Open Hardware in Quantum Technology

Nathan Shammah,[1,*] Anurag Saha Roy,[2] Carmen G. Almudever,[3] Sébastien Bourdeauducq,[4] Anastasiia Butko,[5] Gustavo Cancelo,[6] Susan M. Clark,[7] Johannes Heinsoo,[8] Loïc Henriet,[9] Gang Huang,[10] Christophe Jurczak,[11] Janne Kotilahti,[8] Alessandro Landra,[8] Ryan LaRose,[12] Andrea Mari,[1,13] Kasra Nowrouzi,[5] Caspar Ockeloen-Korppi,[8] Guen Prawiroatmodjo,[14] Irfan Siddiqi,[15,5] and William J. Zeng[1,11]

[1] *Unitary Fund*
[2] *Qruise GmbH, 66113, Saarbruecken, Germany*
[3] *Technical University of Valencia, Spain*
[4] *M-Labs Limited, North Point, Hong Kong*
[5] *Applied Mathematics and Computational Research Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA*
[6] *Fermi National Accelerator Laboratory, Batavia, IL 60510, USA*
[7] *Sandia National Laboratories, Albuquerque, New Mexico 87123, USA*
[8] *IQM Quantum Computers, Espoo 02150, Finland*
[9] *PASQAL, 7 rue Leonard de Vinci, 91300 Massy*
[10] *Accelerator Technology and Applied Physics Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA*
[11] *Quantonation, 75010 Paris, France*
[12] *Center for Quantum Computing, Science, and Engineering, Michigan State University [East Lansing, MI 48824, USA*
[13] *School of Science and Technology, Università di Camerino, 62032 Camerino, Italy*
[14] *Microsoft Quantum, Redmond, Washington 98052, USA*
[15] *Quantum Nanoelectronics Laboratory, Department of Physics, University of California at Berkeley, Berkeley, CA 94720, USA*

Quantum technologies such as communications, computing, and sensing offer vast opportunities for advanced research and development. While an open-source ethos currently exists within some quantum technologies, especially in quantum computer programming, we argue that there are additional advantages in developing open quantum hardware (OQH). Open quantum hardware encompasses open-source software for the control of quantum devices in labs, blueprints and open-source toolkits for chip design and other hardware components, as well as openly-accessible testbeds and facilities that allow cloud-access to a wider scientific community. We provide an overview of current projects in the OQH ecosystem, identify gaps, and make recommendations on how to close them today. More open quantum hardware would accelerate technology transfer to and growth of the quantum industry and increase accessibility in science.

I. INTRODUCTION

The free and open exchange of scientific tools has become more and more important as automation and devices increase their role into scientific fields. The past five years have witnessed an explosion of open-source tools for programming quantum computers. The open nature of these software tools has substantially increased the number of users of quantum computers, and created a new genre of programmer: the "quantum software engineer" [1–6]. This has shaped the development of quantum computing as a whole, leading to the creation of new organizations, job categories, and career paths.

Less attention has been paid to the tools and components actually used to build and control quantum computers – and quantum technologies such as communications and sensing more broadly – as well as efforts making quantum computers more accessible in non-commercial ways. We use the term "open quantum hardware" (OQH) to cover them, and intend for it to explicitly encompass several steps related to the openness associated to hardware in quantum technology, including: (1) open-source software (OSS) for designing quantum processors and other hardware components (used for computation, but also for simulation, sensing, and communication), (2) foundries and facilities for fabricating quantum processors, (3) software for controlling, analyzing and characterizing quantum devices, and (4) software and hardware for making the infrastructure that enables various levels of open-access – from cloud-accessible quantum processors to collaborative testbeds providing non-commercial access and testing, from remotely accessible

* Email: nathan@unitary.fund

This manuscript has been authored by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the U.S. Department of Energy, Office of Science, Office of High Energy Physics.

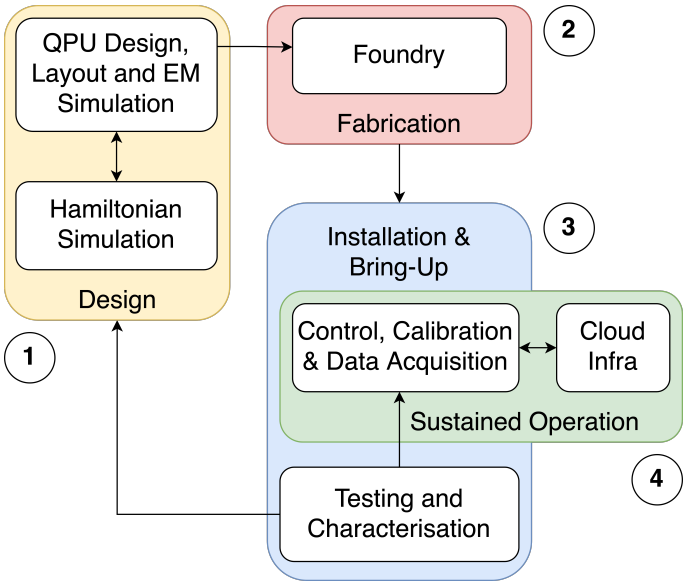

FIG. 1. **Overarching diagram of the Open Quantum Hardware steps and their interconnection**. 1: Design phase, which can involve a loop between simulation of the Hamiltonian and electromagnetic (EM) simulations used to define the QPU design and layout. 2: Fabrication step, e.g., through a foundry. 3: Installation and bring-up, which includes testing and characterization; information collected at this step can be fed back to step 1 to modify designs. 4: Sustained operation, with overlaps (with step 3), and is composed of data acquisition (with control and calibration) and can involve interfacing with infrastructure to provide cloud access to the device or experiment.

research labs to the related cloud infrastructure. These steps covering the life cycle of OQH and their overlapping relations are sketched in Fig. 1.

In this article, we provide an overview of efforts in each of these categories of open quantum hardware (OQH), highlight notable projects in each category, identify gaps, and provide recommendations for closing them. A major theme in this review is that there is much opportunity to promote interoperability, reduce cost, and increase the number of users of quantum technologies.

Enabling the open quantum hardware ecosystem is both the natural complement to the open quantum software ecosystem (which is itself quite robust and sophisticated [2]), as well as the natural extension of efforts to open up classical hardware. This ecosystem also represents a prime opportunity for entities building hardware (of all kinds) to engage in the kind of beneficial, pre-competitive activity which boosts the ecosystem as a whole (and consequently, those builders themselves).

A first generation of projects in open quantum hardware, such as ARTIQ and pyEPR [7, 8], has pioneered a free and open dissemination of tools. We are now witnessing the beginning of a more robust and sophisticated OQH ecosystem [9, 10] which can enable and accelerate scientific progress and discovery on a wider scale. On the one hand, this ecosystem can further extend beyond quantum computing, on which it is mostly focused, to encompass quantum technologies such as quantum sensing and communications. On the other hand, it can further integrate and benefit from the adoption and integration of existing non-quantum open hardware projects and frameworks [11]. What's more, given the substantial investments made around the world [12] – especially in Australia [13], Eurasia [14–19], and North America [20–23]– to support the development of a quantum technologies industry, ensuring those investments yield the best possible fruit is of great importance. One way to do so is through encouraging the development of open communities and ecosystems [24–33] around quantum technology.

To describe the potential of OQH, we can look at the open quantum software ecosystem, which has flourished within the past five years, thanks to seeds planted over ten years ago. As a result, researchers (theorists and experimentalists alike) as well as quantum software engineers worldwide can use open-source software to advance quantum technology and quantum science in many directions, generally building upon an existing stack. An illustrative case is that of the Quantum Toolbox in Python (QuTiP) [34, 35], first released in 2012, which enables the exploration of the effects of noise on a variety of quantum systems interacting with the environment. Building on top of QuTiP, several other tools have emerged, focusing on specific niches such as nontrivial system-environment dynamics [36, 37], notably getting closer and closer to the simulation of QPUs and their diagnostics [38, 39].

When looking at similar efforts in the classical open hardware space, the astounding success of the Arduino project [40] (microcontrollers) as well as the Raspberry Pi [41] (single-board computers) speaks to the power of

Functionality	Examples
Processor Design	DASQA [45, 46], KQCircuits [47], PainterQubits/Devices.jl, pyEPR [8], Qiskit Metal [48], QuCAT [49]
Simulation and diagnostics	KQCircuits [47], Pulser [50], Qiskit Metal [48], QuTiP [34, 35], QuTiP-QIP [39], sc-qubits [38]
Control and data acquisition	ARTIQ [7], Duke-ARTIQ [51], Qua[a] [52], QCoDeS [53], QICK [54], Quantify [55], QubiC [56], qupulse [57], Sinara Open Hardware [58, 59]
Remotely Accessible Labs[b]	Forschungszentrum Jülich through OpenSuperQ [60], Quantum Inspire [61]
Testing (testbeds)	Lawrence Berkeley National Lab's AQT, Sandia National Labs' QSCOUT [62], Sherbrooke's Distriq DevTeQ
Fabrication (foundries)	LPS Qubit Collaboratory, UCSB quantum foundry, QuantWare[c]

[a] partially open-source
[b] excluding commercial providers
[c] private company with support for Qiskit Metal

TABLE I. **OQH examples grouped by functionalities**. The different functionalities in open quantum hardware and examples of representative projects: Processor design, §II A; simulation and diagnostics, §II B; and control and data acquisition, §II B 4. We also list programs related to facilites in OQH: Remotely accessible labs (where here we exclude here the many commercial providers and cloud services, that are discussed in §II D 1); testing (testbeds), §II D 2; and fabrication (foundries), §II D 3.

putting open hardware in the hands of end-users. We can reasonably expect similar benefits within the quantum hardware ecosystem. Compared to only open-source software, an open hardware ecosystem also provides legibility, transparency, and reproducibility of hardware devices, a crucial consideration for supply chains and their management.

Finally, the pre-competitive activity which can take place by opening up quantum hardware extends these benefits by mobilizing institutional momentum from key players who can help the ecosystem adopt a broad ethos of openness, interchangeability, interoperability, and benchmarking, as fostered by bodies like the Quantum Economic Development Consortium (QED-C) in the USA and similar entities in other geographies [42–44]. In doing so, this allows for additional adoption and use of quantum hardware in academic labs and testbeds (a point returned to in Sec. II D 2).

This article is organized as follows: Sec. II gives a high-level overview of the state of the art in OQH, divided among blueprints and software for hardware design (Sec. II A), and software for control and data acquisition (Sec. II B), itself divided into data acquisition (§ II B 1), pulse-level control on hardware (§ II B 2), pulse-level simulation (§ II B 3), and optimal control, calibration and characterization (§ II B 4). In Sec. II C we provide an outlook on the current status and specific need for OQH for quantum error correction. In Sec. II D we review facilities for OQH, such as remotely accessible labs (Sec. II D 1), Testbeds (§ II D 2) and Foundries (§ II D 3). Throughout Sec. II's subsections, we focus on some example projects to bring definiteness to the discussion. In Section III we discuss gaps in OQH and make recommendations to close them. Finally, in Section IV, we give our conclusions.

II. OPEN QUANTUM HARDWARE TODAY

Open quantum hardware encompasses open-source software that is used for designing, analyzing, building, controlling, and programming quantum chips, foundries which build quantum chips, and cloud-accessible labs and testbeds that provide alternative, non-commercially-driven access to them. In this section, we review the state of open quantum hardware along each of these categories. It should be noted that within each of these categories, there can exist multiple stacks – configurations of components which are all inter-operable with one another, but not necessarily with components used in a different stack. Ensuring inter-operability of components across stacks would help ensure a more robust and sophisticated ecosystem. In Table I we summarize the different categories, providing examples of existing open hardware projects. The particular choices of components is again meant to be representative.

