# Memory-Equipped Quantum Architectures: The Power of Random Access

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

David I. Schuster
david.schuster@uchicago.edu
University of Chicago

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

## ABSTRACT

Resonant cavities can be used to extend conventional superconducting transmon-based quantum architectures by adding a few bits of quantum memory to each transmon. Such architectures leverage the long coherence times of cavities creating a "memory-equipped" quantum architecture (MEQC) extending the amount of quantum state a machine can manipulate. However, random access to data will have the greatest effect on improving machine performance. Existing transmon architectures are locally connected and performing gates between distant qubits requires expensive pairwise swaps for execution. Added swap operations increase the probability of errors by increasing both operation count and execution time.

We develop a complete compilation framework with heuristics to optimize for the load-store execution model of MEQC. We reduce the gate count and depth of compiled quantum programs by an average 1.62x and 1.70x, respectively compared to traditional transmon architectures. Based on small noise simulations, MEQC architectures outperform on programs as small as 10 qubits, and in general the probability of no gate errors, dominant in NISQ era, is greater on MEQC. If idle errors become more significant, MEQC will have a greater advantage.

We conclude with an exploration of different architectural choices, such as transmon-transmon connectivity and cavity size, and explore their effect on the performance of the proposed architecture. While we expect due to small initial physical experiments that we have O(10) modes per cavity, the particular choice of cavity size in this 2.5D architecture is an important one. For example, when coherence times are high and we can withstand greater serialization it becomes more advantageous to favor larger cavity sizes. In the early stages of these devices, we expect transmon-transmon interactions to be potentially more expensive than transmon-cavity interactions. Our proposed solution can tolerate potentially up to 12x worse interconnect error.

## CCS CONCEPTS

• **Computer systems organization → Quantum computing**.

## KEYWORDS

quantum computing, compilers, computer architecture


*PACT '20, October 3–7, 2020, Virtual Event, GA, USA*

ACM ISBN 978-1-4503-8075-1/20/10…$15.00
https://doi.org/10.1145/3410463.3414644

**ACM Reference Format:**
Jonathan M. Baker, David I. Schuster, and Frederic T. Chong. 2020. Memory-Equipped Quantum Architectures: The Power of Random Access. In *Proceedings of the 2020 International Conference on Parallel Architectures and Compilation Techniques (PACT '20), October 3–7, 2020, Virtual Event, GA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3410463.3414644

## 1 INTRODUCTION

In the last few years, quantum computing has taken major steps forward towards becoming practical, useful for more than distant, untenable algorithms [11, 25]. More and more devices are becoming available with on the order of 10s of quantum bits, albeit noisy. One of the most important emerging problems is scalability of current hardware in order to execute interesting problems in the current, Noisy-Intermediate Scale Quantum (NISQ) [22], era with the goal to demonstrate realizable quantum advantage or even quantum supremacy [3].

There are several competing technologies for the underlying implementation of qubits such as trapped ions or superconducting circuits. Each of these technologies present unique challenges to scalability, at least without error correction. For example, in superconducting technology with limited connectivity between hardware qubits, numerous additional operations called SWAPs must be added to an input program in order to execute it. For example, in Linear-Nearest-Neighbor and a 2D grid topologies on the order of thousands of additional operations must be inserted for execution, many of which are derived from increased average distance between qubits, as shown in Figure 1. With so many additional operations, the input programs are almost guaranteed to fail. These added operations dramatically increase the execution time of these programs, moving far beyond the coherence limit of current qubits (approximately the lifetime of qubits).

Past efforts have focused primarily on the compilation problem to these small, near-term devices, such as variation aware mappings or SWAP reductions [12, 18, 19, 28, 29]. These techniques have improved the ability of programs to succeed on currently available devices, but do not provide a path to scalability. There are inherit limits in current quantum architectures, such as gate error rates or qubit coherence times. Even with improved error rates or longer qubit lifetimes, the large gate and depth overheads induced by limited connectivity are too cumbersome.

We instead propose a new architecture, Memory-Equipped Quantum Computing (MEQC). We use superconducting qubits equipped with resonator cavities which can store qubits in their modes. Recently, small, physical prototypes of this have been realized and experimented with; however, the architectural implications of this new technology are almost entirely unexplored. For example, in the original experiments, the proposed benefit of this technology was to store less frequently used qubits in long coherent memory.

