Time-Sliced Quantum Circuit Partitioning for Modular Architectures

Jonathan M. Baker
jmbaker@uchicago.edu
University of Chicago

Casey Duckering
cduck@uchicago.edu
University of Chicago

Alexander Hoover
alex8@uchicago.edu
University of Chicago

Frederic T. Chong
chong@cs.uchicago.edu
University of Chicago

ABSTRACT

Current quantum computer designs will not scale. To scale beyond small prototypes, quantum architectures will likely adopt a modular approach with clusters of tightly connected quantum bits and sparser connections between clusters. We exploit this clustering and the statically-known control flow of quantum programs to create tractable partitioning heuristics which map quantum circuits to modular physical machines one time slice at a time. Specifically, we create optimized mappings for each time slice, accounting for the cost to move data from the previous time slice and using a tunable lookahead scheme to reduce the cost to move to future time slices. We compare our approach to a traditional statically-mapped, owner-computes model. Our results show strict improvement over the static mapping baseline. We reduce the non-local communication overhead by 89.8% in the best case and by 60.9% on average. Our techniques, unlike many exact solver methods, are computationally tractable.

CCS CONCEPTS

• **Computer systems organization** → **Quantum computing**; • **Software and its engineering** → *Compilers.*

ACM Reference Format:
Jonathan M. Baker, Casey Duckering, Alexander Hoover, and Frederic T. Chong. 2020. Time-Sliced Quantum Circuit Partitioning for Modular Architectures. In *17th ACM International Conference on Computing Frontiers (CF '20), May 11–13, 2020, Catania, Italy.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3387902.3392617

1 INTRODUCTION

Quantum computing aims to provide significant speedup to many problems by taking advantage of quantum mechanical properties such as superposition and entanglement [6, 53, 59]. Important applications such as Shor's integer factoring algorithm [69] and Grover's unordered database search algorithm [26] provide potentially exponential and quadratic speedups, respectively.

Current quantum hardware of the NISQ era [61], which has on the order of tens to hundreds of physical qubits, is insufficient to run these important quantum algorithms. Scaling these devices even to a moderate sizes with low error rates has proven extremely challenging. Manufacturers of quantum hardware such as IBM and IonQ have had only limited success in extending the number of physical qubits present on a single contiguous piece of hardware. Issues on these devices such as crosstalk error scaling with the number of qubits or increased difficulty in control will limit the size this single-chip architecture can achieve [10, 11].

Due to these challenges, as well as developing technology for communicating between different quantum chips [7, 75], we expect quantum hardware to scale via a modular approach similar to how a classical computer can be scaled increasing the number of processors not just the size of the processors. Two of the leading quantum technologies, ion trap and superconducting physical qubits, are already beginning to explore this avenue and experimentalists project modularity will be the key to moving forward [3, 9, 18, 22, 32, 43, 44]. One such example for ion traps is shown in Figure 2 where many trapped ion devices are connected via a single central optical switch. Technology such as resonant busses in superconducting hardware or optical communication techniques in ion trap devices will enable a more distributed approach to quantum computing, having many smaller, well-connected devices with sparser and more expensive non-local connections between them. Optimistically, due to current technology in the near term, we expect these non-local communication operations to be somewhere between 5-100x higher latency than in-cluster communication.

With cluster-based approaches becoming more prominent, new compiler techniques for mapping and scheduling of quantum programs are needed. As the size of executable computations increase it becomes more and more critical to employ program mappings exhibiting both adaptivity of dynamic techniques and global optimization of static techniques. Key to realizing both advantages is to simplify the problem. Since non-local communication is dominant, we focus on only non-local costs. This simplification, along with static knowledge of all control flow, allows us to map a program in many timeslices with substantial lookahead for future program behavior. This approach would not be computationally tractable on a non-clustered machine.

For devices with many modular components mapping quantum programs translates readily to a graph partitioning problem with a goal of minimizing edge crossings between partitions. This approach is standard in many classical applications such as high

CF '20, May 11–13, 2020, Catania, Italy
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-7956-4/20/05…$15.00
https://doi.org/10.1145/3387902.3392617

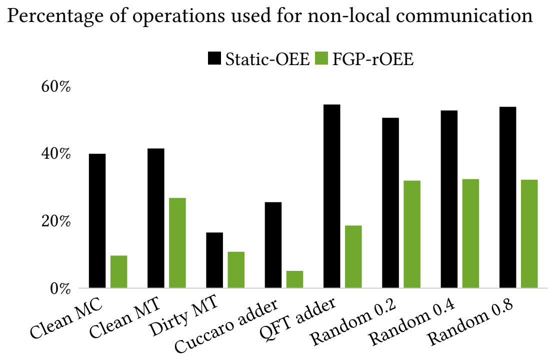

Figure 1: Non-local communication overhead in circuits mapped to cluster-based machines. Our new mapping scheme FPG-rOEE provides reduces the number of operations added for non-local communication on all benchmarks.

performance parallel computing, etc. [34, 66, 70] with the goal of minimizing total latency. Here latency is approximated by the total number of times qubits must be shuttled between different regions of the device. Graph partitioning is known to be hard and heuristics are the dominant approach [19, 23, 30, 37, 57].

While this problem is related to many problems in distributed or parallel computing, there are a few very important distinctions. In a typical quantum program, the control flow is statically known at compile time, meaning all interactions between qubits are known. Furthermore, the no-cloning theorem states we cannot make copies of our data, meaning non-local communication between clusters is *required* to interact data qubits. Finally, any additional non-local operations affect not only latency as they would classically but are directly related to the probability a program will succeed since operations in quantum computing are error prone and therefore reducing non-local communication is especially critical for successful quantum program execution.

Our primary contribution is the development of a complete system for mapping quantum programs to near-term cluster-based quantum architectures via graph partitioning techniques where qubit interaction in-cluster is relatively free compared to expensive out-of-cluster interaction. Our primary goal is to minimize the communication overhead by reducing the number of low-bandwidth, high-latency operations such as moving qubits which are required in order to execute a given quantum program. Rather than partitioning the circuit once to obtain a generally good global assignment of the qubits to clusters, we find a sequence of assignments, one for each time slice in the circuit. This fine-grained approach is much less studied, especially for this class of architectures. With our techniques, we reduce the total number of non-local communication operations by 89.8% in the best case and 60.9% in the average case; Figure 1 shows a few examples of circuits compiled statically versus with our methods.

The rest of the paper is organized as follows: In Section 2, we introduce the basics of quantum circuits and graph partitioning. In Section 3, we introduce our proposed methodology for mapping qubits to the clusters of these modular systems, specifically a method for *fine-grained partitioning*. In Section 4, we introduce a method for applying lookahead weights to tune what is considered *local* at each time slice and evaluate their effect on non-local communication. In Section 5, we introduce the benchmarks we test on and present our explicit toolflow for taking quantum programs to a sequence of mappings which guarantee interacting qubits are moved into the same partition before each time slice using non-local communication. In Section 6, we present our results and provide a brief discussion, and in Section 7, we present a summary of related work for hardware mapping. We conclude in Section 8.

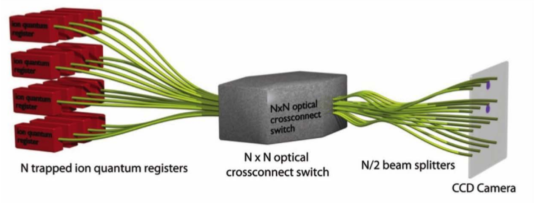

Figure 2: An example modular architecture of qubits in individual ion traps connected with optics proposed by Monroe et al [45]. Communication between traps is supported by photon-mediated entanglement. Similar communication for superconducting qubits [14] can facilitate modular architectures for that technology.

