Advances in Quantum Computation and Quantum Technologies: A Design Automation Perspective

Giovanni De Micheli, *Life Fellow, IEEE*, Jie-Hong R. Jiang, *Member, IEEE*, Robert Rand, Kaitlin Smith, *Member, IEEE*, and Mathias Soeken, *Member, IEEE*

***Abstract*— Universal and fault-tolerant quantum computation is a promising new paradigm that may efficiently conquer difficult computation tasks beyond the reach of classical computation. It motivates the development of various quantum technologies. The rapid progress of quantum technologies accelerates the realization of quantum computers. In this paper, we survey the recent advances in quantum technologies and quantum computation from the design automation perspective.**

***Index Terms*— Design automation, quantum computation, quantum technology.**

I. Introduction

IN MOORE'S era during the previous half-century, the exponential growth of capacities of computing systems has sustained the semiconductor and information technology industries in transforming our daily lives. As the downscaling of transistor sizes approaches the physical limit at the atomic level, Moore's law, which predicted the number of transistors on a chip doubles every 18 to 24 months, is no longer valid. Nevertheless, the end of Moore's era and the demand for big data processing of intelligent systems gave birth to innovative design and technology solutions. For computing devices, more Moore (CMOS) and more than Moore (beyond CMOS) alternatives are under intensive research. For computing systems, new architectures (beyond the conventional von Neumann architecture) and new computation paradigms (beyond the classical Turing machine model) are under extensive exploration.

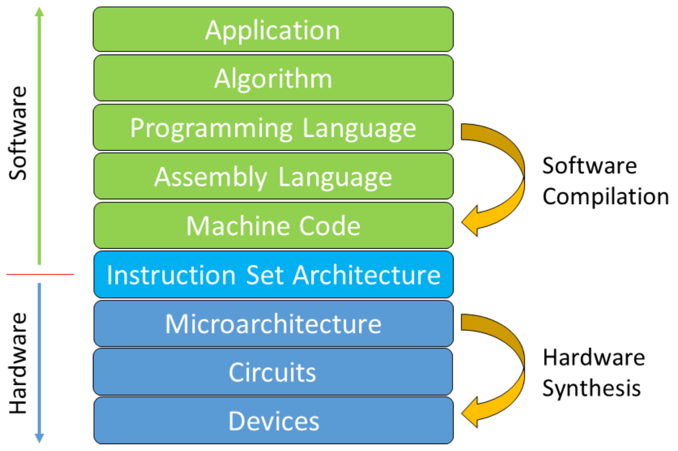

Fig. 1. Abstraction stack of classical/quantum computation.

Quantum computation (QC) is an essential new computing paradigm in post-Moore information technology. The extraordinary quantum properties, primarily superposition and entanglement, offer computation and information processing power beyond the reach of classical computers. Quantum algorithms of various sorts for computations of number factorization [1], solution search [2], quantum simulation [3], annealing [4], linear and differential equation solving [5], [6], etc., are available. They provide provable speed up over classical computation for various applications such as cryptanalysis, constraint solving, combinatorial optimization, and machine learning. Even though quantum computers are not expected to replace classical computers entirely, they are crucial technologies for certain computation accelerations. Consequently, the promise of these quantum algorithms has primarily motivated the extensive developments of quantum hardware and quantum software.

The abstraction stack of quantum computation is similar to that of classical computation as shown in Fig. 1. A more detailed and quantum-specific view on architecture is proposed in [7]. For quantum hardware, both general-purpose quantum processors, e.g., [8], and special-purpose quantum processors, e.g., [9], are under active development. The former follows the

Manuscript received 9 August 2022; accepted 29 August 2022. Date of publication 8 September 2022; date of current version 10 October 2022. The work of Giovanni De Micheli was supported by the Swiss National Science Foundation (SNSF) through the "Supercool: Design Methods and Tools for Superconducting Electronics" under Grant 200021-1920981. The work of Jie-Hong R. Jiang was supported by the Ministry of Science and Technology of Taiwan under Grant MOST 110-2224-E-002-011 and Grant MOST 111-2119-M-002-012. The work of Robert Rand was supported in part by the Enabling Practical-scale Quantum Computing (EPiQC), an NSF Expedition in Computing, under Grant CCF-1730449 and in part by the Air Force Office of Scientific Research under Grant FA95502110051. The work of Kaitlin Smith was supported in part by IBM as a Post-Doctoral Scholar at the University of Chicago and the Chicago Quantum Exchange; in part by the EPiQC, an NSF Expedition in Computing, under Award CCF-1730449; in part by the Software-Tailored Architectures for Quantum co-design (STAQ) under Award NSF Phy-1818914; in part by NSF under Award 2110860; in part by the U.S. Department of Energy Office of Advanced Scientific Computing Research, Accelerated Research for Quantum Computing Program; in part by the NSF Quantum Leap Challenge Institute for Hybrid Quantum Architectures and Networks under Award 2016136; and in part by the U.S. Department of Energy, Office of Science, National Quantum Information Science Research Centers. This article was recommended by Associate Editor-in-Chief C. Ho. *(Corresponding author: Jie-Hong R. Jiang.)*

Giovanni De Micheli is with the Institute of Electrical Engineering and Computer Science, EPFL, 1015 Lausanne, Switzerland (e-mail: giovanni.demicheli@epfl.ch).

Jie-Hong R. Jiang is with the Department of Electrical Engineering, Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 10617, Taiwan (e-mail: jhjiang@ntu.edu.tw).

Robert Rand and Kaitlin Smith are with the Department of Computer Science, The University of Chicago, Chicago, IL 60637 USA (e-mail: rand@uchicago.edu; kns@uchicago.edu).

Mathias Soeken is with Microsoft, 8058 Zürich, Switzerland (e-mail: mathias.soeken@outlook.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JETCAS.2022.3205174.

Digital Object Identifier 10.1109/JETCAS.2022.3205174

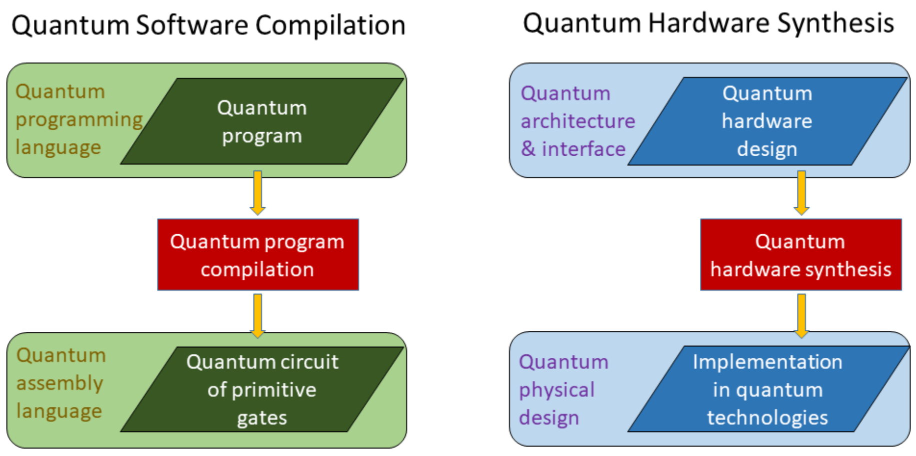
Quantum Software Compilation
Quantum programming language
Quantum program
Quantum program compilation
Quantum assembly language
Quantum circuit of primitive gates
Quantum Hardware Synthesis
Quantum architecture & interface
Quantum hardware design
Quantum hardware synthesis
Quantum physical design
Implementation in quantum technologies

Fig. 2. Software compilation and hardware synthesis for quantum computation.

unitary-gate-based quantum circuit computation model [10], [11], and the latter follows the Ising-model-like annealing procedure [4]. Recent advancements in quantum computer realization have demonstrated quantum advantage [8], showing the advantage of quantum computation over classical computation in a physical experiment. While building large-scale fault-tolerant universal quantum computation remains a long-term challenge, quantum computers based on the noisy intermediate-scale quantum (NISQ) technology [12] are available and exploited in different applications [13], for example, notably, quantum simulations for chemical systems [14], material science [15] and drug discovery [16]. Various physical implementations of quantum processors based on superconducting quantum devices [17], [18], semiconductor quantum dots [19], trapped ions [20], photonics [21], etc., have been proposed and demonstrated. Each competing technology has its advantages and disadvantages. To fully explore the potentials of various realization means, circuits and systems need to be built based on different technologies. Also, circuits and systems interfacing quantum and classical data processing are crucial for scalable quantum computation [22], [23].

For quantum software, a full stack of software engineering is indispensable to release the full power of quantum computing. To date, quantum computation programming languages [24], operating systems [25], compilers [26], [27], [28], [29], [30], and application programs [31] are emerging. Quantum software engineering requires domain knowledge at different abstraction levels. In particular, compiling a quantum algorithm or application into a format executable on a quantum processor requires transforming a high-level programming language code into a low-level quantum assembly code, which consists of a sequence of unitary operations represented as quantum circuits. The compilation requires high-level, logic-gate-level, and physical-level synthesis of design automation techniques. Also, the design of quantum algorithms and circuits has to be verified for correctness. Formal verification [32], simulation [33], and emulation [34] are vital, especially because quantum computers are intrinsically probabilistic and noisy. Many conventional electronic design automation (EDA) techniques for integrated circuit design can be applied and extended for quantum circuit compilation [35], [36].

In addition to the hardware and software aspects of quantum computing, there are important threads of developments in quantum-inspired computing systems and the hybrid quantum-classical computation. Even though fault-tolerant quantum computers are not yet ready, the concept of quantum computation itself has triggered innovative solutions that overcome conventional computation barriers. There are quantum-inspired architectures [37], [38] and algorithms [39] that improve classical computers and algorithms. Classical and quantum computations together may mutually fertilize each other and further advance our knowledge and practice of computation.

Engineering the computation systems largely requires various backgrounds in physics, computer science, electrical engineering, among others. This survey intends to summarize some key developments of design automation methods for quantum hardware and software engineering and provide helpful guides for the readers' further exploration.

The scope of this paper covers the aspects of quantum software compilation and quantum hardware synthesis, as shown in Fig. 2. We primarily focus on general-purpose quantum computation and leave out the subject of quantum annealing. The rest of this paper is organized as follows. Section II first provides the background of quantum processor architecture. Given the instruction set architecture (ISA), i.e., primitive quantum gates, provided by a quantum processor architecture, quantum software compilation can be carried out. The literature on quantum software compilation is then surveyed in Section IV. Then Section V reviews quantum technologies and design automation methods for quantum design. Finally, Section VI concludes this survey.

II. Quantum Processor Architecture

QCs, with their particular device physics, will require specific instruction sets, organization, and hardware to be developed into useful systems that successfully harness the power of quantum mechanics. Some examples of emerging qubit technology include discrete energy levels within superconducting circuits [8], [40], [41], ions trapped by surrounding electrodes [42], [43], [44], neutral atoms secured with optical tweezers [45], [46], and photons travelling through free space or waveguides [47], [48]. Each of these platforms have their unique strengths and methods for implementing logical computation, but none has become the obvious choice for the standard quantum computing platform. All of the aforementioned systems, however, must account for several similar architectural constraints in order to apply quantum superposition, interference, and entanglement for significant computational speedups in solving select problems.

A. No-Cloning of Qubits

Although basis states, or qubits in the fixed state of either $|0\rangle$ or $|1\rangle$ can be duplicated, unknown qubit superposition cannot be copied. This fundamental quantum principle is referred to as the "no cloning theorem" [49], and it has serious implications on the way quantum information is to be processed and stored. In classical computation, the ability to copy bits is frequently exploited within computation, by memory, and for error correction. In quantum software and hardware, however, we loose the ability to replicate data because observing, or measuring, quantum state causes its collapse into classical information. As a result, the design of quantum memory must differ significantly from classical memory hierarchies – quantum data must be actively moved to and from stored and cannot be recovered once read. Additionally, quantum error correction relies on actively applying state transformations on unknown qubit states for stabilization rather than relying on techniques such as either refresh cycles or with copies that perform a majority vote among copies of bits like in classical schemes.

B. Probabilistic Measurement

A key feature that defines qubits from classical bits is their ability to hold superpositions of states. Upon measurement, superpositions collapse into classical states: $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ collapses to either 0 or 1. A single measurement of the final state transformed by a quantum circuit cannot provide adequate insight about the circuit's true output. A more complete picture of the circuit's probabilistic quantum state output must be extracted through multiple measurements that develop a distribution. From the statistics associated with this distribution, quantum state can be inferred. Developing complete state information is possible through quantum tomography, but $O(2^{2n})$ measurements are required for complete state estimation. This amount of measurement might be prohibitive for practical QC use. An efficient quantum program will ensure that the correct outcome has a higher likelihood of being observed through subroutines such as amplitude amplification. However, the probabilistic nature of quantum information does not completely rule out the chance of seeing an incorrect output. Other external factors, such as system noise or errors on state preparation, gate evolution, and measurement, can also result in an undesired circuit result. Thus, many runs, or shots, of a quantum algorithm may be required in order to build statistical confidence in a result. Depending on the algorithm, the number of required circuit shots can be in the thousands.

C. Physical Realization of Quantum Information

The principles of quantum informatics are well defined, but means to physically realize qubits and quantum operations are necessary to make the exciting theoretical promise of quantum computation a reality. According to Di Vincenzo, specific criteria are required for physical quantum architectures to host quantum computation [50]. These conditions include containing well characterized qubits that can be initialized into known states, long coherence times for holding quantum information, the ability to realize quantum gates and measurement, and the ability to interact stationary, or compute, qubits with flying qubits, or those used for communication. Currently, a variety of quantum technologies can encode logical qubits within different media. We call a physical implementation of a radix-2 unit of information a physical qubit. Since today's quantum devices lack error correction, each logical qubit within an algorithm is implemented with one physical qubit in a machine. These NISQ devices are error prone and are up to hundreds of qubits in size [12].

D. Dependence on Classical Processing

For the foreseeable future, quantum resources will be hybrid devices that will require some amount of classical co-processing. For example, in the near-term, variational algorithms [51], [52] will require classical subroutines for optimizing quantum algorithm parameters and moving closer to desired solutions. In the fault-tolerant regime, classical programs will be required to implement the supporting mathematics required for applications such as Shor's quantum factoring [1]. In addition, classical processing will be required for quantum algorithm compilation and optimization as well as for low-level device control. Because of a QCs dependence on classical logic, quantum architecture must be designed to allow for seamless integration with classical infrastructure hosts the quantum processing unit.

E. Multi-Qubit Interaction

To leverage entanglement within algorithms, a feature that provides many quantum algorithms with advantage, qubits must have the ability to communicate and interact. Therefore, quantum state must be able to be mobile by either physically relocating qubits or through transferring quantum state through intermediate qubits if physical qubits are in fixed locations. Additionally, in the cases where large-scale entanglement is required, complex, multi-qubit operations will need to be physically realized by operators that are native to a specific quantum machine. In NISQ computing, qubit-qubit communication is often limited to nearest-neighbors. An important consideration in near-term quantum devices and beyond will be reducing communication overheads associated with multi-qubit interactions.

F. Qubit Sensitivity

Classical programs running on classical hardware infrequently worry about failures in data storage or logic operations. Additionally, failures are rarely injected from the environment unless the classical device is subject to extreme conditions. A significant amount of development has allowed classical computers to be robust to errors. However, qubits are extremely sensitive to external noise and currently suffer from high error rates that corrupt computation. As an attempt to limit the amount of external environmental influence on today's QCs, many implementations require that the QC operate near absolute-zero temperatures. Despite external conditions that introduce minimal heat to the quantum system, quantum systems still suffer from detrimental as a result of imperfect control [53], device defects [54], [55], and cross talk [56]. This noise stemming from imprecision at the physical level causes operational errors that cause gates performed on qubits to diverge from their intended logic and retention errors that limit the duration of time that qubits can hold state information [57].

