



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



浙江大學
ZHEJIANG UNIVERSITY

Language: Python 3, C++

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

Available platforms: Linux, Mac, 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 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



浙江大學
ZHEJIANG UNIVERSITY

Step 1. clone the source code

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

Step 2. install requirements (virtual environment is preferred)

```
cd ./JanusQ  
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/uf50-01.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



浙江大学
ZHEJIANG UNIVERSITY

On: [JanusQ/examples](#) or <https://janusq.github.io/tutorials/Demonstrations>

Janus Quantum

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
```

JanusQ Demo Resources Github Team



Vectorization Model Of Janus-CT

Author: Siwei Tan

Date: 7/4/2024

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

In the current Noisy Intermediate-Scale Quantum (NISQ) era, analyzing quantum programs is challenging due to the tradeoff between fidelity and efficiency. To address this issue, we propose Janus-CT, a unified framework that can vectorize each gate with each element, quantitatively analyze the circuit, and develop multiple downstream models. In this demonstration, we will introduce 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: 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
```

Include 8 examples



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



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



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

Development History of Janus Cloud



浙江大學
ZHEJIANG UNIVERSITY

Four updates since July 2023

Taiyuan Quantum
Provide real-time quantum computing services
Get Start

July 2023

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

Document

Quicode API

1. Circuit Commands

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

```
new(qubit_number, name)
```

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

Parameters

qubit_number – the number of requested quantum bits.

name – quantum circuit variable name.

Example

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

Request a variable 'a' with 3 qubits.

