Extracting Success

Yipeng Huang

Rutgers University

November 16, 2020

Where we are in the semester

Full stack quantum computer engineering

1. Algorithms: QAOA & VQE
2. Programming languages, assertions, stabilizers
3. Google Cirq, IBM Qiskit
4. Quantum circuit simulation and quantum supremacy
5. Extracting success: quantum computer architecture
6. Prototypes: quantum computer microarchitecture

- Programming assignments (2×25 points)
- Seminar presentations (2×25 points)

Compiling from a high-level program to hardware

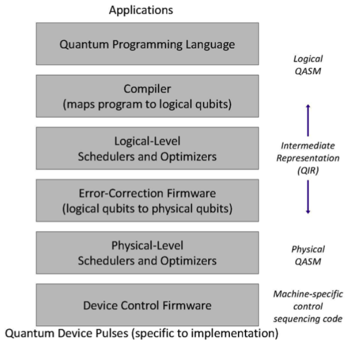
Goals:

1. Correctness: maximizing probability of success!
2. Ease of programming?
3. Compatibility between hardware implementations?

Extreme device/resource constraints:

1. Native gate set
2. Device topology
3. Hardware noise
4. Parallelism constraints

Compiling from a high-level program to hardware



Applications
Quantum Programming Language
Compiler (maps program to logical qubits)
Logical-Level Schedulers and Optimizers
Error-Correction Firmware (logical qubits to physical qubits)
Physical-Level Schedulers and Optimizers
Device Control Firmware
Quantum Device Pulses (specific to implementation)
Logical QASM
Intermediate Representation (QIR)
Physical QASM
Machine-specific control sequencing code

FIGURE 6.1 A generic tool flow for quantum programming. A quantum program is implemented in a domain-specific language (DSL) and then translated into hardware instructions after undergoing a series of compiler transformations and optimizations. A quantum intermediate representation (QIR) of the program can serve as a logical-level analog to conventional assembly code. For programs running on error-corrected qubits, the compiler would link in low-level QEC libraries into the code, transforming the logical qubit operations, to the physical operations on a number of qubits. The qubits of this “expanded” quantum program are then mapped onto a specific hardware implementation accounting for the specific gate operations and connectivity available. At the lowest level, the operations on physical qubits will be generated as instructions of the quantum control processor that orchestrate the specific control pulses (e.g., microwave or optical) required. For more detailed discussion of quantum computer software architectures see [Chong, Frederic T., Diana Franklin, and Margaret Martonosi. "Programming languages and compiler design for realistic quantum hardware." *Nature* 549, no. 7671 (2017): 180.] and [Häner, Thomas, Damian S. Steiger, Krysta Svore, and Matthias Troyer. "A software methodology for compiling quantum programs." *Quantum Science and Technology* 3, no. 2 (2018): 020501.].

Figure: Credit: [Córcoles et al., 2020]

Compiling from a high-level program to hardware

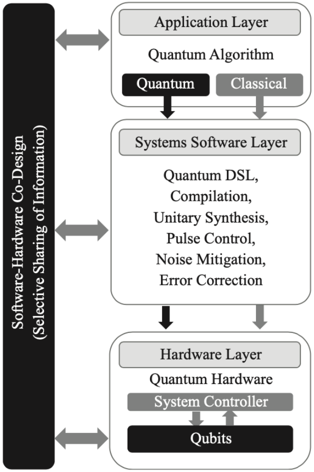
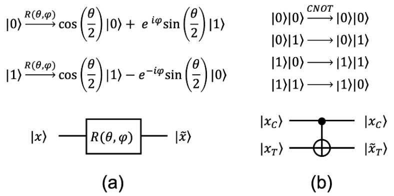


Figure 4.1: Selective sharing of information allows algorithms to use limited resource in NISQ hardware most efficiently.