III. Quantum Programming Languages

Quantum programming languages can be roughly divided into two categories: Languages designed for programming the computers of today (like those operated by IBM, Google, IonQ and Rigetti computing) and those designed for future devices with error-corrected qubits that can implement sophisticated algorithms like Shor's algorithm. We will start with the first class of languages:

A. Near-Term Languages

The best known tool for quantum programming is almost certainly QISKIT, a set of Python libraries designed for programming IBM's quantum computers. QISKIT generates circuits written in OPENQASM, a family of *Quantum Assembly Languages* executable on IBM's hardware. The original QASM was designed for simply printing out quantum circuits, and its successors are similarly low-level. They describe circuits over a small set of gates, including the controlled-NOT (CNOT), U_1, U_2, and U_3, where each of the U_i gates takes in i real numbered values. U_3 can implement any single qubit rotation (U_1 and U_2 are redundant in principle but are included for efficiency's sake) and hence together with CNOT makes up a universal set for quantum computation. For a detailed description of these gates and their functionality we refer the reader to common introductory material, e.g., [58]. OPENQASM 2.0, the standard output for QISKIT, is quite limited in terms of describing anything beyond simple circuits. Measuring a qubit results in a bit that can then be used to control a subsequent gate, however any stronger notion of classical control flow is absent from the language.

Google's CIRQ, like QISKIT, is a Python library designed for constructing quantum circuits. CIRQ has an additional focus on timing, allowing the user to specify which gates can run in parallel and succession. In fact, CIRQ is structured around time windows called Moments, in which a certain group of operations is scheduled. Otherwise, it looks very much like QISKIT, save that there is no separate language for circuit generation and circuit execution, a common quantum programming paradigm.

The third near-term quantum programming language of note is Rigetti's PYQUIL which, like QISKIT, consists of both a low-level circuit language (the "Quantum Instruction Language" QUIL) and a Python library for constructing circuits. Like QISKIT and OPENQASM, QUIL consists of a series of quantum gates while PYQUIL exists to generate QUIL circuits, but QUIL generalizes this model slightly: QUIL provides a "Delay" function, allowing for classical computation while QUIL awaits further instruction. However, this feature, like QUIL's capacity to measure a qubit mid-compilation, isn't supported by Rigetti's machines.

Some of QUIL's forward looking features blur the gap between near-term and long-term languages. The most recent version of OPENQASM, OPENQASM 3.0 [59], does the same. Unlike prior versions of OPENQASM, this new version allows the user to describe families of circuits, parameterized by a classical input not just concrete circuits. At the same time, drawing from CIRQ, it allows users to specify timing information on the gate level. It also allows the user to run classical computation on the results of measurements in order to compute the remainder of the circuit. Such techniques are important for quantum routines like Repeat-Until-Success or variational quantum algorithms, even though they cannot be run on IBM's current machines. In this way, OPENQASM 3.0 more closely resembles the long-term quantum programming languages that we will discuss next, even as it includes features tailored for near-term devices.

B. Long-Term Languages

A variety of languages have been developed for long term quantum computing. A pioneering language in this family is QUIPPER [26], a functional circuit-generation language aimed at evaluating the cost of executing advanced quantum algorithms, including a list of seven diverse and challenging algorithms proposed by IARPA. To do this, QUIPPER provides functionality for generating quantum circuit from Haskell code, bidirectional communication between classical and quantum devices, controlling and reversing of circuits, and automatic uncomputation of ancilla qubits. It can also calculate the resource costs of generated circuits, giving us a sense of the path towards long term quantum algorithms.

The SCAFFOLD language [60] was also developed to explore this path, though with a focus on optimizing circuits to make these complex algorithms more tractable [61]. SCAFFOLD is an imperative language modelled after C and aims to be an quantum algorithm description language, rather than simply a circuit generation language. It allows for the allocation of qubit registers as first class objects and for applying gates and other operations to these arrays. (Qubits, in SCAFFOLD's model, are simply qubit registers of length 1.) Like QUIPPER, SCAFFOLD provides a compiler from classical expressions to quantum ones, allowing for the easy generation of large quantum circuits that behave classically on their inputs.

Along with QISKIT, Microsoft's high-level Q# language [62] has one of the most developed libraries and active developer communities among quantum programming languages. Like SCAFFOLD, it moves away from the circuit model, instead allowing for quantum *operations* that affect the quantum state and pure classical *functions* that leave the quantum state alone. Unlike the many embedded quantum languages, Q# is self-contained and features a syntax and type system reminiscent of C# (although Q# is not a .NET language unlike C#). Q# facilitates simulation, by including certain features like an assert statement that can only be checked in simulation. To get code from Q# and other high-level language running on quantum devices, one can leverage intermediate representations such as QIR [63], which is based on LLVM.

C. Safety Guarantees

There are a wide variety of ways in which quantum programs can go wrong, including trying to copy (or "clone") a qubit, sequencing quantum circuits with different numbers of wires, and discarding ancilla qubits without unentangling them and/or returning them to the $|0\rangle$ state. Some of the earliest quantum programming languages were developed to avoid these kinds of errors, with Selinger's QPL [64] employing a syntactic check that the two arguments to a two-qubit gate were distinct.

The most common approach to avoiding violations of the no-cloning theorem is through using *linear types*, which guarantee that linear data is used precisely once. The quantum lambda calculus [65] uses a variant of this system called *affine types*, which ensure that data is used at most once, allowing for free deletion of qubits. PROTO-QUIPPER, an idealized, self-contained core of the QUIPPER language, similarly uses linearity to prevent cloning or deletion or qubits, patching a key potential source of error in QUIPPER itself.

While linear types are used to prevent cloning, *dependent types*, in which types can depend on language-level terms, ensure that the composition of circuits is well-formed. The QWIRE language [66], embedded in the Coq proof assistant. combines both linear and dependent types to allow for the safe composition of circuits, often drawn from parameterized circuit families. In a similar vein, PROTO-QUIPPER-D [67] combines linear and dependent types, though it avoids the abstraction of separate host and circuit language. Even more recently, Chimaera [68], takes advantage of the linear and dependent types in the Idris 2 language to obtain a quantum programming language that gets no cloning and safe composition for free.

The safe management of ancilla was first addressed by REVERC, a tool for safely compiling reversible circuits, using uncomputation to discard bits. Drawing on REVERC, QWIRE was augmented with a limited range of templates for uncomputing and discarding ancilla, which were proved correct in the Coq proof assistant. The SILQ language [69] was built around safe and automatic uncomputation via `qfree` and `const` annotations that guarantee that a function behaves classically and that a given qubit isn't modified until it can be safely uncomputed. SILQ uses similar annotation to guarantee that functions can be reverse or controlled, by promising that they are measurement-free. In terms of uncomputation, the Unqomp language [70] builds upon SILQ by ensuring that uncomputation isn't merely safe but efficient.

These are merely some of the key safety guarantees made by recent quantum programming languages. Others include guaranteeing that qubits are separable from the rest of the state, as in the Twist programming language [71], through a mixture of static annotations and dynamic assertions and checks. $\lambda_{Q\#}$ [72], a proposed formal core for Q#, uses singleton types to guarantee that multiple aliases of a qubit aren't passed to the same operation, while enforcing stack discipline. PROTO-QUIPPER-DYN [73] uses type annotations to ensure that a quantum function doesn't use dynamic lifting to request a continuation from the classical computer, ensuring that the function can be safely treated not simply as a function, but as a circuit. Novel approach to safety guarantees in quantum programming language continue to appear in the literature, demonstrating how quantum computing continues to benefit from such techniques.

IV. QUANTUM PROGRAM COMPILERS

In this section, we survey various quantum compilation algorithms. We can think of the trace of a quantum program as a *high-level quantum circuit* over *n primary qubits* and a sequence of *high-level quantum gates* that operate on a subset of these *n* qubits. Therefore, depending on the control flow of a program, there may exist several different quantum traces for one program, each may result in a different qubit allocation. Examples for high-level operations are the addition of two quantum registers or a quantum Fourier transform. Note that the description of high-level quantum gates may vary. Operations can be described explicitly in terms of unitary matrices or symbolically, e.g. by a Boolean function that describes a permutation or a phase change of the state's amplitudes. The goal is to execute the trace on a physical quantum computer that is characterized by number of available qubits, a target gate set of available single qubit and two qubit gate operations, and a coupling graph that describes which pairs of qubits allow the execution of two qubit gates. The synthesis step in a quantum compiler takes care of translating the high-level quantum gates to allowed operations in the target gate set, the mapping step assigns the qubits of the input trace to physical qubits and possibly changes the quantum circuit such that the constraints of the coupling map can be fulfilled.

A. Synthesis

1) Hierarchical Synthesis: Hierarchical synthesis methods [74], [75] address the translation of high-level quantum gates that implement *Boolean oracle unitaries*

$$U_f : |x\rangle|y\rangle|0\rangle^{\otimes k} \mapsto |x\rangle|y \oplus f(x)\rangle|0\rangle^{\otimes k}, \qquad (1)$$

where $|x\rangle$ and $|y\rangle$ are quantum registers of length m and l respectively, and $f : \{0,1\}^m \to \{0,1\}^l$ is a Boolean function with m inputs and l outputs. When the Boolean function is represented in terms of a hierarchical logic network composed of r gates, one can readily find quantum circuits that represent U_f with $k \leq r$ helper qubits computing temporary values onto the helper qubits using simpler quantum operations that correspond to the logic gates in the logic network. Researchers have proposed LUT (lookup table) networks [76] to limit the number of inputs to each logic gate. Various techniques to map such oracle unitaries for functions with a small number of inputs have been proposed, e.g., [77], [78], [79]. Other instantiations of hierarchical logic synthesis methods utilize XOR-AND-Inverter graphs (XAGs) and found that the number of helper qubits can be bounded by the number of AND gates in the XAG [80]. This enables the application of logic synthesis methods to the logic networks in order to reduce the number of AND gates. Several algorithms to reduce the so-called multiplicative complexity [81] have been proposed [82], [83], [84], which besides impacting synthesis in quantum program compilers, also are highly relevant in cryptography [82].

Further trade-offs between the number of quantum operations and qubits can be achieved by varying the ways in which and how often temporary operations are computed and uncomputed. These algorithms can be unified as instances of reversible pebble games [85], for which various algorithms have been proposed, both for arbitrary [86], [87] and specific graph topologies [88], [89], [90].

2) Phase Polynomial Synthesis: Another family of unitaries are phase polynomials. A *phase polynomial representation* [91] for a unitary $2^n \times 2^n$ matrix U over n qubits is a tuple $(A, (\theta_1, f_1), \dots, (\theta_l, f_l))$, where $A \in \mathrm{GL}_n(\{0, 1\})$ is a linear Boolean matrix, θ_i are real-valued angles, and f_i are linear Boolean functions over n variables. A phase polynomial representation describes the matrix

$$U : |x\rangle \mapsto \left(\prod_{i=1}^{l} e^{\mathrm{i}\theta_i f_i(x)}\right)|Ax\rangle. \tag{2}$$

All quantum circuits that are composed of CNOT and $R_1(\theta) = \mathrm{diag}(1, e^{\mathrm{i}\theta})$ gates implement phase polynomials, and more importantly, all phase polynomials can be resented by a quantum circuit on n qubits using only CNOT and $R_1(\theta)$ gates. Note that this quantum gate set becomes universal by including the Hadamard gate. If all rotation angles are multiples of $\frac{\pi}{4}$, the quantum circuits correspond to CNOT+T circuits, a subset of the Clifford+T gate set. In [91], the CNOT+T gate set was considered for re-synthesis and T-depth optimization in Clifford+T circuits. One phase polynomial was constructed from maximal CNOT+T subcircuits and then re-synthesized in a way that minimizes T-depth based on matrioid partitioning. The work in [92] highlights the fact that there exist multiple phase polynomials for one unitary and use it for an algorithm to reduce T-count based on Reed-Muller decoders. In [93] a heuristic algorithm is presented that also reduces the number of CNOT gates in a CNOT+R_1 circuit, by not increasing the number of rotation gates. A SAT-based version of the algorithm that finds the minimum number of CNOT gates is presented in [94], and variants that preserve gate coupling constraints [95], [96], [97]. Other correspondences between polynomials and families of quantum circuits are reported in [98] and [99].

3) Relative-Phase Implementations: Synthesis methods, in particular those targeting Boolean oracle unitaries, make use of multiple-controlled Toffoli gates, an X operation on a target qubit controlled on an arbitrary number of control qubits, as intermediate gate set. We can diagonalize a multiple-controlled Toffoli gate, by surrounding the target qubit with Hadamard gates to replace the controlled X operation with a controlled Z operation. A multiple-controlled Z operation can be represented as a phase polynomial and therefore algorithms from the previous section can be applied. Smaller circuits can be found by using helper qubits and breaking down a gate with many control qubits into smaller ones with less control qubits [10]. Some initial constructions where provided in this reference, which were improved in [100] by substituting intermediate Toffoli gates by relative-phase implementations thereof, which require fewer gates from a target gate set such as Clifford+T [101]. More savings are possible when synthesizing several multiple-controlled Toffoli gates at once [102]. A dedicated construction for a three-controlled Toffoli gate is reported in [103]. Relative-gates have also been applied to reduce the cost of arithmetic operations [104], [105] or arbitrary quantum circuit optimization algorithms [106]. Some circuits make use of so-called measurement-based uncomputation [104], which can significantly reduce the cost of the quantum circuit for the cost of mid-circuit measurements. Such techniques are also useful for generalizations of the multiple-controlled Toffoli gate, e.g., table lookup [107]. The work in [108] analyzes a generalization that includes relative-phase implementations as special instances, and uses it to find improved quantum circuits.

4) Unitary Synthesis: The synthesis of arbitrary unitary matrices often starts from a decomposition into CNOT gates and arbitrary single-qubit unitary matrices U [10]. Various constructive algorithms have been proposed, e.g., based on cosine-sine matrix decomposition [109], quantum multiplexors [110], decomposition of isometries [111], or meet-in-the-middle algorithms [112], [113].

Then, single-qubit gates in the resulting circuits can be described by three rotations using Euler decomposition $U = e^{\mathrm{i}\phi} R_z(\theta_1) R_x(\theta_2) R_z(\theta_3)$, up to a global phase $e^{\mathrm{i}\phi}$. This is equal to $e^{\mathrm{i}\phi} R_z(\theta_1) H R_z(\theta_2) H R_z(\theta_3)$, and therefore the problem can be reduced to rotation synthesis of $R_z(\theta)$ operations. Since the target gate set is usually finite, the rotation $R_z(\theta)$ may not be representable using gates drawn from the target gate set. Therefore, in a first step one tries to find an approximation R for $R_z(\theta)$ such that R can be represented using gates from the target gate set and such that $\|R - R_z(\theta)\| \leq \varepsilon$ for some $\varepsilon > 0$. This step is called single-qubit unitary approximation [114]. First constructive algorithms were found by [115], [116], and [117], optimal algorithms shortly after by [118], [119], and [120] for specific gate sets and with generalizations of gate sets in [121]. Further improvements were achieved by allowing measurements in the circuits [122], [123]. The state of the art in single-qubit unitary synthesis reduces the problem to a magnitude approximation problem [124]. After an approximation was found, exact synthesis techniques [125], [126], [127] can be used to map them to the respective gate set.

B. Compilation

1) Mapping to NISQ Devices: Near-term QCs have specific architectural constraints that must be considered during compilation. NISQ devices lack error correction, so each logical qubit within an algorithm is allocated to one physical qubit on a quantum processor. These physical qubits are often limited in their ability to interact directly with other qubits since as most NISQ hardware only supports nearest-neighbor coupling. As a result, the primary goal of a compiler when mapping quantum programs to near-term machines is to optimize for communication: it is preferred to determine mappings that minimize the movement of qubit state during program runtime since computation must be completed within restrictive qubit coherence windows.

