



浙江大學
ZHEJIANG UNIVERSITY

WELCOME TO JanusQ V2.0 TUTORIAL

Session 2 : Installing JanusQ



<https://janusq.github.io/tutorials/>

College of Computer Science and
Technology,
Zhejiang University

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>



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

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

Installation From Source Code



Step 1. clone the source code

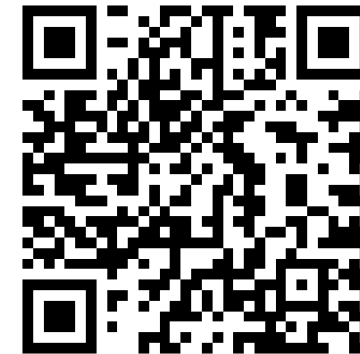
```
git clone git@github.com:JanusQ/JanusQ.git
```

Step 2. install requirements (virtual environment is preferred)

```
cd ./JanusQ  
pip install -r requirements.txt
```

Step 3. compile Janus-SAT

```
cd ./JanusQ/hyqsat  
cmake .  
make install  
cp libm* ../janusq/hyqsat  
cp minisat_core ../janusq/hyqsat
```



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

implemented based on C++.

Testing JanusQ



On “examples/1-1.install_janusq.ipynb”

Test Janus-CT

```
from janusq.analysis.vectorization import *  
  
vec_model = RandomwalkModel()  
vec_model.train()
```

Test Janus-SAT

```
from janusq.hyqsat import readCNF  
  
readCNF("./examples/cnf_examples/test/uf100-  
01.cnf")
```

Test Janus-FEM

```
from janusq.optimizations.readout_mitigation.fem  
import EnumeratedProtocol  
  
protocol = EnumeratedProtocol(4)
```

Test Janus-Cloud


```
from janusq.data_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



浙江大學
ZHEJIANG UNIVERSITY

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

 Janus Quantum

[JanusQ](#) [Demo](#) [Resources](#) [Github](#) [Team](#)

In [1]:

- Getting Started
 - Install JanusQ
- Janus-CT
 - Vectorization model of Janus-CT
 - Fidelity prediction of JanusQ-CT on quantum simulators
 - Fidelity prediction of JanusQ-CT on real quantum devices
 - Noise optimization based on JanusQ-CT
 - Unitary decomposition based on JanusQ-CT
 - extend framework bug identification
- Janus-FEM
 - Readout calibration of Janus-FEM on quantum simulators
 - Readout calibration of Janus-FEM on real quantum devices
- Janus-SAT & CT
 - solve sat domain problem
 - simulate time crystal

Include 8 examples

```
%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
```

Vectorization Model Of Janus-CT

Author: Siwei Tan

Date: 7/4/2024

Based on "[QuCT: A Framework for Analyzing Quantum](#)"

In the current Noisy Intermediate-Scale Quantum programs. Current analysis methods exhibit either tradeoff, we propose Janus-CT, a unified framework to vectorize each gate with each element, quantitatively model, we can develop multiple downstream models for the vectorization model of Janus-CT.



URL

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: whether save result in result dir.
    param {str} result_dir: save result in result dir.
    param {bool} verb: whether print log.
    param {int} cpu_lim: cpu limit(core).
    param {int} mem_lim: memory limit(MB).
    param {bool} strictp: whether strict.
    """
    if verb:
        verb = 1
```




浙江大學
ZHEJIANG UNIVERSITY

WELCOME TO JanusQ V2.0 TUTORIAL

Session 2 Janus/Taiyuan Cloud: Quantum Cloud Platform



<https://janusq.github.io/tutorials/>

College of Computer Science and
Technology,
Zhejiang University

Taiyuan (太元) & Janus



Tai Yuan is a Chinese goddess 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



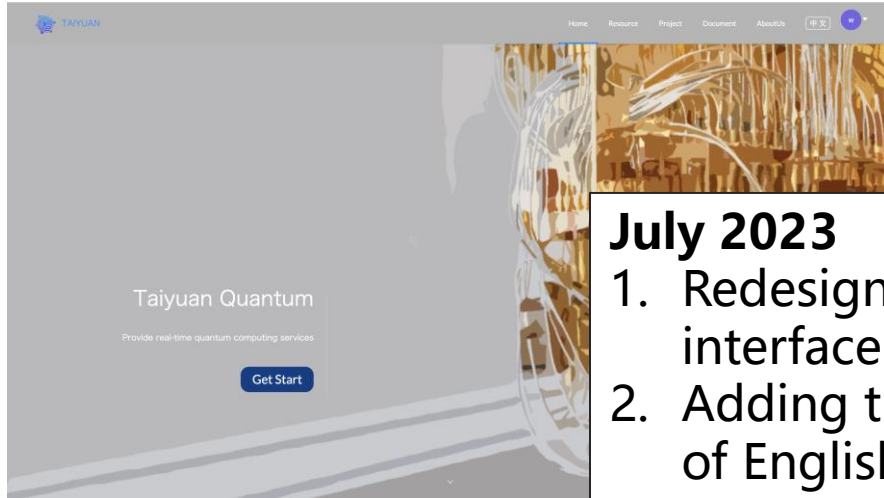
Janus is a two-faced god of the ancient Romans. He is the god of beginning, transition, time, change, dualism, and end. He has two faces simultaneously.

