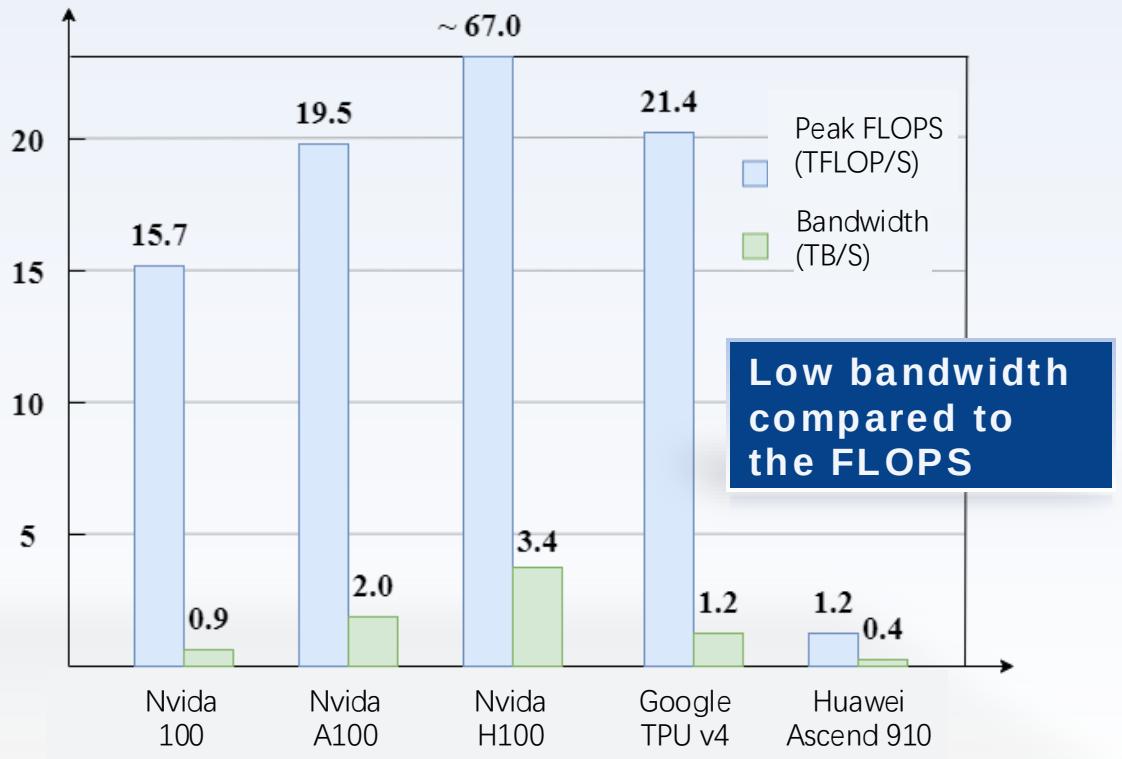


Original dalta up to the year 2010 colecled and plolted by M. Horowitz, F. Labonle, 0. Shacham, K. Olukolun, L. Hammond, and c. BalienNew plot and dala collected for 2010-2021 by K.Rupp

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

Motivation Of Quantum Computing

Storage barrier

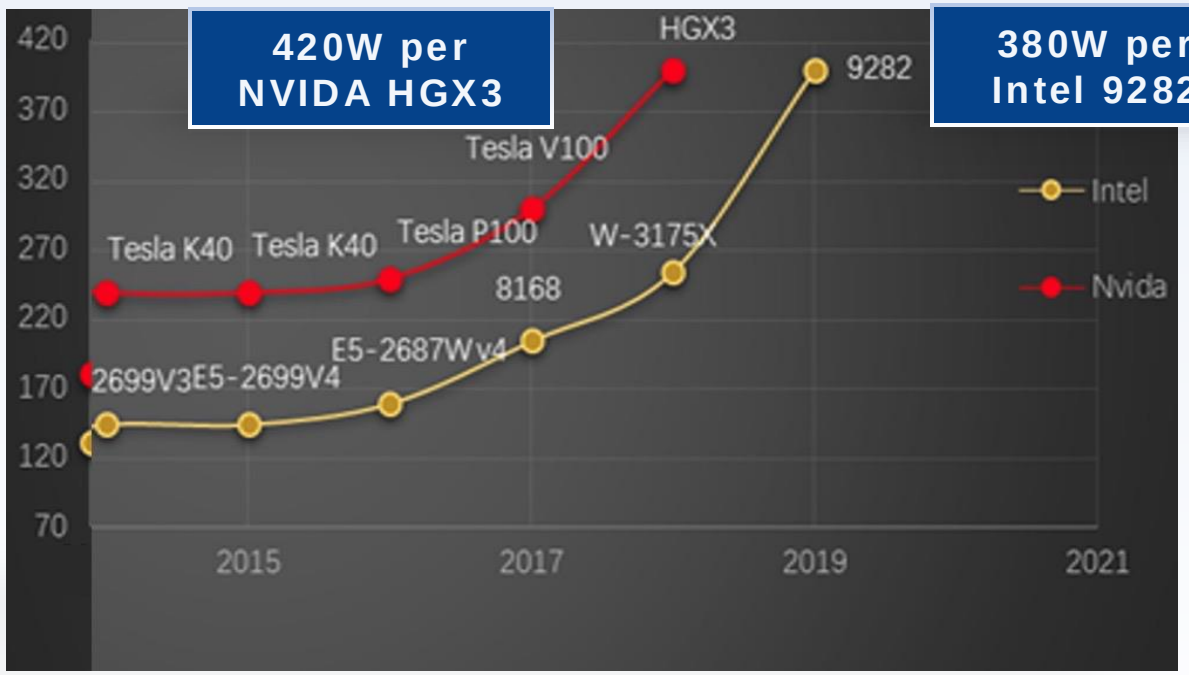


- Computation power and bandwidth is not matched.
- Latency of memory access is high, limiting CPU performance.
- Quantum computing is in memory.
- Non-von Neumann architectures.

Motivation Of Quantum Computing

Power wall

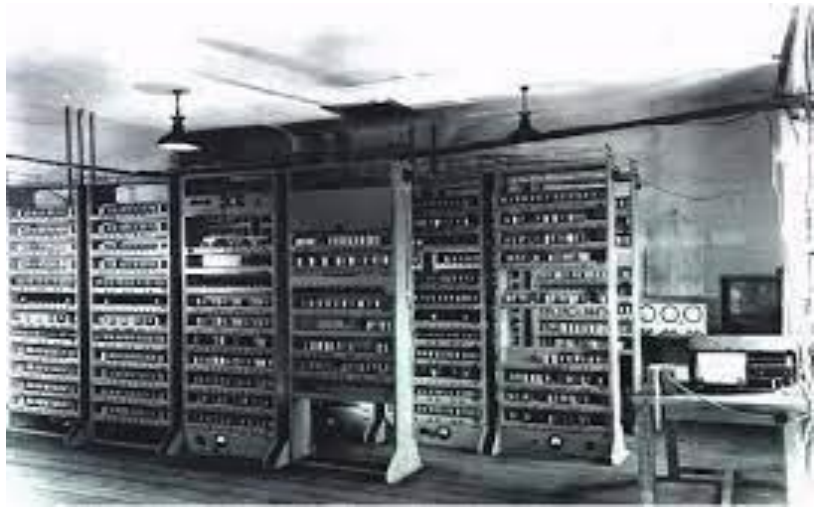
Thermal design power exponentially increases.

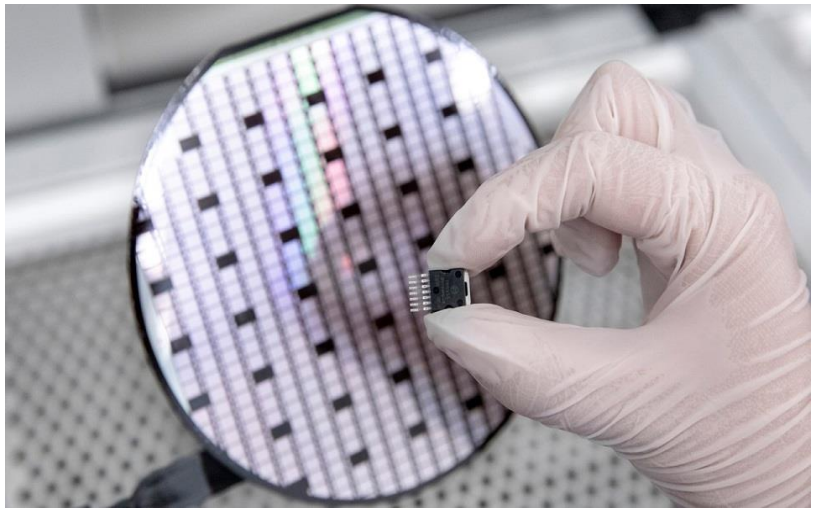


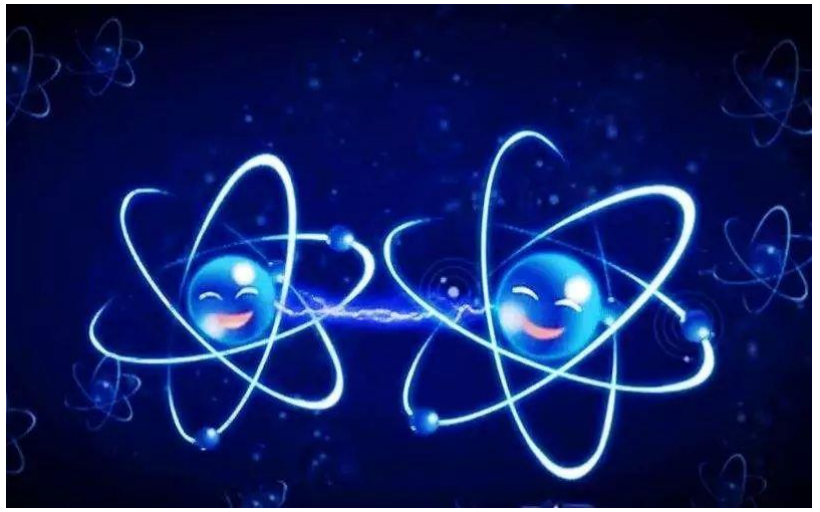
From https://www.blueocean-china.net/faq3/241.html

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

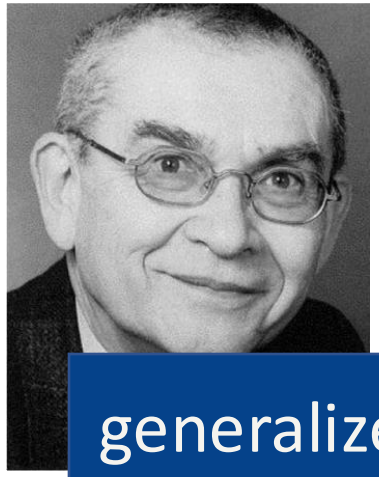
Proposal of Quantum Computing







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



Yuri Manin

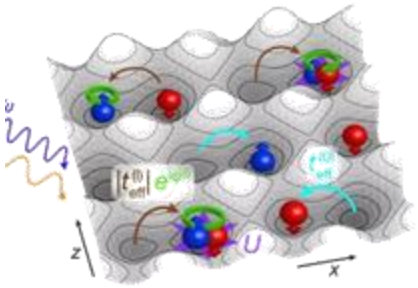
- Algebraic Geometry
- Discrete Geometry (Diophantine Geometry)
- Manin (1980) and Feynman

Richard Feynman

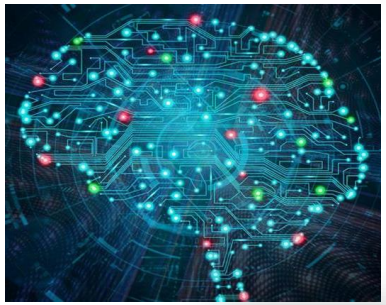
- Feynman Path Integral, Feynman Diagram, Feynman Parton Model
- Quantum Electrodynamics, Nobel Prize in Physics

generalize to other domains such as the database search and cryptography

Basic Concepts of Quantum Computing

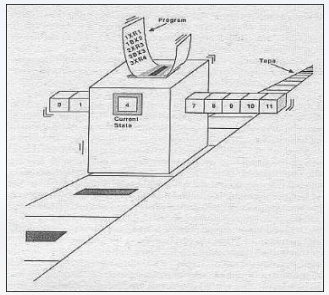
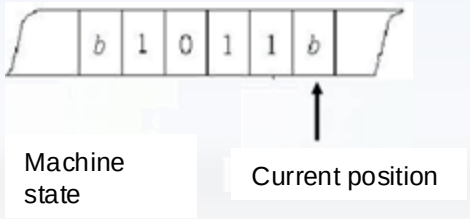
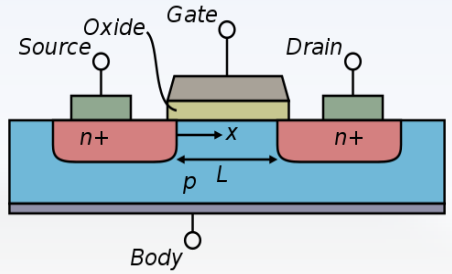
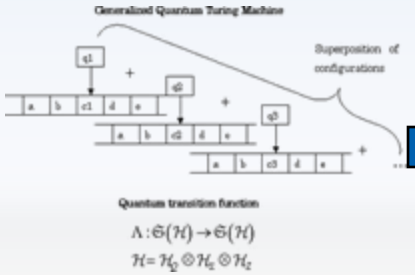
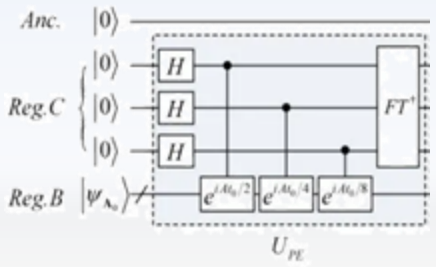
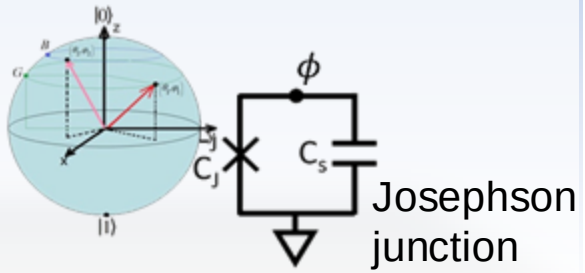