Initially, linear nearest neighbor constraints on a 1D line graph or a 2D grid were considered [128], [129], [130], [131]. With the emerge of coupling graphs from real devices, a method for arbitrary graphs based on A* search was proposed [132]. Significant improvements were found by applying multiple rounds of mapping to find a good initial qubit placement [133], which was further analyzed in [134]. Other approaches are based on transformation and commutation rules [135]. SAT- and SMT-based techniques can be applied to find optimum solutions and have first been proposed in [136] and later improved (in run-time) in [137] and [138]. Other approaches suggest to apply a two-step approach in which the circuit is partitioned into maximal subcircuits for which a qubit placement can be found without changing the circuit. In a second step, SWAP networks are generated to map from one qubit placement to another between subsequent subcircuits in the partition, also called qubit routing. A SAT-based algorithm to find qubit placements for a partition was proposed in [139]. It was shown that qubit placement can be mapped to routings via matching and token swapping problems [140] depending on whether one is interested in a quantum circuit with low SWAP depth or low SWAP count, respectively. Various algorithms to solve the token swapping have been proposed in [141]. In most of these algorithms, each qubit on the target hardware was considered the same, however, the quality of individual qubits and qubit pair interactions can vary. Noise-aware mapping algorithms [142], [143] take this into account to find not only mappings with a low gate count overhead, but also those that favor qubits and qubit interactions with higher quality to reduce effects of noise. Additional examples of quantum compilation frameworks that boost the performance of algorithms on near-term machines via architecture-aware optimization include [144], [145], [146]

2) Mapping to Surface Code: Fault-tolerant quantum computing controls the noise of physical qubit operations by encoding multiple physical qubits into logical qubits through error correction. The surface code [147], [148] is a promising approach to implement such a fault-tolerant quantum computing scheme. A universal set of quantum computing operations can be implemented on top of the surface code, e.g., through lattice surgery [149]. The lower-level surface code gate set often makes use of joint-measurement operations as multi-qubit operations, and applies non-Clifford gates by means of magic state distillation [150] and injection protocols [266]. Joint-measurement operations can be used to implement long-rang SWAP or teleportation operations [151], which enables mapping algorithms that are favorable to SWAP-based mapping algorithms. In [152], an algorithm is proposed that transforms a quantum circuit into a sequence of high-weight Pauli measurement. In [153] proposes a mapping algorithm based on long-range operations and optimizes their scheduling by means of edge-disjoint path compilation.

C. Simulation

Simulation is an essential way to investigate the behavior of a quantum system. However, it is computationally challenging due to the exponential growth of quantum states in the number of quantum bits (qubits) as well as the complex domains in characterizing quantum states and operations. This difficulty motivates Richard Feynman's proposal to build a quantum simulator/computer to simulate a quantum system, rather than using a classical computer.

A quantum computation task proceeds in three steps: initial state preparation, state evolution, and measurement. In gate-based quantum computation, the state evolution is modeled by a sequence of unitary operators that update the state vector. A simulator aims to predict the measurement outcomes. Classical simulation of quantum circuits can be classified into *strong simulation* and *weak simulation*. The former aims to calculate the probabilities of the measurement outcomes with high accuracy; the latter aims to obtain output samples from the probability distribution [154]. Most simulation algorithms focus on strong simulation to calculate the probability amplitudes of one or more quantum states. In weak simulation, a classical computer mimics a quantum computer in sampling the measurement outcomes.

There are two approaches to compute probability amplitudes, namely, Schrödinger's and Feynman's approaches [8]. The former is based on state evolution by updating the state vector gate by gate. The latter is based on path integral by summing over the contributions of probability amplitudes of all paths in the configuration space.

In contrast to Schrödinger's approach computing all probability amplitudes, Feynman's can be much more memory-efficient in computing the probability amplitude of a single quantum state of interest. Tensor-network-based simulation algorithms, e.g., [155], [156], [157], allows the computation complexity not to grow in the number of paths, which can be exponential in the number of qubits and gates of the quantum circuit, but rather in the *tree-width* of the circuit graph. Essentially, the tensor *contraction* operation plays the role of path integral in summing contributing amplitudes. When the graph of a quantum circuit is closer to a tree, i.e., having a smaller value of tree-width, the circuit exhibits less quantum characteristics and can be simulated more easily by classical computers. As noted in [158], Schrödinger's approach works relatively well for quantum circuits with the number of qubits sufficiently limited for a full state vector storable in memory. In contrast, Feynman's approach works relatively well for shallow quantum circuits with a large number of qubits.

Modern quantum circuit simulators follow one of the two approaches or explore their combination. Depending on the underlying data structures, quantum circuit simulation algorithms may vary in their implementation and applicability. In the following, we review some representative implementation choices.

1) Array-Based Methods: One natural choice of data structures for quantum circuit simulation computation is using arrays to store the operator matrix and state vector and support their multiplication in Schrödinger's approach and to store tensors and support their contraction in Feynman's approach.

Simulation by array-based matrix-vector multiplication is commonly available in quantum compilation and simulation tools, e.g., CIRQ [159], QISKIT [160], QUEST [161], QULACS [158]. Although matrix-vector multiplication is well supported by parallel computations on CPUs and GPUs, the full-amplitude computation is hardly scalable to circuits with 50 qbuits even with supercomputing facilities. On the other hand, simulation by array-based tensor-network contraction has been developed and available in tools, e.g., QUIMB [162], QULACS [158]. As the method allows partial and single amplitude computation, it may reduce the memory requirement and allow simulation on larger circuits. As was demonstrated in [163], random quantum circuits with 40, 75 and 200 qubits can be simulated for the computation of full, partial and single amplitude, respectively.

2) Decision-Diagram-Based Methods: Binary decision diagrams (BDDs), particularly the reduced ordered BDDs (ROBDDs) [164], have been widely used as a compact canonical representation for Boolean function manipulation. Efficient BDD packages, e.g., [165], are available and widely applied in various applications, such as electronic design automation and formal verification. The success has motivated their extension, e.g., [166], [167], [168], [169], to quantum circuit simulation to overcome the memory explosion problem of array-based methods. The quantum multiple-valued decision diagram (QMDD) [170] is one of such attempts to represent and manipulate operator matrices and state vectors of complex values. The QMDD-based simulation [168] is shown more effective than the array-based simulation in [171] in cases where quantum states can be compactly represented by QMDDs. QMDDs have been applied to confirm that program semantics are preserved during compilation [172].

Most array-based and QMDD-based methods rely on floating-point numbers to represent complex values and may suffer from the precision problem [173]. The problem is overcome in [173] and [169], where a complex number is represented algebraically using five integer coefficients. In [173], the floating-point numbers in a QMDD are replaced with algebraic complex numbers at the cost of computation overhead due to more expensive arithmetic operations. In contrast, in [169] the algebraic integer coefficients are represented with an ROBDD per bit of an integer coefficient. Simulating an n-qubit quantum circuit corresponds to manipulating n-variable ROBDDs according to a set of pre-characterized Boolean formulas corresponding to the multiplication effects of the supported set of unitary operators. It simulates certain benchmarks up to tens of thousands qubits, and is generally much more scalable than QMDD-based methods. Although [169] supports only algebraically representable unitary gates, the gate collection is already sufficient for universal quantum computation.

The above decision-diagram-based methods follow Schrödinger's approach for quantum state evolution. Nevertheless, decision diagrams can also be applied in Feynman's approach for tensor-network-based computation [174], where a QMDD is relaxed to allow the input and output variables of a qubit to be freely, rather than adjacently, ordered and the tensor contraction operation on decision diagrams is supported.

3) Stabilizer-Rank Methods: There are classes of quantum circuits whose simulation on a classical computer takes time polynomial in the circuit size. One well-known class is the *stabilizer circuits*, which merely consist of Clifford gates. Any Clifford gate can be generated using the CNOT, Hadamard, and phase gates. Algorithms for efficient simulation of stabilizer circuits has been proposed [175], [176].

As universal quantum computation can be achieved by Clifford gates added with some non-Clifford primitive gates, e.g., the T-gate, or with the magic state, the stabilizer circuit simulation algorithm can be extended to detail with circuits with both Clifford and non-Clifford gates [175]. Unsurprisingly, the time complexity grows polynomial in the number of Clifford gates and exponential in the number of non-Clifford gates. There are recent efforts [177], [178], [179] that attempt to alleviate the exponential growth through the notion of *stabilizer rank* [178] by decomposing a state into the superposition of a number of stabilizer states [177].

D. Verification

1) Equivalence Checking: In modern integrated circuit (IC) design flow, equivalence checking plays an important role in ensuring the synthesis steps do not introduce errors. Similar verification requirements are needed in the compilation of quantum programs because the process of quantum program compilation corresponds to a sequence of quantum circuit transformation. In the compilation process, the circuits before and after synthesis have to conform to some equivalence criteria. The strongest and most common notion of equivalence is *total equivalence*, which requires the output quantum states of the two quantum circuits under verification have to be the same modulo a global phase difference.

It is clear that a quantum circuit simulator capable of strong simulation and computing all amplitudes can be used for checking total equivalence. Decision-diagram-based simulation algorithms are advantageous in equivalence checking over array-based counterparts in their canonicity in state vector representation. This canonicity makes equivalence checking easy without having to check individual equivalences of corresponding amplitude pairs, which can be exponential in the number of qubits.

Equivalence checking of quantum circuits have been studied extensively, e.g., [180], [181], [182], [183]. A commonly adopted approach to checking the equivalence between two circuits $U = U_n \cdots U_1$ and $V = V_m \cdots V_1$, for unitary operators U_i and V_j, is to build the *miter circuit* [181], namely, $M = U \cdot V^{-1} = (U_n \cdots U_1) \cdot (V_1^{-1} \cdots V_m^{-1})$. The two circuits are equivalence if and only if $M = e^{\imath\theta} I$, that is, M equals an identify operator up to some global phase $e^{\imath\theta}$. The equivalence checking boils down to computing the multiplication $(U_n \cdots U_1) \cdot (V_1^{-1} \cdots V_m^{-1})$. In fact, the sequence of multiplying out the matrices is flexible and can be exploited to keep decision diagrams of small size. In [182], the multiplication sequence is rewritten as $U_n \cdots U_1 \cdot I \cdot V_1^1 \cdots V_m^\dagger$. The procedure of [182] starts from the identity matrix I in the middle of the sequence and gradually multiplies either to the left or to the right under some strategies to keep the QMDD compact throughout the multiplication process. However, the above matrix multiplication suffers from the floating-point precision problem, which may lead to incorrect answers to equivalence checking. The precision problem of equivalence checking is overcome in [183], which extends the ROBDD-based algebraic representation of state vectors [169] to unitary operators. Besides equivalence checking, fidelity checking and sparsity checking are considered in [183]. Essentially, fidelity checking offers a quantitative way to tell how similar two circuits are if they are not equivalent. The ROBDD-based method [183] can check certain circuits up to thousands of qubits, and is generally more scalable and robust than QMDD-based method [182].

In contrast to the above equivalence checking methods, which follow the Schrödinger's approach, a path-sum formulation following Feynman's approaches is proposed in [32] for quantum circuit verification, including equivalence checking. However, its scalability can be limited to circuits up to 100 qubits.

There are recent efforts that extend the notion of equivalence. Specifically, equivalence checking of noisy quantum circuits is addressed in [184] and [183]. In [185], partial equivalence checking is defined and realized under the ROBDD-based framework [183] to allow observational equivalence with respect to partial measurement and constrained initial states. In [186], equivalence checking of sequential quantum circuits is formulated. For dynamic quantum circuits, where quantum systems interact classical controls, their equivalence checking is considered in [187] and [188].

2) Proof Assistants: Formal verification of quantum programs aims to prove that a program does what it's expected to do. Two of the earliest works in this area used the Coq proof assistant [189] to prove the correctness of basic quantum algorithms, whether expressed directly as a series of quantum operations [190] or within the QWIRE quantum programming language [191]. However, the first approach was limited by a slow underlying matrix library, while the second was limited by the complexity of the QWIRE language [66], and neither was able to verify complex algorithms like Shor's factoring algorithm or Grover's search.

One approach to the complexity of verifying quantum computation was to represent them as *path-sums* [32], which describe a unitary circuits action on basis states as

$$|x\rangle \rightarrow \frac{1}{\sqrt{2^m}} \sum_{y=0}^{2^m-1} e^{2\pi i P(x,y)/2^m} |f(x, y)\rangle$$

where P and f are drawn from a restricted class of functions. Conveniently, most common quantum gates can be represented using path-sums: For instance, the common X and T gates can be written

$$X : |x\rangle \rightarrow e^0 |x \oplus 1\rangle \qquad T : |x\rangle \rightarrow e^{2\pi i \frac{x}{8}} |x\rangle$$

Path-sums can also be composed neatly, allowing for easy processing of a quantum program. Using path-sums, the FEYNMAN tool [32] was able to verify the correctness of a quantum Fourier transform on up to 31 qubits and a hidden-shift algorithm on up to 60, both in a matter of seconds. However, as we might expect, this method scales poorly, typically stalling out in the hundreds of qubits (depending on the algorithm).

In order to reason about quantum programs over arbitrary numbers of qubits (*circuit families* if we're considering the quantum circuit model), we have to do symbolic reasoning, where the number of qubits n appears in the verification statement and proof. QBRICKS [192] does this by generalizing FEYNMAN's path-sums to parameterized path-sums, where parameters to the program can also appear in the proof statement. Using this generalization of path-sums and the verification tool WHY3 [193], along with a range of SMT solvers, QBRICKS was able to verify a range of quantum programs, importantly including both Grover's and (the quantum part of) Shor's algorithms, regardless of input size.

Concurrently with QBRICKS, a different group of researchers developed SQIR [194], a small quantum intermediate representation designed for ease of proof. SQIR provides it's programs semantics both in terms of unitary matrices and superoperators, for programs including measurement. While less automated than QBRICKS, it does provide some automation for simplifying matrix and complex number expressions, and switching between representations of states and operators. SQIR was originally used to prove the quantum parts of Grover's and Shor's algorithms, with later work presenting an end-to-end proof of Shor's algorithm, including the classical parts [195]. It also served as the core of the VOQC compiler [30], which heavily optimizes quantum programs while guaranteeing that the output program is semantically identical to the input.

3) Program Logics: Another, partially overlapping, approach to validating quantum programs is using a program logic in the model of Hoare logic or guarded command language. The first work in this direction was D'Hondt and Panangaden's Quantum Weakest Preconditions [196], which proposed that quantum observables are the correct quantum analogue to Kozen's arithmetic predicates [197] for reasoning about probabilistic programs. Ying [198] used these predicates to write a Hoare-style logic for reasoning about programs in a *quantum while language*, a simple simple imperative language with quantum bits and integers, unitary application, and if and while statement, each of which measure quantum states. Ying's paper included logics for both partial and total correctness, where the former treats non-terminating programs as satisfying the postcondition. In the case of total correctness, the logic includes a rule for while loops that guarantees that the loop terminates with probability 1. The logic, later called QHL, was sufficient to prove the correctness of Grover's algorithm.

A variety of works built on QHL, including extending it to reason about both quantum and classical variables. At the same time, several works moved away from using observables as preconditions, arguing that it's often easier to reason using projectors as predicates over quantum states. These predicates were adopted in an *applied quantum Hoare logic* [199], which added rules for reasoning about programs with error bounds, as we expect for all near-term quantum programs. This logic was used to prove the correctness of the Harrow-Hassidim-Lloyd algorithm [5]. Simultaneously, Unruh [200] developed a projection-based *quantum Hoare logic with ghost variables*, where non-program variables called ghosts can stand in for measured variables in predicates. This allows one to easily express notions like "q is the outcome of a coin flip" by saying "q can be modelled as a member of a measured Bell pair with the ghost g". This logic is used for cryptographic purposes, particularly to verify a quantum one-time pad.

Quantum Hoare logic has been extended to a variety of additional domains including reasoning about program robustness [201], parallel programming [202], separation logic [203], [204], and an EasyCrypt-style relational logic for proving security [205]. The last is of particular interest as it led to qrhl-tool, a dedicated cryptographic proof assistant built upon Isabelle/HOL [206]. QHL itself was also embedded in Isabelle/HOL [207], providing the correctness guarantees of a proof assistant along with the convenience of a Hoare style logic.

A fuller discussion of formal verification in the quantum setting is given in [208].

V. Quantum Technology in Computing Systems

In this section, we provide an overview how superconducting quantum technology is applied in computing systems that process either quantum or classical information.

A. Superconducting Devices for Quantum Computing

Pinpointing the technology that will enable large-scale quantum computation is currently an active area of research. In the last decade, however, superconducting (SC) quantum circuits [209] have amassed popularity from industry and academia alike as a means for creating physical qubits. A leading advantage of SC circuits is that they take advantage of well-established classical fabrication techniques and thus are poised to be favorable for scaling.

