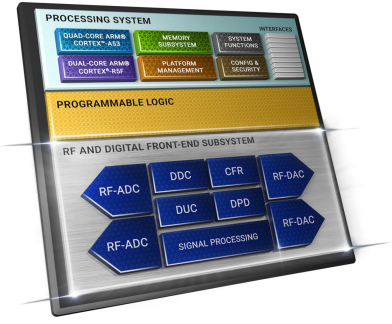
Qibosoq: an open-source framework for quantum circuit RFSoC programming

Rodolfo Carobene, A Candido, J Serrano,
A Orgaz-Fuertes, A Giachero, S Carrazza

1st FPGA Developers' Forum (FDF) 12/06/2024

University of Milano-Bicocca, INFN

arXiv:2310.05851
github.com/qiboteam/qibosoq
qibo.science



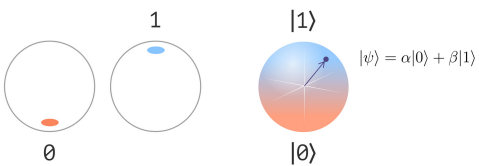
Outline:

1. Quantum Computing
2. Superconducting Quantum Computing
3. RFSoC boards
4. Qibosoq: Qibo Server On QICK
5. Future development

Quantum Computing

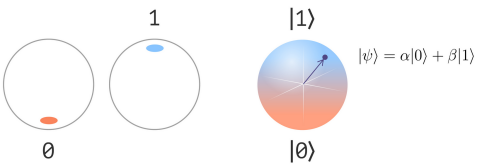
What is quantum computing?

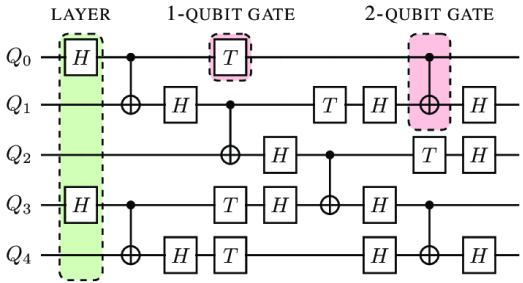
Computing exploiting quantum mechanical phenomena: in particular *superposition*, *entanglement* and *interference*



What is quantum computing?

Computing exploiting quantum mechanical phenomena: in particular *superposition*, *entanglement* and *interference*

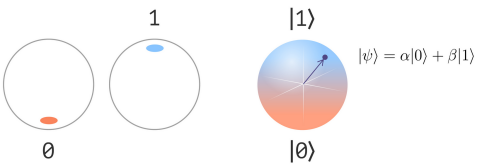


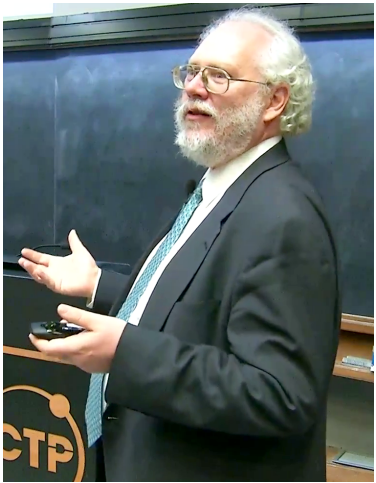


DOI: 10.1109/TQE.2021.3053921

What is quantum computing?

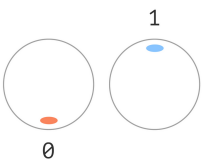
Computing exploiting quantum mechanical phenomena: in particular *superposition*, *entanglement* and *interference*

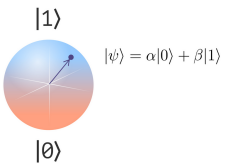


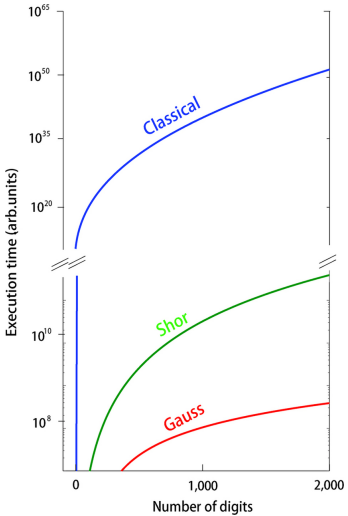


What is quantum computing?

