WELCOME TO JanusQ V2.0 TUTORIAL

Session 2 : Installing JanusQ

ACES Lab
ADVANCED COMPUTING AND EMERGING SERVICE LAB

JanusQ Cloud

浙江大学 计算机科学与技术学院
COLLEGE OF COMPUTER SCIENCE AND TECHNOLOGY
ZHEJIANG UNIVERSITY

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

College of Computer Science and Technology,
Zhejiang University

ASPLOS 2024

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
```

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

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

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-FEM

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

protocol = EnumeratedProtocol(4)
```

Test Janus-SAT

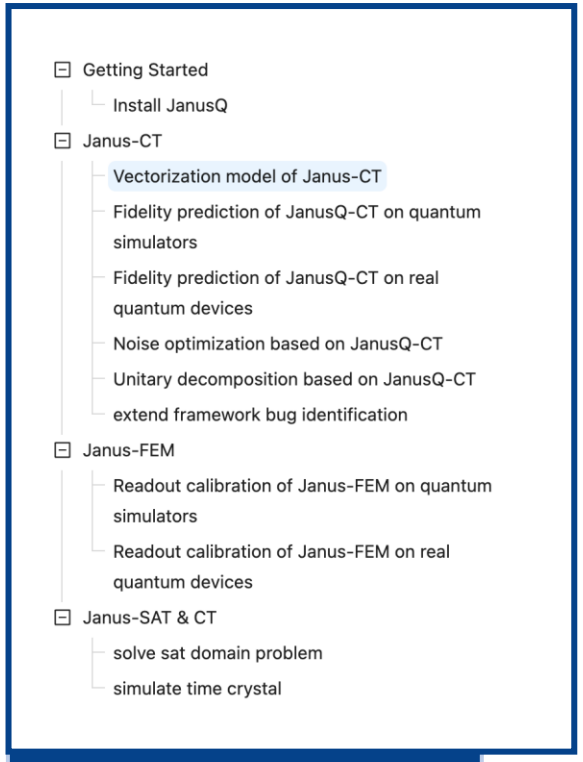
```
from janusq.hyqsat import readCNF

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

Test Janus-Cloud

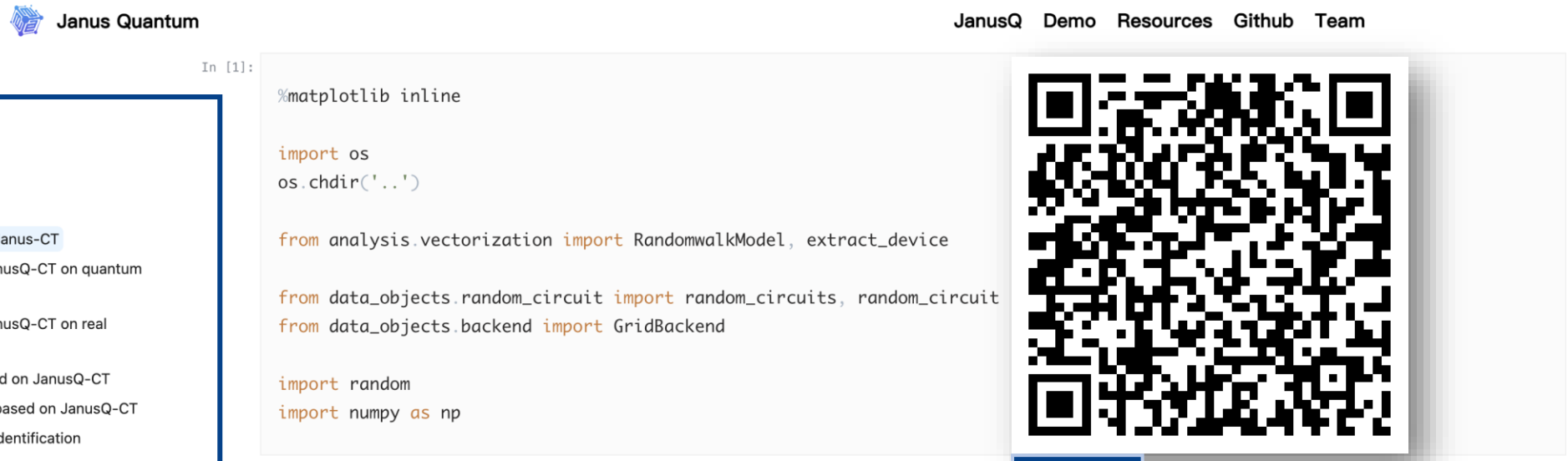
```
from janusq.data_objects.circuit import Circuit
from janusq.cloud_interface import submit, get_result
qc = Circuit([], n_qubits = 3)
qc.h(0, 0)
result = submit(circuit=qc, label= 'GHZ', shots= 3000,
run_type='simulator', API_TOKEN='')
result = get_result(result['data']['result_id'],
run_type='simulator',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

