# Qadence IR for compiling Digital-Analog QPUs

Kaonan Micadei[1], Eduardo H. Matos Maschio[1], Dominik Seitz[1], and Roland Guichard[2]

[1]PASQAL, EDGE Stadium, 1075 ED Amsterdam, the Netherlands
[2]PASQAL, 3rd floor 03 164, 1 Fore St Ave, EC2Y 9DT London, United Kingdom

July 28th

**Abstract**

We introduce a novel method for merging digital and analogue quantum computation in a single pipeline, which also supports differentiable quantum sequences.

## 1 Introduction

A recent quantum computation (QC) model based on the digital-analog quantum computing (DAQC) paradigm [1] has emerged as an alternative to achieve quantum advantage on analog quantum hardware. Analog devices have risen as viable platforms for quantum computing and witnessed a remarkable shift in being routinely capable of handling thousands of qubits. Whereas digital quantum programs are typically written within the so-called quantum circuit model as a composition of operations, or gates, acting on a limited subset of non-interacting qubits, digital-analog programs present salient differences:

- A global interaction Hamiltonian $H_{\text{int}}$ used as a primary computational resource for a given register of interacting qubits, forming complex analog operations. This approach avoids isolating interactions for specific multi-qubit digital gates.
- The completion of analog operations by parameterized local single-qubit rotations.

Pasqal is a full-stack quantum company and hardware manufacturer headquartered in Paris that produces quantum processing units (QPU) based on neutral-atoms. Significant efforts have been dedicated to creating and implementing analog algorithms and porting algorithms and techniques from digital to analog for execution on the QPUs. For that reason, Pasqal developed a Pulser [2], a framework for designing, simulating, and executing low-level pulse sequences in neutral-atom devices with high levels of customisation. However, programming within the DAQC paradigm using Pulser requires the developer to be equipped with deep knowledge of the physical phenomena that drive the qubit dynamics in the register, especially $H_{\text{int}}$. Although Pasqal produces analog devices, a large fraction of the algorithmic development concerns digital quantum machine learning (QML). Of particular importance are digital algorithms employing differential circuits [3] to solve partial differential equations. That led Pasqal to develop Qadence [4], a Torch-based framework for composing differentiable quantum circuits, with support for both digital and analog computations, bringing it closer to a complete DAQC framework. One of the critical challenges in creating such a DAQC framework is the unification of quantum instructions and primitives. Digital QC employs a gate-based representation, while analog QC relies on Hamiltonians and pulse shapes. Interactions between qubits in digital quantum hardware are possible through controlled operations, while they depend on the explicit qubit arrangement in the register layout for analog devices. From these differences, we have developed Qadence 2, a new quantum software development kit that bridges the gap between the two aforementioned paradigms to bring together a unified quantum computing framework.

Qadence 2 is a symbolic framework primarily designed to explore high-level differentiable algorithms in the analog paradigm while maintaining digital capabilities, by abstracting away from quantum circuits and Hamiltonians using an expression-based model. It encompasses a compilation interface, made possible thanks to the development of a device-agnostic Qadence intermediate representation (IR) — nevertheless designed to facilitate targeting neutral atom platforms. Other recent quantum IR proposals, such as QIR [5], where inherently digital-based and relied on incompatible assumptions with neutral atom devices, *e.g.* being oblivious to the topology of the register. Qadence IR, on the other hand, combines an agnostic model of a neutral atom device (`Model`) that holds critical information for analog components with a procedural sequence of instructions to build a pulse sequence. Furthermore, Qadence IR combines quantum and classical instructions to improve differentiability in simulations usi pure hardware differentiation techniques. Among the main benefits: alleviate the highly demanding task of porting existing digital algorithms and techniques to the analog paradigm and identify and optimise elements on the input side before their concretization. Furthermore, specialised backends can be developed targeting classes of problems without tightening them to device specific information encodings.

