QASMBench: A Low-level QASM Benchmark Suite for NISQ Evaluation and Simulation

Ang Li and Sriram Krishnamoorthy
Pacific Northwest National Laboratory, Richland, WA
{ang.li, sriram}@pnnl.gov

***Abstract*—The rapid development of quantum computing in the NISQ era urgently demands a low-level benchmark suite for conveniently evaluating and verifying the properties of selective prototype hardware, the efficiency of different assemblers, optimizers and schedulers, the robustness of distinct error correction technologies, and the performance of various quantum simulators on classical computers. In this paper, we fill this gap by proposing a low-level, light-weighted, and easy-to-use benchmark suite called QASMBench based on the OpenQASM assembly representation. It collects commonly seen quantum algorithms and routines from a variety of domains including chemistry, simulation, linear algebra, searching, optimization, quantum arithmetic, machine learning, fault tolerance, cryptography, etc. QASMBench trades-off between generality and usability. It covers the number of qubits ranging from 2 to 60K, and the circuit depth from 4 to 12M, while keeping most of the benchmarks with qubits less than 16 so they can be directly verified on contemporary public-available cloud quantum machines. QASMBench is available at https://github.com/uuudown/QASMBench.**

***Index Terms*—Quantum, QASM, benchmark, NISQ**

I. Introduction

Quantum computing (QC) [13, 86] has been envisioned as one of the most promising computing paradigms beyond Moore's Law for tackling difficult challenges that are classically intractable arising from various domains like chemistry [49, 67], machine learning [18, 98], cryptography [50, 101], linear algebra [27, 58], finance [94, 112], recommendation [68], physics simulation [36, 43], networking [69, 82], etc. QC relies on fundamental quantum mechanisms such as superposition and entanglement for conducting computation, in hopes of substantial speedups over existing algorithms on classical computers [29, 101].

Despite holding great promise, QC on contemporary noisy-intermediate-scale-quantum (NISQ) [91] devices is still distant from outperforming their classical counterparts regarding general problems. NISQ devices describe the near-term quantum platforms comprising 50 to hundreds of qubits, which is unlikely adequate for realizing full-scale fault-tolerance, but can demonstrate promising results in a bunch of problems and lay the foundations towards practical quantum computing [91].

The QC landscape is rapidly evolving nowadays. On the software side, an explosion of QC software have been developed over classical programming languages (e.g., Python [52, 104], C/C++ [63], JavaScript [61], ML [1]) A list of open-source QC projects, quantum algorithms, and quantum simulators over classical computers can be found in [44], [65], and [92], respectively. On the hardware side, the development of QC testbeds have proceeded to a stage where small working prototypes are available for public use through cloud service, such as Rigetti [95] and IBM quantum experience (QE) [60], showing encouraging results [25, 38]. On the application side, among others, Google has announced quantum supremacy for the task of sampling the output of a pseudo-random quantum circuit, over a 53-qubit system [8]. Several quantum machine learning frameworks (e.g., [22, 61]) and chemistry simulation packages have also been released (e.g., [61, 78]).

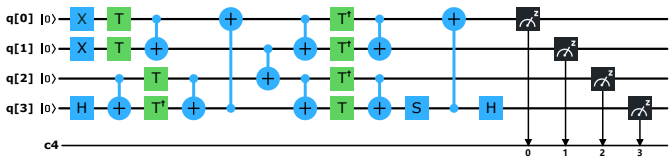

Fig. 1: An example quantum adder circuit with width=4 (i.e., 4 qubits) and depth=23 (i.e., 23 gates) followed by measurement.

To close the gap between practical quantum applications and real quantum machines, contributions from the computer system and architecture community are undoubtedly necessary in tacking several grant challenges [24], including software and hardware verification [84, 107], defining and perforating abstraction boundaries [84], managing parallelism and communication [74], mapping and scheduling computation [51, 99], elevating control complexity [75], hardware-specific optimization [83, 103, 108], investigating and mitigating noise [85, 106], etc. These massive effort, however, are currently evaluated and verified using arbitrarily selected QC benchmarks [34, 74, 75, 83, 85, 99, 106, 107, 108], lacking the common ground for judicious analysis and fair comparison among each other. The community thus urgently requires a light-weighted, low-level and easy-to-use benchmark suite for convenient characterization and evaluation purposes. This is especially the case in the coming NISQ era, as the noise and technology limitation continuously remain big challenges in the near feature. In this paper, we propose a low-level benchmark suite called QASMBench based on the OpenQASM intermediate representation (IR) assembly [31]. It summarizes commonly seen quantum algorithms and routines (e.g., Figure 1) from a variety of distinct domains, including chemistry, simulation, linear algebra, searching, optimization, quantum arithmetic, machine learning, fault tolerance, cryptography, etc. We trade-off between generality and usability. The benchmark suite include three categories: *small*, *medium* and *large*, which cover a wide spectrum of qubits (i.e. circuit width, see Figure 1.) ranging from 2 to 60K, and gates (i.e.,

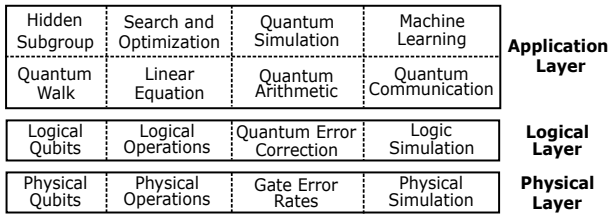

Fig. 2: Quantum computing stack. The applications in the application layer imply the algorithm categories in QASMBench.

circuit depth, Figure 1.) ranging from 4 to 12M. Most of the small and medium benchmarks in QASMBench can be directly uploaded and validated on the IBM quantum machines [60].

This paper is organized as follows: In Section II, we briefly discuss NISQ and the landscape of OpenQASM. In Section III, we introduce the QASMBench benchmark suite. In Section IV, we evaluate the small and medium benchmarks on real quantum machines. Finally in Section V, we summarize.

II. Background

A. Quantum Computing on NISQ Devices

Figure 2 shows the low-level QC stack. Quantum algorithms are usually presumed to be executed on logical qubits where full gate fidelity can be ensured. However, due to decoherence, environmental noise and read-in/out error, logic qubits cannot be directly mapped to physical qubits. This is particularly the case for NISQ without quantum-error-correction (QEC).

NISQ devices describe the near-term quantum platforms incorporating 50 to less than a thousand qubits [91]. They are currently built through a variety of quantum technologies, including superconducting qubits [28, 96], trapped-ions qubits [26, 73], spin qubits [77, 90], photonic qubits [9, 87], etc. To correctly execute a QC circuit, the qubits have to be coherent sufficiently long to allow the accomplishment of all the gate operations, which imposes strict constraints on the depth of the circuit [30]. On the other hand, gate operations should be precisely and quickly performed on the qubits during the coherent window for maintaining desired quantum states. The initial state preparation and output read-out are also of great importance [48]. Due to technology limitation, only certain qubits are directly connected. The topology limits the types of gates that can be performed on certain physical qubits or qubit-tuples. And because of NISQ restrictions, the final results may not be entirely correct (e.g., error rate $\geq$0.1% [91]), as a full-scale quantum error-correction is unlikely feasible with such limited number of qubits. Therefore, a quantum circuit is often executed many times (i.e., shots [60] or repetition [52]) so that the percentage of runs can converge to the correct answers.

