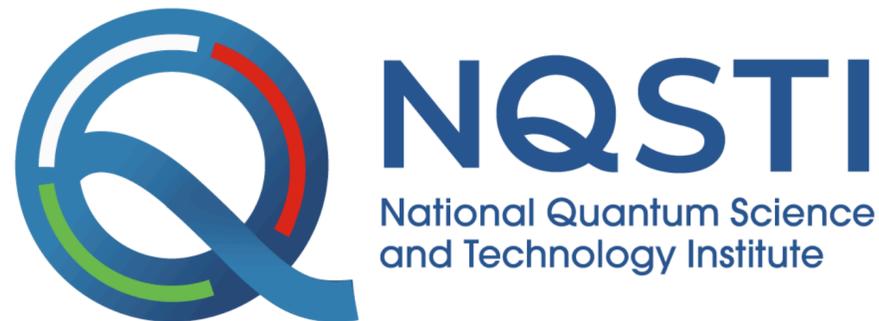


# Introducing Qibo

**An open-source quantum operative system**



**Andrea Pasquale on the behalf of the Qibo collaboration**

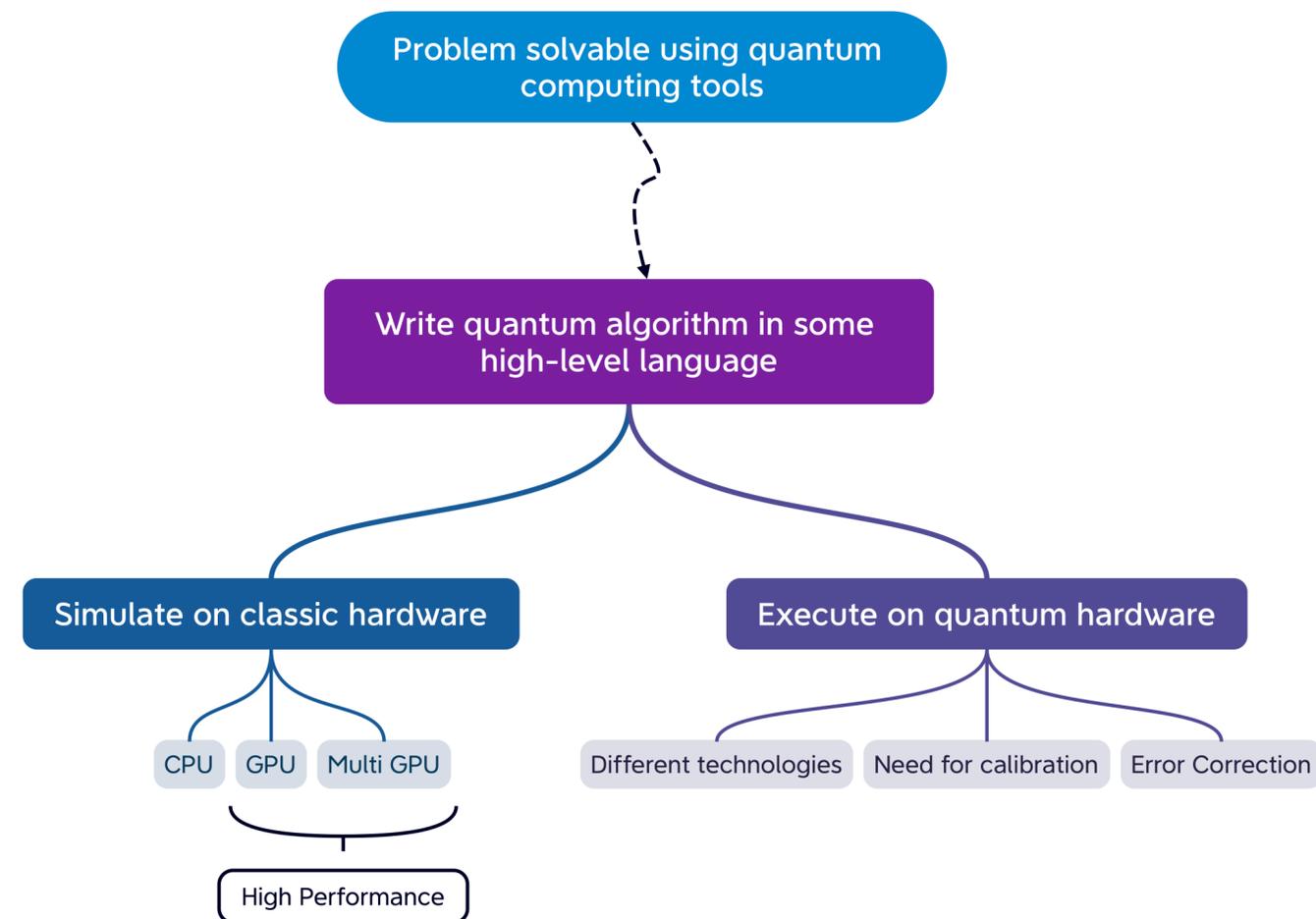
**QUART&T Kick-Off Meeting 21th Feb 2025**

# Outline of the presentation

- Role of middleware in Quantum Computing
- Introducing the **Qibo** framework
- Hardware control: **Qibolab**
- Calibrating superconducting qubits with **Qibocal**

