TOWARDS AN OPEN SOURCE QUANTUM OPERATING SYSTEM

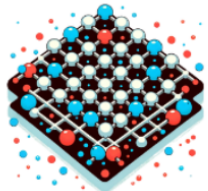
Edoardo Pedicillo on behalf of the Qiboteam

ACAT, 11th March 2024

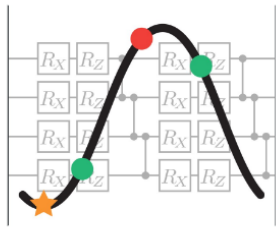
Challenging quantum computing applications

Quantum chemistry

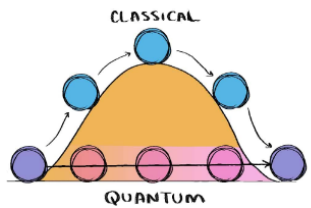
Quantum simulation



Quantum machine learning



Quantum Annealing



Post-quantum Cryptography

Introduction to Quantum Computers

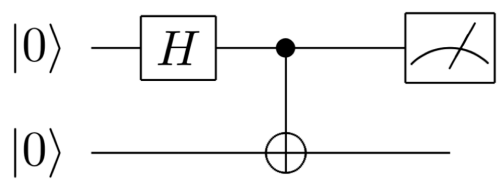
- In a quantum computer bits are replaced by **qubits**
- The state of the qubit is a superposition of two quantum states

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$$

- We can act on qubits with unitary operators represented as **gates** (i.e., Hadamard gate)

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \qquad CX = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

- When we apply a series of either unitary or measurement gates to a styem of qubits, initialized to a known state we obtain a quantum **circuit** ,i.e., Grover, variational quantum circuits.