In this contribution, we will present the architectural design of Qadence 2, from the frontend expression-based programming language to the execution on either computational (Pulser and PyQ-Torch [ref.]) or physical backends (Fresnel) with an emphasis on the Qadence IR. This is crucial to unifying digital and digital-analog paradigms, breaking down the workflow of bringing algorithms to analog hardware and making it possible to design novel high-level algorithms.

## 2 Qadence Intermediate Representation Overview

The Qadence IR framework holds all relevant data structures necessary to build an abstract representation of the user's quantum code in a chosen language, which will then be translated to a native backend representation. It is inspired by the LLVM [6] and Multi-Level Intermediate Representation (MLIR) [7] philosophies to bring language-independent representation and portability with a reusable and extensible infrastructure to account for heterogeneous backend engines (simulators and hardware). Primitive data structures to define minimal components for constructing a sequence of instructions (such as variables allocator, qubit support, quantum instructions, and SSA form [8] variables), as well as backend configuration parameters, are used during the compilation step from a language to the IR data structure, called the Model. It aggregates an atomic arrangement register, a list of abstract instructions, and QPU and backend flags and options [1] to be translated to backend-specific instructions and directives to code execution.

[1]More can be accessed through [9]

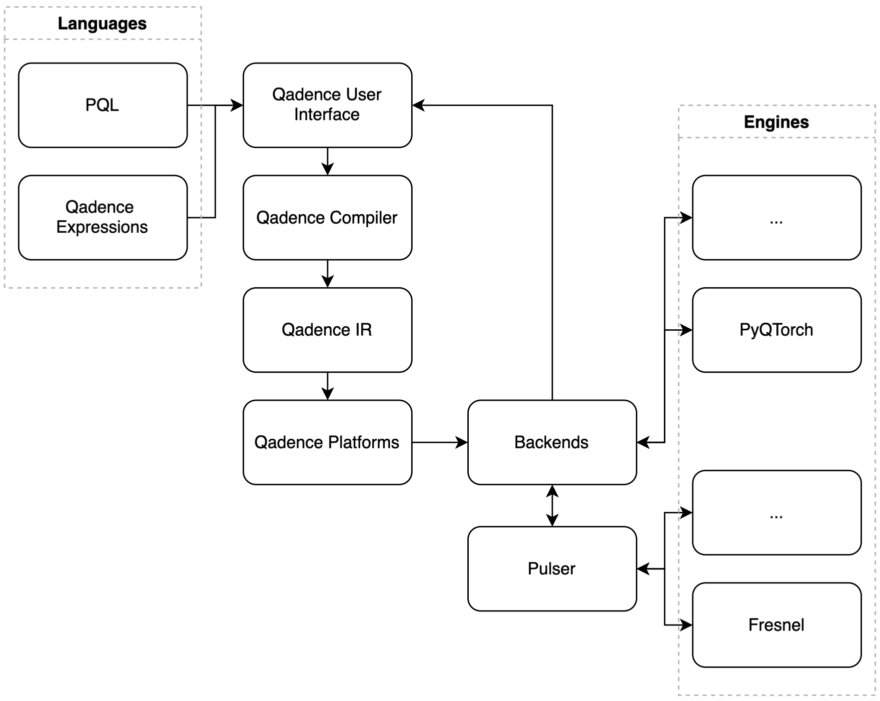

Figure 1: Simplified Qadence 2 diagram. The user represents a quantum code through languages such as PQL and Qadence Expressions, with additional flags and options in the Qadence User Interface. The code and configuration settings are then compiled on Qadence Compiler, converting into the Qadence IR. To reach a simulator or hardware, the IR goes through translations at Qadence Platforms. The result will then be returned from an interface in Platforms through Qadence User Interface.

Each backend is developed to solve a particular task. For instance, PyQTorch's main purpose is to provide differentiability capabilities for digital-analog quantum algorithms to perform Quantum Machine Learning tasks. Pulser's purpose is to design and simulate pulse sequences for programmable arrays of neutral atoms.