B. OpenQASM Intermediate Representation

OpenQASM (*Open Quantum Assembly Language*) [31] is a low-level quantum intermediate representation (IR) for quantum instructions, similar to the traditional *Hardware Description Language* (HDL) like *Verilog* and *VHDL*. OpenQASM is the open-source unified low-level assembly language for IBM quantum machines publically available on cloud [60] that have been investigated and verified by many existing works [30, 59, 71, 83, 84, 85, 108].

TABLE I: OpenQASM supported gates.

Gates	Meaning	Gates	Meaning
U3	3 parameter 2 pulse 1-qubit	CY	Controlled Y
U2	2 parameter 1 pulse 1-qubit	SWAP	Swap
U1	1 parameter 0 pulse 1-qubit	CH	Controlled H
CX	Controlled-NOT	CCX	Toffoli
ID	Idle gate or identity	CSWAP	Fredkin
X	Pauli-X bit flip	CRX	Controlled RX rotation
Y	Pauli-Y bit and phase flip	CRY	Controlled RY rotation
Z	Pauli-Z phase flip	CRZ	Controlled RZ rotation
H	Hadamard	CU1	Controlled phase rotation
S	sqrt(Z) phase	CU3	Controlled U3
SDG	conjugate of sqrt(Z)	RXX	2-qubit XX rotation
T	sqrt(S) phase	RZZ	2-qubit ZZ rotation
TDG	conjugate of sqrt(S)	RCCX	Relative-phase CXX
RX	X-axis rotation	RC3X	Relative-phase 3-controlled X
RY	Y-axis rotation	C3X	3-controlled X
RZ	Z-axis rotation	C3XSQRTX	3-controlled sqrt(X)
CZ	Controlled phase	C4X	4-controlled X

Table I lists the types of gate that are defined in OpenQASM specification (i.e. the "`qelib1.inc`" header file) [31]. Within these gates, the first five gates — `U3`, `U2`, `U1`, `CX`, `ID` are *basis gates* that are natively executed by the hardware [60]. From `X` to `RZ` are *standard gates* defined atomically in OpenQASM. The standard gates will be translated into basis gates during the machine-specific assembling & mapping phase. The remaining gates from `CZ` to `C4X` are *composition gates* that are formed by standard gates. These gates are defined in `qelib1.inc` for the convenience of usage.

OpenQASM is becoming widely supported. A number of popular quantum software frameworks use OpenQASM as one of their output formats in order to be verified on IBM quantum machines [60], including Qiskit [61], Cirq [52], Scaffold [63], ProjectQ [104], etc. Figure 3 shows the landscape of OpenQASM IR. As is shown, for low-level optimization, mapping, scheduling, evaluation, profiling, and simulation, benchmark suites like QASMBench are quite necessary. In the following, we briefly discuss each of the front-end software frameworks.

1) Qiskit: The *Quantum Information Software Kit* (Qiskit) [61] is a quantum software platform developed by IBM. Qiskit is mainly based on Python but is also available in JavaScript and Swift [111]. The package comprises several tools, such as *qiskit-aer* for simulation, *qiskit-ignis* for hardware verification and noise addressing, and *qiskit-aqua* for example applications. OpenQASM can be generated from a Qiskit program using the API: "*QuantumCircuit.qasm()*".

2) Cirq: Cirq [52] is a QC software platform from Google. It is also based on Python. Despite claimed to be the interface for connecting to their 72-qubit Bristlecone quantum computer, this is not currently available to the general users. Cirq incorporates a local simulator for generic gates, and a *Xmon-Simulator* for simulating the native gate-set of Google's quantum computers. In addition, Cirq offers a function "*cirq.Circuit.to_qasm()*" in each circuit object that can generate OpenQASM code to run on an IBM's quantum machine. Note, not all Cirq code can be translated to OpenQASM.

3) Scaffold: Scaffold [63] is a quantum programming language embedded in the C/C++ programming language based on the LLVM compiler toolchain [72]. The major goal of Scaffold is to assist in efficiently developing quantum algorithms and advanced optimization, leveraging the existing LLVM

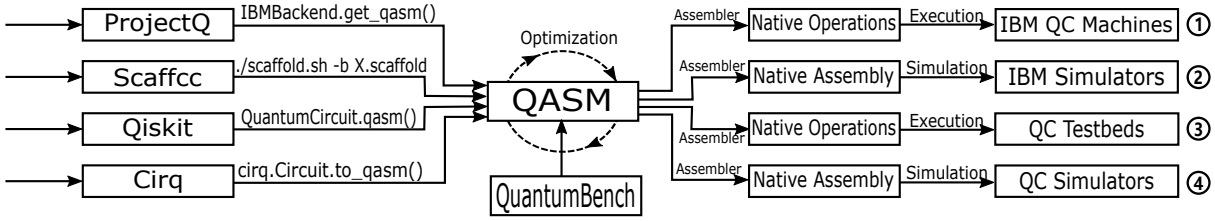

Fig. 3: OpenQASM Lanscape

compiling flow. Scaffold's compiler and scheduler — *Scaffcc* [63] can generate very complicated OpenQASM code as will be seen later. Scaffold program can be compiled by *Scaffcc* to native OpenQASM code using "*-b*" compiling option.

4) ProjectQ: ProjectQ is a quantum software platform developed by Steiger el al. from ETH Zürich [104]. Similar to Scaffold, ProjectQ does not have its own dedicated real quantum backend but relies on classical simulation. However, it provides a way to generate OpenQASM code so that the ProjectQ program can be validated on IBM testbeds through "`IBMBackend.get_qasm()`".

III. QASMBench

We introduce our QASMBench benchmark suite in this section, covering a wide-range of application domains. The benchmarks and routines are collected, generated and produced from a variety of open-source QC software packages, including [31, 52, 61, 63, 78, 80, 97], using the methodologies discussed in Section II-B. Depending on the number of qubits leveraged, we partition QASMBench into three groups:

- **Small-scale**: 2 to 5 qubits, summarizing in Table II.
- **Medium-scale**: 6 to 15 qubits, summarizing in Table III.
- **Large-scale**: $\geq$16 qubits, summarizing in Table IV.

For each item in Table II, III and IV, we list its name, brief description, and the algorithm category it belongs to (see Figure 2), which is based on [81] by adding the categories of quantum arithmetic, quantum machine learning and quantum communication. We show the number of qubits and gates utilized in the routines. The gate number here refers to *standard* OpenQASM gates (not basis gates or composition gates) as discussed in Section II-B, but excluding those gates in a branching "*if*" statement. As discussed, physical qubits in an NISQ device follow a certain topology; since the 2-qubit gates such as `CX` (i.e., `CNOT`) can only be performed between two adjacent qubits, a series of `SWAP` operations can be required to move the relevant qubits until they become directly-connected. This is an important issue in machine-specific mapping & optimization, implying a significant potential overhead. Consequently, we also list the number of `CX` gates in the tables. In the following, we briefly introduce each of the benchmarks.