Development History of Janus Cloud



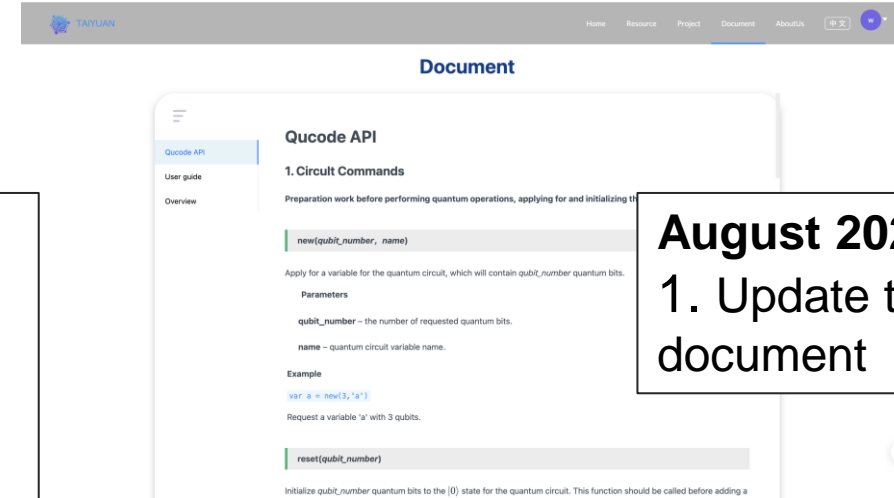
浙江大学
ZHEJIANG UNIVERSITY

Four updates since July 2023



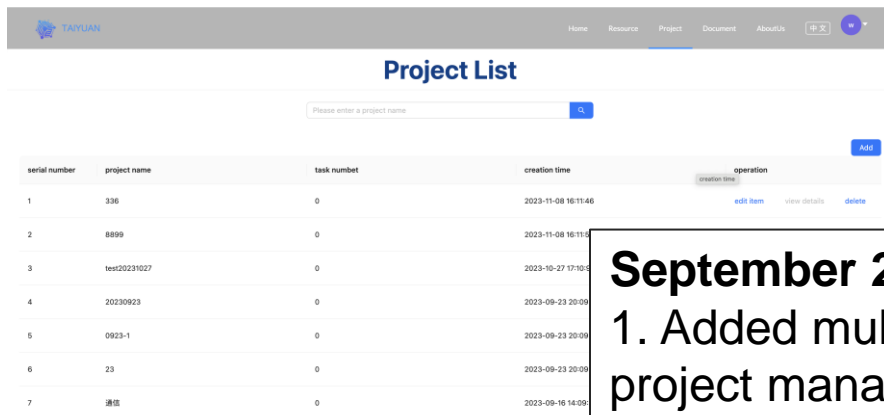
July 2023

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