2 BACKGROUND

2.1 Quantum Programs and Architectures

The typical fundamental unit of quantum information is the qubit (quantum bit). Unlike classical bits which occupy either 1 or 0 at any given time, quantum bits may exist in a superposition of the two basis states $|0\rangle$ and $|1\rangle$. Qubits are manipulated via quantum gates, operations which are both reversible and preserve a valid probability distribution over the basis states. There is a single irreversible quantum operation called measurement, which transforms the qubit to either $|0\rangle$ or $|1\rangle$ probabilistically. Pairs of qubits are interacted via two-qubit gates, which are generally much more expensive in terms of error rates and latency.

There are a variety of competing styles of quantum systems each with a hardware topology specifying the relative location of the machine’s qubits. This topology indicates between which pairs of qubits two-qubit interactions may be performed.

Typical quantum hardware does not readily support long-range multi-qubit operations but does provide a mechanism for moving qubits, either by swapping qubits (in the case of nearest neighbor or 2D-grid devices), teleportation via photon mediated entanglement, physically moving qubits (as in ion-trap devices), or a resonant bus (as in superconducting devices). Interacting qubits which are distant generate additional latency which is undesirable for near-term qubits with limited coherence time (the expected lifetime of a qubit before an error). These machines have expected error rates on the order of 1 in every 100-1000 two-qubit gates [33, 78], and non-local communication has error on average 10-100x worse.

In this paper, we are motivated by a specific set of architectures or extensions to such architectures, as in [5, 39, 48, 73]. In these devices, qubits are arranged into several regions of high connectivity with

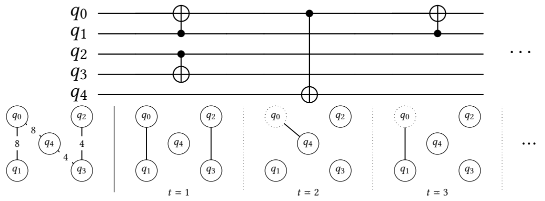

Figure 3: (Top) An example of a quantum program with single-qubit gates not shown. The inputs are on the left and time flows to the right toward the outputs. The two-qubit operations here are CNOT (controlled-NOT). (Bottom) The graph representations of the quantum circuit of the above circuit. On the far left is the total interaction graph where each edge is weighted by the total number of interactions for the whole circuit. To the right is the sequence of time slice graphs, where an edge is only present if the qubits interact in the time slice. The sum of all time slice graphs is the total interaction graph.

expensive communication between the clusters, referred to as non-local communication. These devices naturally lend themselves to mapping techniques which utilize partitioning algorithms.

Quantum programs are often represented as circuit diagrams, for example the one in Figure 3a. We define a *time slice* in a quantum program as a set of operations which are parallel in the circuit representation of the program. We express time slices as a function of both the circuit representation and limitations of the specific architecture. We also define a *time slice range* as a set of contiguous time slices; we also refer to them as *slices* and when no length is specified, it will be assumed to be of length 1.

For evaluation, we consider two primary metrics: the *width* and the *depth* of a circuit. The width is the total number of qubits used and the depth, or the run time, is the total number of time slices required to execute the program. Qubit movement operations which are inserted in order to move interacting qubits into the same partition contribute to the overall depth of the circuit.

We consider two abstract representations of quantum programs: the total interaction graph and a sequence of time slice interaction graphs, examples of which are found in Figure 3b. In both representations, each qubit is a vertex and edges between qubits indicate two-qubit operations acting on these qubits. In the total interaction graph, edges are weighted by the total number of interactions between pairs of qubits. In time slice graphs, an edge with weight 1 exists only if the pair of qubits interact at that time slice.

2.2 Graph Partitioning

Static Partitioning. Finding graph partitions is a well studied problem [23, 31, 37, 57] and is used frequently in classical architecture. In this paper, we consider a variant of the problem which fixes the total number of partitions and bounds the total number of elements in each partition. Specifically, given a fixed number of partitions k, a maximum partition size p, and an undirected weighted graph G with $|V(G)| \leq k \cdot p$ we want to find a k-way assignment of the vertices to partitions such that the weight of edges between vertices in different partitions is minimized. This can be rephrased in terms of statically mapping a quantum circuit to the aforementioned architectures. Let the total interaction graph be G and let k and p fixed by the topology of the architecture. Minimizing the edge weight between partitions corresponds to minimizing the total number of swaps which must be executed.

Solving for an optimal k-way partition is known to be hard [12], but there exist many algorithms which find approximate solutions [23, 37, 57]. There are several heuristic solvers such as in [35, 36] which can be used to find approximate k-way partition of a graph. However, they often cannot make guarantees about the size of the resulting partitions, preventing us from using them for the fixed size partitioning problem.

Partitioning Over Time. Rather than considering a single graph to be partitioned we instead consider the problem of generating a *sequence* of assignments of qubits to clusters, one for each moment of the circuit. We want to minimize the total number of differences between consecutive assignments, naturally corresponding to minimizing the total number of non-local communications between clusters. This problem is much less explored than the prior approach. Partitioning in this way guarantees interacting qubits will be placed in the same partition making the schedule for the input program immediate. In the case of a static partition, which gives only the initial mapping, a further step is needed to generate a schedule.

Optimal Compilation and Exact Solvers. It is too computationally expensive to find a true optimal solution for even reasonably sized input programs. Use of constraint-based solvers has been used recently to look for optimal and near-optimal solutions [47, 51, 52]. Unfortunately, these approaches will not scale in the near-term let alone to larger, error-corrected devices. We explored the use of these solvers but found them to be too slow. Finding a static mapping with SMT is impractical with more than 30 to 40 qubits, and SMT partitioning over time is impractical when number of qubits times the depth became more than 40.

3 MAPPING QUBITS TO CLUSTERS

We define an *assignment* as a set of partitions of the qubits, usually at a specific time slice. We present algorithms which take a quantum circuit and output a *path*, defined as a sequence of assignments of the qubits with the condition that every partitioning in the sequence is *valid.* An assignment is valid if each pair of interacting qubits in a time slice are located within the same partition. Finally, we define the *non-local communication* between consecutive assignments as the total number of operations which must be executed to transition the system from the first assignment to the second assignment. The total communication of a path is the sum over all communication along the path.

3.1 Computing Non-local Communication

To compute the non-local communication overhead between consecutive assignments of n qubits, we first construct a directed graph with multiple edges where the nodes in the graph are the partitions and the edges indicate a qubit moving from partition i to partition j. We extract all 2-cycles from this graph and remove those edges from the graph. We proceed extracting all 3-cycles, and so on and

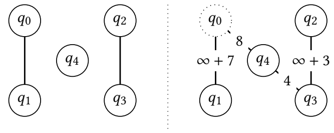

Figure 4: An example of a time slice graph with lookahead weights based on the circuit in Figure 3. We take the graph from the left and add weight to the edges of qubits that interact in the future. In this case, we take the weight equal to the number of times the qubits will interact in the future.

record the number of k-cycles extracted as c_k. When there are no cycles remaining, the total number of remaining edges is r, and the total communication overhead C is given by

$$C = r + \sum_{k=2}^{n} (k-1) \cdot c_k$$

The remaining edges indicate a qubit swapping with an unused qubit. We repeat this process for every pair of consecutive assignments in the path to compute the total non-local communication of the path. These cycles specify where qubits will be moved with non-local communication.

3.2 Baseline Non-local Communication

As a baseline we consider using a *Static Mapping* using an owner computes model, which takes into account the full set of qubit interactions for the circuit, providing a generally good assignment of the qubits for the entire duration of the program, called the static assignment. At each time step in the circuit, a good static assignment ensures, on average, qubits are not *too far* from other qubits they will interact with frequently. We find the assignment which requires the fewest number of swaps from the static assignment but has each pair of interacting qubits in a common partition. These assignments form a path for the computation. We refer to this method of path generation in conjunction with a partitioning algorithm, for example Static Mapping with OEE (Overall Extreme Exchange, discussed further later) is referred to as Static-OEE.