SC qubits utilize Josephson Junctions (JJs), or two superconducting electrodes separated by a thin, insulating tunnel barrier, along with additional capacitors and inductors to implement circuit quantum electrodynamics (cQED). At a high-level, cQED describes how microwave photons and SC circuits interact, and it can be used to define the rules for the realization of quantum information. SC circuits are highly configurable. For example, the properties, dimensions, and number of JJs within an SC circuit influences how the resulting qubit stores and manipulates quantum state. SC cQED devices operate at close to absolute zero temperatures, 20 mK, in order to isolate operational modes from external environmental noise – they act as mesoscopic-scale, artificial atoms with an anharmonic energy spectrum [210], [211]. Typically, the lowest two energy levels of this spectrum are used to realize qubit states: the ground state $|0\rangle$ corresponds to the lowest energy level and the excited state $|1\rangle$ corresponds to the next highest energy level.

In this section, we will focus our study on the SC qubit referred to as the transmon. We note, however, that other types of SC quantum devices based on the JJ exist such as the fluxonium [212] and flux qubit [213]. Transmons were originally developed in 2007 [214] and are the qubits found in the majority of today's cQED devices. The first two-qubit interaction of fidelity higher than 99% was demonstrated in 2014 [215], and there have been many more qubit-qubit couplings of similar quality reported since [41], [216], [217], [218]. Single-qubit operations can be implemented at higher fidelity with error rates of order 10^{-4}. Broadly, transmon qubits are categorized as 'fixed-frequency' with a single JJ or 'flux-tunable' with two parallel JJs that allow qubit frequency to be adjusted with the application of an external magnetic field. The frequency of a transmon qubit corresponds to the difference between the $|1\rangle$ and $|0\rangle$ energy levels in the transmon's anharmonic spectrum.

Emerging transmons QCs are promising but are still NISQ-era prototypes with under 150 qubits [219], [220], [221]. They also suffer from non-trivial noise during computation stemming from limited coherence windows (currently in the range of 10s of microseconds) [222], measurement errors [223], and imprecise control that results from incomplete system characterization [224]. Since transmon devices are not error corrected, quantum programs must be transformed into highly-customized executables to both optimize for native gate sets and mitigate platform-specific noise on near-term QCs [142]. Although much progress has been made in terms of boosting two-qubit gate fidelity, one of the of the biggest challenges facing today's transmon qubits is obtaining qubit-qubit interaction fidelities that satisfy thresholds required for fault tolerance. Flux-tunable quantum devices enable two-qubit interactions via dynamically tuning qubits into resonance conditions [225], [226] or via parametric modulation of tunable elements [227], [228]. Tunability of SC qubit circuits, however, comes at the cost of qubit coherence and implementation scalibility due to the footprint of the necessary classical control hardware. Conversely, two-qubit interactions are enabled by carefully designed microwave drives in fixed-frequency SC qubit devices [216]. The qubit frequencies of nearest and next-nearest neighbors must be allocated with adequate spacing to avoid frequency collisions and poor qubit interactions in fixed frequency devices [229]. Unfortunately, imprecision associated with today's QC fabrication causes small imperfections to appear in JJ positioning, component dimensions, and surrounding layers, influencing operational characteristics of transmons during computation [230]. In the case of the fixed-frequency transmons, fabrication imprecision that results in component imperfections often causes qubit frequency to deviate from its ideal, resulting in spectral overlaps that induce frequency collisions. Despite challenges associated with frequency collisions, however, fixed-frequency transmons are characterized by recent improvements in coherence, stability, and controlability [231], [232], [233], [234] that making them promising for scaling.

B. Superconducting Electronics in Classical Pipelined Computing

Quantum computing leverages two important properties at the device level: superposition and entanglement. Other technologies and architectures can exploit device quantum properties different that superposition and entanglement, and thus realize classical (as compared to quantum) computing with specific important characteristics. Indeed, *superconducting electronic* (SCE) circuits provides us with the ability of realizing digital logic gates where the information is quantized. Logic gates can be designed so that a single quantum of information is exchanged. Moreover, information transmission is very rapid on wires, because of the lack of parasitic resistance in the superconducting domain, typically at few degrees Kelvin (typically 4 K). Switching is achieved by *Josephson junctions* (JJ), that are designed by interposing a thin insulator in a gap of a superconducting wire of Niobium.

Superconducting electronics are quite attractive for the following reasons. First, the technology can match and extend present performance requirements, e.g., ALU prototypes have been shown to run at and above 50 GHz clock rates. Second, SCE devices manipulate flux quanta $\o = h/2e$ with energy $2 \cdot 10^{-19} J$ or $5 \cdot 10^3\ kT\ ln\ 2$ at 4 K. Thus, SCE circuits can be realized to operate much closer to the minimum energy limit, and roughly two orders of magnitude better as compared to CMOS. This prediction is confirmed by prototypes [235]. Third, these circuits can work as interface at 4 K between the host and a quantum computer, enabling information processing in a classic way that is in local proximity to a quantum chip. Thus, superconducting electronic circuits can be employed as both standalone accelerators and as bridges to quantum computing. For this last reason, an expanded description is justified here.

IBM led a strong effort in SCE in the 70s with the objective of building computers that would outperform the currently-available technology. The circuits utilized Josephson junctions exhibiting hysteresis in their resistive states (i.e., resistive and superconductive). The JJ acts as a switch that can be set and reset by applying a current. A logic TRUE is associated with the JJ in its resistive state, and a logic FALSE with its superconductive state. This effort faded in the mid 80s, because of various drawbacks, including the choice of materials and the latching operation of logic [236].

Likharev [236] brought back strong interest in SCE by proposing rapid single flux quantum (RSFQ) circuits. In these circuits, the logic values (TRUE, FALSE) are represented by the presence or absence of single flux quantum (SFQ) pulses called fluxons with $\o = h/2e = 2 \cdot 10^{-15} Wb$ corresponding approximately to a 2 mv pulse lasting 1 ps. Junctions are DC biased and when a pulse is applied to the junction, it can be sufficient to drive the current level over its threshold and to generate another pulse that can be propagated through the circuit. This type of behavior is often called *Josephson transmission line* (JTL) and it is the basic operational principle of RSFQ circuits that propagates flux pulses. A specific feature of RSFQ circuits is that logic gates are clocked, and that the overall circuit is pipelined. The RSFQ technology evolved in many directions. Energy-efficient SFQ (eSFQ and ERFSQ) [237] and low-voltage RSFQ (LV-RSFQ) [238] employ specific bias networks and low supply voltages respectively to reduce the power consumption. Dynamic flux single quantum (DSFQ) logic [239] introduces self-resetting gates that ease the clocking requirements. Various realizations of ALUs have been reported, with deep-pipelined, wave-pipelined and asynchronous operation.

SCE circuit design has several peculiarities and constraints, that may vary in the different SCE families. We highlight two constraints. First, each gate is triggered by a clock or bias signal in conjunction with the logic input signal. Thus, circuits operate in pipelined mode, and input to logic gates have to be present simultaneously, thus requiring that logic inputs have the same logic depth or distance from the primary inputs. A circuit with such a property is said to be balanced. Second, logic gates generate pulses that cannot sustain multiple fanouts, and thus splitters have to be used. As a result, SCE design requires specific electronic design tools. The Coldflux project [240], under the auspices of the IARPA Supertools program, has addressed design electronic design automation (EDA) problems for SCE, including automatic circuit balancing [241], [242] and splitter insertion [243]. Some researchers addressed the splitter and buffer insertion in AQFP [244], [245], [246] while others considered a flow where the logic network is reduced first for depth using algebraic methods, followed by Boolean substitution and splitter insertion [247]. Lee presented an exact formulation of buffer/splitter insertion via SMT as well as an improved heuristic algorithm [248]. Researchers at Synopsys recently published the results of a full synthesis of a 4-bit AMD 2901 microcontroller from RTL code to layout in an ERFSQ standard cell library from Hypres [249]. Several research activities have addressed physical design of SCE circuits, such as synchronization [250], placement and routing [250], [251], [252], cell libraries [252], and parasitics extraction and mitigation [253]. Krylov and Friedman [254] have recently authored a comprehensive book on various aspect of SCE design with a wide set of references to current works.

C. Adiabatic Superconducting Electronics

Recent research work has addressed technologies that target low-energy consumption. This can be achieved by using adiabatic mode of operation and AC power (i.e., alternating current supply). Two technologies are particularly relevant: reciprocal quantum logic (RQL) [235] researched and developed at Northrop Grumman, and adiabatic quantum flux parametron (AQFP) [255] pursued at Yokohama National University (YNU) in Japan. A parametron is a resonant circuit with a nonlinear reactive element [256]. We describe AQFP in more detail.

The fundamental element in AQFP is the clocked buffer. Two loops, involving each a JJ and an inductor, are used to store logic information in terms of flux quanta depending on the direction of an input current signal and the magnetic coupling to other inductors. When an input and the supply trigger are present, an output current pulse is generated. The direction of the current pulse encodes the logic value TRUE or FALSE. A buffer can be made into an inverting buffer by switching the terminals of the output coupled inductor. Thus, inversion comes at no cost in this technology. It was shown [257], [258] that the "parallel combination" of three AQFP buffers yields a majority gate, which is the basic logic primitive of this technology. The 2-input logic AND and OR gates can be realized by modifying one buffer (of the majority gate) so that a small imbalance in the loop design yields always a logic FALSE or TRUE as output respectively. Based on these principles, a simple and modular cell library can be built from buffers (regular, inverted or modified) and branch cells (used to join and split signals) [257]. A full EDA flow from an HDL description to a cell-based physical design has been created by researchers at YNU [259]. In particular, tools for logic synthesis need to address balancing, majority-based synthesis (described in the next section) and splitter insertion [245], [246], [247].

The majority paradigm in logic synthesis [260], [261] is based on a formulation of a new Boolean algebra using the majority and complementation operators. The algebra can be expressed in terms of five axioms (commutativity, associativity, distributivity, majority and self-duality) and was shown to be sound and complete [261]. From a theoretical standpoint, algorithms based on the majority paradigm enable the search for an optimum or optimal solution in a connected design space, which provides the existence of a path to the optimum (even though such path may be hard to find and with over-polynomial length, as the problem is computationally intractable). From a practical standpoint, tools based on the majority algebra were shown to achieve circuits 15% better in delay in average as compared to other methods after physical design in ASICs [261]. This fact was also validated on commercial tools. Libraries of algorithms for logic optimization are publicly available, such as the mockturtle library (https://github.com/lsils/mockturtle).

D. Wave Pipelining

Wave pipelining (WP) [262] is a technique to speed up the computation by allowing two or more waves of signals to propagate in between two registers. In a WP circuit the clock frequency of the registers can be higher than the maximum propagation delay, to capture wave-fronts of data as they propagate from the source to the sink register. It is quintessential that the waves do not mix, which implies that I/O paths need to have the same delay, or to mismatch by a small quantity that eventually poses a bound on the clock frequency. There are examples of RSFQ ALU designs that exploit wave pipelining [263]. Whereas in standard (synchronous) RSFQ the clock triggers the computation at logic gates, in asynchronous wave-pipelined RSFQ signals are held so that a logic stage does not start operating until all signals from the previous stage are available. This obviates the local clocking [264] and enables multiple data waves to propagate simultaneously. As the overall performance is limited by the signal arrival-time mismatches, then SCE can benefit from path delay equalization as in CMOS WP [262] and furthermore WP path balancing can be combined with majority logic synthesis transformations, to achieve correct and optimal SCE digital circuits.

E. Summary

Recent realizations of SCE circuits have shown remarkable performances. For example, Ke [265] showed the realization of a low-power 8-point, 7-bit FFT processor running at 47.8 GHz consuming 5.3mW in SFQ technology. An AQFP adiabatic processor has been realized [244] with switching energy at 1.4 zJ with a 5 GHz AC clock. Even by considering a 1000x energy loss in cryocooling, this realization is still two-orders of magnitude more efficient as compared to 7nm CMOS according to [244]. These very positive results make us very optimistic about the potentials of SCE as a superconducting technology, especially for low-energy high-throughput computation. Nevertheless, scaling up SCE design is challenging, as the support of EDA tools is still in its infancy.

VI. Conclusion

In this survey paper, we covered recent advances in quantum computation and quantum technologies from the design automation perspective. Due to the rapid progress and diversified interdisciplinary studies, it is not possible to mention all important related work of the intended subject. However, we tried to provide a skeleton of some key elements in the abstraction stack of quantum computation sketched in Fig. 1 based on our limited knowledge. We hope this survey can serve as a helpful guide for the readers to find entry points for further investigations.

Acknowledgment

This paper was produced by the IEEE Publication Technology Group, Piscataway, NJ, USA. The authors are grateful to the anonymous reviewers for valuable comments.

References

[1] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM J. Comput.*, vol. 26, no. 5, pp. 1484–1509, 1997, doi: 10.1137/S0097539795293172.

[2] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proc. 28th Annu. ACM Symp. Theory Comput. (STOC)*, 1996, pp. 212–219, doi: 10.1145/237814.237866.

[3] S. Lloyd, "Universal quantum simulators," *Science*, vol. 273, no. 5278, pp. 1073–1078, Aug. 1996, doi: 10.1126/science.273.5278.1073.

[4] T. Kadowaki and H. Nishimori, "Quantum annealing in the transverse Ising model," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 58, no. 5, pp. 5355–5363, Nov. 1998, doi: 10.1103/physreve.58.5355.

[5] A. W. Harrow, A. Hassidim, and S. Lloyd, "Quantum algorithm for linear systems of equations," *Phys. Rev. Lett.*, vol. 103, no. 15, Oct. 2009, Art. no. 150502.

[6] D. W. Berry, "High-order quantum algorithm for solving linear differential equations," *J. Phys. A, Math. Gen.*, vol. 47, no. 10, Feb. 2014, Art. no. 105301, doi: 10.1088/1751-8113/47/10/105301.

[7] N. C. Jones *et al.*, "Layered architecture for quantum computing," *Phys. Rev. X*, vol. 2, no. 3, Jul. 2012, Art. no. 031007, doi: 10.1103/physrevx.2.031007.

[8] F. Arute *et al.*, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, 2019.

[9] M. W. Johnson *et al.*, "Quantum annealing with manufactured spins," *Nature*, vol. 473, no. 7346, p. 127, 2011, doi: 10.1038/nature10012.

[10] A. Barenco *et al.*, "Elementary gates for quantum computation," *Phys. Rev. A, Gen. Phys.*, vol. 52, no. 5, p. 3457, Nov. 1995.

[11] E. T. Campbell, B. M. Terhal, and C. Vuillot, "Roads towards fault-tolerant universal quantum computation," *Nature*, vol. 549, no. 7671, pp. 172–179, Sep. 2017, doi: 10.1038/nature23460.

[12] J. Preskill, "Quantum computing in the NISQ era and beyond," 2018, *arXiv:1801.00862*.

[13] A. D. Córcoles *et al.*, "Challenges and opportunities of near-term quantum computing systems," *Proc. IEEE*, vol. 108, no. 8, pp. 1338–1352, Aug. 2020, doi: 10.1109/JPROC.2019.2954005.

[14] A. Kandala *et al.*, "Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets," 2017, *arXiv:1704.05018*.

[15] H. Ma, M. Govoni, and G. Galli, "Quantum simulations of materials on near-term quantum computers," *npj Comput. Mater.*, vol. 6, no. 1, pp. 1–8, Jul. 2020, doi: 10.1038/s41524-020-00353-z.

[16] Y. Cao, J. Romero, and A. Aspuru-Guzik, "Potential of quantum computing for drug discovery," *IBM J. Res. Develop.*, vol. 62, no. 6, pp. 6:1–6:20, Nov. 2018.

[17] P. Krantz, M. Kjaergaard, F. Yan, T. P. Orlando, S. Gustavsson, and W. D. Oliver, "A quantum engineer's guide to superconducting qubits," *Appl. Phys. Rev.*, vol. 6, no. 2, Jun. 2019, Art. no. 021318, doi: 10.1063/1.5089550.

