QUANTUM MACHINES

TUTORIAL

How to Dramatically Increase the Initialization Fidelity of Your Qubits with QUA

Here's how to use Active Reset with the pulse-level programming language QUA to reduce qubit initialization time and improve its fidelity.

INTRODUCTION

Qubit initialization is crucial for useful quantum computing, and there are various methods in which it is implemented. Whenever we see circuits showing various algorithms, we take it as a given that the qubits all start out in their ground state. However, in reality, we cannot take this fact for granted. Qubit reset, a technique used to bring the qubits to their ground state, can be implemented using various methods. The simplest and least demanding method for qubit reset, or qubit state initialization, is to wait for the qubits to decay to their thermal state. While this is indeed a possibility, it suffers from several drawbacks. Firstly, it's very time costly, as you need to wait for the qubits to fall back to their ground state. In fact, as qubit lifetimes steadily improve, this time is expected to grow prohibitively long and become a major bottleneck for achieving high throughput quantum circuits. Secondly, there's always spontaneous excitation that occurs; there's always a chance that the qubit is not actually in the ground state, thus causing the preparation fidelity to be limited by the thermal distribution of the qubit states.

Active reset tackles both of these problems. It 1) reduces the qubit initialization time, and 2) improves the probability that the qubit is indeed in the ground state after the initialization. Here we will show you how a 3-way collaboration between Q-CTRL, using their Boulder Opal, the DuBois lab at LLNL, and Quantum Machines' Quantum Orchestration Platform, allows you to perform highly efficient qubit reset protocols with state machine logic that is cleverly optimized for superconducting qubits. If you want to implement similar Q-CTRL protocols with QUA, please refer to the how-to guide for integrating Q-CTRL's Boulder Opal with QUA, and the Q-CTRL python add-on package for QUA.

There are several figures of merit which need to be considered when examining various initialization schemes. Specifically, we assess the quality of initialization by measuring two parameters: the ground state preparation probability, and the time required to reach that probability. We can thus plot various initialization schemes, where the x-axis is the preparation time, and the y-axis is the preparation success probability. The closer the curve to the top left (low preparation time, high success probability), the better. An example of such a family of curves can be seen in Figure 1.

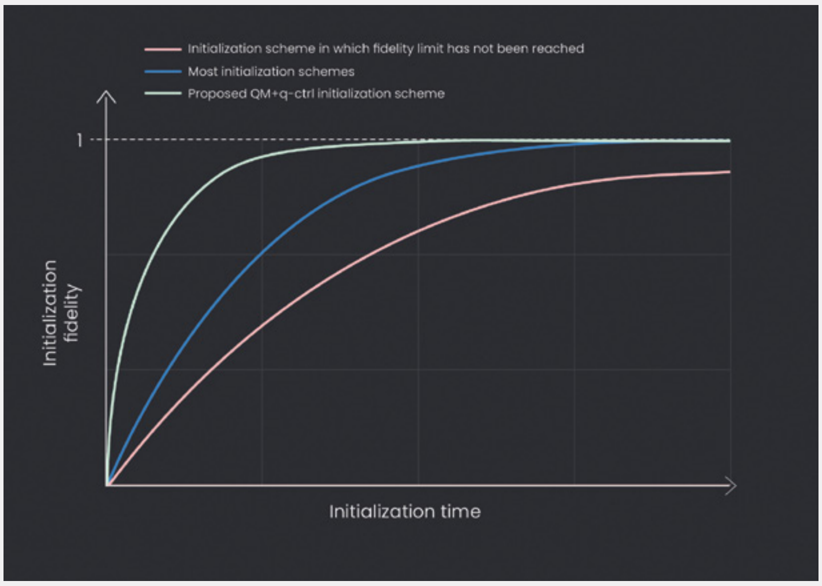


Figure 1: *Schematic depiction of fidelity over time for various initialization schemes. The red line represents an initialization scheme that cannot, in principle, reach unit fidelity. The blue and green schemes can both reach unit fidelity, but the green one will achieve a given fidelity for a shorter initialization time than the blue and is therefore superior to it.*