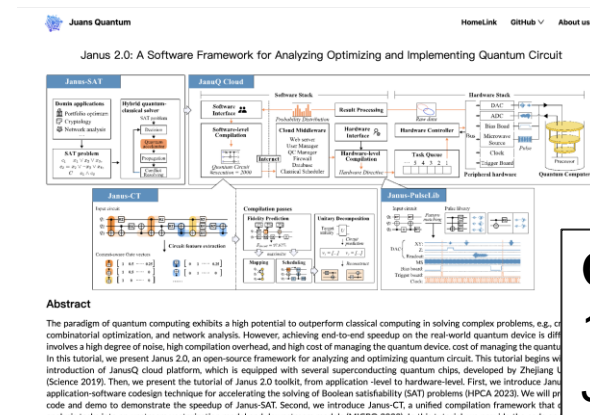
August 2023

1. Update the document



September 2023

1. Added multi-level project management.



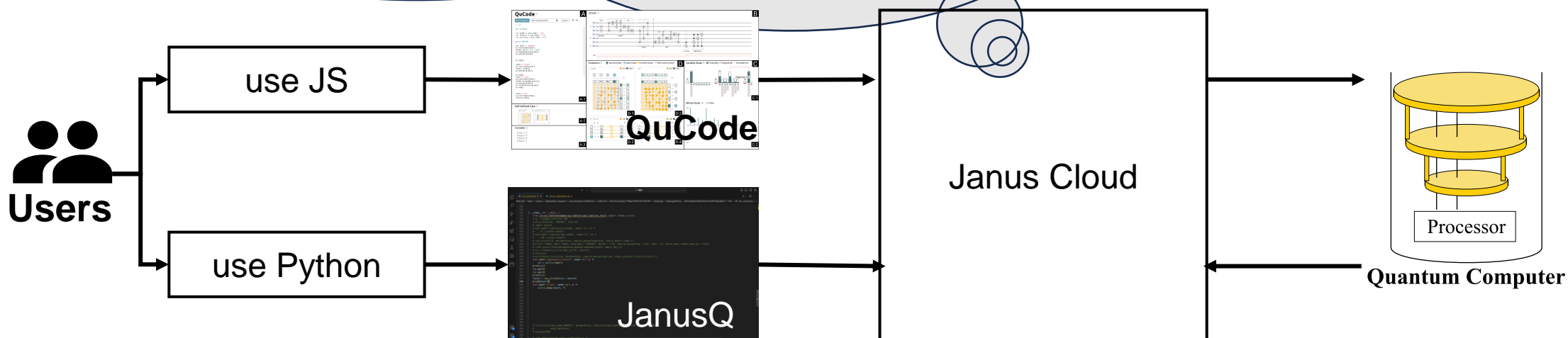
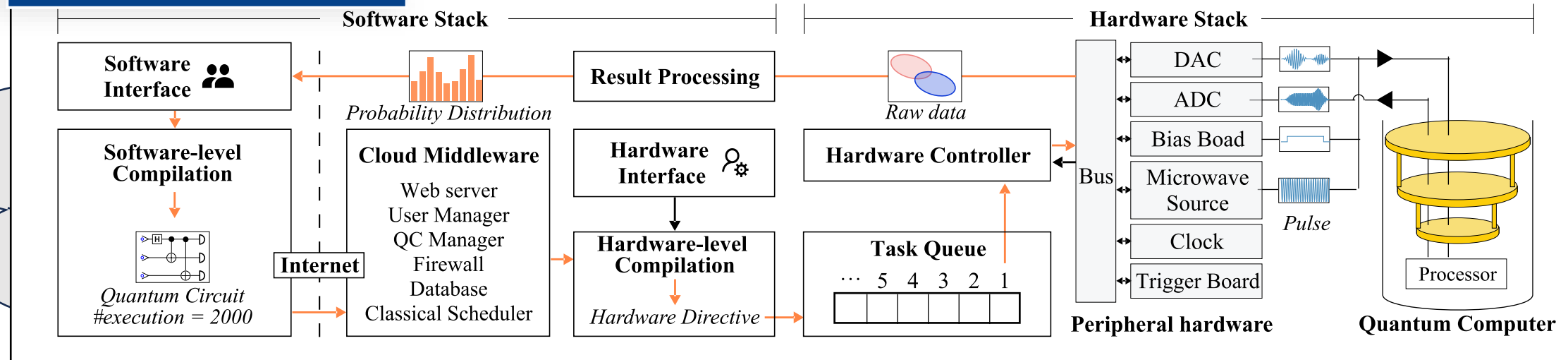
October 2023