```
reset(qubit_number)
```

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

August 2023

1. Update the document

serial number	project name	task number	creation time	operation
1	338	0	2023-11-09 16:11:46	edit item view details delete
2	8899	0	2023-11-08 16:11:5	edit item view details delete
3	test20231027	0	2023-10-27 17:10:5	edit item view details delete
4	20230923	0	2023-09-23 20:09	edit item view details delete
5	0923-1	0	2023-09-23 20:09	edit item view details delete
6	23	0	2023-09-23 20:09	edit item view details delete
7	通信	0	2023-09-16 14:09	edit item view details delete

September 2023

1. Added multi-level project management.

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

Janus-SAT

Davis applications
Portfolio system
Cryptography
Quantum satisfiability

Janus-CT

Boolean satisfiability
Quantum circuit
Quantum SAT

Janus-Q Cloud

Hybrid quantum-classical solver
Software stack
Subfunctional Compilation
Cloud Middleware
Hardware stack

Janus-PulseJ

Compiler pass
Value Description
Task Order
Hardware Decoder

Abstract

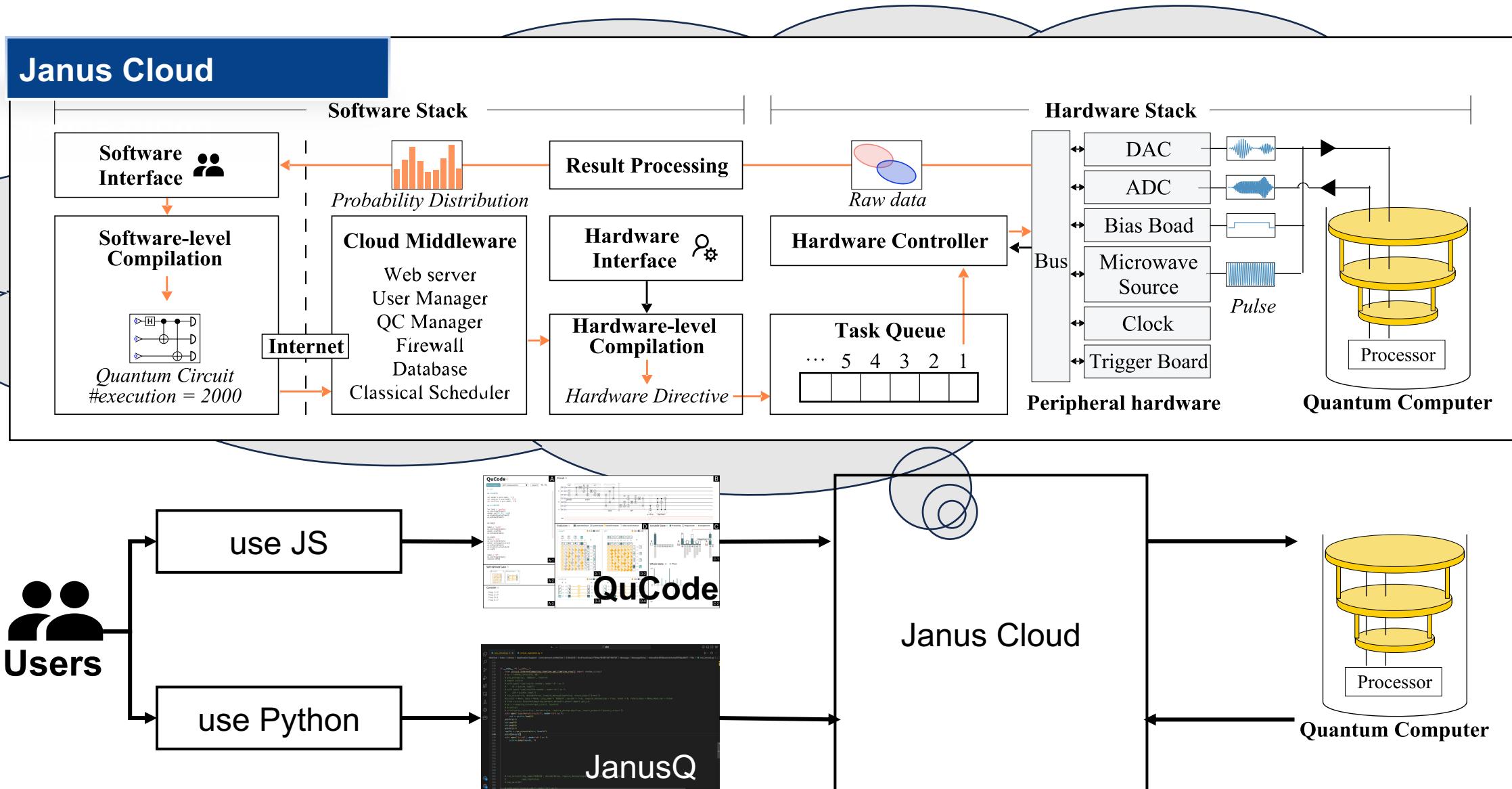
The paradigm of quantum computing exhibits a high potential to outperform classical computing in solving complex problems, e.g., combinatorial optimization, and network analysis. However, achieving and maintaining speedup on the real-world quantum device is still involved in a high degree of noise, high compilation overhead, and high cost of managing the quantum device, cost of managing the quantum circuit, and cost of solving Boolean satisfiability (SAT) problems.

In this tutorial, we present Janus 2.0, an open-source framework for analyzing and optimizing quantum circuit. This tutorial begins with the introduction of JanusQ cloud platform, which is equipped with several superconducting