A. Small-scale Benchmarks

W-state: W-state describes an entangled quantum state of three qubits following $|W\rangle = \frac{1}{\sqrt{3}}(|001\rangle + |010\rangle + |100\rangle)$. It is useful for representing a specific type of multipartite entanglement occurring in several quantum information applications [39]. It can be used for ensuring the robustness of ensemble-based quantum memories [45].

Adder: A ripple-carry adder [33, 110] takes two n-bit numbers as input and computes their sum in place while outputting a single bit implying the carry. We included a 4-qubit adder (see Figure 1) in small-scale, a 10-qubit adder in medium-scale, and a 18-qubit adder in large-scale.

Basis_change: This benchmark routine is from Google's OpenFermion library [78] for manipulating fermionic systems towards chemistry simulation on quantum computers. It shows how to use a quantum circuit to transform the single-particle basis of an linearly connected electronic structure so as to realize exact evolution under a random Hermitian one-body fermionic operator.

Basis_trotter: This routine benchmark is also from OpenFermion library [78]. It is designed to implement Trotter [17, 76] steps for an actual molecule *LiH* at equilibrium geometry, so as to simulate basis molecular Hamiltonians taking the following form $H = \sum_{pq} T_{pq} a_p^\dagger a_q + \sum_{pqrs} V_{pqrs} a_p^\dagger a_q a_r^\dagger a_s$. The circuit uses only 4 qubits but remains quite deep.

Cat_state: The cat state in QC, named after Schrödinger's cat [53], is a quantum state that is composed of two diametrically opposed conditions simultaneously, e.g., a cat be alive and dead at the same time. It is a circuit with merely 4 gates.

Deutsch: The Deutsch-Jozsa algorithm [35] is among the first examples to demonstrate that a quantum algorithm can achieve exponentially speedup over any possible deterministic classical algorithms. It shows a black box problem that can be efficiently resolved by QC, but would require a lot of queries to the black box using a deterministic classical computer.

Error_correction_d3: Quantum error correction (QEC) is to introduce redundancy to the original circuit by using additional physical qubits so that the state of the original circuit is effectively protected against decoherence. The surface code is often considered to be among the most promising QEC approaches as it can cope with error rates around 10^{-2} [47]. This benchmark performing distance-3 5-qubit surface code is from [80], which is the smallest code that can correct arbitrary single-qubit error. There are other QEC-related benchmarks in QASMBench, such as **qec_sm**, **qec_en** and **seca**.

Fredkin: The Fredkin gate or `CSWAP` gate [110] is a universal gate, which implies that any logical or arithmetic operations can be entirely built with Fredkin gates. It describes a circuit with three inputs and three outputs that keeps the first bit unchanged, and swaps the remaining two if and only if the first bit is 1. Fredkin gate is an OpenQASM composition gate.

Grover: Grover's algorithm [55] can be described as to search in a database, which offers quadratic speedup over classical

TABLE II: QASMBench Small-scale Benchmark.

Benchmark	Description	Algorithms	Qubits	Gates	CX	Ref
wstate	W-state preparation and assessment	Logical Operation	3	30	9	[31]
adder	Quantum ripple-carry adder	Quantum Arithmetic	4	23	10	[63]
basis_change	Transform the single-particle baseis of an linearly connected electronic structure	Quantum Simulation	3	53	10	[78]
basis_trotter	Implement Trotter steps for molecule LiH at equilibrium geometry	Quantum Simulation	4	1626	582	[78]
cat_state	Coherent superposition of two coherent states with opposite phase	Logical Operation	4	4	3	[63]
deutsch	Deutsch algorithm with 2 qubits for $f(x) = x$	Hidden Subgroup	2	5	1	[31]
error_correctiond3	Error correction with distance 3 and 5 qubits	Error Correction	5	114	49	[80]
fredkin	Controlled-swap gate	Logical Operation	3	19	8	[63]
grover	Grover's algorithm	Search and Optimization	2	16	2	[2]
hs4	Hidden subgroup problem	Hidden Subgroup	4	28	4	[63]
inverseqft	Performs an exact inversion of quantum Fourier tranform	Hidden Subgroup	4	8	0	[31]
ipea	Iterative phase estimation algorithm	Hidden Subgroup	2	68	30	[31]
iswap	An entangling swapping gate	Logical Operation	2	9	2	[31]
linearsolver	Solver for a linear equation of one qubit	Linear Equation	3	19	4	[20]
lpn	Learning parity with noise	Machine Learning	5	11	2	[97]
pea	Phase estimation algorithm	Hidden Subgroup	5	98	42	[31]
qec_sm	Repetition code syndrome measurement	Error Correction	5	5	4	[31]
qft	Quantum Fourier transform	Hidden Subgroupe	4	36	12	[31]
qec_en	Quantum repetition code encoder	Error Correction	5	25	10	[97]
teleportation	Quantum teleportation	Quantum Communication	3	8	2	[42]
toffoli	Toffoli gate	Logical Operation	3	18	6	[63]
variational	Variational ansatz for a Jellium Hamiltonian with a linear-swap network	Quantum Simulation	4	54	16	[78]
vqe_uccsd	Variational quantum eigensolver with UCCSD	Linear Equation	4	220	88	[63]
shor	Shor's algorithm	Hidden Subgroup	5	64	30	[61]
bell	Circuit equivalent to Bell inequality test	Logic Operation	4	33	7	[52]

TABLE III: QASMBench Medium-scale Benchmarks.

Benchmark	Description	Domain	Qubits	Gates	CX	Ref
adder	Quantum ripple-carry adder	Quantum Arithmetic	10	142	65	[31]
bv	Bernstein-Vazirani Algorithm	Hidden Subgroup	14	41	13	[31]
cc	Counterfeit coin finding problem	Search and Optimization	12	22	11	[31]
ising	Ising model simulation via QC	Quantum Simulation	10	480	90	[63]
multiply	Performing 3×5 in a quantum circuit	Quantum Arithmetic	13	98	40	[3]
qf21	Using quantum phase estimation to factor the number 21	Hidden Subgroup	15	311	115	[4]
qft	Quantum Fourier transform	Hidden Subgroup	15	540	210	[31]
qpe	Quantum phase estimation algorithm	Hidden Subgroup	9	123	43	[5]
sat	Boolean satisfiability problem via QC	Searching and Optimization	11	679	252	[31]
seca	Shor's error correction algorithm for teleportation	Error Correction	11	216	84	[6]
simons	Simon's algorithm	Hidden Subgroup	6	44	14	[7]
vqe_uccsd	Variational quantum eigensolver with UCCSD	Linear Equation	6	2282	1052	[63]
vqe_uccsd	Variational quantum eigensolver with UCCSD	Linear Equation	8	10808	5488	[63]
qaoa	Quantum approximate optimization algorithm	Search and Optimization	6	270	54	[52]
bb84	A quantum key distribution circuit	Quantum Communication	8	27	0	[52]
multipler	Quantum multipler	Quantum Arithmetic	15	574	246	[52]

TABLE IV: QASMBench Large-scale Benchmarks.

