AQiPT: Atomic Quantum Information Processing Toolbox[*]

Manuel Morgado
CESQ & Institut de Science et d'Ingénierie Supramoléculaires (ISIS, UMR7006), University of Strasbourg and CNRS[†]

(Dated: January 28, 2022)

Quantum software solutions tackles different layers of abstraction within the stack of near-term quantum devices. Here

Quantum computing algorithms and circuits represent high-level of abstraction of how quantum devices can solve computational problems using state-of-the-art quantum experiment and devices via electrical pulses that control different stages and parts of the experiments.

Here we present AQiPT as an adaptable open-source software that aim to integrate different layers of a quantum stack solution into a harmonic and standard framework that bridge the high-level tasks as quantum circuits with the Hamiltonian formalism and ultimately with the generation of pulses used in quantum experiments.

CONTENTS

INTRODUCTION

Quantum simulators and quantum computer devices requires a high-level of controllability at all level of processes of the software and hardware. Starting at the highest level, the application of quantum gates at defined points of the quantum circuits are desired. However, the desired quantum gates are a result of a sequence of Hamiltonians that implemented in order to obtain a final quantum state. The quantum state of the quantum register in the quantum device is a result of the evolution operation of the executed Hamiltonians that are implemented in the experiment via a sequence of physical operations, usually represented by a sequence of pulses in the different devices and instruments that compose the experiment. Nowadays, this electrical pulses can be generated with high precision and timing by using commercial [?] and custom digitizers [?] and arbitrary waveform generators based in modern FPGA technology.

Quantum software systems for NISQ devices [] is a field in development where many aspects of quantum engineering most be consider e.g., stack development, canonical-native transpilation, low-level compilation and optimization of pulses. The development of a standard software system for any physical platform must take into account all the complexity that this experiments can comprise, without losing the full-controlability of the sequence details. Besides many open-source software packages has come out for different layers of the quantum stack for the development of quantum computing [? ? ?], simulations of quantum dynamics [?] and the already developed software for waveform generation in FPGA boards [? ?]; there is not a integral and standard framework where this computational tools can be used simultaneously for the realization of experiments as well for theoretical simulations. Although OpenQASM [?] is close approach for this challenge, the lack of experimental awareness of this assembly language is a drawback to overcome.

In this work we introduce AQiPT as a standard framework for the realization of experimental sequences and the hardware-aware compilation of experimental pulses associated to quantum information processing tasks with the possibility of recycling in theoretical simulations. AQiPT is a modular, open-source and Python based framework for hardware-aware designing of experimental

[*] `https://github.com/manuelmorgado/AQiPT`
[†] morgadovargas@unistra.fr

sequences and simulation of quantum information processing circuits.

This work is organized in four main parts: starting with a short introduction of Quantum Software in NISQ devices **??** with an overview of the solutions and challenges, followed by the description of AQiPT **??**: concept, architecture, design, class-map, modules and license. In the third section we focus in the explanation of how AQiPT can be extended and adapted to specific experimental setups and theoretical models. Finally, some remarks and steps to contribute and build a developer community for AQiPT.

I. QUANTUM SOFTWARE AT NISQ

Although recent advances with NISQ devices using superconducting circuits, trapped ions, photons and atomic system are currently at the edge of starting test quantum error correction codes towards faul-tolerance quantum computing, a full-programmable quantum computer is far beyond from current platforms, not only in the hardware but at the software level. The software stack of a universal quantum computer (UQC) should not rest only in the layers of the quantum processor, but also in the additional components that give the sense of universal to the candidates of quantum computers.

A. Quantum software stack

A full quantum stack concern the different layers of programability for the different components of the UQC. The four main different quantum components (i.e., QPU, QRAM, QMemory and QBus) are interconnected in such a way that the quantum computation happens without intervention of the classical world until its final readout. A classical bus has the task of linking to the classical user.

AQiPT is a starting point of a software system, it tries to include and consider all the different layers for a QPU from the high-level quantum circuit design to the programation of high-speed FPGAs. A complete operative system for UQC is composed by the different sub-stacks.