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

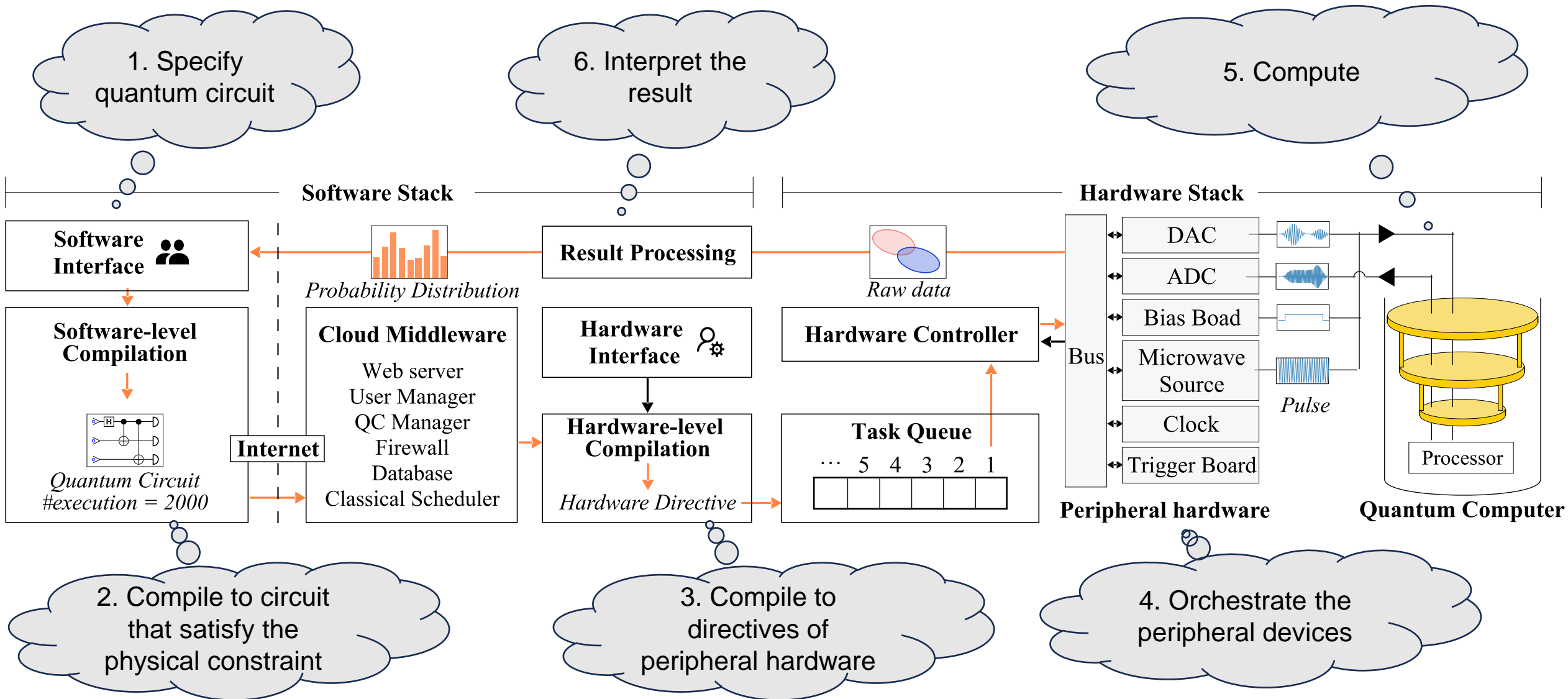
What we can do on Janus Platform



Janus Cloud



What we can do on Janus Platform

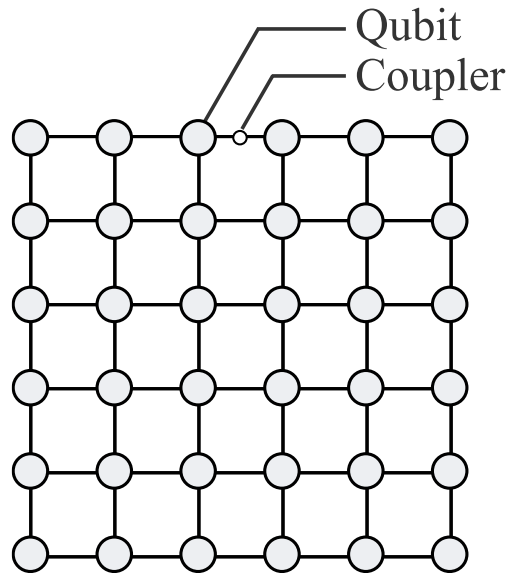


Different compared to other quantum cloud platforms

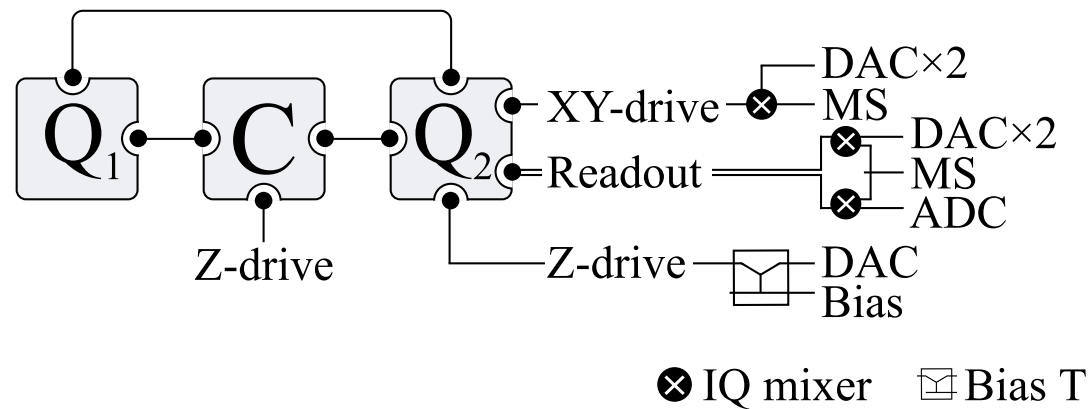


Custom hardware & more flexibility to improve the fidelity

Achieving 99% accuracy in the medical image classification.



(a) Processor topology.



(b) Sketch of two qubits and a coupler.