3.3 Fine Grained Partitioning

The primary approach we developed to dynamically map a circuit to hardware is *Fine Grained Partitioning* (FGP). In this algorithm, we find an assignment at every time slice using the time slice graphs. By default, these time slice graphs give only immediately local information about the circuit but have no knowledge about upcoming interactions. Alone, they only specify the constraints of which qubits interact in that time slice. The key advantage for this method is using *lookahead weights*. The main idea is to construct modified time slice graphs capturing more structure in the circuit than the default time slice graphs. We refer to these graphs as time slice graphs with lookahead weights, or *lookahead graphs*.

To construct the lookahead graph at time t, we begin with the original time slice graph and give the edges present infinite weight. For every pair of qubits we add the weight

$$w_t(q_i, q_j) = \sum_{t<m\leq T} I(m, q_i, q_j) \cdot D(m-t)$$

to their edge, where D is some monotonically decreasing, non-negative function, which we call the lookahead function, and $I(m, q_i, q_j)$ is an indicator that is 1 if q_i and q_j interact in time slice m and 0 otherwise, and T is the number of time slices in the circuit. The new time slice graphs consider the remainder of the circuit, more heavily weighting sooner interactions. The effectively infinite weight on edges between interacting qubits is present to guarantee any assignment will place interacting qubits into the same partition. An example is shown in Figure 4.

The final mapping of the qubits in our model is obtained by partitioning each of these time slices. Iteratively, we find the next assignment with a partitioning algorithm, seeded with the assignment obtained from the previous time slice. The first can choose a seed randomly or use the static assignment (presented in 3.2). The new weights in the time slice graphs will force any movement necessary in the partitioning algorithm. Together, these assignments give us a valid path for the circuit to be mapped into our hardware.

3.4 Choosing the Partitioning Algorithm

We assume full connectivity within clusters and the ability to move between clusters. These assumptions give us the liberty to tap into well studied partitioning algorithms. The foundation of many partitioning algorithms is largely considered to be the Kernighan-Lin heuristic for partitioning graphs with bounded partition sizes [23, 37, 57]. The KL heuristic selects pairs of vertices in a graph to exchange between partitions based on the weights between the vertices themselves and the total weight between the vertices and the partitions.

We consider a natural extension of the KL algorithm, Overall Extreme Exchange presented by Park and Lee [57]. The OEE algorithm finds a sequence of pairs of vertices to exchange and makes as many exchanges as give it an overall benefit. Using OEE, the Fine Grained Partitioning scheme often over corrects (see Figure 7). If a qubit needs to interact in another partition, then it can "drag along" a qubit it is about to interact with because OEE attempts to minimize weight between partitions regardless of its relation to the previous or next time slice graphs. Choosing an optimal partitioning algorithm would not give better solutions to our non-local communication based mapping problem. Instead, we consider a more relaxed version of a partitioning algorithm using the KL heuristic.

Relaxing the Partitioning Algorithm. We provide relaxed version of the algorithm better suited to generating a path over time, called relaxed-OEE (rOEE). We run OEE until the partition is valid for the time slice (all interacting qubits are in the same partition) and then make no more exchanges. This is similar in approach to finding the time slice partitions in our Static Mapping approaches. It is critically important we make our exchange choices using lookahead weights applied to the time slice graphs. Choosing without information about the upcoming circuit provides no insight into which qubits are beneficial to exchange. As a side benefit, making this change strictly speeds up OEE, an already fast heuristic algorithm. Although a strict asymptotic time bound for OEE is difficult to prove, rOEE never took more than a few seconds on any instance it was given.

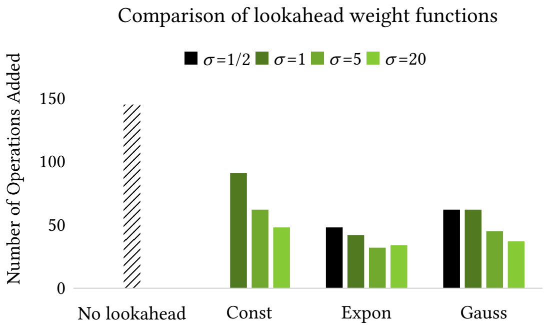

Figure 5: The effect of different lookahead functions with various σ on non-local communication in the Cuccaro adder, a very regular circuit, with 76 data and 24 ancilla qubits using FGP-rOEE. We see the exponential function outperforms the others for a circuit of highly regular structure.

With such a significant non-local communication overhead improvement (see Figure 7), this relaxed KL partitioning algorithm is much better suited for the problem at hand. It has the ability to take into account local structure in the circuit and avoid over correcting and swapping qubits unnecessarily.

4 LOOKAHEAD WEIGHTS

Finding a suitable lookahead weight function to use in Fine Grained Partitioning is necessary to maximize the benefit gained from choosing our swaps appropriately between time slices. We only require the lookahead function to be monotonically decreasing and non-negative. Throughout this section, we denote our lookahead weight function as D.

4.1 Natural Candidates

We explore a few natural candidate weighting functions from the huge space of possible functions. In each of the functions we explore below, we vary a stretching factor or scale σ which can be tuned for the given circuit, providing a trade-off between local and global information.

Constant Function.

$$D(n) = \begin{cases} 1 & n \leq \sigma \\ 0 & n > \sigma \end{cases}$$

A constant function captures a fixed amount of local information in the circuit. This is just the number of times the pair of qubits interact in the next σ time slices. For $\sigma = 0$, this function corresponds to no lookahead applied.

Exponential Decay.

$$D(n) = 2^{-n/\sigma}$$

An exponential is a natural way to model a decaying precedence. When $\sigma \leq 1$, any interaction will always have a weight at least as high as the sum of interactions after it.

Gaussian Decay.

$$D(n) = e^{-n^2/\sigma^2}$$

Similar to an exponential, a Gaussian is natural to model decaying precedence with more weight given to local interactions.

4.2 Evaluating Lookahead Functions

To evaluate the choice of lookahead function as well as choice of σ, we study Fine Grained Partitioning using rOEE with all of the above candidate functions with varying σ on benchmarks of various types: those with lots of local structure (a quantum ripple carry adder), those with very little structure (a random circuit), and those which lie somewhere in between (a Generalized Toffoli decomposition).

In Figure 5, we show an example of a circuit which benefits from having a large scale σ, the Cuccaro Adder [15]. In contrast, all of the random benchmarks benefit from having small σ values, functions which decay quickly even for small n.

We also compare the different natural lookahead functions we described in the previous section on some representative benchmarks in Figure 6. In these figures, we see the exponential decay has a clear benefit over the rest in the structured circuits of the Multi-Control gate and the Cuccaro Adder. In random circuits, there seems to be no clear benefit to any of the lookahead functions, so long as they have some small lookahead scaling factor. So, we use exponential decay with $\sigma = 1$ for our primary benchmarks in Section 5.

5 EXPERIMENTAL SETUP

All experiments were run on an Intel(R) Xeon(R) Silver 4100 CPU at 2.10 GHz with 128 GB of RAM with 32 cores running Ubuntu 16.04.5. Each test was run on a single core. Our framework runs on Python 3.6.5 using Google's Cirq framework for circuit processing and for implementing our benchmarks [1]. For testing exact solvers, we used the Z3 SMT solver [16], though results could not be obtained for the size of benchmarks tested because Z3 never completes on problems this size.