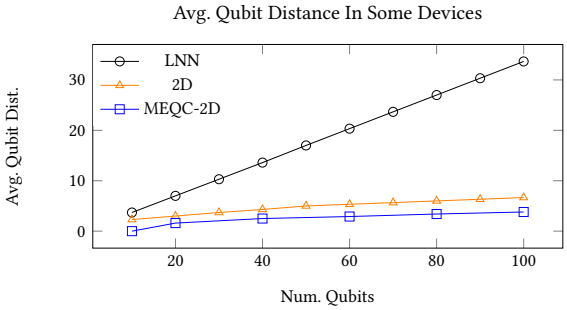


**Figure 1: Average qubit distance on several instances of near-term target architectures. Average qubit distance approximates how many SWAPs are necessary to interact an arbitrary pair of qubits. LNN architectures scale extremely poorly in this metric resulting in a large number of added gates and depth. 2D and MEQC architectures scale much better. We show this translates into reduced number of gates and reduced depth as we scale into the future.**

Qubits stored in the modes of the attached cavity are expected to have an order of magnitude longer coherence times than typical transmon coherence times. This directly will reduce the frequency of idle errors on qubits which are unused. Idle errors are due inability to isolate qubits perfectly from the environment while still being able to manipulate them. In other superconducting devices, all qubits are roughly equally subject to these errors. In the proposed architecture, stored qubits are more isolated, resulting in protection from these idle errors.

This benefit is secondary to another hugely important feature for near- and intermediate- term and that is connectivity. In the proposed architecture, the transmon-cavity technology enables a gate to be applied, via transmon mediation, to any pair of qubits stored in the same cavity. This random access to stored qubits greatly improves local connectivity between qubits in these devices. This translates into significant reduction in compilation overhead, reducing the need for large numbers of SWAPs, but only if the compilation procedure properly accounts for these well connected regions. While experimentalists anticipated the coherence times of cavities to be the critical advantage, we find in our proposed MEQC architecture the primary benefit is this random access. For our proposed architecture, we provide a complete compilation framework transforming input quantum circuits to ones executable in hardware utilizing transmon-local memory. Our framework explicitly maximizes the advantages of both of these features to minimize compilation overhead.

Neither of these gains come for free. In the proposed architecture, in order to execute a gate we must first load the qubit from memory into a transmon which can be manipulated. Therefore, operations on this architecture require on average two additional operations in the form of Loads and Stores. For small programs, this architecture will require more operations than others. Furthermore, since there is only a single operational transmon per cavity, this prevents gates from being executed in parallel on qubits located in the same cavity. The gains of this new architecture in terms of scalability outweigh these downsides, specifically by reducing the number of total operations required to execute input programs.

In summary, we make the following contributions:

- We introduce a new scalable quantum architecture, MEQC, which takes advantage of recent quantum memory-like hardware increasing local qubit connectivity and protects infrequently used qubits from idle errors.
- We develop a full compilation framework for MEQC devices which includes heuristics to maximize time spent in long coherent memory when unused and heuristics for mapping and routing of qubits during execution which minimizes the total number of SWAP operations inserted.
- We demonstrate our system reduces required gate and depth overhead substantially over other competitive options and subsequently increases the chance to succeed.
- We analyze different architecture-specific design choices such as number of modes per cavity, and top-level transmon connectivity. Furthermore, our system is able tolerate up to 12x worse transmon-transmon interconnect error which is expected to be the dominant source of error in systems utilizing transmon-cavity technology in near-term. We conclude our architecture presents a path towards scalability in the near and intermediate term.

## 2 BACKGROUND

### 2.1 Quantum Bits and Quantum Logic

In classical computing, bits exist as either 1 or 0. The quantum analog is the quantum bit, or qubit. Unlike the classical bit, the qubit may exist in a *superposition* of the 0 and 1 state, written as $|0\rangle$ and $|1\rangle$. That is, a qubit exists in a quantum state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ with $\alpha, \beta \in \mathbb{C}$. When a qubit is observed, however, it collapses into a classical bit, observed only ever as a 0 or 1. The outcome of this observation is probabilistic: we observe 0 with probability $|\alpha|^2$ and observe 1 with probability $|\beta|^2$. We call these elements $|0\rangle$ and $|1\rangle$ the computational basis states. When we extend to multiple qubits we have exponentially many basis elements. For example, in a two qubit system the state can exist in a superposition of the states $|00\rangle, |01\rangle, |10\rangle$ and $|11\rangle$.