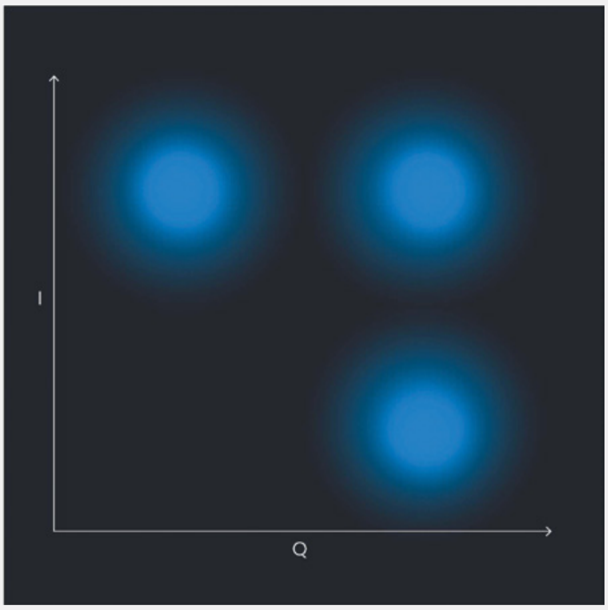


Figure 2: Qubit readout: Schematic depiction of IQ plane with qubit state measurements. The bottom right quadrant represents a qubit in the g state, the top right the e state, and the top left quadrant the f state.

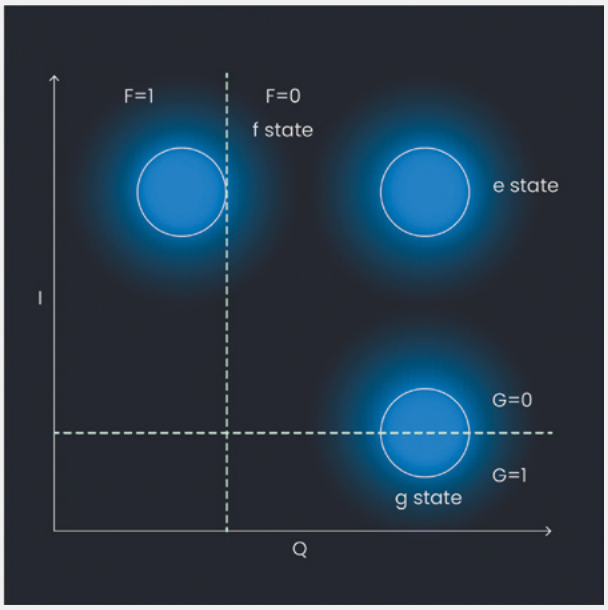


Figure 3: Qubit readout IQ plane with decision regions plotted.

Now let's discuss what qubit readout looks like in this case. In Figure 2 we show an IQ plane diagram. This is a histogram made up of many points, where each point is the result of an individual measurement. We can see that the measurement results cluster in three "blobs" which are Gaussian distributions, each of which corresponds to ground (g), first-excited (e), and second-excited (f) states. We use this histogram to divide the IQ plane into three "decision regions", shown in Figure 3). Once we've determined these regions, we classify each subsequent measurement as being either e, g, or f by the region it belongs to. Here we define a boolean with 0 as false, 1 as true. We introduce thresholds represented as dashed lines, with g_bound being the threshold between the ground and excited state, and e_threshold being the threshold between the excited and f state.

QUBIT INITIALIZATION IMPLEMENTATION

Let's first discuss the basic idea of active reset by showing the simplest example of an active reset protocol. We start with a measurement of our qubit, and if the result points to the excited state, we play a pi pulse. This is a very simple protocol, as can be seen in the code block below; we just need to include one threshold bound which differentiates between the ground and excited state (here dubbed `g_bound`). While this method is very fast, it typically does not allow us to reach very high fidelities due to measurement errors, discrimination errors, and pi pulse errors. That's why we need to go a step further, which is the goal of this demonstration.

```
measure('meas_op_res', 'res', None,\
                q.demod.full('integ_w_s', Q))

With if_(Q > g_bound):
    play('pi', 'qubit')
```

Now, let's discuss the wait-until-success active reset scheme we propose. First, we introduce the code, which looks like this, with `g_bound` being the ground state threshold and e_bound being the excited state threshold.