quantum chips, developed by Zhejiang University (Science 2019). Then, we present the tutorial of Janus 2.0 toolkit, from application-level to hardware-level. First, we introduce Janus-SAT toolkit, which is based on Davis algorithm and provides a hybrid quantum-classical solver. Second, we present the Janus-CT toolkit, which is based on Davis algorithm and provides a unified compilation framework for solving Boolean satisfiability (SAT) problems (HPCA 2023). We will provide code and demo to demonstrate the speedup of Janus-SAT. Second, we introduce Janus-CT, a unified compilation framework that can analyze tasks into an upstream vectorization model and downstream models (MICRO 2023). In this tutorial, we provide the code and

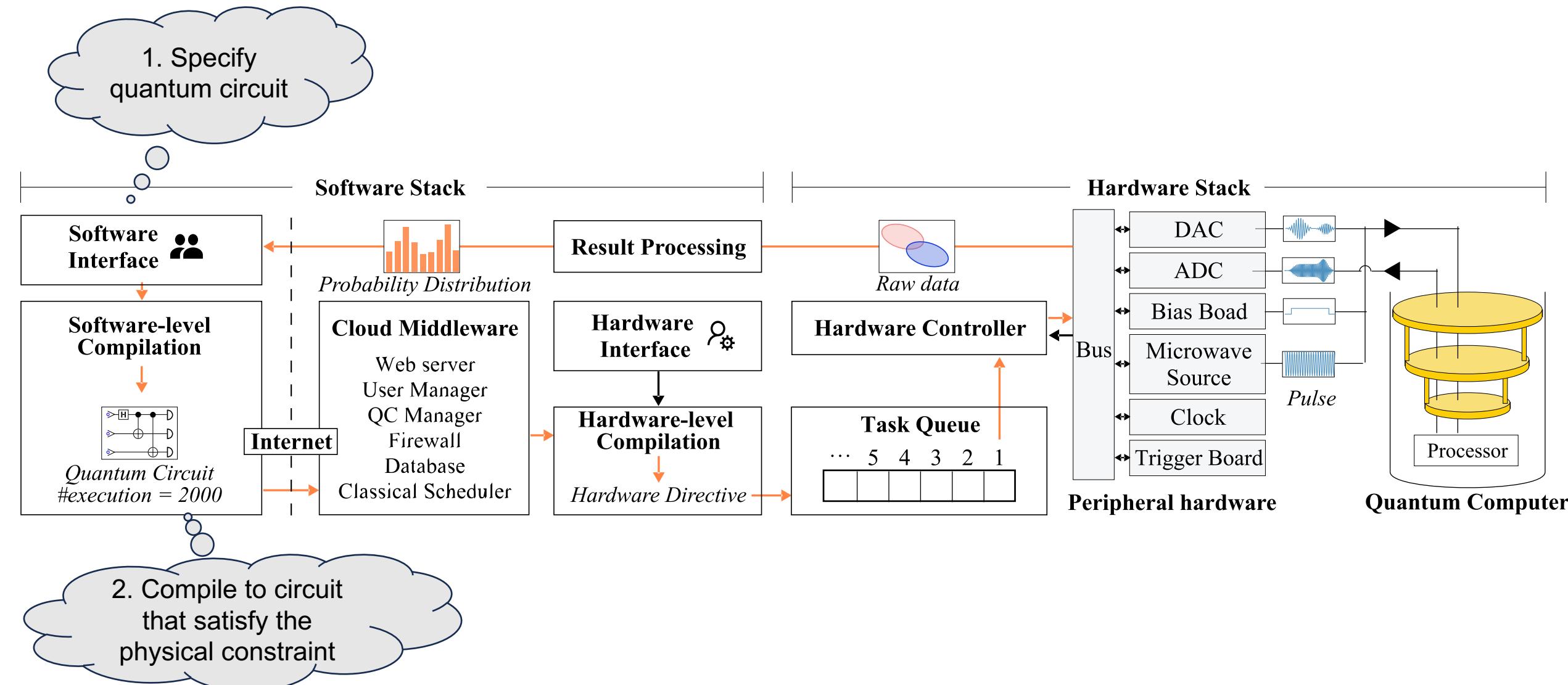
October 2023

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

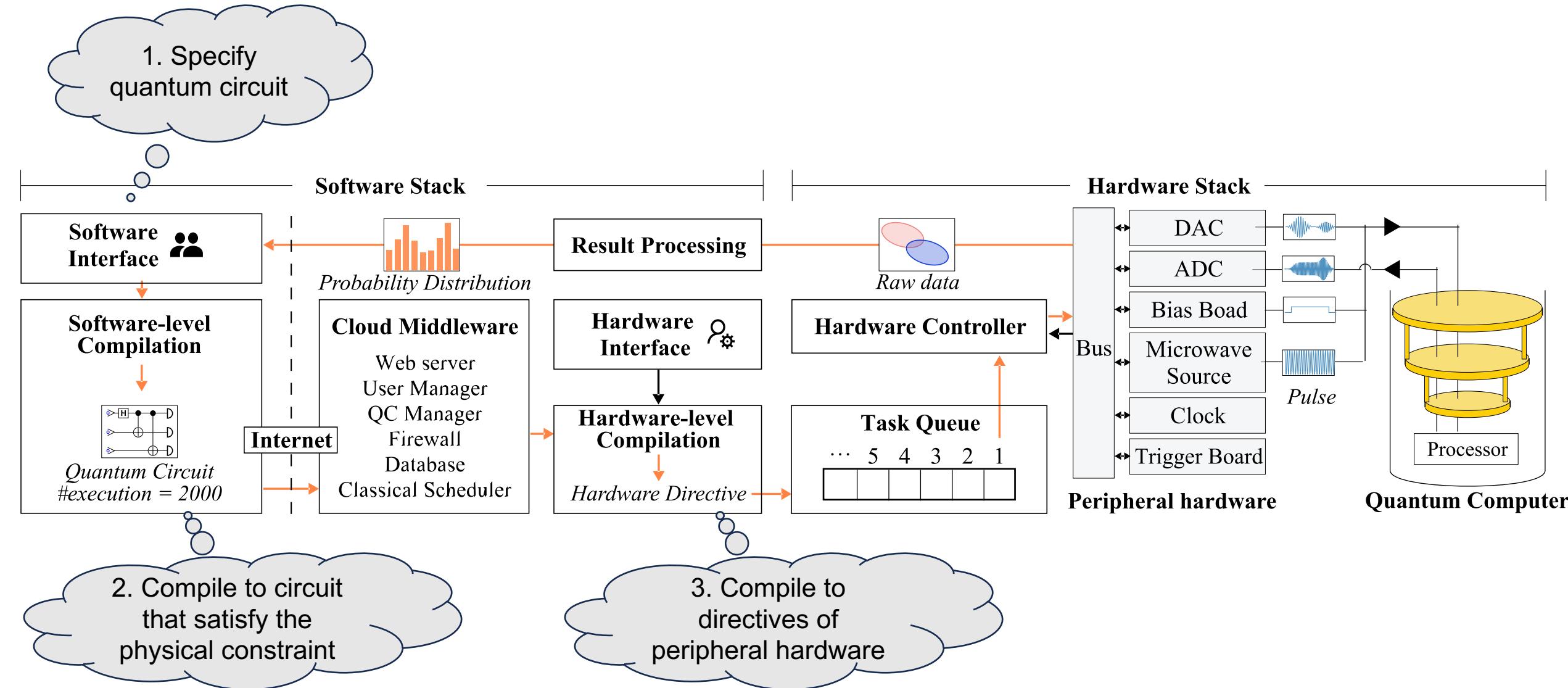
What we can do on Janus Platform