A full quantum stack for a UQC is shown in **??**. Here there is mainly three layers:

- QIP layer (Frontend - API): Dedicated to the development of quantum information processing tasks such as: direct applications in transversal fields e.g., finances [], chemistry [], cryptography [] etc via the Gateway. Additionally, a Mapper it is necessary to map logical qubits into physical qubits and have this physical qubits can be map in the Quantum backend in order to implement the logical quantum circuits into a backend-aware physical qubit circuit.
- Physics layer (Physical interpreter - Kernel): This layer is composed by mainly by three sub-stacks for the Quantum Processor Unit (QPU), the Quantum RAM (QRAM) and the Quantum Memory (QMemory). Here is where the quantum mechanics of platform used for such elements plays a role in the encoding of the quantum register made of qubits and qudits, including a noise model fro them and the configuration of the processor. The
- Low-level layer

B. Current solutions

The existing packages that can be found are well developed for quantum computing and applications such as Qiskit [] developed mostly for superconducting quantum devices, Pennylane [], Inspired [], QuTip [], Openpulse []

C. Current challenges

II. AQIPT

Several software packages has been developed for the different layers of a quantum stack scheme and different platforms for quantum devices, as were mentioned previously. Starting from the most outlaying layers where the goal is bring tools to deal with quantum circuit and algorithms tasks into most deep layers, closer to the quantum backend and its classical control system. However, none of the existing packages has a full integration between the experimental physics of the quantum system and the formalism of quantum computing at the algebra level i.e., how a quantum CNOT gate becomes a series of pulses that realize physical operations which controls different devices in the experiment in order to orchestrates a change of the quantum states of the physical platform. AQiPT y Python library module that is committed to bridge this two aspects by considering the atomic physics of the system, using some of the pre-existing tools in both fields [fig. 1]