Figure: Credit: [Ding and Chong, 2020]

Native gate set



$$|0\rangle \xrightarrow{R(\theta,\varphi)} \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\varphi}\sin\left(\frac{\theta}{2}\right)|1\rangle$$

$$|1\rangle \xrightarrow{R(\theta,\varphi)} \cos\left(\frac{\theta}{2}\right)|1\rangle - e^{-i\varphi}\sin\left(\frac{\theta}{2}\right)|0\rangle$$

$$|0\rangle|0\rangle \xrightarrow{CNOT} |0\rangle|0\rangle$$
$$|0\rangle|1\rangle \longrightarrow |0\rangle|1\rangle$$
$$|1\rangle|0\rangle \longrightarrow |1\rangle|1\rangle$$
$$|1\rangle|1\rangle \longrightarrow |1\rangle|0\rangle$$

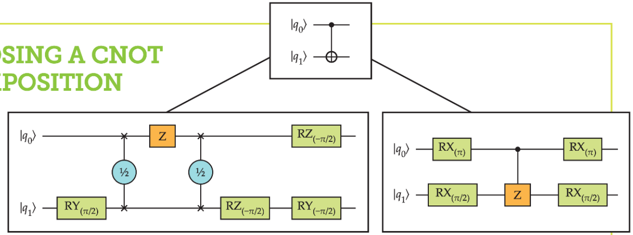
FIG. 1. The rotation and controlled-NOT (CNOT) gates are an example of a universal quantum gate family when available on all qubits, with explicit evolution (above) and quantum circuit block schematics (below). (a) The single-qubit rotation gate $R(\theta, \phi)$, with two continuous parameters θ and ϕ, evolves input qubit state $|x\rangle$ to output state $|\tilde{x}\rangle$. (b) The CNOT (or reversible XOR) gate on two qubits evolves two (control and target) input qubit states $|x_C\rangle$ and $|x_T\rangle$ to output states $|\tilde{x}_C = x_C\rangle$ and $|\tilde{x}_T = x_C \oplus x_T\rangle$, where $\oplus$ is addition modulo 2, or equivalently the XOR operation.

Figure: Credit: [Alexeev et al., 2020]

- Clifford + T ISA is sensible for an error-corrected machine
- But for NISQ machine, best two-qubit gate is dependent on native gate set

Native gate set

BOX 2. CHOOSING A CNOT GATE DECOMPOSITION

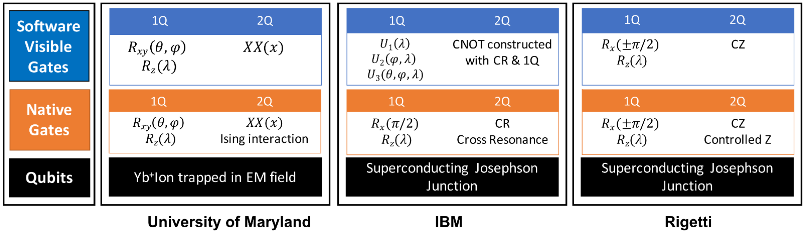


A logical controlled-NOT, or CNOT, gate is an entangling operation that flips the target qubit between 1 and 0 if the control qubit is in the 1 state. It must be decomposed into a sequence of native quantum gates for the qubit technology to perform the gate operation on the specific qubit system. Two possible decompositions are shown, where RX, RY, and RZ denote rotations around the *x*-, *y*- and *z*-axes, respectively. A system performance simulation could provide metrics to help choose which CNOT to incorporate into the specific design.

Depending on the fidelity of the single- and two-qubit gates in the circuits and on their speed, researchers may want to choose only one to implement the logical CNOT in the system. Depending on the performance of the qubits available at a particular point in the execution of the algorithm, it is also possible to choose a different logical CNOT sequence.

Figure: Credit: [Matsuura et al., 2019]

Native gate set



	University of Maryland 1Q	University of Maryland 2Q	IBM 1Q	IBM 2Q	Rigetti 1Q	Rigetti 2Q
Software Visible Gates	$R_{xy}(\theta,\varphi)$ $R_z(\lambda)$	$XX(x)$	$U_1(\lambda)$ $U_2(\varphi,\lambda)$ $U_3(\theta,\varphi,\lambda)$	CNOT constructed with CR & 1Q	$R_x(\pm\pi/2)$ $R_z(\lambda)$	CZ
Native Gates	$R_{xy}(\theta,\varphi)$ $R_z(\lambda)$	$XX(x)$ Ising interaction	$R_x(\pi/2)$ $R_z(\lambda)$	CR Cross Resonance	$R_x(\pm\pi/2)$ $R_z(\lambda)$	CZ Controlled Z
Qubits	Yb⁺Ion trapped in EM field		Superconducting Josephson Junction		Superconducting Josephson Junction	