A. Blueprints and software for hardware design

Blueprints for hardware design are classic examples of "open hardware" [46, 63–71]. In modern device design, software tools have been developed to tackle various aspects of this task, known as computer-aided design (CAD). In quantum computing research, while quantum device design is often published in peer-reviewed papers and micrographs are included in the supporting materials and figures, generally, CAD drawings are not shared in the open. Notable exceptions include pyEPR [8], Qiskit Metal [48], and KQCircuits [47], three projects enabling various capabilities for chip design of superconducting-circuit (SC) qubits.

Superconducting circuits incorporating non-linear devices, such as Josephson junctions and nanowires, are among the leading platforms for emerging quantum technologies. Using pyEPR, one can design and optimize SC circuits and control dissipative Hamiltonain parameters in a systematic, simple and robust approach. This reduces the number of required ab-initio simulations. pyEPR has been used on a variety of circuit quantum electrodynamics (cQED) devices and architectures, from 2D to 3D, including "2.5D" (flip-chip), demonstrating 1% to 10% agreement for non-linear coupling and modal Hamiltonian parameters over five-orders of magnitude and across a dozen samples.

Finite Element Methods (FEM) simulations, as those shown in Figure 2, can be obtained with PyEPR using the energy participation ratio approach (EPR). EPR unifies the design of dissipation and Hamiltonians around a single concept — the energy participation, a number between zero and one — in a single-step electromagnetic simulation. After the FEM simulations have validated the qubit properties (and possibly its coupling to a cavity), one is ready to fabricate the SC QPU. To this end, a photomask is generated to place the SC qubits on the substrate of a processor. You can find a typical CAD drawing of a superconducting flip chip QPU and an example mask generated by KQCircuits [47] in Figure 3 and more information about KQCircuits in the frame below (*Example 1*).

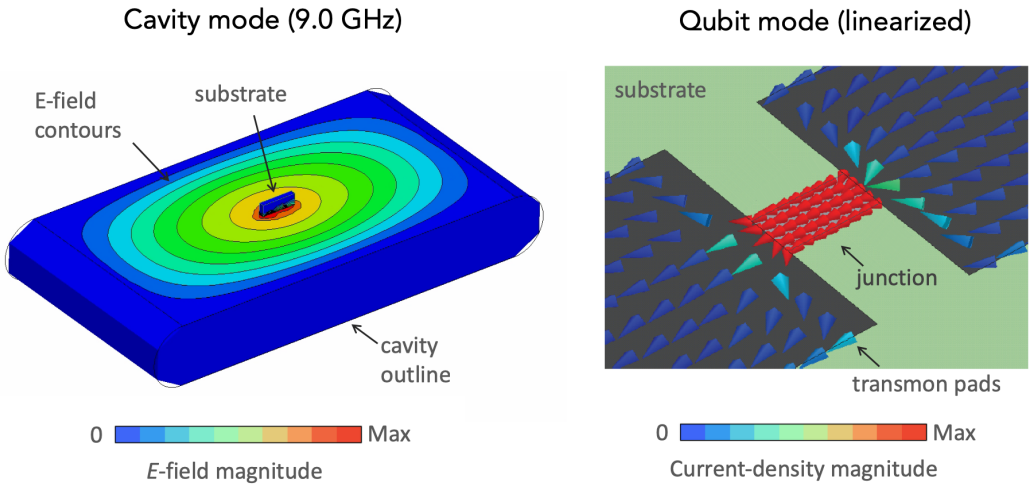

FIG. 2. **Finite Element Methods (FEM) chip simulations** Design and FEM simulations of a cavity (left) and qubit (right) modes, using PyEPR [8]. The contour plot of cavity modes displays the intensity of the electric field at 9.0 GHz. The Current-density magnitude of a transmon qubit, showing transmon pads connected by a Josephson junction [adapted from Ref. [8]].

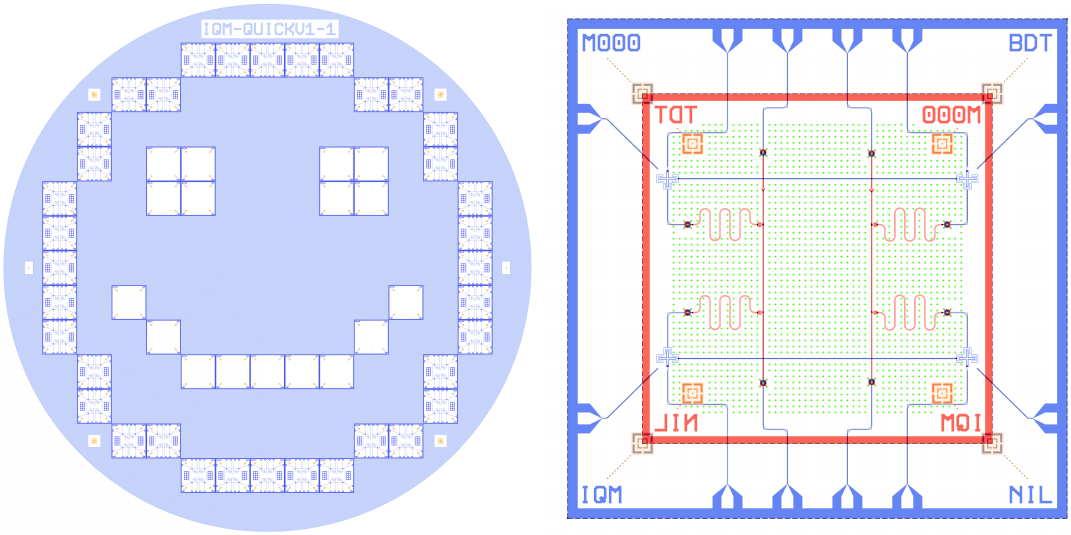

FIG. 3. **Photomask layout and chip design.** An open-source photomask (left) provided in KQCircuits as a demo example. An individual SC quantum processor layout (right) based on flip chip technology, where blue is one substrate, red is the other substrate and in green denotes the bump bonds. The chip contains four transmons capacitively coupled and two buses for multiplexed readout. The code for the mask can be accessed here here and for the chip here.

Example 1. HARDWARE DESIGN.

KQCircuits: KLayout Python library for integrated quantum circuit design.

KQCircuits [47] is an open-source Python library created by IQM, a full-stack quantum computing startup, for designing superconducting quantum circuits. It automates the layout and simulation part of the design process, outputting chip layouts and photomask files in OASIS or GDSII formats, standard formats for the specification of data structures of photomasks, and integrated circuit layout. The layout files are sent to a mask manufacturer to produce a physical mask which is then used for the fabrication process. As a part of the mask export process, KQCircuits also produces other files such as netlists, to help with design verification. While KQCircuits itself cannot be used to perform simulations, it can be used to export automated simulation scripts and files for popular electromagnetic field simulation toolkits, such as Ansys HFSS/Q3D [72], pyEPR [8] (Energy Participation Ratio simulation), Sonnet [73], and Elmer [74]. These simulations use the parameterized geometry as the physical mask layouts produced by KQCircuits. A comparison between KQCircuits and Qiskit Metal is provided in Table II.

KQCircuits generates multi-layer 2-dimensional-geometry representing common structures in quantum processing units (QPU). An important part of KQCircuits are the definitions of parameterized geometrical objects, or "elements", and a framework for easily defining new ones. Generating the designs using code makes it easy to quickly create different variations of designs and helps to avoid costly human errors. The combination of many elements into a full QPU design is made easier by features such as named reference points used for automatic positioning of elements relative to each other and Graphical User Interface (GUI) editing. Furthermore, KQCircuits includes a library of premade chips, many of which have been manufactured and used for testing at IQM.

KQCircuits works on top of KLayout [11], an open-source layout design software that is mainly used for classical electronics. Advantageously, existing KLayout functionalities can be used for quantum hardware other than classical, while KQCircuits only needs to add features specific to QPU design. This connection with KLayout can also help to bring together the wider electronics open-source hardware community and open quantum hardware community.

In terms of remaining challenges for chip design tools, we note that computational electromagnetic simulation of quantum circuits is still heavily reliant on the use of proprietary and expensive tools such as Ansys HFSS or Sonnet or CST Studio (full-wave and capacitance simulations in particular). Open-source tools such openEMS or MEEP or Scuff-EM which are used elsewhere in the field of microwave simulations find very little adoption and application in quantum device design. Notable open-source exceptions are Elmer [74] and Palace [75]. Elmer is an open-source parallel multi-physics Finite Element Methods (FEM) software used for quantum device simulation on

Software	Purpose	Simulations	Circuit analysis	Input/output (I/O)
Qiskit Metal	Full-stack quantum processor design.	Ansys HFSS/Q3D, Ansys, GDS, etc. by plugins. Work in progress on open-source EM renders.	EPR, impedance, quasi-lumped LOM, lumped.	Python based. Connects to Ansys, GDS, etc. by plugins. Work in progress on open-source EM renders.
KQCircuits	Design parametrization and automation for quantum processors. Extensive components, chips and masks library.	Ansys HFSS/Q3D, Sonnet, Elmer (open-source) supported.	EPR, cross sectional EM properties, London equations and netlist export.	Import/export GDSII and OASIS format. Netlist export. Automated simulation scripts and files export.

TABLE II. **Open hardware design of SC qubit processors**. Qiskit Metal, supported by IBM, and KQCircuits, developed at IQM, are two software packages for open hardware design for SC qubit processors. We highlight their purpose, integration with FEM simulation software, circuit analysis characteristics and input and output file formats. EPR: Energy Participation Ratio, GDS: Graphic Design System, HFSS: High-Frequency Structure Simulator, LOM: Lumped Oscillator Model.

desktop and High Performance Computing (HPC), e.g., 3D layouts capacitance matrices, cross section of layouts, London equations, which in the framework of OpenSuperQPlus (European Open-Access Quantum Computer Project [60]) is developed with the partnership between CSC and IQM. Elmer along with Gmsh [76] has also been integrated as a backend for mesh generation and FEM simulation in Qiskit Metal. Palace (Parallel, Large-scale Computational Electromagnetics) [75], is an open-source parallel FEM software capable of full-wave electromagnetics simulations developed at AWS, with out-of-the-box support for use of large scale cloud HPC resources.

B. Control and data acquisition

In this section, we discuss the features and components involved in the execution and readout of quantum experiments. This field is not easily defined, as the operation of a QPU or quantum technology experiment is informed already at the highest level with the definition of abstract instructions, e.g., for quantum programming, gates in the form of a quantum circuit in a high-level software development kit (SDK). To narrow the field, we focus here on control that gets “closer to the metal", e.g., lower than gate-level quantum circuit compilation in quantum computing. This field is already populated by a number of projects, spanning from pulse-level simulation of devices for diagnostics, characterization and calibration, to pulse-level control and data acquisition during experiments runs [7, 34, 38, 39, 53, 54, 77–93]. We highlight that core to these tasks is the exchange of data through application programming interfaces (APIs).

1. Data acquisition

Quantum hardware for control and readout of quantum systems includes a user interface to operate the instrument, for instance, through a touch-based front panel or remotely accessible GUI. However, to implement more complex tasks such as characterization, tuning, calibration and general operation of QPU elements that involve coordination and timing between several different pieces of programmable hardware, it is essential to add a software layer that runs on a control computer and communicates with hardware directly through a programming interface. This software layer orchestrates the flow of control and data acquisition commands and is responsible for the overall operation of the QPU via the quantum hardware.

On the software side, there exist quite a few alternatives for QPU control and data acquisition, starting with QCoDeS [53], which is a Python framework developed by Microsoft Quantum. Another example of Control and Data acquisition software is ARTIQ [7], which is also used to control hardware, as discussed in Sec. II B 4. Control and data acquisition software encompasses various elements and instruments: the quantum device, the firmware that interconnects with the instrument drivers, parameters that are controlled and measured to generate datasets, which are stored, analyzed and explored via data visualization tools, as depicted in Fig. 4.