Physical Simulation

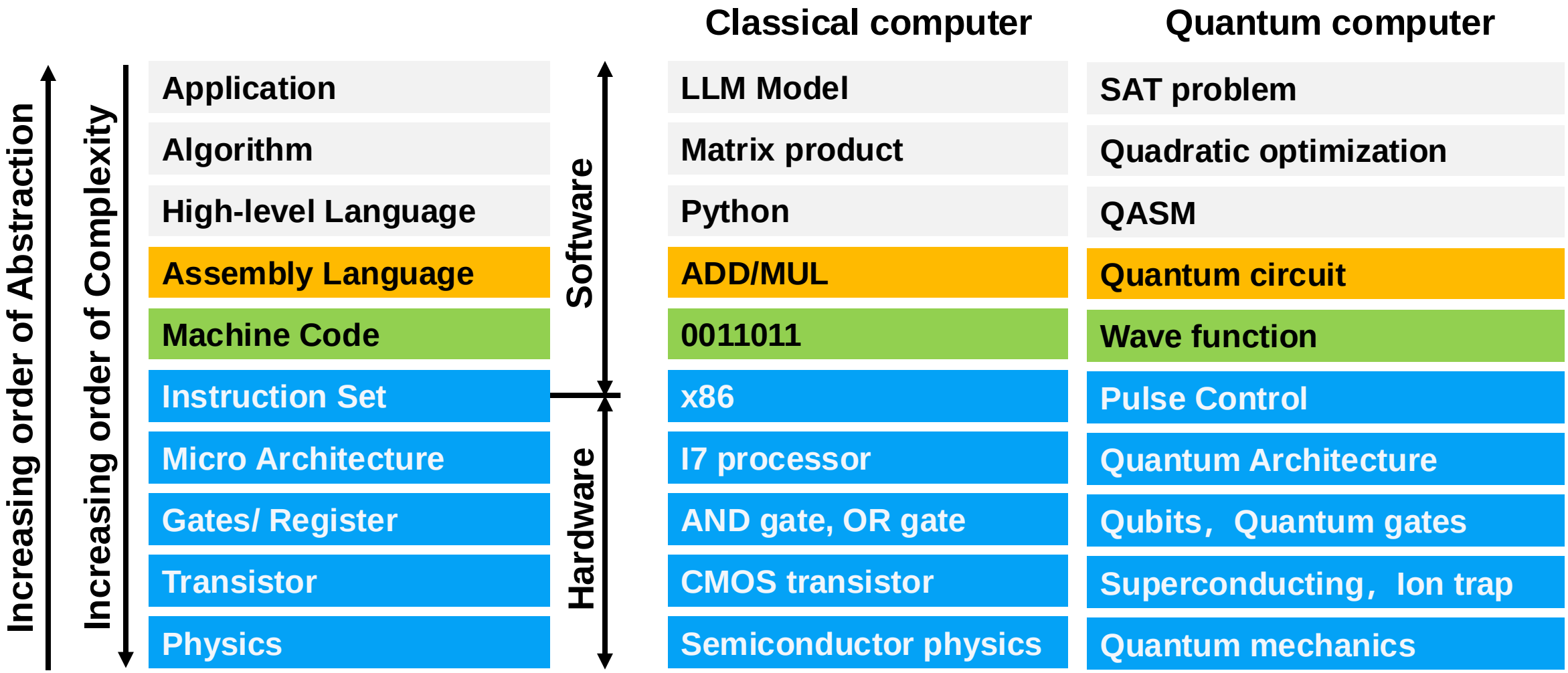


Factor

Application

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

Quantum Computing Architecture

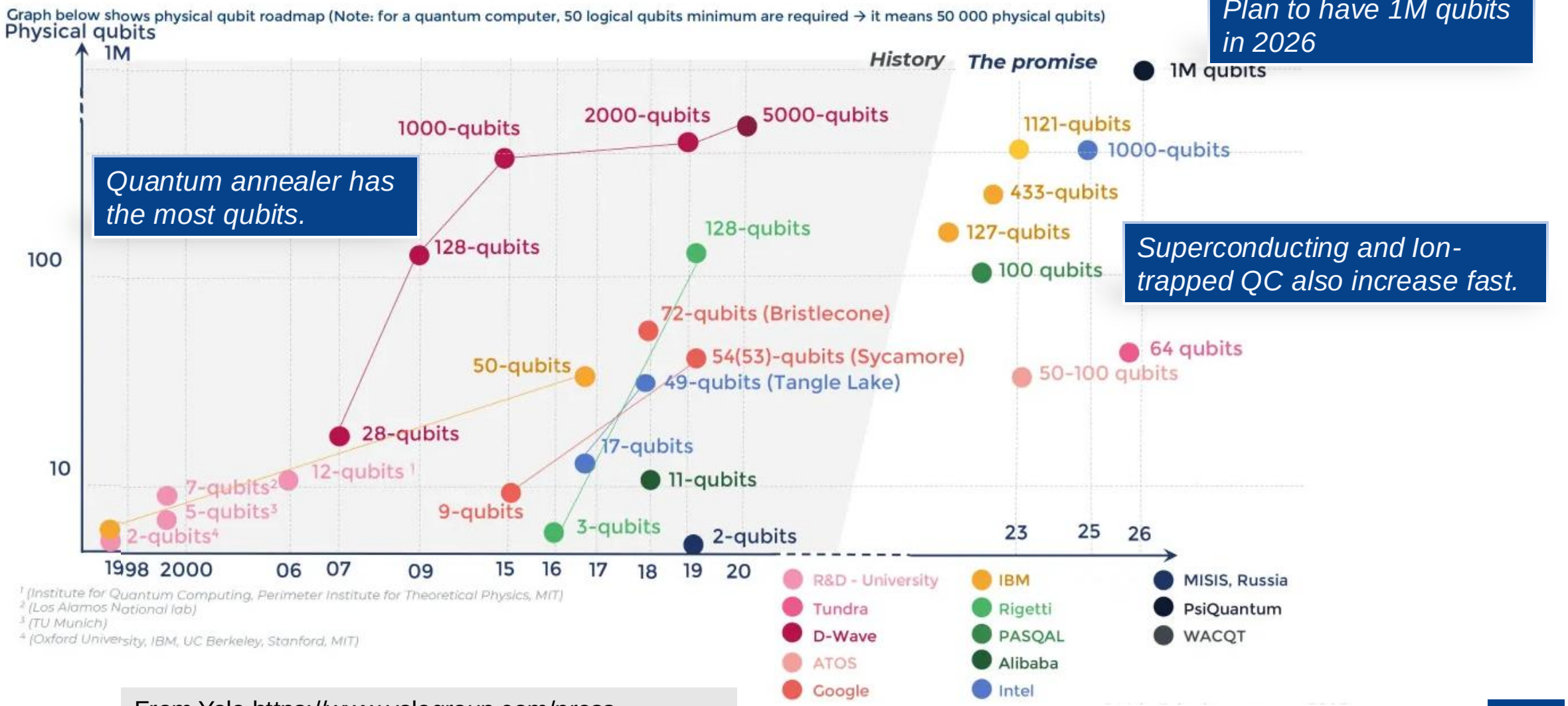


	Classical computer	Quantum computer
Application	LLM Model	SAT problem
Algorithm	Matrix product	Quadratic optimization
High-level Language	Python	QASM
Assembly Language	ADD/MUL	Quantum circuit
Machine Code	0011011	Wave function
Instruction Set	x86	Pulse Control
Micro Architecture	I7 processor	Quantum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

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

Classical and quantum computing has similar architecture

Development of Quantum Computer



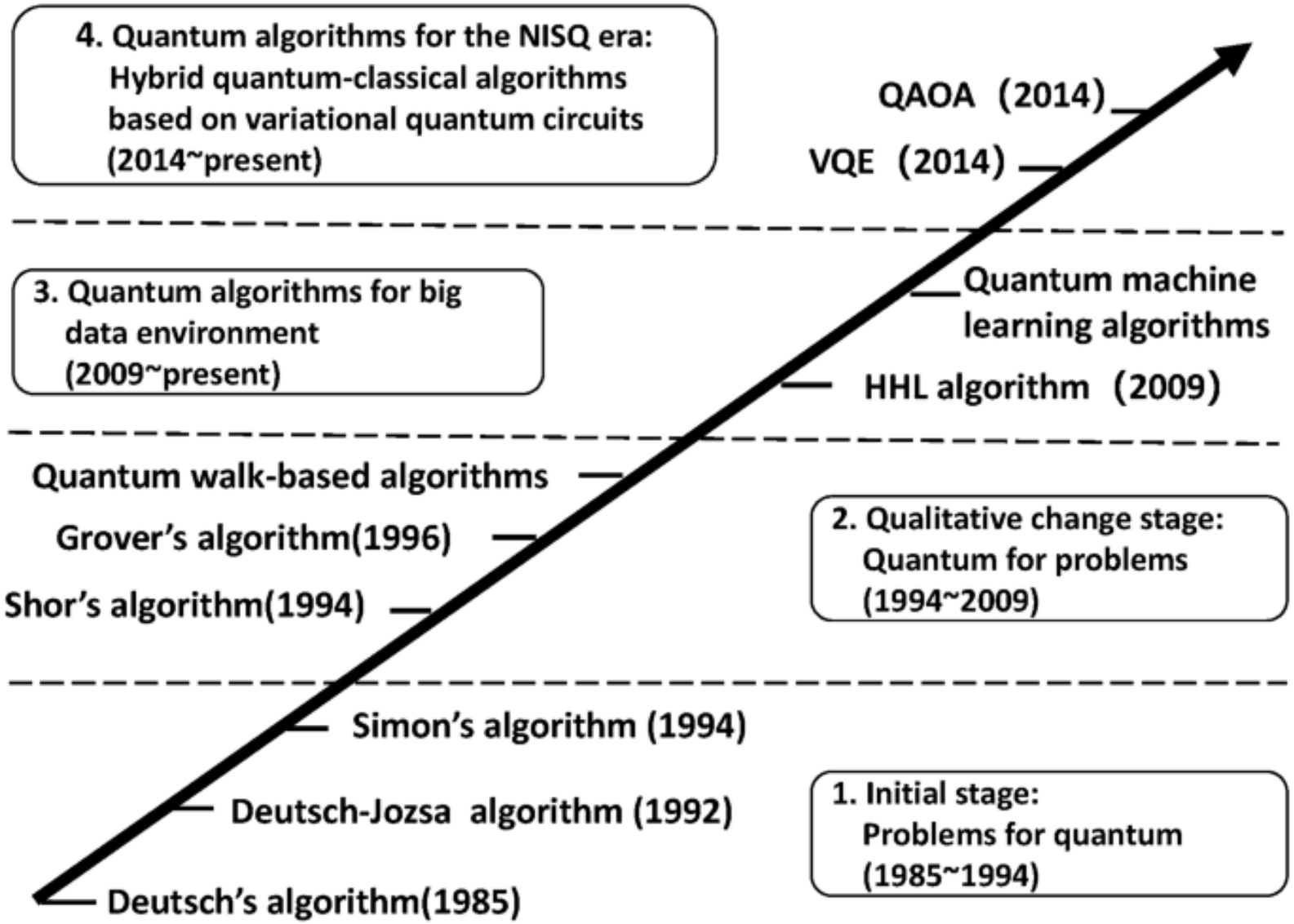
Plan to have 1M qubits in 2026

Quantum annealer has the most qubits.

Superconducting and Ion-trapped QC also increase fast.

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

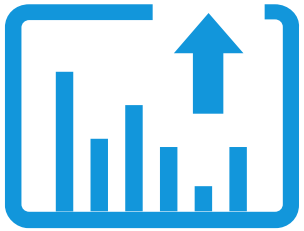
Development of Quantum Algorithm



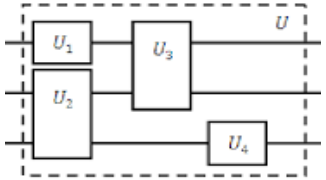
Zhang, S., Li, L. A brief introduction to quantum algorithms. *CCF Trans. HPC* **4**, 2022

Mathematical Model of Quantum Computing

Qubits



Quantum Evolution



Quantum Circuit

Quantum Bit (Qubit)

Single qubit

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

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

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

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

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

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

Quantum Bit (Qubit)

Single qubit

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

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

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

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

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

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

Multiple qubits

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

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

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

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

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

Quantum Evolution

Unitary matrix

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

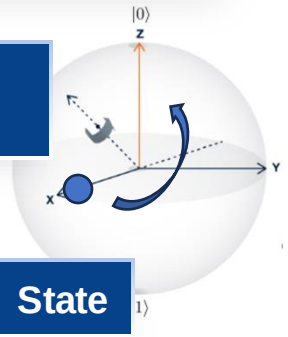
$$UU^{\dagger} = I$$

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

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

Bloch sphere

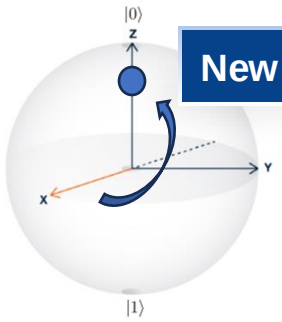
X+Z axes



State

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

π rotation around X+Z axes



New state

Quantum Evolution

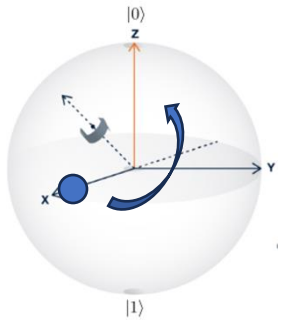
Unitary matrix

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

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

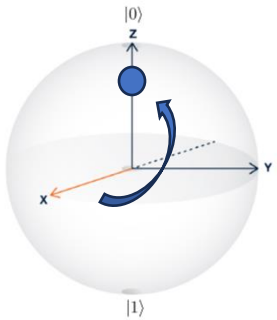
The evolution of qubit state $|\varphi\rangle$ is represented as:

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



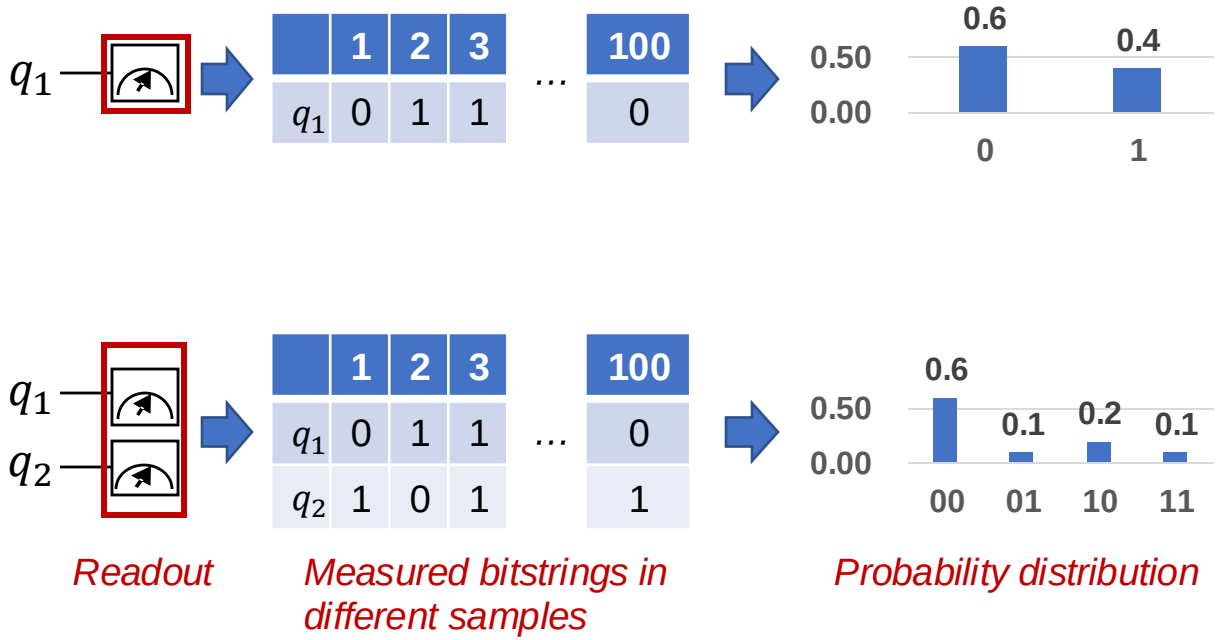
$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

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



Quantum readout

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



Quantum Circuit

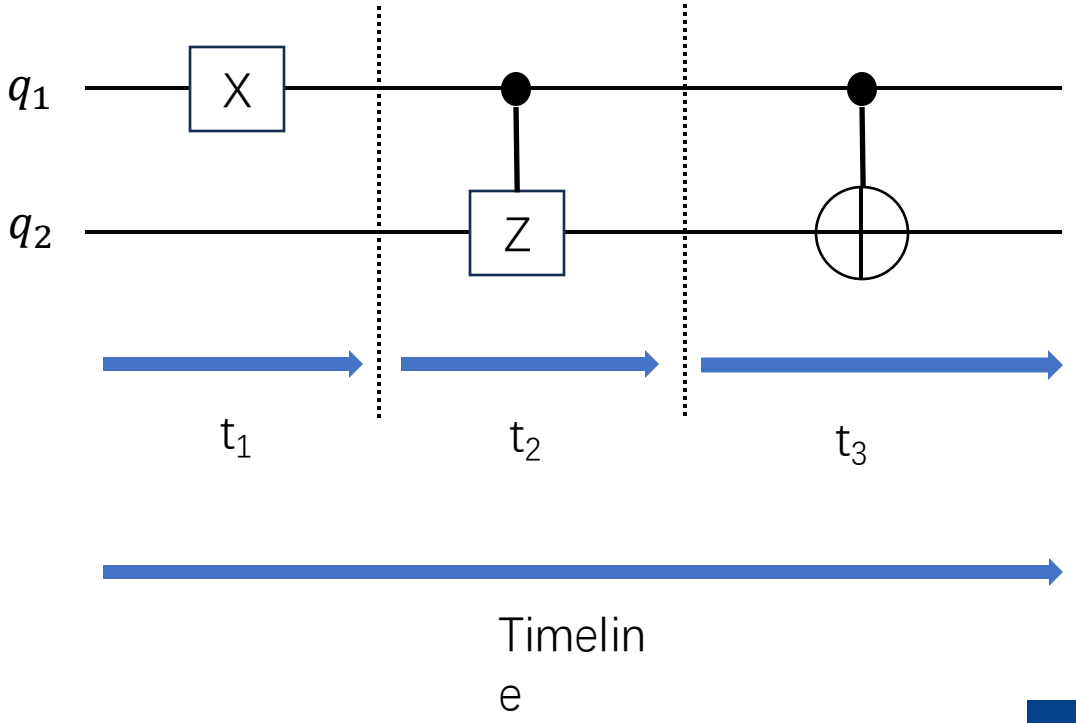
Quantum gates

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

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

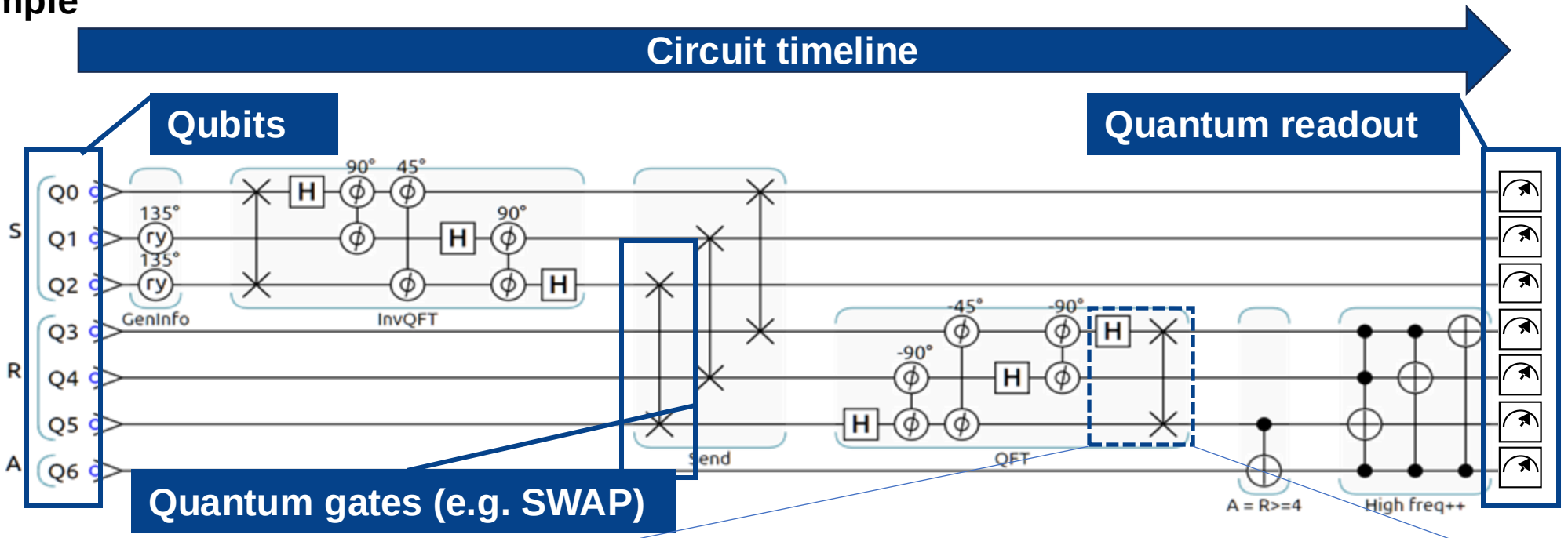
Qubit timeline

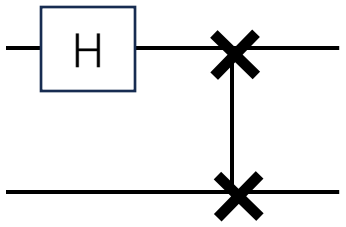
Each line in a quantum circuit represents **a qubit**. The quantum gate in a line is applied on the same qubits **from left to right in time direction.**



Quantum Circuit

For example





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

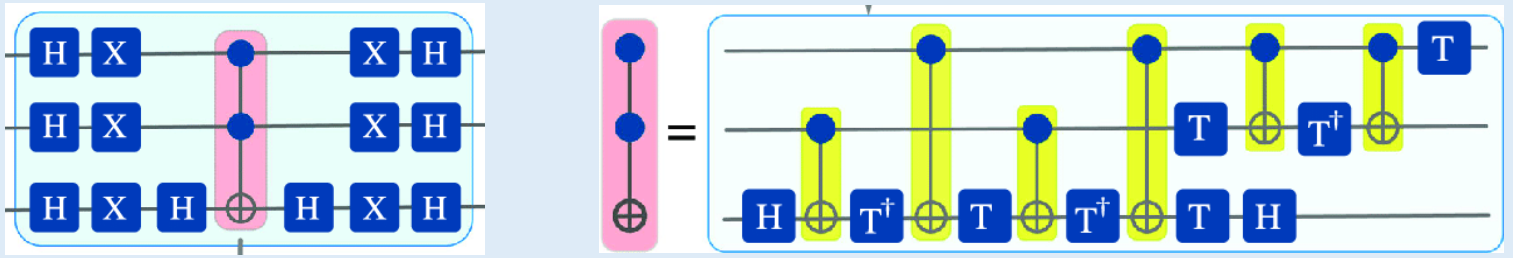
Implementation of Quantum Circuit

On superconducting quantum computer

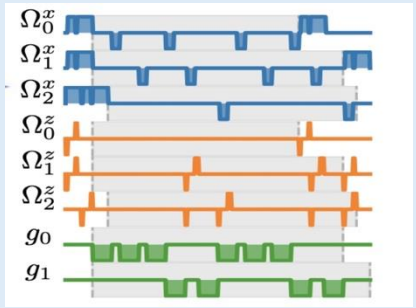
Step 1. Circuit statement

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

Step 2. Circuit compilation



Step3. Circuit execution



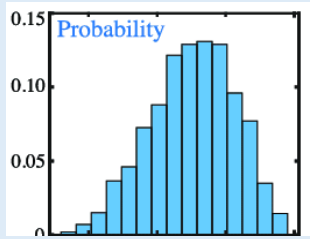
Pulse generation

Quantum device

Step4. Result processing

Error emigration

Visualization

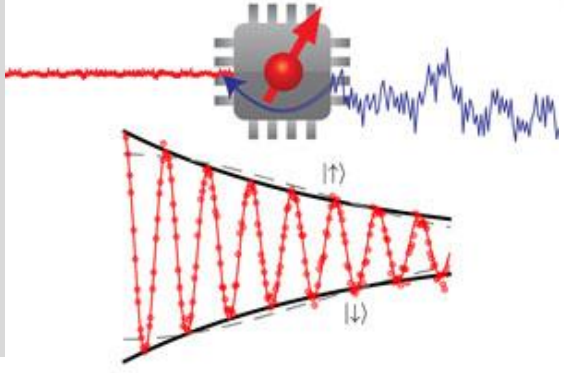


Probability distribution

Challenge in Quantum Computing

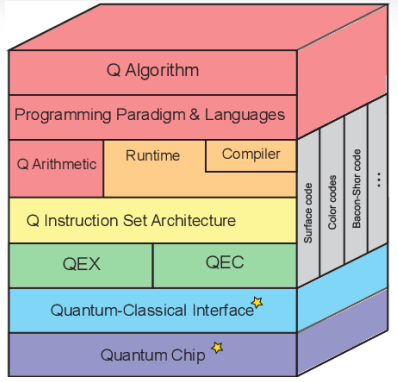
Noise

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



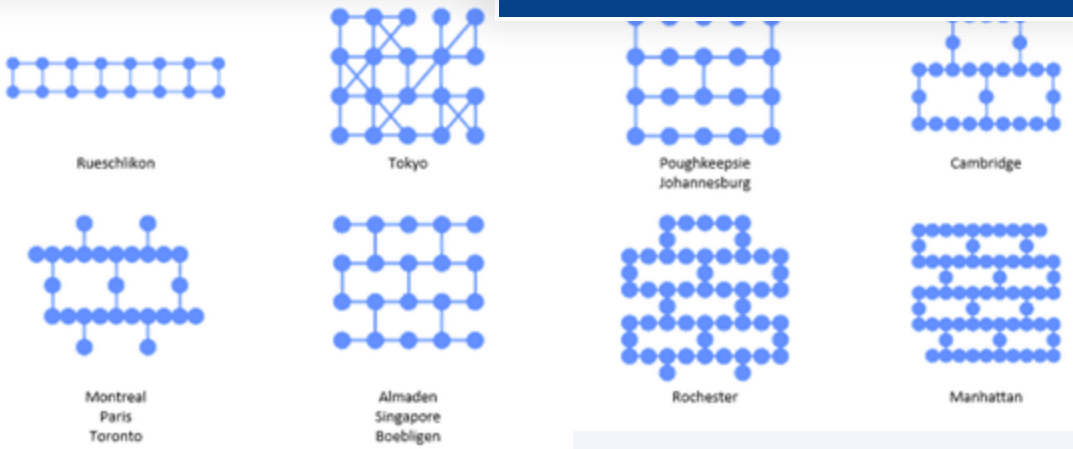
Instructions

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



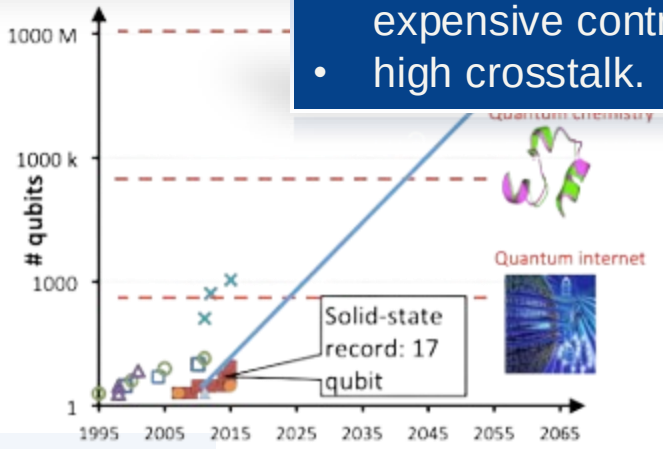
Topology

Locality in communication



Scalability

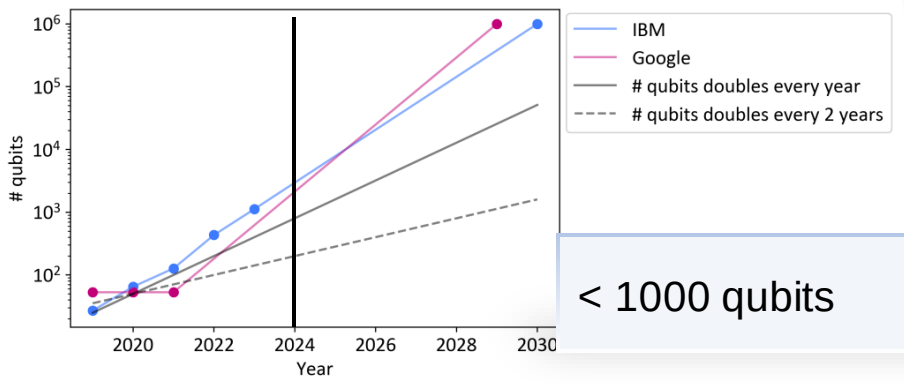
- poor fabrication techniques expensive control systems
- high crosstalk.