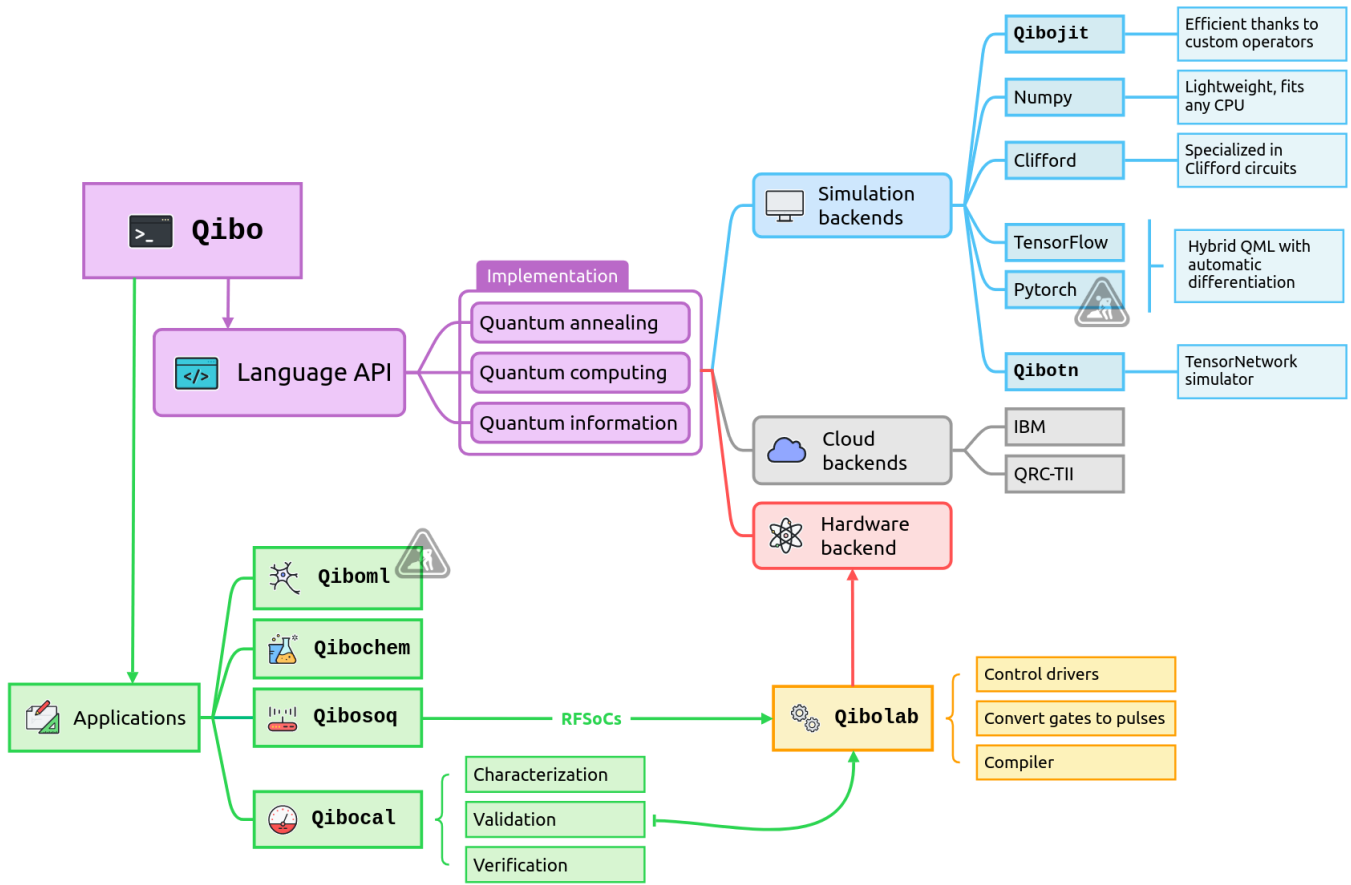


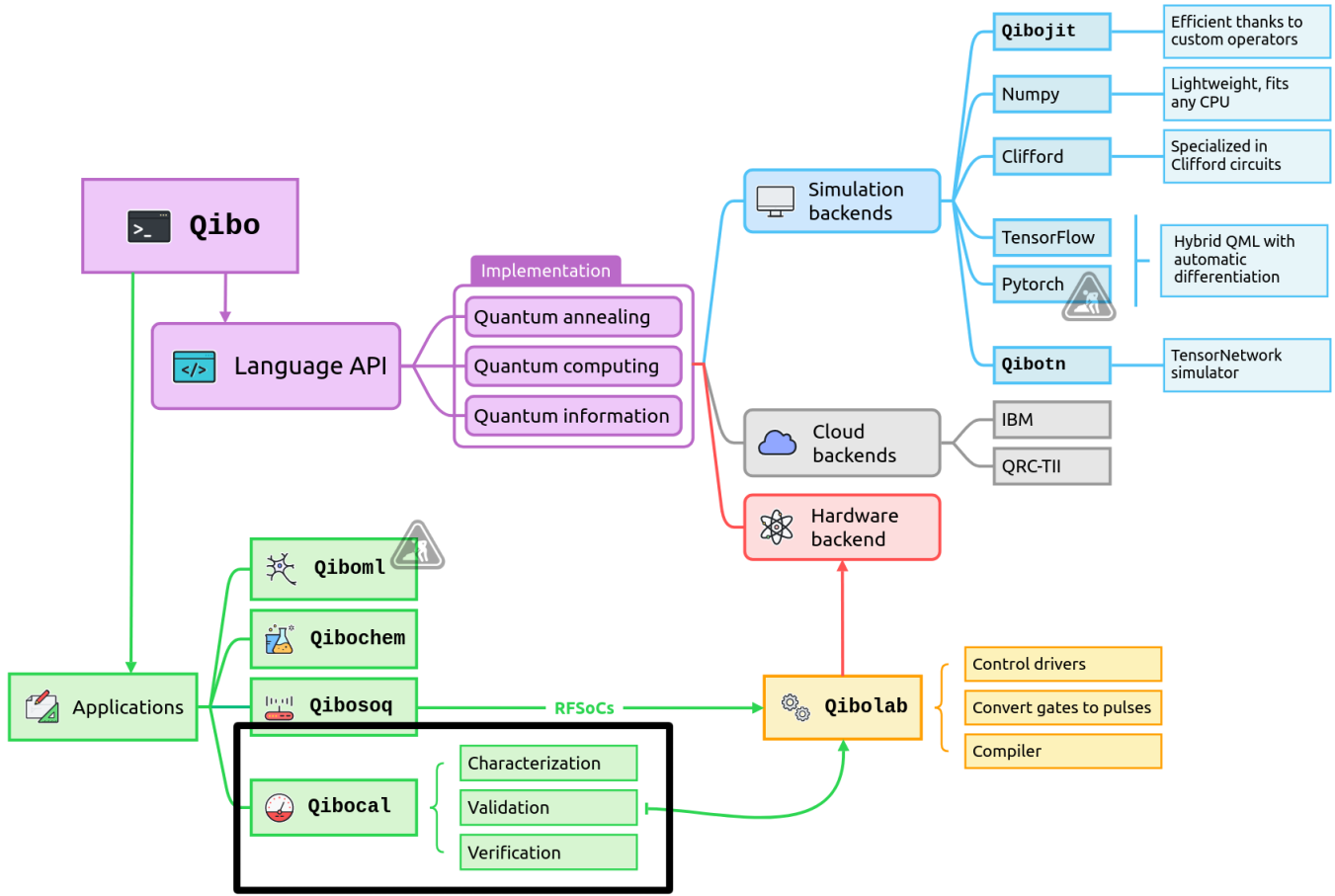
Qibo ecosystem

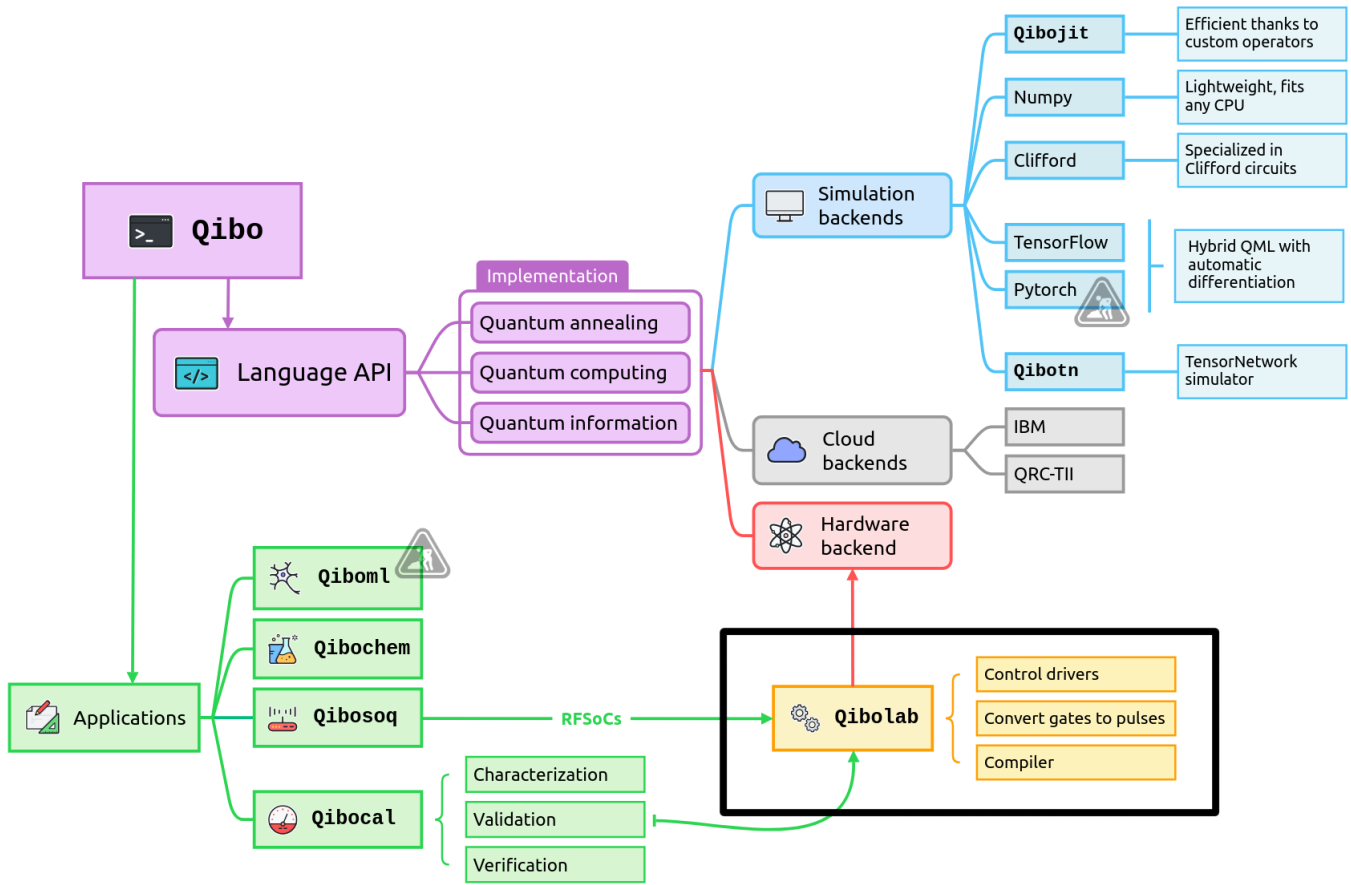
Open-source full stack API for quantum simulation, hardware control and calibration

Quantum Journal

qiboteam/qibo


Qibo
Language API
Implementation
Quantum annealing
Quantum computing
Quantum information
Simulation backends
Qibojit
Efficient thanks to custom operators
Numpy
Lightweight, fits any CPU
Clifford
Specialized in Clifford circuits
TensorFlow
Pytorch
Hybrid QML with automatic differentiation
Qibotn
TensorNetwork simulator
Cloud backends
IBM
QRC-TII
Hardware backend
Applications
Qiboml
Qibochem
Qibosoq
RFSoCs
Qibocal
Characterization
Validation
Verification
Qibolab
Control drivers
Convert gates to pulses
Compiler


Qibo
Language API
Implementation
Quantum annealing
Quantum computing
Quantum information
Simulation backends
Qibojit
Efficient thanks to custom operators
Numpy
Lightweight, fits any CPU
Clifford
Specialized in Clifford circuits
TensorFlow
Pytorch
Hybrid QML with automatic differentiation
Qibotn
TensorNetwork simulator
Cloud backends
IBM
QRC-TII
Hardware backend
Applications
Qiboml
Qibochem
Qibosoq
RFSoCs
Qibocal
Characterization
Validation
Verification
Qibolab
Control drivers
Convert gates to pulses
Compiler


Qibo
Language API
Implementation
Quantum annealing
Quantum computing
Quantum information
Simulation backends
Qibojit
Efficient thanks to custom operators
Numpy
Lightweight, fits any CPU
Clifford
Specialized in Clifford circuits
TensorFlow
Pytorch
Hybrid QML with automatic differentiation
Qibotn
TensorNetwork simulator
Cloud backends
IBM
QRC-TII
Hardware backend
Applications
Qiboml
Qibochem
Qibosoq
RFSoCs
Qibocal
Characterization
Validation
Verification
Qibolab
Control drivers
Convert gates to pulses
Compiler