Benchmark	Description	Domain	Qubits	Gates	CX	Ref
ising	Ising model simulation via QC	Quantum Simulation	500	5494	998	[63]
ising	Ising model simulation via QC	Quantum Simulation	1000	10994	1998	[63]
bigadder	Quantum ripple-carry adder	Quantum Arithmetic	18	284	130	[31]
bv	Bernstein-Vazirani algorithm	Hidden Subgroup	19	56	18	[31]
cc	Counterfeit coin finding problem via QC	Hidden Subgroup	18	34	17	[31]
qft	Quantum Fourier tranform	Hidden Subgroup	20	970	380	[31]
square_root	Computing the square root of an number via amplitude amplification	Quantum Arithmetic	480	16064	6274	[63]
class_number	Compute the class group of a real quadratic number field	Hidden Subgroups	60052	31110504	12460637	[63]

searching approaches. It takes a black-box oracle realizing the following function: $f(x) = 1$ if $x = y$, $f(x) = 0$ if $x \neq y$, and find y within a randomly ordered sequence of N items using $O(\sqrt{N})$ operations and $O(N \log N)$ gates with probability $p \geq 2/3$. Grover's algorithm is appealing in that it determines with high probability the unique input given the output is pre-known, which can be seen as function inversion.

HS4: The *Hidden-subgroup* algorithm generalizes Shor's algorithm. It determines the subgroup within a group that is closely related to prime factorization and graph isomorphism [40, 66]. This benchmark is from Scaffold [63].

Pea & Ipea: The *phase-estimation-aglorithm* [86] is an algorithm to compute the eigenvalue $e^{2\pi i\theta}$ of a unitary operator operating on m qubits with an eigenvector $|\psi\rangle$ such that $U|\psi\rangle = e^{2\pi i\theta}|\psi\rangle$, $0 \leq \theta < 1$. The ipea benchmark here refers to iterative pea, which describes a 2-qubit system including a read-out ancilla bit, and a physical system bit [37].

Iswap: The iswap gate is an entangling swapping gate where the qubits obtain a phase of i if the state of the qubits is swapped [93].

Linearsolver: This benchmark realizes the HHL solver [58] for a linear equation of 1-qubit. The HHL algorithm estimates the outcome of a scalar measurement on the solution vector to a given linear equation system. We orginally tried to dump the multi-qubits HHL routine from Cirq [52], but encountered error in generating the OpenQASM code due to the mismatch between Google's native quantum IR and OpenQASM (e.g., OpenQASM does not support CCCRy gate).

LPN: This machine learning problem of learning a hidden parity function defined by the unknown binary string in the

presence of noise (known as *learning parity with noise* or LPN) is very promising for NISQ devices. It is believed to be computationally intractable through classical approaches [88], but for QC on NISQ devices, the presence of noise provides great potential advantages [32].

QFT & Inverseqft: The *Quantum Fourier Transform* or QFT [29] applies *Fourier* transformations to wave function amplitudes. It is a linear transformation over the quantum bits. It is the quantum analogue of the inverse discrete Fourier transform. QFT is an important component of many quantum algorithms, including Shor's algorithm, quantum phase estimation, hidden subgroup problem, etc.

Teleportation: Quantum teleportation [15, 59] is for exchanging information (rather than physical particle). The goal is to transmit a qubit without knowing its state from one location to another [19]. It involves classical communication and relies on pre-entanglement between qubits at the two locations.

Toffoli: Similar to the Fredkin gate, the Toffoli (i.e., `CCX`) gate [109] is another universal gate. It refers to a circuit with three inputs and three outputs that inverts the third qubit if the first two qubits are both 1, otherwise all bits stay unchanged. Toffoli gate is a composition gate defined in OpenQASM.

Variational: Variational quantum algorithm is to optimize a parameterized quantum circuit ansatz applied to some initial state for minimizing a cost function defined according to the output state [64, 79]. When applied in quantum simulation (simulation using QC rather than simulating QC in a classical computer), the goal of the algorithm is often to prepare ground states. The cost function is the expectation value of a Hamiltonian. If the initial state is $|\psi\rangle$, the Hamiltonian is H, the quantum circuit ansatz is $U(\vec{\theta})$, $\vec{\theta}$ are the variational parameters, then the cost function is $E(\vec{\theta}) = \langle\psi| U^{\dagger}(\vec{\theta}) H U(\vec{\theta}) |\psi\rangle$.

VQE_uccsd: Variational-Quantum-Eigensolver or VQE [89] is a quantum and classical hybrid algorithm for determining the eigenvalues of a matrix H. When applied in quantum simulation, H is often the Hamiltonian of a targeting system. In VQE, a quantum routine is run inside a classical optimization loop. VQE is one of the most promising algorithms for NISQ that can be adopted for simulating many-body systems such as molecular electronic structures. UCCSD stands for Unitary Coupled-Cluster Single and Double excitations [10, 54].

Shor: Shor's algorithm [100] is a polynomial-time QC algorithm for integer factorization. It searches the prime factors of an integer N using QC, showing theoretically exponential speedups over classical algorithms. Shor's algorithm is vital for quantum cryptography.

Bell: Bell's theorem is to state that no classical theory of local hidden variables can produce the predictions of quantum mechanics [12], which is about quantum entanglement. This routine simulates a circuit equivalent to a Bell inequality test.

B. Medium-scale Benchmarks

BV: The *Bernstein-Vazirani* algorithm [16] is an extension of the Deutsch-Josza algorithm [35]. It takes a black-box oracle realizing $f : \{0,1\}^n \rightarrow \{0,1\}$. It states $f(x)$ is a dot-product between x and a string $s \in \{0,1\}^n$, i.e., $f(x) = xs = x_1 s_1 + x_2 s_2 + \cdots + x_n s_n$. It finds s in a single query to the oracle.

CC: The *counterfeit coin* problem is a mathematical puzzle that attempts to find all false coins from a given bunch of coins using a balance scale [62]. This benchmark is to resolve the *cc* problem through QC.

Ising: The *quantum Ising model* [23, 70] consists of an array of quantum spins arranged in a certain lattice. The spins, which are expressed in quantum operators, can only interact with their neighbors. The phase transition is due to quantum fluctuation introduced by the transverse field.

SAT: The *Boolean Satisfiability* problem (SAT) determines whether there is an assignment of variables that satisfies a given Boolean function [11, 105] which is one of the first proven NP-complete problems. This benchmark shows the QC solution using 11 qubits.

Simon: Simon's algorithm [102] was among the first quantum algorithms to show exponential speedups over the best classical deterministic or probabilistic algorithm. It finds an unknown nonzero $s \in \{0,1\}^n$ that satisfies $f(x) = f(x \oplus s)$.

QAOA: The quantum approximate optimization algorithm, or QAOA [41, 113] is another hybrid quantum-classical variational algorithm that is quite suitable for NISQ. It is designed to solve combinatorial optimization problems. Similar to VQE, it also includes a quantum subroutine inside a classical searching loop. The quantum state is prepared according to a set of variational parameters. Based on the measurement outputs, the parameters are optimized by a classical computer.