Constrained by physical implementation

Results of Challenges

Limited Hardware Resource

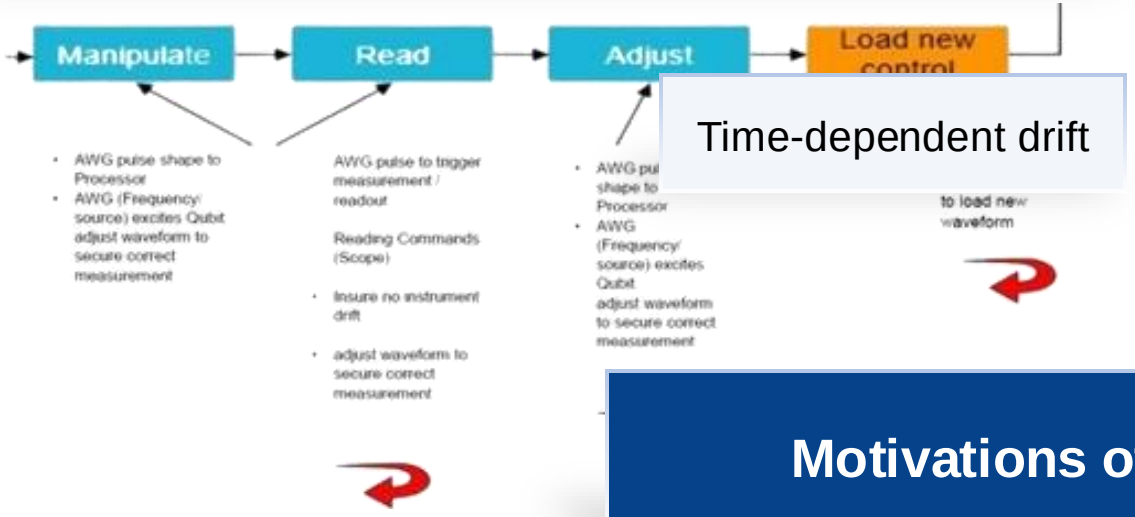


< 1000 qubits

Compilation Complexity

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

Difficulty Of Hardware Calibration



Time-dependent drift

Rare quantum advantage

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

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

Motivations of JanusQ

Janus (Taiyuan) Quantum Cloud Platform

Taiyuan (太元) & Janus

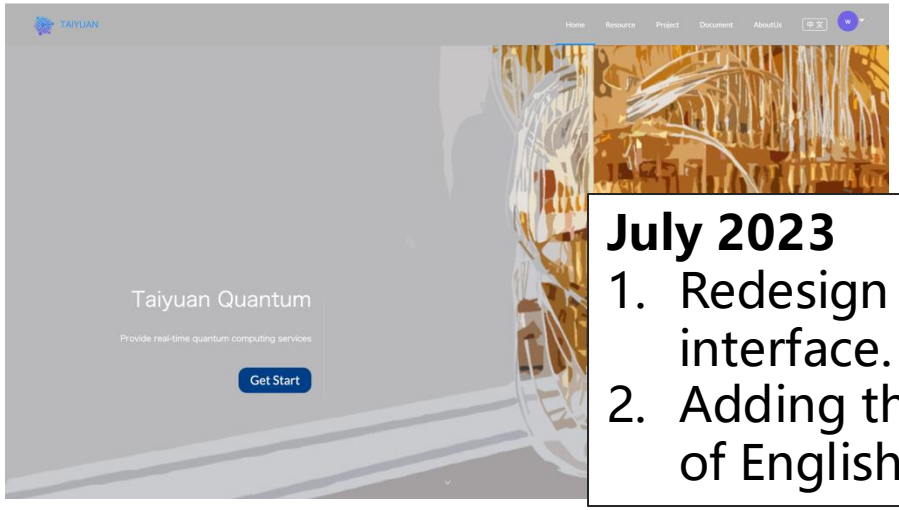
Yin and Yang

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

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

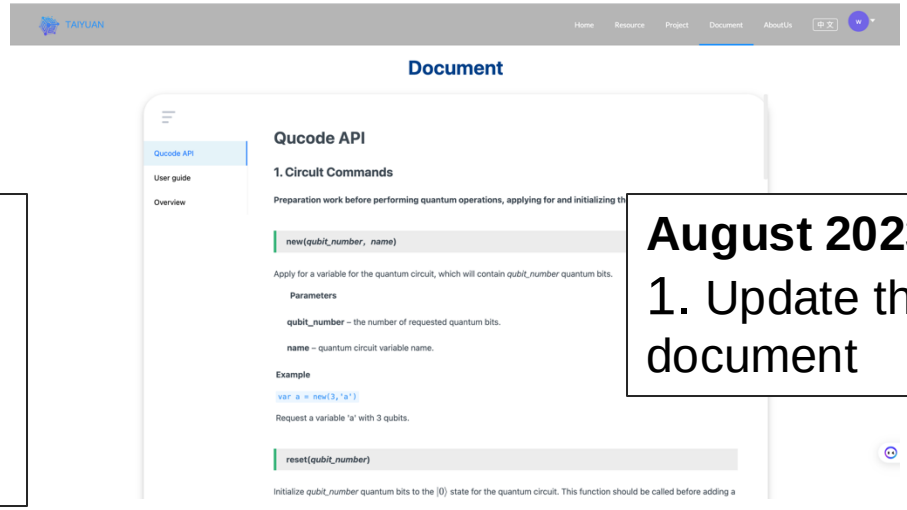
Development History of Janus Cloud

Four updates since July 2023



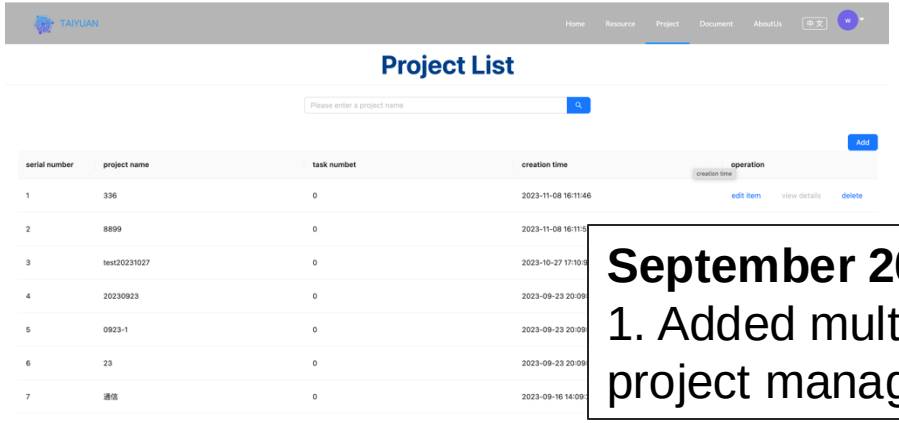
July 2023

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



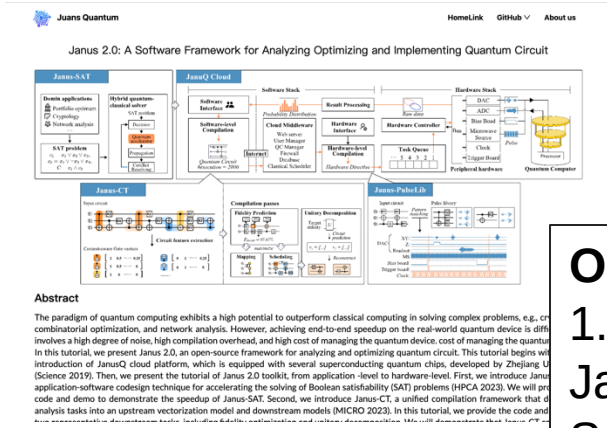
August 2023

1. Update the document



September 2023

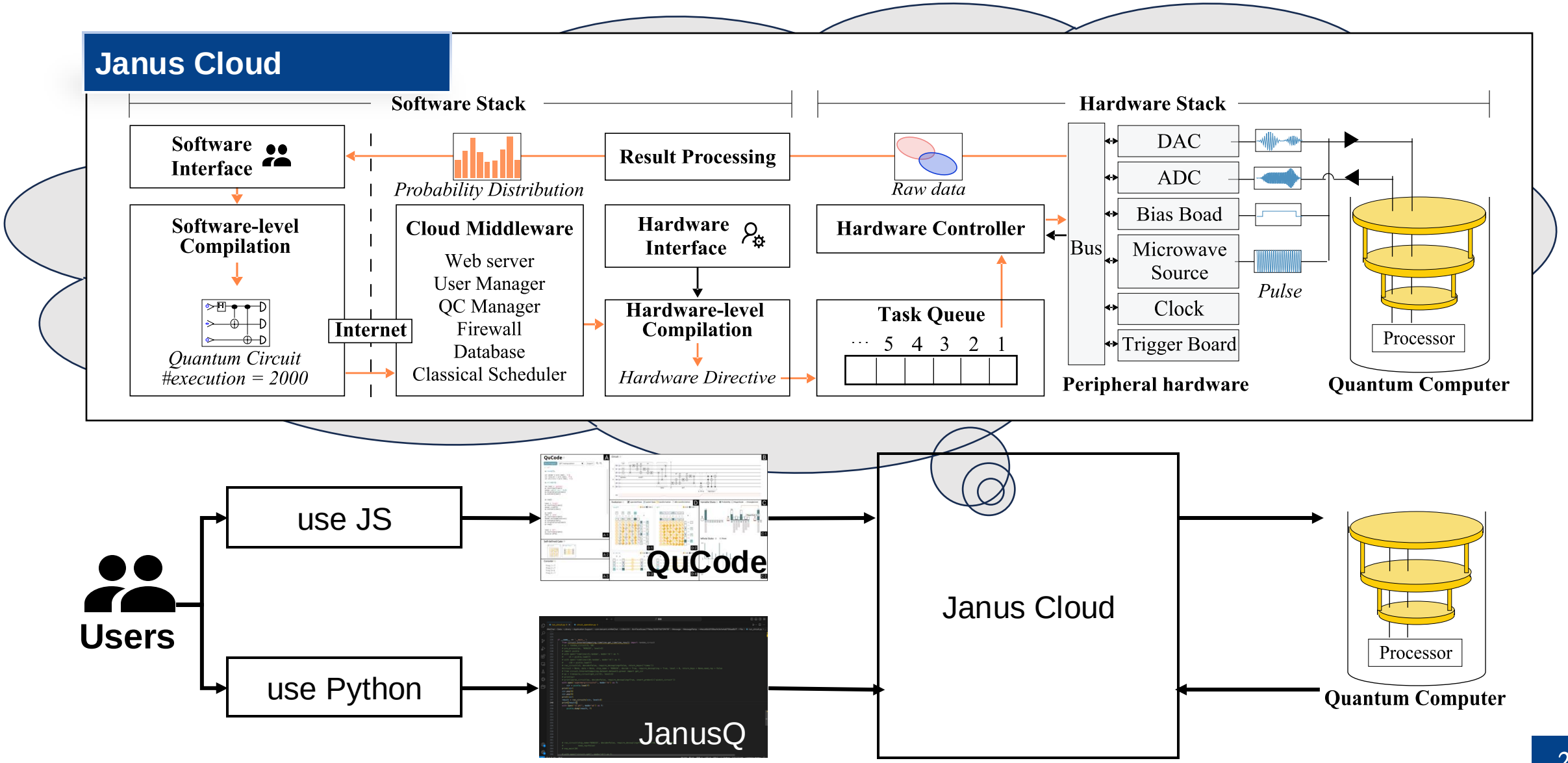
1. Added multi-level project management.