**Middleware**

# What is middleware?

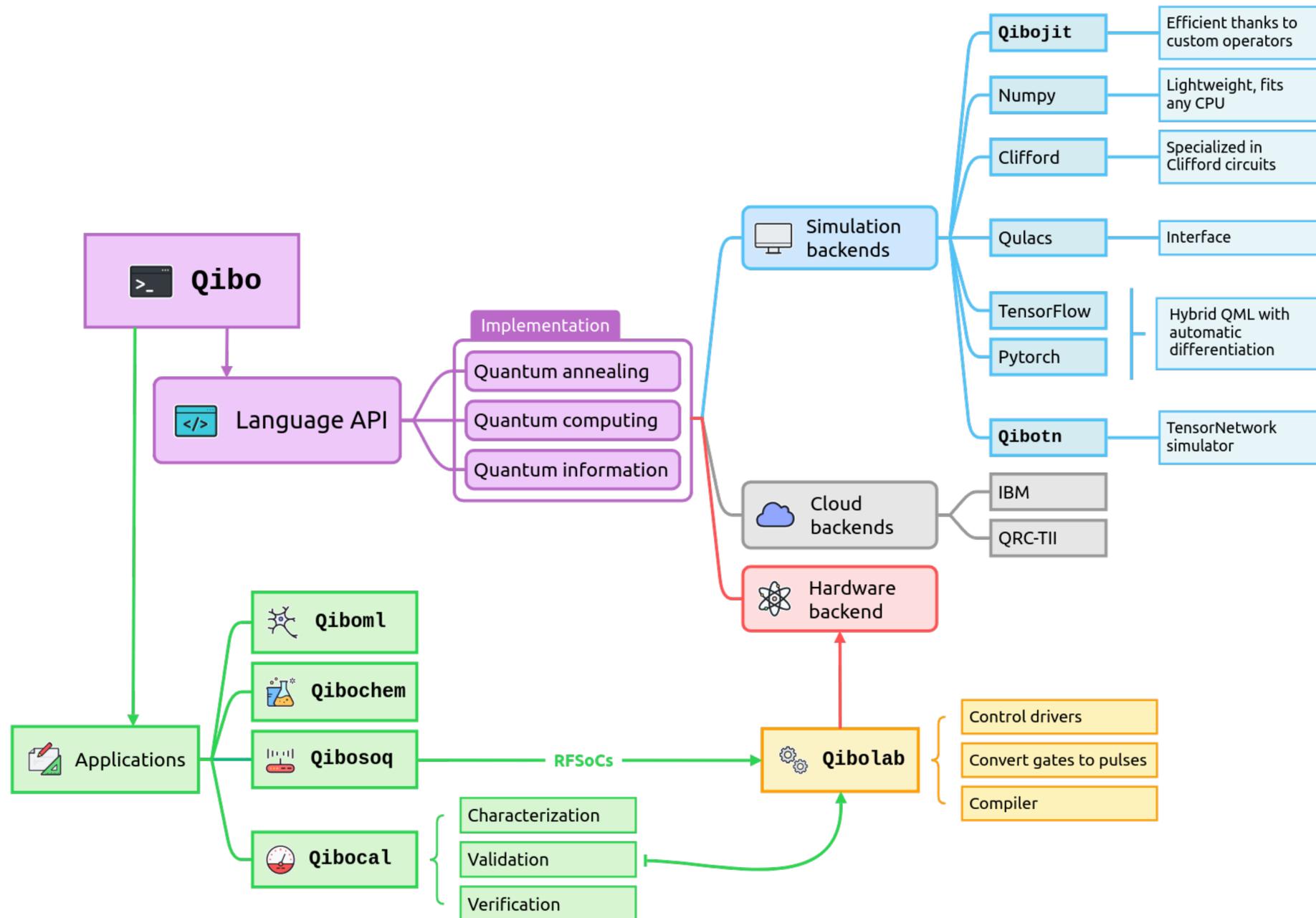


*“Middleware for Quantum is the software that sits between the quantum hardware and the quantum algorithms, providing an abstraction layer that hides the complexities of the underlying heterogeneous hardware and allows developers to focus on the logic of their quantum algorithms within the context of quantum-classical applications.”*

Faro, I., Sitdikov, I., Valiñas, D. G., Fernandez, F. J. M., Codella, C., & Glick, J. (2023). Middleware for quantum: An orchestration of hybrid quantum-classical systems. In *2023 IEEE International Conference on Quantum Software (QSW)* (pp. 1–8). <https://doi.org/10.1109/QSW59989.2023.00011>