5.1 Benchmarks

We benchmark the performance of our circuit mapping algorithms on some common sub-circuits used in many algorithms (for example Shor's and Grovers) and, for comparison, on random circuits. Our selection of benchmarks covers a wide variety of internal structure. For every benchmark, we use a representative cluster-based architecture with 100 qubits with 10 clusters each containing 10 qubits but our methods are not limited to any size. We sweep over the number of qubits used from 50 to 100, when in the cases of a few benchmarks the remaining qubits are available for use as either clean or dirty ancilla[1] A selected cross section of our benchmarks is shown in Table 1.

Generalized Toffoli Gate. The Generalized Toffoli gate (C^nU) is an n-controlled U gate for any single qubit unitary U and is well studied [8, 21, 25, 28, 50, 72]. A C^nX gate works by performing an X gate on the target conditioned on all control qubits being in the $|1\rangle$ state. There are many known decompositions [4, 24, 29] both with and without the use of ancilla. A complete description of generating these circuits is given by [2], which provides a method for using clean ancilla.

[1] An ancilla is a temporary quantum bit used often to reduce the depth or gate count of a circuit. "Clean" indicates the initial state of the ancilla is known while "dirty" means the state is unknown.

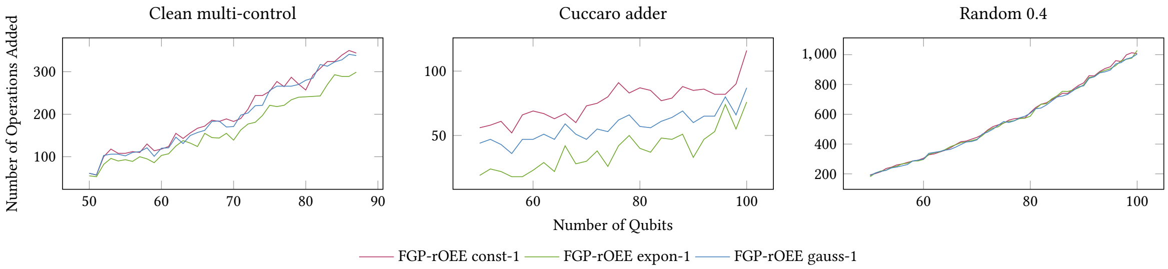

Figure 6: The non-local communication, measured in number of operations between clusters added, for our representative benchmark circuits mapped by each FGP-rOEE using different lookahead functions, each with $\sigma = 1$. The x-axis is the number of input/output qubits. The remainder are used as ancilla for clean multi-control. The exponential function is better on all instances of Clean multi-control and Cuccaro adder, and there is no substantial advantage of one function over the others in the random circuit.

Table 1: Depth and operation counts for a subset of our benchmarks

	Clean multi-control			**Clean multi-target**			**Dirty multi-target**			**Cuccaro adder**		
Data Qubits	50	76	87	50	76	100	50	76	100	50	76	100
Depth	82	265	846	17	22	99	26	34	99	435	669	885
Two Qubit Op Count (Unmapped)	760	2040	2488	57	85	99	103	157	99	505	778	1030
Non-local Comm. Ops (Static-OEE)	288	1297	1928	35	60	169	34	31	169	159	243	365
Non-local Comm. Ops (FGP-rOEE expon-1)	55	218	299	21	31	72	17	19	72	19	42	76

	QFT adder			**Random 0.2**			**Random 0.4**			**Random 0.8**		
Data Qubits	50	76	100	50	76	100	50	76	100	50	76	100
Depth	72	111	147	15	23	30	28	41	54	46	67	86
Two Qubit Op Count (Unmapped)	625	1444	2500	246	588	995	477	1156	1997	965	2260	3944
Non-local Comm. Ops (Static-OEE)	512	1144	2542	180	486	863	344	993	1795	682	1944	3462
Non-local Comm. Ops (FGP-rOEE expon-1)	131	329	541	96	275	498	181	552	1028	386	1070	1964

Multi-Target Gate. The multi-target gate performs a single-qubit gate on many targets conditioned on a single control qubit being in the $|1\rangle$ state. This is useful in several applications such as one quantum adder design [25] and can also be used in the implementation of error correcting codes [17]. These circuits can be generated with different numbers of ancilla (both clean and dirty), as given by [2].

Arithmetic Circuits. Arithmetic circuits in quantum computing are typically used as subcircuits of much larger algorithms like Shor's factoring algorithm and are well studied [21, 25, 41]. Many arithmetic circuits, such as modular exponentiation, lie either at the border or beyond the range of NISQ era devices, typically requiring either error correction or large numbers of data ancilla to execute. We examine two types of quantum adders - the Cuccaro Adder and the QFT Adder - as representatives of a class of highly structured and highly regular arithmetic circuits [15, 63].

Random Circuit. The gates presented above have a lot of regular structure when decomposed into circuits. We want to contrast this with circuits with less structure. We create these random circuits by picking some probability p and some number of samples and generate an interaction between two qubits with probability p for each sample. These circuits have the same structure as QAOA solving a min-cut problem on a random graph with edge probability p, so these circuits are a realistic benchmark.

5.2 Circuit to Hardware

We begin with a quantum program which is specified at the gate level, consisting of one and two qubit gates. We then generate the total interaction and time slice graphs, where we assume gates are inserted at the earliest possible time. Any further optimization, such as via commutivity or template matching, should be done prior to mapping the program to hardware. We also take the specifications of the hardware, such as number of clusters and the maximum size of the clusters, which constrain possible mappings.

We use our rOEE as our algorithm for Fine Grained Partitioning. Therefore, we pass the total interaction graph to a static partitioning algorithm to obtain a good starting assignment. This serves as a seed to rOEE rather than starting with a random assignment which may introduce unnecessary starting communication. To the time slice graphs, we apply the lookahead function to obtain the lookahead graphs. We run rOEE on this set of graphs to obtain an assignment sequence such that at every time slice qubits which interact appear in the same bucket. This assignment describes what non-local communication is added before each slice. Finally, we compute the cost and insert the necessary movement operations into the circuit to move interacting qubits into the same partition, this is a path. As a byproduct, by generating a partitioning over time, we obtain a schedule of operations to be performed.

6 RESULTS AND DISCUSSION

We run our mapping algorithms on each of our benchmark circuits. The results are shown in Figure 7.

Baseline mapping and the original version of OEE perform worse than our best scheme on any benchmark tested. Baseline mapping uses global structure of the graph, but often maintains this structure too much throughout the execution of the circuit. This lack of local awareness and rigid nature of the Static Mapping limits its usefulness. Most out of the box graph partitioning algorithms are designed to only minimize the edge weight between partitions; this will tend to over correct for local structure in the circuit. FGP can overcome this limitation with its choice of partitioning algorithm. By relaxing the partitioning algorithm and not requiring local optimality, we only move qubits until all interacting pairs are together, we require far fewer non-local operations.

The most noticeable changes between FGP-OEE and FGP-rOEE are on the clean multi-control gate with many controls and on the Cuccaro adder. Here, there are often consecutive, overlapping operations with little parallelism. With this structure, after the first operation is performed, the original OEE algorithm will exchange qubits to comply with the next time slice for the next operation. OEE is required to separate qubits which will later interact. To minimize the total crossing weight between partitions, more qubits are shuffled around, usually towards this displaced qubit. In rOEE, this reshuffle optimization never takes place because we terminate once each pair of interacting qubits in a time slice is placed in a common partition. The reshuffling detriments the overall non-local communication when running the circuit because of how often qubits will be displaced from their common interaction partners. In rOEE, not reshuffling keeps the majority of the qubits in sufficiently good spots and the displaced qubit has the opportunity to immediately move back with its interaction partners later.

We include the algorithm Fixed Length Slicing as an alternative not presented in this paper. It is a method with slower computation which explores grouping time slices at fixed intervals. Fixed Length Slicing was consistently the best performing time slice range based mapping algorithm, so we present it in our results. FLS-OEE only beats FGP-rOEE on some instances of the multi-target benchmarks and consistently performs worse on all other benchmarks.