BB84: *BB84* [14] is a quantum key distribution (QKD) protocol which is the first quantum cryptographic protocol [21] based on the no-cloning quantum laws for providing provably secure key generation. It relies on the fact that it is not possible to obtain information distinguishing two non-orthogonal states without disturbing the signal.

Multiply: This benchmark demonstrates arithmetic multiplication using a quantum circuit, which is an extension of the adder [46]. We use two implementations from [52] and [3].

C. Large-scale Benchmarks

Squarer_root: This routine relies on amplitude amplification [56] to find the square root of an n-bit number using Grover's search technique.

Class_number: This is a problem from computational algebraic number theory. It is to compute the class group of a real quadratic number field [57].

IV. Evaluation

We have evaluated the small-scale benchmarks listed in Table II on the *Burlington* quantum machine of the IBM Quantum Experience [60] which supports up to 5 qubits. The device topology and the error rates for the single-qubit `U2` gate and the `CX` (i.e., `CNOT`) gate are shown in Figure 29. We ran 1024 times (i.e., *shot=1024* which is the default value) on the backends. Figure 4 to 28 show the results. We also evaluated the medium benchmark routines in Table III on the *Melbourne*

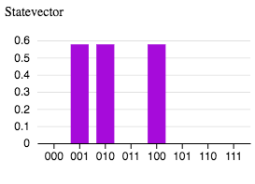

Fig. 4: wstate_n3

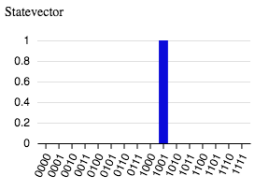

Fig. 5: adder_n4

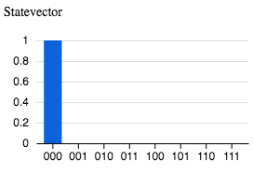

Fig. 6: basis_change_n3

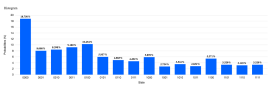

Fig. 7: basis_trotter_n4

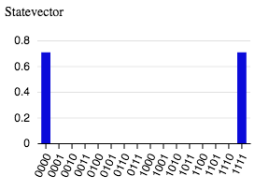

Fig. 8: cat_state_n4

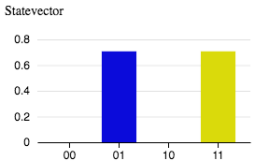

Fig. 9: deutsch_n2

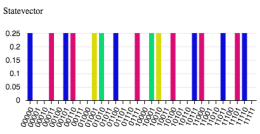

Fig. 10: error_correctiond3_n5

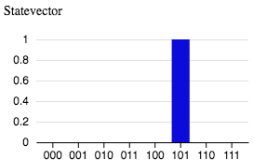

Fig. 11: fredkin_n3

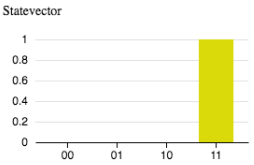

Fig. 12: grover_n2

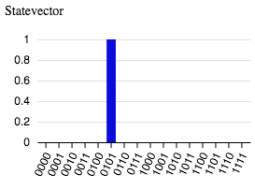

Fig. 13: hs4_n4

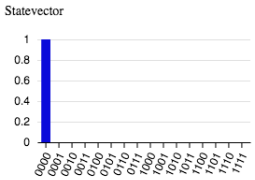

Fig. 14: inverseqft_n4

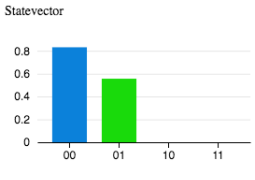

Fig. 15: ipea_n2

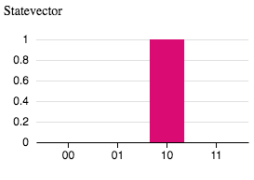

Fig. 16: iswap_n2

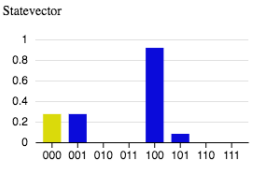

Fig. 17: linearsolver_n3

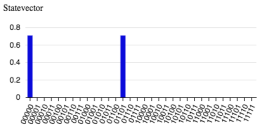

Fig. 18: lpn_n5

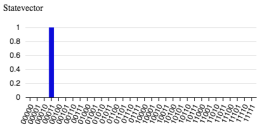

Fig. 19: pea_n5

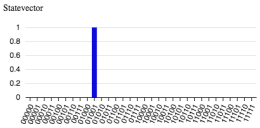

Fig. 20: qec_sm_n5

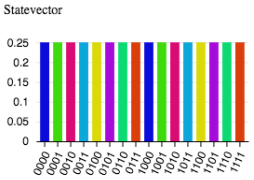

Fig. 21: qft_n4

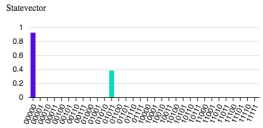

Fig. 22: qec_en_n5

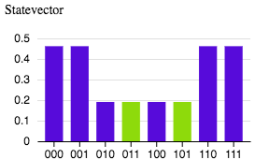

Fig. 23: teleportation_n3

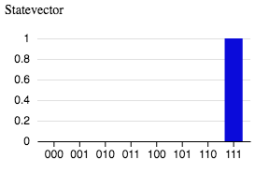

Fig. 24: toffoli_n3

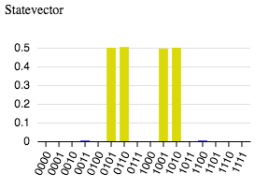

Fig. 25: variational_n4

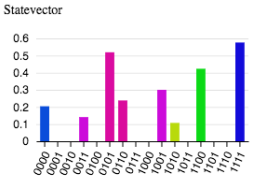

Fig. 26: vqe_uccsd_n4

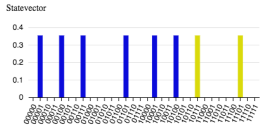

Fig. 27: shor_n5

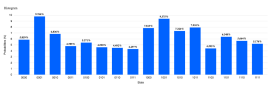

Fig. 28: bell_n4

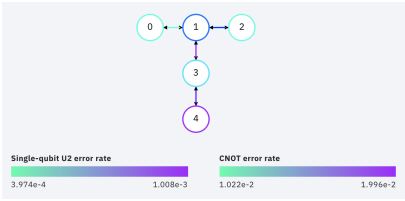

Fig. 29: IBM *Burlington* quantum system topology and error rates. The max U2 single-qubit error rate is ∼0.1%. The max CX error rate is ∼2%.

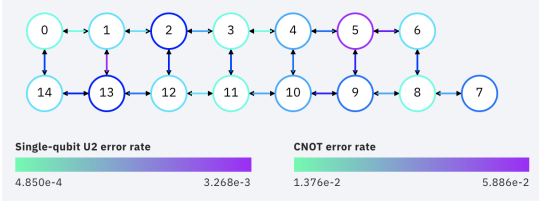

Fig. 30: IBM *Melbourne* quantum system topology and error rates. The max U2 single-qubit error rate is ∼0.3%. The max CX error rate is ∼6%.

quantum machine which supports up to 15 qubits. The device topology and error rates are shown in Figure 30. However, due to page limitation, we provide them in the GitHub repository.