[18] M. Kjaergaard *et al.*, "Superconducting qubits: Current state of play," *Annu. Rev. Condens. Matter Phys.*, vol. 11, no. 1, pp. 369–395, Mar. 2020, doi: 10.1146/annurev-conmatphys-031119-050605.

[19] L. M. K. Vandersypen and M. A. Eriksson, "Quantum computing with semiconductor spins," *Phys. Today*, vol. 72, no. 8, pp. 38–45, Aug. 2019, doi: 10.1063/PT.3.4270.

[20] I. Pogorelov *et al.*, "Compact ion-trap quantum computing demonstrator," *PRX Quantum*, vol. 2, no. 2, Jun. 2021, Art. no. 020343, doi: 10.1103/prxquantum.2.020343.

[21] M. Gong *et al.*, "Quantum walks on a programmable two-dimensional 62-qubit superconducting processor," *Science*, vol. 372, no. 6545, pp. 948–952, May 2021, doi: 10.1126/science.abg7812.

[22] J. C. Bardin *et al.*, "Design and characterization of a 28-nm bulk-CMOS cryogenic quantum controller dissipating less than 2 mW at 3 K," *IEEE J. Solid-State Circuits*, vol. 54, no. 11, pp. 3043–3060, Nov. 2019.

[23] F. Jazaeri, A. Beckers, A. Tajalli, and J.-M. Sallese, "A review on quantum computing: From qubits to front-end electronics and cryogenic MOSFET physics," in *Proc. 26th Int. Conf. 'Mixed Design Integr. Circuits Syst.'*, Jun. 2019, pp. 15–25.

[24] B. Heim *et al.*, "Quantum programming languages," *Nature Rev. Phys.*, vol. 2, no. 12, pp. 709–722, Dec. 2020, doi: 10.1038/s42254-020-00245-7.

[25] H. Corrigan-Gibbs, D. J. Wu, and D. Boneh, "Quantum operating systems," in *Proc. 16th Workshop Hot Topics Operating Syst.*, May 2017, pp. 7–81.

[26] A. S. Green, P. L. Lumsdaine, N. J. Ross, P. Selinger, and B. Valiron, "Quipper: A scalable quantum programming language," in *Proc. ACM SIGPLAN Conf. Program. Lang. Design Implement.*, 2013, pp. 333–342, doi: 10.1145/2462156.2462177.

[27] A. JavadiAbhari *et al.*, "ScaffCC: A framework for compilation and analysis of quantum computing programs," in *Proc. 11th ACM Conf. Comput. Frontiers*, May 2014, pp. 1:1–1:10.

[28] M. Amy and V. Gheorghiu, "staq—A full-stack quantum processing toolkit," *Quantum Sci. Technol.*, vol. 5, no. 3, Jun. 2020, Art. no. 034016, doi: 10.1088/2058-9565/ab9359.

[29] S. Sivarajah, S. Dilkes, A. Cowtan, W. Simmons, A. Edgington, and R. Duncan, "t|keta⟩: A retargetable compiler for NISQ devices," *Quantum Sci. Technol.*, vol. 6, no. 1, Nov. 2020, Art. no. 014003.

[30] K. Hietala, R. Rand, S.-H. Hung, X. Wu, and M. Hicks, "A verified optimizer for quantum circuits," *Proc. ACM Program. Lang.*, vol. 5, p. 37, Jan. 2021. [Online]. Available: https://github.com/inQWIRE/SQIR

[31] J. R. McClean *et al.*, "OpenFermion: The electronic structure package for quantum computers," *Quantum Sci. Technol.*, vol. 5, no. 3, Jun. 2020, Art. no. 034014.

[32] M. Amy, "Towards large-scale functional verification of universal quantum circuits," in *Proc. 15th Int. Conf. Quantum Phys. Log. (QPL)*, in Electronic Proceedings in Theoretical Computer Science, vol. 287, P. Selinger and G. Chiribella, Eds. Waterloo, NSW, Australia: Open Publishing Association, Jan. 2019, pp. 1–21.

[33] T. Häner and D. S. Steiger, "0.5 petabyte simulation of a 45-qubit quantum circuit," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, Nov. 2017, pp. 33:1–33:10.

[34] T. Häner, D. S. Steiger, K. Svore, and M. Troyer, "A software methodology for compiling quantum programs," 2016, *arXiv:1604.01401*.

[35] M. Soeken, T. Häner, and M. Roetteler, "Programming quantum computers using design automation," in *Proc. Design, Automat. Test Eur. Conf. Exhib. (DATE)*, Mar. 2018, pp. 137–146, doi: 10.23919/DATE.2018.8341993.

[36] M. Soeken, M. Roetteler, N. Wiebe, and G. D. Micheli, "LUT-based hierarchical reversible logic synthesis," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 38, no. 9, pp. 1675–1688, Sep. 2019, doi: 10.1109/TCAD.2018.2859251.

[37] S. Matsubara *et al.*, "Digital annealer for high-speed solving of combinatorial optimization problems and its applications," in *Proc. 25th Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2020, pp. 667–672.

[38] T. Takemoto *et al.*, "A 144 Kb annealing system composed of 9×16 Kb annealing processor chips with scalable chip-to-chip connections for large-scale combinatorial optimization problems," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2021, pp. 64–66.

[39] M. Aramon, G. Rosenberg, E. Valiante, T. Miyazawa, H. Tamura, and H. G. Katzgraber, "Physics-inspired optimization for quadratic unconstrained problems using a digital annealer," *Frontiers Phys.*, vol. 7, p. 48, Apr. 2019, doi: 10.3389/fphy.2019.00048.

[40] V. Havlíček *et al.*, "Supervised learning with quantum-enhanced feature spaces," *Nature*, vol. 567, no. 7747, pp. 209–212, 2019.

[41] P. Jurcevic *et al.*, "Demonstration of quantum volume 64 on a superconducting quantum computing system," *Quantum Sci. Technol.*, vol. 6, no. 2, Apr. 2021, Art. no. 025020.

[42] Y. Nam *et al.*, "Ground-state energy estimation of the water molecule on a trapped-ion quantum computer," *npj Quantum Inf.*, vol. 6, no. 1, pp. 1–6, Dec. 2020.

[43] K. Wright *et al.*, "Benchmarking an 11-qubit quantum computer," *Nature Commun.*, vol. 10, no. 1, pp. 1–6, 2019.

[44] J. Zhang *et al.*, "Observation of a many-body dynamical phase transition with a 53-qubit quantum simulator," *Nature*, vol. 551, no. 7682, pp. 601–604, Nov. 2017.

[45] H. Levine *et al.*, "Parallel implementation of high-fidelity multiqubit gates with neutral atoms," *Phys. Rev. Lett.*, vol. 123, no. 17, Oct. 2019, Art. no. 170503.

[46] H. Bernien *et al.*, "Probing many-body dynamics on a 51-atom quantum simulator," *Nature*, vol. 551, no. 7682, pp. 579–584, Nov. 2017.

[47] J. L. O'Brien, A. Furusawa, and J. Vučković, "Photonic quantum technologies," *Nature Photon.*, vol. 3, no. 12, pp. 687–695, 2009.

[48] E. Knill, R. Laflamme, and G. J. Milburn, "A scheme for efficient quantum computation with linear optics," *Nature*, vol. 409, no. 6816, pp. 46–52, Jan. 2001.

[49] W. K. Wootters and W. H. Zurek, "A single quantum cannot be cloned," *Nature*, vol. 299, no. 5886, pp. 802–803, Oct. 1982.

[50] D. P. DiVincenzo, "The physical implementation of quantum computation," *Fortschritte Phys.*, vol. 48, nos. 9–11, pp. 771–783, Feb. 2000.

[51] A. Peruzzo *et al.*, "A variational eigenvalue solver on a photonic quantum processor," *Nature Commun.*, vol. 5, no. 1, pp. 1–7, 2014.

[52] N. Moll *et al.*, "Quantum optimization using variational algorithms on near-term quantum devices," *Quantum Sci. Technol.*, vol. 3, no. 3, Jul. 2018, Art. no. 030503.

[53] J. J. Wallman and J. Emerson, "Noise tailoring for scalable quantum computation via randomized compiling," *Phys. Rev. A, Gen. Phys.*, vol. 94, no. 5, Nov. 2016, Art. no. 052325.

[54] J. B. Hertzberg *et al.*, "Laser-annealing Josephson junctions for yielding scaled-up superconducting quantum processors," *npj Quantum Inf.*, vol. 7, no. 1, pp. 1–8, Dec. 2021.

[55] A. Nersisyan *et al.*, "Manufacturing low dissipation superconducting quantum processors," in *IEDM Tech. Dig.*, Dec. 2019, pp. 1–31.

[56] M. Sarovar, T. Proctor, K. Rudinger, K. Young, E. Nielsen, and R. Blume-Kohout, "Detecting crosstalk errors in quantum information processors," *Quantum*, vol. 4, p. 321, Sep. 2020.

[57] S. S. Tannu and M. K. Qureshi, "Not all qubits are created equal: A case for variability-aware policies for NISQ-era quantum computers," in *Proc. 24th Int. Conf. Architectural Support Program. Lang. Operating Syst.*, Apr. 2019, pp. 987–999.

[58] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge, U.K.: Cambridge Univ. Press, 2000.

[59] A. Cross *et al.*, "OpenQASM 3: A broader and deeper quantum assembly language," *ACM Trans. Quantum Comput.*, vol. 3, no. 3, pp. 1–50, Sep. 2022. [Online]. Available: https://qiskit.github.io/openqasm

[60] A. Javadi-Abhari *et al.*, "Scaffold: Quantum programming language," Princeton Univ., Princeton, NJ, USA, Tech. Rep., TR-934-12, Jun. 2012. [Online]. Available: https://www.cs.princeton.edu/research/techreps/TR-934-12

[61] A. JavadiAbhari *et al.*, "ScaffCC: Scalable compilation and analysis of quantum programs," *Parallel Comput.*, vol. 45, pp. 2–17, Jun. 2015.

[62] K. M. Svore *et al.*, "Q#: Enabling scalable quantum computing and development with a high-level DSL," in *Proc. Real World Domain Specific Lang. Workshop*. New York, NY, USA: Association for Computing Machinery, Feb. 2018, p. 7.

[63] QIR Alliance. (2021). *QIR Specification*. [Online]. Available: https://qir-alliance.org

[64] P. Selinger, "Towards a quantum programming language," *Math. Struct. Comput. Sci.*, vol. 14, no. 4, pp. 527–586, Aug. 2004. [Online]. Available: https://www.mathstat.dal.ca/~selinger/papers/papers/qpl.pdf

[65] P. Selinger and B. Valiron, "A lambda calculus for quantum computation with classical control," *Math. Struct. Comput. Sci.*, vol. 16, no. 3, pp. 527–552, Jun. 2006. [Online]. Available: https://www.mscs.dal.ca/~selinger/papers/papers/qlambda-mscs.pdf

[66] J. Paykin, R. Rand, and S. Zdancewic, "QWIRE: A core language for quantum circuits," in *Proc. 44th ACM SIGPLAN Symp. Princ. Program. Lang.*, New York, NY, USA, Jan. 2017, pp. 846–858.

[67] P. Fu, K. Kishida, and P. Selinger, "Linear dependent type theory for quantum programming languages: Extended abstract," in *Proc. 35th Annu. ACM/IEEE Symp. Log. Comput. Sci.*, New York, NY, USA, Jul. 2020, pp. 440–453.

[68] L.-J. Dandy, E. Jeandel, and V. Zamdzhiev. (Nov. 2021). *Qimaera: Type-Safe (Variational) Quantum Programming in Idris*. [Online]. Available: https://github.com/zamdzhiev/Qimaera

[69] B. Bichsel, M. Baader, T. Gehr, and M. Vechev, "Silq: A high-level quantum language with safe uncomputation and intuitive semantics," in *Proc. 41st ACM SIGPLAN Conf. Program. Lang. Design Implement.*, New York, NY, USA, Jun. 2020, pp. 286–300. [Online]. Available: https://files.sri.inf.ethz.ch/website/papers/pldi20-silq.pdf

[70] A. Paradis, B. Bichsel, S. Steffen, and M. Vechev, "Unqomp: Synthesizing uncomputation in quantum circuits," in *Proc. 42nd ACM SIGPLAN Int. Conf. Program. Lang. Design Implement.*, New York, NY, USA, Jun. 2021, pp. 222–236.

[71] C. Yuan, C. McNally, and M. Carbin, "Twist: Sound reasoning for purity and entanglement in quantum programs," *Proc. ACM Program. Lang.*, vol. 6, p. 30, Jan. 2022. [Online]. Available: https://github.com/psg-mit/twist-popl22

[72] K. Singhal, K. Hietala, S. Marshall, and R. Rand, "Q# as a quantum algorithmic language," in *Proc. 19th Int. Conf. Quantum Phys. Log. (QPL)*. Oxford, U.K.: Open Publishing Association, Jun. 2022, pp. 1–22. [Online]. Available: https://ks.cs.uchicago.edu/publication/q-algol/

[73] P. Fu, K. Kishida, N. J. Ross, and P. Selinger, "Proto-Quipper with dynamic lifting," 2022, *arXiv:2204.13041*.

[74] M. Rawski, "Application of functional decomposition in synthesis of reversible circuits," in *Proc. Int. Conf. Reversible Comput.*, 2015, pp. 285–290, doi: 10.1007/978-3-319-20860-2_20.

[75] M. Soeken, M. Roetteler, N. Wiebe, and G. De Micheli, "Logic synthesis for quantum computing," 2017, *arXiv:1706.02721*.

[76] M. Soeken, M. Roetteler, N. Wiebe, and G. De Micheli, "Hierarchical reversible logic synthesis using LUTs," in *Proc. 54th Annu. Design Autom. Conf.*, Jun. 2017, pp. 78:1–78:6.

[77] K. Fazel, M. A. Thornton, and J. E. Rice, "ESOP-based Toffoli gate cascade generation," in *Proc. IEEE Pacific Rim Conf. Commun., Comput. Signal Process.*, Aug. 2007, pp. 206–209.

[78] C. Bandyopadhyay, H. Rahaman, and R. Drechsler, "Improved cube list based cube pairing approach for synthesis of ESOP based reversible logic," *Trans. Comput. Sci.*, vol. 24, pp. 129–146, 2014, doi: 10.1007/978-3-662-45711-5_8.

[79] G. Meuli, M. Soeken, M. Roetteler, and G. De Micheli, "Enumerating optimal quantum circuits using spectral classification," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Oct. 2020, pp. 1–5.

[80] G. Meuli, M. Soeken, E. Campbell, M. Roetteler, and G. De Micheli, "The role of multiplicative complexity in compiling low T-count Oracle circuits," in *Proc. Int. Conf. Comput.-Aided Design*, 2019, pp. 1–8, doi: 10.1109/ICCAD45719.2019.8942093.

[81] R. Mirwald and C. P. Schnorr, "The multiplicative complexity of quadratic Boolean functions," in *Proc. 28th Annu. Symp. Found. Comput. Sci.*, Oct. 1987, pp. 141–150, doi: 10.1109/SFCS.1987.57.

[82] J. Boyar, P. Matthews, and R. Peralta, "Logic minimization techniques with applications to cryptology," *J. Cryptol.*, vol. 26, no. 2, pp. 280–312, Apr. 2013, doi: 10.1007/s00145-012-9124-7.

[83] E. Testa, M. Soeken, L. Amarù, and G. De Micheli, "Reducing the multiplicative complexity in logic networks for cryptography and security applications," in *Proc. 56th Annu. Design Autom. Conf.*, Jun. 2019, p. 74, doi: 10.1145/3316781.3317893.

[84] E. Testa, M. Soeken, H. Riener, L. G. Amarù, and G. De Micheli, "A logic synthesis toolbox for reducing the multiplicative complexity in logic networks," in *Proc. Design, Automat. Test Eur. Conf. Exhib. (DATE)*, Mar. 2020, pp. 568–573.

[85] C. H. Bennett, "Time/space trade-offs for reversible computation," *SIAM J. Comput.*, vol. 18, no. 4, pp. 766–776, Aug. 1989, doi: 10.1137/0218053.