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



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

Creating an Account



Go to Janus Cloud

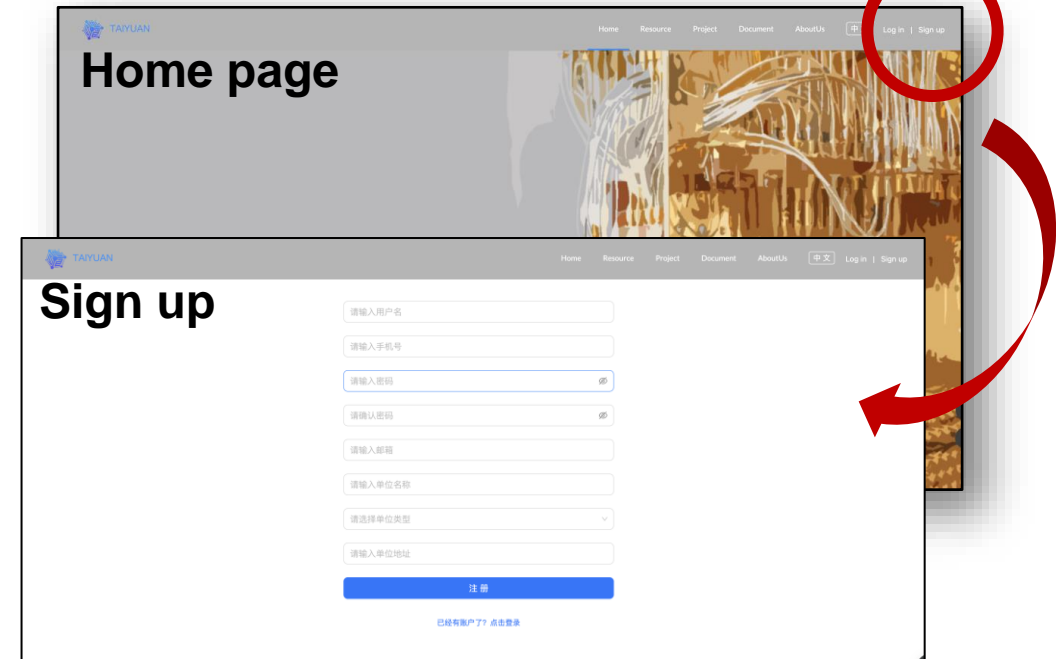
Open <http://janusq.zju.edu.cn/home>



QR code

Sign up

Click sign up button



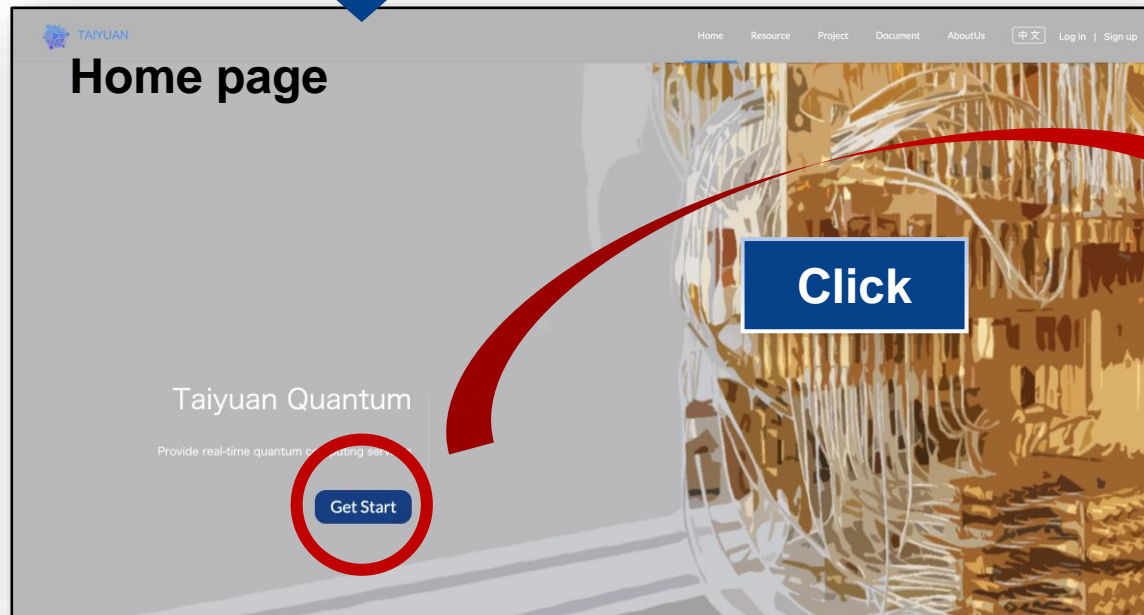
- After signing up, you will get an API key for task submission.
- **The quantum computer can be accessed on the website during the tutorial !!**

Running Program On the Website



Submitting on website

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



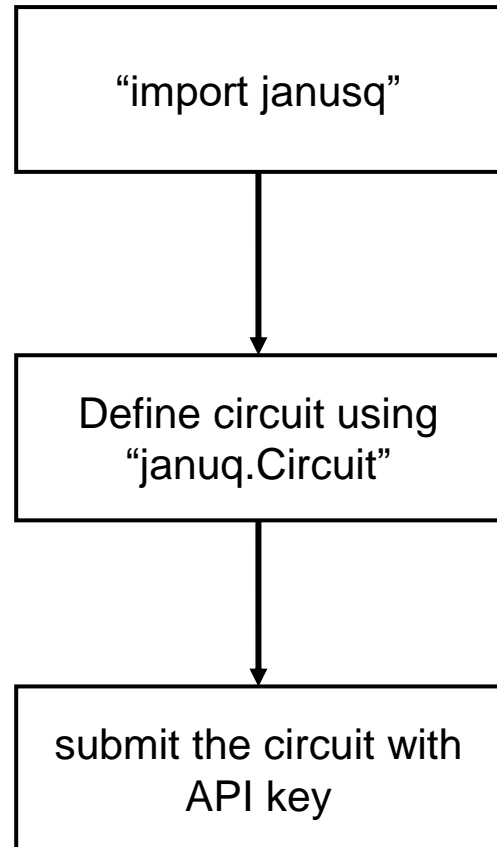
Click

Programming on the Code editor



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

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

Run Program
QFT manipulation
Export

// QFT

qc.reset(0)