In Figure 1, we show the percentage of operations used for non-local communication for each of the benchmark circuits, and in Table 2 we show the percent improvement of our algorithm over the baseline. On average, we save over 60% of the non-local communication operations added. When each non-local communication operation is implemented in hardware, the amount of time each takes is significantly longer than the operations between the qubits in the clusters [46]. Based on current communication technology, we expect these non-local communication operations to take anywhere from 5x to 100x longer than local in-cluster operations. Furthermore, the choice in technology limits how many of these expensive operations can be performed in parallel.

Table 2: Comparison of Static-OEE against FGP-rOEE

% Reduction	min	max	gmean
Clean multi-control	78.1	84.9	81.9
Clean multi-target	30.8	59.6	44.7
Dirty multi-target	22.6	65.1	39.9
Cuccaro adder	79.1	89.8	85.0
QFT adder	76.6	84.5	81.5
Random 0.2	52.4	57.8	55.3
Random 0.4	53.6	59.0	57.0
Random 0.8	57.0	60.4	59.1
Aggregate	**22.6**	**89.8**	**60.9**

Table 3: Estimated execution times of the clean multi-control benchmark with 76 data qubits and 24 ancilla

	Sequential Comm.		**Parallel Comm.**	
Multiplier	Static-OEE	FGP-rOEE	Static-OEE	FGP-rOEE
$5x$	2.0 ms	0.41 ms	0.67 ms	0.26 ms
$10x$	4.0 ms	0.73 ms	1.3 ms	0.43 ms
$100x$	39 ms	6.6 ms	12 ms	3.6 ms

In Table 3 we compute the estimated running time (two-qubit gates take 300ns [33] and the multiplier indicates how many times longer non-local communication operations take) based on this ratio of costs and show that by substantially reducing the non-local communication via FGP-rOEE, we can drastically reduce the run time. We compare our algorithm to the baseline when non-local communication can be performed in parallel (such as in optically connected ion trap devices) and when it is forced to occur sequentially (as when using a resonant bus in superconducting devices). Based on current technology, a 5-10x multiplier is optimistic while 100x is realistic in the near term.

7 RELATED WORK

Current quantum hardware is extremely restricted and has prompted research aimed at making the most of current hardware conditions. This usually amounts to a few main categories of optimization. Circuit optimization at a high level to reduce the number of gates or depth via template matching as in [42, 65] or via other optimization techniques as in [49, 79]. Other work focuses on optimization at the device level, such as by breaking the circuit model altogether as in [68] or by simply improving pulses via Quantum Optimal Control [76].

At an architectural level, optimization has been studied for many different types hardware with various topologies. The general strategy in most of these works is to reduce SWAP counts with the same motivation as this work, as in [27, 56, 71, 74, 77, 79, 80]. Much of this work focuses primarily on linear nearest neighbor (LNN) architectures or 2D lattice architectures as in [58, 60, 62, 64, 67]. Some work has focused on ion trap mappings as in [20] though the architecture of this style of device closely resembles a 2D architecture. Some work has recently focused on optimization around specific error rates in near term machines as in [40, 47]. Many of these techniques promise an extension to arbitrary topologies but are not specifically designed to accommodate cluster-based architectures. Work by [13] has explored using graph partitioning to reduce swap counts in near term machines, but their focus is on LNN architectures exclusively. Other work focuses on architectures of the more distant future with error correction [38, 54, 55].

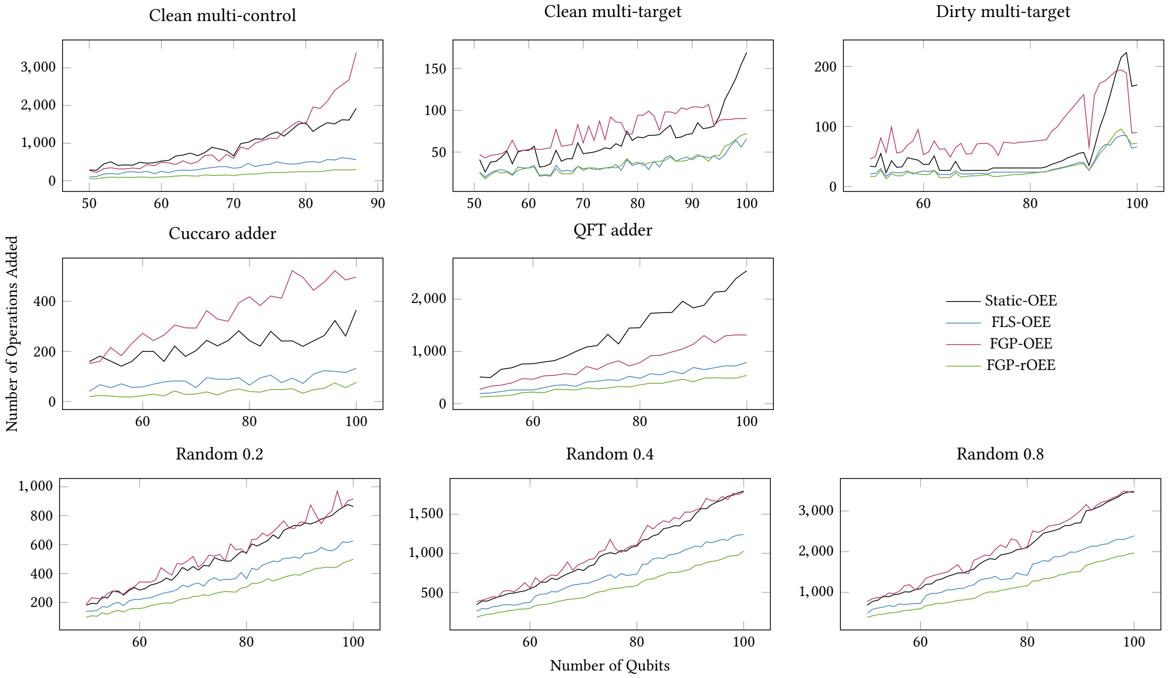

Figure 7: The non-local communication overhead for our benchmark circuits mapped by each mapping algorithm. The x-axis is the number of qubits that are used in the circuit. The y-axis is the number of non-local communication operations inserted to make the circuit executable in our hardware model. In Clean multi-control, Clean multi-target, and Dirty multi-target, the remainder of the 100 qubits are used as ancilla (clean or dirty determined by the circuit name). FGP-rOEE outperforms all other mapping algorithms on all but the multi-target circuits, and shows substantial improvement over the static baseline. As the size of the circuit increases, rOEE tends to outperform by a greater margin, indicating scales better into the future.

8 CONCLUSION

Alternative to using near-optimal graph partitioning algorithms to find a single static assignment for an entire circuit, we show considering the locality in a circuit during a mapping gives a reduction in the total non-local communication required when running a quantum circuit. There is a natural restriction in using static mappings suggesting the problem of mapping qubits to cluster-based architectures has a different structure than partitioning a single graph for minimum weight between the partitions. Our modification to OEE no longer attempts to optimize the weights at every time slice. It is much more effective in practice to guide the partitioning based on heuristics and not to find the optimal value for every time slice. Optimality at every time slice does not correspond to a global reduction in non-local communication overhead.

We propose to use similar schemes for other cluster-based quantum hardware, especially those based on internally connected clusters. In our model, the different clusters of the architecture are also very well connected, but is not limited to only this specific instance of a clustered architecture. Our proposed algorithm produces partitions based on a simplifying assumption about the connectivity of the clusters because the cost of non-local communication is substantially more expensive than any in-cluster operations. Our method can be adapted to other cluster-based architectures by first applying our partitioning algorithm to obtain good clusters of operations and then adding a device-specific scheduling algorithm for scheduling much cheaper in-cluster operations.