Document and Example

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



Include 8 examples



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: 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
```

WELCOME TO JanusQ V2.0 TUTORIAL

Session 2 Janus/Taiyuan Cloud: Quantum Cloud Platform

ACES Lab
ADVANCED COMPUTING AND EMERGING SERVICE LAB

JanusQ Cloud

浙江大学 计算机科学与技术学院
COLLEGE OF COMPUTER SCIENCE AND TECHNOLOGY
ZHEJIANG UNIVERSITY

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

College of Computer Science and Technology,
Zhejiang University

ASPLOS 2024

Taiyuan (太元) & Janus

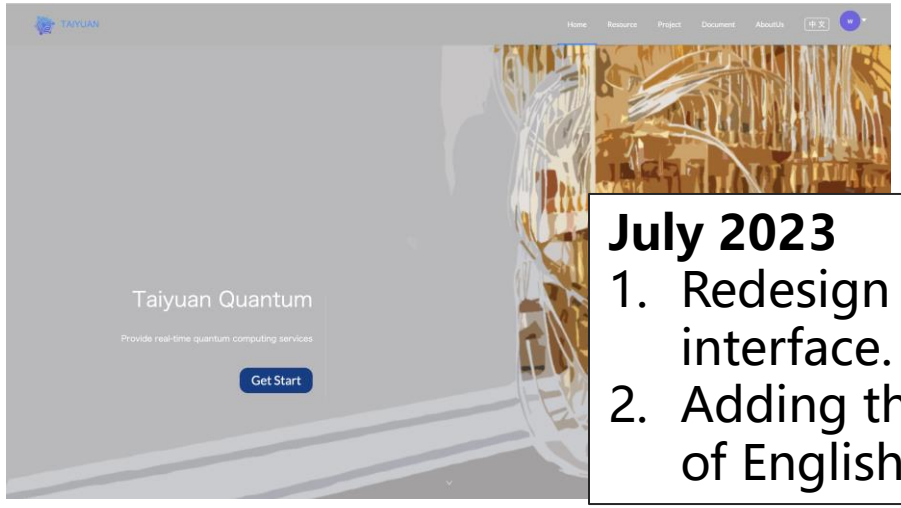
Yin and Yang

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.

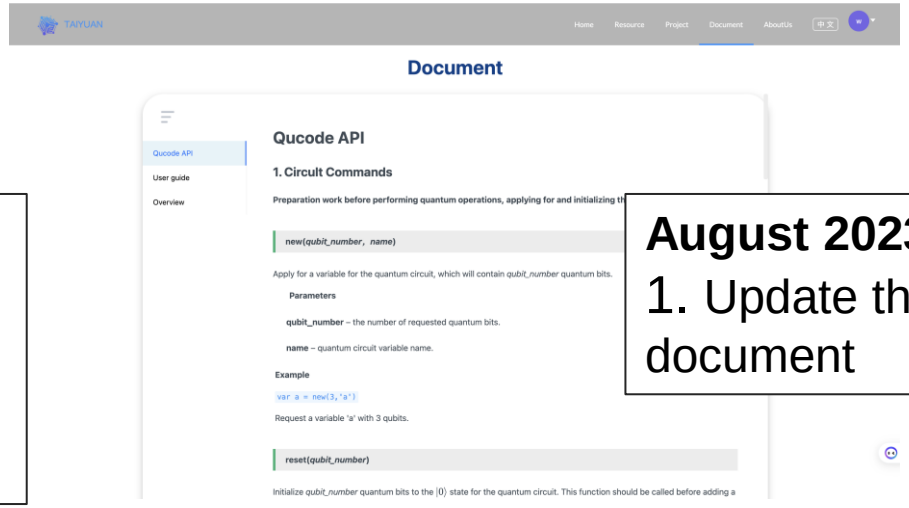
Development History of Janus Cloud

Four updates since July 2023



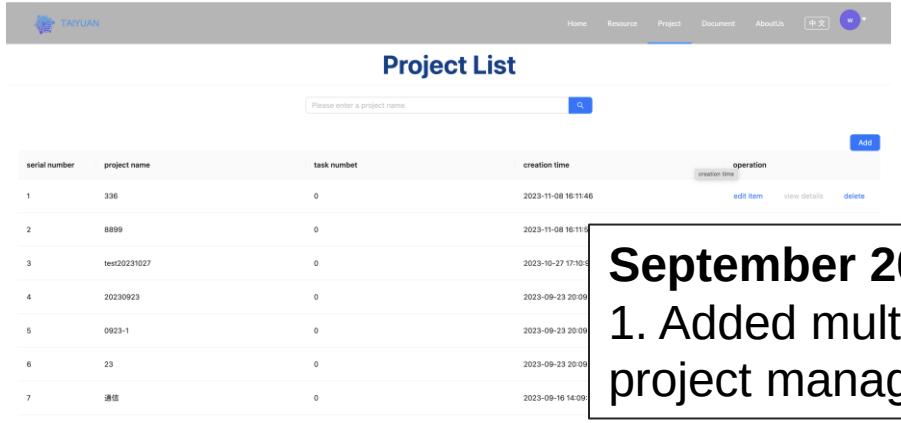
July 2023

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