Computing exploiting quantum mechanical phenomena: in particular *superposition*, *entanglement* and *interference*





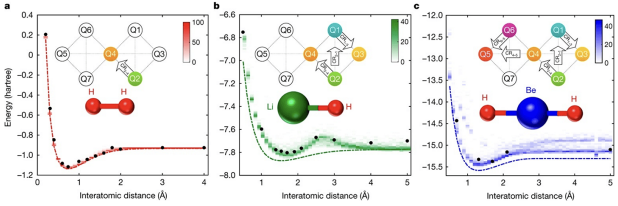


DOI: 10.1038/srep00260

What for?

Applications of Quantum Computing

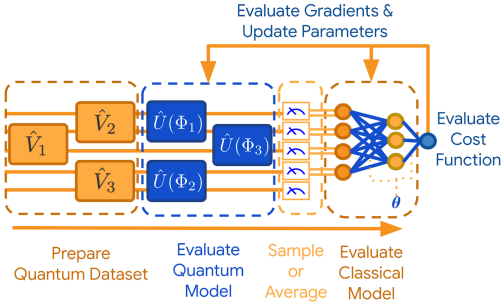
- **Simulations**: *"Nature isn't classical, dammit, and if you want to make a simulation of nature, you'd better make it quantum mechanical, and by golly it's a wonderful problem, because it doesn't look so easy"*



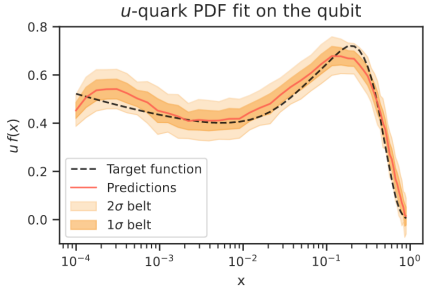
DOI: 1038/nature23879

Applications of Quantum Computing

- **Simulations**
- **Machine Learning**: quantum computing is *differentiable* and circuits can behave naturally as a Neural Network