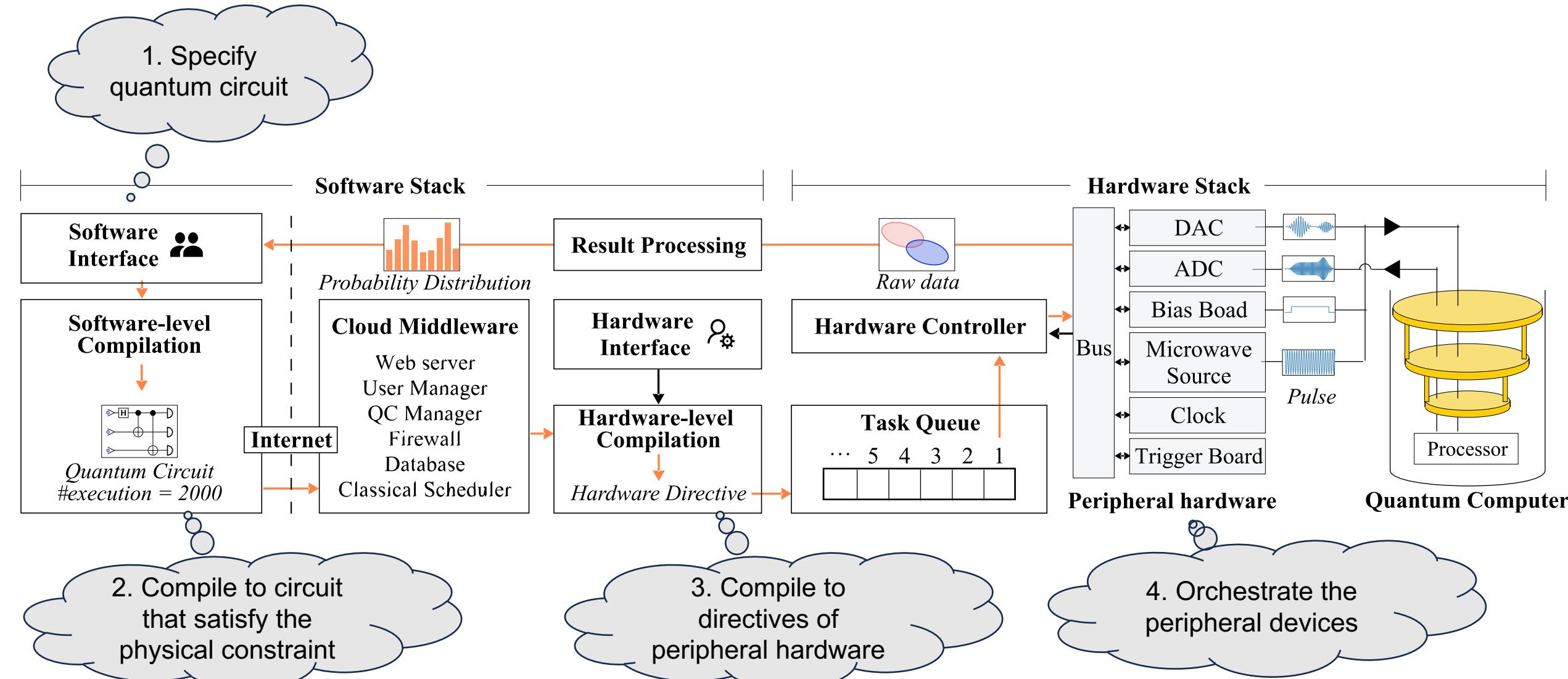
What we can do on Janus Platform



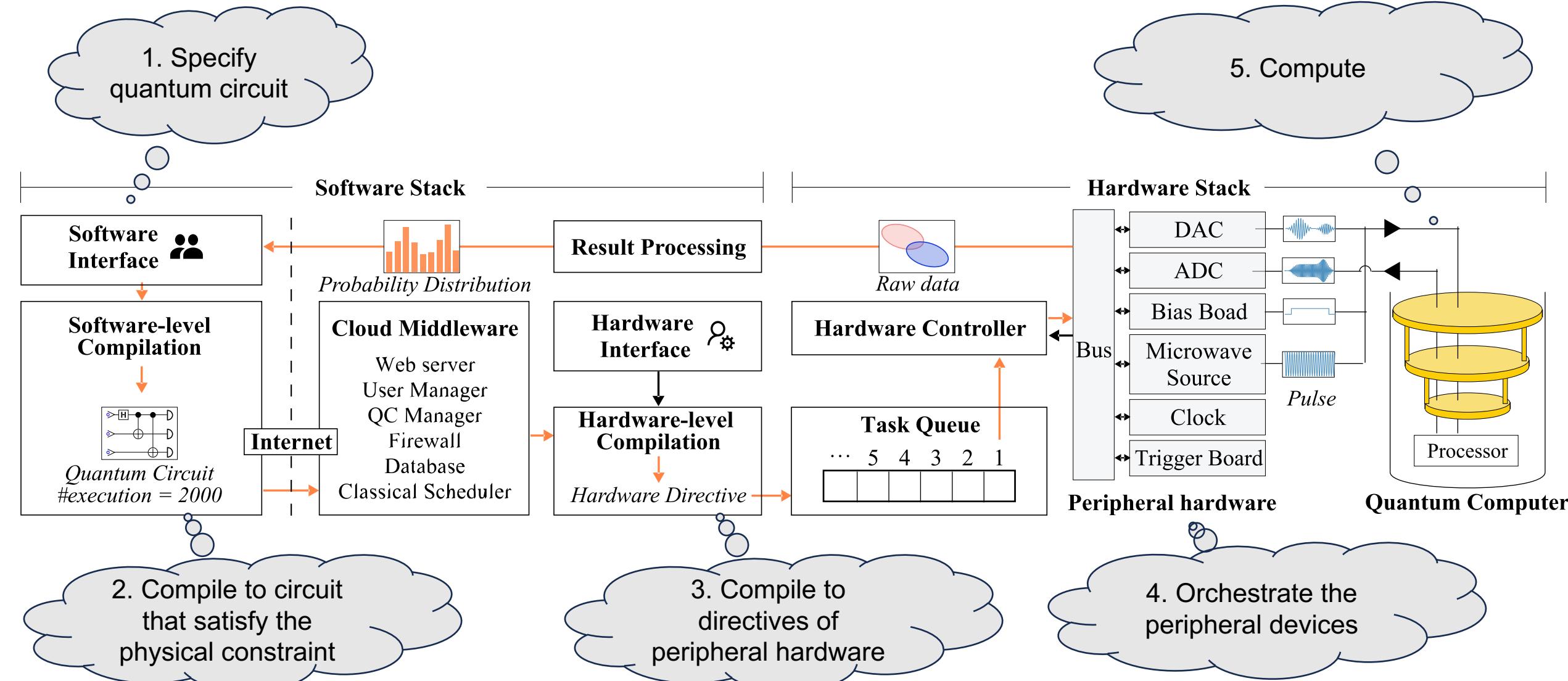
What we can do on Janus Platform



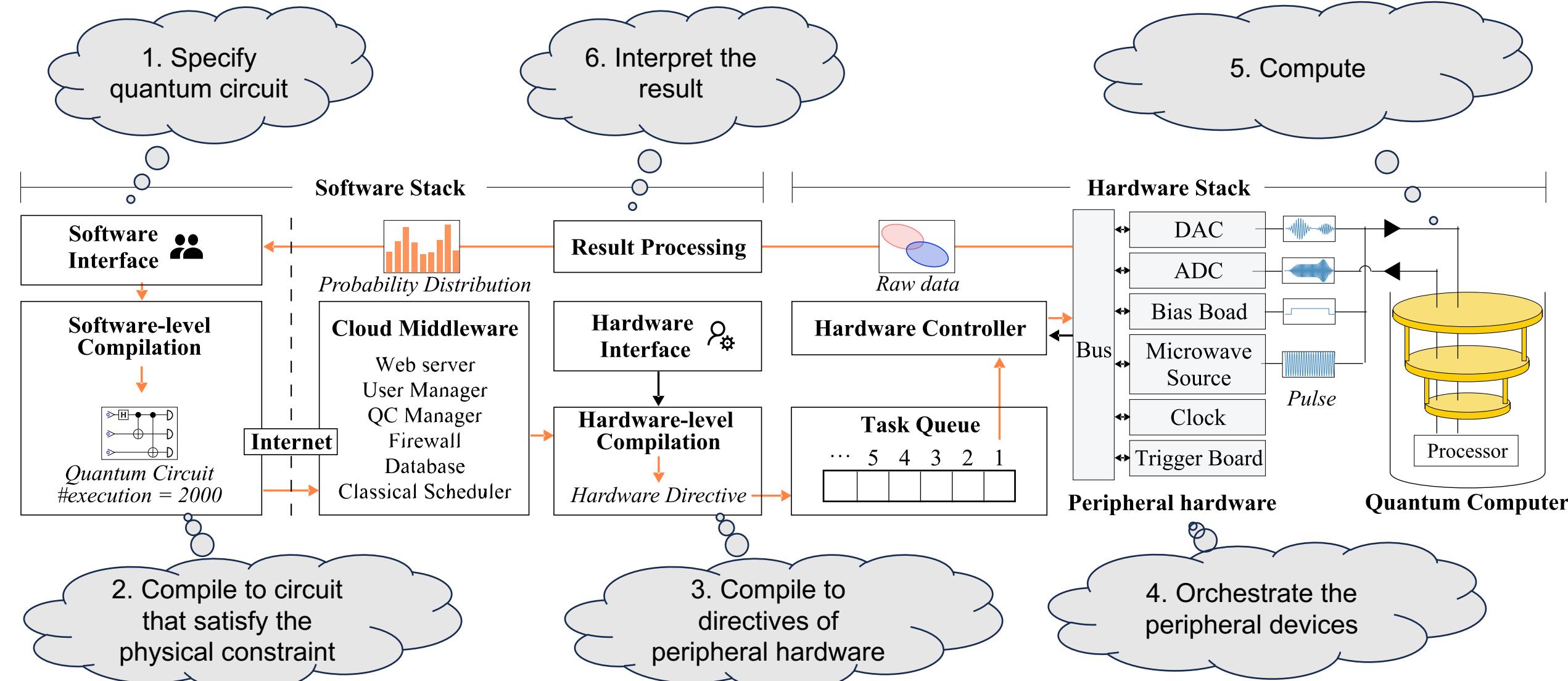
What we can do on Janus Platform



What we can do on Janus Platform

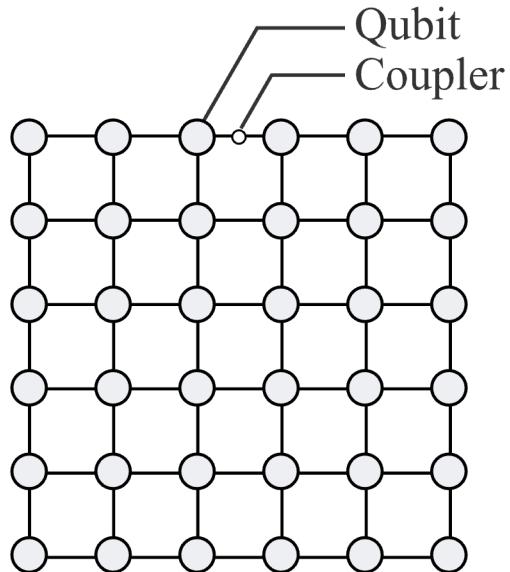


What we can do on Janus Platform

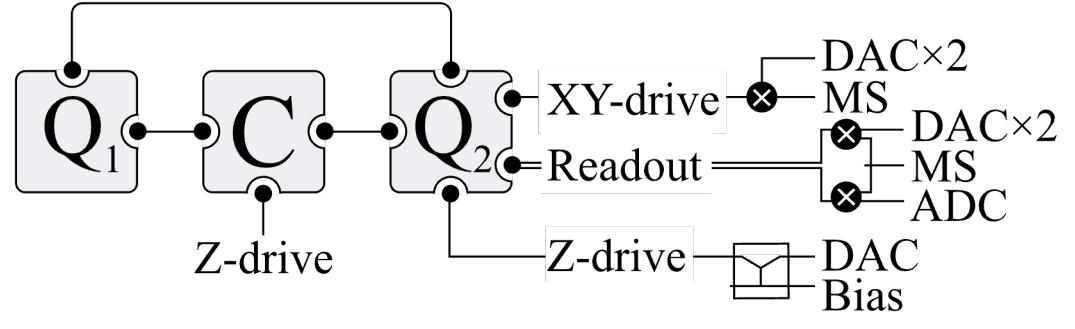


Different compared to other quantum cloud platforms

Custom hardware & more flexibility to improve the fidelity



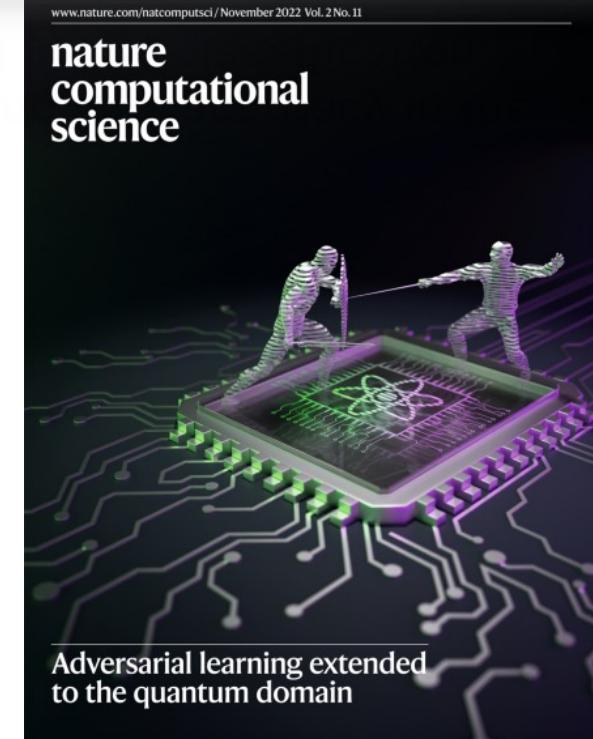
(a) Processor topology.



(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



浙江大學
ZHEJIANG UNIVERSITY

Go to Janus Cloud

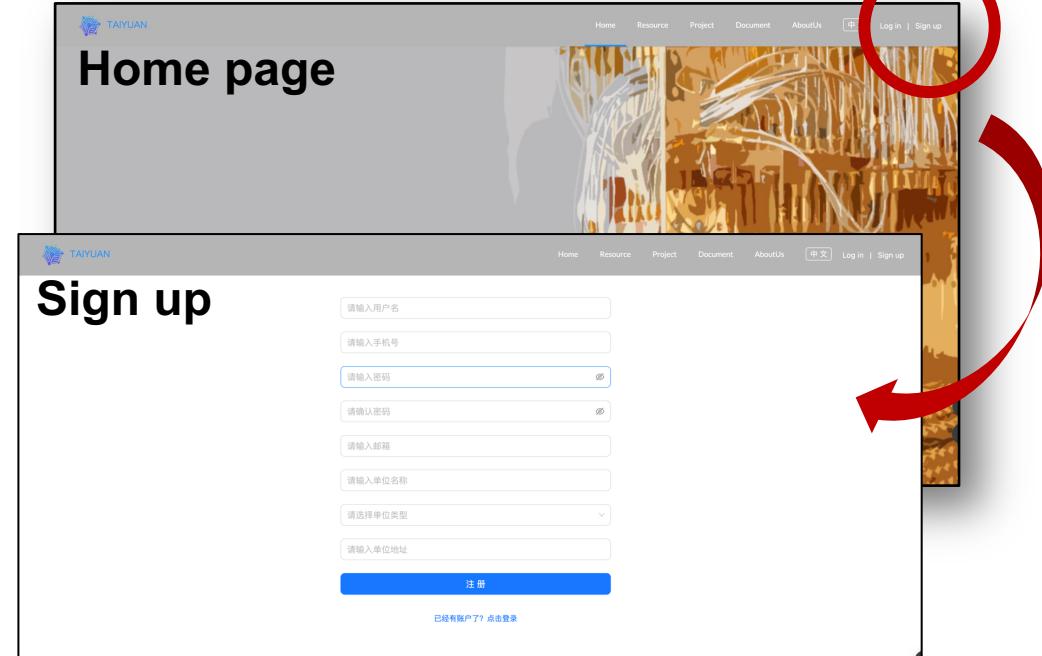
Open <http://janusq.zju.edu.cn/home>



QR code

Sign up

Click sign up button