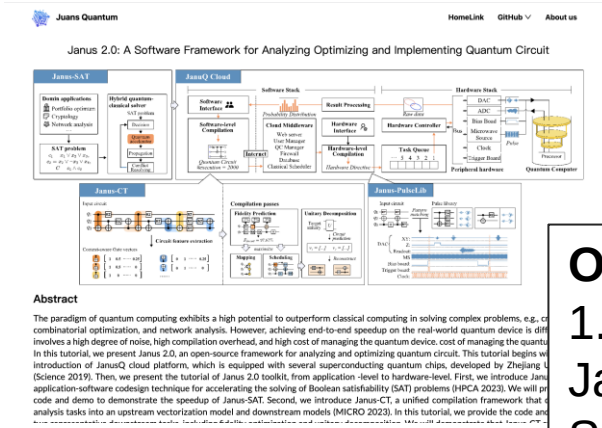
August 2023

1. Update the document



September 2023

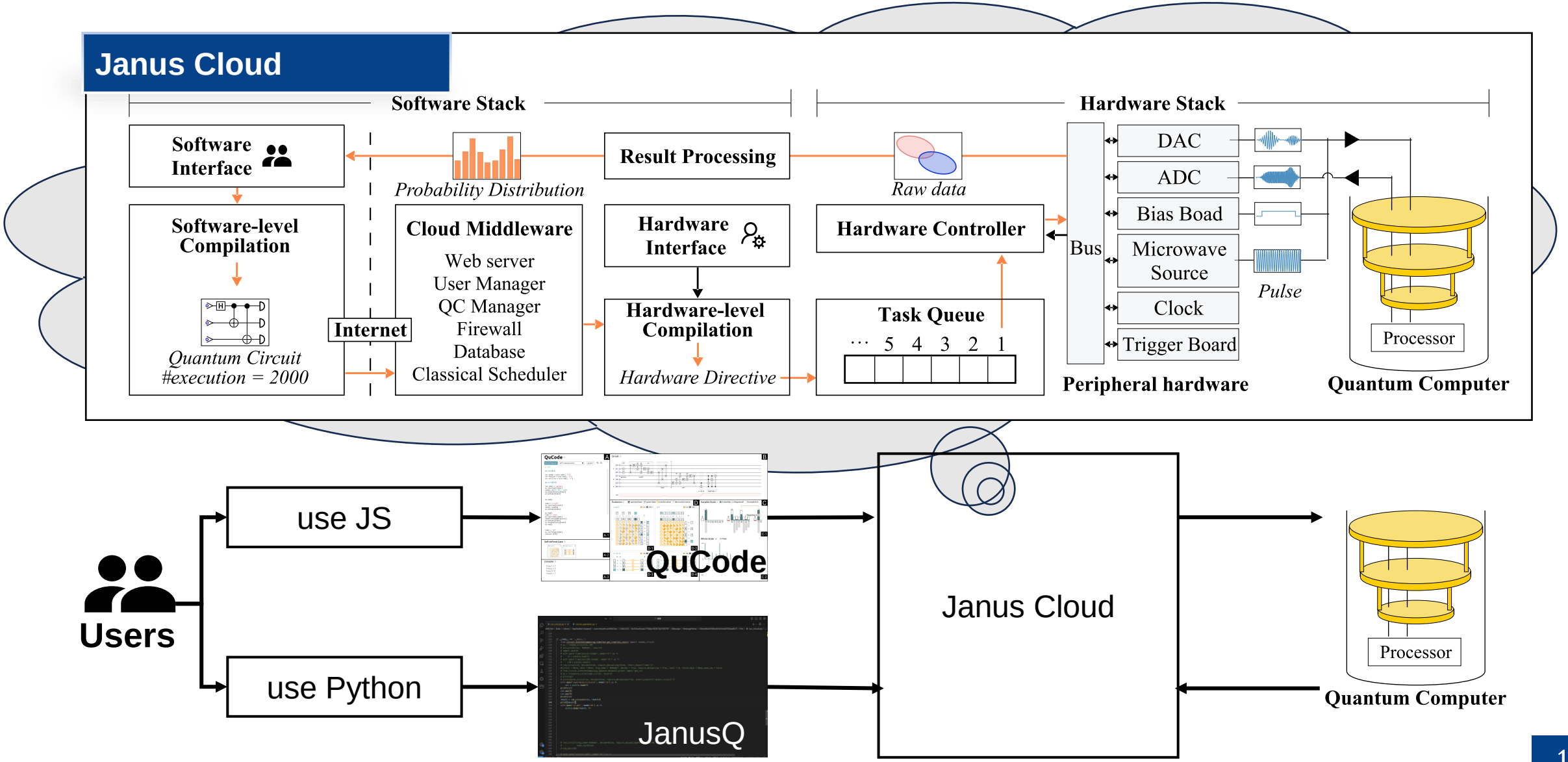
1. Added multi-level project management.



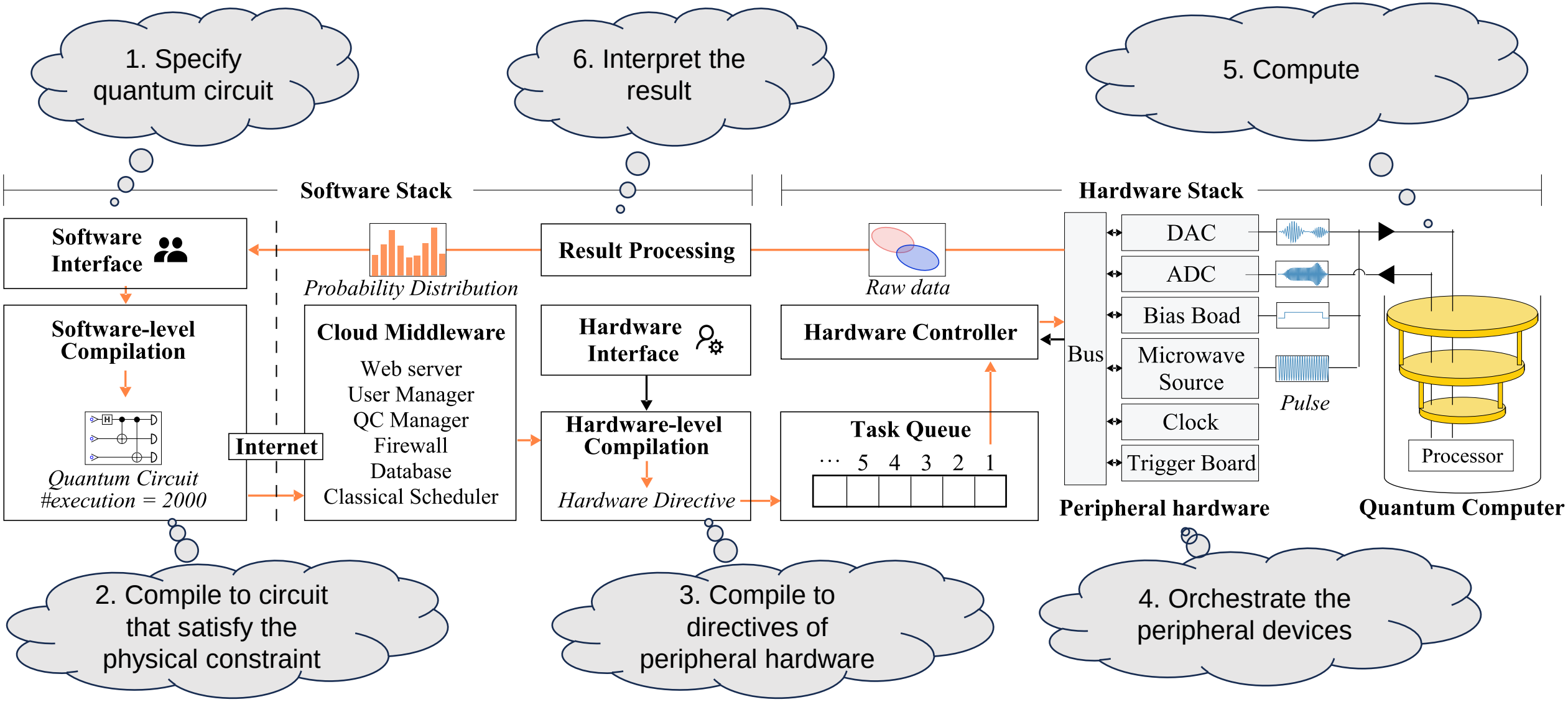
October 2023

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

What we can do on Janus Platform



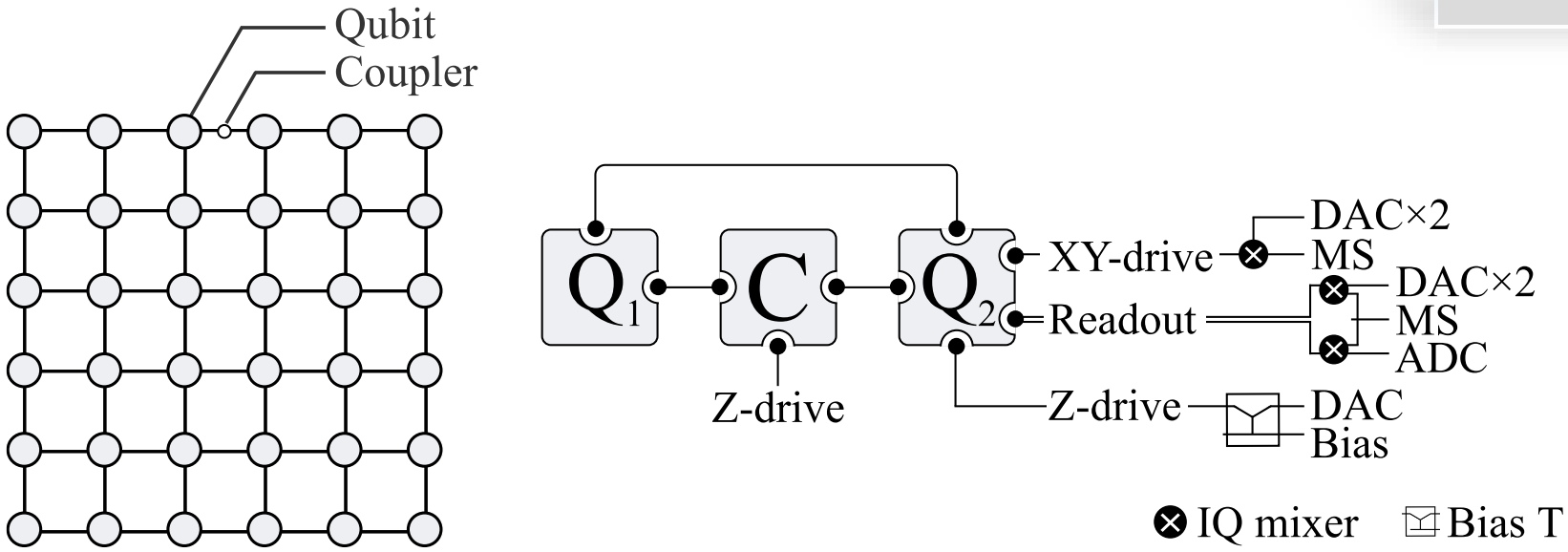
What we can do on Janus Platform



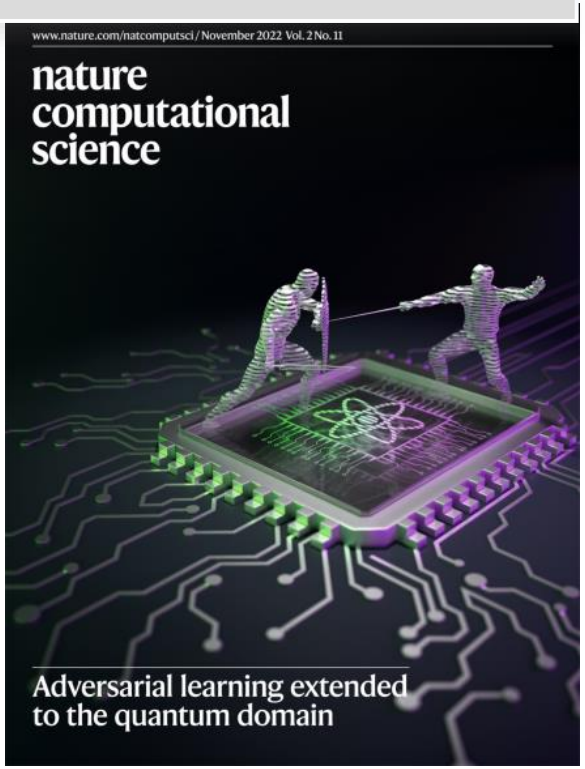