[86] A. Parent, M. Roetteler, and K. M. Svore, "REVS: A tool for space-optimized reversible circuit synthesis," in *Proc. Int. Conf. Reversible Comput.*, 2017, pp. 90–101, doi: 10.1007/978-3-319-59936-6_7.

[87] G. Meuli, M. Soeken, M. Roetteler, N. Bjorner, and G. D. Micheli, "Reversible pebbling game for quantum memory management," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2019, pp. 288–291, doi: 10.23919/DATE.2019.8715092.

[88] E. Knill, "An analysis of Bennett's pebble game," 1995, *arXiv:math/9508218*.

[89] S. M. Chan, "Just a pebble game," in *Proc. IEEE Conf. Comput. Complex.*, Jun. 2013, pp. 133–143, doi: 10.1109/CCC.2013.22.

[90] B. Komarath, J. Sarma, and S. Sawlani, "Reversible pebble game on trees," in *Proc. Int. Conf. Comput. Combinatorics*, 2015, pp. 83–94, doi: 10.1007/978-3-319-21398-9_7.

[91] M. Amy, D. Maslov, and M. Mosca, "Polynomial-time T-depth optimization of Clifford+T circuits via matroid partitioning," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 33, no. 10, pp. 1476–1489, Oct. 2014, doi: 10.1109/TCAD.2014.2341953.

[92] M. Amy and M. Mosca, "T-count optimization and Reed–Müller codes," 2016, *arXiv:1601.07363*.

[93] M. Amy, P. Azimzadeh, and M. Mosca, "On the CNOT-complexity of CNOT-PHASE circuits," 2017, *arXiv:1712.01859*.

[94] G. Meuli, M. Soeken, and G. De Micheli, "SAT-based {CNOT, T} quantum circuit synthesis," in *Proc. Int. Conf. Reversible Comput.*, 2018, pp. 175–188, doi: 10.1007/978-3-319-99498-7_12.

[95] A. Kissinger and A. Meijer-van de Griend, "CNOT circuit extraction for topologically-constrained quantum memories," 2019, *arXiv:1904.00633*.

[96] B. Nash, V. Gheorghiu, and M. Mosca, "Quantum circuit optimizations for NISQ architectures," 2019, *arXiv:1904.01972*.

[97] A. Meijer-van de Griend and R. Duncan, "Architecture-aware synthesis of phase polynomials for NISQ devices," 2020, *arXiv:2004.06052*.

[98] A. Montanaro, "Quantum circuits and low-degree polynomials over $\mathbb{F}_2$," *J. Phys. A, Math. Theor.*, vol. 50, no. 8, Jan. 2017, Art. no. 084002, doi: 10.1088/1751-8121/aa565f.

[99] C. M. Dawson, H. L. Haselgrove, A. P. Hines, D. Mortimer, M. A. Nielsen, and J. T. Osborne, "Quantum computing and polynomial equations over the finite field $\mathbb{Z}_2$," 2004, *arXiv:quant-ph/0408129*.

[100] D. Maslov, "Advantages of using relative-phase Toffoli gates with an application to multiple control Toffoli optimization," *Phys. Rev. A, Gen. Phys.*, vol. 93, no. 2, Feb. 2016, Art. no. 022311.

[101] C. Jones, "Low-overhead constructions for the fault-tolerant Toffoli gate," *Phys. Rev. A, Gen. Phys.*, vol. 87, no. 2, Feb. 2013, Art. no. 022328.

[102] S. Esaki and S. Yamashita, "Reducing T-count when decomposing many MPMCT gates simultaneously," in *Proc. 50th IEEE Int. Symp. Multiple-Valued Logic*, Miyazaki, Japan, Nov. 2020, pp. 22–27, doi: 10.1109/ISMVL49045.2020.00-35.

[103] C. Gidney and N. Cody Jones, "A CCCZ gate performed with 6 T gates," 2021, *arXiv:2106.11513*.

[104] C. Gidney, "Halving the cost of quantum addition," *Quantum*, vol. 2, p. 74, Jun. 2018, doi: 10.22331/q-2018-06-18-74.

[105] K. Oonishi, T. Tanaka, S. Uno, T. Satoh, R. Van Meter, and N. Kunihiro, "Efficient construction of a control modular adder on a carry-lookahead adder using relative-phase Toffoli gates," *IEEE Trans. Quantum Eng.*, vol. 3, pp. 1–18, 2022.

[106] S. Kuroda and S. Yamashita, "Optimization of quantum Boolean circuits by relative-phase Toffoli gates," in *Proc. Int. Conf. Reversible Comput.*, 2022, pp. 20–27, doi: 10.1007/978-3-031-09005-9_2.

[107] C. Gidney, "Windowed quantum arithmetic," 2019, *arXiv:1905.07682*.

[108] M. Amy and N. J. Ross, "Phase-state duality in reversible circuit design," *Phys. Rev. A, Gen. Phys.*, vol. 104, no. 5, Nov. 2021, Art. no. 052602, doi: 10.1103/PhysRevA.104.052602.

[109] M. Möttönen, J. J. Vartiainen, V. Bergholm, and M. M. Salomaa, "Quantum circuits for general multiqubit gates," *Phys. Rev. Lett.*, vol. 93, no. 13, Sep. 2004, Art. no. 130502, doi: 10.1103/physrevlett.93.130502.

[110] V. V. Shende, S. S. Bullock, and I. L. Markov, "Synthesis of quantum-logic circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 6, pp. 1000–1010, Jun. 2006, doi: 10.1109/TCAD.2005.855930.

[111] R. Iten, R. Colbeck, I. Kukuljan, J. Home, and M. Christandl, "Quantum circuits for isometries," *Phys. Rev. A, Gen. Phys.*, vol. 93, Mar. 2016, Art. no. 032318, doi: 10.1103/PhysRevA.93.032318.

[112] M. Amy, D. Maslov, M. Mosca, and M. Roetteler, "A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits," *IEEE Trans. Comput.-Aided Design Integr.*, vol. 32, no. 6, pp. 818–830, Jun. 2013, doi: 10.1109/TCAD.2013.2244643.

[113] O. D. Matteo and M. Mosca, "Parallelizing quantum circuit synthesis," *Quantum Sci. Technol.*, vol. 1, no. 1, Mar. 2016, Art. no. 015003.

[114] J. Bourgain and A. Gamburd, "A spectral gap theorem in $SU(d)$," 2011, *arXiv:1108.6264*.

[115] V. Kliuchnikov, D. Maslov, and M. Mosca, "Asymptotically optimal approximation of single qubit unitaries by Clifford and T circuits using a constant number of ancillary qubits," *Phys. Rev. Lett.*, vol. 110, no. 19, May 2013, Art. no. 190502, doi: 10.1103/PhysRevLett.110.190502.

[116] P. Selinger, "Efficient Clifford+T approximation of single-qubit operators," *Quantum Inf. Comput.*, vol. 15, no. 1, pp. 159–180, Jan. 2015.

[117] A. Bocharov, Y. Gurevich, and K. M. Svore, "Efficient decomposition of single-qubit gates into V basis circuits," *Phys. Rev. A, Gen. Phys.*, vol. 88, no. 1, Jul. 2013, Art. no. 012313.

[118] N. J. Ross and P. Selinger, "Optimal ancilla-free Clifford+T approximation of z-rotations," 2014, *arXiv:1403.2975*.

[119] N. J. Ross, "Optimal ancilla-free Clifford+V approximation of z-rotations," 2014, *arXiv:1409.4355*.

[120] A. Blass, A. Bocharov, and Y. Gurevich, "Optimal ancilla-free Pauli+V circuits for axial rotations," *J. Math. Phys.*, vol. 56, no. 12, Dec. 2015, Art. no. 122201, doi: 10.1063/1.4936990.

[121] V. Kliuchnikov, A. Bocharov, M. Roetteler, and J. Yard, "A framework for approximating qubit unitaries," 2015, *arXiv:1510.03888*.

[122] A. Paetznick and K. M. Svore, "Repeat-until-success: Non-deterministic decomposition of single-qubit unitaries," 2013, *arXiv:1311.1074*.

[123] A. Bocharov, M. Roetteler, and K. M. Svore, "Efficient synthesis of probabilistic quantum circuits with fallback," *Phys. Rev. A, Gen. Phys.*, vol. 91, no. 5, May 2015, Art. no. 052317, doi: 10.1103/PhysRevA.91.052317.

[124] V. Kliuchnikov, K. Lauter, R. Minko, A. Paetznick, and C. Petit, "Shorter quantum circuits," 2022, *arXiv:2203.10064*.

[125] V. Kliuchnikov and J. Yard, "A framework for exact synthesis," 2015, *arXiv:1504.04350*.

[126] M. Amy, A. N. Glaudell, and N. J. Ross, "Number-theoretic characterizations of some restricted Clifford+T circuits," *Quantum*, vol. 4, p. 252, Apr. 2020, doi: 10.22331/q-2020-04-06-252.

[127] T. Kalajdzievski and N. Quesada, "Exact and approximate continuous-variable gate decompositions," *Quantum*, vol. 5, p. 394, Feb. 2021, doi: 10.22331/q-2021-02-08-394.

[128] A. G. Fowler, S. J. Devitt, and L. C. L. Hollenberg, "Implementation of Shor's algorithm on a linear nearest neighbour qubit array," *Quantum Inf. Comput.*, vol. 4, no. 4, pp. 237–251, Jul. 2004.

[129] M. Saeedi, R. Wille, and R. Drechsler, "Synthesis of quantum circuits for linear nearest neighbor architectures," *Quantum Inf. Process.*, vol. 10, no. 3, pp. 355–377, Oct. 2010, doi: 10.1007/s11128-010-0201-2.

[130] R. Wille, A. Lye, and R. Drechsler, "Exact reordering of circuit lines for nearest neighbor quantum architectures," *IEEE Trans. Comput.-Aided Design Integr.*, vol. 33, no. 12, pp. 1818–1831, Dec. 2014.

[131] J. X. Lin, E. R. Anschuetz, and A. W. Harrow, "Using spectral graph theory to map qubits onto connectivity-limited devices," *ACM Trans. Quantum Comput.*, vol. 2, no. 1, pp. 1–30, Apr. 2021, doi: 10.1145/3436752.

[132] A. Zulehner, A. Paler, and R. Wille, "An efficient methodology for mapping quantum circuits to the IBM QX architectures," 2017, *arXiv:1712.04722*.

[133] G. Li, Y. Ding, and Y. Xie, "Tackling the qubit mapping problem for NISQ-era quantum devices," 2018, *arXiv:1809.02573*.

[134] A. Paler, "On the influence of initial qubit placement during NISQ circuit compilation," 2018, *arXiv:1811.08985*.

[135] T. Itoko, R. Raymond, T. Imamichi, and A. Matsuo, "Optimization of quantum circuit mapping using gate transformation and commutation," *Integration*, vol. 70, pp. 43–50, Jan. 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167926019302755

[136] R. Wille, L. Burgholzer, and A. Zulehner, "Mapping quantum circuits to IBM QX architectures using the minimal number of SWAP and H operations," in *Proc. 56th Annu. Design Autom. Conf.*, Jun. 2019, pp. 1–6.

[137] L. Burgholzer, S. Schneider, and R. Wille, "Limiting the search space in optimal quantum circuit mapping," 2021, *arXiv:2112.00045*.

[138] B. Tan and J. Cong, "Optimal layout synthesis for quantum computing," in *Proc. 39th Int. Conf. Comput.-Aided Design*, New York, NY, USA, Nov. 2020, pp. 1–9, doi: 10.1145/3400302.3415620.

[139] W. Hattori and S. Yamashita, "Quantum circuit optimization by changing the gate order for 2D nearest neighbor architectures," in *Proc. Int. Conf. Reversible Comput.*, 2018, pp. 228–243, doi: 10.1007/978-3-319-99498-7_16.

[140] A. M. Childs, E. Schoute, and C. M. Unsal. (2019). *Circuit Transformations for Quantum Architectures*. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2019/10395/

[141] B. Schmitt, M. Soeken, and G. D. Micheli, "Symbolic algorithms for token swapping," in *Proc. IEEE 50th Int. Symp. Multiple-Valued Log. (ISMVL)*, Nov. 2020, pp. 28–33.

[142] P. Murali, J. M. Baker, A. Javadi-Abhari, F. T. Chong, and M. Martonosi, "Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers," in *Proc. 24th Int. Conf. Architectural Support Program. Lang. Operating Syst.*, Apr. 2019, pp. 1015–1029.

[143] A. Ash-Saki, M. Alam, and S. Ghosh, "QURE: Qubit re-allocation in noisy intermediate-scale quantum computers," in *Proc. 56th Annu. Design Autom. Conf.*, Jun. 2019, pp. 1–6.

[144] Y. Shi *et al.*, "Optimized compilation of aggregated instructions for realistic quantum computers," in *Proc. 24th Int. Conf. Architectural Support Program. Lang. Operating Syst.*, Apr. 2019, pp. 1031–1044.

[145] P. Murali, D. C. Mckay, M. Martonosi, and A. Javadi-Abhari, "Software mitigation of crosstalk on noisy intermediate-scale quantum computers," in *Proc. 25th Int. Conf. Architectural Support Program. Lang. Operating Syst.*, Mar. 2020, pp. 1001–1016.

[146] K. N. Smith *et al.*, "TimeStitch: Exploiting slack to mitigate decoherence in quantum circuits," *ACM Trans. Quantum Comput.*, Jul. 2022, doi: 10.1145/3548778.

[147] A. Y. Kitaev, "Fault-tolerant quantum computation by anyons," *Ann. Phys.*, vol. 303, no. 1, pp. 2–30, Jan. 2003.

[148] S. B. Bravyi and A. Y. Kitaev, "Quantum codes on a lattice with boundary," 1998, *arXiv:quant-ph/9811052*.

[149] C. Horsman, A. G. Fowler, S. Devitt, and R. V. Meter, "Surface code quantum computing by lattice surgery," *npj Quantum Inf.*, vol. 14, no. 12, pp. 1–27, 2012.

[150] S. Bravyi and A. Kitaev, "Universal quantum computation with ideal Clifford gates and noisy ancillas," *Phys. Rev. A, Gen. Phys.*, vol. 71, no. 2, Feb. 2005, Art. no. 022316.

[151] P. Pham and K. M. Svore, "A 2D nearest-neighbor quantum architecture for factoring in polylogarithmic depth," 2012, *arXiv:1207.6655*.

[152] D. Litinski, "A game of surface codes: Large-scale quantum computing with lattice surgery," *Quantum*, vol. 3, p. 128, Mar. 2019, doi: 10.22331/q-2019-03-05-128.

[153] M. Beverland, V. Kliuchnikov, and E. Schoute, "Surface code compilation via edge-disjoint paths," *PRX Quantum*, vol. 3, no. 2, May 2022, Art. no. 020342, doi: 10.1103/PRXQuantum.3.020342.

[154] Y. Ding and F. T. Chong, *Quantum Computer Systems: Research for Noisy Intermediate-Scale Quantum Computers* (Synthesis Lectures on Computer Architecture). San Rafael, CA, USA: Morgan & Claypool Publishers, 2020.

[155] I. L. Markov and Y. Shi, "Simulating quantum computation by contracting tensor networks," *SIAM J. Comput.*, vol. 38, no. 3, pp. 963–981, Jan. 2008.

[156] S. Boixo, S. V. Isakov, V. N. Smelyanskiy, and H. Neven, "Simulation of low-depth quantum circuits as complex undirected graphical models," 2017, *arXiv:1712.05384*.

[157] B. Villalonga *et al.*, "A flexible high-performance simulator for verifying and benchmarking quantum circuits implemented on real hardware," *npj Quantum Inf.*, vol. 5, no. 1, p. 86, Dec. 2019.

[158] Y. Suzuki *et al.*, "Qulacs: A fast and versatile quantum circuit simulator for research purpose," *Quantum*, vol. 5, p. 559, Oct. 2021.

[159] *CIRQ*, CIRQ Developers, Barangaroo, NSW, Australia, 2022, doi: 10.5281/zenodo.6599601.