The screenshot shows two overlapping web pages. The top page is the 'Home page' with a background image of a person working on a computer. The bottom page is the 'Sign up' form. The 'Sign up' button at the bottom of the form is highlighted with a red circle. A red arrow points from the text 'Click sign up button' to this highlighted area.

TAYUAN

Home page

TAYUAN

Sign up

请输入用户名

请输入手机号

请输入密码

请输入确认密码

请输入邮箱

请输入单位名称

请选择单位类型

请输入单位地址

注册

已经有账户了？点击登录

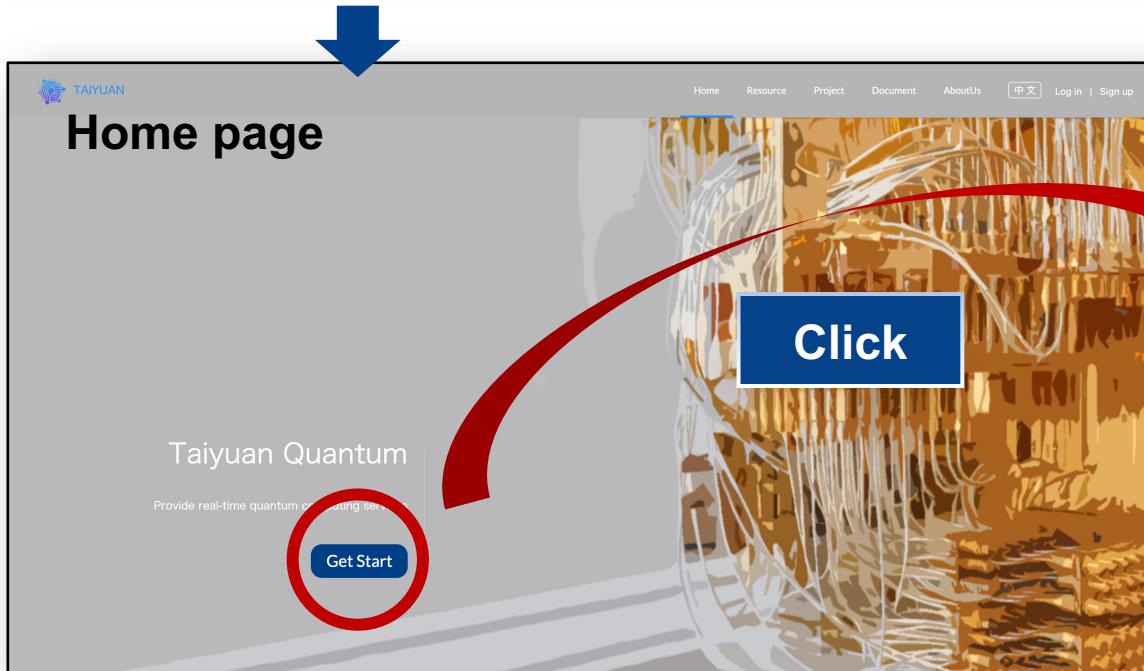
- After signing up, you will get an API key for task submission.
- **The quantum computer can be accessed on the website during the tutorial !!**

Running Program On the Website

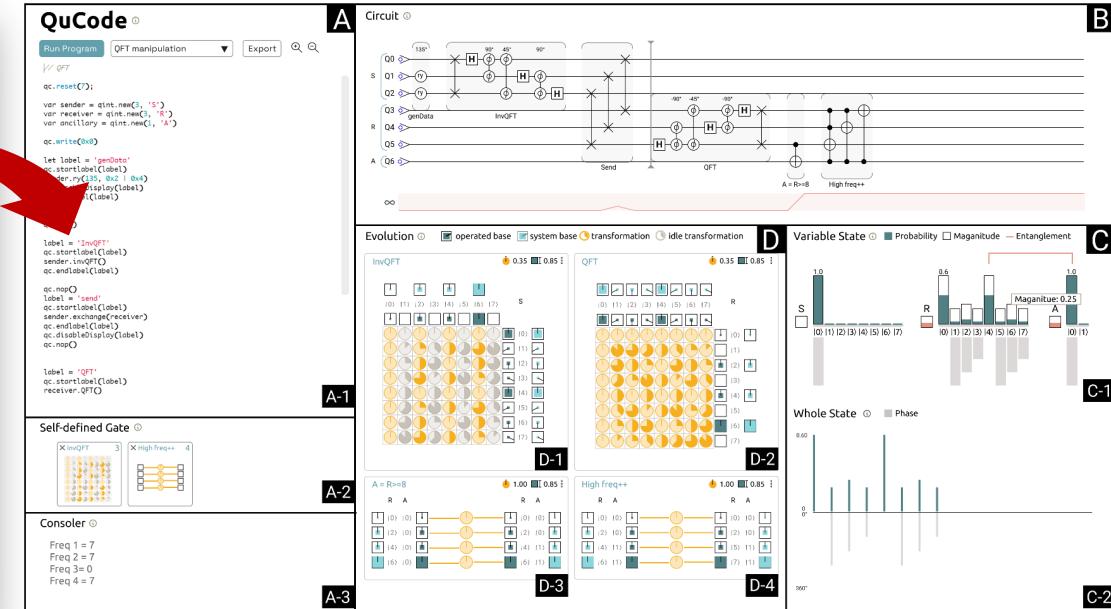


Submitting on website

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



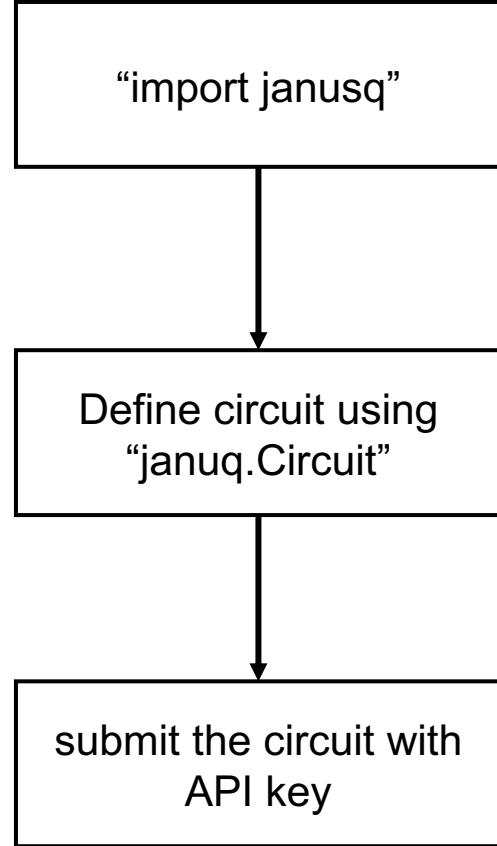
Programming on the Code editor



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

Running Program By Python API

Submitting by Python