# Introducing Qibo

# Introducing Qibo



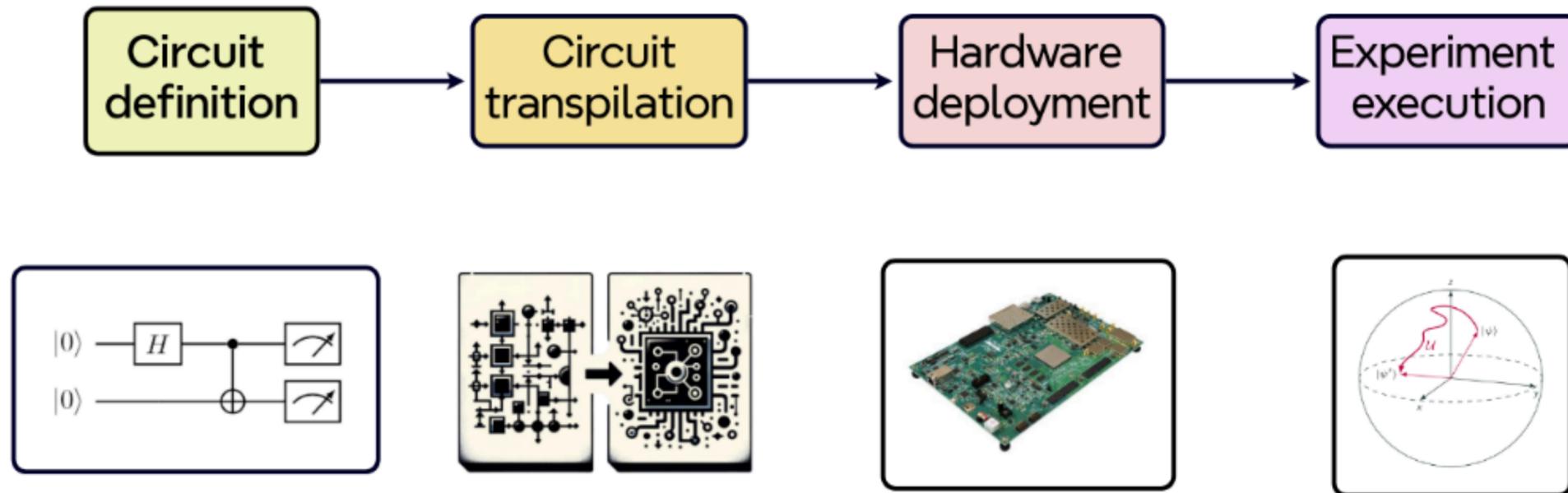
An open-source solution for self-hosted quantum computing setups.

<https://qibo.science/>

# Qibolab

Efthymiou, S., Orgaz-Fuertes, A., Carobene, R., Cereijo, J., Pasquale, A., et al. (2024).  
Qibolab: an open-source hybrid quantum operating system.  
*Quantum*, 8, 1247. <https://doi.org/10.22331/q-2024-02-12-1247>

# How to control qubits?



# Introducing Qibolab

We upgraded Qibo to run on hardware by:

- introducing API for **microwave pulses**
- Writing drivers for electronics
- Providing compilation/transpilation

```
from qibolab import create_platform

# Define platform and load specific runcard
platform = create_platform("my_platform")

# Create a pulse sequence based on native gates of qubit 0
natives = platform.natives.single_qubit[0]
sequence = natives.RX() | natives.MZ()

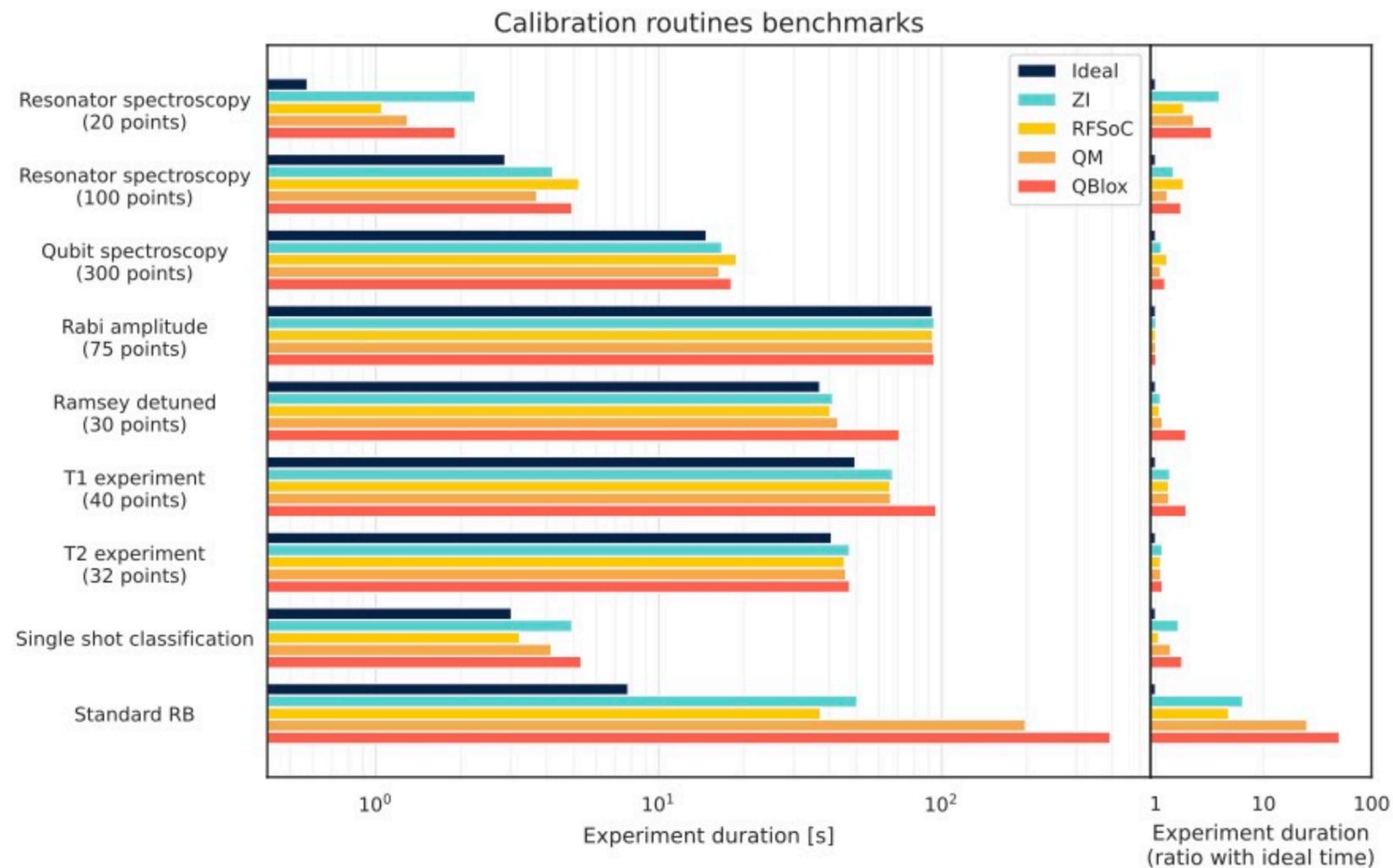
# Connects to lab instruments using the details specified in the calibration settings.
platform.connect()

# Execute a pulse sequence
results = platform.execute([sequence], nshots=1000)

# Grab the acquired shots corresponding to
# the measurement using its pulse id.
# The ``PulseSequence`` structure is list[tuple[ChannelId, Pulse]]
# therefore we need to index it appropriately
# to get the acquisition pulse
readout_id = sequence.acquisitions[0][1].id
print(results[readout_id])

# Disconnect from the instruments
platform.disconnect()
```