We will introduce the following time variables (these times are general, and are all quite conservative and can be reduced to decrease run-time):

- `waitTime:` minimal wait time between readout, around 1 us.
- `buffer:` buffer added in case the user has measured something earlier; 10 us as a buffer.
- `T_1:` lifetime of cavity with some buffer (for example, lifetime x 6 = 6.4 us)
- `'Integ_w_c/integ_w_s':` demodulation weights

```
def initialize(I, Q, g_bound, e_bound, waitTime):
    wait(buffer, 'resonator')
    measure('meas_op_res', 'res', None,\
                      demod.full('integ_w_c', I),\
                      demod.full('integ_w_s', Q))
    with while_(Q > g_bound):
        wait(waitTime, 'res')
        align(resonator, 'qubit')
        play('pi01', 'drive01')


        measure('meas_op_res', 'res', None,\
            demod.full('integ_w_c', I),\
            demod.full('integ_w_s', Q))

        with while_(I < e_bound):
            wait(waitTime,'res')
            align('resonator', 'qubit')
            play('pi12', 'drive12')
            align('resonator', 'qubit')
            measure('meas_op_res', 'res', None,\
                demod.full('integ_w_c', I),\
                demod.full('integ_w_s', Q))
        wait(t_1, 'res')
    wait(7500,'res')
```

Now let's break it down a little. The idea in this code is that we want to constantly assess if we're in the ground, excited, or f state (f being any higher-order state). We look at the outputs we get from our qubit, namely the I and Q values (in-phase and quadrature components) [1]. We know which Q values correspond to the Gaussian distribution of the various states we're examining. We place two bounds in the IQ plane. The first one, `g_bound`, distinguishes between the g state and all other states. We place this bound close to the g Gaussian peak. As we lower the value of the bound, we will classify fewer and fewer points as ground – only those for which we have more certainty. Thus, we can use this parameter to control the fidelity of the initialization scheme – as we decrease it we will increase the fidelity, but spend more time in the initialization scheme. This is a general property of initialization schemes.

If Q < `g_bound`, we classify the measurement as being definitely in the ground state. Otherwise, we assume we're in the excited state and thus play a pi pulse to get to the ground state. We then check if we're in the f state by checking whether or not we've passed `e_bound` (the bound between the excited and f state). If so, we apply a pi pulse between the e and f states and try again. There's a chance it's now in the ground state, so we check if that is the case by running the loop again. If not, we will be in the excited state and thus apply another pi pulse. We loop over this until we're sure the qubit is in the ground state.

Something that we must recognize here is that the OPX+ and QUA give us the flexibility to perform complex decision sequences and state machine implementations that take into account higher excited states. This can be seen in Figure 4. There is no hardcoded logic or specific assumptions made in the design of the OPX+ to enable this sequence, and many sequences such as this one can be created by you, limited only by your imagination! (And feedback latency, which is approximately 200 ns for the OPX+).

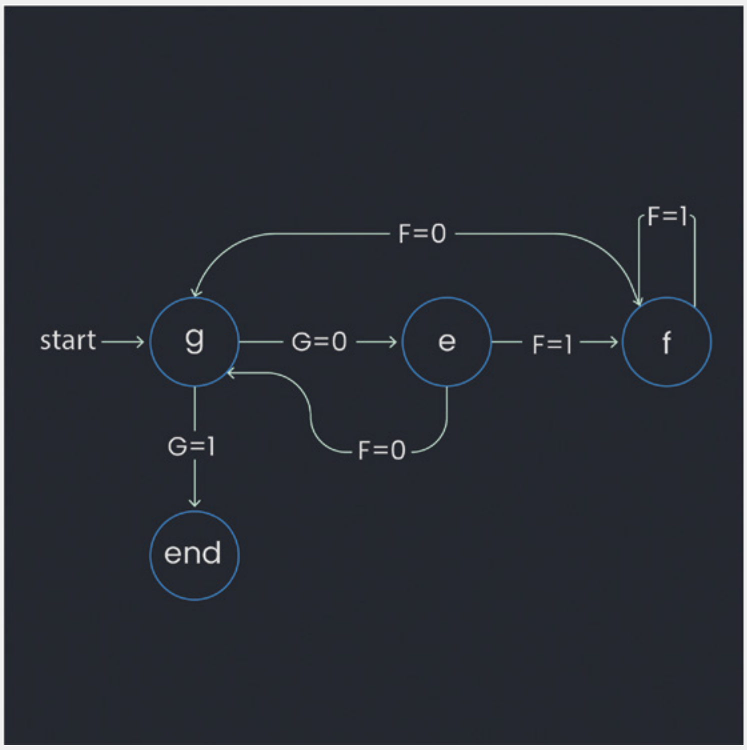


Figure 4: *State machine for qubit initialization scheme. See explanation in the main text.*