In classical logic, bits are manipulated via gates which take many bits and output a single output bit which can then be fanned out if needed, that is copies can be made. In contrast, in quantum logic every operation performed has the same number of inputs and outputs and we cannot make copies of our data, i.e. we have a so-called *no cloning theorem*. Furthermore, these operations must be reversible. Multi-qubit operators, for example the controlled NOT gate, or CNOT can be used to achieve quantum entanglement.

On near-term devices there is limited connectivity, that is only certain pairs of physical qubits may interact. In order to interact distant qubits, we can move qubits around via sequences of SWAP operations. These SWAPs are typically decomposed as three CNOT gates, which when two qubit gate error rates are around 1% on these devices can severely limit what can successfully be executed.

### 2.2 Near-Term Quantum Architectures

In the last several years, a number of competing technologies have emerged such as superconducting qubits and trapped ions. These

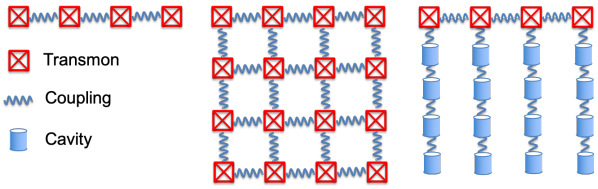


**Figure 2: Both current and the proposed MEQC device. On the left is a LNN device where adjacent superconducting transmons are coupled enabling two qubit interactions. In the center is a 2D mesh architecture common among current manufacturers. On the right is the proposed MEQC architecture with transmons arranged in a line. Each transmon has an attached cavity which stores in memory multiple qubits. To operate on the qubits, they must first be loaded into the transmons.**

have been physically realized in systems containing on the order of 10s of qubits [2, 16, 23]. These devices have limited connectivity with qubit connections specified by an underlying topology. We represent this connectivity as a graph with nodes corresponding to physical qubits and edges indicating valid two-qubit interactions. Some near-term topologies are found in Figure 2. For superconducting technology, common underlying topologies are Linear Nearest Neighbor (LNN) and 2D Mesh connectivity. We use these two styles of architectures as baselines to compare the proposed memory-equipped superconducting architecture (MEQC).

## 2.3 Compilation to Quantum Architectures

In the near-term, quantum programs are typically specified as quantum circuits. In order to execute this program, the circuit is decomposed into only single and two qubit gates. Then, the circuit level qubits are mapped to the physical qubits of the device. Because movement operations are expensive, qubits which interact often in the circuit should be placed in close proximity. For each operation specified in the circuit, if the qubits are already adjacent, nothing additional needs to be done. If they are not adjacent, a sequence of SWAP operations are inserted into the circuit to as to make the qubits adjacent on hardware. This process is known as *routing*.

Due to error, it is critical to minimize the number of added operations to circuits, in this case these SWAP operations. Special care should be taken to map qubits to hardware in an effective manner. It is imperative to compile input programs in a way which requires the least amount of additional overhead in terms of number of gates and in added depth (roughly the length of the critical path from start to end of a program).

## 2.4 Errors on Quantum Devices

Quantum devices in the NISQ era are subjected to fairly significant error rates, somewhere on the order of 1 per 100-1000 operations [15, 27]. As such, the output of a quantum computation may be erroneous and it is common for a program to be run thousands of times to collect a distribution of outputs. In an ideal case, the correct answer appears with substantially higher probability than all of the other wrong answers.

There are a variety of different types of errors which can occur during the execution of a program [21]. The first are coherent errors such as bit-flip or phase-flip errors. These typically occur due to an error in the application of the gate to the qubits. The one and two qubit error rates of the device approximate how often this type of error occurs. These gates are easier to model and we expect this class of errors to be dominant in the near-term. Another type of error is decoherence errors, such as amplitude damping. This type of error is due to interaction with the environment; physical qubits are often kept isolated from the environment to avoid these types of decoherence errors, however, perfect isolation cannot be achieved because in order to do something useful we need to manipulate the qubits. The T1 times, or coherence times, of qubits and the amount of time it takes to execute a gate approximate how frequently these types of errors will occur. A good quantum device has long coherence times and low error rates. Finally, one other common error is crosstalk which occurs when multiple gates (usually two qubit gates) occur in parallel on adjacent sets of qubits. Unfortunately, crosstalk error is hard to approximate in larger systems, but it is suspected crosstalk in our proposed system is no worse than crosstalk in current superconducting systems.