Figure 1. Hardware qubit technology, native gate set, and software-visible gate set in the systems used in our study. Each qubit technology lends itself to a set of native gates. For programming, vendors expose these gates in a software-visible interface or construct composite gates with multiple native gates.

Figure: Credit: [Murali et al., 2019]

Two qubit gates remain dominant sources of errors.

Compiling from a high-level program to hardware

Goals:

1. Correctness: maximizing probability of success!
2. Ease of programming?
3. Compatibility between hardware implementations?

Extreme device/resource constraints:

1. Native gate set
2. Device topology
3. Hardware noise
4. Parallelism constraints

Device topology

- ion trap qubits: fully connected topology
- superconducting qubits: arbitrary qubits cannot directly interact; needs chain of swap gates

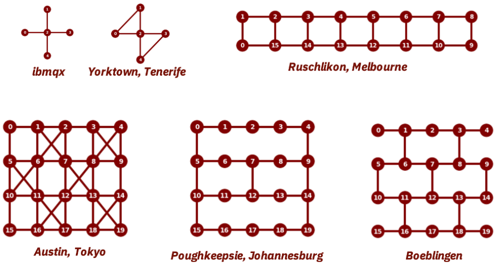
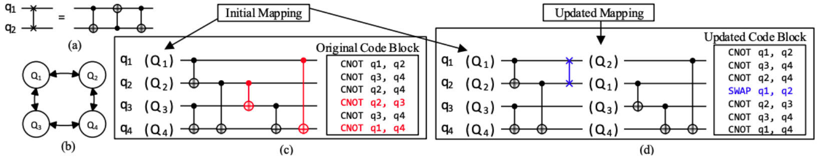


Fig. 1. Examples of several IBM cloud accessible devices. The top left 5-qubit device was the first one made available via the IBM Quantum Experience [40]. The one to the right of it was made available after including additional entangling gates between two pairs of qubits. A 16-qubit device was made available approximately a year after the first device. The devices in the bottom row show three variations of 20-qubit devices available to members of the IBM Q Network [41].

Figure: Credit: [Córcoles et al., 2020]

Device topology



Original Code Block

```
CNOT q1, q2
CNOT q3, q4
CNOT q2, q4
CNOT q2, q3
CNOT q3, q4
CNOT q1, q4
```

Updated Code Block

```
CNOT q1, q2
CNOT q3, q4
CNOT q2, q4
SWAP q1, q2
CNOT q2, q3
CNOT q3, q4
CNOT q1, q4
```

Figure 3. (a) SWAP Gate Decomposition, (b) Physical Qubit Coupling Graph Example, (c) Original Quantum Circuit, (d) Updated Hardware-Compliant Quantum Circuit

Figure: Credit: [Li et al., 2019]

Superconducting qubits: arbitrary qubits cannot directly interact; needs chain of swap gates

Device topology

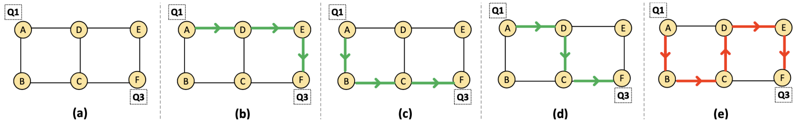


Figure 3. (a) Layout of a 6-qubit quantum computer, (b)-(e) are possible routes from A to F. Note that options (b)(c)(d) have an identical number of swaps and (e) incur higher swaps. An intelligent policy would choose one from (b)(c)(d).