ACTIVE RESET RESULTS

We have shown you this protocol which we proposed to run, and we can wax poetic about how much better this works. However, we know you won't take our word at face value. Here, we show you some results we've obtained.

Figure 5 shows the success probability using a confusion matrix. The x-axis displays the measured states, while the y-axis the prepared states. To obtain the statistics, we perform an active reset sequence, then prepare a state and immediately measure and classify the result. To have the highest fidelity, we want the prepared and measured states to be the same for each given 0 (g), 1 (e), and 2 (f) state. Thus, we want the highest probabilities on the diagonals of the confusion matrix. For clarity, the measurements without active reset involve an 8 ms wait, the thermalization method discussed in the introduction. As shown in the figure, active reset increases these diagonal probabilities by quite a large margin. To quantify the results: the residual thermal population without active reset is 4.77%, while with active reset it drops to 0.94%.

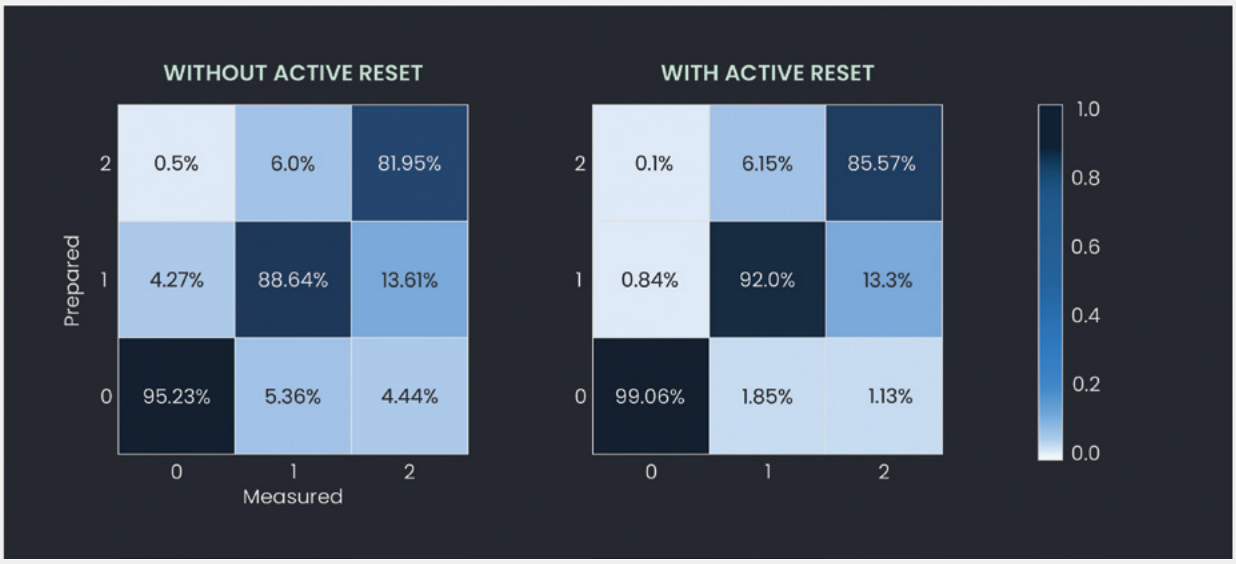


Figure 5: *Confusion matrix showing the measured and prepared state results with and without active reset.*

We hope we've convinced you about our earlier claim that active reset improves the probability that the qubit is indeed in the ground state after the initialization. Next, let's discuss the claim that this active reset protocol reduces the qubit initialization time. We do this by running various experiments on the qubit and measuring the run times. This was done in collaboration with Q-CTRL, which provided the algorithms used in the experiments (active reset enables a wider range of optimization techniques on high coherence qubits, for example, closed-loop SU(3)). The results can be seen in Figure 6. This is not a minor improvement: Note that the y-axis is logarithmic! We can see that the open-loop SU(2) calibration time, for example, dropped by more than 2 orders of magnitude. Additionally, the closed-loop SU(3) experiment time dropped from a day, making it impractical to apply, to about 45 minutes, a far more reasonable time.