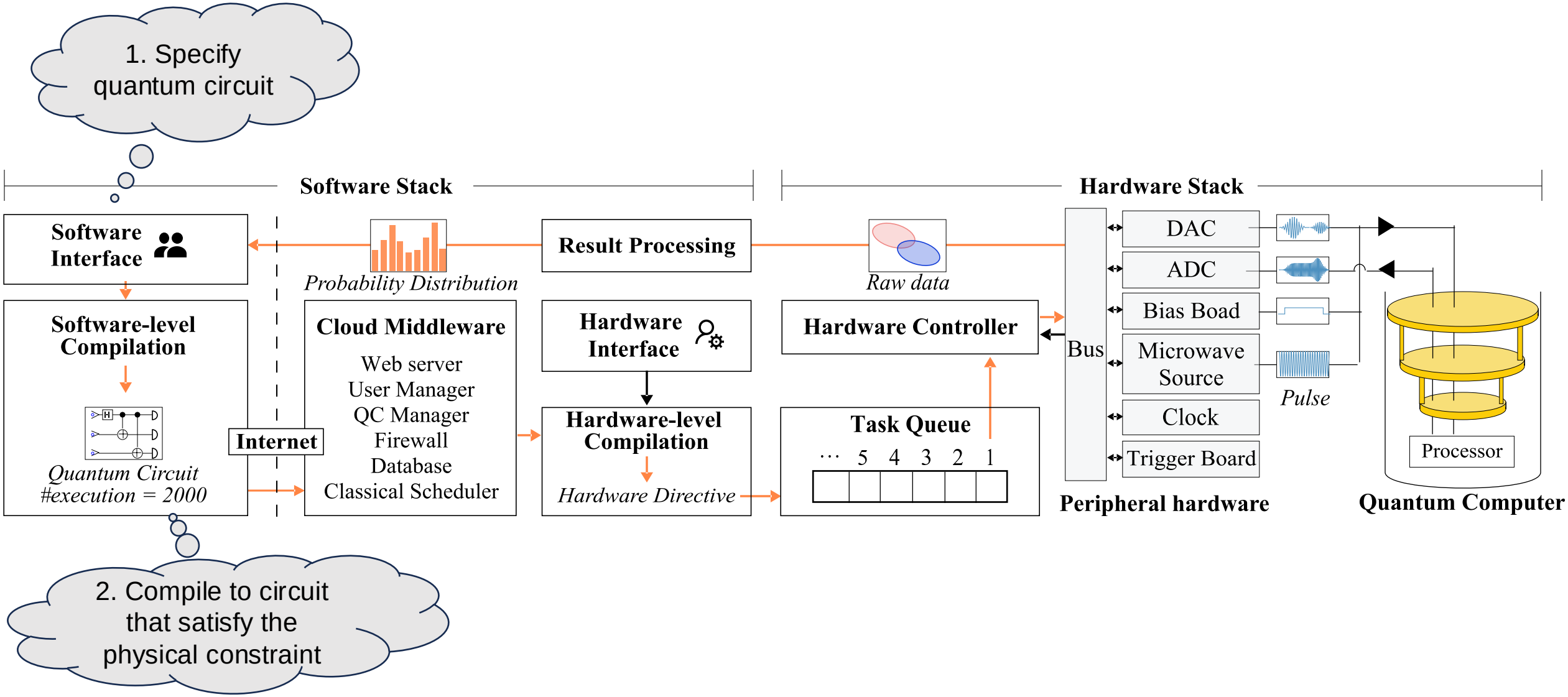
October 2023

1. Enable the online Janus-CT and Janus-SAT

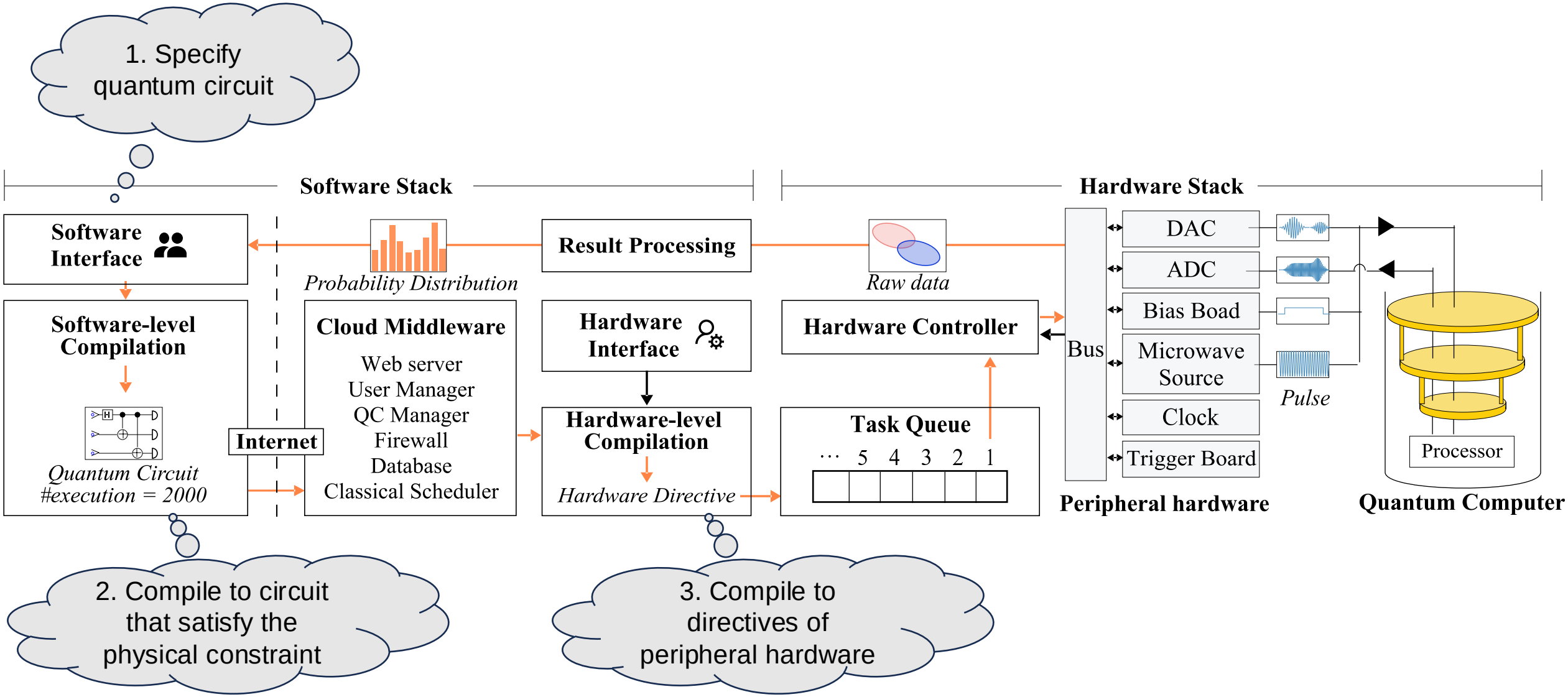
What we can do on Janus Platform



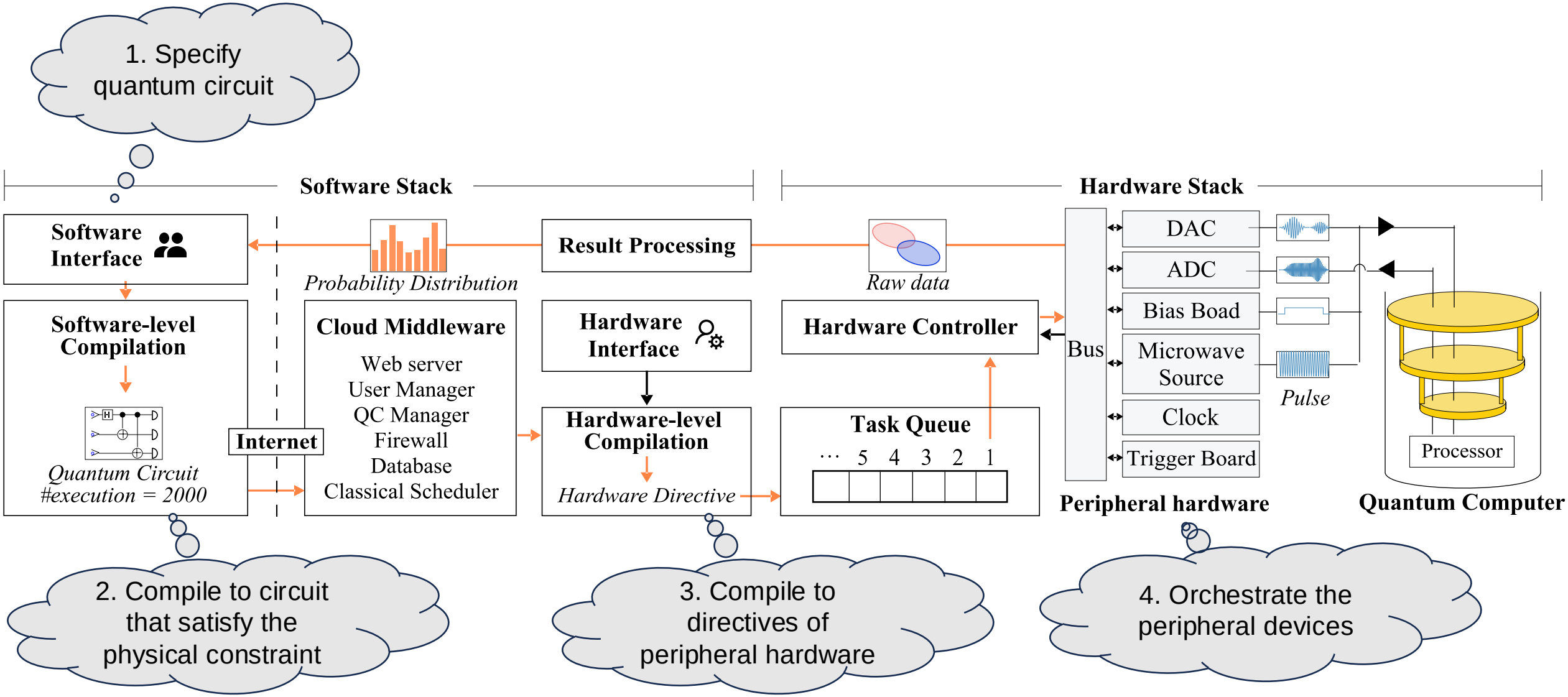
What we can do on Janus Platform



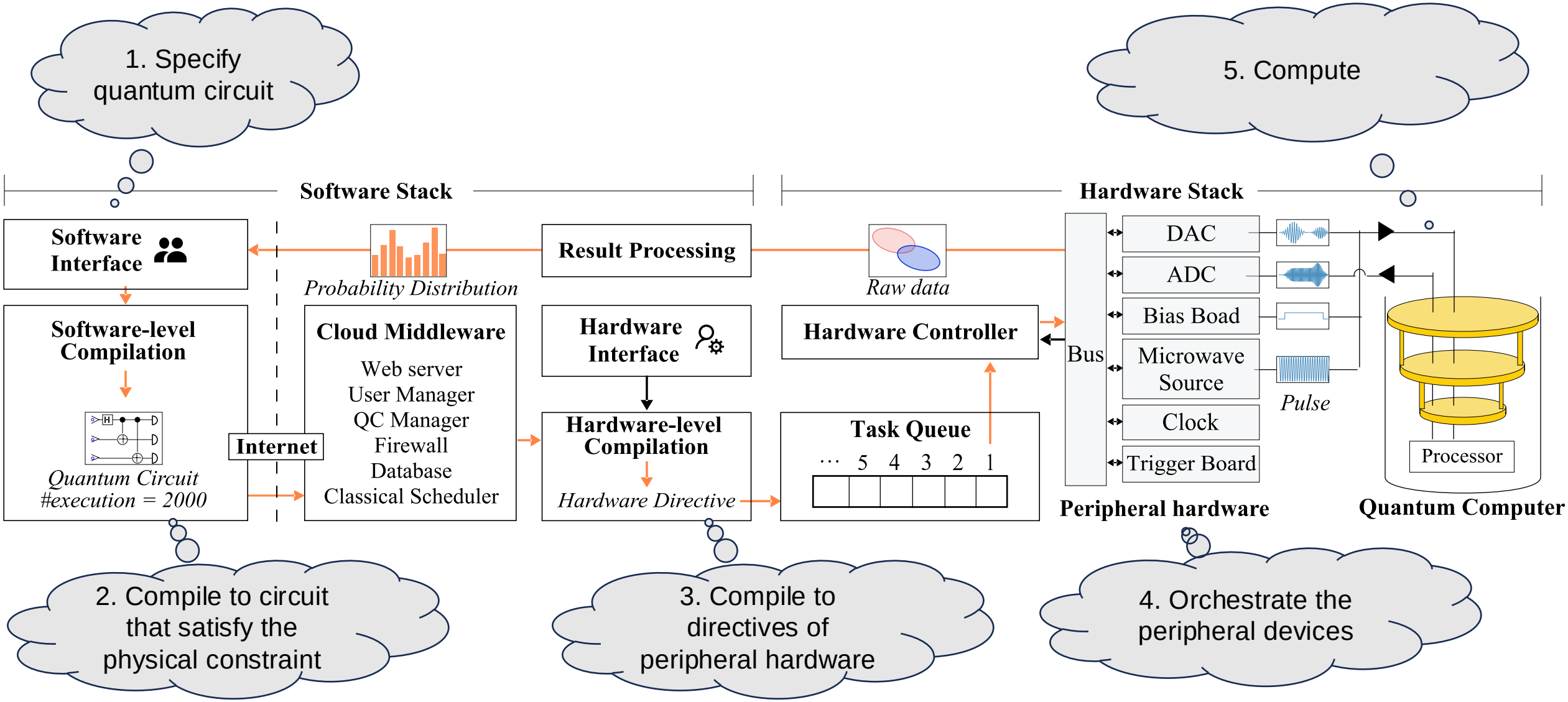
What we can do on Janus Platform



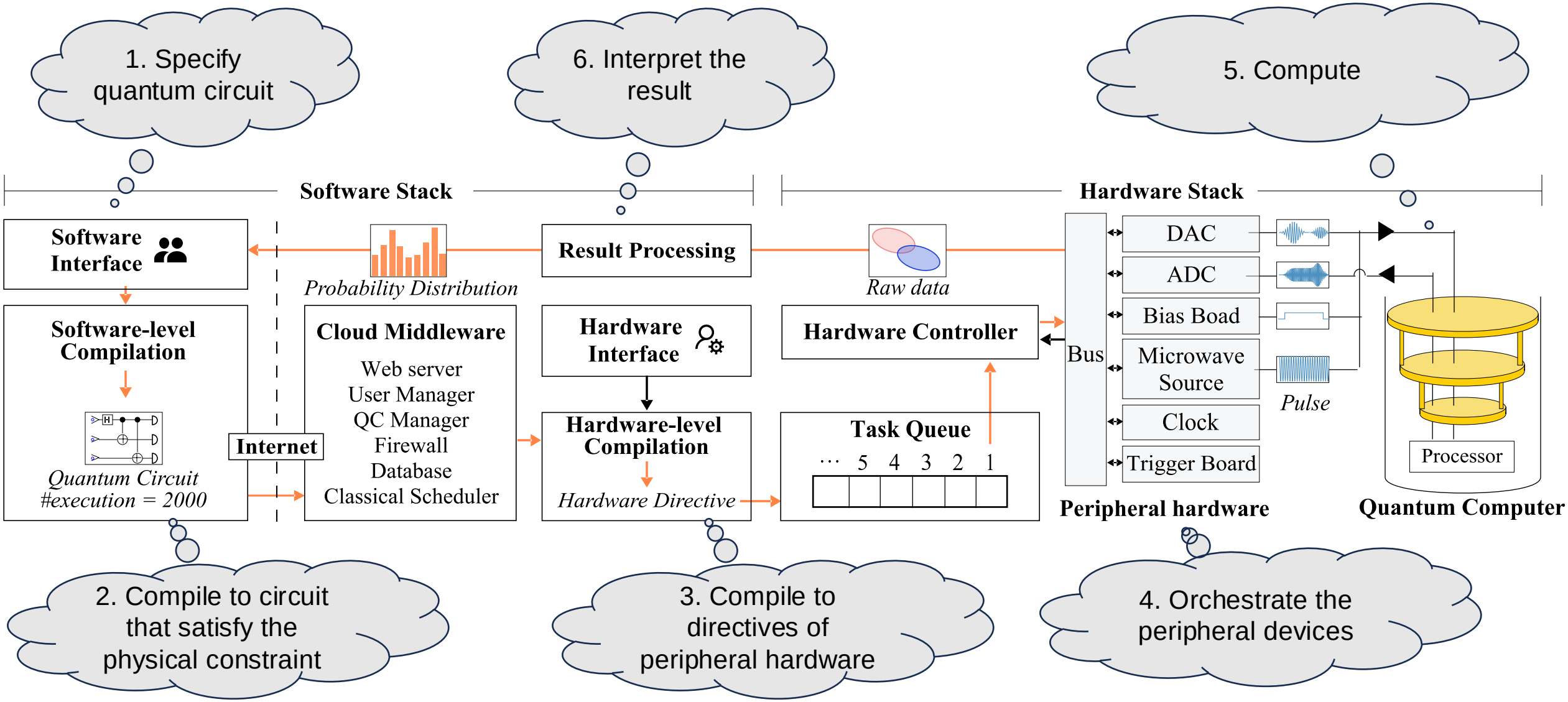
What we can do on Janus Platform



What we can do on Janus Platform

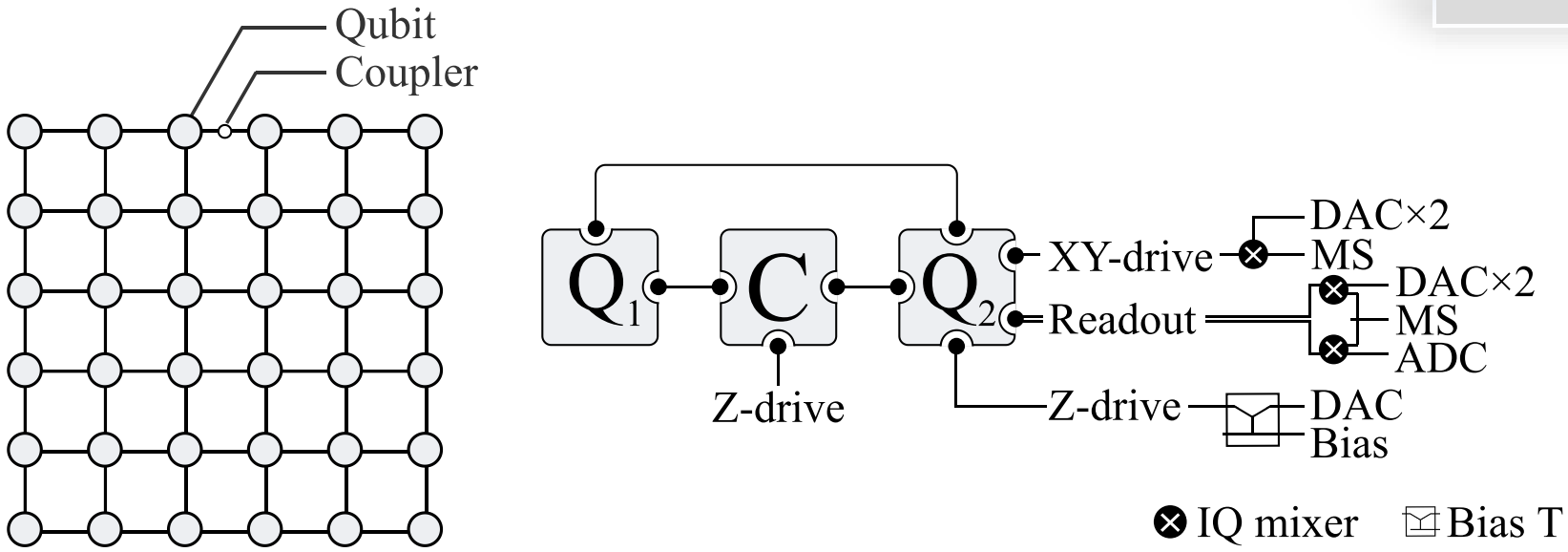


What we can do on Janus Platform



Different compared to other quantum cloud platforms

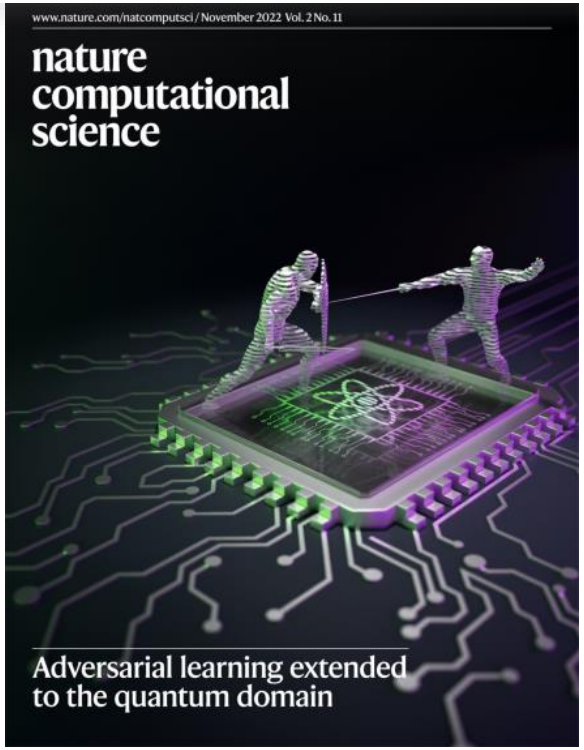
Custom hardware & more flexibility to improve the fidelity



(a) Processor topology. (b) Sketch of two qubits and a coupler.

Customize topology, processor topology, qubit frequency for academic researches.

Achieving 99% accuracy in the medical image classification.



Cover of Volume 2 Issue 11, November 2022, Nature computational science

Creating an Account

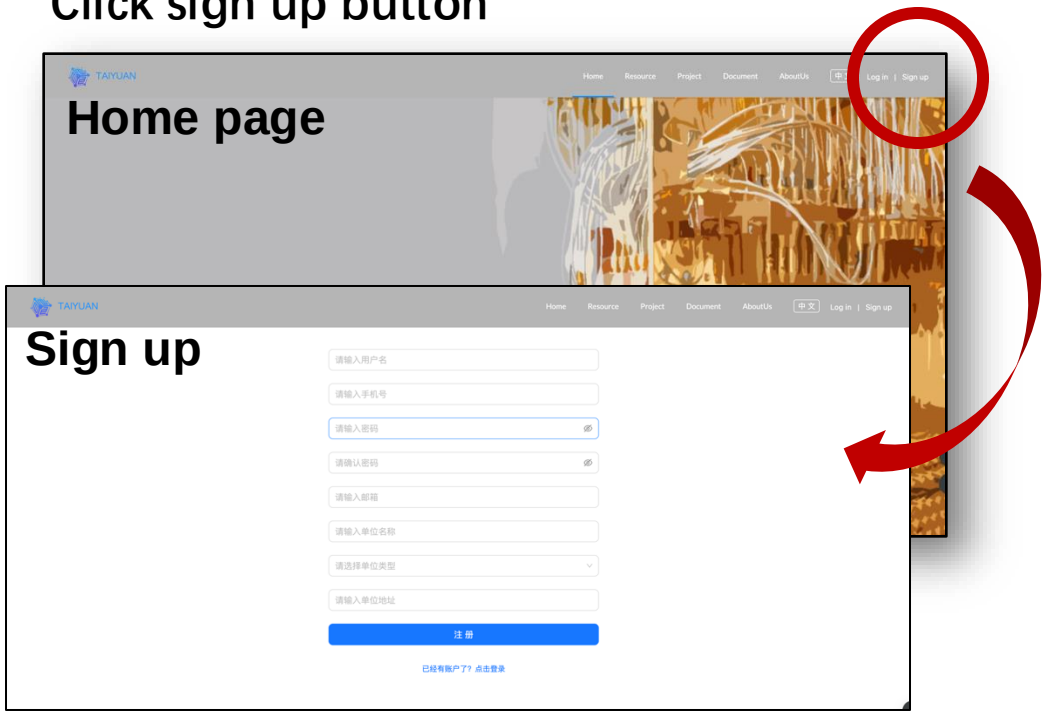
Go to Janus Cloud

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

QR code

Sign up

Click sign up button

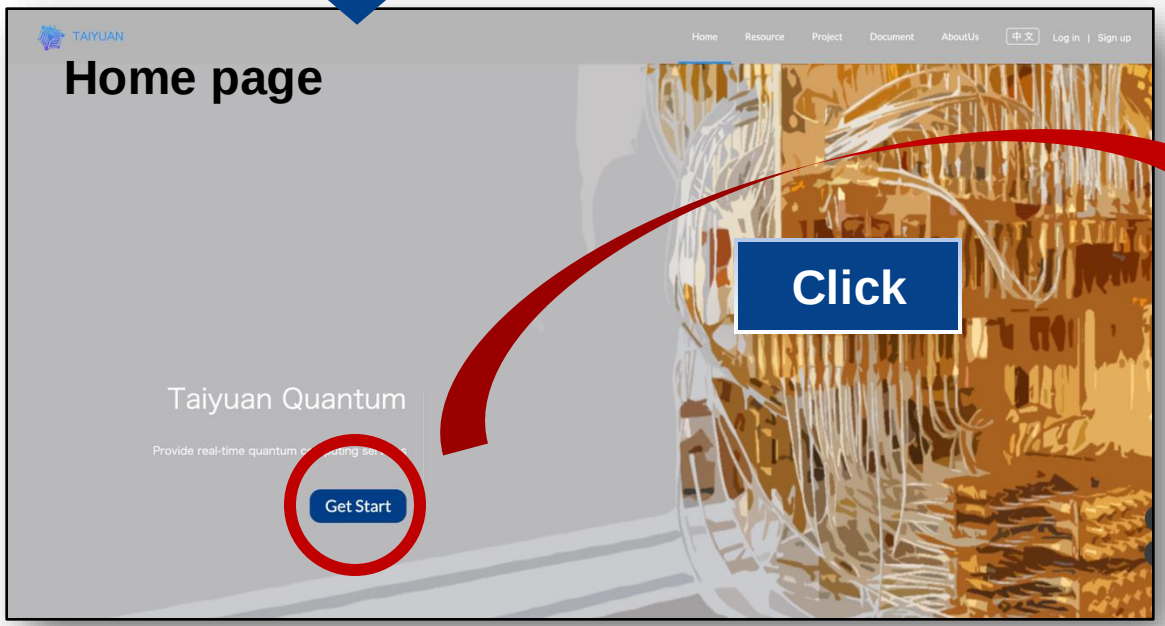


- After signing up, you will get an API key for task submission.
- **The quantum computer can be accessed on the website during the tutorial !!**