```
import 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)
```

The code snippet demonstrates the submission of a quantum circuit using the Python API. It imports the required packages, defines a circuit with four qubits, and performs three CNOT gates between qubits 0 and 1, 1 and 2, and 2 and 3. The circuit is then submitted to a simulator with 3000 shots and a specific label. Finally, the result is retrieved and printed.

QuCode: Online Quantum Program Editor



浙江大学
ZHEJIANG UNIVERSITY

The figure displays the QuCode software interface, which includes a Quantum circuit view, a Program editor, an Evolution view, and a State view.

A. Quantum circuit view: Shows a quantum circuit diagram with multiple qubits (Q0 to Q6) and various gates like H, RY, and CNOT. The circuit includes sections labeled "InvQFT", "Send", and "QFT". A red box highlights the "QFT" section.

B. Program editor: Displays the quantum circuit code in JavaScript-like syntax. It includes sections for generating data ("genData"), performing an inverse Quantum Fourier Transform ("InvQFT"), sending data ("Send"), and applying a Quantum Fourier Transform ("QFT").

C. Evolution view: Shows the evolution of the system over time. It includes four panels (D-1 to D-4) showing the state of qubits S, R, A, and the whole system. Each panel contains a grid of plots representing the probability distribution of different states (e.g., |00>, |11>).

D. State view: Shows the state of the system. It includes three bar charts (D-3, D-4, D-5) showing the probability and magnitude of different states for qubits A, R, and S. A red box highlights the "Maganitude" chart for qubit A.

E. Console: Displays the console output with frequency measurements: Freq 1 = 7, Freq 2 = 7, Freq 3 = 0, Freq 4 = 7.

QuCode: Online Quantum Program Editor



浙江大学
ZHEJIANG UNIVERSITY

The figure shows the Qiskit Terra program editor interface. The top navigation bar includes 'Run Program', 'QFT manipulation' dropdown, 'Export', and search icons.

A: Circuit diagram for a quantum circuit involving multiple qubits (Q0 to Q6) and classical bits (R, A). The circuit includes an 'InvQFT' block, a 'Send' block, and a 'QFT' block. Annotations like 'A = R>=8' and 'High freq++' are present. The circuit starts with a `qc.reset(7);` command.

A-1: Self-defined Gate section showing two gates: `X invQFT` (3 controls) and `X High freq++` (4 controls).

A-2: Consoler output showing Freq values: Freq 1 = 7, Freq 2 = 7, Freq 3 = 0, Freq 4 = 7.

B: Evolution and Variable State analysis. Evolution plots show the state of qubits S, R, and A over time. Variable State plots show Probability, Magnitude, and Entanglement for S, R, and A.

C: Whole State plot showing phase and magnitude for the entire system.

D: Detailed evolution plots for InvQFT, QFT, and two user-defined gates: `A = R>=8` and `High freq++`.

QuCode: Online Quantum Program Editor



浙江大学
ZHEJIANG UNIVERSITY

QuCode: Online Quantum Program Editor



浙江大學
ZHEJIANG UNIVERSITY

QuCode ⓘ

Run Program | QFT manipulation | Export | 🔍 | 🔍