Different compared to other quantum cloud platforms

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

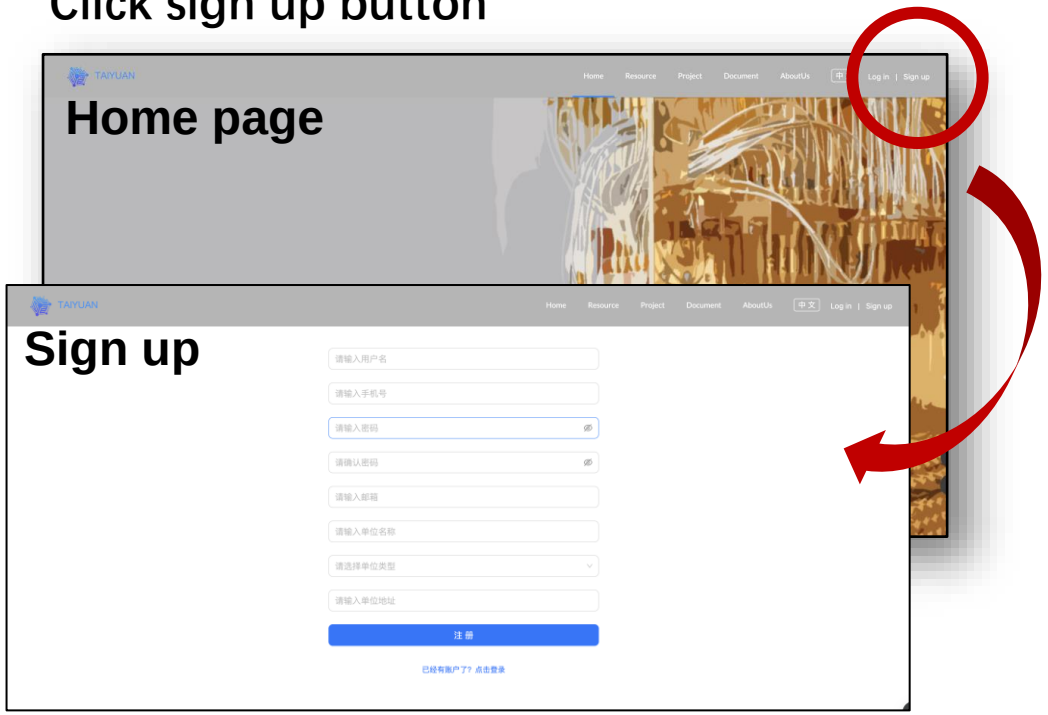
Go to Janus Cloud

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

QR code

Sign up

Click sign up button

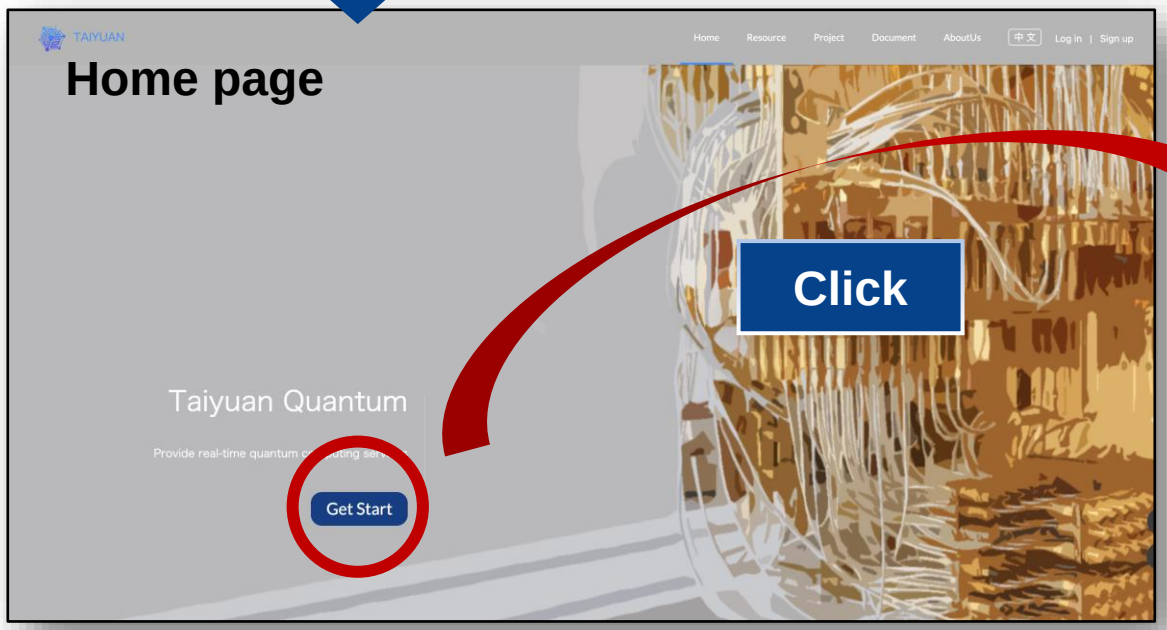


- After signing up, you will get an API key for task submission.
- **The quantum computer can be accessed on the website during the tutorial !!**