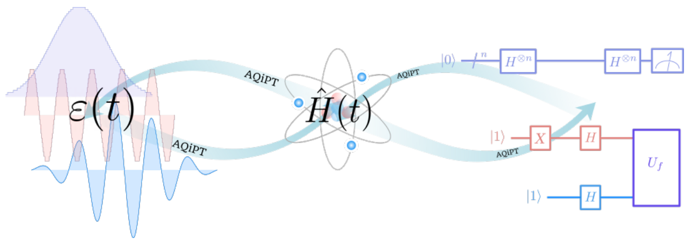

FIG. 1. Philosophy of AQiPT

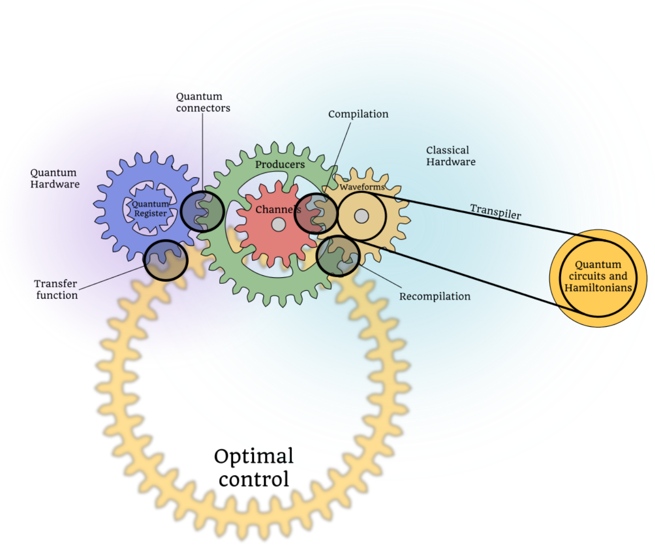

FIG. 2. Concept of AQiPT

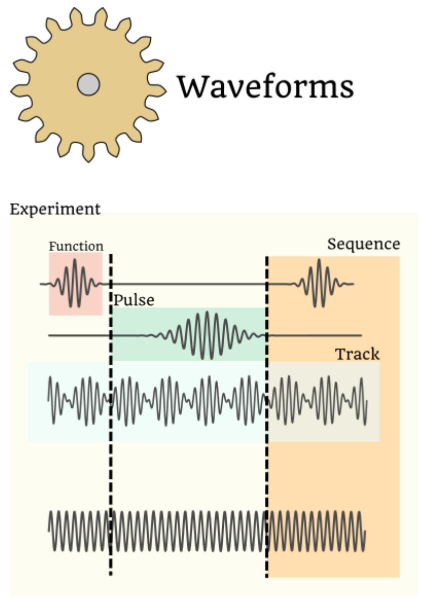

FIG. 3. Waveforms formalism of AQiPT

A. Concept

The way of visualizing 'How AQiPT is build inside?' is shown in the following scheme:

Popular modules already include optimized tools for realizing and simulating quantum algorithms and circuits, as well physical system described via theoretical Hamiltonians with extra options of noise modeling (e.g., Qiskit and QuTiP [ref]). However, open-source hardware-aware modules that ultimately take a quantum information processing task from end-to-end has not been yet develop, besides that there is not a established formalism for experimental sequences.

Therefore, the module presented in the work start from the level of abstraction of experimental sequences that represent the execution of pre-designed quantum circuit (algorithm) using an external module. The formalism for a *Experiment* sequence of *Waveforms* can be either created using two approaches:

- Using blocks of *Sequence*, representing different stages of the experiment e.g., atom loading, evaporation cooling, optical pumping, Rydberg excitation, detection etc.

- Using blocks of *Track*, representing the full set of waveforms along the experimental run in a single device/instrument or "*Producer*" as we explain later (see section of architecture).

In both approaches the *Waveforms* are built using small pieces of float-points called *Functions* that might be given by its analytic form or just an extrapolation of points. These *Functions* are grouped to form a *Pulse*, and a set of pulses of different *Producers* in the same time-domain interval constitute a *Sequence*. However, in the second approach, a sequencial set of *Pulses* for a given *Producer* for the whole time-domain, represent a *Track*. Ultimately, the organized set of Sequences or Tracks represent the *Experiment* waveforms set. 3

The waveforms are send to the Producers, using assigned *Channels*, excuting the ultimately interaction with the *Quantum Register*. Experimental sequences and classical hardware is not perfect, therefore a series of sub-elements has to be include to reach high-performance quantum operations using developed techniques like optimal control (over pulses and Hamiltonians) and distortion cancellation of waveforms using transfer function techniques. Sub-elements for the transpilation of gates and re-compilation of pulses is another contribution that can be included with AQiPT in order to include the optimization of final *Waveforms* into the *Producers* as well as in the development of algorithms that exploids the properties and features of the platform.

B. Architecture

The architecture of the experimental setup and the dataflow can be rather complex depending of the particular implementaiton of the quantum device. The module presented in this work consider a minimal-standard basic elements for it optimal use.

The core layer of the architecture, called "Backend layer" contains what we call "Producers" which holds the main role, they are meant to represent instruments, passive or active devices within the experimental setup, we generally classify a few of them:

[classification]

The Producers are controlled by a more general dedicated device that we call "Main Controller". The Main