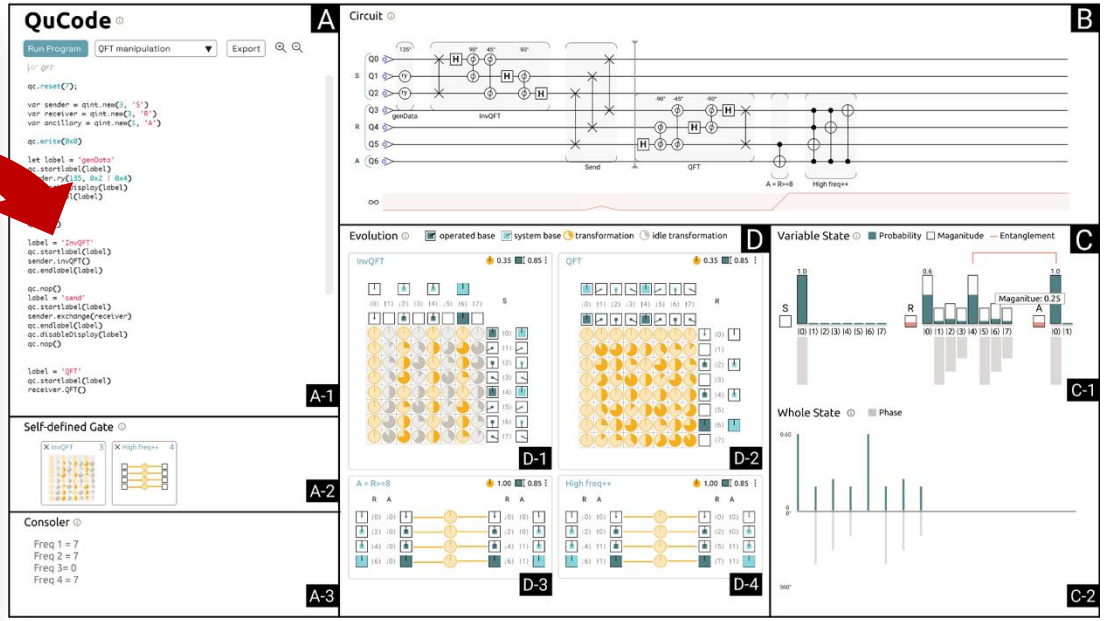
Running Program On the Website

Submitting on website

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



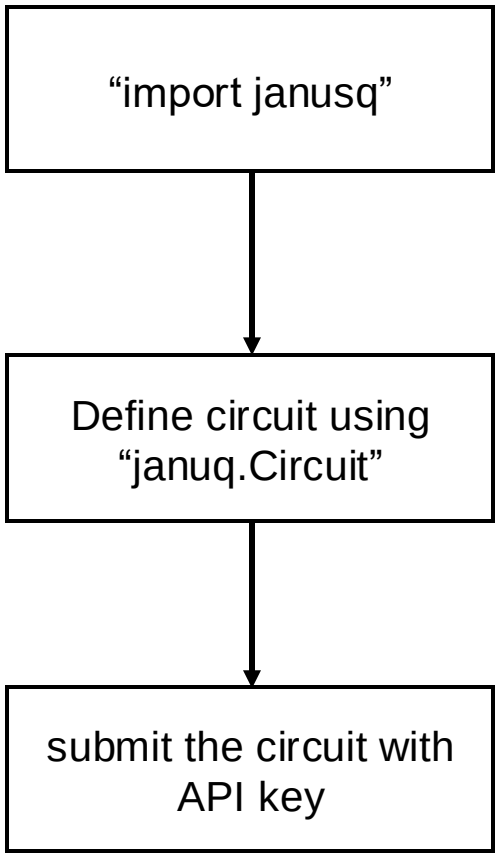
Programming on the Code editor



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

Running Program By Python API

Submitting by Python



Import package

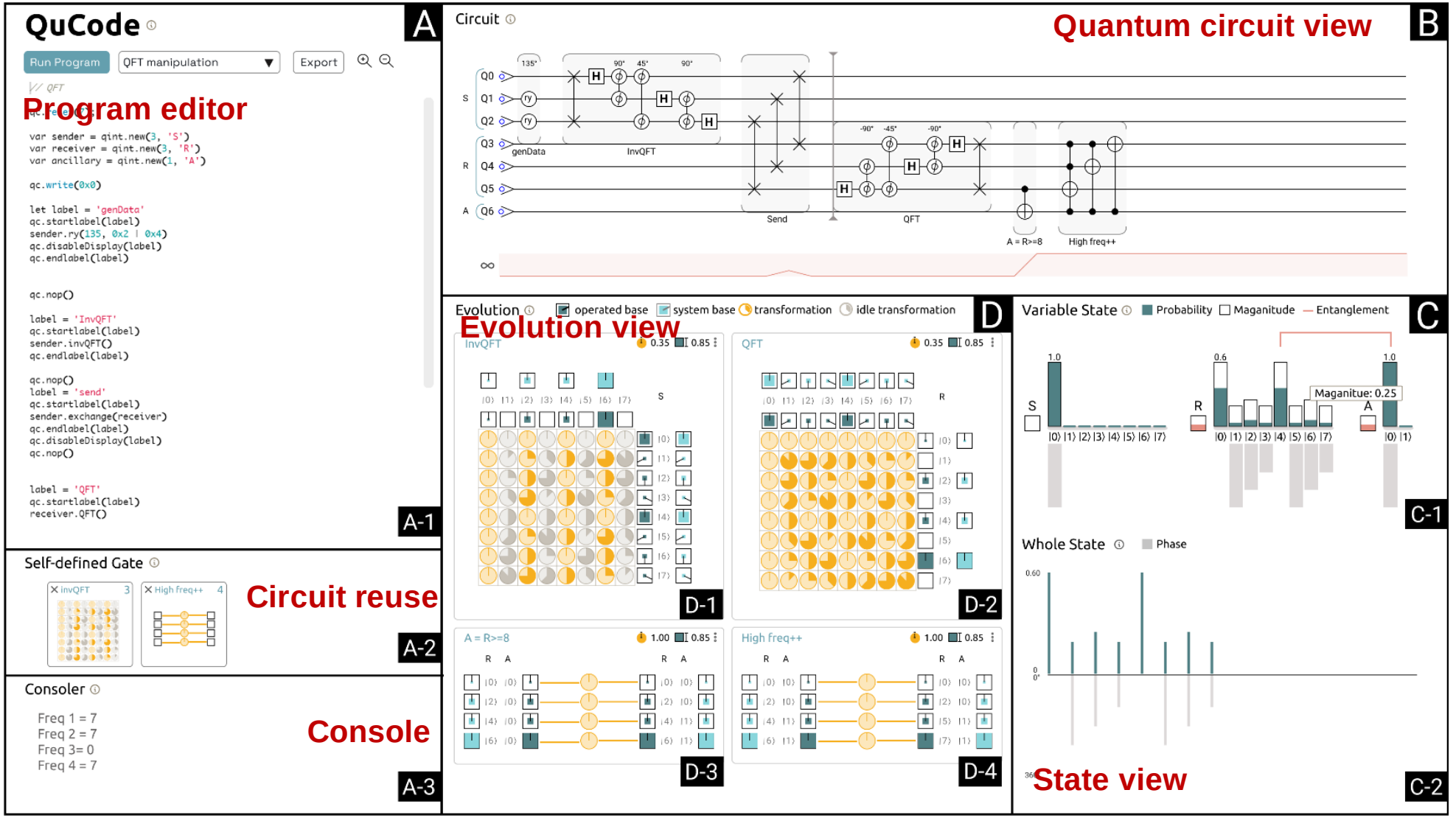
Define a circuit

Submit circuit

Get result

```
import matplotlib.pyplot as plt
from janusq.cloud_interface import submit, get_result
from janusq.data_objects.circuit import 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)

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

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

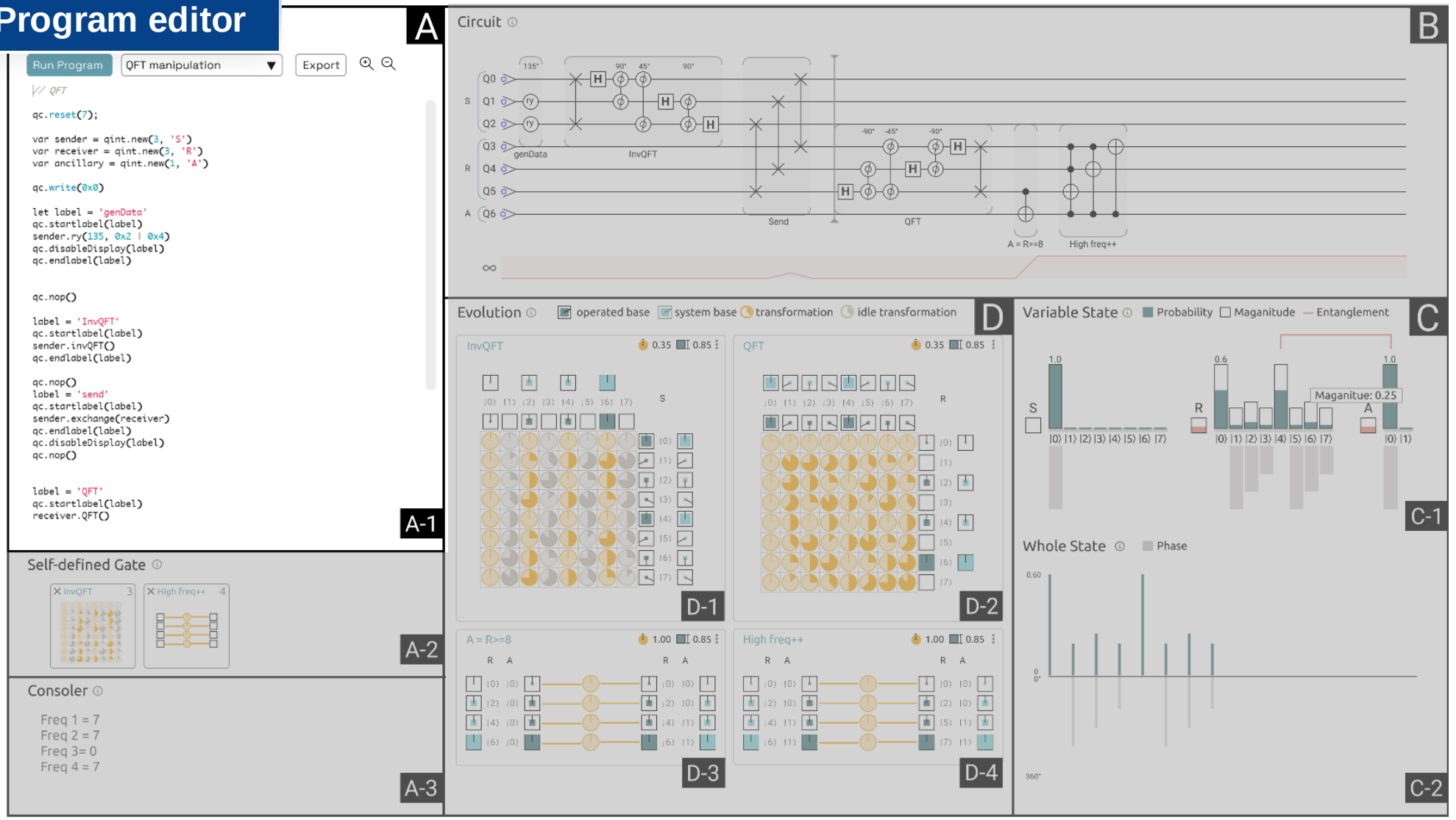
QuCode: Online Quantum Program Editor



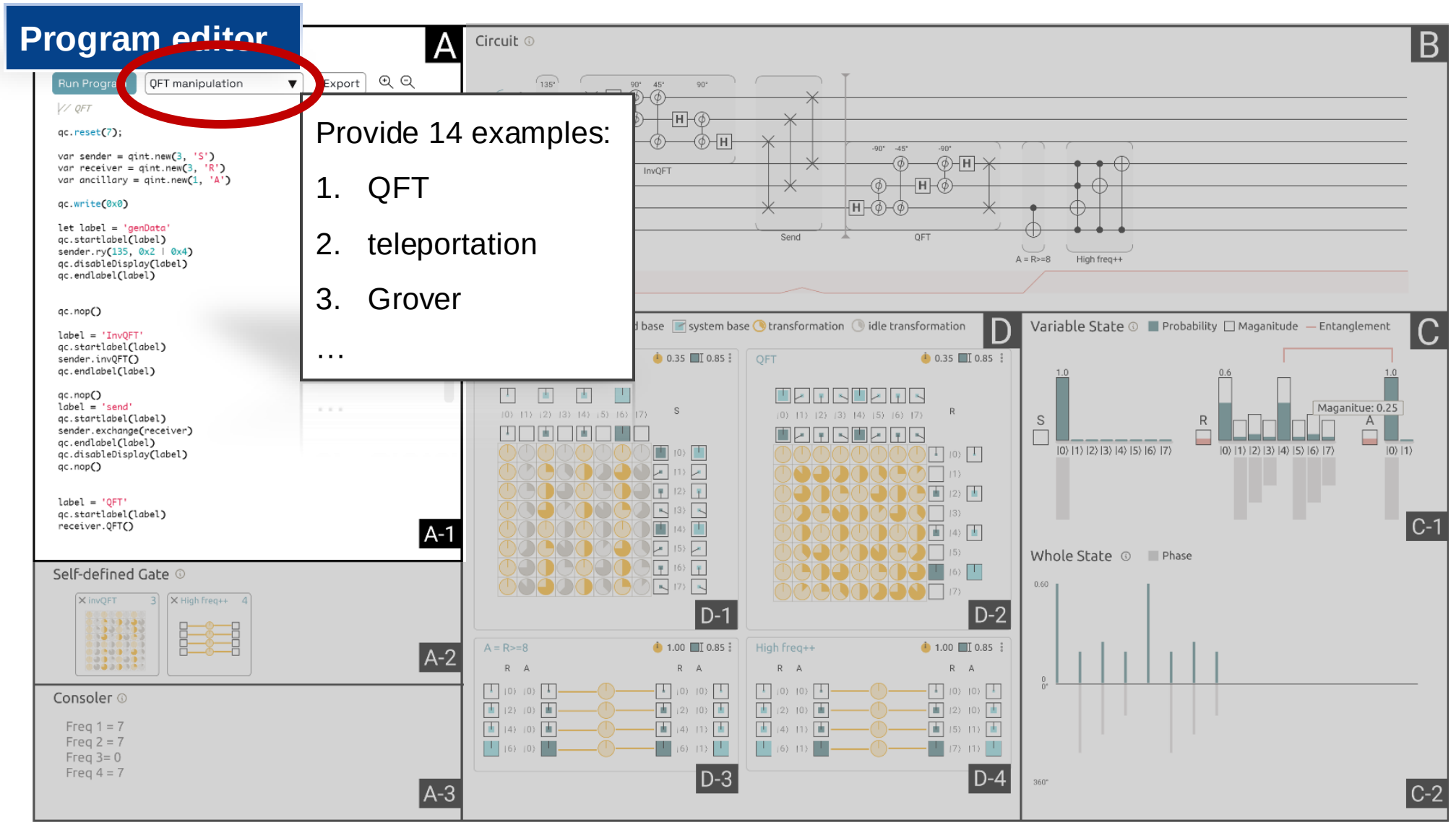
QuCode: Online Quantum Program Editor

ZHEJIANG UNIVERSITY

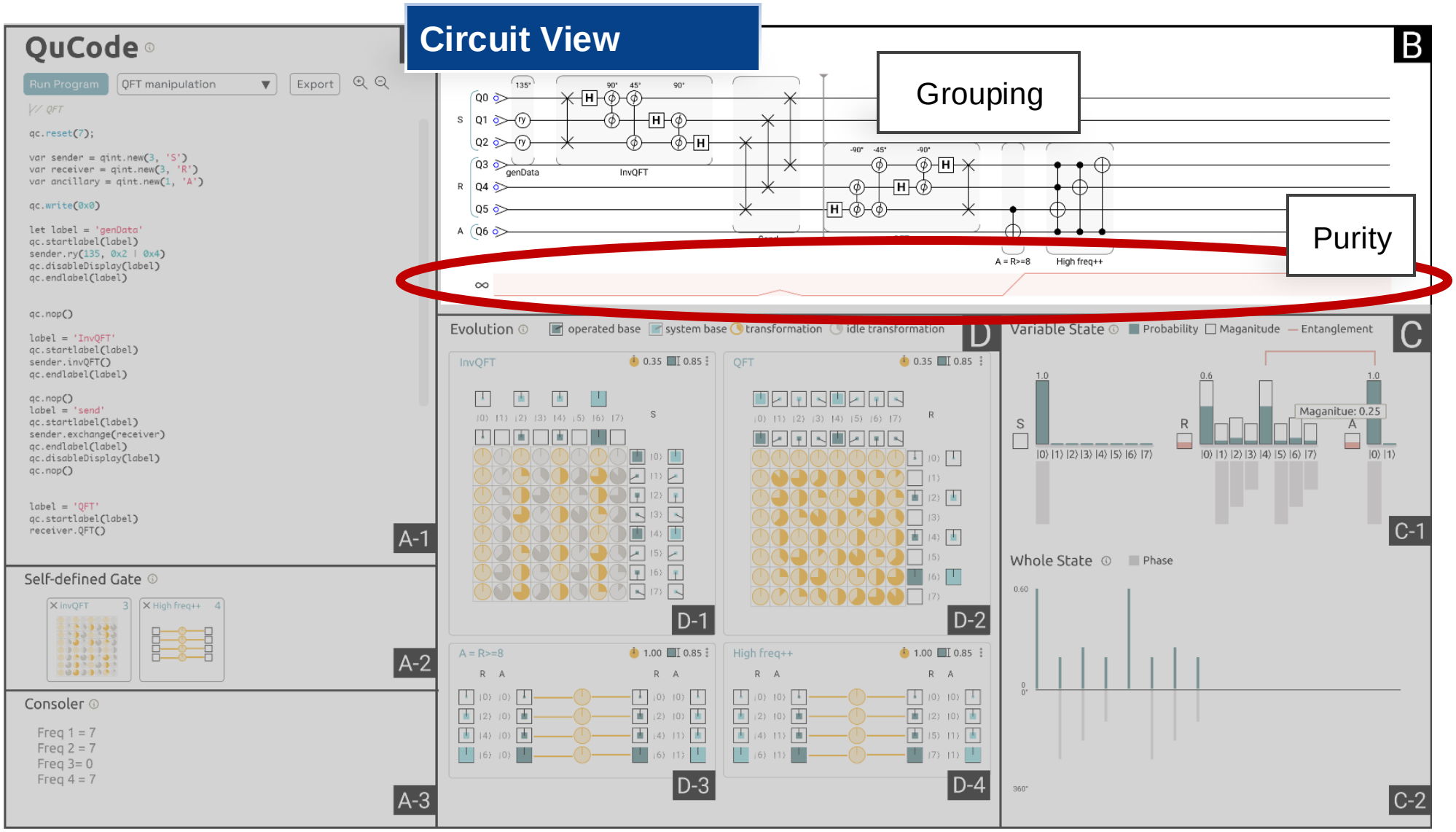
Program editor



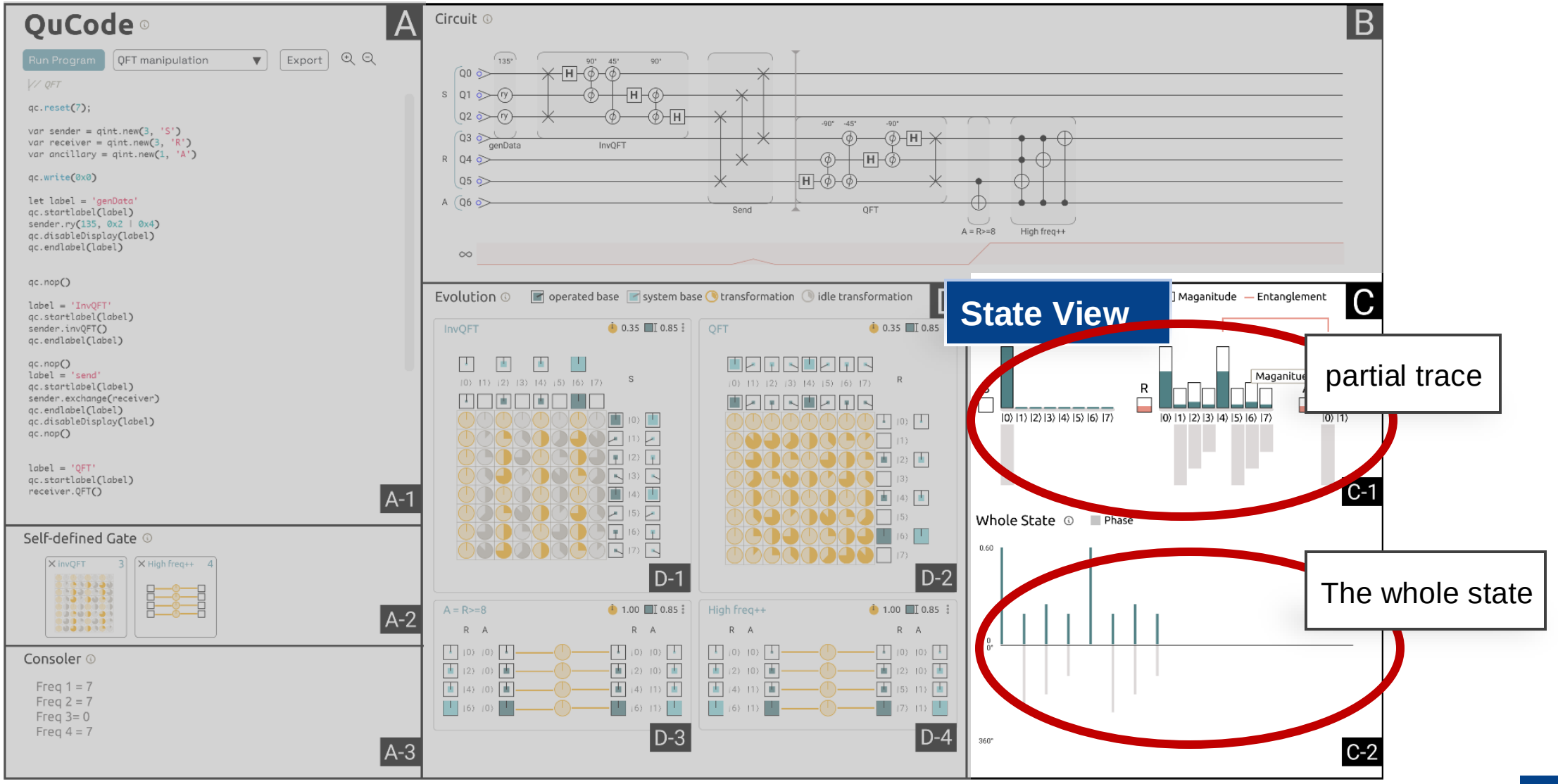
QuCode: Online Quantum Program Editor



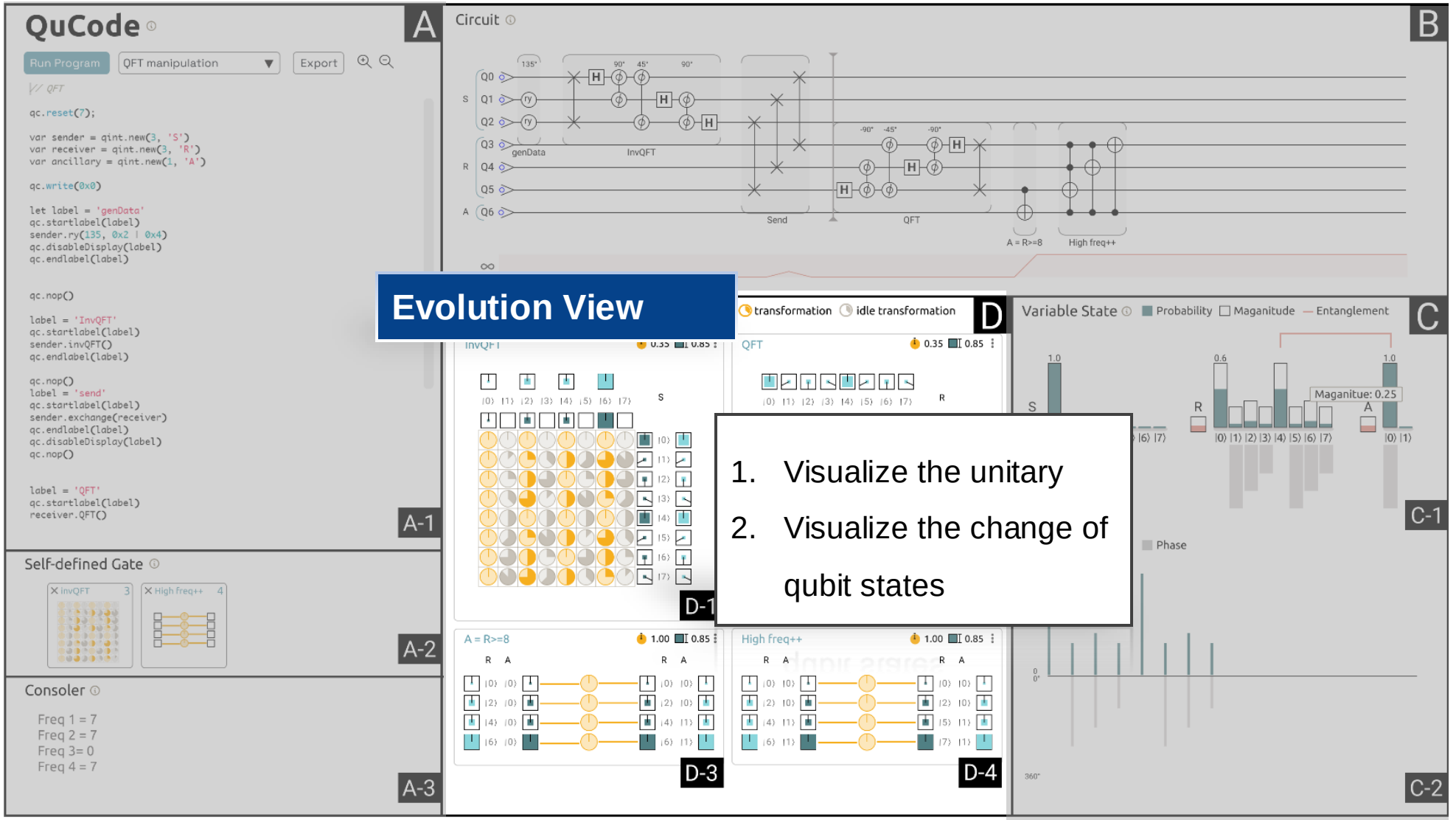
QuCode: Online Quantum Program Editor



QuCode: Online Quantum Program Editor