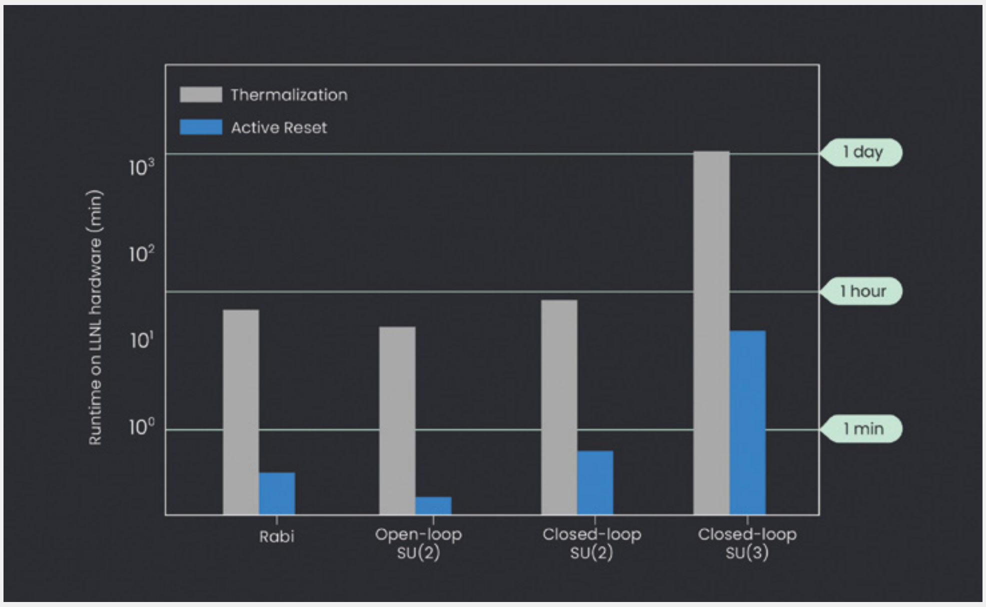


Figure 6**:** *Various qubit operations and their runtime on LLNL hardware. These measurements were done on thermalized qubits (orange), as well as qubits which were prepared using the active reset protocol introduced here (blue). Note the* ***logarithmic*** *y-axis.*

As an aside, we will show you an example of how this new active reset protocol actually cleans up real-life experiments. In Figure 7 we show the results when we perform a Power Rabi measurement, using thermalization-based qubit reset as well as active reset. Not only is the experimental runtime faster, as shown in Figure 6, but the measurement results become cleaner, yielding more conclusive and regular oscillations. This can have a profound impact on future research in superconducting qubits.