Figure: Credit: [Tannu and Qureshi, 2019]

Superconducting qubits: arbitrary qubits cannot directly interact; needs chain of swap gates

Compiling from a high-level program to hardware

Goals:

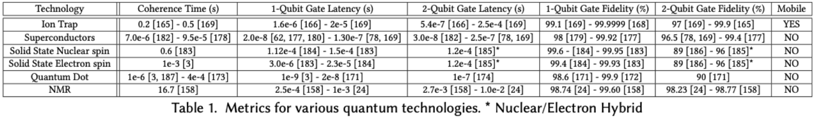
1. Correctness: maximizing probability of success!
2. Ease of programming?
3. Compatibility between hardware implementations?

Extreme device/resource constraints:

1. Native gate set
2. Device topology
3. Hardware noise
4. Parallelism constraints

Hardware noise

1. Decoherence error
2. Gate error (imprecise control of single qubit, two qubit gates)
3. Measurement error



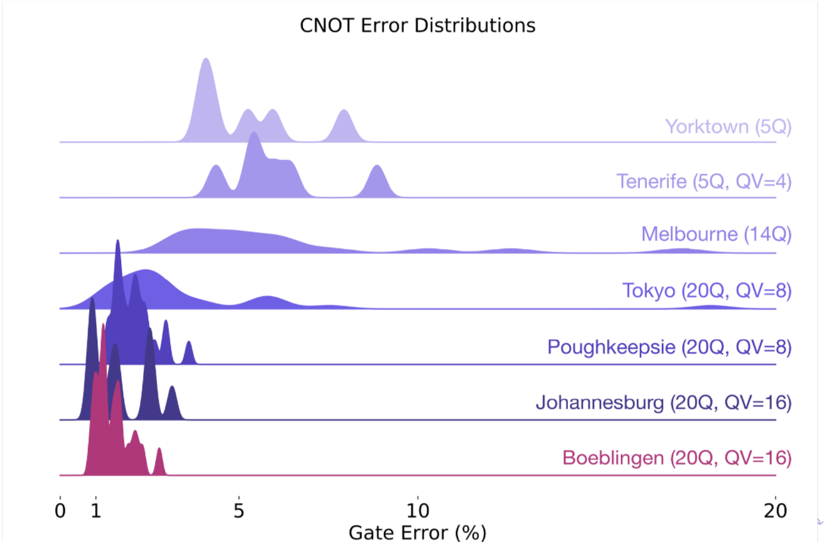
Technology	Coherence Time (s)	1-Qubit Gate Latency (s)	2-Qubit Gate Latency (s)	1-Qubit Gate Fidelity (%)	2-Qubit Gate Fidelity (%)	Mobile
Ion Trap	0.2 [165] - 0.5 [169]	1.6e-6 [166] - 2e-5 [169]	5.4e-7 [166] - 2.5e-4 [169]	99.1 [169] - 99.9999 [168]	97 [169] - 99.9 [165]	YES
Superconductors	7.0e-6 [182] - 9.5e-5 [178]	2.0e-8 [62, 177, 180] - 1.30e-7 [78, 169]	3.0e-8 [182] - 2.5e-7 [78, 169]	98 [179] - 99.92 [177]	96.5 [78, 169] - 99.4 [177]	NO
Solid State Nuclear spin	0.6 [183]	1.12e-4 [184] - 1.5e-4 [183]	1.2e-4 [185]*	99.6 - [184] - 99.95 [183]	89 [186] - 96 [185]*	NO
Solid State Electron spin	1e-3 [3]	3.0e-6 [183] - 2.3e-5 [184]	1.2e-4 [185]*	99.4 [184] - 99.93 [183]	89 [186] - 96 [185]*	NO
Quantum Dot	1e-6 [3, 187] - 4e-4 [173]	1e-9 [3] - 2e-8 [171]	1e-7 [174]	98.6 [171] - 99.9 [172]	90 [171]	NO
NMR	16.7 [158]	2.5e-4 [158] - 1e-3 [24]	2.7e-3 [158] - 1.0e-2 [24]	98.74 [24] - 99.60 [158]	98.23 [24] - 98.77 [158]	NO

Table 1. Metrics for various quantum technologies. * Nuclear/Electron Hybrid

Figure: Credit: [Resch and Karpuzcu, 2019]

Hardware noise

1. Decoherence error
2. Gate error (imprecise control of single qubit, two qubit gates)
3. Measurement error



Hardware noise

Stochastic, uncorrelated noise

	Quantum noise mixtures (Pauli errors)	Quantum noise channels
Pauli-X type	Bit flip noise	Amplitude damping noise (related to T1 time)
Pauli-Z type	Phase flip noise	Phase damping noise (related to T2 time)
Combinations	Symmetric / asymmetric depolarizing noise	Generalized amplitude damping
Simulation technique	Can model as probabilistic ensembles of state vectors	Requires density matrix representation

Table: Summary of canonical quantum noise models.