# Compatibility with several control electronics



We run several experiments on all the control electronics.

$$T_{\text{ideal}} = N_{\text{shots}} \times \sum_{i=1}^M (T_{\text{sequence}, i} + T_{\text{relaxation}})$$

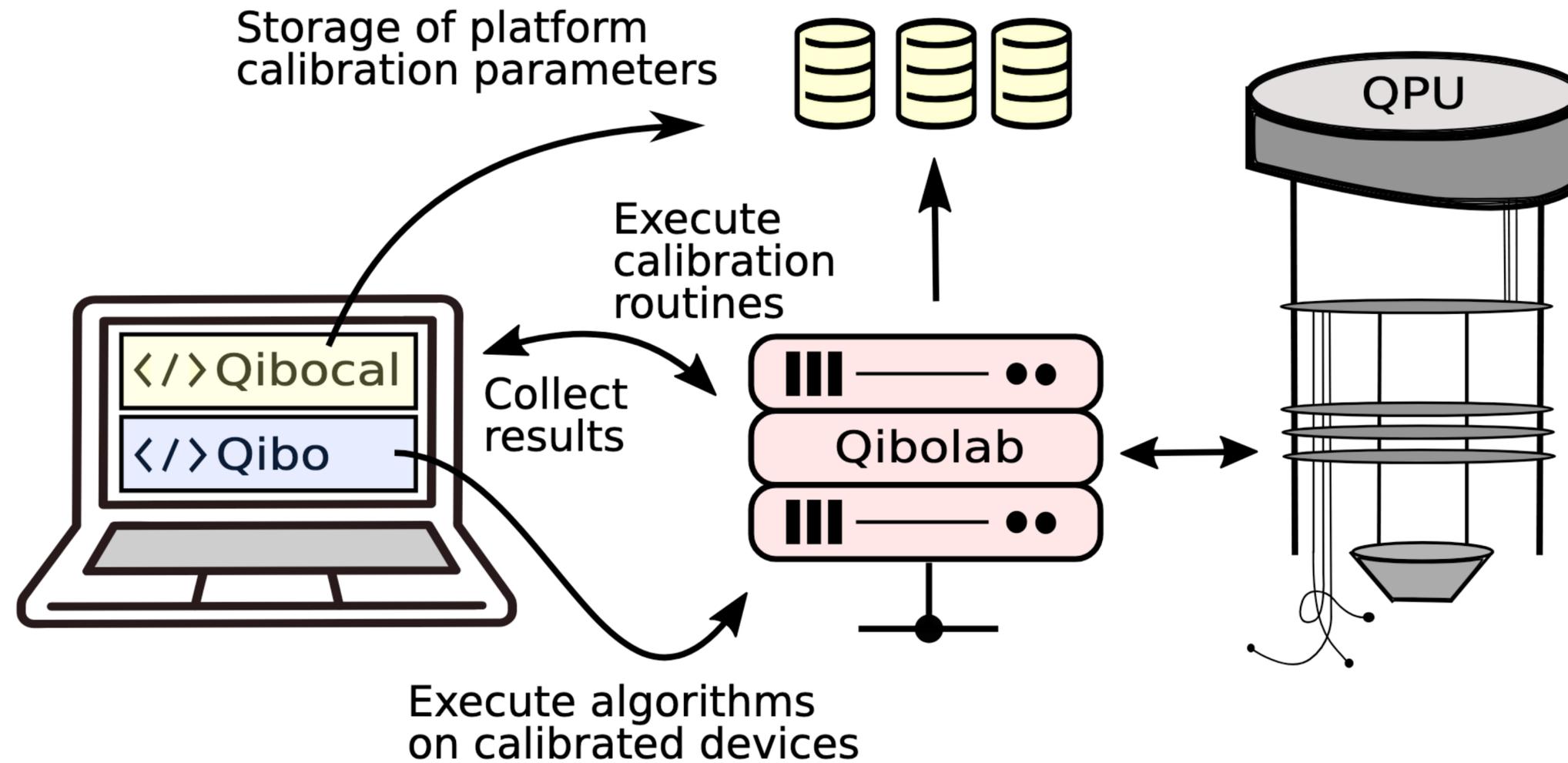
# Qibocal

Pasquale, A., & Pedicillo, E., et al. (2024).