V. Conclusion

In this paper, we propose a light-weighted, low-level and easy-to-use benchmark suite called QASMBench based on the OpenQASM quantum assembly language. It collects commonly seen quantum algorithms and routines from a variety of domains with distinct properties, serving the need for convenient quantum computing characterization and evaluation purposes for the computer system and architecture communities.

Acknowledgement

This research was supported by PNNL's Quantum Algorithms, Software, and Architectures (QUASAR) LDRD Initiative. It was also partially supported by the U.S. DOE Office of Science, Office of Advanced Scientific Computing Research, under award 66150: "CENATE - Center for Advanced Architecture Evaluation". The Pacific Northwest National Laboratory is operated by Battelle for the U.S. Department of Energy under contract DE-AC05-76RL01830.

REFERENCES

[1] Thorsten Altenkirch and Jonathan Grattage. A functional quantum programming language. In *20th Annual IEEE Symposium on Logic in Computer Science (LICS'05)*, pages 249–258. IEEE, 2005.

[2] ANAKIN, 2019. https://github.com/AgentANAKIN/Grover-s-Algorithm.

[3] ANAKIN, 2019. https://github.com/AgentANAKIN/Quantum-Multiplication.

[4] ANAKIN, 2019. https://github.com/AgentANAKIN/Quantum-Factoring-21.

[5] ANAKIN, 2019. https://github.com/AgentANAKIN/Quantum-Phase-Estimation.

[6] ANAKIN, 2019. https://github.com/AgentANAKIN/Shors-Error-Correction-Algorithm.

[7] ANAKIN, 2019. https://github.com/AgentANAKIN/Simon-s-Algorithm.

[8] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell, et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, 2019.

[9] Alán Aspuru-Guzik and Philip Walther. Photonic quantum simulators. *Nature physics*, 8(4):285–291, 2012.

[10] Panagiotis Kl Barkoutsos, Jerome F Gonthier, Igor Sokolov, Nikolaj Moll, Gian Salis, Andreas Fuhrer, Marc Ganzhorn, Daniel J Egger, Matthias Troyer, Antonio Mezzacapo, et al. Quantum algorithms for electronic structure calculations: Particle-hole hamiltonian and optimized wave-function expansions. *Physical Review A*, 98(2):022322, 2018.

[11] Carlos Barrón-Romero. Classical and quantum algorithms for the boolean satisfiability problem. *arXiv preprint arXiv:1510.02682*, 2015.

[12] John S Bell. On the einstein podolsky rosen paradox. *Physics Physique Fizika*, 1(3):195, 1964.

[13] Paul Benioff. The computer as a physical system: A microscopic quantum mechanical hamiltonian model of computers as represented by turing machines. *Journal of statistical physics*, 22(5):563–591, 1980.

[14] Charles H Bennett and Gilles Brassard. Quantum cryptography: Public key distribution and coin tossing. *arXiv preprint arXiv:2003.06557*, 2020.

[15] Charles H Bennett, Gilles Brassard, Claude Crépeau, Richard Jozsa, Asher Peres, and William K Wootters. Teleporting an unknown quantum state via dual classical and einstein-podolsky-rosen channels. *Physical review letters*, 70(13):1895, 1993.

[16] Ethan Bernstein and Umesh Vazirani. Quantum complexity theory. *SIAM Journal on computing*, 26(5):1411–1473, 1997.

[17] Dominic W Berry, Graeme Ahokas, Richard Cleve, and Barry C Sanders. Efficient quantum algorithms for simulating sparse hamiltonians. *Communications in Mathematical Physics*, 270(2):359–371, 2007.

[18] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549(7671):195–202, 2017.

[19] Dik Bouwmeester, Jian-Wei Pan, Klaus Mattle, Manfred Eibl, Harald Weinfurter, and Anton Zeilinger. Experimental quantum teleportation. *Nature*, 390(6660):575–579, 1997.

[20] BramDo. Quantum examples qasm, 2017. https://github.com/BramDo/quantum_examples_qasm.

[21] Cyril Branciard, Nicolas Gisin, Barbara Kraus, and Valerio Scarani. Security of two quantum cryptography protocols using the same four qubit states. *Physical Review A*, 72(3):032301, 2005.

[22] Michael Broughton, Guillaume Verdon, Trevor McCourt, Antonio J Martinez, Jae Hyeon Yoo, Sergei V Isakov, Philip Massey, Murphy Yuezhen Niu, Ramin Halavati, Evan Peters, et al. Tensorflow quantum: A software framework for quantum machine learning. *arXiv preprint arXiv:2003.02989*, 2020.

[23] Bikas K Chakrabarti, Amit Dutta, and Parongama Sen. *Quantum Ising phases and transitions in transverse Ising models*, volume 41. Springer Science & Business Media, 2008.

[24] Frederic T Chong. Quantum computing is getting real: Architecture, pl, and os roles in closing the gap between quantum algorithms and machines. In *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 285–285, 2018.

[25] Lukasz Cincio, Yiğit Subaşı, Andrew T Sornborger, and Patrick J Coles. Learning the quantum algorithm for state overlap. *New Journal of Physics*, 20(11):113022, 2018.

[26] Juan I Cirac and Peter Zoller. Quantum computations with cold trapped ions. *Physical review letters*, 74(20):4091, 1995.

[27] B David Clader, Bryan C Jacobs, and Chad R Sprouse. Preconditioned quantum linear system algorithm. *Physical review letters*, 110(25):250504, 2013.

[28] John Clarke and Frank K Wilhelm. Superconducting quantum bits. *Nature*, 453(7198):1031–1042, 2008.

[29] Don Coppersmith. An approximate fourier transform useful in quantum factoring. *arXiv preprint quant-ph/0201067*, 2002.

[30] Antonio D Córcoles, Abhinav Kandala, Ali Javadi-Abhari, Douglas T McClure, Andrew W Cross, Kristan Temme, Paul D Nation, Matthias Steffen, and JM Gambetta. Challenges and opportunities of near-term quantum computing systems. *arXiv preprint arXiv:1910.02894*, 2019.

[31] Andrew W Cross, Lev S Bishop, John A Smolin, and Jay M Gambetta. Open quantum assembly language. *arXiv preprint arXiv:1707.03429*, 2017. Repo: https://github.com/Qiskit/openqasm.

[32] Andrew W Cross, Graeme Smith, and John A Smolin. Quantum learning robust against noise. *Physical Review A*, 92(1):012327, 2015.

[33] Steven A Cuccaro, Thomas G Draper, Samuel A Kutin, and David Petrie Moulton. A new quantum ripple-carry addition circuit. *arXiv preprint quant-ph/0410184*, 2004.

[34] Poulami Das, Swamit S Tannu, Prashant J Nair, and Moinuddin Qureshi. A case for multi-programming quantum computers. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 291–303, 2019.

[35] David Deutsch and Richard Jozsa. Rapid solution of problems by quantum computation. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, 439(1907):553–558, 1992.