CREDITS: blog.paperspace.com

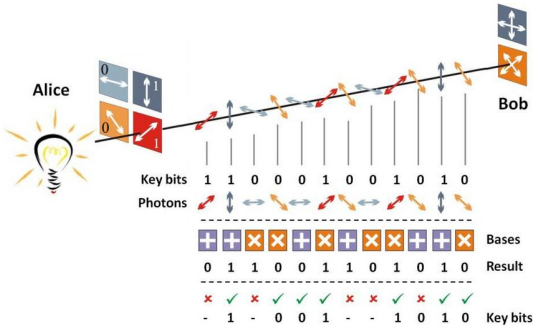


DOI: arXiv:2308.06313
DOI: arXiv:2308.05657

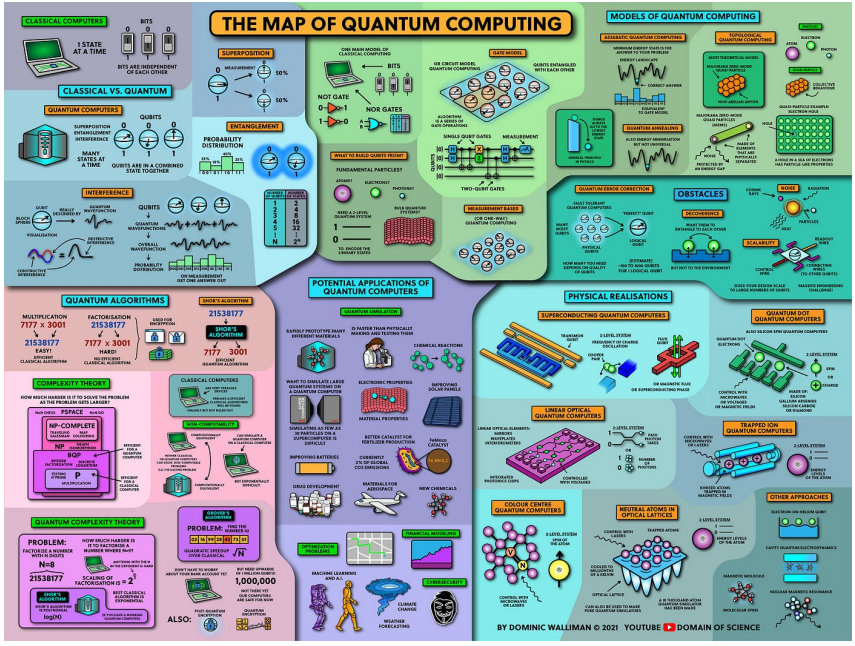
QML promises better scalability but has not yet show a practical "advantage"

Applications of Quantum Computing

- **Simulations**
- **Machine Learning**
- **Cybersecurity (BB-84)**
- **Batteries** (faster and more efficient)
- **Optimization and modeling** (finance, traffic, weather...)



DOI: arXiv:2003.06557
DOI: 10.1007/978-3-319-30201-0_27



CREDITS: Domain of Science

Quantum computer technologies

Superconducting circuits

- Low coherence times
- Fast operations
- Configurable
- Supported by Google, IBM, Rigetti

Ions

- High coherence times
- Slow operations
- Easily entangled
- Supported by IonQ, Honeywell

Photonics

- Easily scalable
- Specific program requires specific chip
- PsiQuantum, Xanadu

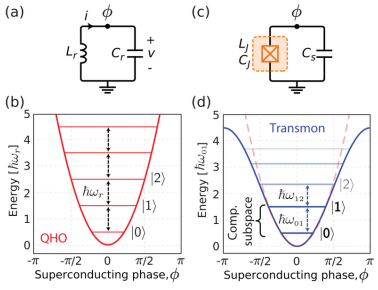
Quantum dots, **Diamond vacancies**, **Neutral atoms**

Many other technologies are being studied: there is yet no clear winner!

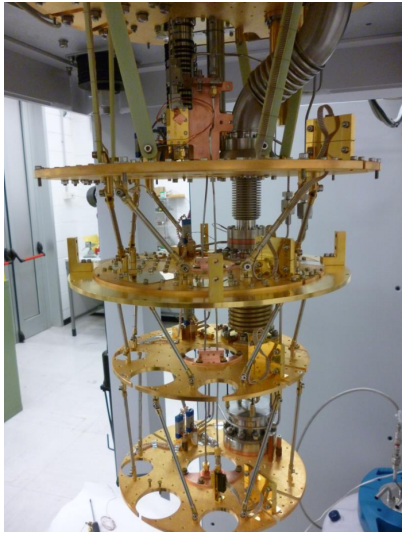
Superconducting Quantum Computing

Artificial atoms in cryostats

Superconducting qubits exploit non-linear inductors (**Josephson Junctions**) to build anharmonic oscillators where it is possible to restrict the subspace to a Two Level System