```
// QFT
qc.reset(7);

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

qc.write(0x0)

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

qc.nop()

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

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

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

Circuit View

Grouping

Purity

A-1

D-1

C-1

A-2

D-2

C-2

A-3

D-3

C-3

B

C

D

A-4

D-4

C-4

A-5

D-5

C-5

A-6

D-6

C-6

A-7

D-7

C-7

A-8

D-8

C-8

A-9

D-9

C-9

A-10

D-10

C-10

A-11

D-11

C-11

A-12

D-12

C-12

A-13

D-13

C-13

A-14

D-14

C-14

A-15

D-15

C-15

A-16

D-16

C-16

A-17

D-17

C-17

A-18

D-18

C-18

A-19

D-19

C-19

A-20

D-20

C-20

A-21

D-21

C-21

A-22

D-22

C-22

A-23

D-23

C-23

A-24

D-24

C-24

A-25

D-25

C-25

A-26

D-26

C-26

A-27

D-27

C-27

A-28

D-28

C-28

A-29

D-29

C-29

A-30

D-30

C-30

A-31

D-31

C-31

A-32

D-32

C-32

A-33

D-33

C-33

A-34

D-34

C-34

A-35

D-35

C-35

A-36

D-36

C-36

A-37

D-37

C-37

A-38

D-38

C-38

A-39

D-39

C-39

A-40

D-40

C-40

A-41

D-41

C-41

A-42

D-42

C-42

A-43

D-43

C-43

A-44

D-44

C-44

A-45

D-45

C-45

A-46

D-46

C-46

A-47

D-47

C-47

A-48

D-48

C-48

A-49

D-49

C-49

A-50

D-50

C-50

A-51

D-51

C-51

A-52

D-52

C-52

A-53

D-53

C-53

A-54

D-54

C-54

A-55

D-55

C-55

A-56

D-56

C-56

A-57

D-57

C-57

A-58

D-58

C-58

A-59

D-59

C-59

A-60

D-60

C-60

A-61

D-61

C-61

A-62

D-62

C-62

A-63

D-63

C-63

A-64

D-64

C-64

A-65

D-65

C-65

A-66

D-66

C-66

A-67

D-67

C-67

A-68

D-68

C-68

A-69

D-69

C-69

A-70

D-70

C-70

A-71

D-71

C-71

A-72

D-72

C-72

A-73

D-73

C-73

A-74

D-74

C-74

A-75

D-75

C-75

A-76

D-76

C-76

A-77

D-77

C-77

A-78

D-78

C-78

A-79

D-79

C-79

A-80

D-80

C-80

A-81

D-81

C-81

A-82

D-82

C-82

A-83

D-83

C-83

A-84

D-84

C-84

A-85

D-85

C-85

A-86

D-86

C-86

A-87

D-87

C-87

A-88

D-88

C-88

A-89

D-89

C-89

A-90

D-90

C-90

A-91

D-91

C-91

A-92

D-92

C-92

A-93

D-93

C-93

A-94

D-94

C-94

A-95

D-95

C-95

A-96

D-96

C-96

A-97

D-97

C-97

A-98

D-98

C-98

A-99

D-99

C-99

A-100

D-100

C-100

QuCode: Online Quantum Program Editor



浙江大學
ZHEJIANG UNIVERSITY

QuCode

Circuit

Evolution

State View

A
B

Self-defined Gate

Consoler

C

D

A-1
B
C

A-2
C-1
C-2

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

State View

Legend: Maganitude (blue), Entanglement (red)

partial trace

The whole state

QuCode: Online Quantum Program Editor



浙江大學
ZHEJIANG UNIVERSITY

QuCode

Circuit

Evolution View

Self-defined Gate

Consoler

A
Circuit
B

D
C

D-1

D-2

D-3

D-4

1. Visualize the unitary

2. Visualize the change of qubit states

Simulation of Time crystal



Mathematical formulation

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

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

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

A layer of rotation gates along the x axis

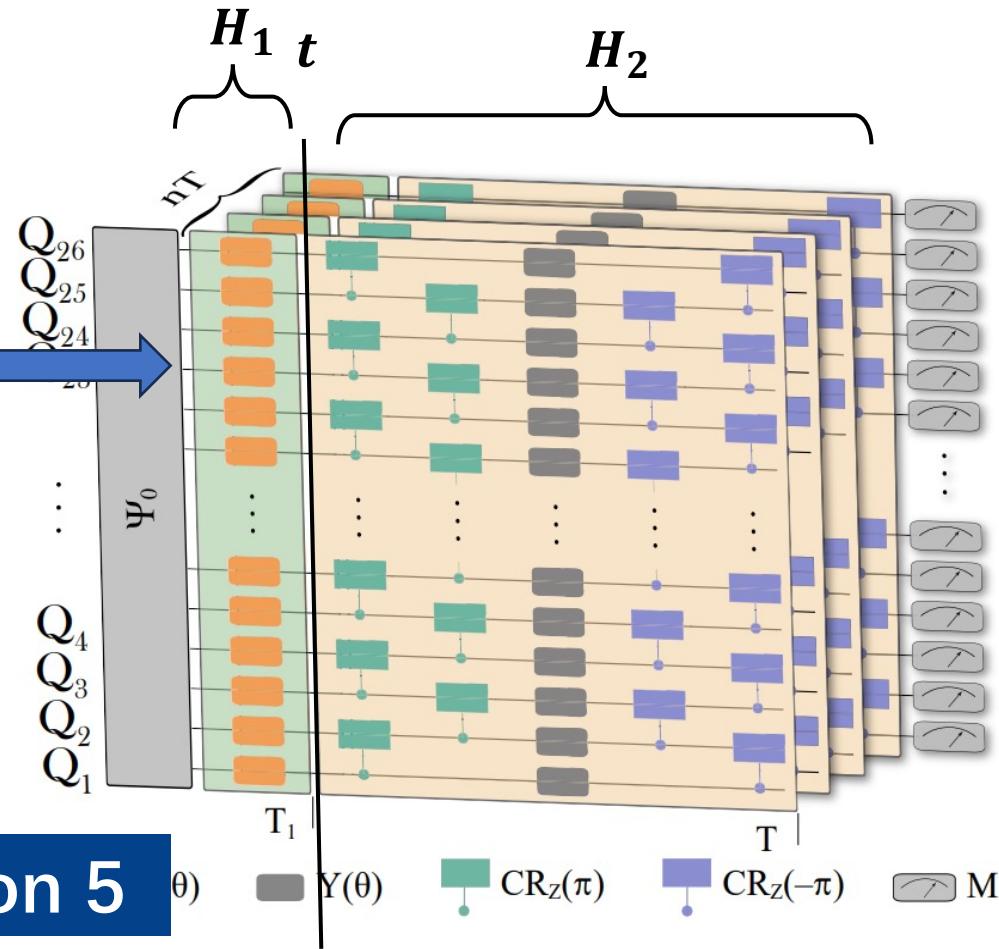
$$H_2 \equiv - \sum_k J_k \sigma_{k-1}^z \sigma_k^x \sigma_{k+1}^z$$

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

20 random disorder instances (L)

Introduced In Session 5

Circuit Implementation





浙江大學
ZHEJIANG UNIVERSITY

Thanks for listening!