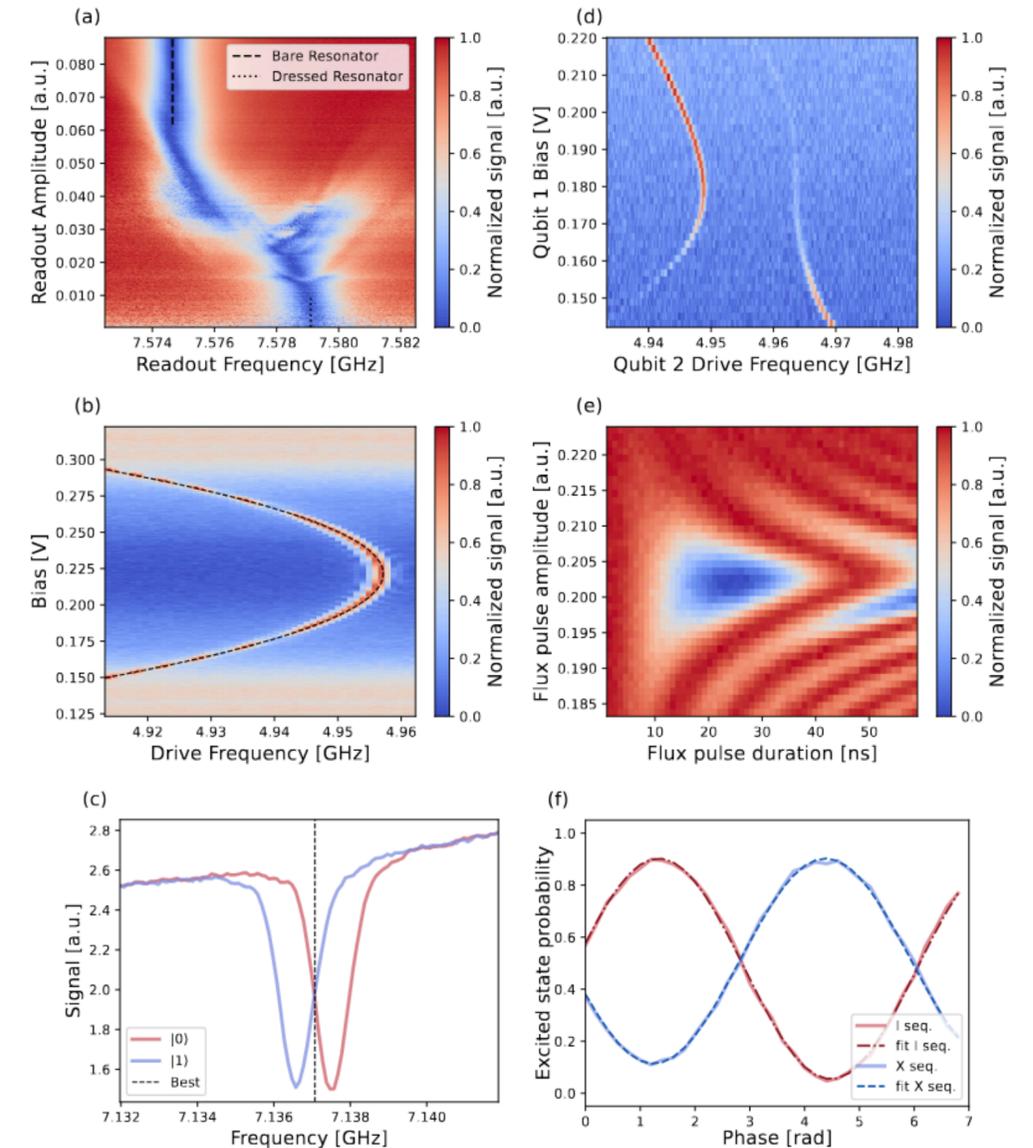
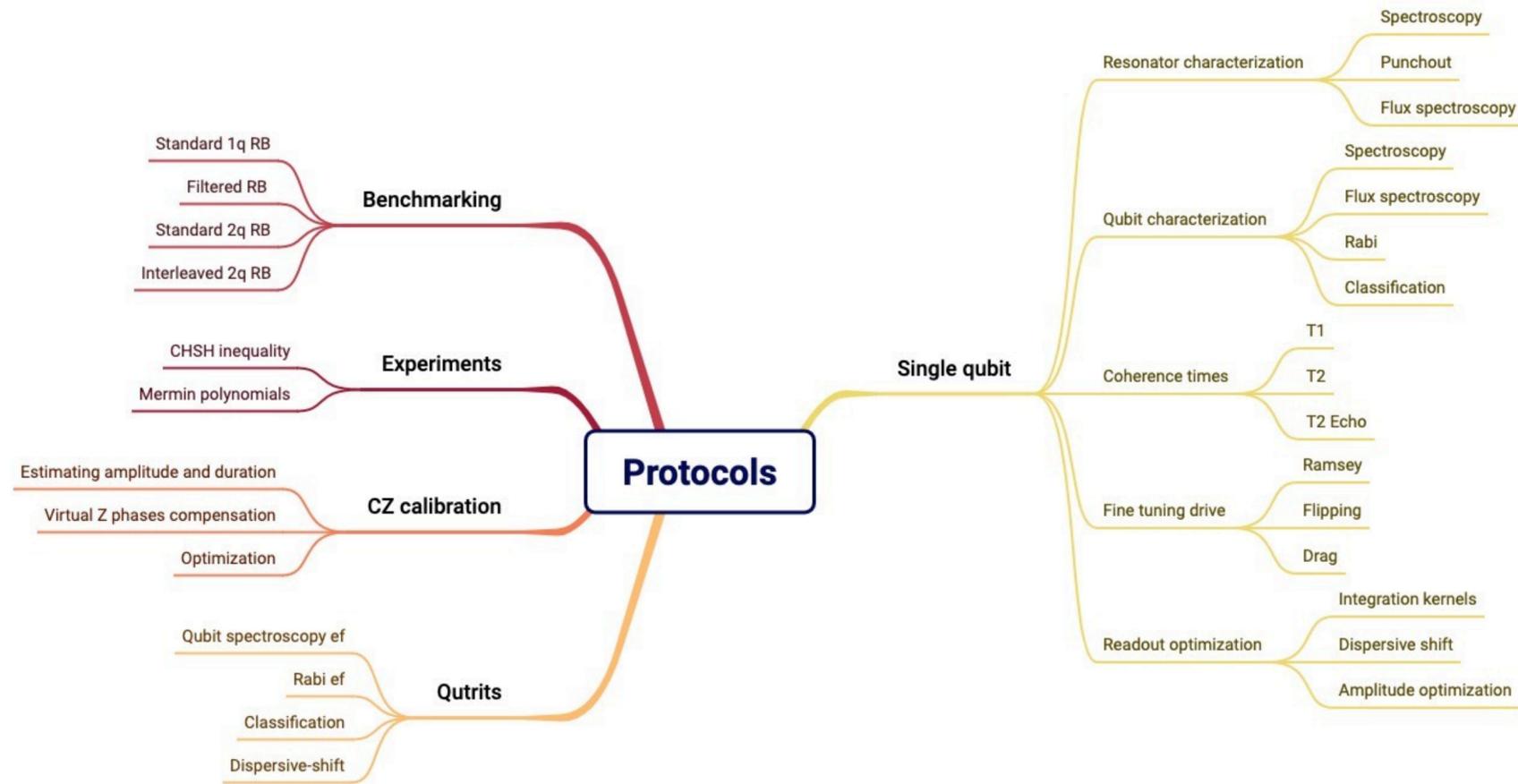
*Qibocal: an open-source framework for calibration of self-hosted quantum devices.*