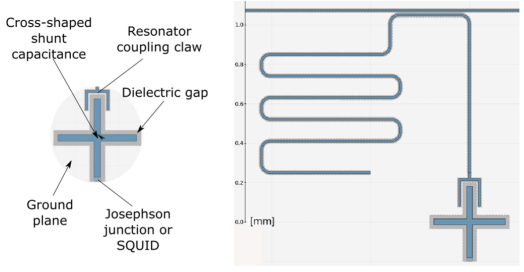


DOI: 10.1063/1.5089550

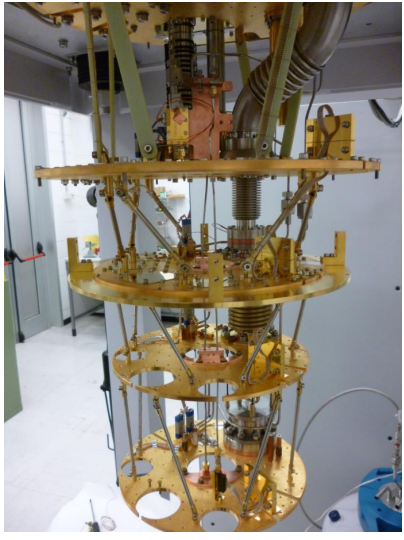


Artificial atoms in cryostats

Superconducting qubits exploit non-linear inductors (**Josephson Junctions**) to build anharmonic oscillators where it is possible to restrict the subspace to a Two Level System



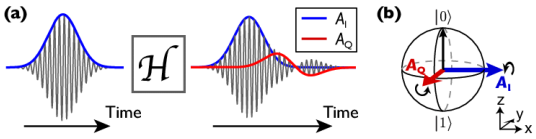
10.1109/TASC.2024.3350582
R. Moretti et al. (2024)



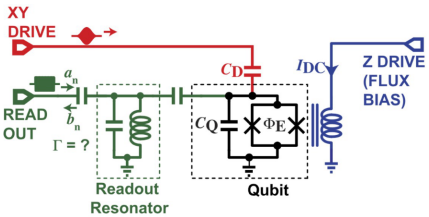
Readout and control

Control (gates) and Readout of the qubit is doen with high-frequency pulses.

- Duration: ~ 40 ns - $\sim 2\,\mu$s
- Frequency ~ 4 GHz - ~ 10 GHz



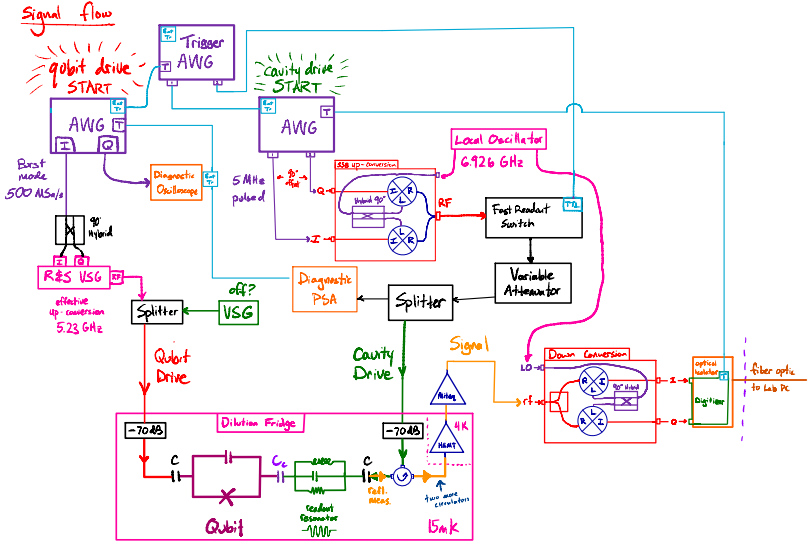
DOI: 10.1103/PhysRevLett.110.040502



DOI: 10.1109/JSSC.2019.2937234

The system is extremely sensitive to noise: low-latency, low-noise, high-sampling is key

Readout and control: multiple-instruments



What can we put inside a single FPGA?