For Model data to be adequately translated to each backend's logic and data structure, the Qadence Platforms framework was designed. Its main goal is to use the tools to connect definitions from both ends and effectively apply them. From there, data will be aggregated in an interface instance to provide the necessary methods for code execution, such as sampling, state vectors, and expectation values. The dataflow can be visualized at the diagram in the Fig. 1.

## 3 Conclusion

Qadence IR provides a powerful and flexible framework for implementing quantum programs with neutral-atom devices. While it effectively bridges the gap between analog and digital quantum computing paradigms, it also provides a platform-independent approach while handling critical hardware implementation elements internally. The framework's support for multiple languages and abstract pulse descriptions ensures the versatility needed to perform tasks for Quantum Machine

Learning, going a step further into the quantum algorithm development and implementation in a DAQC era.

## References

[1] Adrian Parra-Rodriguez, Pavel Lougovski, Lucas Lamata, Enrique Solano, and Mikel Sanz. Digital-analog quantum computation. *Physical Review A*, 101(2), February 2020. ISSN 2469-9934. doi: 10.1103/physreva.101.022305. URL http://dx.doi.org/10.1103/PhysRevA.101.022305.

[2] Henrique Silvério, Sebastián Grijalva, Constantin Dalyac, Lucas Leclerc, Peter J. Karalekas, Nathan Shammah, Mourad Beji, Louis-Paul Henry, and Loïc Henriet. Pulser: An open-source package for the design of pulse sequences in programmable neutral-atom arrays. *Quantum*, 6: 629, January 2022. ISSN 2521-327X. doi: 10.22331/q-2022-01-24-629. URL http://dx.doi.org/10.22331/q-2022-01-24-629.

[3] Oleksandr Kyriienko, Annie E. Paine, and Vincent E. Elfving. Solving nonlinear differential equations with differentiable quantum circuits. *Physical Review A*, 103(5), May 2021. ISSN 2469-9934. doi: 10.1103/physreva.103.052416. URL http://dx.doi.org/10.1103/PhysRevA.103.052416.

[4] Dominik Seitz, Niklas Heim, João P. Moutinho, Roland Guichard, Vytautas Abramavicius, Aleksander Wennersteen, Gert-Jan Both, Anton Quelle, Caroline de Groot, Gergana V. Velikova, Vincent E. Elfving, and Mario Dagrada. Qadence: a differentiable interface for digital-analog programs, 2024. URL https://arxiv.org/abs/2401.09915.

[5] Thomas Lubinski, Cassandra Granade, Amos Anderson, Alan Geller, Martin Roetteler, Andrei Petrenko, and Bettina Heim. Advancing hybrid quantum-classical computation with real-time execution, 2022. URL https://arxiv.org/abs/2206.12950.

[6] Chris Lattner. LLVM: An Infrastructure for Multi-Stage Optimization. Master's thesis, Computer Science Dept., University of Illinois at Urbana-Champaign, Urbana, IL, Dec 2002. *See* http://llvm.cs.uiuc.edu.

[7] Chris Lattner, Mehdi Amini, Uday Bondhugula, Albert Cohen, Andy Davis, Jacques Pienaar, River Riddle, Tatiana Shpeisman, Nicolas Vasilache, and Oleksandr Zinenko. Mlir: Scaling compiler infrastructure for domain specific computation. In *2021 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*, pages 2–14, 2021. doi: 10.1109/CGO51591.2021.9370308.

[8] Keith D. Cooper and Linda Torczon. Chapter 5 - intermediate representations. In Keith D. Cooper and Linda Torczon, editors, *Engineering a Compiler (Second Edition)*, pages 221–268. Morgan Kaufmann, Boston, second edition edition, 2012. ISBN 978-0-12-088478-0. doi: https://doi.org/10.1016/B978-0-12-088478-0.00005-0. URL https://www.sciencedirect.com/science/article/pii/B9780120884780000050.

[9] Kaonan Micadei, Eduardo H. Matos Maschio, Dominik Seitz, and Roland Guichard. Qadence 2 ir. URL https://pasqal-io.github.io/qadence2-ir/latest/.