QuCode: Online Quantum Program Editor



Simulation of Time crystal

Mathematical formulation

$$H(t) = \begin{cases} H_1, & for\ 0 \le t < T_1 \\ H_2, & for\ T_1 \le t < T \end{cases}$$

$$H_1 \equiv \left(\frac{\pi}{2}\right) \sum_k \sigma_k^x$$

$$H_2 \equiv -\sum_k J_k\, \sigma_{k-1}^z \sigma_k^x \sigma_{k+1}^z$$

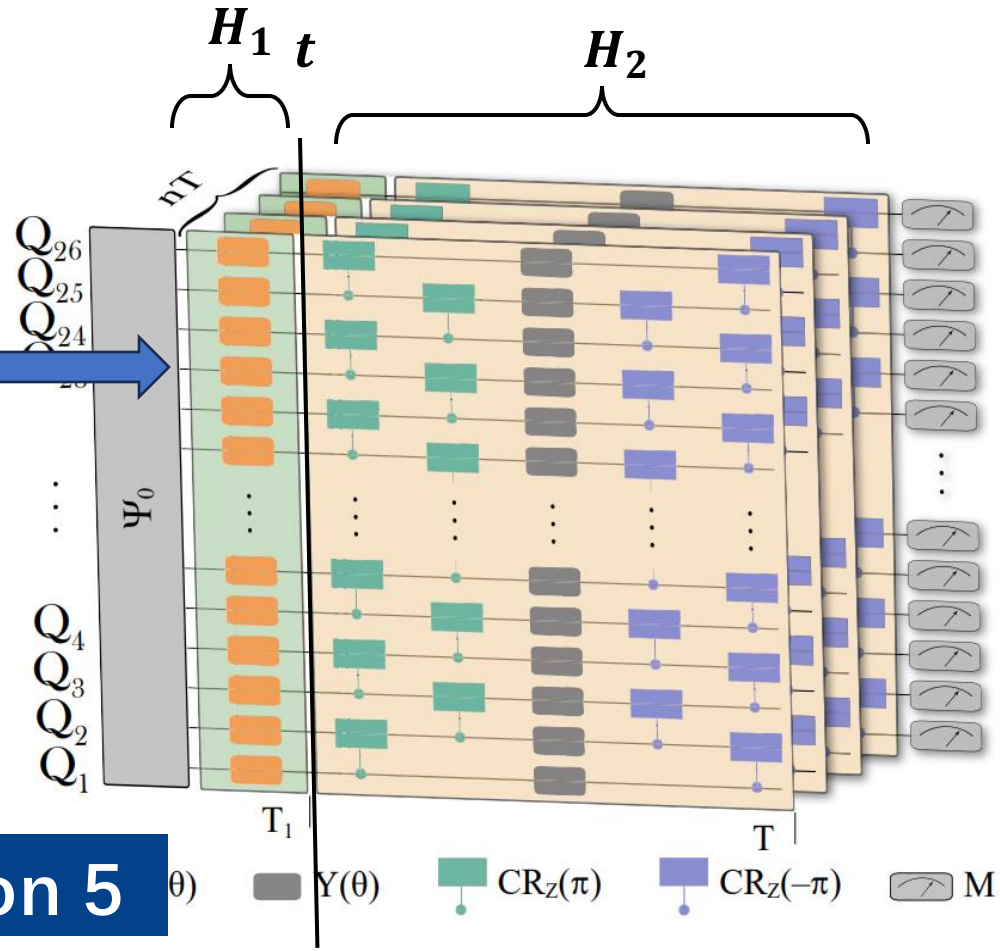
$$J_k \in [0, 2]$$

$$U_1(t) = e^{-iH_1}$$

A layer of rotation gates along the x axis

20 random disorder instances $\{J_k\}$

Circuit Implementation



Introduced In Session 5

Installing JanusQ

https://github.com/JanusQ/JanusQ

Installing Methods

Language: Python 3, C++

Available platforms: Linux, Mac, Windows

Janus-SAT has not been tested on Mac and Windows.

Hardware requirements:

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

Hardware used in the experiment

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

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

Installation:

From Docker (recommend)

From Source code

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

Installation 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/janusq
```