To achieve frictionless control of a QPU, the software typically abstracts away the hardware elements using drivers. For example in the QCoDeS library [53], an instrument driver is a Python class that implements methods to operate

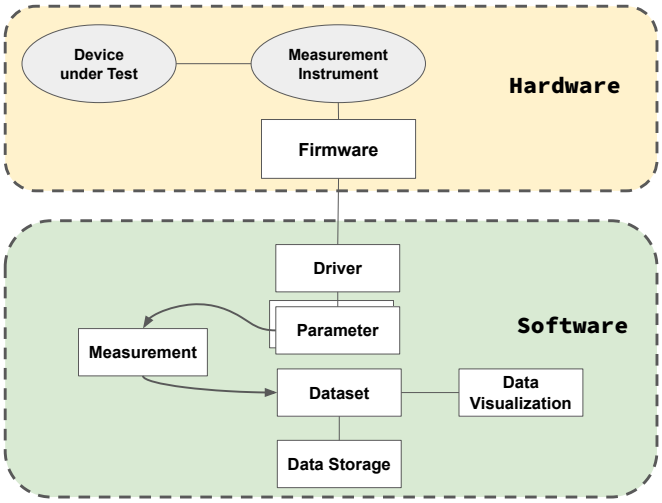

FIG. 4. **Control and data acquisition.** The diagram shows some of the most important elements in the software and hardware stack involved in a quantum experiment. The hardware part of the stack includes the device under test or operation, the measurement instrument and the firmware. A driver provides interaction between the software and hardware. Drivers implement parameters that can be controlled (set) or read (get) over several measurements. The result of each measurement is stored into a dataset, that is read for data analysis, visualization and storage.

the hardware using commands that are sent to the device via the programming interface. Namely, a quantum hardware instrument, such as an arbitrary waveform generator, can be used to generate a radio frequency pulse for a rotation of a qubit, e.g., an X gate. In order to program this pulse, the driver implements parameters such as frequency, amplitude, phase and pulse time. The driver will then use these values to construct the right programming commands to send to the instrument to set up the pulse. The QCoDeS documentation [53] contains driver examples to control over 50 instruments.

The data acquisition sofware uses instrument drivers to form an abstraction layer to the several pieces of hardware that control a QPU. Parameters can also be used in higher levels of abstraction on top of instrument parameters to represent QPU device elements that are controlled and read via the hardware layer, such as a qubit readout circuit. The parameters are then used in automated measurement routines with the goal to characterize, calibrate or program a QPU or its different elements. This requires saving, analyzing and visualizing the data that is recorded by these instruments in a dataset. Most data acquisition software frameworks therefore also include data storage abstractions and data visualization tools.

In order to control the QPU elements using quantum programming instructions, the control signals need to be calibrated for optimal gate and read-out fidelity. As shown in Fig. 1, calibration values are obtained via device testing and calibration measurements using the data acquisition software, analysis routines and data storage in the Cloud. More details can be found in section II B 4 on optimal control, calibration and characterization for high-fidelity quantum operations.

A way to approach the software-hardware interface is through an Instruction Set Architecture (ISA) [94] as it is done in classical computing. The ISA acts as a contract between hardware and software by explicitly describing the set of available operations, machine instruction format, supported data types, registers, and memory addressing. A well-designed ISA can present a very compact and efficient method to access the specialized features of a computing device. Graphical Processor Units (GPUs) [95] and their success in becoming an integral part of many modern devices is an example of instruction set design and standardization leading to widespread specialized architecture adoption.

There are a few quantum ISA implementations [96–98], with some that explore this control and data acquisition approach, such as QUASAR [99] developed at Advanced Quantum Testbed (AQT), Lawrence Berkeley National Laboratory (LBNL). QUASAR is based on the RISC-V ISA [100] - an open-source architecture that revolutionized the classical computing field and continues expanding toward emerging technology applications. QUASAR has been demonstrated in conjunction with QubiC [83] (QubiC is discussed separately in section II B 2, below), executing experiments, including mid-circuit measurement and feed-forward, on superconducting quantum processors (QPU) at AQT, LBNL.

Example 2. CONTROL AND DATA ACQUISITION

QUASAR: A Quantum Instruction Set Architecture

The QUASAR development started in 2017 and went through several iterations. It is a project defining a quantum Instruction Set Architecture (ISA) to interface software with hardware. The first version was a tightly-coupled extension that required significant changes to the processor micro-architecture. Later, the implementation moved towards a more decoupled approach. The current implementation is a RISC-V Rocket Core [101] extended with the QUASAR co-processor (note that RISC-V is a very popular open standard ISA in classical open hardware. RISC-V is based on the established reduced instruction set computer (RISC) principles, and the Rocket Core is a processor that implements RISC-V.) In contrast to classical instruction sets, quantum ISAs operate on different types of computations. Traditional general-purpose operations are supplemented with a set of basic quantum gates applied on direct-addressed qubits and/or qubit registers. The RISC-V core executes the main program; when a quantum instruction occurs at the fetch pipeline stage, it forwards it via the RoCC interface [102] to be decoded and executed by the QUASAR co-processor. Such an architectural solution allows the main core to continue performing computations while the quantum backend is generating the control pulses. Moreover, the RISC-V core can perform complex computations, such as floating-point arithmetic for phase estimation, and make conditional branching during algorithm execution based on the qubit measurement data. That makes the QUASAR implementation flexible enough to accommodate complex hybrid experiments for classical-quantum computations. The RISC-V Rocket core runs a Linux kernel that facilitates remote communication with the user host machine and provides additional functionalities for the quantum software stack.

2. *Pulse-level control with hardware integration*

In recent years, several pulse-level control software packages have been developed with the idea of providing end-users with a tool to program all the relevant device-specific physical parameters of the system [7, 39, 54, 77, 79, 103]. This approach allows for a finer level of control over pulses during the application of gates, and also makes it possible to directly use the Hamiltonian of the system as a resource for computation. Within the gate-level framework of quantum circuits, pulse-level control and simulation enables a greater level of flexibility and the ability to implement optimal control schemes. Within the analog quantum simulation framework, pulse-level control notably allows practitioners to take advantage of the mimetic capabilities of the hardware. Exposing the hardware controls at such a low level is intended to help quantum developers design software procedures while having the specific characteristics of the hardware in mind [104].

Control hardware, e.g., field programmable gate arrays (FPGAs), arbitrary waveform generators (AWGs), sequencers, have historically been closed-source, either provided by industrial manufacturers or by in-house developed systems. The ARTIQ/Sinara project is a notable exception, focused on the fast control of ion-based quantum processors through FPGAs, as detailed in the box below (*Example 3*).

Example 3. CONTROL AND DATA ACQUISITION.

ARTIQ: Advanced Real-Time Infrastructure for Quantum physics.

The ARTIQ experiment control and data acquisition system was initiated in 2013 at the NIST Ion Storage Group, in partnership with M-Labs Ltd, to address the deficiencies observed with control systems developed in-house by physicists or based on existing commercial solutions. A key feature of the ARTIQ system is a high-level Python-based programming language that helps describe complex experiments, which is compiled and executed on dedicated FPGA hardware with nanosecond timing resolution and sub-microsecond latency. Using the abstractions provided by Python, ARTIQ has the capability to handle the entire control stack from quantum circuits to pulse-level programming. ARTIQ also supports connecting several FPGAs together and synchronizing their clocks in the sub-nanosecond regime, which greatly expands the input/output (I/O) scalability of the system.

Initial versions of the ARTIQ system ran on physicist-designed hardware based on FPGA development kits with custom I/O expansion boards. In order to improve the quality, availability and reproducibility of the hardware, the Sinara project [59] was started in collaboration with Warsaw University of Technology. The Sinara project developed a modular system with carrier FPGA cards (Kasli, Kasli-SoC) controlling various so-called Eurocard Extension Modules (EEMs) catering to the needs of each experiment – such as digital I/O, Analog to Digital Converter (ADC), Digital to Analog Converter (DAC), Direct Digital Synthesis (DDS), Phase-Locked Loop (PLL) synthesizer, AWG. There is ongoing work by Duke University on firmware [51] and by Warsaw University of Technology on hardware [58] to port ARTIQ to the RF-SoC platform, with similar capabilities to QICK.

Combined with the ARTIQ software, the Sinara hardware has encountered substantial success, with almost a thousand quantum physics experiments relying on ARTIQ/Sinara systems.

ARTIQ, including its gateware, the firmware, and the ARTIQ tools and libraries are licensed as LGPLv3+. The Sinara hardware designs are licensed under CERN OHL. This ensures that a user of ARTIQ or Sinara hardware designs obtains broad rights to use, redistribute, study, and modify them.

More recently, other FPGA-based projects have emerged beyond ARTIQ, for SC qubits control, such as QubiC [56] and QICK [54], two projects based on software, firmware and hardware composed of radio-frequency system-on-chip (RF-SoC) boards. Open-source FPGA-based control efforts for SC qubits started in 2018 with QubiC, developed at the Advanced Quantum Testbed (AQT) based at LBNL (more information on this facility is given in Sec. II D 2, discussing Testbeds). The first version of QubiC was implemented on the XilinX VC707 platform [83, 105] informed by actual quantum information science experiments at AQT. To reduce cost, complexity, and space requirements, customized hardware components were designed and fabricated, such as in-phase and quadrature (I/Q) mixing modules integrating filters, amplifiers, and bias_Ts on printed circuit boards (PCBs) with Electromagnetic Interference (EMI) shielding, the design of which was published and open-sourced to benefit the community [106]. Additionally, automated single- and two-qubit gate calibration protocols were developed using QubiC [56].

To take advantage of the growing capabilities of recent RF-SoC platforms, QubiC was then ported to the XilinX ZCU216 platform [107], capable of direct generation of control pulses at higher frequencies (up to 10GHz, in the second Nyquist zone). Additionally, a novel Distributed Processor design was created and implemented to enable distributed decision-making and branching, such as mid-circuit measurement and feed-forward [108]. The different iterations of QubiC implementations are shown in Figure 5. To form a prototype full-stack quantum control system with increased potential for future experiments, QubiC and QUASAR (described above in II B 1) have been integrated and applied to quantum computing experiments at AQT, LBNL [109].

Boards from the QICK project have reached over 40 labs in the USA and abroad after a little more than two years. Using QICK helps reduce equipment costs and increases the performance of quantum information science experiments with SC qubits [110]. All of QICK implementations have been on Zynq boards, with the firmware provided by PYNQ (Python for Zynq), compatible with multiple generations of RF-SoC FPGAs. An image of a QICK board is shown in Figure 6. The usage of QICK has already expanded from SC qubits to atomic, molecular, and optical physics qubits and spin qubits (Nitrogen-Vacancy-center qubits), on the hardware side. On the application side, QICK is used not only for quantum computing but also for quantum sensing experiments, e.g., for dark matter candidates detection [113], as well as for RF-SoC control in particle physics detection beyond quantum technology.

3. *Pulse-level simulation*

Simulation is important to design an experiment before running it on hardware and then to validate and interpret the results after data collection. Pulse-level simulation can be employed for quantum optics experiments, quan-

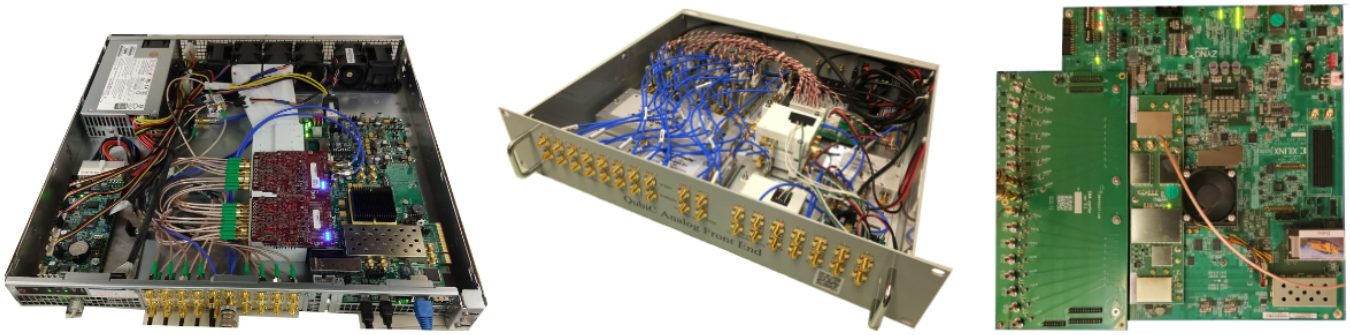

FIG. 5. **QubiC, through the years:** QubiC 1.0, implemented initially on XilinX VC707, in its chassis with auxiliary components (left), the Analog Front End chassis for QubiC 1.0 (middle), and QubiC 2.0 implemented on XilinX ZCU216 with a custom SMA fanout board. A customized Surface Mount Assembly (SMA) fan-out board has been designed and recently fabricated to fully utilize all the channels of the ZCU216 board. Separately, a low-cost DAC extension was developed for the VC707 platform, to meet the varying frequency needs of different superconducting QPU architectures. The QubiC team has recently demonstrated heterogeneous synchronization of two sets of VC707 boards with low-cost DACs and two ZCU216 platforms [107].