QIBOLAB

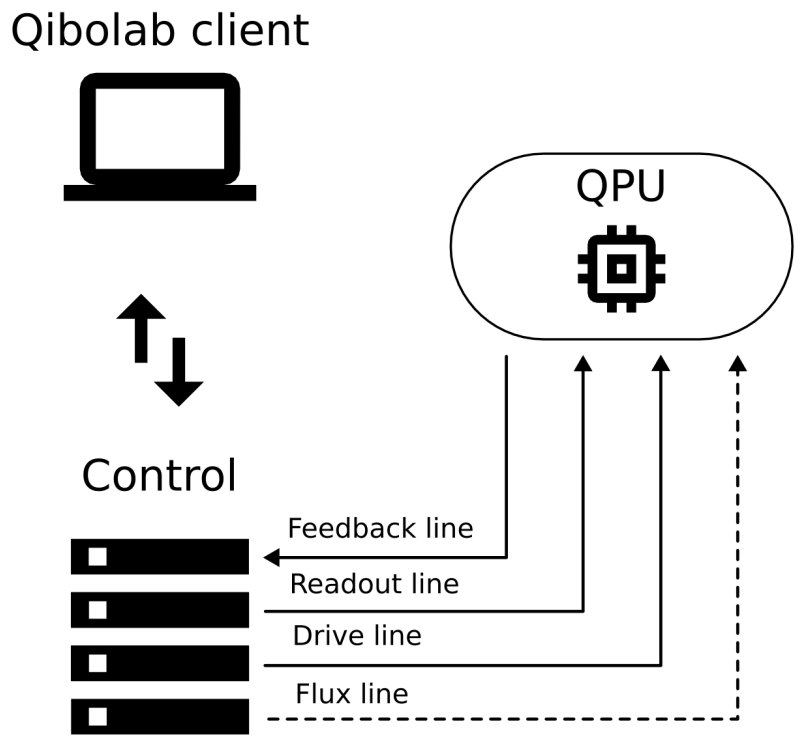
open-source software library for quantum hardware control

Quantum lab


Input-Output lines
Cryostat
Cryostat
Control
Electronics
Oscilloscope

How does a quantum lab work?

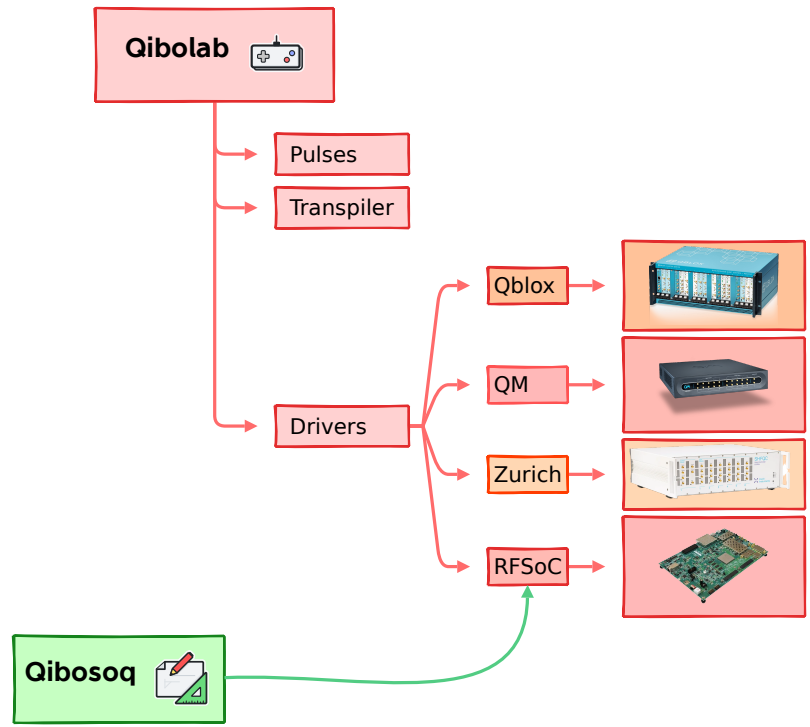
- The host computer running Qibolab communicates with the different electronics used to control a QPU.
- The readout and feedback channels measure the qubits,
- the drive channel applies gates,
- the flux channels for tuning their frequency.



Software abstraction

Qibolab provides two main interface objects:

- the `Pulse` object for defining arbitrary pulses to be played on qubits,
- the `Platform` which is used to execute these pulses on a specific QPU and set of instruments.



Qibocal

A reporting tool for calibration using Qibo

arxiv:2303.10397

qiboteam/qibocal

Motivation

Let’s suppose the following:

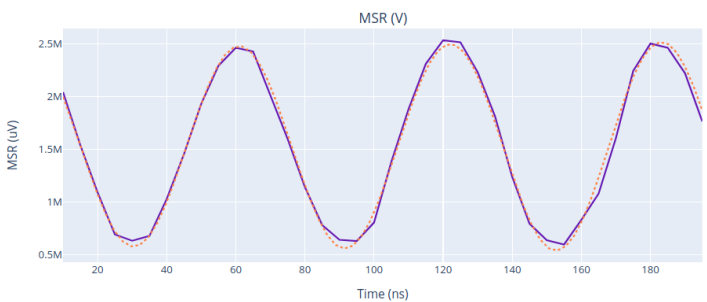
We have a QPU (self-hosted).