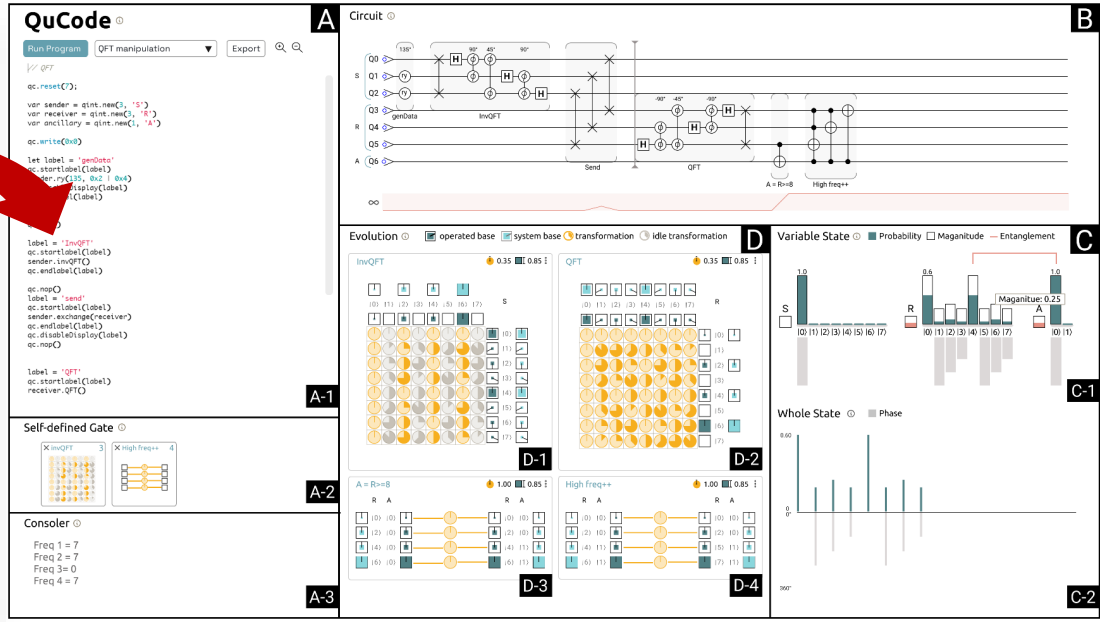
Running Program On the Website

Submitting on website

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



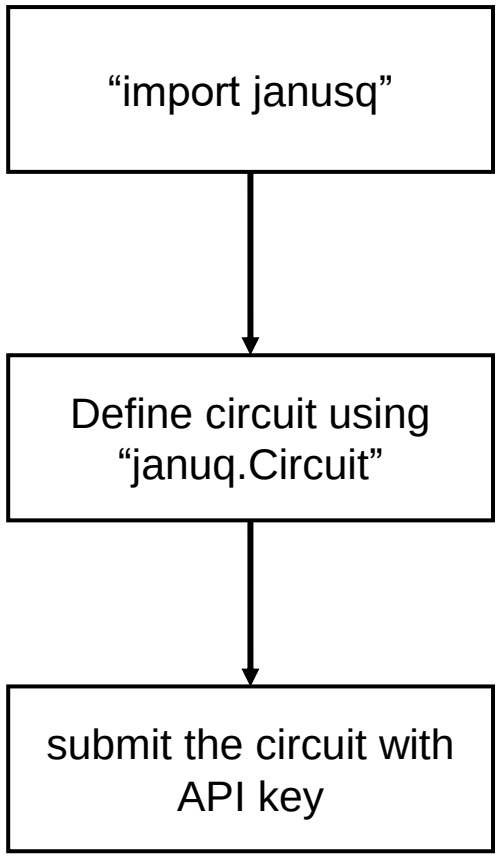
Programming on the Code editor



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

Running Program By Python API

Submitting by Python



Import package

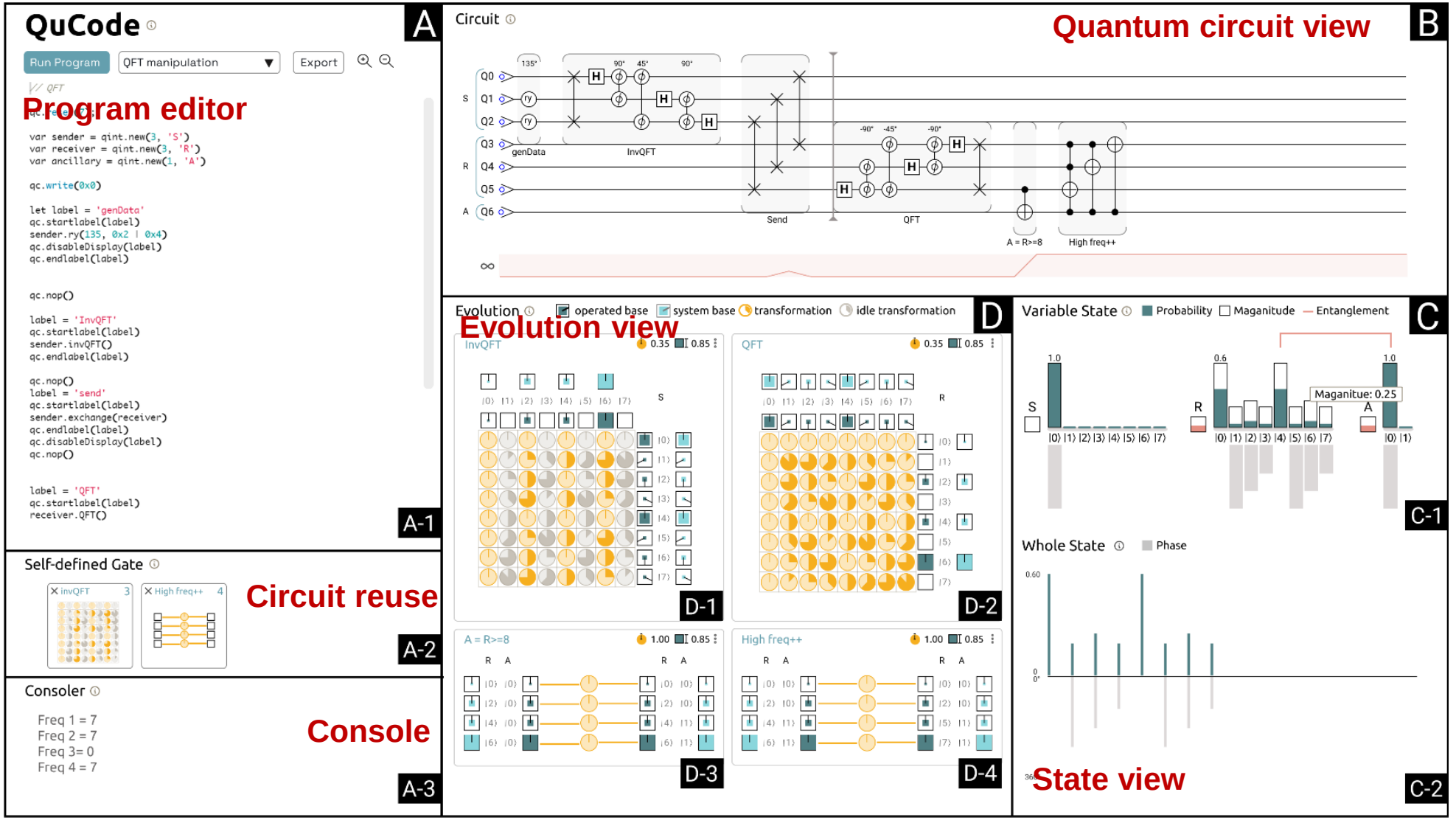
Define a circuit

Submit circuit

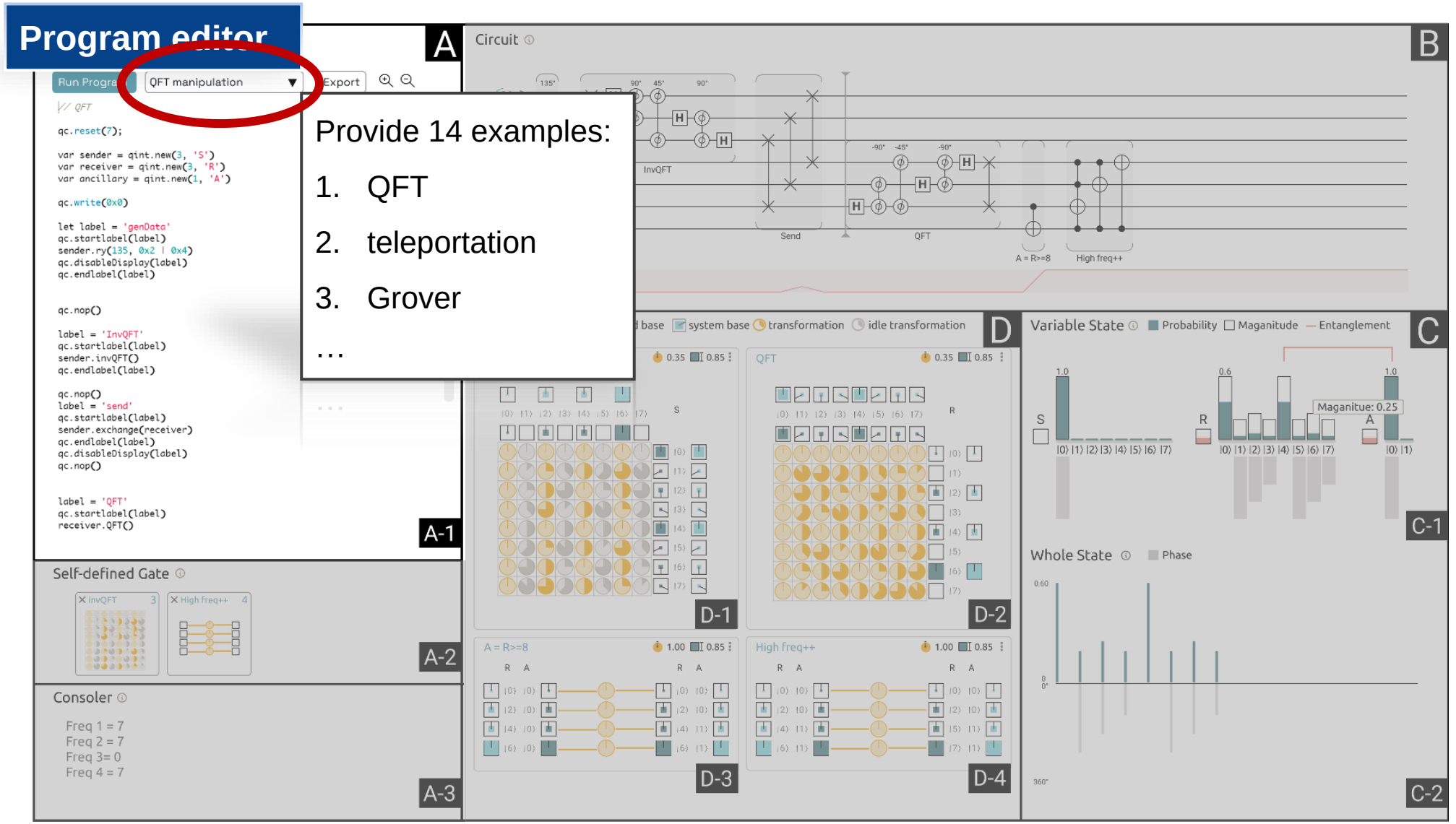
Get result