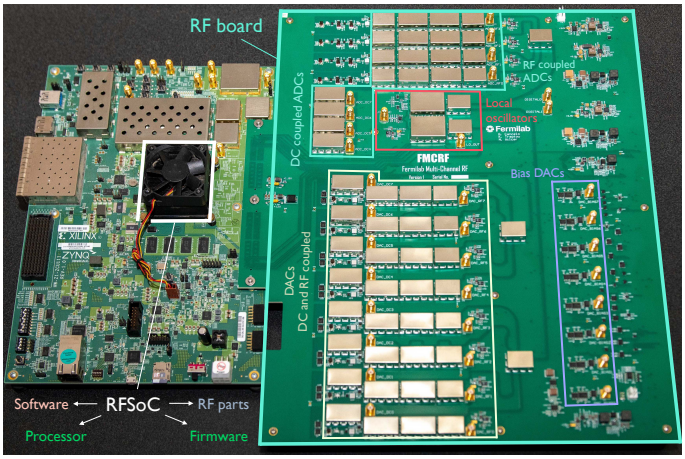

FIG. 6. **The Quantum Instrumentation Control Kit (QICK).** The QICK consists of two pieces of hardware: a commercial RFSoC evaluation board (left), which connects to the QICK RF&DC custom board (right) which can be used for additional signal amplification and filtering. QICK supports the AMD-Xilinx ZCU111 [111] (shown in the picture with its companion custom board) and the AMD-Xilinx ZCU216 [112] (not shown) with a new companion board under fabrication and testing.

tum simulation, and both for gate-level (digital) quantum computing and analog quantum computing. In quantum computing, although gate-level instructions are the most common formalism on the user-side to write quantum algorithms, there can be a compiler that transforms discrete operations into pulses. Examples of pulse-level simulators include Pulser [79] and Bloqade.jl, two toolboxes for neutral-atom QPUs respectively developed in Python and Julia. qutip-qip is a simulation toolkit that provides the freedom to design any QPU and simulate pulse level execution, integrating it QuTiP's noisy time evolution [39]. We summarize information on Pulser in *Example 4*.

Example 4. PULSE LEVEL SIMULATION. **Pulser: Library for pulse-level/analog control of neutral atom devices.**
The Pulser framework is an open-source software library for designing pulse sequences for neutral-atom QPUs [50]. In such devices, individual atoms are placed in arrays of microscopic traps [114–116] and their quantum state is manipulated through the application of laser or microwave pulses. Using Pulser, developers can control all the relevant physical parameters of the qubit register and the driving channels.
The central object in Pulser is called a sequence, which is linked to a device. The device contains information about the hardware constraints, such as the maximal number of qubits available, the minimal pairwise distance between qubits which can be reached or the number of lasers and the maximal amount of laser power available for each of them. These constraints are enforced upon the register, where the neutral-atom array is defined, and upon the pulses that are added to the sequence. Each pulse is defined by its phase, amplitude and detuning waveforms, and pulses are sequentially added to the channels that are available on the device. The resulting program can subsequently be sent to QPUs and executed on the hardware after a device-specific calibration-aware compilation step. In addition to providing an interface to hardware, Pulser includes a built-in emulator relying on QuTiP [34] that faithfully reproduces the hardware behavior. To this extent Pulser acts also as a simulator for QPU design.

4. Optimal control, calibration and characterization

High fidelity operations require high fidelity optimized controls. The process of generating optimal pulses traditionally involves a model-based open-loop optimal control step (in simulation) such as GRAPE, Krotov, GOAT algorithms [117–119] followed by a model-free closed-loop calibration step (in hardware). The former relies on building a sophisticated model of the system and the optimality of the generated pulses is constrained by the model's capability to faithfully reproduce the imperfections of open quantum systems. The latter usually involves the use of a gradient-free optimization algorithm, e.g., Nelder-Mead or CMA-ES to optimize the parameters of an open-loop optimal pulse on the quantum hardware benchmarked with a fidelity measure such as ORBIT [120]. Open-source libraries such as JuliaQuantumControl, QuOCS and C3-Toolset [80, 81] aim to provide a broad range of the optimal control functionalities previously discussed, with easy to use interfaces in Python or Julia. These tools also incorporate automatic differentiation (AD) techniques which have made it possible to obtain gradients of arbitrary order for free, even with the most complex numerical simulations. Such gradients are essential for the optimization of pulse parameters in open-loop control. AD capabilities are provided by the use of standard Machine Learning frameworks such as TensorFlow or JAX [121, 122].

In order to improve the fidelity of the pulse obtained through open-loop control, both the model parameters and the model itself need refinement. One approach for refining model parameters is the technique of data-driven system identification also known as characterisation. This is achieved through learning model parameters from data collected during experiments performed on the hardware, e.g., the calibration step previously outlined. Refining the model translates to building a complex digital twin that models not only the quantum dynamics but also all of accessory classical electronics and their non-ideal behavior. These two techniques — Model Learning and Quantum-Classical Digital Twin — are tightly integrated as a unified solution in the C3-Toolset package [80], as discussed below (*Example 5*).

Example 5. OPTIMAL CONTROL, CALIBRATION AND CHARACTERIZATION.
C3: An integrated tool-set for Control, Calibration and Characterization.
The C3-Toolset package provides an API for directly interfacing with hardware to use the pulses generated by optimal control and further optimize them using closed-loop calibration algorithms. Users have access to a high-level Qiskit interface that allows them to run gate or pulse level quantum circuits on the full-physics noisy high-fidelity differentiable simulator. In recent years, a variety of machine learning for quantum control techniques have been closely integrated in the quantum device bring up process. Besides the previously mentioned process of learning model parameters, reinforcement learning (RL) has been particularly useful in a variety of applications. The open-source library rbqoc [123] details a technique for noise-robust control using RL while C3-Toolset uses RL for both learned optimizers and Bayesian experiment design.
Calibration is generally needed for most platforms. Depending on the architecture, this involves different control signals. A difference between academic research labs and industry is the tendency to automate and standardize calibrations tasks. Tests are run periodically to validate the status of the system before running an experiment and during an experiment. On a schematic level, these can be seen as continuous integration (CI) tests, involving hardware control but similar to open-source testing for software projects. If the job does not pass a test, automatic re-calibration is prompted.

C. Designs for quantum error correction

Many applications of quantum computing, networking, and sensing technologies will require a significant reduction in error rates. In the long term, developing quantum hardware with built in error correction can address this need through fault-tolerant quantum computing. In this section, we review some of the open-source software available today for studying and simulating quantum error correcting codes. This software is becoming increasingly important as quantum error correction moves from theory into practice.

Quantum error correcting codes are often based on stabilizer subsets of quantum computation and so can benefit from specialized simulators. A leading open-source package for stabilizer simulation today is Stim [124] which improves on existing stabilizer simulators available in Qiskit, Cirq, or other packages.

Other libraries sit on top of simulators like stim and allow users to explore quantum error correcting code designs. These packages help inform longer term planning for quantum processor architecture. Examples include plaquette [125] and stac [126]. Both of these are Python libraries that have built in examples of quantum error correcting codes and allow you to generate logical circuits to study performance. There are also tools that act as compilers, translating logical circuits into physical circuits that can be run directly on capable hardware, e.g. the suite of tools developed at latticesurgery.com for compiling to the surface code.

Finally, there are open implementations of the decoding algorithms used to decide how to best correct error detected by different quantum codes. PyMatching [127, 128] is an example open-source decoder. This software can help inform the design of processors specifically adapted to run quantum error correction. For example, stim was used to simulate novel superconducting qubit designs to support surface code parity measurements in [129].

There remain many important opportunities to improve quantum error correction. For example, fast classical control of QPUs is essential for feedback loops in quantum error correction implementation. Tailored tools could reduce the existing gap between hardware operation and software instructions. More generally, increasing the ecosystem of open-source software available in this field will both support new breakthrough ideas as well as accelerate the transition from theory to practice in building fault-tolerant quantum computers.

D. Open quantum hardware facilities

In these section we review the state of the art with respect to facilities that can enable a robust OQH ecosystem. We find three qualitatively different categories: open-access to QPUs and remotely accessible research labs, collaborative testbeds, and facilities for fabrication such as foundries.

1. *Remotely accessible labs and cloud-connected labs*

The idea of open remote labs, i.e., laboratories that can be remotely accessed by users to perform real experiments is not new in the context of scientific research and education [130, 131]. For example, the educational tool HYPATIA [132] can be used to perform particle-physics experiments with real data produced by the ATLAS experiment at CERN. In

a similar way, quantum computers can be used as quantum remote labs, i.e., advanced laboratories in which quantum experiments can be performed for purely scientific purposes without any computational motivation [62, 133–139]. For example in Ref. [140] a quantum computer was used for testing quantum fluctuation theorems, while in Ref. [141] a quantum computer was used to prepare a many-body quantum system in a time-crystalline phase.

Superconducting-circuit quantum computers originally became available online from IBM Quantum with the IBM Quantum Experience, Rigetti Computing with the Rigetti Quantum Cloud Services, and other providers. These have had an impact on the way research is done in quantum computing and quantum optics as well as enabling access for students for outreach activities, due to the fact that experiments can be done much more easily (even by theorists), from the cloud. We are now in a second phase with more providers putting their devices online, encompassing more technologies – ion-based (IonQ, Quantinuum), atom-based (e.g., Infleqtion, Pasqal, QuEra), photonics-based (e.g., Quandela, Xanadu), quantum-dot based, etc.

Turning the attention to institutional frameworks, the European OpenSuperQ project (and the follow on OpenSuperQPlus project) [60] is similarly aimed at developing public-owned full-stack open-access superconducting systems. Multiple remotely accessible QPUs are online or currently being set up as part of this project at the Delft University of Technology, the Wallenberg Centre for Quantum Technology (Chalmers), the Walther-Meissner-Institut (Munich), and at the Forschungszentrum Juelich (from phase 1). A hybrid form of cloud access is also emerging, in which cloud providers give access to different quantum processor providers. Further sustaining open access to QPU providers can be important for scientific discovery.

Over the years, the level of control over device properties in compilation, optimization, qubit mapping, etc., has increased, spilling notably into remote pulse level control, both for digital (e.g., Qiskit pulse [77]) and analog quantum computing (e.g., Pasqal's Pulse studio [115]). The deeper the access, the more advanced the control researchers have over hardware and the greater the opportunity for integration of open-hardware features with cloud-access. An example of novel interaction between cloud providers and OQH is given by the Amazon AWS application developed for QICK, called SideQICK, which is being integrated into a more general cloud queue for quantum devices [142].

We also witness examples of a hybrid model of industry-public interaction enabling open access. Quantum computers are being shipped to research centers and further integrated with HPC centers, which also act as simulator infrastructure (such as the EU HPCQS). This provides new avenues for industry-academic collaborations and purpose-specific hardware customization, use cases and access. It will be important to foster academic research labs putting their quantum devices online, building upon the open-source tools described in the previous sections. This could further change the research landscape, enabling more researchers to test research ideas on more devices, novel architectures and platforms, and in turn facilitate technology transfer.