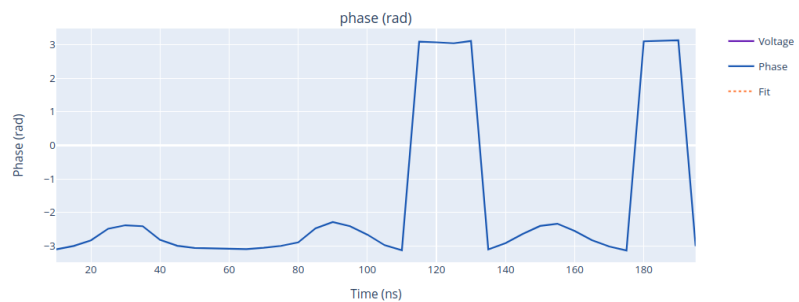
We have control over what we send to the QPU.

We know how to convert quantum circuits to pulses.

Can I **trust** my results? **NO!**

Characterization and **calibration** are an essential step to properly operate emerging quantum devices.





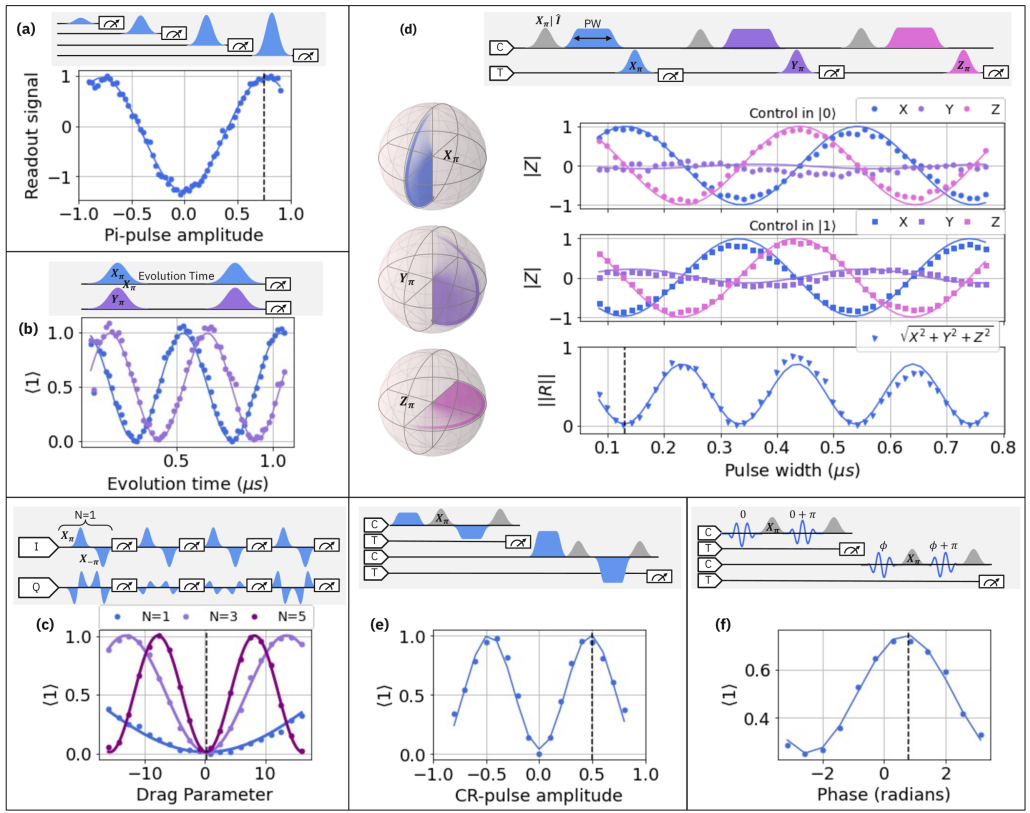
How to calibrate superconducting devices?

In superconducting qubits gates are implemented through microwave pulses.

Several protocols need to be executed to extract specific parameters.

After an initial calibration more advanced experiments can be performed in order to:

- improve readout
- run benchmarking protocols
- reach optimal control



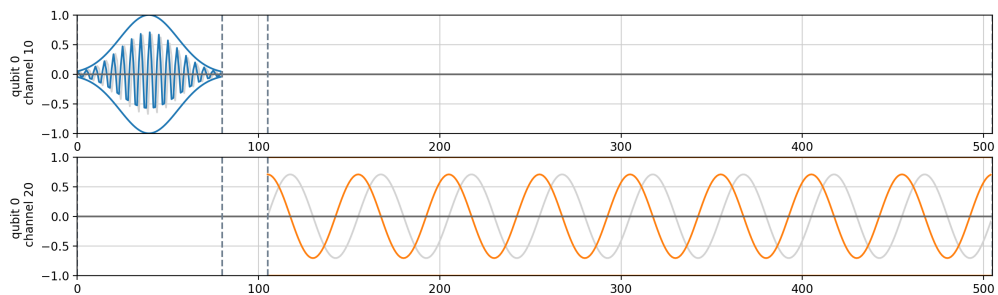
From gates to pulses

Given a general single qubit gate it is possible to decompose it in **RX** and **RZ** gates