A relaxed version with well chosen lookahead functions of a heuristic outperforms a well selected initial static mapping. Using lookahead weights has been explored previously, as in [80], and more can be done to better choose the lookahead function, for example based on a metric of circuit regularity. Techniques for mapping which attempt to solve for near optimal mappings will not scale and instead heuristics will be the dominant approach. Our approach is computationally tractable and adaptable to changes in machine architecture, such as additional or varied size clusters.

Non-local communication overhead in quantum programs makes up a large portion of all operations performed, therefore, minimizing communication is critical. In recent hardware [46], the cost of moving between clusters makes non-trivial computation impossible with current standards for mapping qubits to hardware. Reducing this bottleneck or finding algorithms to reduce the non-local communication are critical for quantum computation. We reduce this cost substantially in cluster-based architectures (see Table 3).

ACKNOWLEDGMENTS

This work is funded in part by EPiQC, an NSF Expedition in Computing, under grants CCF-1730449/1832377; in part by STAQ under grant NSF Phy-1818914; and in part by DOE grants DE-SC0020289 and DE-SC0020331.

REFERENCES

[1] 2018. Cirq: A python framework for creating, editing, and invoking Noisy Intermediate Scale Quantum (NISQ) circuits. https://github.com/quantumlib/cirq.

[2] Jonathan M. Baker, Casey Duckering, Alexander Hoover, and Frederic T. Chong. 2019. Decomposing Quantum Generalized Toffoli with an Arbitrary Number of Ancilla. arXiv:arXiv:1904.01671

[3] Aniruddha Bapat, Zachary Eldredge, James R Garrison, Abhinav Deshpande, Frederic T Chong, and Alexey V Gorshkov. 2018. Unitary entanglement construction in hierarchical networks. *Physical Review A* 98, 6 (2018), 062328.

[4] Adriano Barenco, Charles H. Bennett, Richard Cleve, David P. DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A. Smolin, and Harald Weinfurter. 1995. Elementary gates for quantum computation. *Phys. Rev. A* 52 (Nov 1995), 3457–3467. Issue 5. https://doi.org/10.1103/PhysRevA.52.3457

[5] A. Bermudez, X. Xu, R. Nigmatullin, J. O'Gorman, V. Negnevitsky, P. Schindler, T. Monz, U. G. Poschinger, C. Hempel, J. Home, F. Schmidt-Kaler, M. Biercuk, R. Blatt, S. Benjamin, and M. Müller. 2017. Assessing the progress of trapped-ion processors towards fault-tolerant quantum computation. Phys. Rev. X 7, 041061 (2017). (2017). https://doi.org/10.1103/PhysRevX.7.041061 arXiv:arXiv:1705.02771

[6] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. 2016. Quantum Machine Learning. Nature 549, 195-202 (2017). *Nature* 549 (13 Sep 2016), 195 EP –. https://doi.org/10.1038/nature23474 arXiv:arXiv:1611.09347

[7] Brad R Blakestad, Aaron Vandevender, Christian Ospelkaus, Jason Amini, Joseph W Britton, Dietrich G Leibfried, and David J Wineland. 2009. High Fidelity Transport of Trapped-Ion Qubits through an X-Junction Trap Array| NIST. *Nature Physics* 102, Nature Physics (2009).

[8] Alex Bocharov, Martin Roetteler, and Krysta M. Svore. 2017. Factoring with qutrits: Shor's algorithm on ternary and metaplectic quantum architectures. *Phys. Rev. A* 96 (Jul 2017), 012306. Issue 1. https://doi.org/10.1103/PhysRevA.96.012306

[9] Teresa Brecht, Wolfgang Pfaff, Chen Wang, Yiwen Chu, Luigi Frunzio, Michel H Devoret, and Robert J Schoelkopf. 2016. Multilayer microwave integrated quantum circuits for scalable quantum computing. *npj Quantum Information* 2 (2016), 16002.

[10] Kenneth R Brown, Jungsang Kim, and Christopher Monroe. 2016. Co-designing a scalable quantum computer with trapped atomic ions. *npj Quantum Information* 2 (2016), 16034.

[11] Colin D Bruzewicz, John Chiaverini, Robert McConnell, and Jeremy M Sage. 2019. Trapped-ion quantum computing: Progress and challenges. *Applied Physics Reviews* 6, 2 (2019), 021314.

[12] Thang Nguyen Bui and Curt Jones. [n.d.]. Finding good approximate vertex and edge partitions is NP-hard. 42, 3 ([n. d.]), 153 – 159. https://doi.org/10.1016/0020-0190(92)90140-Q

[13] Amlan Chakrabarti, Susmita Sur-Kolay, and Ayan Chaudhury. 2011. Linear Nearest Neighbor Synthesis of Reversible Circuits by Graph Partitioning. arXiv:arXiv:1112.0564

[14] Kevin S. Chou, Jacob Z. Blumoff, Christopher S. Wang, Philip C. Reinhold, Christopher J. Axline, Yvonne Y. Gao, L. Frunzio, M. H. Devoret, Liang Jiang, and R. J. Schoelkopf. 2018. Deterministic teleportation of a quantum gate between two logical qubits. *Nature* 561, 7723 (2018), 368–373. https://doi.org/10.1038/s41586-018-0470-y

[15] Steven A. Cuccaro, Thomas G. Draper, Samuel A. Kutin, and David Petrie Moulton. 2004. A new quantum ripple-carry addition circuit. *arXiv e-prints*, Article quant-ph/0410184 (Oct 2004), quant-ph/0410184 pages. arXiv:quant-ph/quant-ph/0410184

[16] Leonardo de Moura and Nikolaj Bjørner. 2008. Z3: An Efficient SMT Solver. In *Tools and Algorithms for the Construction and Analysis of Systems*, C. R. Ramakrishnan and Jakob Rehof (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 337–340.

[17] Simon J. Devitt, Kae Nemoto, and William J. Munro. 2009. Quantum Error Correction for Beginners. Rep. Prog. Phys. 76 (2013) 076001. (2009). https://doi.org/10.1088/0034-4885/76/7/076001 arXiv:arXiv:0905.2794

[18] Michel H Devoret and Robert J Schoelkopf. 2013. Superconducting circuits for quantum information: an outlook. *Science* 339, 6124 (2013), 1169–1174.

[19] C. H. Q. Ding and H. D. Simon. 2001. A min-max cut algorithm for graph partitioning and data clustering. In *Proceedings 2001 IEEE International Conference on Data Mining*. 107–114. https://doi.org/10.1109/ICDM.2001.989507

[20] Mohammad Javad Dousti and Massoud Pedram. 2012. Minimizing the Latency of Quantum Circuits During Mapping to the Ion-trap Circuit Fabric. In *Proceedings of the Conference on Design, Automation and Test in Europe* (Dresden, Germany) *(DATE '12)*. EDA Consortium, San Jose, CA, USA, 840–843. http://dl.acm.org/citation.cfm?id=2492708.2492917

[21] Thomas G. Draper. 2000. Addition on a Quantum Computer. arXiv:arXiv:quant-ph/0008033

[22] L-M Duan and Christopher Monroe. 2010. Colloquium: Quantum networks with trapped ions. *Reviews of Modern Physics* 82, 2 (2010), 1209.

[23] Charles M Fiduccia and Robert M Mattheyses. 1982. A linear-time heuristic for improving network partitions. In *19th Design Automation Conference*. IEEE, 175–181.

[24] Craig Gidney. 2015. Constructing Large Controlled Nots. http://algassert.com/circuits/2015/06/05/Constructing-Large-Controlled-Nots.html

[25] Craig Gidney. 2017. Factoring with n+2 clean qubits and n-1 dirty qubits. arXiv:arXiv:1706.07884

[26] Lov K. Grover. 1996. A Fast Quantum Mechanical Algorithm for Database Search. In *ANNUAL ACM SYMPOSIUM ON THEORY OF COMPUTING*. ACM, 212–219.