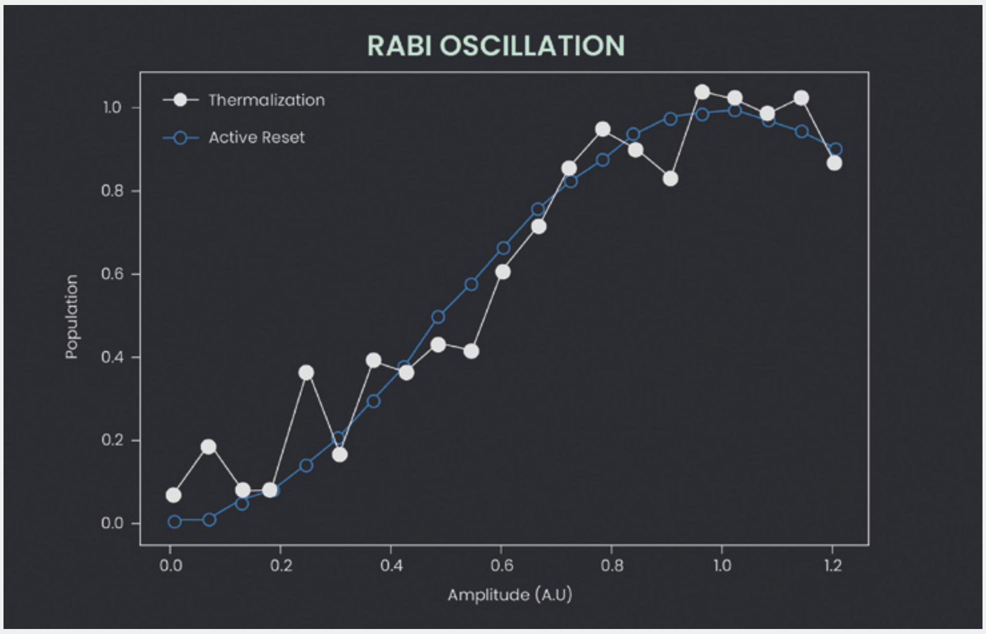


Figure 7: *Power Rabi measurement with thermal-based initialization (white) and active reset initialization (blue) for the same run time. The SNR of the active reset method is clearly superior to the thermal state preparation.*

To tie it all together. We can clearly see the advantage of using the proposed active reset scheme presented, as it both increases the fidelity of the initialization (increases the probability of ground state initialization), and decreases the time it takes. This will have profound implications on experimental runtimes, both in industry and academia. Having resolved this issue, we're excited to see how much easier, and faster, the world of quantum computing will progress.

Have any ideas about the ultimate repeat until success algorithm? Any ideas for useful applications of our protocol? If so, we'd love to hear it!
Drop us a line at info@quantum-machines.co.

References

[1] P. Krantz, M. Kjaergaard, F. Yan, T. Orlando, S. Gustavsson, and W. Oliver, "A quantum engineer's guide to superconducting qubits", Applied Physics Reviews, vol. 6, no. 2, p. 021318, 2019. Available: 10.1063/1.5089550.

The Quantum Orchestration Platform

AN END TO END QUANTUM CONTROL SOLUTION TO DRIVE THE FASTEST TIME TO RESULTS, AT ANY SCALE

OPX+

RUN STATE OF THE ART EXPERIMENTS WITH EASE

An architecture designed from the ground up for quantum control, the OPX+ lets you run the quantum experiments of your dreams right from the installation. With a quantum feature-rich environment, the OPX+ is built for scale and performance. Now, you can **run the most complex quantum algorithms and experiments in a fraction of the development time**.

PULSE PROCESSING UNIT

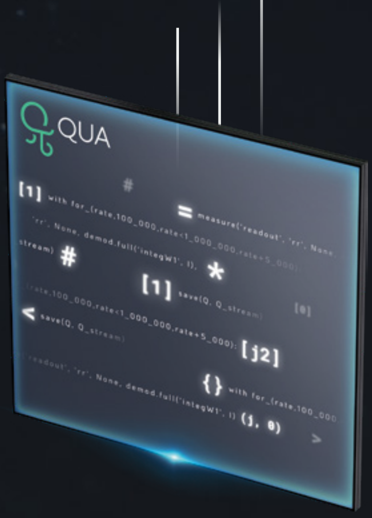
ACHIEVE THE FASTEST TIME TO RESULTS

Within the OPX+ is the Pulse Processing Unit, QM's leading-edge quantum control technology. Progress with incomparable speed and extreme flexibility. Run even the most demanding experiments efficiently, with the fastest runtimes and the lowest latencies in the industry, including quantum protocols that require **real-time waveform generation, real-time waveform acquisition, real-time comprehensive processing, and control flow.**