arXiv. <https://arxiv.org/abs/2410.00101>

# What can Qibocal do?



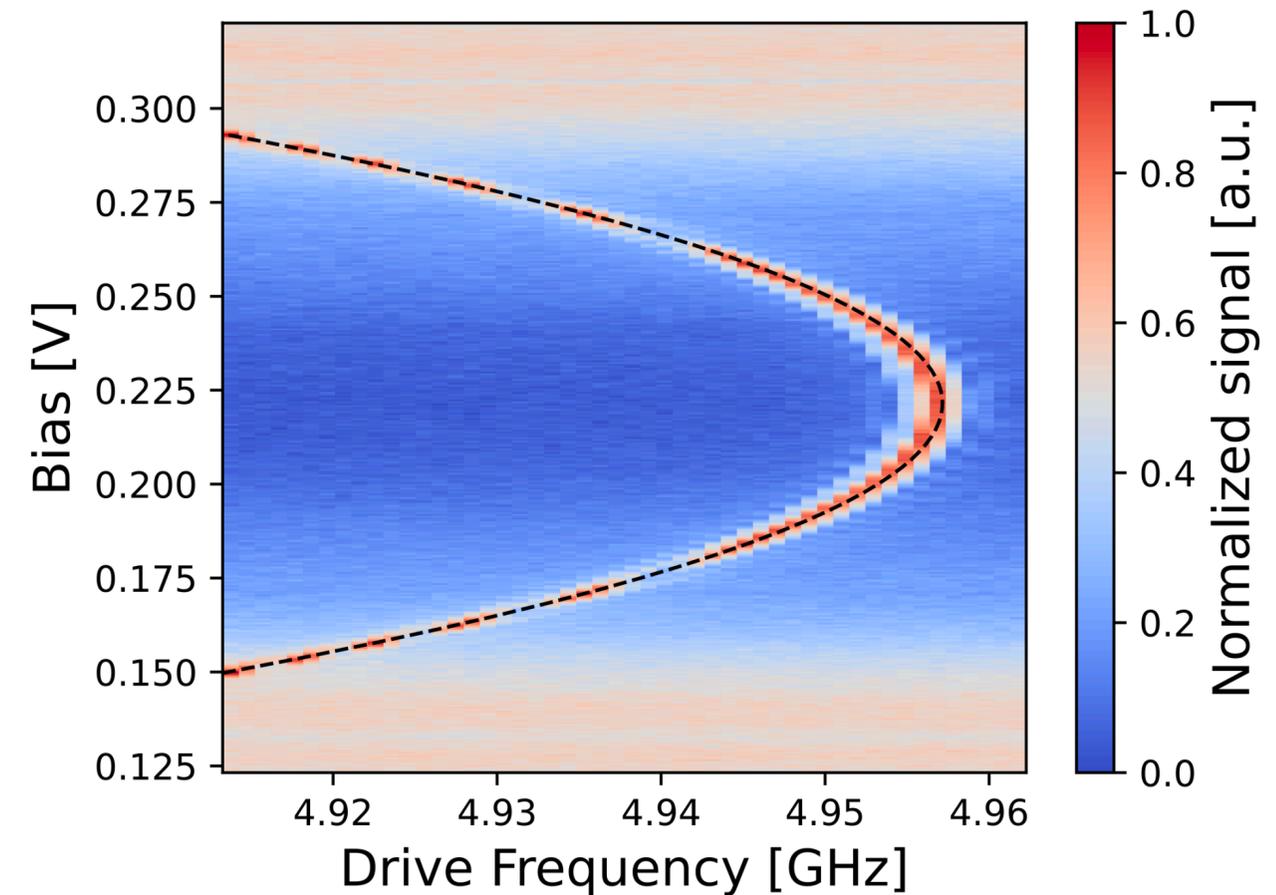
# Experiments available