$$U3(\theta, \phi, \lambda) = RZ(\phi)RX(-\pi/2)RZ(\theta)RX(\pi/2)RZ(\lambda)$$

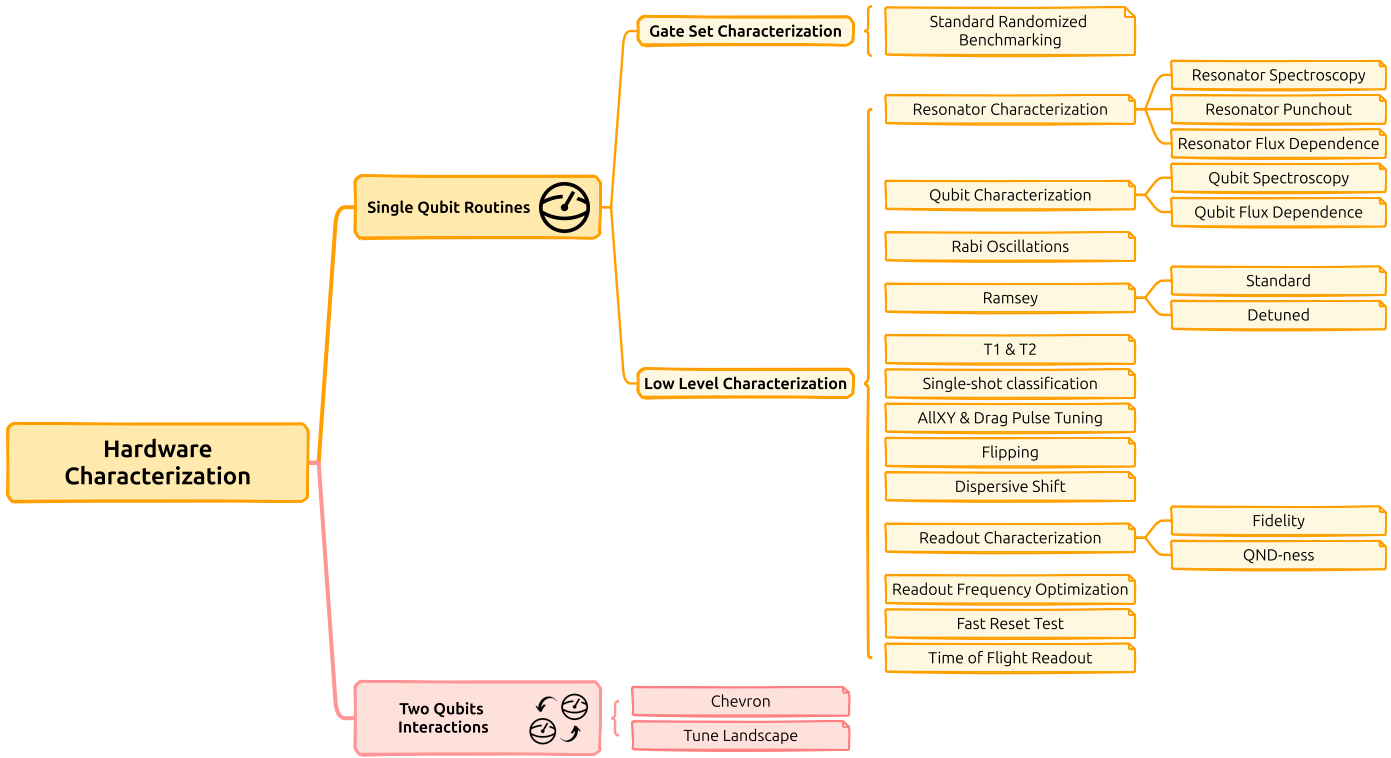
From the level of pulses:

- An **RX** is a Gaussian pulse calibrated by Rabi experiment
- An **RZ** is a change in the virtual phase of the pulses.
- An **MZ** is a rectangular pulse calibrated by readout optimization routines