2. Deeply collaboratively testbeds

While cloud-based platforms allow scientists and the public alike to submit their circuits and receive the results, deeper, customized access to the full stack to enable more involved experiments and R&D efforts is generally not possible on these platforms. To provide such access to users in Academia, Industry, and National Labs, the US Department of Energy has funded two testbed programs: QSCOUT (Quantum Scientific Computing Open User Testbed), based on trapped ions and located at Sandia National Laboratories, and AQT (Advanced Quantum Testbed), based on superconducting qubits and located at Lawrence Berkeley National Laboratory. A similar network called QuTest [143] managed by TNO at Delft, Netherlands has recently been kicked off with the goal of providing a federated network of testbeds by bringing together 13 service providers and 11 industrial users from the European quantum community. These platforms allow low-level access to the full stack of quantum hardware (including programming languages [144, 145], gate-pulse shaping [103], noise injection, unique qubit/qudit states, specialized calibrations, comparison of compilation techniques, etc.) allowing users to probe how the machines actually work and how to make them work better. The testbeds foster deep collaborations between the host laboratory and the users.

Collaborations on these testbeds have so far resulted in demonstrations of scientific applications of near-term quantum computers, and the development of many tools to benchmark quantum computers as well as to characterize and mitigate errors on them QPUs. This could further change the research landscape, enabling more researchers to test research ideas on more devices. Moreover, the existence of testbeds would be crucial to enable reproducible benchmarks, for quantum hardware and for application-driven tasks. Testbeds are key to facilitate startup creation in the quantum space, as they lower the barrier for testing prototypes with expensive equipment (such as dilution fridges).

3. Fabrication and foundries

As described in Section II A, open-source tools exist to facilitate the design and creation of quantum processors *in silico*. However, translating those designs into actual hardware can be difficult and expensive. Current arrangements for producing hardware on the basis of designs fall into four categories: partnerships, the construction of research-grade fab facilities, the use of academic foundries, and "quantum-fab-as-a-service".

Several companies in recent years have begun partnerships with major manufacturers to procure supply lines for their hardware designs. For example, photonic startups PsiQuantum and Xanadu have both signed agreements with semiconductor manufacturers GlobalFoundries [146, 147]. Trapped-ion startup IonQ has sourced some of its traps from Sandia National Laboratories [62]. These sorts of partnerships are typically only available to startups or other corporations, and allow for the use of use of existing manufacturing techniques and scalable processes. However, whether rapid prototype of hardware can be achieved is unclear.

An alternative approach is to stand up one's own fabrication facility. Startup Rigetti Computing is notable in this space for their "Fab-1" facility [148], which allows the company to prototyping new hardware. However, such a facility can be expensive to construct, and requires access to large amounts of capital. In the United States, in recognition of the need for fabrication facilities, various foundries are being stood up around the country, including the UCSB quantum foundry [149], MonArk quantum foundry [150], and the LPS Qubit Collaboratory [151]. The European Commission has established a somewhat similar program called Qu-Pilot [143] under the umbrella of the Quantum Flagship consisting of 21 partners from 9 different countries, with the goal of developing and providing access to federated European production facilities linking existing infrastructure. The overall coordination of the project is managed by VTT, Finland.

Finally, some startups have leveraged this need toward the development of new businesses. For example, startup Quantware in the Netherlands [152] helps designing, developing, and fabricating hardware for customers. In Canada, the NSERC CREATE programs have partnered with CMC Microsystems for novel superconducting circuits workshops at the Stewart Blusson Quantum Matter Institute. In the future, more industry players, industry consortia, startup incubators, and academic partnerships could enable even more facilities for processor fabrication, inspired by existing electronics industry frameworks, such as the Efabless [153] and the Google Silicon project [154].

III. DISCUSSION: CURRENT GAPS & FUTURE RECOMMENDATIONS

In Sec. II we reviewed OQH today, with an overview of the various OQH categories and deep dives into selected projects. From this overview it is possible to draw an overarching picture and outline some major topics across the OQH ecosystem. In the sections below we list these topics, identifying gaps and making recommendations to close them.

OQH growth and maintenance across technologies and architectures. Currently, most of the OQH projects focus on tools for quantum computers. There are considerable opportunities for expanding OQH projects and tools to accelerate scientific discovery and tech transfer in quantum communication, quantum metrology, and quantum sensing. Moreover, given the early stage of the field, for specific functionalities, tooling supporting only a given set of architectures may be currently available. For example, the open hardware tools for chip design are mostly revolving around SC qubits (and partially ion traps). There is room for tools for other QPU architectures, such as atom-based, spin-based, and photonics-based QPUs. With respect to control, open hardware projects including firmware and hardware, such as FPGAs, have been developed mostly for ion traps and SC qubits and can more broadly applied to other architectures. Moreover, while it is possible to find multiple open-source FPGA (and general open-hardware electronics projects with many being applicable to physics), one would be hard-pressed to find anything that comes closest to an optical frequency comb in photonics tooling.

APIs and standards for instruction sets. We note that interoperability is a major challenge in the OQH ecosystem. From APIs in the software stack to software-hardware integration via ISA, there is work to be done. Currently, there is considerable duplication on higher software stack and one needs to do conversions of quantum programs in order to run them on different QPUs. Frameworks such as the Quantum Intermediate Representation (QIR) project and OpenQASM (Open Quantum Assembly Language) [155, 156] can help at the higher level. Broader support for common frameworks getting further down at the hardware control level would be desired, such as QUASAR and QUA.

Benchmarks. Open-source benchmarking suites can have strong impact on the scientific community, providing information on the state of the art of current hardware architectures. So far we note that quantum technologies have been confronted with a lack of standardized benchmarks. Benchmarks are useful for evaluating the performance of

quantum devices and algorithms, as well as assessing the relative performance compared to purely-classical solutions in relevant applications or non-conventional computers [157]. Thankfully, efforts in the standardization of benchmarks, such as those provided by the QED-C, a U.S.-based industry consortium [42–44] have been occurring. Pipelines for crowd-sourced benchmark results is also needed: https://metriq.info/ is one attempt, inspired by projects such as Papers With Code and MLCommons in machine learning. Pipelines including OSS and OQH can simplify the accomplishment of standardized benchmarks.

Open access to hardware. Currently, there exist several cloud-accessible QPUs and cloud service frameworks. Providers have made QPUs accessible in a variety of ways, including direct and free access, as well as through partnerships within institutional frameworks (e.g.,ORNL's Quantum Computing User Program (QCUP) for national labs and federally-funded consortia). However, the research community would benefit from further access enabled by grants and other awards. Ensuring such frameworks and access points increase as soon as possible is necessary for creating a more equitable research landscape. Moreover, while several research labs have infrastructure to enable remote access and data acquisition of experiments, this is generally for internal use. Sometimes, access is extended to partnering organizations or collaborating researchers. However, there is a general lack of research labs who have put their device online for cloud access to a wider community of users and researchers. This is due to the overhead in infrastructure involved in making such devices operational and maintained. OQH can help with this regard, by providing researchers with existing tools that can be plugged in to bring up experiments from research labs.

Reproducibility. In order to facilitate reproducibility of results, journals and academic guidelines can further encourage (and mandate where appropriate) the sharing of open software along with research publications. Funding agencies, universities and other research stakeholders can facilitate these efforts by recognizing software artifacts as important, citable outcomes for measuring research impact.

OQH policy and intellectual property. Implementing policy that is open-hardware aware can favor the growth of a OQH ecosystem. For example, learning from existing open hardware projects beyond quantum technology can help avoid known pitfalls. A practical example is the adoption of an integrated toolchain fostering collaboration between academic labs, software developers, and industry. An example at the facility level is the investment into the establishment of shared facilities, such as quantum foundries, which can provide a flywheel effect for startups. Policy makers and quantum experts have the opportunity to work together, in order to ensure that informed decisions are made about the application of export control and intellectual property protection laws to quantum hardware. On the one hand, this could ensure that policy makers provide clear guidance about how efforts to open up quantum hardware will be treated from the perspective of these laws. On the other hand, researchers can develop expertise on the correct use of licenses for open-source software and open hardware. This guidance can include implementing processes through which one can more easily assess what hardware-based technology should be open-sourced, and what should not, making strategic calls at a narrow level (researcher, company) and wider level (common good for society). Researchers need to balance against intellectual property (IP) & strategic competitive considerations for certain components and technologies that could provide commercial exploitation. At the same time, they can consider when OSS can be used in business models, helping grow a community of users and creators around products.

Support of OQH in education and academia. We believe it is important to foster OSS and OQH adoption in education and research. One actionable item is to encourage schools and universities to include in course material and labs the usage of open-source software [158] and open hardware [159–161]. Programs that prepare the future quantum workforce are fundamental to broaden the acquisition of talents [5, 6, 24]. This includes developing communities across domain knowledge.

A bottom-up approach to strengthening OQH is by encouraging academic labs to pull together their systems with existing open-source tools, so they can create do-it-yourself small-scale testbeds. If open-source tools are not available, researchers can consider developing and open-sourcing new tools or using less-closed tools (e.g., proprietary but with a layer of open APIs). Fostering the creation of new open-hardware projects includes engaging in activities, programs and material that facilitates the open-sourcing of existing projects, such as Ref. [162], which provides information about starting a scientific OSS project. Starting an OQH project includes general challenges found by other OSS projects (testing, documentation, distribution, maintenance, community relations) and some specific ones, such as how to distribute physical devices to users, or work with experimental labs on ensuring device control.

An example of a top-down approach to supporting open-source projects (and open hardware) is provided by funding agencies programs acknowledging the impact of OSS projects in science. One notable example is the NSF POSE program. Further tailored funding in programs focused in OQH could strengthen the ecosystem addressing some of its characteristics.

Community building: projects support and governance. Similarly to other OSS projects, also OQH projects need to interact with their community of users, engage feedback, create guidelines to foster code contributions and meetings. They can benefit from non-profit organization support for their governance, as typically scientific OSS projects start in a research lab or group and then grow in scope and need for representation of more parties and partners. There is a need to simplify the bureaucracy in compliance at institutional at government level to contribute to open-source projects and open hardware, e.g., by equipping national labs with default policies that would allow researchers to contribute to open-source projects with permissive licenses approved by the Open Source Initiative (OSI). Often the originating organization such a national lab has restrictions that prevent an open-source project from growing independently. This highlights the need for alternatively supporting organizations to incubate and house open-source projects. Examples of non-profit organizations that do this in the classical space are NumFOCUS, the Linux Foundation, and the Open Source Hardware Association (OSHA). Unitary Fund is a 501(c)(3) non-profit organization that offers this support specifically to quantum technology projects.

IV. CONCLUSIONS

Quantum computers – and quantum devices in general – provide a fascinating prospect to bridge a platform for fundamental science, technology transfer in engineering, and novelties in cloud computing and actuators. An open hardware approach in quantum technology will preserve, and possibly expand, the pre-competitive space necessary to make ideas flourish and enable accessible benchmarks and evaluations that have already characterized quantum software.

As we have illustrated, quantum technology provides unique challenges and features with regards to the usability and feasibility of open-hardware stack: from process design toolkit to foundries, from simulation software to control and data acquisition, from cloud infrastructure to testbeds. All these layers can benefit from collaborative innovation. Learning from lessons learned in the early days of quantum software, and from other open hardware verticals in conventional computing and tooling, as well as from a snapshot of current gaps in the quantum ecosystem, we hope that the quantum industry can efficiently implement innovation, build lively communities, and expand its workforce.

V. ACKNOWLEDGMENTS

This work stemmed from a workshop [10] hosted at IEEE QCE 2021. The authors thank Gary Steele and Zlatko Minev for fruitful discussions. NS thanks the organizers of IEEE QCE 2021 for the opportunity to start the discussion of OQH which culminated in this manuscript, and many of the co-authors for their contributions at that workshop. This material was funded in part by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research Quantum Testbed Program. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government. SAND2023-09347O.