Bit flip noise channel

$$|0\rangle \rightarrow \mathit{BitFlip}(0.64) \rightarrow \begin{cases} P(|0\rangle) = 0.64 \\ P(|1\rangle) = 0.36 \end{cases}$$

We represent such a mixture of quantum states as a density matrix:

$$\begin{aligned}
&0.64\,|0\rangle\langle 0| + 0.36\,|1\rangle\langle 1| \\
&= 0.64 \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix} + 0.36 \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \end{bmatrix} \\
&= 0.64 \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} + 0.36 \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} 0.64 & 0 \\ 0 & 0.36 \end{bmatrix}
\end{aligned}$$

(Conventions from [Nielsen and Chuang, 2011, Chapter 8.3])

Density matrix representation

$$0.64\,|0\rangle\langle 0| + 0.36\,|1\rangle\langle 1| = \begin{bmatrix} 0.64 & 0 \\ 0 & 0.36 \end{bmatrix}$$

More general representation:
$\rho = \sum_j p_j \,|\psi_j\rangle\langle\psi_j|$

Quantum (noise) channel

A quantum channel $\mathcal{E}(\rho)$ acts on mixed state ρ:

$$\mathcal{E}(\rho) = \sum_k E_k \rho E_k^\dagger$$

Bit flip noise channel

The bit flip channel flips the state of a qubit with probability $1 - p$. It has two elements:

$$E_0 = \sqrt{p}I = \sqrt{p}\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}$$
$$E_1 = \sqrt{1-p}X = \sqrt{p}\begin{bmatrix}0 & 1\\1 & 0\end{bmatrix}$$

Bit flip noise channel

The bit flip noise channel $\mathcal{E}_{bitflip}(0.64)$ acts on the $|0\rangle$ state like so:

$$\mathcal{E}_{bitflip}(\begin{bmatrix}1 & 0\\0 & 0\end{bmatrix})$$

$$= \sum_k E_k \begin{bmatrix}1 & 0\\0 & 0\end{bmatrix} E_k^\dagger$$

$$= 0.8 \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} \begin{bmatrix}1 & 0\\0 & 0\end{bmatrix} 0.8 \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} + 0.6 \begin{bmatrix}0 & 1\\1 & 0\end{bmatrix} \begin{bmatrix}1 & 0\\0 & 0\end{bmatrix} 0.6 \begin{bmatrix}0 & 1\\1 & 0\end{bmatrix}$$

$$= \begin{bmatrix}0.64 & 0\\0 & 0.36\end{bmatrix}$$

Phase flip noise channel

The phase flip channel flips the phase of a qubit with probability $1 - p$. It has two elements:

$$E_0 = \sqrt{p}I = \sqrt{p}\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}$$

$$E_1 = \sqrt{1-p}Z = \sqrt{p}\begin{bmatrix}1 & 0\\0 & -1\end{bmatrix}$$

Hardware noise

	Quantum noise mixtures (Pauli errors)	Quantum noise channels
Pauli-X type	Bit flip noise	Amplitude damping noise (related to T1 time)
Pauli-Z type	Phase flip noise	Phase damping noise (related to T2 time)
Combinations	Symmetric / asymmetric depolarizing noise	Generalized amplitude damping
Simulation technique	Can model as probabilistic ensembles of state vectors	Requires density matrix representation

Table: Summary of canonical quantum noise models.