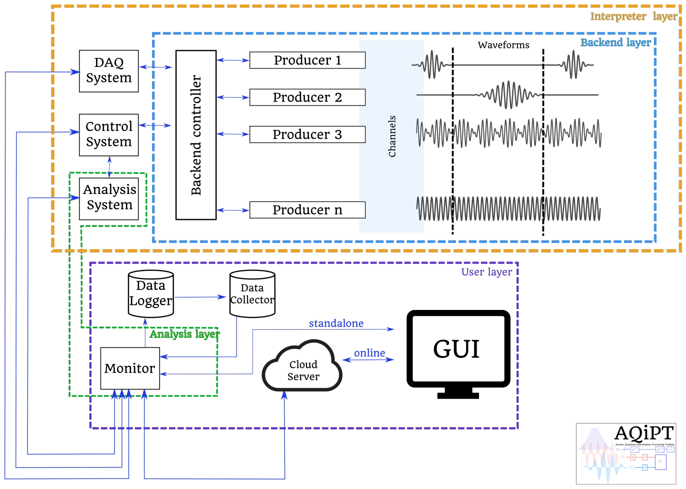

FIG. 4. Architecture of AQiPT

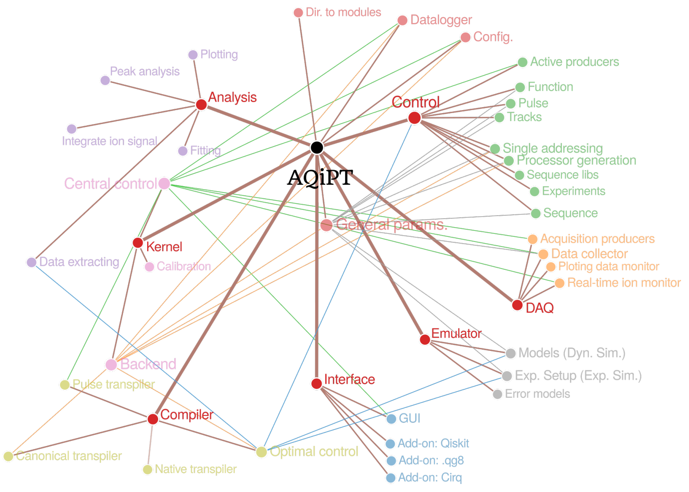

FIG. 5. Map of modules of AQiPT.

Controller can command each Producer using waveforms send via analog or digital channels, as well as acquire information/signal from each of them. The Main Controller at the software level has three types of task: acquisition, control and analysis. These tasks together with the core layer, constitute the "Interpreter layer" which via an abstract layer called "Analysis layer" connects to a disjoint "User layer" that covers the interaction with an actual user via a Monitor and the data pipeline for storage ("Data collector"), logging ("Data logger") and sharing ("Cloud server").

C. Modules

Libraries for quantum computing can be rather complex due to the level of control of the quantum system using classical hardware. Hence, a landscape of the commands and their main goal within the experiments and in the more abstracts processes is rather important for an efficient use and easy-to-access library. The scheme below shows how each of the modules of AQiPT are related to each other.

In the first release of AQiPT there are 3 internal modules for local configuration and data, one for general purpose module and 7 main modules: Analysis, Compiler, Control, DAQ, Emulator, Interface and Kernel.

Analysis

This module contains a group of functions that provide a series of tools to process data in different formats. The input data can be use a feed for functions dedicated to optimal control function class, emulator models, datalogging etc.

Compiler

The task of compilation can happen at different layers of the stack e.g., compilation of algorithms, Hamiltonians and pulses. The compiler of AQiPT is meant to compile the modified pulses originated by the quantum circuit after the transpilation to native gates and adapted to the experiment via cancellation of distorsions, optimal control and feedback from theoretical models for the quantum system and noise. This module has a strong connection to the Control module since it will re-adapt the experimental waveforms for new executions.

Control

Control module contain the core of the experimental awareness, where a set of instructions can be generated using the formalism for experimental sequences, drivers and experimental control software such as Labscript **??** can be use.

DAQ

The measurement of the system is rather crucial, depending of the instruments used in the experiment the type of acquisition might vary. Thus a set of general purposes for image analysis and signal treatment has been located in this module.

Emulator

The emulator module contains 3 main classes that are dedicated to bridging the theory and experiments (see Philosophy section), AtomicModel helps in creating a Hamiltonian model based in the atomic physics of the system whenever using data from the kernel main function. This class works as a building block for atomicQRegister, which is a class to create larger sets of qubits considering all the mechanisms available for the qubits.

Interface

This module contain the user interface elements to connect with the main functions of the modules.

Kernel

The kernel of AQiPT contain all the relevant information of the atomic element that can be used later for the atomic models that will describe the system of the qubit.