[36] David P DiVincenzo. The physical implementation of quantum computation. *Fortschritte der Physik: Progress of Physics*, 48(9-11):771–783, 2000.

[37] Miroslav Dobšíček, Göran Johansson, Vitaly Shumeiko, and Göran Wendin. Arbitrary accuracy iterative quantum phase estimation algorithm using a single ancillary qubit: A two-qubit benchmark. *Physical Review A*, 76(3):030306, 2007.

[38] Eugene F Dumitrescu, Alex J McCaskey, Gaute Hagen, Gustav R Jansen, Titus D Morris, T Papenbrock, Raphael C Pooser, David Jarvis Dean, and Pavel Lougovski. Cloud quantum computing of an atomic nucleus. *Physical review letters*, 120(21):210501, 2018.

[39] Wolfgang Dür, Guifre Vidal, and J Ignacio Cirac. Three qubits can be entangled in two inequivalent ways. *Physical Review A*, 62(6):062314, 2000.

[40] Mark Ettinger and Peter Hoyer. A quantum observable for the graph isomorphism problem. *arXiv preprint quant-ph/9901029*, 1999.

[41] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*, 2014.

[42] Serguei Fedortchenko. A quantum teleportation experiment for undergraduate students. *arXiv preprint arXiv:1607.02398*, 2016.

[43] Richard P Feynman. Simulating physics with computers. *Int. J. Theor. Phys*, 21(6/7), 1999.

[44] Mark Fingerhuth. Open-source quantum software projects, 2020. https://github.com/qosf/awesome-quantum-software.

[45] Michael Fleischhauer and Mikhail D Lukin. Quantum memory for photons: Dark-state polaritons. *Physical Review A*, 65(2):022314, 2002.

[46] G Florio and D Picca. Quantum implementation of elementary arithmetic operations. *arXiv preprint quant-ph/0403048*, 2004.

[47] Austin G Fowler, Matteo Mariantoni, John M Martinis, and Andrew N Cleland. Surface codes: Towards practical large-scale quantum computation. *Physical Review A*, 86(3):032324, 2012.

[48] Xiang Fu, Michiel Adriaan Rol, Cornelis Christiaan Bultink, J Van Someren, Nader Khammassi, Imran Ashraf, RFL Vermeulen, JC De Sterke, WJ Vlothuizen, RN Schouten, et al. An experimental microarchitecture for a superconducting quantum processor. In *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 813–825, 2017.

[49] Iulia M Georgescu, Sahel Ashhab, and Franco Nori. Quantum simulation. *Reviews of Modern Physics*, 86(1):153, 2014.

[50] Nicolas Gisin, Grégoire Ribordy, Wolfgang Tittel, and Hugo Zbinden. Quantum cryptography. *Reviews of modern physics*, 74(1):145, 2002.

[51] Pranav Gokhale, Jonathan M Baker, Casey Duckering, Natalie C Brown, Kenneth R Brown, and Frederic T Chong. Asymptotic improvements to quantum circuits via qutrits. In *Proceedings of the 46th International Symposium on Computer Architecture*, pages 554–566, 2019.

[52] Google. URL: https://github.com/quantumlib/Cirq.

[53] John Gribbin. *In search of Schrodinger's cat: Quantum physics and reality*. Bantam, 2011.

[54] Harper R Grimsley, Daniel Claudino, Sophia E Economou, Edwin Barnes, and Nicholas J Mayhall. Is the trotterized uccsd ansatz chemically well-defined? *Journal of Chemical Theory and Computation*, 2019.

[55] Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996.

[56] Lov K Grover. Quantum computers can search rapidly by using almost any transformation. *Physical Review Letters*, 80(19):4329, 1998.

[57] Sean Hallgren. Fast quantum algorithms for computing the unit group and class group of a number field. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 468–474, 2005.

[58] Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical review letters*, 103(15):150502, 2009.

[59] Ni-Ni Huang, Wei-Hao Huang, and Che-Ming Li. Identification of networking quantum teleportation on 14-qubit ibm universal quantum computer. *Scientific reports*, 10(1):1–12, 2020.

[60] IBM. Ibm quantum experience. URL: https://quantum-computing.ibm.com/.

[61] IBM. Qiskit: Elements for building a quantum future. URL: https://github.com/Qiskit/qiskit.

[62] Kazuo Iwama, Harumichi Nishimura, Rudy Raymond, and Junichi Teruyama. Quantum counterfeit coin problems. In *International Symposium on Algorithms and Computation*, pages 85–96. Springer, 2010.

[63] Ali JavadiAbhari, Shruti Patil, Daniel Kudrow, Jeff Heckey, Alexey Lvov, Frederic T Chong, and Margaret Martonosi. Scaffcc: a framework for compilation and analysis of quantum computing programs. In *Proceedings of the 11th ACM Conference on Computing Frontiers*, pages 1–10, 2014. Repo: https://github.com/epiqc/ScaffCC.

[64] Tyson Jones, Suguru Endo, Sam McArdle, Xiao Yuan, and Simon C Benjamin. Variational quantum algorithms for discovering hamiltonian spectra. *Physical Review A*, 99(6):062304, 2019.

[65] Stephen Jordan. Quantum algorithm zoo. URL: https://quantumalgorithmzoo.org/.

[66] Richard Jozsa. Quantum factoring, discrete logarithms, and the hidden subgroup problem. *Computing in science & engineering*, 3(2):34–43, 2001.

[67] Abhinav Kandala, Antonio Mezzacapo, Kristan Temme, Maika Takita, Markus Brink, Jerry M Chow, and Jay M Gambetta. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nature*, 549(7671):242–246, 2017.

[68] Iordanis Kerenidis and Anupam Prakash. Quantum recommendation systems. *arXiv preprint arXiv:1603.08675*, 2016.

[69] H Jeff Kimble. The quantum internet. *Nature*, 453(7198):1023–1030, 2008.

[70] Henning Labuhn, Daniel Barredo, Sylvain Ravets, Sylvain De Léséleuc, Tommaso Macrì, Thierry Lahaye, and Antoine Browaeys. Tunable two-dimensional arrays of single rydberg atoms for realizing quantum ising models. *Nature*, 534(7609):667–670, 2016.

[71] Ryan LaRose. Overview and comparison of gate level quantum software platforms. *Quantum*, 3:130, 2019.

[72] Chris Lattner and Vikram Adve. Llvm: A compilation framework for lifelong program analysis & transformation. In *International Symposium on Code Generation and Optimization, 2004. CGO 2004.*, pages 75–86. IEEE, 2004.

[73] Dietrich Leibfried, Rainer Blatt, Christopher Monroe, and David Wineland. Quantum dynamics of single trapped ions. *Reviews of Modern Physics*, 75(1):281, 2003.

[74] Gushu Li, Yufei Ding, and Yuan Xie. Tackling the qubit mapping problem for nisq-era quantum devices. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 1001–1014, 2019.

[75] Gushu Li, Yufei Ding, and Yuan Xie. Towards efficient superconducting quantum processor architecture design. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 1031–1045, 2020.