```
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)
```

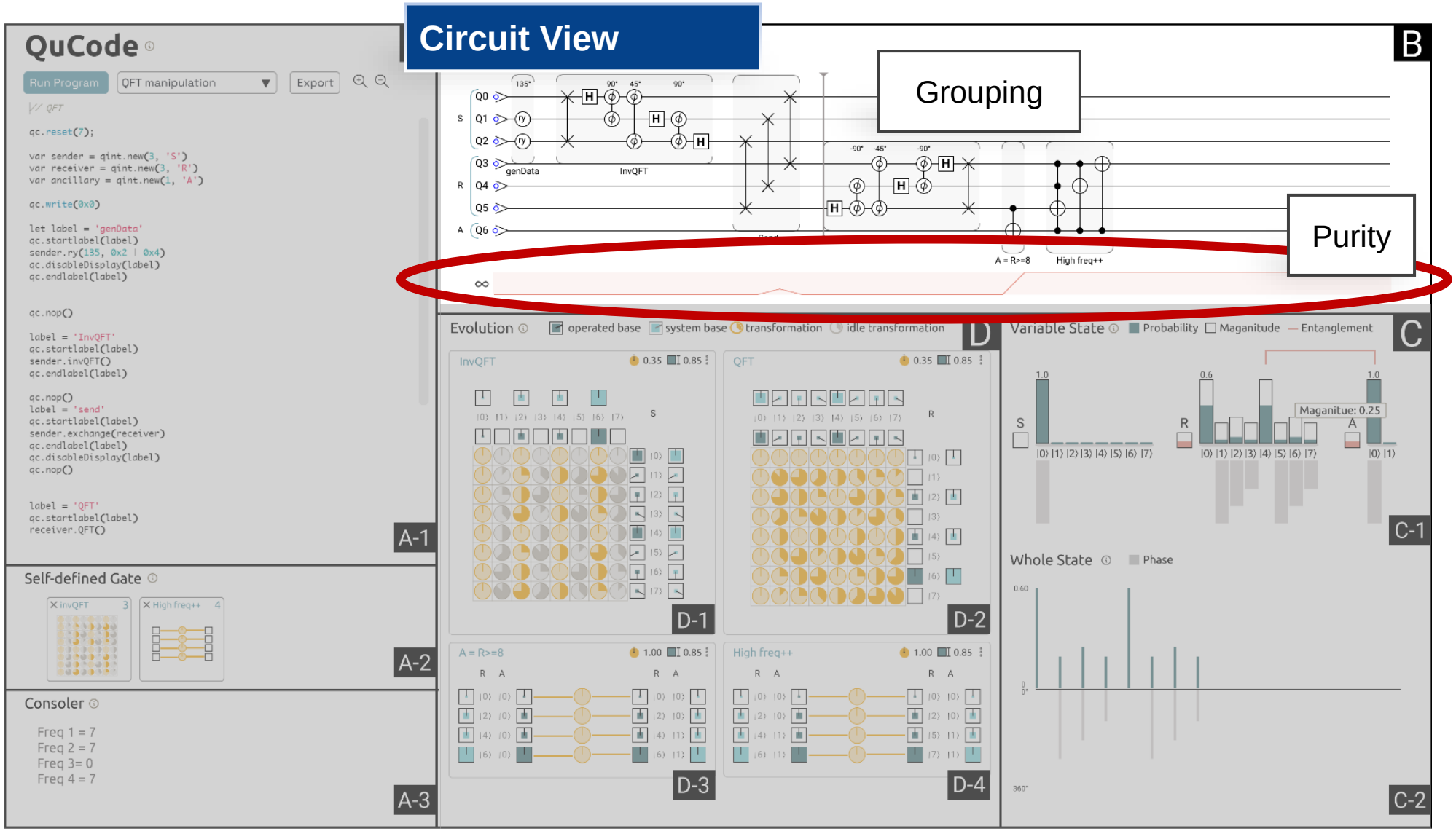
QuCode: Online Quantum Program Editor



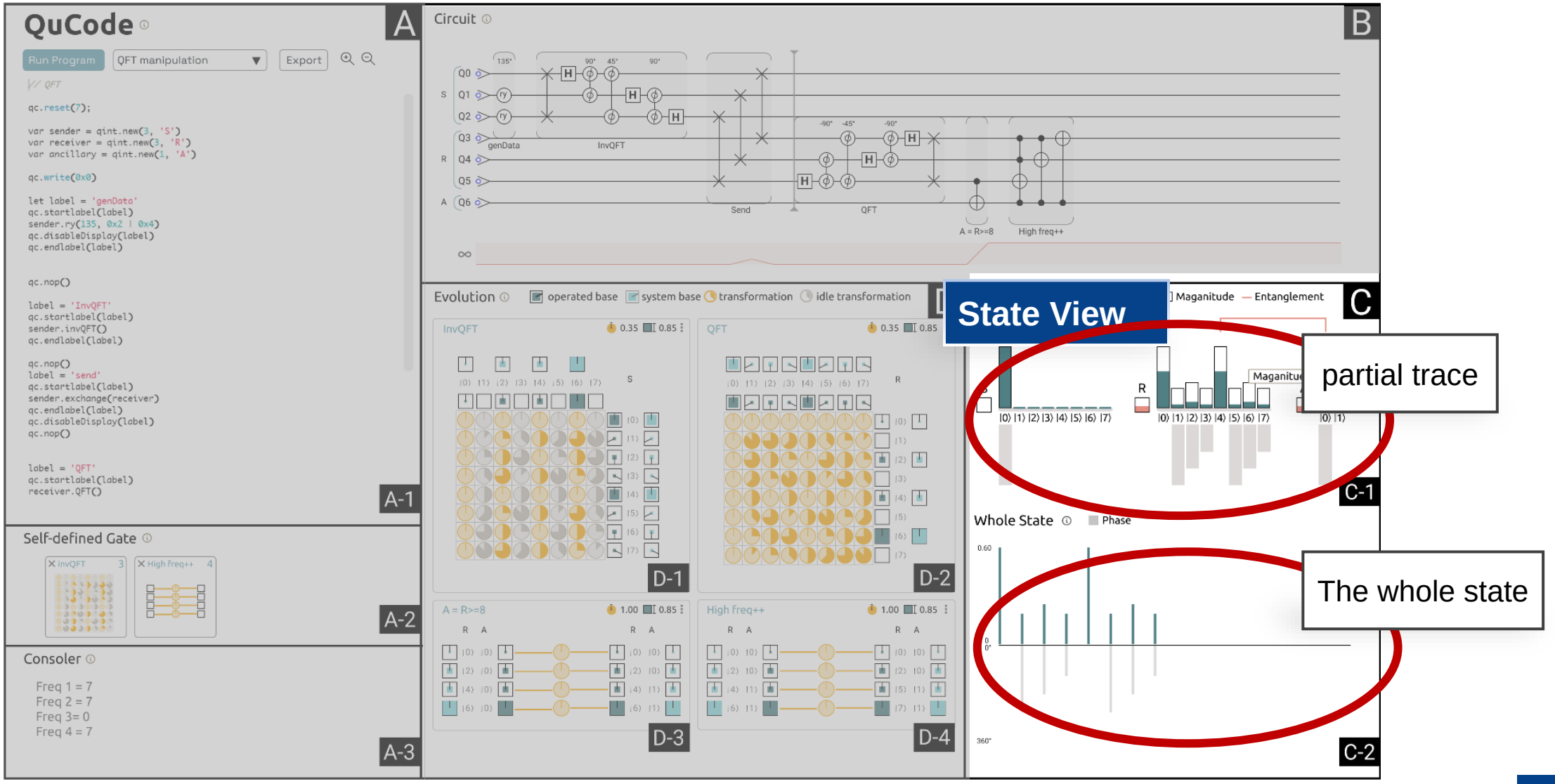
QuCode: Online Quantum Program Editor



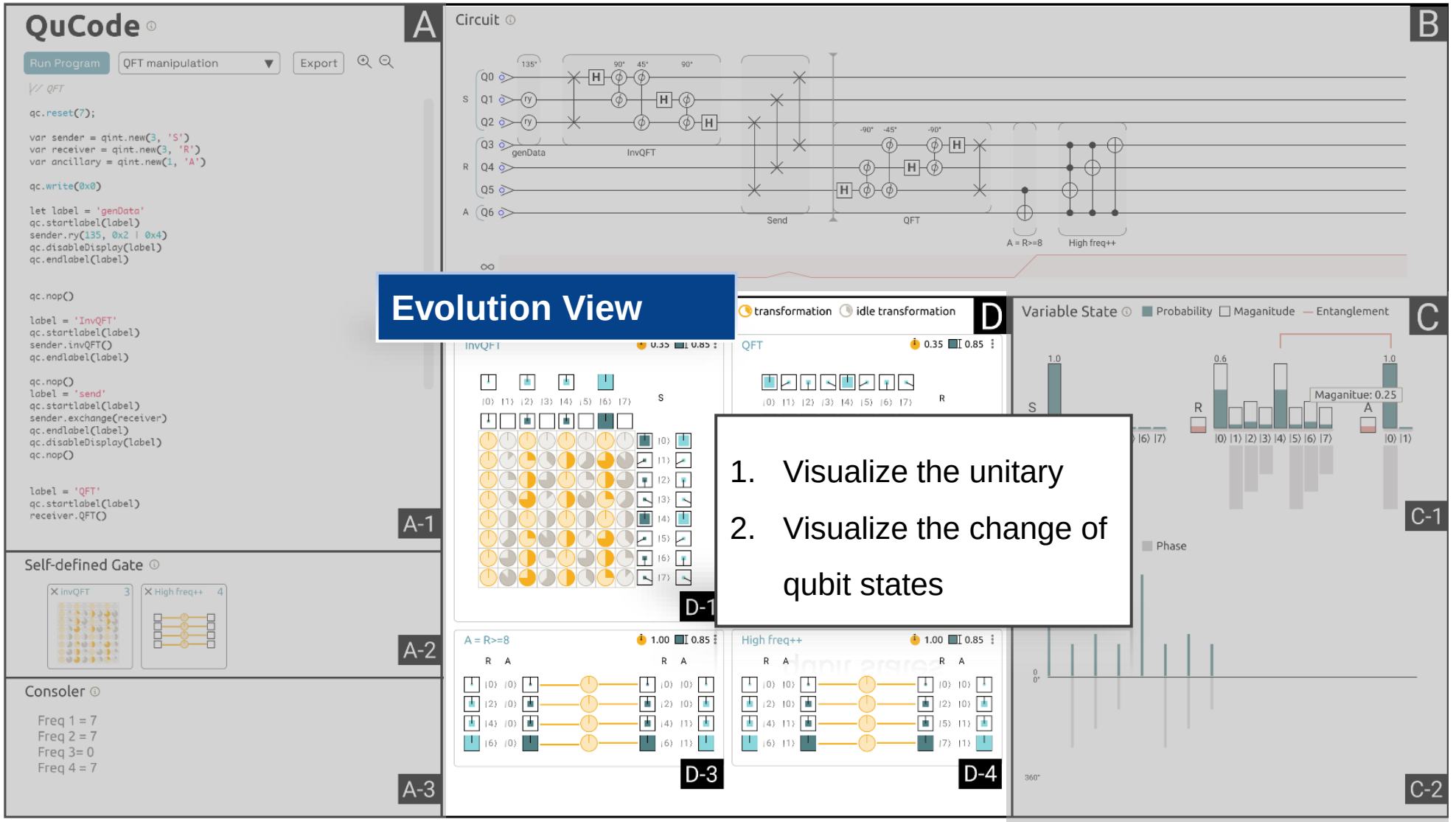
QuCode: Online Quantum Program Editor



QuCode: Online Quantum Program Editor



QuCode: Online Quantum Program Editor



Simulation of Time crystal

浙江大学 ZHEJIANG UNIVERSITY

Mathematical formulation