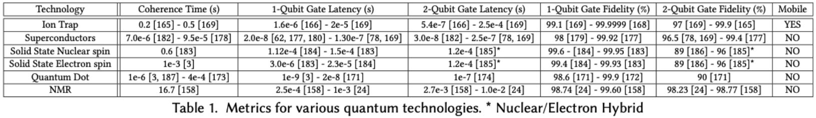
Amplitude damping noise channel

The amplitude damping channel leaves $|0\rangle$ alone while probabilistically flipping $|1\rangle$. It has two elements:

$$E_0 = \begin{bmatrix} 1 & 0 \\ 0 & \sqrt{1-\gamma} \end{bmatrix}$$
$$E_1 = \begin{bmatrix} 0 & \sqrt{\gamma} \\ 0 & 0 \end{bmatrix}$$

Hardware noise

1. Decoherence error
2. Gate error (imprecise control of single qubit, two qubit gates)
3. Measurement error



Technology	Coherence Time (s)	1-Qubit Gate Latency (s)	2-Qubit Gate Latency (s)	1-Qubit Gate Fidelity (%)	2-Qubit Gate Fidelity (%)	Mobile
Ion Trap	0.2 [165] - 0.5 [169]	1.6e-6 [166] - 2e-5 [169]	5.4e-7 [166] - 2.5e-4 [169]	99.1 [169] - 99.9999 [168]	97 [169] - 99.9 [165]	YES
Superconductors	7.0e-6 [182] - 9.5e-5 [178]	2.0e-8 [62, 177, 180] - 1.30e-7 [78, 169]	3.0e-8 [182] - 2.5e-7 [78, 169]	98 [179] - 99.92 [177]	96.5 [78, 169] - 99.4 [177]	NO
Solid State Nuclear spin	0.6 [183]	1.12e-4 [184] - 1.5e-4 [183]	1.2e-4 [185]*	99.6 - [184] - 99.95 [183]	89 [186] - 96 [185]*	NO
Solid State Electron spin	1e-3 [3]	3.0e-6 [183] - 2.3e-5 [184]	1.2e-4 [185]*	99.4 [184] - 99.93 [183]	89 [186] - 96 [185]*	NO
Quantum Dot	1e-6 [3, 187] - 4e-4 [173]	1e-9 [3] - 2e-8 [171]	1e-7 [174]	98.6 [171] - 99.9 [172]	90 [171]	NO
NMR	16.7 [158]	2.5e-4 [158] - 1e-3 [24]	2.7e-3 [158] - 1.0e-2 [24]	98.74 [24] - 99.60 [158]	98.23 [24] - 98.77 [158]	NO

Table 1. Metrics for various quantum technologies. * Nuclear/Electron Hybrid

Figure: Credit: [Resch and Karpuzcu, 2019]

But what about correlated noise events?

Compiling from a high-level program to hardware

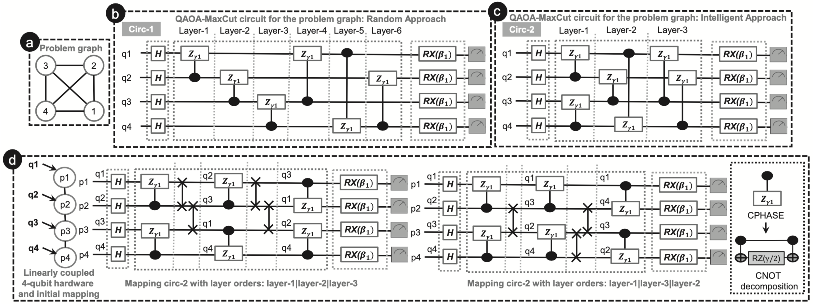
Extreme device/resource constraints:

1. Native gate set
2. Device topology
3. Hardware noise
4. Parallelism constraints

Generic strategies:

- Minimize two-qubit gates while respecting native gate set and topology
- Minimize quantum circuit depth while completing whole circuit
- Maximize parallelism while avoiding crosstalk noise