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

Quantum circuit view

Evolution view

InvQFT

0.35 | 0.85

D-1

QFT

0.35 | 0.85

D-2

A = R >= 8

1.00 | 0.85

D-3

High freq++

1.00 | 0.85

D-4

Variable State

S

C-1

R

C-1

A

C-1

Whole State

Phase

C-2

QuCode: Online Quantum Program Editor



Program editor

Run Program

QFT manipulation

Provide 14 examples:

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

```
// 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);

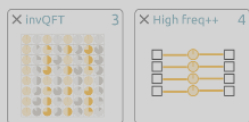
qc.nop();

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();
```

Self-defined Gate



Console

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

A

Circuit

B

A-1

A-2

A-3

D-1

D-3

D-2

D-4

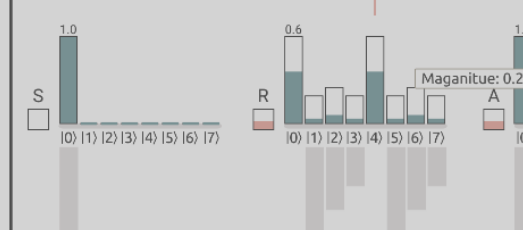
C

C-1

C-2

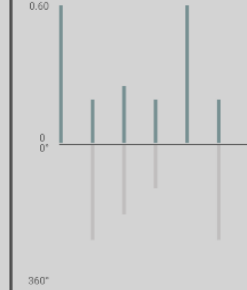
Variable State

Probability Magnitude Entanglement

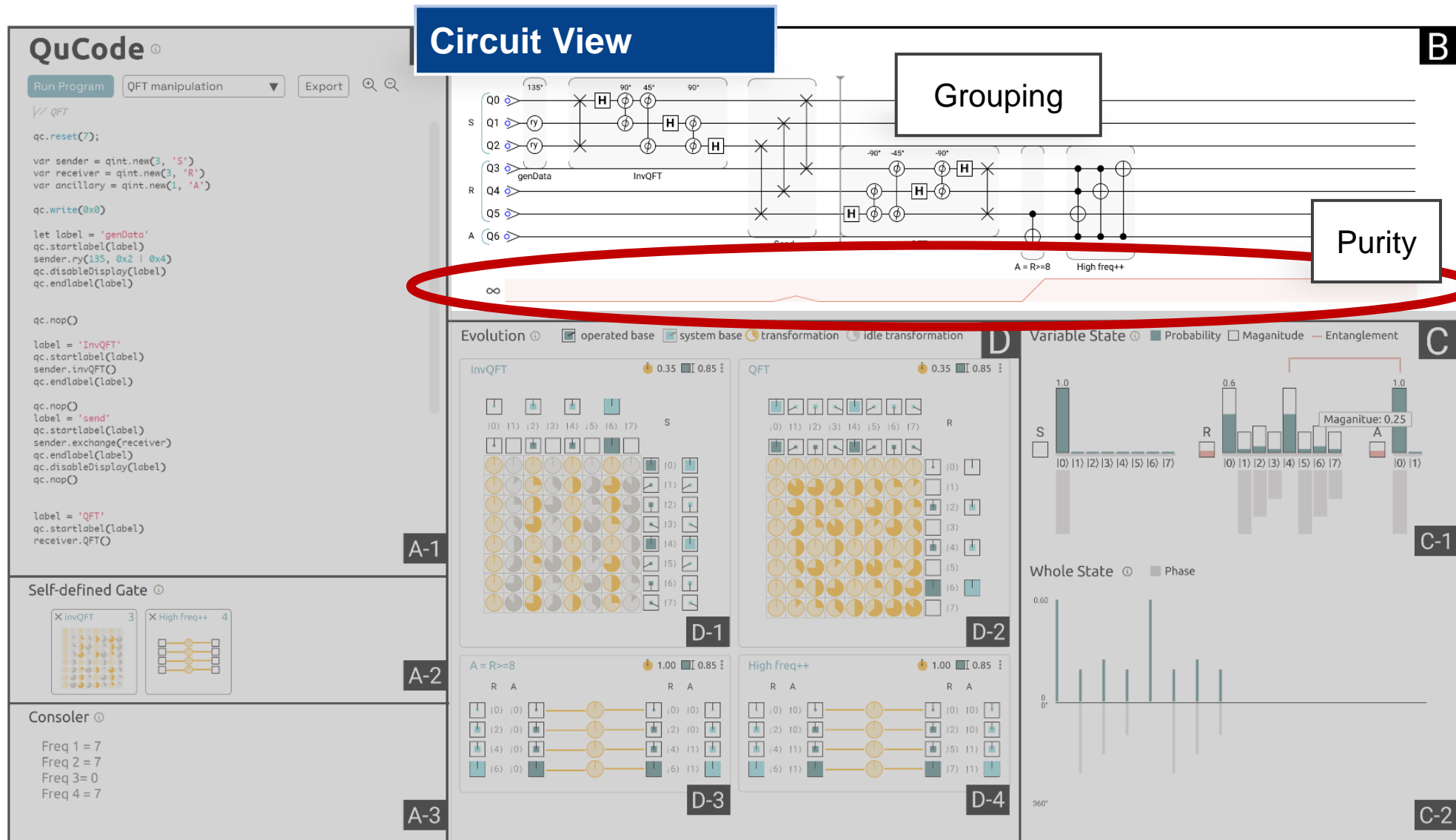


Whole State

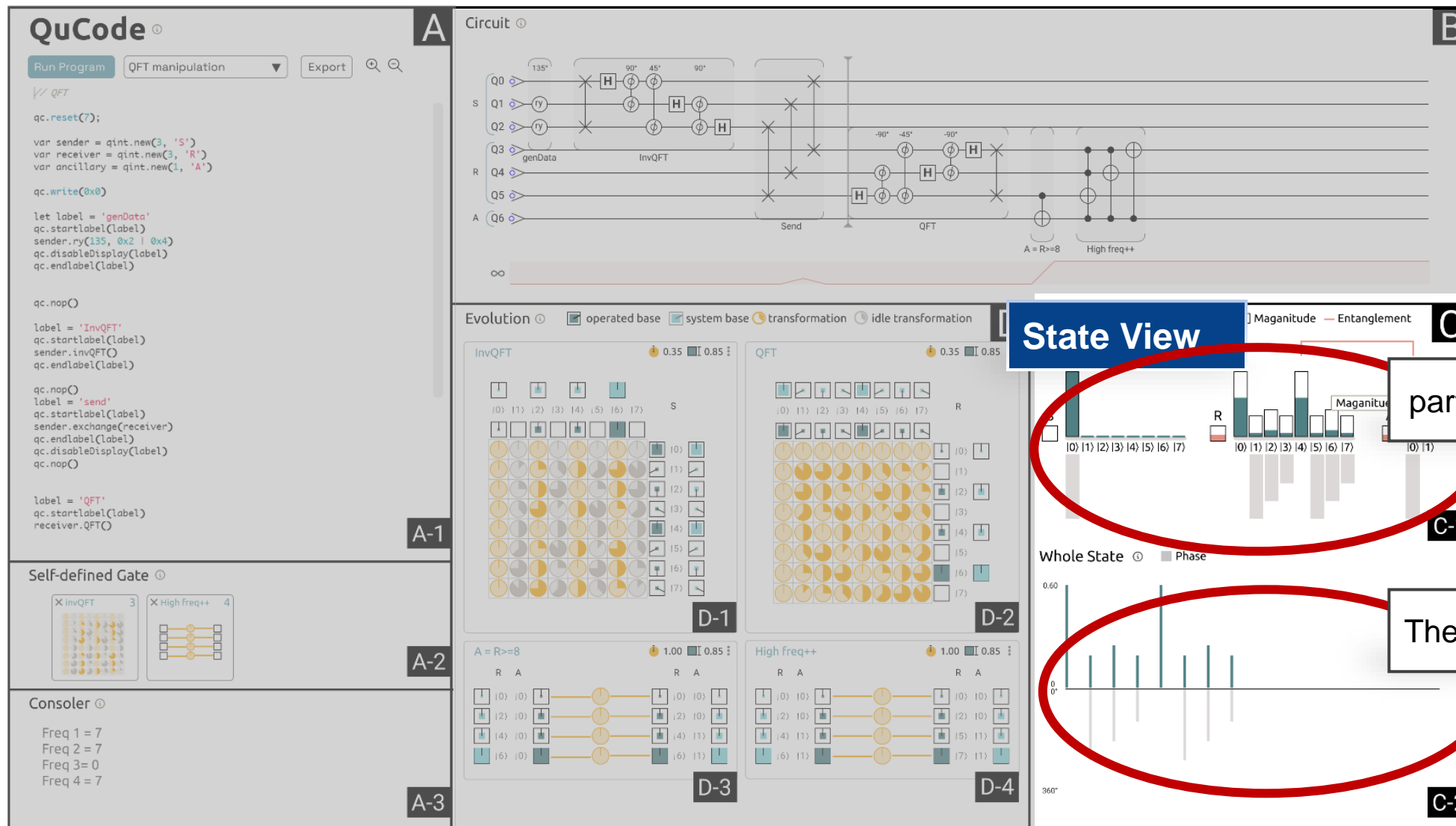
Phase



QuCode: Online Quantum Program Editor



QuCode: Online Quantum Program Editor



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()
qc.endLabel(label)
```

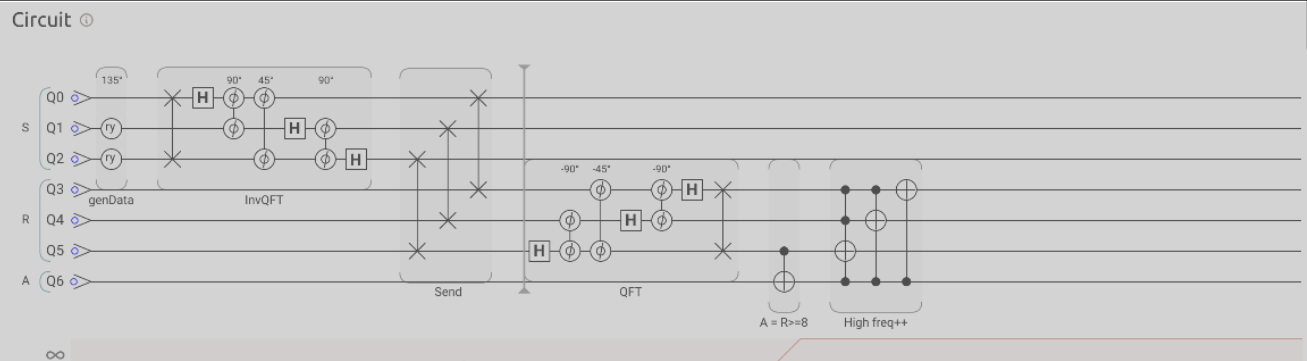
Self-defined Gate

InvQFT 3 High Freq++ 4

Console

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

Circuit



Evolution View

transformation idle transformation

InvQFT 0.35 0.85

QFT 0.35 0.85

A = R = 8 1.00 0.85