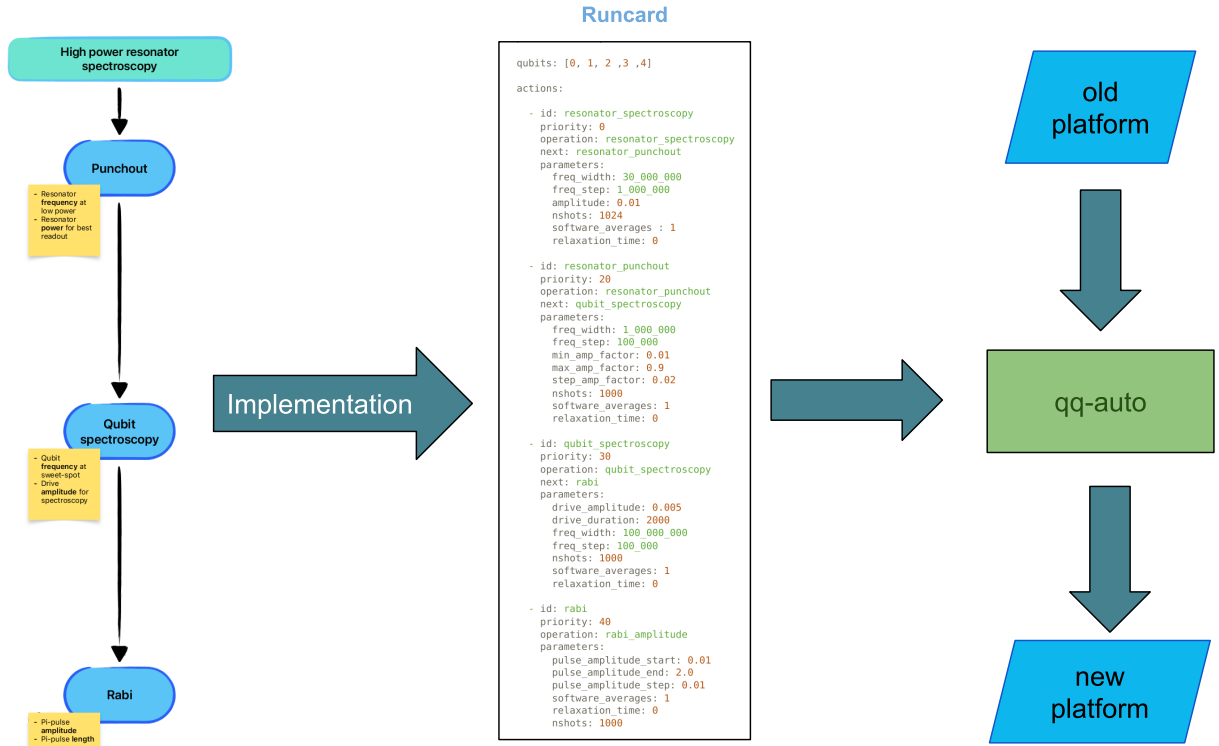


How an RX and measurement gate is performed at the pulses level on a qubit.

Qubits Characterization



How to perform an experiment



REPORT

Qibocal Reports

- Home
  - Timestamp
  - Summary
- Actions
  - Ramsey - 0

Ramsey Experiment

Export to pdf

Platform: tii_rfsoc4x2
Run date: 2023-05-07
Start time (UTC): 05:20:51
End time (UTC): 05:21:19

Summary

In the table below we show the libraries and respective versions used in Ramsey Experiment.

Library	Version
numpy	1.23.5
qibo	0.1.13
qibocal	0.0.2
qibolab	0.0.3

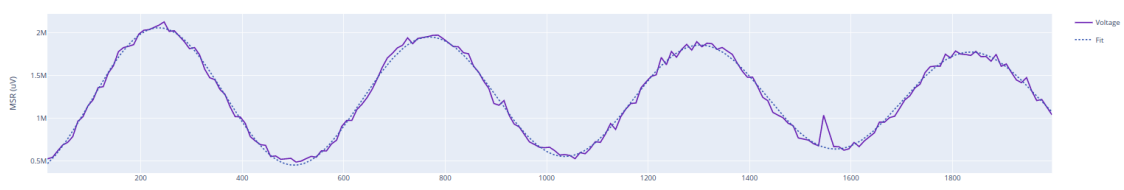
Actions

Please find below data generated by actions:

Ramsey - 0

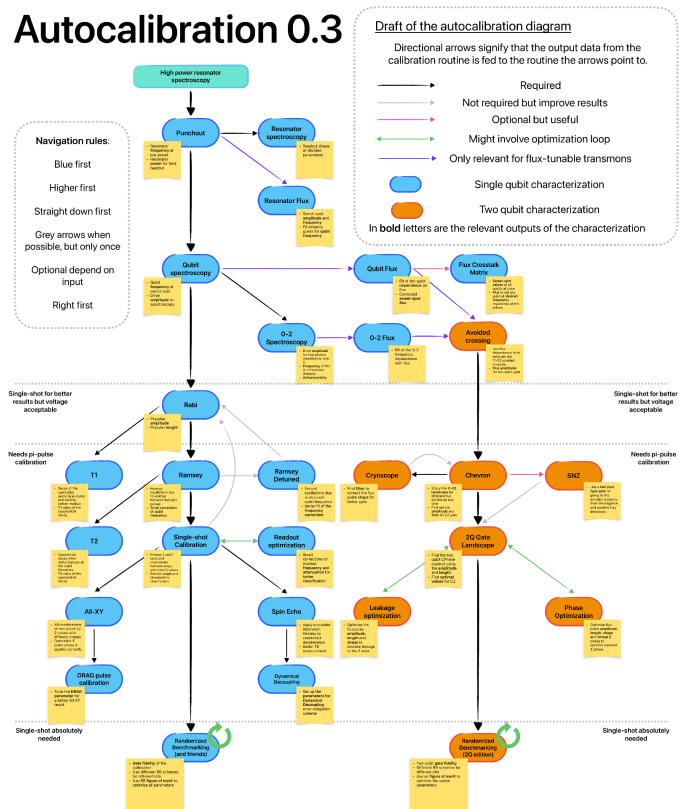
- Qubit 0

qubit	Fitting Parameter	Value
0	delta_frequency	-625,626.0 Hz
0	drive_frequency	5542303347.0 Hz