Parallelism constraints



a
Problem graph
b
QAOA-MaxCut circuit for the problem graph: Random Approach
Circ-1
Layer-1 Layer-2 Layer-3 Layer-4 Layer-5 Layer-6
c
QAOA-MaxCut circuit for the problem graph: Intelligent Approach
Circ-2
Layer-1 Layer-2 Layer-3
d
Linearly coupled 4-qubit hardware and initial mapping
Mapping circ-2 with layer orders: layer-1|layer-2|layer-3
Mapping circ-2 with layer orders: layer-1|layer-3|layer-2
CPHASE
CNOT decomposition
$RZ(\gamma/2)$

Fig. 1. (a) A 4-node 3-Regular graph, (b) a randomly constructed QAOA-MaxCut instance (circ-1) of the 4-node graph with p = 1, (c) an optimized circuit (circ-2) for the problem with reduced number of layers, (d) SWAP addition during circuit compilation for a target hardware with different layer orders.

Figure: Credit: [Alam et al., 2020]

Some types of gates commute, so we can move earlier or later.

Parallelism constraints

1. Amount of parallelism available in the instruction stream
2. Achievable parallelism in the control microarchitecture ("each student gets one coaxial input")
3. Safe parallelism despite crosstalk due to spatiotemporal and spectral overlap

Compiling from a high-level program to hardware

Goals:

1. Correctness: maximizing probability of success!
2. Ease of programming?
3. Compatibility between hardware implementations?

Extreme device/resource constraints:

1. Native gate set
2. Device topology
3. Hardware noise
4. Parallelism constraints

Primary sources

- [Ding and Chong, 2020, Chapters 4,6,7]
- [Córcoles et al., 2020, Section III.B]
- [National Academies of Sciences, Engineering, and Medicine, 2019, Chapter 6.5]
- [Martonosi and Roetteler, 2019, Chapter 6]

Alam, M., Ash-Saki, A., and Ghosh, S. (2020).
Circuit compilation methodologies for quantum approximate optimization algorithm.
2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO).

Alexeev, Y., Bacon, D., Brown, K. R., Calderbank, R., Carr, L. D., Chong, F. T., DeMarco, B., Englund, D., Farhi, E., Fefferman, B., Gorshkov, A. V., Houck, A., Kim, J., Kimmel, S., Lange, M., Lloyd, S., Lukin, M. D., Maslov, D., Maunz, P., Monroe, C., Preskill, J., Roetteler, M., Savage, M., and Thompson, J. (2020).
Quantum computer systems for scientific discovery.

Córcoles, A. D., Kandala, A., Javadi-Abhari, A., McClure, D. T., Cross, A. W., Temme, K., Nation, P. D., Steffen, M., and Gambetta, J. M. (2020).
Challenges and opportunities of near-term quantum computing systems.
Proceedings of the IEEE, 108(8):1338–1352.

Ding, Y. and Chong, F. T. (2020).
Quantum computer systems: Research for noisy intermediate-scale quantum computers.
Synthesis Lectures on Computer Architecture, 15(2):1–227.

Li, G., Ding, Y., and Xie, Y. (2019).
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.

Martonosi, M. and Roetteler, M. (2019).
Next steps in quantum computing: Computer science's role.

arXiv preprint arXiv:1903.10541.

Matsuura, A., Johri, S., and Hogaboam, J. (2019).
A systems perspective of quantum computing.
PhT, 72(3):40–46.

Murali, P., Linke, N. M., Martonosi, M., Abhari, A. J., Nguyen, N. H., and Alderete, C. H. (2019).
Full-stack, real-system quantum computer studies: Architectural comparisons and design insights.
In *Proceedings of the 46th International Symposium on Computer Architecture*, ISCA '19, page 527–540, New York, NY, USA. Association for Computing Machinery.

National Academies of Sciences, Engineering, and Medicine (2019).
Quantum Computing: Progress and Prospects.
The National Academies Press, Washington, DC.

Nielsen, M. A. and Chuang, I. L. (2011).
Quantum Computation and Quantum Information: 10th Anniversary Edition.
Cambridge University Press, USA, 10th edition.

Resch, S. and Karpuzcu, U. R. (2019).
Quantum computing: An overview across the system stack.

Tannu, S. S. and Qureshi, M. K. (2019).
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*, ASPLOS '19, page 987–999, New York, NY, USA. Association for Computing Machinery.