This work was supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Accelerated Research in Quantum Computing under Award Number DE-SC0020266. AM acknowledges support from the PNRR MUR project PE0000023-NQSTI.

[1] W. Zeng, B. Johnson, R. Smith, N. Rubin, M. Reagor, C. Ryan, and C. Rigetti, First quantum computers need smart software, Nature **549**, 149 (2017).
[2] M. Fingerhuth, T. Babej, and P. Wittek, Open source software in quantum computing, PLOS ONE **13**, e0208561 (2018).
[3] C. Monroe, M. G. Raymer, and J. Taylor, The U.S. national quantum initiative: From act to action, Science **364**, 440 (2019).
[4] M. F. J. Fox, B. M. Zwickl, and H. J. Lewandowski, Preparing for the quantum revolution: What is the role of higher education?, Phys. Rev. Phys. Educ. Res. **16**, 020131 (2020).
[5] A. Asfaw, A. Blais, K. R. Brown, J. Candelaria, C. Cantwell, L. D. Carr, J. Combes, D. M. Debroy, J. M. Donohue, S. E. Economou, *et al.*, Building a quantum engineering undergraduate program, IEEE Transactions on Education 10.1109/TE.2022.3144943 (2022).

[6] A. S. Dzurak, J. Epps, A. Laucht, R. A. Malaney, A. Morello, H. I. Nurdin, J. J. Pla, A. Saraiva, and C. H. Yang, Development of an undergraduate quantum engineering degree, IEEE Transactions on Quantum Engineering **3**, 1 (2022).

[7] S. Bourdeauducq, R. Jördens, P. Zotov, J. Britton, D. Slichter, D. Leibrandt, D. Allcock, A. Hankin, F. Kermarrec, Y. Sionneau, R. Srinivas, T. R. Tan, and J. Bohnet, Artiq 1.0 (2016).

[8] Z. K. Minev, Z. Leghtas, S. O. Mundhada, L. Christakis, I. M. Pop, and M. H. Devoret, Energy-participation quantization of Josephson circuits (2021), arXiv:2010.00620 [quant-ph].

[9] T. L. Scholten and D. Greenberg, Workshop on Software for Quantum Applications, Algorithms, and Workflows at QCE20 (2020).

[10] T. L. Scholten and N. Shammah, Workshop on open quantum hardware: Accelerating the control, and use, of quantum computing systems at QCE21 (2021).

[11] M. Köfferlein, KLayout (2007).

[12] R. Thew, T. Jennewein, and M. Sasaki, Focus on quantum science and technology initiatives around the world, Quantum Sci. Technol **5**, 010201 (2019).

[13] T. Roberson and A. G. White, Charting the Australian quantum landscape, Quantum Science and Technology **4**, 020505 (2019).

[14] Q. Zhang, F. Xu, L. Li, N.-L. Liu, and J.-W. Pan, Quantum information research in China, Quantum Science and Technology **4**, 040503 (2019).

[15] M. Riedel, M. Kovacs, P. Zoller, J. Mlynek, and T. Calarco, Europe's quantum flagship initiative, Quantum Science and Technology **4**, 020501 (2019).

[16] E. Gibney, Billion-euro quantum project takes shape, Nature **545**, 16 (2017).

[17] P. Knight and I. Walmsley, UK national quantum technology programme, Quantum Science and Technology **4**, 040502 (2019).

[18] Y. Yamamoto, M. Sasaki, and H. Takesue, Quantum information science and technology in Japan, Quantum Science and Technology **4**, 020502 (2019).

[19] A. Fedorov, A. Akimov, J. Biamonte, A. Kavokin, F. Y. Khalili, E. Kiktenko, N. Kolachevsky, Y. Kurochkin, A. Lvovsky, A. Rubtsov, *et al.*, Quantum technologies in russia, Quantum Science and Technology **4**, 040501 (2019).

[20] C. I. Merzbacher, US quantum initiatives: from R&D to innovation, in *Emerging Imaging and Sensing Technologies for Security and Defence V; and Advanced Manufacturing Technologies for Micro- and Nanosystems in Security and Defence III*, Vol. 11540, edited by G. S. Buller, R. C. Hollins, R. A. Lamb, M. Laurenzis, A. Camposeo, M. Farsari, L. Persano, and L. E. Busse, International Society for Optics and Photonics (SPIE, 2020).

[21] M. G. Raymer and C. Monroe, The US national quantum initiative, Quantum Science and Technology **4**, 020504 (2019).

[22] C. Monroe, M. G. Raymer, and J. Taylor, The us national quantum initiative: From act to action, Science **364**, 440 (2019).

[23] B. Sussman, P. Corkum, A. Blais, D. Cory, and A. Damascelli, Quantum Canada, Quantum Science and Technology **4**, 020503 (2019).

[24] C. D. Aiello, D. D. Awschalom, H. Bernien, T. Brower, K. R. Brown, T. A. Brun, J. R. Caram, E. Chitambar, R. D. Felice, K. M. Edmonds, M. F. J. Fox, S. Haas, A. W. Holleitner, E. R. Hudson, J. H. Hunt, R. Joynt, S. Koziol, M. Larsen, H. J. Lewandowski, D. T. McClure, J. Palsberg, G. Passante, K. L. Pudenz, C. J. K. Richardson, J. L. Rosenberg, R. S. Ross, M. Saffman, M. Singh, D. W. Steuerman, C. Stark, J. Thijssen, A. N. Vamivakas, J. D. Whitfield, and B. M. Zwickl, Achieving a quantum smart workforce, Quantum Science and Technology **6**, 030501 (2021).

[25] P. Mirowski, The future(s) of open science, Social Studies of Science **48**, 171 (2018).

[26] A. Powell, Democratizing production through open source knowledge: From open software to open hardware, Media, Culture & Society **34**, 691 (2012).

[27] R. Viseur, From open source software to open source hardware, in *IFIP International Conference on Open Source Systems* (Springer, 2012) pp. 286–291.

[28] K. P. Gerhardt, E. J. Olson, S. M. Castillo-Hair, L. A. Hartsough, B. P. Landry, F. Ekness, R. Yokoo, E. J. Gomez, P. Ramakrishnan, J. Suh, D. F. Savage, and J. J. Tabor, An open-hardware platform for optogenetics and photobiology, Scientific Reports **6**, 35363 (2016).

[29] F. Morreale, G. Moro, A. Chamberlain, S. Benford, and A. P. McPherson, Building a maker community around an open hardware platform, in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17 (Association for Computing Machinery, New York, NY, USA, 2017) p. 6948–6959.

[30] K. Blind, M. Böhm, P. Grzegorzewska, A. Katz, S. Muto, S. Pätsch, and T. Schubert, The impact of open source software and hardware on technological independence, competitiveness and innovation in the EU economy (2021).

[31] A. Soriano, L. Marín, M. Vallés, A. Valera, and P. Albertos, Low cost platform for automatic control education based on open hardware., IFAC Proceedings Volumes **47**, 9044 (2014), 19th IFAC World Congress.

[32] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, *et al.*, Scipy 1.0: fundamental algorithms for scientific computing in Python, Nature Methods **17**, 261 (2020).

[33] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, Array programming with NumPy, Nature **585**, 357 (2020).

[34] J. Johansson, P. Nation, and F. Nori, QuTiP: An open-source python framework for the dynamics of open quantum systems, Computer Physics Communications **183**, 1760 (2012).

[35] J. Johansson, P. Nation, and F. Nori, QuTiP 2: A Python framework for the dynamics of open quantum systems, Computer Physics Communications **184**, 1234 (2013).

[36] N. Shammah, S. Ahmed, N. Lambert, S. De Liberato, and F. Nori, Open quantum systems with local and collective incoherent processes: Efficient numerical simulations using permutational invariance, Phys. Rev. A **98**, 063815 (2018).

[37] N. Lambert, S. Ahmed, M. Cirio, and F. Nori, Modelling the ultra-strongly coupled spin-boson model with unphysical modes, Nature Communications **10**, 10.1038/s41467-019-11656-1 (2019).

[38] P. Groszkowski and J. Koch, Scqubits: a Python package for superconducting qubits, Quantum **5**, 583 (2021).

[39] B. Li, S. Ahmed, S. Saraogi, N. Lambert, F. Nori, A. Pitchford, and N. Shammah, Pulse-level noisy quantum circuits with QuTiP, Quantum **6**, 630 (2022).

[40] M. Banzi and M. Shiloh, *Getting Started with Arduino*, 4th ed. (Make Books - Imprint of: O'Reilly Media, Sebastopol, CA, 2022).

[41] M. Richardson and S. Wallace, *Getting started with Raspberry PI* (O'Reilly Media, Inc., 2012).

[42] T. Lubinski, S. Johri, P. Varosy, J. Coleman, L. Zhao, J. Necaise, C. H. Baldwin, K. Mayer, and T. Proctor, Application-oriented performance benchmarks for quantum computing (2023), arXiv:2110.03137 [quant-ph].

[43] T. Lubinski, C. Coffrin, C. McGeoch, P. Sathe, J. Apanavicius, and D. E. B. Neira, Optimization applications as quantum performance benchmarks (2023), arXiv:2302.02278 [quant-ph].

[44] M. Amico, H. Zhang, P. Jurcevic, L. S. Bishop, P. Nation, A. Wack, and D. C. McKay, Defining standard strategies for quantum benchmarks (2023), arXiv:2303.02108 [quant-ph].

[45] J. Jagatheesan, DASQA (2023).

[46] J. Kunasaikaran, K. Mato, and R. Wille, A framework for the design and realization of alternative superconducting quantum architectures (2023), arXiv:2305.07052 [quant-ph].

[47] D. Cucurachi, A. Guthrie, J. Heinsoo, S. Inel, D. Janzsó, M. Jenei, K. Juliusson, J. Kotilahti, A. Landra, C. Ockeloen-Korppi, J. Räbinä, N. Savola, P. Smirnov, and E. Takala, KQCircuits (2021).

[48] Z. K. Minev, T. G. McConkey, J. Drysdale, P. Shah, D. Wang, M. Facchini, G. Harper, J. Blair, H. Zhang, N. Lanzillo, S. Mukesh, W. Shanks, C. Warren, and J. M. Gambetta, Qiskit Metal: An Open-Source Framework for Quantum Device Design & Analysis (2021).

[49] M. F. Gely and G. A. Steele, QuCAT: quantum circuit analyzer tool in python, New Journal of Physics **22**, 013025 (2020).

[50] H. Silvério, S. Grijalva, C. Dalyac, L. Leclerc, P. J. Karalekas, N. Shammah, M. Beji, L.-P. Henry, and L. Henriet, Pulser: An open-source package for the design of pulse sequences in programmable neutral-atom arrays (2021), arXiv:2104.15044 [quant-ph].

[51] A. V. Horn, Duke artiq repository (2023).

[52] L. Ella, L. Leandro, O. Wertheim, Y. Romach, R. Szmuk, Y. Knol, N. Ofek, I. Sivan, and Y. Cohen, Quantum-classical processing and benchmarking at the pulse-level (2023), arXiv:2303.03816 [quant-ph].

[53] J. H. Nielsen, Qcodes/qcodes: Qcodes 0.40.0 (2023).

[54] L. Stefanazzi, K. Treptow, N. Wilcer, C. Stoughton, S. Montella, C. Bradford, G. Cancelo, S. Saxena, H. Arnaldi, S. Sussman, A. Houck, A. Agrawal, H. Zhang, C. Ding, and D. I. Schuster, The QICK (Quantum Instrumentation Control Kit): Readout and control for qubits and detectors (2021).

[55] K. J. Mesman, F. Battistel, E. Reehuis, D. de Jong, M. J. Tiggelman, J. Gloudemans, J. C. van Oven, and C. C. Bultink, Q-profile: Profiling tool for quantum control stacks applied to the quantum approximate optimization algorithm (2023), arXiv:2303.01450 [quant-ph].