CREDIT: J Ramirez (NIST)

RFSoC boards

All-in-one solution: Radio Frequency System on Chip (RFSoC)

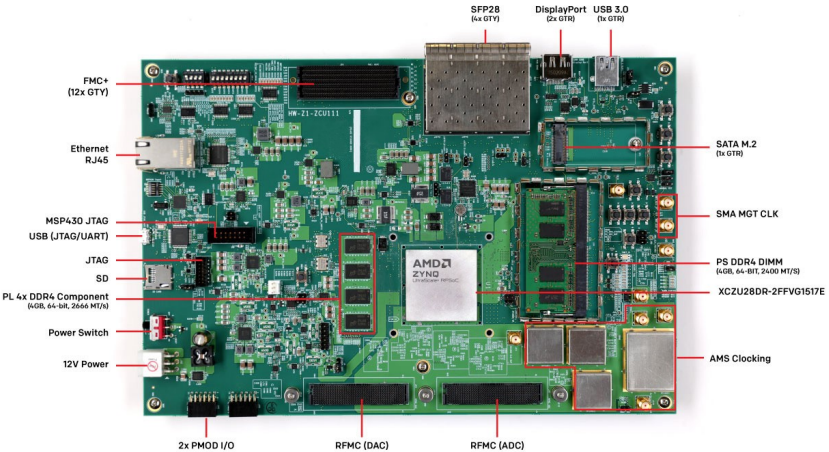
What do we require?

- Speed
- High sampling rate (DACs and ADCs)
- Configurability
- Scalability
- Cost-effectivness
- Usability

All-in-one solution: Radio Frequency System on Chip (RFSoC)

What do we require?

- Speed
- High sampling rate (DACs and ADCs)
- Configurability
- Scalability
- Cost-effectivness
- Usability



CREDITS: AMD

RFSoC firmwares for Quantum Control

QICK

- Fully Open-Source
- Produced by FNAL
- Single timed-processor
- New version in beta-testing
- Large community
- DOI: arXiv:2110.00557

QubiC

- Partially Open-Source
- Produced by Berkley
- Multiple Sequencer: one for each DAC
- DOI: arXiv:2309.10333

Presto

- Proprietary
- Produced by Intermodulation Products
- Multiple Sequencer: one for each DAC
- DOI: arXiv:2205.15253

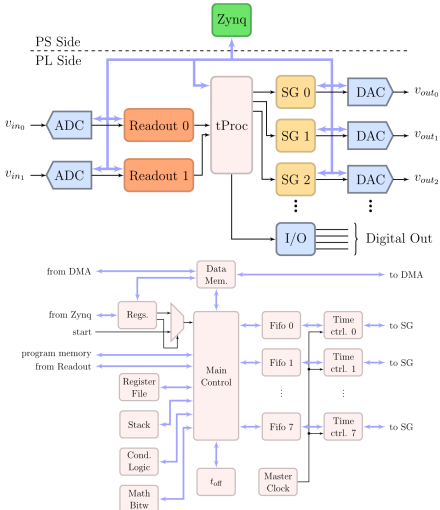
These projects all share the use of some tools from DSP: Direct Digital Synthesis (DDS) and Digital Down Conversion (DDC) to work at high-frequencies

QICK: Quantum Instrumentation Control Kit

QICK exploits Python Productivity for ZYNQ (**PYNQ**) to enable communication between the PS and PL part of the SoC. To coordinate DACs and ADCs it uses a **single timed-processor** that can be programmed with an **assembly-like language**.

It is currently used in many labs around the world.

DOI: arXiv.2110.00557



QICK: Quantum Instrumentation Control Kit

QICK exploits Python Productivity for ZYNQ (**PYNQ**) to enable communication between the PS and PL part of the SoC. To coordinate DACs and ADCs it uses a **single timed-processor** that can be programmed with an **assembly-like language**.
It is currently used in many labs around the world.