D. Licence

E. Blueprint

III. IMPLEMENTATION

A. Include new modules/classes

B. Integration to experimental setup

C. Migration to simulations

D. Pulse compilation

E. Examples

IV. REMARKS

A. How to contribute?

ACKNOWLEDGEMENTS

This work is supported by the 'Investissements d'Avenir' programme through the Excellence Initiative of the University of Strasbourg (IdEx) and the University of Strasbourg Institute for Advanced Study (USIAS). M.M acknowledges the French National Research Agency (ANR) through the Programme d'Investissement d'Avenir under contract ANR17-EURE-0024 and QUSTEC funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 847471.

[1] N. D. Birell and P. C. W. Davies, *Quantum Fields in Curved Space* (Cambridge University Press, 1982).
[2] R. P. Feynman, Phys. Rev. **94**, 262 (1954).
[3] A. Einstein, Yu. Podolsky, and N. Rosen (EPR), Phys. Rev. **47**, 777 (1935).
[4] J. G. P. Berman and J. F. M. Izrailev, Stability of nonlinear modes, Physica D **88**, 445 (1983).
[5] E. B. Davies and L. Parns, Trapped modes in acoustic waveguides, Q. J. Mech. Appl. Math. **51**, 477 (1988).
[6] E. Witten, (2001), hep-th/0106109.
[7] E. Beutler, in *Williams Hematology*, Vol. 2, edited by E. Beutler, M. A. Lichtman, B. W. Coller, and T. S. Kipps (McGraw-Hill, New York, 1994) Chap. 7, pp. 654–662, 5th ed.
[8] E. Beutler, in *Williams Hematology*, Vol. 2, edited by E. Beutler, M. A. Lichtman, B. W. Coller, and T. S. Kipps (McGraw-Hill, New York, 1994) 5th ed., Chap. 7, pp. 654–662.
[9] D. E. Knuth, in *Fundamental Algorithms*, The Art of Computer Programming, Vol. 1 (Addison-Wesley, Reading, Massachusetts, 1973) Section 1.2, pp. 10–119, 2nd ed., a full INBOOK entry.
[10] J. S. Smith and G. W. Johnson, Philos. Trans. R. Soc. London, Ser. B **777**, 1395 (2005).
[11] W. J. Smith, T. J. Johnson, and B. G. Miller, Surface chemistry and preferential crystal orientation on a silicon surface (2010), J. Appl. Phys. (unpublished).
[12] V. K. Smith, K. Johnson, and M. O. Klein, Surface chemistry and preferential crystal orientation on a silicon surface (2010), J. Appl. Phys. (submitted).
[13] U. Ünderwood, N. Ñet, and P. P̄ot, Lower bounds for wishful research results (1988), talk at Fanstord University (A full UNPUBLISHED entry).
[14] M. P. Johnson, K. L. Miller, and K. Smith, personal communication (2007).
[15] J. Smith, ed., *AIP Conf. Proc.*, Vol. 841 (2007).
[16] W. V. Oz and M. Yannakakis, eds., *Proc. Fifteenth Annual*, All ACM Conferences No. 17, ACM (Academic Press, Boston, 1983) a full PROCEEDINGS entry.
[17] Y. Burstyn, Proceedings of the 5th International Molecular Beam Epitaxy Conference, Santa Fe, NM (2004), (unpublished).
[18] B. Quinn, ed., *Proceedings of the 2003 Particle Accelerator Conference, Portland, OR, 12-16 May 2005* (Wiley, New York, 2001) albeit the conference was held in 2005, it was the 2003 conference, and the proceedings were published in 2001; go figure.
[19] A. G. Agarwal, Proceedings of the Fifth Low Temperature Conference, Madison, WI, 1999, Semiconductors **66**, 1238 (2001).
[20] R. Smith, Hummingbirds are our friends, J. Appl. Phys. (these proceedings) (2001), abstract No. DA-01.
[21] J. Smith, Proc. SPIE **124**, 367 (2007), required title is missing.
[22] T. Térrific, *An $O(n \log n / \log\log n)$ Sorting Algorithm*, Wishful Research Result 7 (Fanstord University, Computer Science Department, Fanstord, California, 1988) a full TECHREPORT entry.
[23] J. Nelson, TWI Report 666/1999 (Jan. 1999) required institution missing.