[56] Y. Xu, G. Huang, J. Balewski, A. Morvan, K. Nowrouzi, D. I. Santiago, R. K. Naik, B. Mitchell, and I. Siddiqi, Automatic qubit characterization and gate optimization with qubic, ACM Transactions on Quantum Computing **4**, 10.1145/3529397 (2022).

[57] S. Humpohl, L. Prediger, pcerf, P. Eendebak, P. Bethke, A. Willmes, J. Bergmann, M. Meyer, T. Hangleiter, E. Kammerloher, and qutech lab, qupulse (2023).

[58] Sinara open hardware project (2022).

[59] P. Kulik, M. Sowiński, G. Kasprowicz, D. Allcock, C. Ballance, S. Bourdeauducq, J. Britton, M. Gąska, T. Harty, J. Jarosiński, R. Jördens, M. Kiepiela, N. Krackow, D. Nadlinger, K. Poźniak, T. Przywózki, D. Slichter, F. Świtakowski, M. Weber, A. Wojciechowski, and W. Zhang, Latest developments in the Sinara open hardware ecosystem, in *2022 IEEE International Conference on Quantum Computing and Engineering (QCE)* (2022) pp. 799–802.

[60] OpenSuperQPlus, Open superconducting quantum computers (2023).

[61] QuTech, Quantum inspire (2023).

[62] S. M. Clark, D. Lobser, M. C. Revelle, C. G. Yale, D. Bossert, A. D. Burch, M. N. Chow, C. W. Hogle, M. Ivory, J. Pehr, B. Salzbrenner, D. Stick, W. Sweatt, J. M. Wilson, E. Winrow, and P. Maunz, Engineering the quantum scientific computing open user testbed, IEEE Transactions on Quantum Engineering **2**, 1 (2021).

[63] Y. Y. Gao, M. A. Rol, S. Touzard, and C. Wang, Practical guide for building superconducting quantum devices, PRX Quantum **2**, 040202 (2021).

[64] C. Tahan, Opinion: Democratizing Spin Qubits, Quantum **5**, 584 (2021).

[65] X. Fu, M. A. Rol, C. C. Bultink, J. van Someren, N. Khammassi, I. Ashraf, R. F. L. Vermeulen, J. C. de Sterke, W. J. Vlothuizen, R. N. Schouten, C. G. Almudever, L. DiCarlo, and K. Bertels, An experimental microarchitecture for a superconducting quantum processor, in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-50 '17 (Association for Computing Machinery, New York, NY, USA, 2017) p. 813–825.

[66] G. Feng, S.-Y. Hou, H. Zou, W. Shi, S. Yu, Z. Sheng, X. Rao, K. Ma, C. Chen, B. Ren, G. Miao, J. Xiang, and B. Zeng, Spinq triangulum: a commercial three-qubit desktop quantum computer (2022).

[67] A. Castellanos-Gomez, M. Buscema, R. Molenaar, V. Singh, L. Janssen, H. S. J. van der Zant, and G. A. Steele, Deterministic transfer of two-dimensional materials by all-dry viscoelastic stamping, 2D Materials **1**, 011002 (2014).

[68] J. Ziegler, T. McJunkin, E. Joseph, S. S. Kalantre, B. Harpt, D. Savage, M. Lagally, M. Eriksson, J. M. Taylor, and J. P. Zwolak, Toward robust autotuning of noisy quantum dot devices, Phys. Rev. Applied **17**, 024069 (2022).

[69] R. Pollice, G. dos Passos Gomes, M. Aldeghi, R. J. Hickman, M. Krenn, C. Lavigne, M. Lindner-D'Addario, A. Nigam, C. T. Ser, Z. Yao, and A. Aspuru-Guzik, Data-driven strategies for accelerated materials design, *Accounts of Chemical Research*, Accounts of Chemical Research **54**, 849 (2021).

[70] W. Ma, Z. Liu, Z. A. Kudyshev, A. Boltasseva, W. Cai, and Y. Liu, Deep learning for the design of photonic structures, Nature Photonics **15**, 77 (2021).

[71] S. Molesky, Z. Lin, A. Y. Piggott, W. Jin, J. Vucković, and A. W. Rodriguez, Inverse design in nanophotonics, Nature Photonics **12**, 659 (2018).

[72] Ansys, Ansys (2023).

[73] S. Software, Sonnet (2023).

[74] CSC, Elmer (2023).

[75] AWS, Palace (2023).

[76] C. Geuzaine and J.-F. Remacle, Gmsh: A 3-d finite element mesh generator with built-in pre- and post-processing facilities, International Journal for Numerical Methods in Engineering **79**, 1309 (2009).

[77] T. Alexander, N. Kanazawa, D. J. Egger, L. Capelluto, C. J. Wood, A. Javadi-Abhari, and D. C. McKay, Qiskit pulse: programming quantum computers through the cloud with pulses, Quantum Science and Technology **5**, 044006 (2020).

[78] H. Ball, M. J. Biercuk, A. Carvalho, J. Chen, M. Hush, L. A. De Castro, L. Li, P. J. Liebermann, H. J. Slatyer, C. Edmunds, V. Frey, C. Hempel, and A. Milne, Software tools for quantum control: Improving quantum computer performance through noise and error suppression (2020).

[79] H. Silvério, S. Grijalva, C. Dalyac, L. Leclerc, P. J. Karalekas, N. Shammah, M. Beji, L.-P. Henry, and L. Henriet, Pulser: An open-source package for the design of pulse sequences in programmable neutral-atom arrays, Quantum **6**, 629 (2022).

[80] N. Wittler, F. Roy, K. Pack, M. Werninghaus, A. S. Roy, D. J. Egger, S. Filipp, F. K. Wilhelm, and S. Machnes, Integrated tool set for control, calibration, and characterization of quantum devices applied to superconducting qubits, Phys. Rev. Applied **15**, 034080 (2021).

[81] A. Saha Roy, K. Pack, N. Wittler, and S. Machnes, Software tool-set for automated quantum system identification and device bring up, arXiv preprint arXiv:2205.04829 10.48550/ARXIV.2205.04829 (2022).

[82] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, JAX: composable transformations of Python+NumPy programs (2018).

[83] Y. Xu, G. Huang, J. Balewski, R. Naik, A. Morvan, B. Mitchell, K. Nowrouzi, D. I. Santiago, and I. Siddiqi, QubiC: An open-source FPGA-based control and measurement system for superconducting quantum information processors, IEEE Transactions on Quantum Engineering **2**, 1 (2021).

[84] G. Li, A. Wu, Y. Shi, A. Javadi-Abhari, Y. Ding, and Y. Xie, On the co-design of quantum software and hardware, in *Proceedings of the Eight Annual ACM International Conference on Nanoscale Computing and Communication*, NANOCOM '21 (Association for Computing Machinery, New York, NY, USA, 2021).

[85] L. M. Roch, F. HÃ€se, C. Kreisbeck, T. Tamayo-Mendoza, L. P. E. Yunker, J. E. Hein, and A. Aspuru-Guzik, Chemos: An orchestration software to democratize autonomous discovery, PLOS ONE **15**, 1 (2020).

[86] M. Krenn, J. S. Kottmann, N. Tischler, and A. Aspuru-Guzik, Conceptual understanding through efficient automated design of quantum optical experiments, Phys. Rev. X **11**, 031044 (2021).

[87] A. A. Melnikov, H. P. Nautrup, M. Krenn, V. Dunjko, M. Tiersch, A. Zeilinger, and H. J. Briegel, Active learning machine learns to create new quantum experiments, Proceedings of the National Academy of Sciences **115**, 1221 (2018).

[88] K. N. Smith, G. S. Ravi, T. Alexander, N. T. Bronn, A. Carvalho, A. Cervera-Lierta, F. T. Chong, J. M. Chow, M. Cubeddu, A. Hashim, L. Jiang, O. Lanes, M. J. Otten, D. I. Schuster, P. Gokhale, N. Earnest, and A. Galda, Summary: Chicago quantum exchange (CQE) pulse-level quantum control workshop (2022).

[89] J. M. Perkel, Why jupyter is data scientists' computational notebook of choice, Nature **563**, 145 (2018).

[90] J. M. Arrazola, T. R. Bromley, J. Izaac, C. R. Myers, K. Brádler, and N. Killoran, Machine learning method for state preparation and gate synthesis on photonic quantum computers, Quantum Science and Technology **4**, 024004 (2019).

[91] R. D. King, J. Rowland, S. G. Oliver, M. Young, W. Aubrey, E. Byrne, M. Liakata, M. Markham, P. Pir, L. N. Soldatova, *et al.*, The automation of science, Science **324**, 85 (2009).

[92] P. S. Gromski, A. B. Henson, J. M. Granda, and L. Cronin, How to explore chemical space using algorithms and automation, Nature Reviews Chemistry **3**, 119 (2019).

[93] J. P. Zwolak and J. M. Taylor, Colloquium: Advances in automation of quantum dot devices control (2021).

[94] A. Shaout and T. Eldos, On the classification of computer architecture, International Journal of Science and Technology **14** (2003).

[95] E. Lindholm, J. Nickolls, S. Oberman, and J. Montrym, Nvidia tesla: A unified graphics and computing architecture, IEEE Micro **28**, 39 (2008).

[96] S. Balensiefer, L. Kregor-Stickles, and M. Oskin, An evaluation framework and instruction set architecture for ion-trap based quantum micro-architectures, in *32nd International Symposium on Computer Architecture (ISCA'05)* (IEEE, 2005) pp. 186–196.

[97] R. S. Smith, M. J. Curtis, and W. J. Zeng, A practical quantum instruction set architecture (2017), arXiv:1608.03355 [quant-ph].

[98] X. Fu, L. Riesebos, M. Rol, J. Van Straten, J. Van Someren, N. Khammassi, I. Ashraf, R. Vermeulen, V. Newsum,