High freq++ 1.00 0.85

Variable State

Probability Maganitude Entanglement

S R A

Phase

Maganitue: 0.25

- Visualize the unitary
- Visualize the change of qubit states

Mathematical formulation

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

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

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

A layer of rotation gates along the x axis

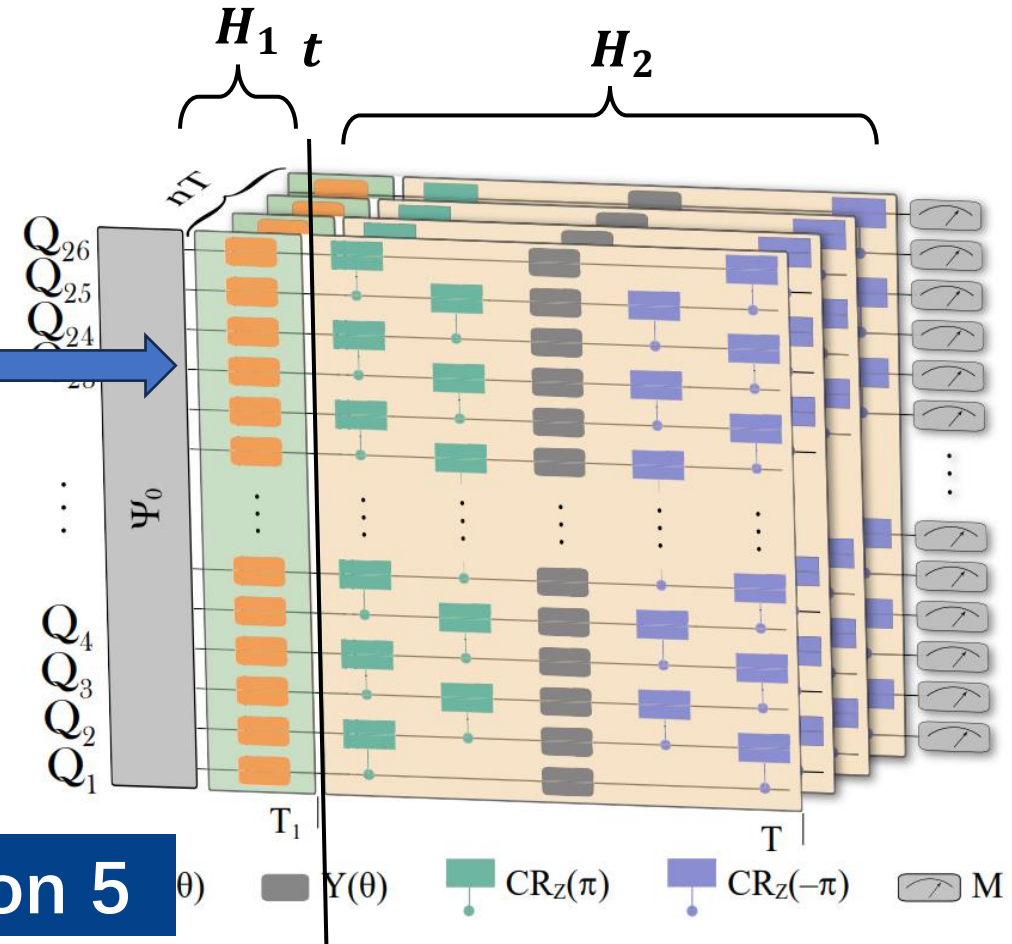
$$H_2 \equiv - \sum_k J_k \sigma_{k-1}^z \sigma_k^x \sigma_{k+1}^z$$

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

20 random disorder instances (J_k)

Introduced In Session 5

Circuit Implementation





浙江大學
ZHEJIANG UNIVERSITY

Thanks for listening!