[76] Seth Lloyd. Universal quantum simulators. *Science*, pages 1073–1078, 1996.

[77] R Maurand, X Jehl, D Kotekar-Patil, A Corna, H Bohuslavskyi, R Laviéville, L Hutin, S Barraud, M Vinet, M Sanquer, et al. A cmos silicon spin qubit. *Nature communications*, 7(1):1–6, 2016.

[78] Jarrod McClean, Nicholas Rubin, Kevin Sung, Ian David Kivlichan, Xavier Bonet-Monroig, Yudong Cao, Chengyu Dai, Eric Schuyler Fried, Craig Gidney, Brendan Gimby, et al. Openfermion: the electronic structure package for quantum computers. *Quantum Science and Technology*, 2020. Repo: https://github.com/quantumlib/OpenFermion-Cirq.

[79] Jarrod R McClean, Jonathan Romero, Ryan Babbush, and Alán Aspuru-Guzik. The theory of variational hybrid quantum-classical algorithms. *New Journal of Physics*, 18(2):023023, 2016.

[80] Kristel Michielsen, Madita Nocon, Dennis Willsch, Fengping Jin, Thomas Lippert, and Hans De Raedt. Benchmarking gate-based quantum computers. *Computer Physics Communications*, 220:44–55, 2017.

[81] Ashley Montanaro. Quantum algorithms: an overview. *npj Quantum Information*, 2(1):1–8, 2016.

[82] WJ Munro, KA Harrison, AM Stephens, SJ Devitt, and Kae Nemoto. From quantum multiplexing to high-performance quantum networking. *Nature Photonics*, 4(11):792, 2010.

[83] Prakash Murali, Jonathan M Baker, Ali Javadi-Abhari, Frederic T Chong, and Margaret Martonosi. Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 1015–1029, 2019.

[84] Prakash Murali, Norbert Matthias Linke, Margaret Martonosi, Ali Javadi Abhari, Nhung Hong Nguyen, and Cinthia Huerta Alderete. Full-stack, real-system quantum computer studies: architectural comparisons and design insights. In *Proceedings of the 46th International Symposium on Computer Architecture*, pages 527–540, 2019.

[85] Prakash Murali, David C McKay, Margaret Martonosi, and Ali Javadi-Abhari. Software mitigation of crosstalk on noisy intermediate-scale quantum computers. *arXiv preprint arXiv:2001.02826*, 2020.

[86] Michael A Nielsen and Isaac Chuang. Quantum computation and quantum information, 2002.

[87] Jeremy L O'brien, Akira Furusawa, and Jelena Vučković. Photonic quantum technologies. *Nature Photonics*, 3(12):687, 2009.

[88] Daniel K Park, June-Koo K Rhee, and Soonchil Lee. Noise-tolerant parity learning with one quantum bit. *Physical Review A*, 97(3):032327, 2018.

[89] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J Love, Alán Aspuru-Guzik, and Jeremy L O'brien. A variational eigenvalue solver on a photonic quantum processor. *Nature communications*, 5:4213, 2014.

[90] Jarryd J Pla, Kuan Y Tan, Juan P Dehollain, Wee H Lim, John JL Morton, David N Jamieson, Andrew S Dzurak, and Andrea Morello. A single-atom electron spin qubit in silicon. *Nature*, 489(7417):541–545, 2012.

[91] John Preskill. Quantum computing in the nisq era and beyond. *Quantum*, 2:79, 2018.

[92] Quantiki. List of qc simulators, 2020. https://www.quantiki.org/wiki/list-qc-simulators.

[93] SE Rasmussen and NT Zinner. Simple implementation of high fidelity controlled-i swap gates and quantum circuit exponentiation of non-hermitian gates. *arXiv preprint arXiv:2002.11728*, 2020.

[94] Patrick Rebentrost, Brajesh Gupt, and Thomas R Bromley. Quantum computational finance: Monte carlo pricing of financial derivatives. *Physical Review A*, 98(2):022321, 2018.

[95] Rigetti. Rigetti: Think quantum. URL: https://rigetti.com/.

[96] Chad Rigetti, Jay M Gambetta, Stefano Poletto, BLT Plourde, Jerry M Chow, AD Córcoles, John A Smolin, Seth T Merkel, JR Rozen, George A Keefe, et al. Superconducting qubit in a waveguide cavity with a coherence time approaching 0.1 ms. *Physical Review B*, 86(10):100506, 2012.

[97] Gonçalo Sampaio. Code in qasm for quantum circuits and algorithms., 2017. https://github.com/sampaio96/Quantum-Computing.

[98] Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione. An introduction to quantum machine learning. *Contemporary Physics*, 56(2):172–185, 2015.

[99] Yunong Shi, Nelson Leung, Pranav Gokhale, Zane Rossi, David I Schuster, Henry Hoffmann, and Frederic T Chong. Optimized compilation of aggregated instructions for realistic quantum computers. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 1031–1044, 2019.

[100] Peter W Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134. Ieee, 1994.

[101] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.

[102] Daniel R Simon. On the power of quantum computation. *SIAM journal on computing*, 26(5):1474–1483, 1997.

[103] Kaitlin N Smith and Mitchell A Thornton. A quantum computational compiler and design tool for technology-specific targets. In *Proceedings of the 46th International Symposium on Computer Architecture*, pages 579–588, 2019.

[104] Damian S Steiger, Thomas Häner, and Matthias Troyer. Projectq: an open source software framework for quantum computing. *Quantum*, 2:49, 2018.

[105] Juexiao Su, Tianheng Tu, and Lei He. A quantum annealing approach for boolean satisfiability problem. In *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2016.

[106] Swamit S Tannu and Moinuddin Qureshi. Ensemble of diverse mappings: Improving reliability of quantum computers by orchestrating dissimilar mistakes. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 253–265, 2019.

[107] Swamit S Tannu and Moinuddin K Qureshi. Mitigating measurement errors in quantum computers by exploiting state-dependent bias. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 279–290, 2019.

[108] Swamit S Tannu and Moinuddin K Qureshi. Not all qubits are created equal: a case for variability-aware policies for nisq-era quantum computers. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 987–999, 2019.

[109] Tommaso Toffoli. Reversible computing. In *International Colloquium on Automata, Languages, and Programming*, pages 632–644. Springer, 1980.

[110] Vlatko Vedral, Adriano Barenco, and Artur Ekert. Quantum networks for elementary arithmetic operations. *Physical Review A*, 54(1):147, 1996.

[111] Michael Wilde, Mihael Hategan, Justin M Wozniak, Ben Clifford, Daniel S Katz, and Ian Foster. Swift: A language for distributed parallel scripting. *Parallel Computing*, 37(9):633–652, 2011.

[112] Stefan Woerner and Daniel J Egger. Quantum risk analysis. *npj Quantum Information*, 5(1):1–8, 2019.

[113] Leo Zhou, Sheng-Tao Wang, Soonwon Choi, Hannes Pichler, and Mikhail D Lukin. Quantum approximate optimization algorithm: performance, mechanism, and implementation on near-term devices. *arXiv preprint arXiv:1812.01041*, 2018.