K. Loh, *et al.*, eQASM: An executable quantum instruction set architecture, in *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)* (IEEE, 2019) pp. 224–237.
[99] A. Butko, G. Michelogiannakis, S. Williams, C. Iancu, D. Donofrio, J. Shalf, J. Carter, and I. Siddiqi, Understanding quantum control processor capabilities and limitations through circuit characterization, in *2020 International Conference on Rebooting Computing (ICRC)* (IEEE Computer Society, Los Alamitos, CA, USA, 2020) pp. 66–75.
[100] A. Waterman, Y. Lee, R. Avizienis, D. A. Patterson, and K. Asanović, *The RISC-V Instruction Set Manual Volume II: Privileged Architecture Version 1.9*, Tech. Rep. UCB/EECS-2016-129 (EECS Department, University of California, Berkeley, 2016).
[101] K. Asanović, R. Avižienis, J. Bachrach, S. Beamer, D. Biancolin, C. Celio, H. Cook, P. Dabbelt, J. Hauser, A. Izraelevitz, S. Karandikar, B. Keller, D. Kim, J. Koenig, Y. Lee, E. Love, M. Maas, A. Magyar, H. Mao, M. Moreto, A. Ou, D. Patterson, B. Richards, C. Schmidt, S. Twigg, H. Vo, and A. Waterman, *"The Rocket Chip Generator"*, Tech. Rep. ("UCB/EECS-2016-17, EECS Department, University of California", "2016").
[102] A. Amid, D. Biancolin, A. Gonzalez, D. Grubb, S. Karandikar, H. Liew, A. Magyar, H. Mao, A. Ou, N. Pemberton, P. Rigge, C. Schmidt, J. Wright, J. Zhao, Y. S. Shao, K. Asanović, and B. Nikolić, Chipyard: Integrated design, simulation, and implementation framework for custom socs, IEEE Micro **40**, 10 (2020).
[103] D. Lobser, J. Goldberg, A. J. Landahl, P. Maunz, B. C. A. Morrison, K. Rudinger, A. Russo, B. Ruzic, D. Stick, J. V. D. Wall, and S. M. Clark, JaqalPaw: A guide to defining pulses and waveforms for Jaqal (2023), arXiv:2305.02311 [quant-ph].
[104] R. B.-S. Tsai, H. Silvério, and L. Henriet, Pulse-level scheduling of quantum circuits for neutral-atom devices (2022), arXiv:2206.05144 [quant-ph].
[105] G. Huang, Y. Xu, L. Doolittle, U. Baek, and I. Siddiqi, Scalable fpga-based qubit control hardware, in *APS March Meeting Abstracts*, Vol. 2019 (2019) pp. V26–014.
[106] Y. Xu, G. Huang, D. I. Santiago, and I. Siddiqi, Radio frequency mixing modules for superconducting qubit room temperature control systems, Review of Scientific Instruments **92** (2021).
[107] Y. Xu, G. Huang, N. Fruitwala, R. Naik, K. Nowrouzi, D. Santiago, and I. Siddiqi, Fpga module synchronization for qubic, in *APS March Meeting Abstracts*, Vol. 2023 (2023).
[108] N. Fruitwala, G. Huang, Y. Xu, R. Naik, K. Nowrouzi, D. Santiago, and I. Siddiqi, Distributed processor for fpga-based superconducting qubit control using qubic, Bulletin of the American Physical Society (2023).
[109] A. Butko, Y. Xu, G. Huang, R. Naik, D. Santiago, and I. Siddiqi, Real-time fast feedback experiment enabled by a customized fpga-based control system, Bulletin of the American Physical Society (2023).
[110] H. Johnson, S. Zorzetti, and J. Saniie, Exploration of optimizing fpga-based qubit controller for experiments on superconducting quantum computing hardware, arXiv preprint arXiv:2305.06976 (2023).
[111] AMD-Xilinx, Zcu111 evaluation board user guide (ug1271) (2023).
[112] AMD-Xilinx, Zcu216 evaluation board user guide (ug1390) (2020).
[113] A. V. Dixit, S. Chakram, K. He, A. Agrawal, R. K. Naik, D. I. Schuster, and A. Chou, Searching for dark matter with a superconducting qubit, Phys. Rev. Lett. **126**, 141302 (2021).
[114] A. Browaeys and T. Lahaye, Many-body physics with individually controlled Rydberg atoms, Nature Physics **16**, 132 (2020).
[115] L. Henriet, L. Beguin, A. Signoles, T. Lahaye, A. Browaeys, G.-O. Reymond, and C. Jurczak, Quantum computing with neutral atoms, Quantum **4**, 327 (2020).
[116] M.-T. Nguyen, J.-G. Liu, J. Wurtz, M. D. Lukin, S.-T. Wang, and H. Pichler, Quantum optimization with arbitrary connectivity using rydberg atom arrays, PRX Quantum **4**, 010316 (2023).
[117] J. Werschnik and E. Gross, Quantum optimal control theory, Journal of Physics B: Atomic, Molecular and Optical Physics **40**, R175 (2007).
[118] C. P. Koch, U. Boscain, T. Calarco, G. Dirr, S. Filipp, S. J. Glaser, R. Kosloff, S. Montangero, T. Schulte-Herbrüggen, D. Sugny, *et al.*, Quantum optimal control in quantum technologies. strategic report on current status, visions and goals for research in europe, EPJ Quantum Technology **9**, 19 (2022).
[119] M. Goerz, D. Basilewitsch, F. Gago-Encinas, M. G. Krauss, K. P. Horn, D. M. Reich, and C. Koch, Krotov: A Python implementation of Krotov's method for quantum optimal control, SciPost Physics **7**, 10.21468/scipostphys.7.6.080 (2019).
[120] J. Kelly, R. Barends, B. Campbell, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, A. G. Fowler, I.-C. Hoi, E. Jeffrey, A. Megrant, J. Mutus, C. Neill, P. J. J. O'Malley, C. Quintana, P. Roushan, D. Sank, A. Vainsencher, J. Wenner, T. C. White, A. N. Cleland, and J. M. Martinis, Optimal quantum control using randomized benchmarking, Phys. Rev. Lett. **112**, 240504 (2014).
[121] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, TensorFlow: A system for Large-Scale machine learning, in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)* (USENIX Association, Savannah, GA, 2016) pp. 265–283.
[122] R. Frostig, M. J. Johnson, and C. Leary, Compiling machine learning programs via high-level tracing, Systems for Machine Learning **4** (2018).
[123] T. Propson, B. E. Jackson, J. Koch, Z. Manchester, and D. I. Schuster, Robust quantum optimal control with trajectory optimization, Phys. Rev. Applied **17**, 014036 (2022).
[124] C. Gidney, Stim: a fast stabilizer circuit simulator, Quantum **5**, 497 (2021).
[125] Q. Design, Plaquette, `https://github.com/qc-design/plaquette` (2023).
[126] A. Khalid, Stac, `https://github.com/abdullahkhalids/stac` (2023).
[127] O. Higgott, Pymatching: A python package for decoding quantum codes with minimum-weight perfect matching, arxiv

e-prints, arXiv preprint arXiv:2105.13082 (2021).
[128] O. Higgott and C. Gidney, Sparse blossom: correcting a million errors per core second with minimum-weight matching, arXiv preprint arXiv:2303.15933 (2023).
[129] M. J. Reagor, T. C. Bohdanowicz, D. R. Perez, E. A. Sete, and W. J. Zeng, Hardware optimized parity check gates for superconducting surface codes, arXiv preprint arXiv:2211.06382 (2022).
[130] J. Sáenz, J. Chacón, L. De La Torre, A. Visioli, and S. Dormido, Open and low-cost virtual and remote labs on control engineering, Ieee Access **3**, 805 (2015).
[131] L. Gomes and S. Bogosyan, Current trends in remote laboratories, IEEE Transactions on industrial electronics **56**, 4744 (2009).
[132] C. Kourkoumelis and S. Vourakis, Hypatia—an online tool for atlas event visualization, Physics Education **49**, 21 (2014).
[133] J. C. Norman, D. Jung, Y. Wan, and J. E. Bowers, Perspective: The future of quantum dot photonic integrated circuits, APL Photonics **3**, 030901 (2018).
[134] J. L. DuBois, V. Lordi, Y. J. Rosen, and X. Wu, Request for the establishment of quantum information foundries, Technical Report 10.2172/1670543 (2020).
[135] E. Bussmann, R. E. Butera, J. H. G. Owen, J. N. Randall, S. M. Rinaldi, A. D. Baczewski, and S. Misra, Atomic-precision advanced manufacturing for si quantum computing, MRS Bulletin **46**, 607 (2021).
[136] E. Altman, K. R. Brown, G. Carleo, L. D. Carr, E. Demler, C. Chin, B. DeMarco, S. E. Economou, M. A. Eriksson, K.-M. C. Fu, M. Greiner, K. R. Hazzard, R. G. Hulet, A. J. Kollár, B. L. Lev, M. D. Lukin, R. Ma, X. Mi, S. Misra, C. Monroe, K. Murch, Z. Nazario, K.-K. Ni, A. C. Potter, P. Roushan, M. Saffman, M. Schleier-Smith, I. Siddiqi, R. Simmonds, M. Singh, I. Spielman, K. Temme, D. S. Weiss, J. Vučković, V. Vuletić, J. Ye, and M. Zwierlein, Quantum simulators: Architectures and opportunities, PRX Quantum **2**, 017003 (2021).
[137] D. Awschalom, K. K. Berggren, H. Bernien, S. Bhave, L. D. Carr, P. Davids, S. E. Economou, D. Englund, A. Faraon, M. Fejer, S. Guha, M. V. Gustafsson, E. Hu, L. Jiang, J. Kim, B. Korzh, P. Kumar, P. G. Kwiat, M. Lončar, M. D. Lukin, D. A. Miller, C. Monroe, S. W. Nam, P. Narang, J. S. Orcutt, M. G. Raymer, A. H. Safavi-Naeini, M. Spiropulu, K. Srinivasan, S. Sun, J. Vučković, E. Waks, R. Walsworth, A. M. Weiner, and Z. Zhang, Development of quantum interconnects (quics) for next-generation information technologies, PRX Quantum **2**, 017002 (2021).
[138] Y. Alexeev, D. Bacon, K. R. Brown, R. Calderbank, L. D. Carr, F. T. Chong, B. DeMarco, D. Englund, E. Farhi, B. Fefferman, A. V. Gorshkov, A. Houck, J. Kim, S. Kimmel, M. Lange, S. Lloyd, M. D. Lukin, D. Maslov, P. Maunz, C. Monroe, J. Preskill, M. Roetteler, M. J. Savage, and J. Thompson, Quantum computer systems for scientific discovery, PRX Quantum **2**, 017001 (2021).
[139] V. Frey, R. Rademacher, E. Durso-Sabina, N. Greenberg, N. Videnov, M. L. Day, R. Islam, and C. Senko, Programming the full stack of an open-access quantum computer (2021), arXiv:2106.06549 [quant-ph].
[140] A. Solfanelli, A. Santini, and M. Campisi, Experimental verification of fluctuation relations with a quantum computer, PRX Quantum **2**, 030353 (2021).
[141] X. Mi, M. Ippoliti, C. Quintana, A. Greene, Z. Chen, J. Gross, F. Arute, K. Arya, J. Atalaya, R. Babbush, *et al.*, Time-crystalline eigenstate order on a quantum processor, Nature **601**, 531 (2022).
[142] D. Liukkonen, Cloud queue for quantum devices (2023).
[143] Q. Flagship, The quantum future begins in prague (2023).
[144] B. C. A. Morrison, A. J. Landahl, D. S. Lobser, K. M. Rudinger, A. E. Russo, J. W. Van Der Wall, and P. Maunz, Just another quantum assembly language (Jaqal), in *2020 IEEE Int. Conf. Quant. Comp. Eng. (QCE)* (2020) pp. 402–408.
[145] Jaqal (2020).
[146] PsiQuantum, PsiQuantum and GLOBALFOUNDRIES to Build the World's First Full-scale Quantum Computer (2022).
[147] Xanadu, Xanadu announces collaboration with GlobalFoundries to accelerate development of fault-tolerant photonic quantum computers (2022).
[148] J. Hsu, Rigetti Launches Full-Stack Quantum Computing Service and Quantum IC Fab (2017).
[149] UCSB, UCSB NSF Quantum Foundry (2022).
[150] MonArk, MonArk NSF Quantum Foundry (2022).
[151] LPS, LPS Qubit Collaboratory (2022).
[152] Quantware, Quantware (2022).
[153] Efabless (2023).
[154] Google silicon (2023).
[155] A. Cross, A. Javadi-Abhari, T. Alexander, N. D. Beaudrap, L. S. Bishop, S. Heidel, C. A. Ryan, P. Sivarajah, J. Smolin, J. M. Gambetta, and B. R. Johnson, OpenQASM 3: A broader and deeper quantum assembly language, ACM Transactions on Quantum Computing **3**, 1 (2022).
[156] A. W. Cross, L. S. Bishop, J. A. Smolin, and J. M. Gambetta, Open quantum assembly language (2017), arXiv:1707.03429 [quant-ph].
[157] N. Mohseni, P. L. McMahon, and T. Byrnes, Ising machines as hardware solvers of combinatorial optimization problems, Nature Reviews Physics **4**, 363 (2022).
[158] A. M. C. Dawes, Undergraduate quantum mechanics: a numerical approach using QuTiP (2019), arXiv:1909.13651 [physics.ed-ph].
[159] F. Bouquet, J. Bobroff, M. Fuchs-Gallezot, and L. Maurines, Project-based physics labs using low-cost open-source hardware, American Journal of Physics **85**, 216 (2017).
[160] Z. Gingl, J. Mellár, T. Szépe, G. Makan, R. Mingesz, G. Vadai, and K. Kopasz, Universal arduino-based experimenting system to support teaching of natural sciences, Journal of Physics: Conference Series **1287**, 012052 (2019).

[161] B. Ulmann, S. Köppel, and D. Killat, Open hardware analog computer for education – design and application, in *2021 Kleinheubach Conference* (2021) pp. 1–2.
[162] N. Shammah, S. Ahmed, S. Kaiser, A. Panigrahi, and tiagob01, Make your code count (2023).