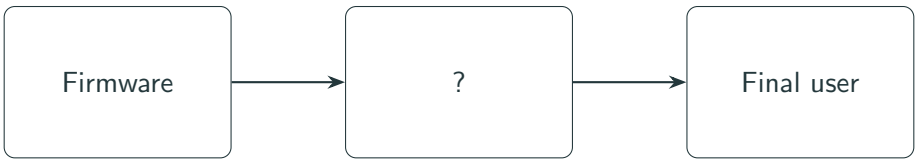
DOI: arXiv.2110.00557

- Limited to pulse-level operations
- Scalability (multi-board syncronization) is complex
- Software (assembly-based) is not really user-friendly
- Limited Configurability

Qibosoq: Qibo Server On QICK

How to provide usability?

How to best bridge the gap between a complex, yet working, firmware and a “clueless” user?



How to provide usability?

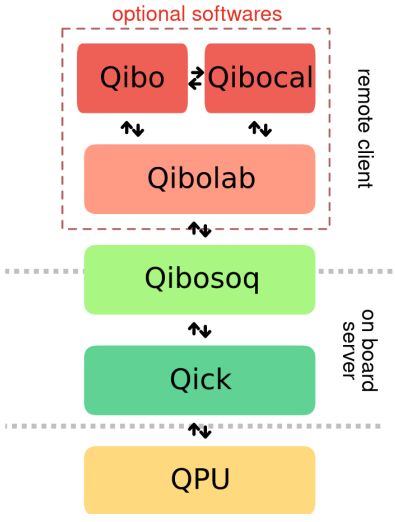
How to best bridge the gap between a complex, yet working, firmware and a “clueless” user?

Qibosoq integrates the QICK-supported RFSoCs in the QIBO framework and provides a simple, fast and intuitive means to control qubits

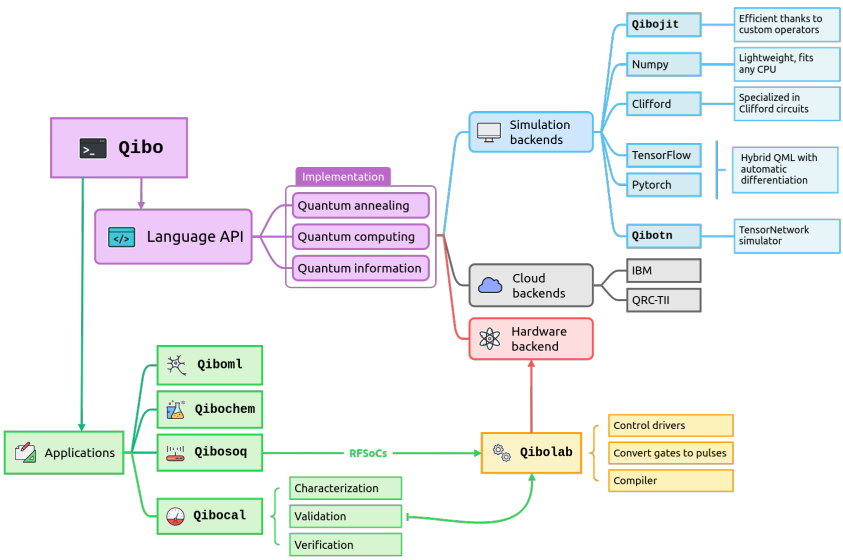
Qibosoq is an open-source software package which extends Qick's potential to execute quantum algorithms on self-hosted quantum hardware platforms through Qibo.