Step 3. start image

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

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

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

Step 5. Or use Jupyter Lab to connect the docker

http://localhost:9999/lab

The password of the docker is “” (empty).

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

Installation 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
pip install -r requirements.txt
```

Step 3. compile Janus-SAT

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

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

implemented based on C++.

Testing JanusQ

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

Test Janus-CT

```
from janusq.analysis.vectorization import *

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

Test Janus-FEM

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

protocol = EnumeratedProtocol(4)
```

Test Janus-SAT

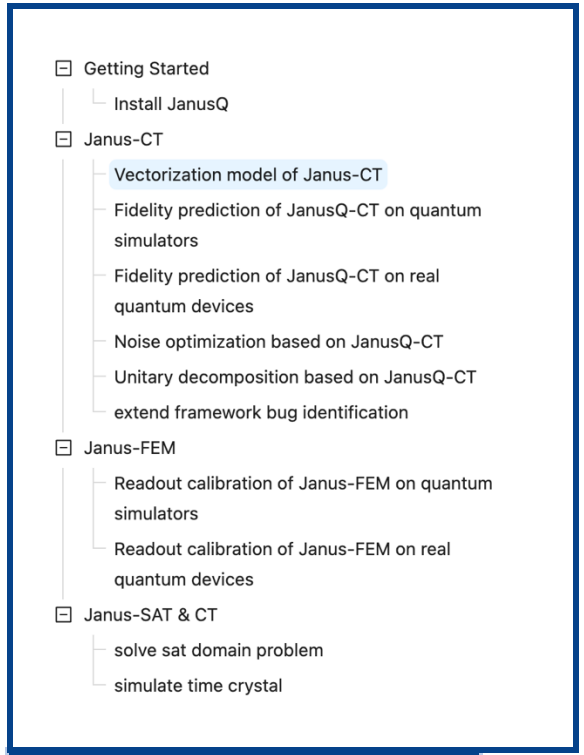
```
from janusq.application.hyqsat import readCNF

readCNF("./data/cnf_examples/test/uf100-01.cnf")
```

Test Janus-Cloud

```
from janusq.objects.circuit import Circuit
from janusq.cloud_interface import submit, get_result
qc = Circuit([], n_qubits = 3)
qc.h(0, 0)
result = submit(circuit=qc, label= 'GHZ', shots= 3000,
run_type='simulator', API_TOKEN='')
result = get_result(result['data']['result_id'],
run_type='simulator', result_format='probs')
print(result)
```

Document and Example

On: **JanusQ/examples** or **https://janusq.github.io/tutorials/Demonstrations**



Include 8 examples

In [1]:

```
%matplotlib inline

import os
os.chdir('..')

from analysis.vectorization import RandomwalkModel, extract_device

from data_objects.random_circuit import random_circuits, random_circuit
from data_objects.backend import GridBackend

import random
import numpy as np
```

URL

Vectorization Model Of Janus–CT

Author: Siwei Tan

Date: 7/4/2024

Based on "QuCT: A Framework for Analyzing Qua

In the current Noisy Intermediate–Scale Quantur
programs. Current analysis methods exhibit eithe
tradeoff, we propose Janus–CT, a unified framew
vectorize each gate with each element, quantitati
model, we can develope multiple downstream m
the vectorization model of Janus–CT.

Document of functions

```
def solve_by_minisat(cnf_file, save=False, result_dir=".", verb=1, cpu_lim=0, mem_lim=0, strictp=False):
    '''
    description: using minisat method to solve sat domain problem.
    param {str} cnf_file: input a cnf file, which needs to be solve.
    param {bool} save: weather save result in result dir.
    param {str} result_dir: save result in result dir.
    param {bool} verb: weather print log.
    param {int} cpu_lim: cpu limit(core).
    param {int} mem_lim: memory limit(MB).
    param {bool} strictp: weather strict.
    '''
    if verb:
        verb = 1
```

浙江大學
ZHEJIANG UNIVERSITY

Thanks for listening!