QUA

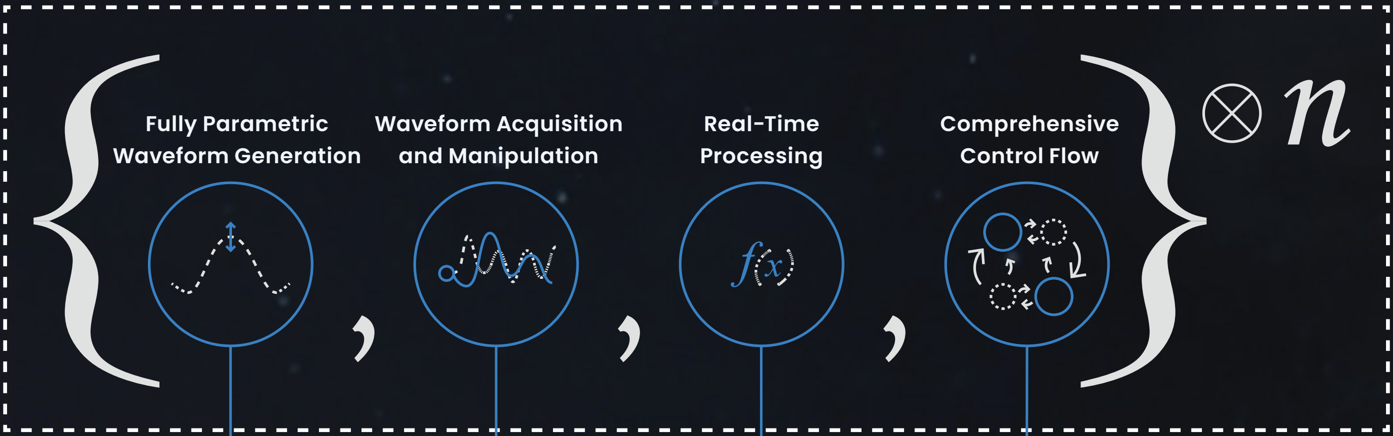
CODE QUANTUM PROGRAMS SEAMLESSLY

Implement the protocols of your wildest dreams as easily as writing pseudocode. Designed for quantum control, QUA is the first universal quantum pulse-level programming language. Code even the most advanced programs and run them with the best possible performance. Natively describe your most challenging experiments, from complex AI-based multi-qubit calibrations to multi-qubit quantum error correction.



**All of the information above is also valid for the OPX*

YOUR PROTOCOLS LIVE IN THIS PHASE SPACE


Fully Parametric Waveform Generation
Waveform Acquisition and Manipulation
Real-Time Processing
Comprehensive Control Flow
f(x)
⊗ n

Real-Time Waveform Generation

- Fully parametrized: length, frequency, phase (relative & absolute), amplitude, bandwidth, chirp
- Compensations: Crosstalk matrix, FIR and IIR filters
- And many more

Real-Time Measurements

- High fidelity analog to digital conversion
- General integration & demodulation
- Weighted Integrations accumulated integrations, sliced integrations, etc.
- Time Tagging, TTL counting

Real-Time Processing

Turing complete:
- Basic arithmetics
- Evaluation of trigonometric functions
- Vector operations
- Casting of variable types
- And more (Turing complete)

Real-Time Control Flow

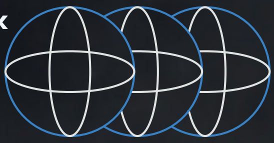
- If/else
- For loops
- While loops
- Switch case

Real-Time 'Quantum' Estimations

- State estimations
- Error estimations
- Bayesian estimations
- Correlation functions
- Neural nets based estimations

Real-Time Multi-Qubit Feedback

- Qubit stabilization & tracking
- Quantum error correction
- And many more



THE QUANTUM ORCHESTRATION PLATFORM COVERS THIS SPACE!

- Easily express quantum algorithms and experimental protocols that comprise all of the above.
- Seamlessly sync measurements, real-time calculations, and pulses of different quantum elements.
- Loop over a wide range of parameters in real-time, including intermediate frequencies, amplitudes, phases, delays, integration parameters, measurement axes, etc.
- Use if/else and switch-case statements to condition operations in real time (real time feedback).
- Define procedures (macros) to be reused in the code and access an extensive family of libraries.

If you wish to learn more:
info@quantum-machines.co

About Quantum Machines

Quantum Machines (QM) drives quantum breakthroughs that accelerate the path towards the new age of quantum computing. The company's Quantum Orchestration Platform (QOP) fundamentally redefines the control and operations architecture of quantum processors.

The full-stack hardware and software platform is capable of running even the most complex algorithms right out of the box, including quantum error correction, multi-qubit calibration, and more. Helping achieve the full potential of any quantum processor, the QOP allows for unprecedented advancement and speed-up of quantum technologies as well as the ability to scale into the thousands of qubits. Visit us at: www.quantum-machines.co