[160] G. Aleksandrowicz *et al.*, "Qiskit: An open-source framework for quantum computing," IBM, Tech. Rep., Jan. 2019, doi: 10.5281/zenodo.2562111.

[161] T. Jones, A. Brown, I. Bush, and S. C. Benjamin, "QuEST and high performance simulation of quantum computers," *Sci. Rep.*, vol. 9, no. 1, p. 10736, Dec. 2019.

[162] J. Gray, "Quimb: A Python package for quantum information and many-body calculations," *J. Open Source Softw.*, vol. 3, no. 29, p. 819, Sep. 2018.

[163] Z. Wang *et al.*, "A quantum circuit simulator and its applications on Sunway TaihuLight supercomputer," *Sci. Rep.*, vol. 11, no. 1, p. 355, Dec. 2021.

[164] R. E. Bryant, "Graph-based algorithms for Boolean function manipulation," *IEEE Trans. Comput.*, vol. C-35, no. 8, pp. 677–691, Aug. 1986.

[165] F. Somenzi, "CUDD: CU decision diagram package (release 2.4.2)," Univ. Colorado Boulder, Boulder, CO, USA, Tech. Rep., 2005.

[166] G. F. Viamontes, I. L. Markov, and J. P. Hayes, *Quantum Circuit Simulation*. Dordrecht, The Netherlands: Springer, 2009.

[167] V. Samoladas, "Improved BDD algorithms for the simulation of quantum circuits," in *Proc. Eur. Symp. Algorithms*, 2008, pp. 720–731.

[168] A. Zulehner and R. Wille, "Advanced simulation of quantum computations," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 38, no. 5, pp. 848–859, May 2019.

[169] Y.-H. Tsai, J.-H.-R. Jiang, and C.-S. Jhang, "Bit-slicing the Hilbert space: Scaling up accurate quantum circuit simulation," in *Proc. 58th ACM/IEEE Design Autom. Conf. (DAC)*, Dec. 2021, pp. 439–444.

[170] P. Niemann, R. Wille, D. M. Miller, M. A. Thornton, and R. Drechsler, "QMDDs: Efficient quantum function representation and manipulation," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 35, no. 1, pp. 86–99, Jul. 2016.

[171] T. Grurl, J. Fus, S. Hillmich, L. Burgholzer, and R. Wille, "Arrays vs. decision diagrams: A case study on quantum circuit simulators," in *Proc. IEEE 50th Int. Symp. Multiple-Valued Log. (ISMVL)*, Nov. 2020, pp. 176–181.

[172] K. N. Smith and M. A. Thornton, "A quantum computational compiler and design tool for technology-specific targets," in *Proc. 46th Int. Symp. Comput. Archit.*, Jun. 2019, pp. 579–588.

[173] P. Niemann, A. Zulehner, R. Drechsler, and R. Wille, "Overcoming the tradeoff between accuracy and compactness in decision diagrams for quantum computation," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 12, pp. 4657–4668, Dec. 2020.

[174] X. Hong, X. Zhou, S. Li, Y. Feng, and M. Ying, "A tensor network based decision diagram for representation of quantum circuits," *ACM Trans. Design Autom. Electron. Syst.*, vol. 27, no. 6, pp. 1–30, Nov. 2022.

[175] S. Aaronson and D. Gottesman, "Improved simulation of stabilizer circuits," *Phys. Rev. A, Gen. Phys.*, vol. 70, no. 5, Nov. 2004, Art. no. 052328.

[176] S. Anders and H. J. Briegel, "Fast simulation of stabilizer circuits using a graph-state representation," *Phys. Rev. A, Gen. Phys.*, vol. 73, no. 2, Feb. 2006, Art. no. 022334.

[177] H. J. Garcia, I. L. Markov, and A. W. Cross, "On the geometry of stabilizer states," *Quantum Inf. Comput.*, vol. 14, no. 7, pp. 683–720, May 2014.

[178] S. Bravyi, G. Smith, and J. A. Smolin, "Trading classical and quantum computational resources," *Phys. Rev. X*, vol. 6, no. 2, Jun. 2016, Art. no. 021043.

[179] S. Bravyi, D. Browne, P. Calpin, E. Campbell, D. Gosset, and M. Howard, "Simulation of quantum circuits by low-rank stabilizer decompositions," *Quantum*, vol. 3, p. 181, Sep. 2019.

[180] G. F. Viamontes, I. L. Markov, and J. P. Hayes, "Checking equivalence of quantum circuits and states," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, Nov. 2007, pp. 69–74.

[181] S. Yamashita and I. L. Markov, "Fast equivalence-checking for quantum circuits," in *Proc. IEEE/ACM Int. Symp. Nanosc. Archit.*, Jun. 2010, pp. 23–28.

[182] L. Burgholzer and R. Wille, "Advanced equivalence checking for quantum circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 40, no. 9, pp. 1810–1824, Sep. 2021.

[183] C.-Y. Wei, Y.-H. Tsai, C.-S. Jhang, and J.-H.-R. Jiang, "Accurate BDD-based unitary operator manipulation for scalable and robust quantum circuit verification," in *Proc. 59th ACM/IEEE Design Autom. Conf.*, Jul. 2022, pp. 523–528.

[184] X. Hong, M. Ying, Y. Feng, X. Zhou, and S. Li, "Approximate equivalence checking of noisy quantum circuits," in *Proc. 58th ACM/IEEE Design Autom. Conf. (DAC)*, Dec. 2021, pp. 637–642.

[185] T.-F. Chen, J.-H. R. Jiang, and M.-H. Hsieh, "Partial equivalence checking of quantum circuits," in *Proc. Int. Conf. Quantum Comput. Eng.*, 2022, pp. 1–11.

[186] Q. Wang, R. Li, and M. Ying, "Equivalence checking of sequential quantum circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 9, pp. 3143–3156, Sep. 2022.

[187] X. Hong, Y. Feng, S. Li, and M. Ying, "Equivalence checking of dynamic quantum circuits," 2021, *arXiv:2106.01658*.

[188] L. Burgholzer and R. Wille, "Handling non-unitaries in quantum circuit equivalence checking," in *Proc. 59th ACM/IEEE Design Autom. Conf.*, Jul. 2022, pp. 1–6.

[189] The Coq Development Team, "The Coq proof assistant," Tech. Rep., Jan. 2022, doi: 10.5281/zenodo.5846982.

[190] J. Boender, F. Kammüller, and R. Nagarajan, "Formalization of quantum protocols using Coq," in *Proc. 12th Int. Workshop Quantum Phys. Log. (QPL)*, in Electronic Proceedings in Theoretical Computer Science, vol. 195, C. Heunen, P. Selinger, and J. Vicary, Eds. Waterloo, NSW, Australia: Open Publishing Association, Nov. 2015, pp. 71–83.

[191] R. Rand, J. Paykin, and S. Zdancewic, "QWIRE practice: Formal verification of quantum circuits in Coq," in *Proc. 14th Int. Conf. Quantum Phys. Log. (QPL)*, in Electronic Proceedings in Theoretical Computer Science, vol. 266, B. Coecke and A. Kissinger, Eds. Waterloo, NSW, Australia: Open Publishing Association, Feb. 2018, pp. 119–132.

[192] C. Chareton, S. Bardin, F. Bobot, V. Perrelle, and B. Valiron, "An automated deductive verification framework for circuit-building quantum programs," in *Programming Languages and Systems* (Lecture Notes in Computer Science), vol. 12648, N. Yoshida, Ed. Cham, Switzerland: Springer, Mar. 2021, pp. 148–177.

[193] J.-C. Filliâtre and A. Paskevich, "Why3—Where programs meet provers," in *Proc. Eur. Symp. Program.* Berlin, Germany: Springer, 2013, pp. 125–128.

[194] K. Hietala, R. Rand, S.-H. Hung, L. Li, and M. Hicks, "Proving quantum programs correct," in *Proc. 12th Int. Conf. Interact. Theorem Proving*, in Leibniz International Proceedings in Informatics (LIPIcs), vol. 193, L. Cohen and C. Kaliszyk, Eds. Dagstuhl, Germany: Schloss Dagstuhl—Leibniz-Zentrum für Informatik, Jun. 2021, pp. 21:1–21:19. [Online]. Available: https://github.com/inQWIRE/SQIR

[195] Y. Peng *et al.* (Apr. 2022). *A Formally Certified End-to-End Implementation of Shor's Factorization Algorithm*. [Online]. Available: https://github.com/inQWIRE/SQIR/tree/main/examples/shor

[196] E. D'Hondt and P. Panangaden, "Quantum weakest preconditions," *Math. Struct. Comput. Sci.*, vol. 16, no. 3, pp. 429–451, Jun. 2006.

[197] D. Kozen, "A probabilistic PDL," in *Proc. 15th Annu. ACM Symp. Theory Comput.*, D. S. Johnson *et al.*, Eds. Boston, MA, USA, 1983, pp. 291–297, doi: 10.1145/800061.808758.

[198] M. Ying, "Floyd-Hoare logic for quantum programs," *ACM Trans. Program. Lang. Syst.*, vol. 33, no. 6, pp. 1–49, Dec. 2011.

[199] L. Zhou, N. Yu, and M. Ying, "An applied quantum Hoare logic," in *Proc. 40th ACM SIGPLAN Conf. Program. Lang. Design Implement.*, Phoenix, AZ, USA, Jun. 2019, pp. 1149–1162, doi: 10.1145/3314221.3314584.

[200] D. Unruh, "Quantum Hoare logic with ghost variables," in *Proc. 34th Annu. ACM/IEEE Symp. Log. Comput. Sci. (LICS)*, Vancouver, BC, Canada, Jun. 2019, pp. 1–13. [Online]. Available: https://ieeexplore.ieee.org/document/8785779/

[201] S.-H. Hung, K. Hietala, S. Zhu, M. Ying, M. Hicks, and X. Wu, "Quantitative robustness analysis of quantum programs," *Proc. ACM Program. Lang.*, vol. 3, pp. 1–29, Jan. 2019.

[202] M. Ying, L. Zhou, Y. Li, and Y. Feng, "A proof system for disjoint parallel quantum programs," *Theor. Comput. Sci.*, vol. 897, pp. 164–184, Jan. 2022.

[203] L. Zhou, G. Barthe, J. Hsu, M. Ying, and N. Yu, "A quantum interpretation of bunched logic & quantum separation logic," in *Proc. 36th Annu. ACM/IEEE Symp. Log. Comput. Sci. (LICS)*, Los Alamitos, CA, USA, Jun. 2021, pp. 1–14.

[204] X.-B. Le, S.-W. Lin, J. Sun, and D. Sanan, "A quantum interpretation of separating conjunction for local reasoning of quantum programs based on separation logic," *Proc. ACM Program. Lang.*, vol. 6, p. 36, Jan. 2022.

[205] D. Unruh, "Quantum relational Hoare logic," *Proc. ACM Program. Lang.*, vol. 3, pp. 1–31, Jan. 2019.

[206] T. Nipkow, M. Wenzel, and L. C. Paulson, *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*. Berlin, Germany: Springer, 2002.

[207] J. Liu *et al.*, *Formal Verification of Quantum Algorithms Using Quantum Hoare Logic* (Lecture Notes in Computer Science), vol. 11562. Cham, Switzerland: Springer, 2019, pp. 187–207, doi: 10.1007/978-3-030-25543-5_12.

[208] C. Chareton, S. Bardin, D. Lee, B. Valiron, R. Vilmart, and Z. Xu, "Formal methods for quantum programs: A survey," 2021, *arXiv:2109.06493*.

[209] J. Majer *et al.*, "Coupling superconducting qubits via a cavity bus," *Nature*, vol. 449, no. 7161, pp. 443–447, Sep. 2007.

[210] S. M. Girvin, "Circuit QED: Superconducting qubits coupled to microwave photons," in *Quantum Machines: Measurement and Control of Engineered Quantum Systems*. Oxford Univ. Press, 2014, pp. 113–256.

[211] W. D. Oliver and P. B. Welander, "Materials in superconducting quantum bits," *MRS Bull.*, vol. 38, no. 10, pp. 816–825, Oct. 2013.

[212] V. E. Manucharyan, J. Koch, L. I. Glazman, and M. H. Devoret, "Fluxonium: Single Cooper-pair circuit free of charge offsets," *Science*, vol. 326, no. 5949, pp. 113–116, Oct. 2009.

[213] J. E. Mooij, T. P. Orlando, L. Levitov, L. Tian, C. H. van der Wal, and S. Lloyd, "Josephson persistent-current qubit," *Science*, vol. 285, no. 5430, pp. 1036–1039, Aug. 1999.

[214] J. Koch *et al.*, "Charge-insensitive qubit design derived from the Cooper pair box," *Phys. Rev. A, Gen. Phys.*, vol. 76, no. 4, Oct. 2007, Art. no. 042319.

[215] R. Barends *et al.*, "Logic gates at the surface code threshold: Superconducting qubits poised for fault-tolerant quantum computing," 2014, *arXiv:1402.4848*.

[216] S. Sheldon, E. Magesan, J. M. Chow, and J. M. Gambetta, "Procedure for systematically tuning up cross-talk in the cross-resonance gate," *Phys. Rev. A, Gen. Phys.*, vol. 93, no. 6, Jun. 2016, Art. no. 060302.

[217] S. S. Hong *et al.*, "Demonstration of a parametrically activated entangling gate protected from flux noise," *Phys. Rev. A, Gen. Phys.*, vol. 101, no. 1, Jan. 2020, Art. no. 012302.

[218] R. Acharya *et al.*, "Suppressing quantum errors by scaling a surface code logical qubit," 2022, *arXiv:2207.06431*.

[219] (2021). *Rigetti Computing Announces Next-Generation 40 Q and 80 Q Quantum Systems*. [Online]. Available: https://investors.rigetti.com/news-releases/news-release-details/rigetti-computing-announces-next-generation-40q-and-80q-quantum

[220] (2021). *Quantum Computer Datasheet*. [Online]. Available: https://quantumai.google/hardware/datasheet/weber.pdf

[221] (2022). *Eagle's Quantum Performance Progress*. [Online]. Available: https://research.ibm.com/blog/eagle-quantum-processor-performance

[222] J. J. Burnett *et al.*, "Decoherence benchmarking of superconducting qubits," *npj Quantum Inf.*, vol. 5, no. 1, pp. 1–8, Dec. 2019.

[223] P. D. Nation, H. Kang, N. Sundaresan, and J. M. Gambetta, "Scalable mitigation of measurement errors on quantum computers," *PRX Quantum*, vol. 2, no. 4, Nov. 2021, Art. no. 040326.

[224] N. Wittler *et al.*, "Integrated tool set for control, calibration, and characterization of quantum devices applied to superconducting qubits," *Phys. Rev. A, Gen. Phys.*, vol. 15, no. 3, Mar. 2021, Art. no. 034080.

[225] L. DiCarlo *et al.*, "Demonstration of two-qubit algorithms with a superconducting quantum processor," *Nature*, vol. 460, no. 7252, pp. 240–244, 2009.

[226] R. Barends *et al.*, "Coherent Josephson qubit suitable for scalable quantum integrated circuits," *Phys. Rev. Lett.*, vol. 111, no. 8, Aug. 2013, Art. no. 080502.

[227] D. C. McKay, S. Filipp, A. Mezzacapo, E. Magesan, J. M. Chow, and J. M. Gambetta, "Universal gate for fixed-frequency qubits via a tunable bus," *Phys. Rev. A, Gen. Phys.*, vol. 6, no. 6, Dec. 2016, Art. no. 064007.

[228] S. Caldwell *et al.*, "Parametrically activated entangling gates using transmon qubits," *Phys. Rev. A, Gen. Phys.*, vol. 10, no. 3, 2018, Art. no. 034050.

[229] M. Brink, J. M. Chow, J. Hertzberg, E. Magesan, and S. Rosenblatt, "Device challenges for near term superconducting quantum processors: Frequency collisions," in *IEDM Tech. Dig.*, Dec. 2018, pp. 1–6.

[230] J. Kreikebaum, K. O'Brien, A. Morvan, and I. Siddiqi, "Improving wafer-scale Josephson junction resistance variation in superconducting quantum coherent circuits," *Supercond. Sci. Technol.*, vol. 33, no. 6, 2020, Art. no. 06LT02.