Qibosoq

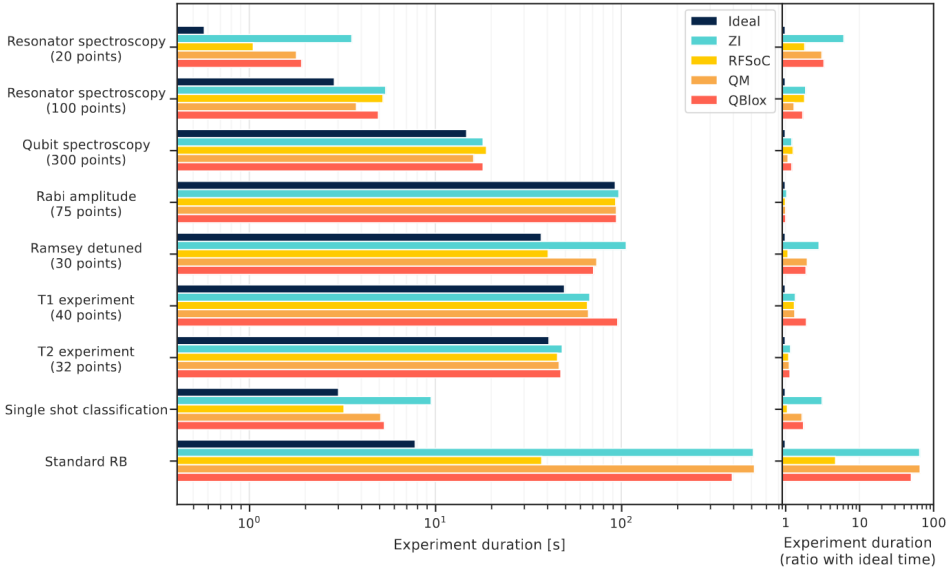
- Python API to easily write QICK programs
- Standardized client-server communication protocol
- Integrated into the Qibo framework:
  - Qibo: Simulations and algorithm deployment (gate-based)
  - Qibolab: Unified control of instruments (pulse-based)
  - Qibocal: Calibration experiments



Qibo framework: qibo.science

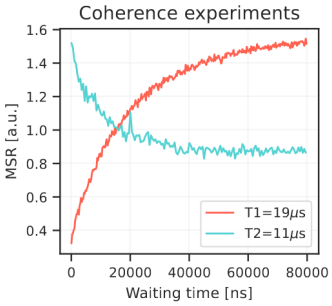


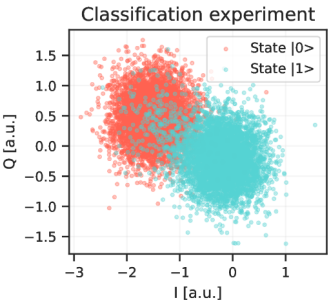
Speed benchmark

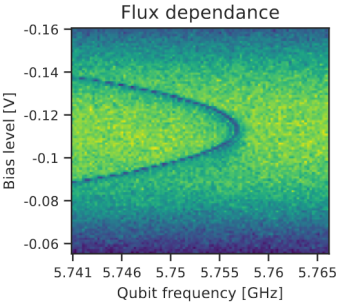


DOI: arXiv:2308.06313

Pulse-calibration and experiments







Qibosoq: now

- Code is fully open-source and issues/pull_requests are welcomed!
- Community is consolidating: new labs starting to use it for different type of qubits
- Actual development is almost stopped, while waiting for QICK 2.0
- Qibosoq and RFSoCs effectively cut by 10 the cost of controlling a QPU

Qibosoq is now used in multiple labs:

Future development

Firmware modfications

If the objective is to build a platform for multiple purposes that can be used by anyone, we still have a lot of work to do:

- To provide a scalable platform we need to leve the single-processor model and rather structure a "**controller**" for each DAC/ADC that can be configured indipendently
- The RFSoCs that we usually use have a lot of **not-used hardware** that could provide many benefits
- The possibility to modify the firmware is still licensed and requires skills hard to master, we should find a way to provide multiple configurable firmware blocks pre-compiled! $\rightarrow$ **partial reconfiguration**

Thank you for the attention!

And special thanks to my group in Milano-Bicocca, my supervisor (prof. A Giachero), my PhD collegues and the whole group in TII.

Research supported by the national centers NQSTI (PE0000023) and ICSC (CN00000013)