[24] W. K. Fields, ECE Report No. AL944 (2005) required institution missing.
[25] Y. M. Zalkins, e-print arXiv:cond-mat/040426 (2008).
[26] J. Nelson, U.S. Patent No. 5,693,000 (12 Dec. 2005).
[27] J. K. Nelson, M.S. thesis, New York University (1999).
[28] É. Masterly, *Mastering Thesis Writing*, Master's project, Stanford University, English Department (1988), a full MASTERSTHESIS entry.
[29] S. M. Smith, Ph.D. thesis, Massachusetts Institute of Technology (2003).
[30] S. R. Kawa and S.-J. Lin, J. Geophys. Res. **108**, 4201 (2003), DOI:10.1029/2002JD002268.
[31] F. P. Phony-Baloney, *Fighting Fire with Fire: Festooning French Phrases*, PhD dissertation, Fanstord University, Department of French (1988), a full PHDTHESIS entry.
[32] D. E. Knuth, *Seminumerical Algorithms*, 2nd ed., The Art of Computer Programming, Vol. 2 (Addison-Wesley, Reading, Massachusetts, 1981) a full BOOK entry.
[33] J. C. Knvth, The programming of computer art, Vernier Art Center, Stanford, California (1988), a full BOOKLET entry.
[34] R. Ballagh and C. Savage, Bose-einstein condensation: from atomic physics to quantum fluids, proceedings of the 13th physics summer school (World Scientific, Singapore, 2000) cond-mat/0008070.
[35] R. Ballagh and C. Savage, Bose-einstein condensation: from atomic physics to quantum fluids, in *Proceedings of the 13th Physics Summer School*, edited by C. Savage and M. Das (World Scientific, Singapore, 2000) cond-mat/0008070.
[36] W. Opechowski and R. Guccione, Introduction to the theory of normal metals, in *Magnetism*, Vol. IIa, edited by G. T. Rado and H. Suhl (Academic Press, New York, 1965) p. 105.
[37] W. Opechowski and R. Guccione, Introduction to the theory of normal metals, in *Magnetism*, Vol. IIa, edited by G. T. Rado and H. Suhl (Academic Press, New York, 1965) p. 105.
[38] W. Opechowski and R. Guccione, Introduction to the theory of normal metals, in *Magnetism*, Vol. IIa, edited by G. T. Rado and H. Suhl (Academic Press, New York, 1965) p. 105.
[39] J. M. Smith, Molecular dynamics (Academic, New York, 1980).
[40] V. E. Zakharov and A. B. Shabat, Exact theory of two-dimensional self-focusing and one-dimensional self-modulation of waves in nonlinear media, Zh. Eksp. Teor. Fiz. **61**, 118 (1971), [Sov. Phys. JETP **34**, 62 (1972)].
[41] J. M. Smith, in *Molecular Dynamics*, edited by C. Brown (Academic, New York, 1980).
[42] D. D. Lincoll, Semigroups of recurrences, in *High Speed Computer and Algorithm Organization*, Fast Computers No. 23, edited by D. J. Lipcoll, D. H. Lawrie, and A. H. Sameh (Academic Press, New York, 1977) 3rd ed., Part 3, pp. 179–183, a full INCOLLECTION entry.
[43] A. V. Oaho, J. D. Ullman, and M. Yannakakis, On notions of information transfer in VLSI circuits, in *Proc. Fifteenth Annual ACM*, Boston, 1982, All ACM Conferences No. 17, edited by W. V. Oz and M. Yannakakis, ACM (Academic Press, New York, 1983) pp. 133–139, a full INPROCEDINGS entry.
[44] L. Manmaker, *The Definitive Computer Manual*, Chips-R-Us, Silicon Valley, silver ed. (1986), a full MANUAL entry.