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

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

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

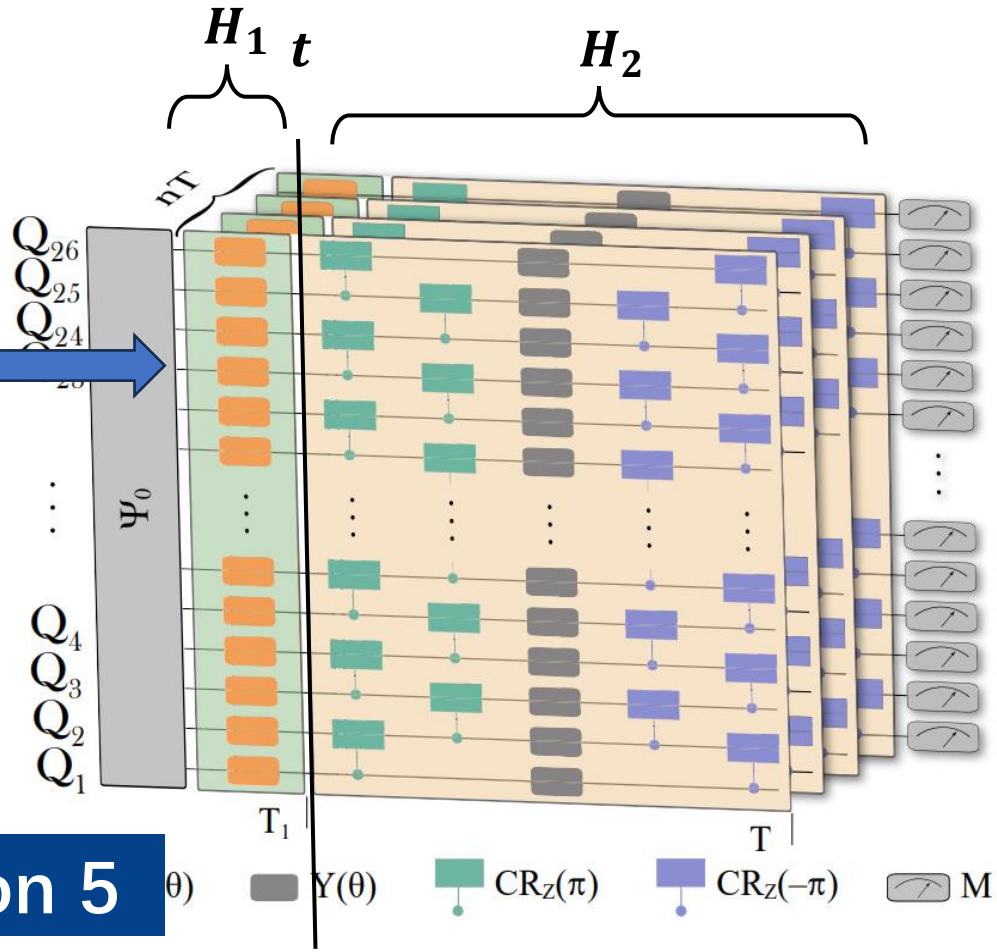
$$J_k \in [0, 2]$$

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

A layer of rotation gates along the x axis

20 random disorder instances $\{J_k\}$

Circuit Implementation



Introduced In Session 5

浙江大学
ZHEJIANG UNIVERSITY

Thanks for listening!