[27] Gian Giacomo Guerreschi and Jongsoo Park. 2017. Two-step approach to scheduling quantum circuits. arXiv:arXiv:1708.00023

[28] Thomas Häner, Martin Roetteler, and Krysta M. Svore. 2017. Factoring Using 2N + 2 Qubits with Toffoli Based Modular Multiplication. *Quantum Info. Comput.* 17, 7-8 (June 2017), 673–684. http://dl.acm.org/citation.cfm?id=3179553.3179560

[29] Y. He, M.-X. Luo, E. Zhang, H.-K. Wang, and X.-F. Wang. 2017. Decompositions of n-qubit Toffoli Gates with Linear Circuit Complexity. *International Journal of Theoretical Physics* 56 (July 2017), 2350–2361. https://doi.org/10.1103/PhysRevA.75.022313

[30] Bruce Hendrickson and Robert Leland. 1995. An improved spectral graph partitioning algorithm for mapping parallel computations. *SIAM Journal on Scientific Computing* 16, 2 (1995), 452–469.

[31] Bruce Hendrickson and Robert Leland. 1995. A multi-level algorithm for partitioning graphs. (1995).

[32] David Hucul, Justin E Christensen, Eric R Hudson, and Wesley C Campbell. 2017. Spectroscopy of a synthetic trapped ion qubit. *Physical review letters* 119, 10 (2017), 100501.

[33] ibm0 [n.d.]. IBM Quantum Devices. https://quantumexperience.ng.bluemix.net/qx/devices. Accessed: 2019-03-16.

[34] F. M. Johannes. 1996. Partitioning of VLSI circuits and systems. In *33rd Design Automation Conference Proceedings, 1996*. 83–87. https://doi.org/10.1109/DAC.1996.545551

[35] George Karypis and Vipin Kumar. 1998. Multilevel k-way Partitioning Scheme for Irregular Graphs. *J. Parallel and Distrib. Comput.* 48, 1 (1998), 96 – 129. https://doi.org/10.1006/jpdc.1997.1404

[36] George Karypis and Vipin Kumar. 2009. MeTis: Unstructured Graph Partitioning and Sparse Matrix Ordering System, Version 4.0. http://www.cs.umn.edu/~metis.

[37] Brian W Kernighan and Shen Lin. 1970. An efficient heuristic procedure for partitioning graphs. *Bell system technical journal* 49, 2 (1970), 291–307.

[38] L. Lao, B. van Wee, I. Ashraf, J. van Someren, N. Khammassi, K. Bertels, and C. G. Almudever. 2018. Mapping of Lattice Surgery-based Quantum Circuits on Surface Code Architectures.

[39] B. Lekitsch, S. Weidt, A. G. Fowler, K. Mølmer, S. J. Devitt, C. Wunderlich, and W. K. Hensinger. 2015. Blueprint for a microwave trapped-ion quantum computer. Science Advances Vol. 3, no. 2 (2017). (2015). https://doi.org/10.1126/sciadv.1601540 arXiv:arXiv:1508.00420

[40] Gushu Li, Yufei Ding, and Yuan Xie. 2018. Tackling the Qubit Mapping Problem for NISQ-Era Quantum Devices. arXiv:arXiv:1809.02573

[41] Igor L. Markov and Mehdi Saeedi. 2012. Constant-Optimized Quantum Circuits for Modular Multiplication and Exponentiation. Quantum Information and Computation, Vol. 12, No. 5&6, pp. 0361-0394, 2012. (2012). arXiv:arXiv:1202.6614

[42] D. Maslov, G. W. Dueck, D. M. Miller, and C. Negrevergne. 2008. Quantum Circuit Simplification and Level Compaction. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 27, 3 (March 2008), 436–444. https://doi.org/10.1109/TCAD.2007.911334

[43] Dmitri Maslov, Yunseong Nam, and Jungsang Kim. 2018. An outlook for quantum computing [point of view]. *Proc. IEEE* 107, 1 (2018), 5–10.

[44] Christopher Monroe and Jungsang Kim. 2013. Scaling the ion trap quantum processor. *Science* 339, 6124 (2013), 1164–1169.

[45] C. Monroe, R. Raussendorf, A. Ruthven, K. R. Brown, P. Maunz, L.-M. Duan, and J. Kim. 2014. Large-scale modular quantum-computer architecture with atomic memory and photonic interconnects. *Phys. Rev. A* 89 (Feb 2014), 022317. Issue 2. https://doi.org/10.1103/PhysRevA.89.022317

[46] Emily Mount, Daniel Gaultney, Geert Vrijsen, Michael Adams, So-Young Baek, Kai Hudek, Louis Isabella, Stephen Crain, Andre van Rynbach, Peter Maunz, et al. 2016. Scalable digital hardware for a trapped ion quantum computer. *Quantum Information Processing* 15, 12 (2016), 5281–5298.

[47] Prakash Murali, Jonathan M. Baker, Ali Javadi Abhari, Frederic T. Chong, and Margaret Martonosi. 2019. Noise-Adaptive Compiler Mappings for Noisy Intermediate-Scale Quantum Computers. arXiv:arXiv:1901.11054

[48] R. K. Naik, N. Leung, S. Chakram, Peter Groszkowski, Y. Lu, N. Earnest, D. C. McKay, Jens Koch, and D. I. Schuster. [n.d.]. Random access quantum information processors using multimode circuit quantum electrodynamics. 8, 1 ([n. d.]), 1904. https://doi.org/10.1038/s41467-017-02046-6

[49] Yunseong Nam, Neil J. Ross, Yuan Su, Andrew M. Childs, and Dmitri Maslov. [n.d.]. Automated optimization of large quantum circuits with continuous parameters. 4, 1 ([n. d.]), 23. https://doi.org/10.1038/s41534-018-0072-4

[50] Michael A. Nielsen and Isaac L. Chuang. 2011. *Quantum Computation and Quantum Information: 10th Anniversary Edition* (10th ed.). Cambridge University Press, New York, NY, USA.

[51] Tony Nowatzki, Newsha Ardalani, Karthikeyan Sankaralingam, and Jian Weng. 2018. Hybrid Optimization/Heuristic Instruction Scheduling for Programmable

Accelerator Codesign. In *Proceedings of the 27th International Conference on Parallel Architectures and Compilation Techniques* (Limassol, Cyprus) *(PACT '18)*. ACM, New York, NY, USA, Article 36, 15 pages. https://doi.org/10.1145/3243176.3243212

[52] Tony Nowatzki, Michael Sartin-Tarm, Lorenzo De Carli, Karthikeyan Sankaralingam, Cristian Estan, and Behnam Robatmili. 2014. A Scheduling Framework for Spatial Architectures Across Multiple Constraint-Solving Theories. *ACM Trans. Program. Lang. Syst.* 37, 1, Article 2 (Nov. 2014), 30 pages. https://doi.org/10.1145/2658993

[53] P. J. J. O'Malley, R. Babbush, I. D. Kivlichan, J. Romero, J. R. McClean, R. Barends, J. Kelly, P. Roushan, A. Tranter, N. Ding, B. Campbell, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, A. G. Fowler, E. Jeffrey, E. Lucero, A. Megrant, J. Y. Mutus, M. Neeley, C. Neill, C. Quintana, D. Sank, A. Vainsencher, J. Wenner, T. C. White, P. V. Coveney, P. J. Love, H. Neven, A. Aspuru-Guzik, and J. M. Martinis. 2016. Scalable Quantum Simulation of Molecular Energies. *Phys. Rev. X* 6 (Jul 2016), 031007. Issue 3. https://doi.org/10.1103/PhysRevX.6.031007

[54] Alexandru Paler, Simon J. Devitt, Kae Nemoto, and Ilia Polian. 2014. Mapping of Topological Quantum Circuits to Physical Hardware. *Scientific Reports* 4 (11 Apr 2014), 4657 EP –. http://dx.doi.org/10.1038/srep04657 Article.