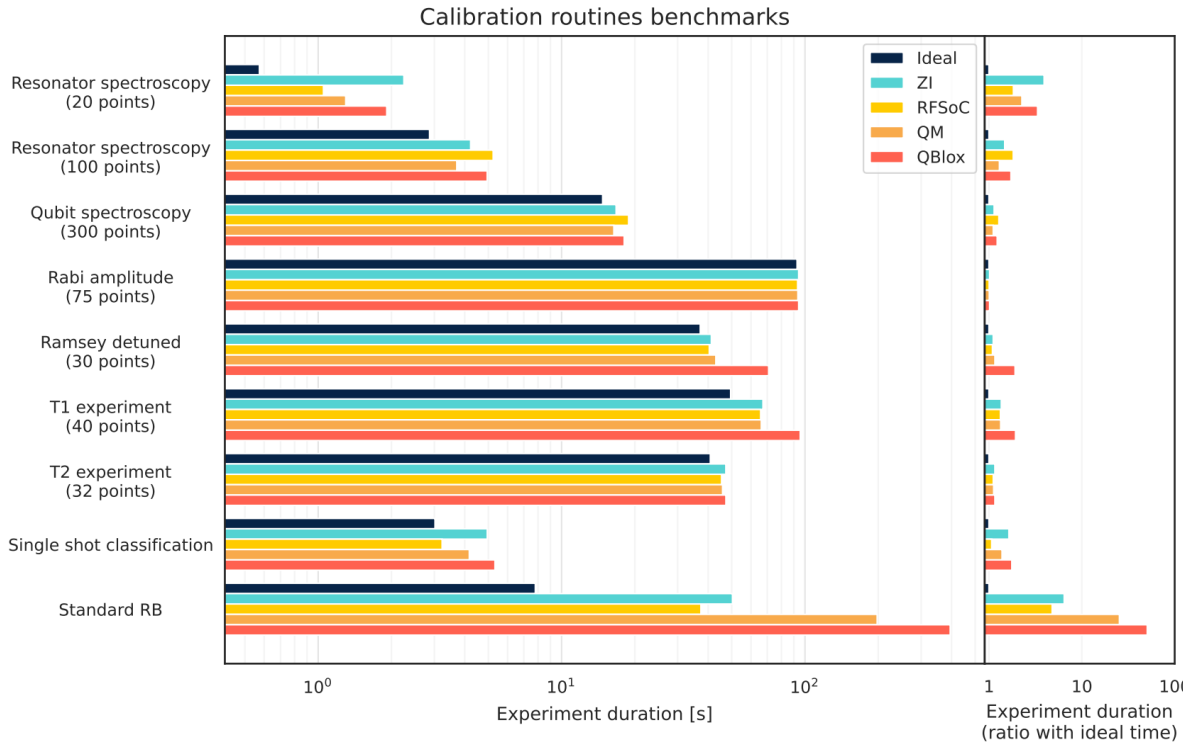


Towards automation

- Specify a direct acyclic graph with various experiments
- Parameters computed are fed from one routine to the next
- Full quantum chip recalibration



QIBOLAB + QIBOCAL



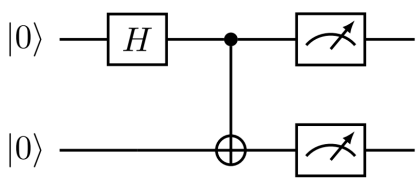
$$T_{real} = T_{qibo} + T_{instrument} + T_{ideal}$$

Qibo+Qibolab+Qibocal

1. Circuit definition

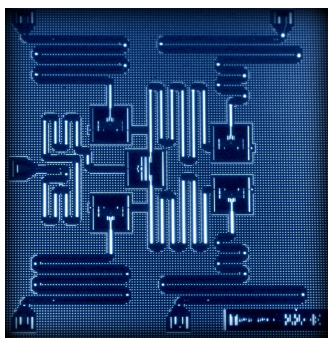
```
from qibo import Circuit, gates

c = Circuit(2)
c.add(gates.H(0))
c.add(gates.CNOT(0,1))
c.add(gates.M(0,1))
shots = c(nshots=1000)
```

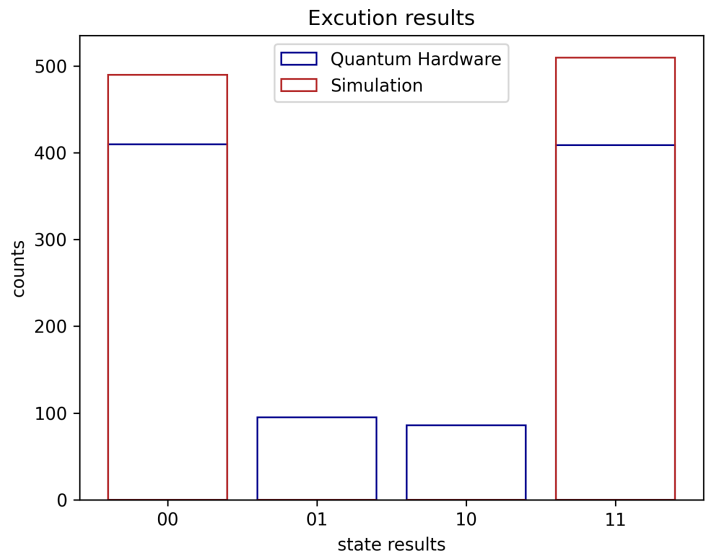


2. Circuit compilation

Translate the circuit into a sequence of pulses considering the chip topology.



3. Hardware execution



Thanks for listening!