in Qibocal



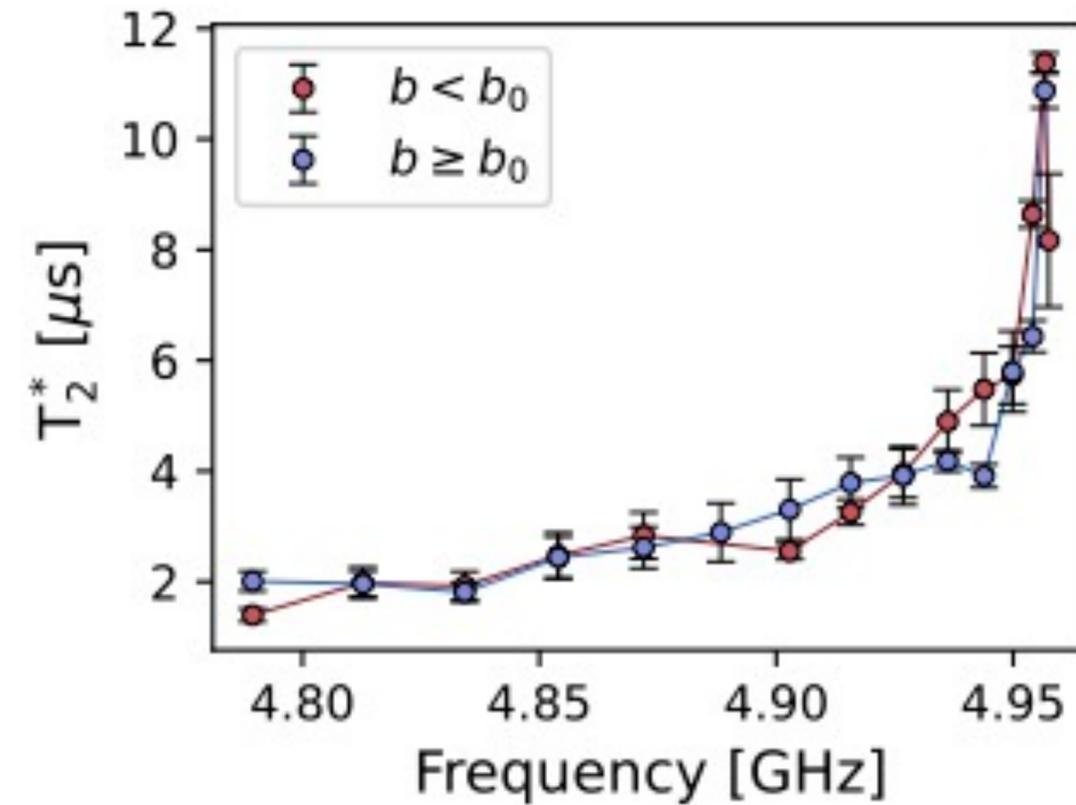
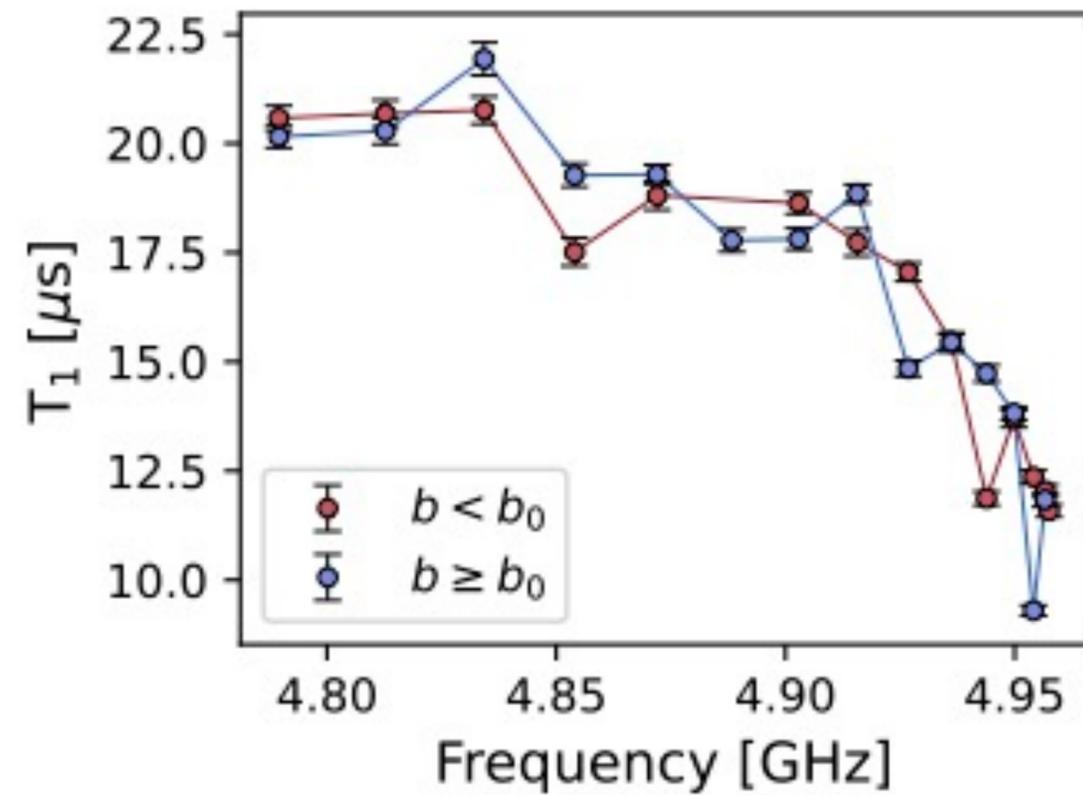
# More complicated experiments