[55] Alexandru Paler, Ilia Polian, Kae Nemoto, and Simon J. Devitt. 2015. Fault-Tolerant High Level Quantum Circuits: Form, Compilation and Description. Quantum Science and Technology, 2, 025003 (2017). (2015). https://doi.org/10.1088/2058-9565/aa66eb arXiv:arXiv:1509.02004

[56] Alexandru Paler, Alwin Zulehner, and Robert Wille. 2018. NISQ circuit compilers: search space structure and heuristics. arXiv:arXiv:1806.07241

[57] Taehoon Park and Chae Y Lee. 1995. Algorithms for partitioning a graph. *Computers & Industrial Engineering* 28, 4 (1995), 899–909.

[58] M. Pedram and A. Shafaei. 2016. Layout Optimization for Quantum Circuits with Linear Nearest Neighbor Architectures. *IEEE Circuits and Systems Magazine* 16, 2 (Secondquarter 2016), 62–74. https://doi.org/10.1109/MCAS.2016.2549950

[59] Alejandro Perdomo-Ortiz, Alexander Feldman, Asier Ozaeta, Sergei V. Isakov, Zheng Zhu, Bryan O'Gorman, Helmut G. Katzgraber, Alexander Diedrich, Hartmut Neven, Johan de Kleer, Brad Lackey, and Rupak Biswas. 2017. On the readiness of quantum optimization machines for industrial applications. arXiv:arXiv:1708.09780

[60] Paul Pham and Krysta M. Svore. 2013. A 2D Nearest-neighbor Quantum Architecture for Factoring in Polylogarithmic Depth. *Quantum Info. Comput.* 13, 11-12 (Nov. 2013), 937–962. http://dl.acm.org/citation.cfm?id=2535639.2535642

[61] John Preskill. 2018. Quantum Computing in the NISQ era and beyond. *Quantum* 2 (Aug. 2018), 79. https://doi.org/10.22331/q-2018-08-06-79

[62] D. Ruffinelli and B. Baran. 2016. A multiobjective approach to linear nearest neighbor optimization for 2D quantum circuits. In *2016 XLII Latin American Computing Conference (CLEI)*. 1–8. https://doi.org/10.1109/CLEI.2016.7833378

[63] Lidia Ruiz-Perez and Juan Carlos Garcia-Escartin. 2014. Quantum arithmetic with the Quantum Fourier Transform. Quantum Inf Process (2017) 16: 152. (2014). https://doi.org/10.1007/s11128-017-1603-1 arXiv:arXiv:1411.5949

[64] Mehdi Saeedi, Robert Wille, and Rolf Drechsler. 2011. Synthesis of Quantum Circuits for Linear Nearest Neighbor Architectures. *Quantum Information Processing* 10, 3 (June 2011), 355–377. https://doi.org/10.1007/s11128-010-0201-2

[65] Masahide Sasaki, Alberto Carlini, and Richard Jozsa. 2001. Quantum template matching. *Phys. Rev. A* 64 (Jul 2001), 022317. Issue 2. https://doi.org/10.1103/PhysRevA.64.022317

[66] Kirk Schloegel, George Karypis, and Vipin Kumar. 2003. Sourcebook of Parallel Computing. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, Chapter Graph Partitioning for High-performance Scientific Simulations, 491–541. http://dl.acm.org/citation.cfm?id=941480.941499

[67] Alireza Shafaei, Mehdi Saeedi, and Massoud Pedram. 2013. Optimization of Quantum Circuits for Interaction Distance in Linear Nearest Neighbor Architectures. In *Proceedings of the 50th Annual Design Automation Conference* (Austin, Texas) *(DAC '13)*. ACM, New York, NY, USA, Article 41, 6 pages. https://doi.org/10.1145/2463209.2488785

[68] Yunong Shi, Nelson Leung, Pranav Gokhale, Zane Rossi, David I. Schuster, Henry Hoffman, and Fred T. Chong. 2019. Optimized Compilation of Aggregated Instructions for Realistic Quantum Computers. (2019). https://doi.org/10.1145/3297858.3304018 arXiv:arXiv:1902.01474

[69] Peter W. Shor. 1997. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM J. Comput.* 26, 5 (Oct. 1997), 1484–1509. https://doi.org/10.1137/S0097539795293172

[70] H.D. Simon. 1991. Partitioning of unstructured problems for parallel processing. *Computing Systems in Engineering* 2, 2 (1991), 135 – 148. https://doi.org/10.1016/0956-0521(91)90014-V Parallel Methods on Large-scale Structural Analysis and Physics Applications.

[71] Marcos Yukio Siraichi, Vinícius Fernandes dos Santos, Sylvain Collange, and Fernando Magno Quintao Pereira. 2018. Qubit Allocation. In *Proceedings of the 2018 International Symposium on Code Generation and Optimization* (Vienna, Austria) *(CGO 2018)*. ACM, New York, NY, USA, 113–125. https://doi.org/10.1145/3168822

[72] Francesco Tacchino, Chiara Macchiavello, Dario Gerace, and Daniele Bajoni. 2018. An Artificial Neuron Implemented on an Actual Quantum Processor. arXiv:arXiv:1811.02266

[73] Colin J. Trout, Muyuan Li, Mauricio Gutierrez, Yukai Wu, Sheng-Tao Wang, Luming Duan, and Kenneth R Brown. 2017. Simulating the performance of a distance-3 surface code in a linear ion trap. (2017). https://doi.org/10.1088/1367-2630/aab341 arXiv:arXiv:1710.01378

[74] Davide Venturelli, Minh Do, Eleanor Rieffel, and Jeremy Frank. 2017. Compiling quantum circuits to realistic hardware architectures using temporal planners. 2017 Quantum Sci. Technol. - also related to proceedings of IJCAI 2017, and ICAPS SPARK Workshop 2017. (2017). https://doi.org/10.1088/2058-9565/aaa331 arXiv:arXiv:1705.08927

[75] Andreas Wallraff. 2018. Deterministic Quantum State Transfer and Generation of Remote Entanglement using Microwave Photons. In *APS Meeting Abstracts*.

[76] J. Werschnik and E. K. U. Gross. 2007. Quantum Optimal Control Theory. arXiv:arXiv:0707.1883

[77] Mark Whitney, Nemanja Isailovic, Yatish Patel, and John Kubiatowicz. 2007. Automated Generation of Layout and Control for Quantum Circuits. In *Proceedings of the 4th International Conference on Computing Frontiers* (Ischia, Italy) *(CF '07)*. ACM, New York, NY, USA, 83–94. https://doi.org/10.1145/1242531.1242546

[78] K. Wright, K. M. Beck, S. Debnath, J. M. Amini, Y. Nam, N. Grzesiak, J. S. Chen, N. C. Pisenti, M. Chmielewski, C. Collins, K. M. Hudek, J. Mizrahi, J. D. Wong-Campos, S. Allen, J. Apisdorf, P. Solomon, M. Williams, A. M. Ducore, A. Blinov, S. M. Kreikemeier, V. Chaplin, M. Keesan, C. Monroe, and J. Kim. 2019. Benchmarking an 11-qubit quantum computer. arXiv:arXiv:1903.08181

[79] Xin Zhang, Hong Xiang, Tao Xiang, Li Fu, and Jun Sang. 2018. An efficient quantum circuits optimizing scheme compared with QISKit. arXiv:arXiv:1807.01703

[80] Alwin Zulehner, Alexandru Paler, and Robert Wille. 2017. An Efficient Methodology for Mapping Quantum Circuits to the IBM QX Architectures. arXiv:arXiv:1712.04722