[231] J. M. Chow *et al.*, "Universal quantum gate set approaching fault-tolerant thresholds with superconducting qubits," *Phys. Rev. Lett.*, vol. 109, no. 6, Aug. 2012, Art. no. 060501.

[232] A. D. Córcoles *et al.*, "Demonstration of a quantum error detection code using a square lattice of four superconducting qubits," *Nature Commun.*, vol. 6, no. 1, pp. 1–10, Nov. 2015.

[233] M. Takita *et al.*, "Demonstration of weight-four parity measurements in the surface code architecture," *Phys. Rev. Lett.*, vol. 117, no. 21, Nov. 2016, Art. no. 210505.

[234] J. Gambetta, J. M. Chow, and M. Steffen, "Building logical qubits in a superconducting quantum computing system," *NPJ Quantum Inf.*, vol. 3, no. 2, pp. 1–7, 2017.

[235] Q. P. Herr, A. Y. Herr, O. T. Oberg, and A. G. Ioannidis, "Ultra-low-power superconductor logic," *J. Appl. Phys.*, vol. 109, no. 10, May 2011, Art. no. 103903, doi: 10.1063/1.3585849.

[236] K. K. Likharev and V. K. Semenov, "RSFQ logic/memory family: A new Josephson-junction technology for sub-terahertz-clock-frequency digital systems," *IEEE Trans. Appl. Supercond.*, vol. 1, no. 1, pp. 3–28, Mar. 1991, doi: 10.1109/77.80745.

[237] O. A. Mukhanov, "Energy-efficient single flux quantum technology," *IEEE Trans. Appl. Supercond.*, vol. 21, no. 3, pp. 760–769, Jun. 2011, doi: 10.1109/TASC.2010.2096792.

[238] M. Tanaka, A. Kitayama, T. Koketsu, M. Ito, and A. Fujimaki, "Low-energy consumption RSFQ circuits driven by low voltages," *IEEE Trans. Appl. Supercond.*, vol. 23, no. 3, Jun. 2013, Art. no. 1701104, doi: 10.1109/TASC.2013.2240555.

[239] G. Krylov and E. G. Friedman, "Asynchronous dynamic single-flux quantum majority gates," *IEEE Trans. Appl. Supercond.*, vol. 30, no. 5, pp. 1–7, Aug. 2020, doi: 10.1109/TASC.2020.2978428.

[240] C. J. Fourie *et al.*, "ColdFlux superconducting EDA and TCAD tools project: Overview and progress," *IEEE Trans. Appl. Supercond.*, vol. 29, no. 5, pp. 1–7, Aug. 2019, doi: 10.1109/TASC.2019.2892115.

[241] N. K. Katam and M. Pedram, "Logic optimization, complex cell design, and retiming of single flux quantum circuits," *IEEE Trans. Appl. Supercond.*, vol. 28, no. 7, pp. 1–9, Oct. 2018, doi: 10.1109/TASC.2018.2856833.

[242] G. Pasandi and M. Pedram, "A dynamic programming-based, path balancing technology mapping algorithm targeting area minimization," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2019, pp. 1–8, doi: 10.1109/ICCAD45719.2019.8942053.

[243] T. Jabbari, G. Krylov, J. Kawa, and E. G. Friedman, "Splitter trees in single flux quantum circuits," *IEEE Trans. Appl. Supercond.*, vol. 31, no. 5, pp. 1–6, Aug. 2021, doi: 10.1109/TASC.2021.3070802.

[244] C. L. Ayala, T. Tanaka, R. Saito, M. Nozoe, N. Takeuchi, and N. Yoshikawa, "MANA: A monolithic adiabatic iNtegration architecture microprocessor using 1.4 zJ/op superconductor Josephson junction devices," in *Proc. IEEE Symp. VLSI Circuits*, Jun. 2020, pp. 1–2, doi: 10.1109/VLSICircuits18222.2020.9162792.

[245] R. Cai *et al.*, "A majority logic synthesis framework for adiabatic quantum-flux-parametron superconducting circuits," in *Proc. Great Lakes Symp. VLSI*, May 2019, pp. 189–194, doi: 10.1145/3299874.3317980.

[246] R. Cai, O. Chen, A. Ren, N. Liu, N. Yoshikawa, and Y. Wang, "A buffer and splitter insertion framework for adiabatic quantum-flux-parametron superconducting circuits," in *Proc. IEEE 37th Int. Conf. Comput. Design (ICCD)*, Nov. 2019, pp. 429–436, doi: 10.1109/ICCD46524.2019.00067.

[247] E. Testa, S.-Y. Lee, H. Riener, and G. De Micheli, "Algebraic and Boolean optimization methods for AQFP superconducting circuits," in *Proc. 26th Asia South Pacific Design Autom. Conf.*, Jan. 2021, pp. 779–785, doi: 10.1145/3394885.3431606.

[248] S.-Y. Lee, H. Riener, and G. De Micheli, "Beyond local optimality of buffer and splitter insertion for AQFP circuits," in *Proc. 59th ACM/IEEE Design Autom. Conf.*, Jul. 2022, pp. 445–450.

[249] L. Amarú *et al.*, "First demonstration of a superconducting electronics microcontroller RTL-to-GDSII flow," in *Proc. Government Microcircuit Appl. Crit. Technol. Conf. (GOMACTech)*, 2021, pp. 1–4.

[250] R. Bairamkulov, T. Jabbari, and E. G. Friedman, "QuCTS—Single flux quantum clock tree synthesis," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, early access, Oct. 26, 2022, doi: 10.1109/TCAD.2021.3123141.

[251] T. Jabbari, G. Krylov, S. Whiteley, E. Mlinar, J. Kawa, and E. G. Friedman, "Interconnect routing for large-scale RSFQ circuits," *IEEE Trans. Appl. Supercond.*, vol. 29, no. 5, pp. 1–5, Aug. 2019.

[252] L. Schindler *et al.*, "Standard cell layout synthesis for row-based placement and routing of RSFQ and AQFP logic families," in *Proc. IEEE Int. Supercond. Electron. Conf. (ISEC)*, Jul. 2019, pp. 1–5.

[253] T. Jabbari and E. G. Friedman, "Flux mitigation in wide superconductive striplines," *IEEE Trans. Appl. Supercond.*, vol. 32, no. 5, pp. 1–6, Aug. 2022.

[254] G. Krylov and E. G. Friedman, *Single Flux Quantum Integrated Circuit Design*. Cham, Switzerland: Springer, 2022.

[255] N. Takeuchi, D. Ozawa, Y. Yamanashi, and N. Yoshikawa, "An adiabatic quantum flux parametron as an ultra-low-power logic device," *Supercond. Sci. Technol.*, vol. 26, no. 3, Mar. 2013, Art. no. 035010, doi: 10.1088/0953-2048/26/3/035010.

[256] E. Goto, "The parametron, a digital computing element which utilizes parametric oscillation," *Proc. IRE*, vol. 47, no. 8, pp. 1304–1316, Aug. 1959, doi: 10.1109/JRPROC.1959.287195.

[257] N. Takeuchi, Y. Yamanashi, and N. Yoshikawa, "Adiabatic quantum-flux-parametron cell library adopting minimalist design," *J. Appl. Phys.*, vol. 117, no. 17, May 2015, Art. no. 173912, doi: 10.1063/1.4919838.

[258] C. L. Ayala, N. Takeuchi, Y. Yamanashi, T. Ortlepp, and N. Yoshikawa, "Majority-logic-optimized parallel prefix carry look-ahead adder families using adiabatic quantum-flux-parametron logic," *IEEE Trans. Appl. Supercond.*, vol. 27, no. 4, pp. 1–7, Jun. 2017, doi: 10.1109/TASC.2016.2642041.

[259] C. L. Ayala *et al.*, "A semi-custom design methodology and environment for implementing superconductor adiabatic quantum-flux-parametron microprocessors," *Supercond. Sci. Technol.*, vol. 33, no. 5, May 2020, Art. no. 054006, doi: 10.1088/1361-6668/ab7ec3.

[260] L. Amarú, P.-E. Gaillardon, and G. De Micheli, "Majority-inverter graph: A novel data-structure and algorithms for efficient logic optimization," in *Proc. The 51st Annu. Design Autom. Conf. Design Autom. Conf.*, 2014, p. 194, doi: 10.1145/2593069.2593158.

[261] L. Amaru, P. E. Gaillardon, and G. D. Micheli, "Majority-inverter graph: A new paradigm for logic optimization," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 35, no. 5, pp. 806–819, May 2015, doi: 10.1109/TCAD.2015.2488484.

[262] W. P. Burleson, M. Ciesielski, F. Klass, and W. Liu, "Wave-pipelining: A tutorial and research survey," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 6, no. 3, pp. 464–474, Sep. 1998, doi: 10.1109/92.711317.

[263] T. V. Filippov *et al.*, "20 GHz operation of an asynchronous wave-pipelined RSFQ arithmetic-logic unit," *Phys. Proc.*, vol. 36, pp. 59–65, Jan. 2012, doi: 10.1016/j.phpro.2012.06.130.

[264] Z. J. Deng, N. Yoshikawa, J. A. Tierno, S. R. Whiteley, and T. Van Duzer, "Asynchronous circuits and systems in superconducting RSFQ digital technology," in *Proc. 4th Int. Symp. Adv. Res. Asynchronous Circuits Syst.*, Mar./Apr. 1998, pp. 274–285, doi: 10.1109/ASYNC.1998.666512.

[265] F. Ke, O. Chen, Y. Wang, and N. Yoshikawa, "Demonstration of a 47.8 GHz high-speed FFT processor using single-flux-quantum technology," *IEEE Trans. Appl. Supercond.*, vol. 31, no. 5, pp. 1–5, Aug. 2021, doi: 10.1109/TASC.2021.3059984.

[266] M. Beverland, E. Campbell, M. Howard, and V. Kliuchnikov, "Lower bounds on the non-Clifford resources for quantum computations," *Quantum Sci. Technol.*, vol. 5, no. 3, Jun. 2020, Art. no. 035009.

Giovanni De Micheli (Life Fellow, IEEE) received the Nuclear Engineer degree from the Politecnico di Milano in 1979, and the M.S. and Ph.D. degrees in electrical engineering and computer science from the University of California at Berkeley in 1980 and 1983, respectively.

He is currently a Professor in electronics and computer Science at EPFL, Lausanne, Switzerland. He is credited for the invention of the network on chip design automation paradigm and for the creation of algorithms and design tools for electronic design automation (EDA). He is also the Director of the Integrated Systems Laboratory, EPFL. Previously, he was a Professor of electrical engineering at Stanford University. He was the Director of the Electrical Engineering Institute (IEL), EPFL, from 2008 to 2019, and a Program Leader of the Swiss Federal Nano-Tera.ch Program. He is the author of *Synthesis and Optimization of Digital Circuits* (McGraw-Hill, 1994), the coauthor and/or the co-editor of ten other books and of over 900 technical articles. His citation H-index is above 100 according to Google Scholar. His current research interests include several aspects of design technologies for integrated circuits and systems, such as synthesis for emerging technologies. He is also interested in heterogeneous platform design, including electrical components and biosensors, as well as in data processing of biomedical information.

Prof. De Micheli is a fellow of ACM and AAAS. He is a member of the Academia Europaea and an International Honorary Member of the American Academy of Arts and Sciences. He is a member of the Scientific Advisory Board of IMEC (Leuven, B) and STMicroelectronics. He was a recipient of the 2020 IEEE/TC Achievement Award in Cyberphysical Systems, the 2020 IEEE/CEDA Richard Newton Technical Impact Award, the 2019 ACM/SIGDA Pioneering Achievement Award, the 2016 EDAA Lifetime Achievement Award, the 2016 IEEE/CS Harry Goode Award for seminal contributions to design and design tools of networks on chips, the 2012 IEEE/CAS Mac Van Valkenburg Award for contributions to theory, practice and experimentation in design methods and tools, and the 2003 IEEE Emanuel Piore Award for contributions to computer-aided synthesis of digital systems. He received the Golden Jubilee Medal for outstanding contributions to the IEEE CAS Society in 2000, the D. Pederson Award for the Best Paper on the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS in 1987 and 2018, and several Best Paper Awards, including DAC in 1983 and 1993, DATE in 2005, Nanoarch in 2010 and 2012, and Mobihealth in 2016. He has been the Chair of several conferences, including Memocode in 2014, DATE in 2010, pHealth in 2006, VLSI SOC in 2006, DAC in 2000, and ICCD in 1989. He has been serving IEEE in several capacities, namely the Division 1 Director (2008–2009), the Co-Founder and the President Elect of the IEEE Council on EDA (2005–2007), the President of the IEEE CAS Society in 2003, and the Editor-in-Chief of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS (1997–2001).

Jie-Hong R. Jiang (Member, IEEE) received the B.S. and M.S. degrees in electronics engineering from the National Chiao Tung University, Hsinchu, Taiwan, in 1996 and 1998, respectively, and the Ph.D. degree in electrical engineering and computer sciences from the University of California at Berkeley, Berkeley, CA, USA, in 2004.

He is currently a Professor with the Department of Electrical Engineering and the Graduate Institute of Electronics Engineering, National Taiwan University, Taipei, Taiwan. He leads the Applied Logic and Computation Laboratory, and worked extensively on logic synthesis, formal verification, electronic design automation, and computation models of biological and physical systems.

Dr. Jiang is a member of the Phi Tau Phi and the Association for Computing Machinery.

Robert Rand received the B.A. degree in mathematics and computer science from Yeshiva University in 2011 and the Ph.D. degree in computer science from the University of Pennsylvania in 2018.

He is currently an Assistant Professor of computer science at The University of Chicago (UChicago), where he leads the Chicago Quantum Programming Languages Laboratory (ChiQP). Before joining UChicago, he was a Basili Post-Doctoral Fellow at the Joint Center for Quantum Information and Computer Science (QUiCS), University of Maryland. He is a member of the Chicago's Programming Languages Research Group, the Chicago Quantum Exchange, and the Argonne National Laboratory, where he maintains an affiliate appointment. His main projects include the QWIRE quantum circuit language, the VOQC compiler for quantum circuits, and a stabilizer-based type system for quantum programs. He is also developing tools for promising models of quantum computation like the ZX-calculus and the one-way quantum computer. He also works on formalizing and verifying existing languages like Microsoft's Q# and the quantum assembly language OpenQASM, as well as programs like Grover's search algorithm and Shor's factoring algorithm. He led the development of the INQWIRE QuantumLib, an open source library for verified quantum computing in the Coq proof assistant, which underlies many of his projects, including his online textbook *Verified Quantum Computing*. His main research interests include applying techniques from programming languages and formal verification to the domain of quantum computation.

Kaitlin Smith (Member, IEEE) received the B.S. degree in mathematics and electrical engineering and the M.S. and Ph.D. degrees in electrical engineering from Southern Methodist University (SMU) in 2014, 2015, and December 2019, respectively. She is currently a CQE/IBM Post-Doctoral Scholar with the Department of Computer Science, The University of Chicago. Within the focus of quantum computing, her work involves technology-aware programming, computer architecture, distributed computing, and security. She was named a 2021 MIT EECS Rising Star. She was a recipient of the 2021 IEEE Computer Society Technical Committee on Multiple Valued Logic (TC-MVL) Kenneth C. Smith Early Career Award in Microelectronics.

Mathias Soeken (Member, IEEE) received the Ph.D. degree (Dr.-Ing.) in computer science from the University of Bremen, Germany, in 2013. From 2009 to 2015, he worked at the University of Bremen, Germany, where he was a Co-Founder of RevKit, a framework for reversible logic synthesis. From 2015 to 2020, he has been a Post-Doctoral Scientist with the École Polytechnique Fédérale Lausanne (EPFL), Switzerland, where he pioneered the EPFL Logic Synthesis Libraries, a set of industrial-strength academic logic synthesis tools, which today are the backbone of many logic synthesis and quantum compilation tools. He currently works at the Quantum Team, Microsoft. His research interests include logic synthesis, quantum computing, and formal verification.