In this experiment we bias the qubit away from the sweet-spot and after a recalibration which includes:

- Estimating qubit frequency according to new bias
- Rabi experiment
- Single shot classification
- Ramsey
- Extract  $T_1$  and  $T_2$

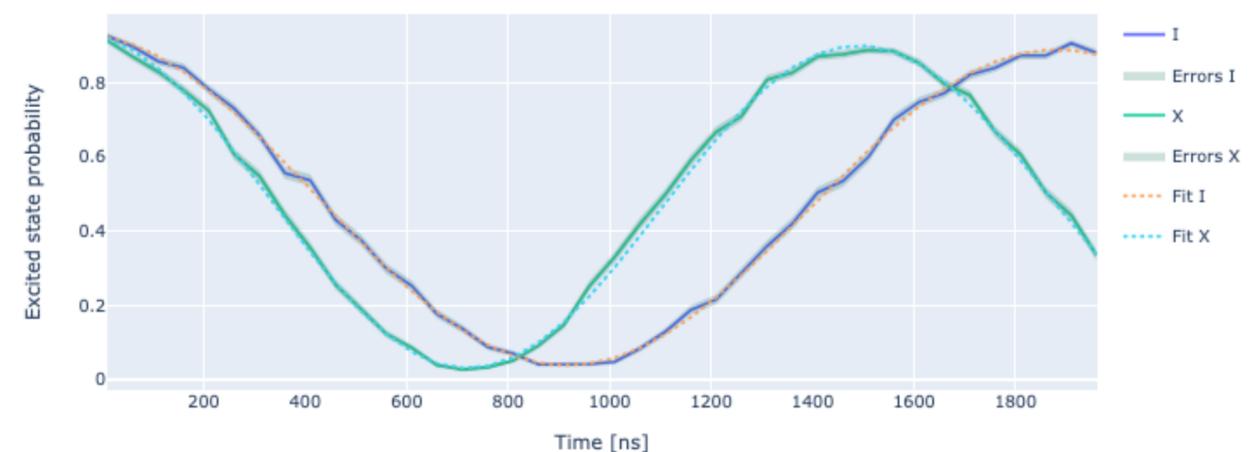
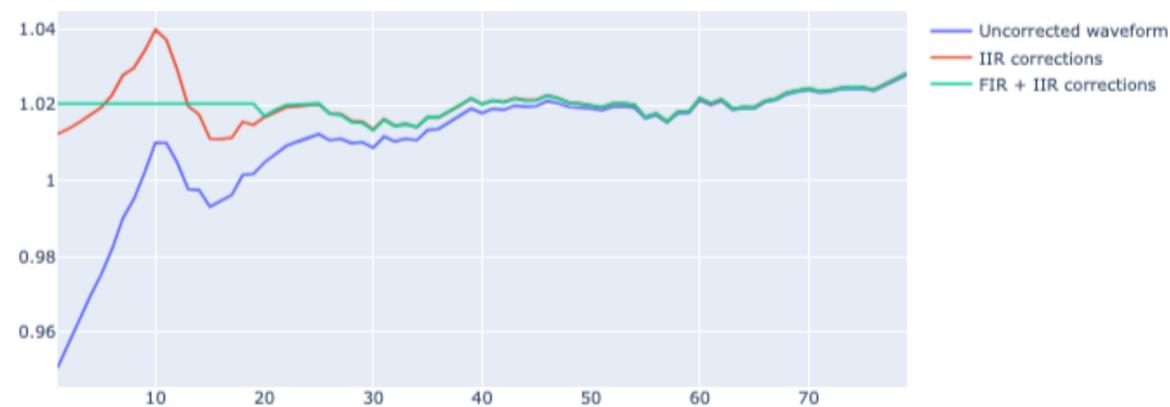


# Study qubit decoherence away from sweet spot



# Current Qibocal developments

- Update to Qibolab 0.2 just completed
- Incoming refactor to simplify protocols deployment
- Cross resonance protocols
- Measure ZZ interaction
- Correcting pre-distortions to improve CZ



# Outlook

We have presented the Qibo project which is currently funded by the QNIX project.

The compatibility with RFSoc FPGAs drivers currently under development will enable to run Qibo on the devices that will be tested in the QUART&T project.

Everyone is invited to contribute to Qibo, we are on GitHub: <https://github.com/qiboteam/qibo>

**Thanks for listening**