We are interested in designing a new type of scalable quantum architecture, specifically, one with long coherent memory attached to computational transmon qubits. As such we are unable to execute our benchmarks on realizations of these larger-scale devices. Instead, we model error in this and other competing architecture and use simulation results to determine performance. We use the Kraus operator formalism and density matrix simulation allowing us to inject noise channels into an executable circuit specific for a target architecture and approximate how well, here by measuring fidelity, the circuit performs compared to the ideal no noise version.

Simulating quantum systems is hard, requiring a large amount (exponential) of memory [13], and we are only able to do this for small benchmarks. An alternative approach is to use the probabilities of errors and for every operation in the circuit randomly draw if an error occurs or not. A successful program is one in which no errors are drawn, that is we assume any error causes the program to fail. In practice, this isn't always the case but this method provides a simple way to underestimate the probability of success (overestimate the probability of failure) for larger circuits on the order to 50 to 100 qubits. We use this worst-case estimation for programs with many qubits which even with the largest supercomputers would be unable to simulate.

# 3 A MEMORY-EQUIPPED QUANTUM ARCHITECTURE

In this section, we will present the proposed architecture termed "Memory-Equipped Quantum Computing" due to long coherent

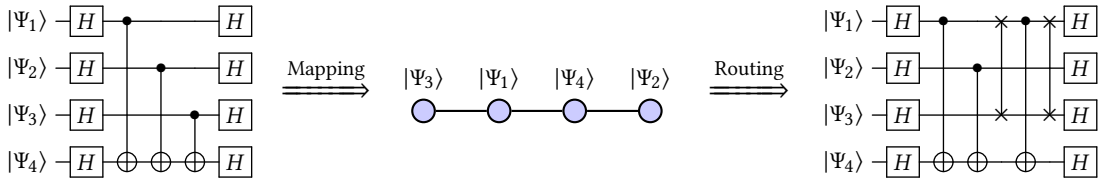


**Figure 3: Compiling to near-term devices is a multi-step process. First we map the logical, circuit qubits to the physical hardware qubits. Based on this placement and the input program, we insert SWAPs in order to interact distant qubits. Here we compile a simple quantum program, Bernstein-Vazirani, to a 4 qubit LNN architecture. Quantum programs, like the input program on the left are a sequence of gates specified on qubits. In this example, based on the given mapping, a pair of SWAPs are required to execute a CNOT between $|\Psi_3\rangle$ and $|\Psi_4\rangle$.**

resonator cavities which resembles memory attached to each of the transmon, computational qubits. Recently, some key hardware components have been physically realized [14, 20]. These initial experiments are significant but limited in scope and their architectural implications remain unexplored. We present an architecture which takes advantage of these memory-like components. These components have several proposed benefits but also have technology-dependent constraints.

Our proposed architecture, while focused on superconducting qubit technology can be extended readily to other technologies such as ion trap devices. As we will see, our cavity model well approximates the operation of an ion trap device, specifically in connectivity. Furthermore, inter-trap communication technology is being demonstrated and is analogous to the transmon-transmon interactions presented here [6, 26]. As such, our algorithms extend to these other technologies, though would require some modification regarding Loads, Stores, etc. which are technology specific.

We focus on the specific technology of [20] which realizes a multi-mode quantum "memory," but in general we naturally support other hardware components which are memory-like. For multi-mode quantum memory, at a high level qubits stored in a mode of a resonator cavity which have long coherence times (T1), about an order of magnitude longer than traditional superconducting qubits. When qubits are unused or idle they can be well isolated from the environment in these cavities, reducing the frequency of decoherence errors occurring. We consider these cavities as a sort of attached quantum memory to the actual computational qubits, here realized as transmons. These transmons in theory can be strung together or connected to each other in configurations similar to other superconducting technology.

The qubits stored in memory cannot be operated on except by special operations called Load and Store (transmon-mode iSWAPs) which essentially transfer the qubit stored in memory to the parent transmon. Therefore, in order to perform a single qubit operation on a qubit $Q_1$ it must first be loaded into a transmon, the operation then is enacted, and then $Q_1$ is replaced back into memory. In order to perform any action on the stored qubits, an additional two gates must be inserted. However, the modes of the resonator can be accessed like random access memory meaning this Load and Store pair is all that is needed. Besides improved resistance to decoherence errors, our architecture has the added benefit of full connectivity between qubits within the same cavity. This means we can interact pairs of qubits in the same cavity without needing to add a large number of SWAPs as we would typically. These transmon-resonator pairs are strung together as in Figure 2.

This does not mean SWAPs are unnecessary altogether. SWAPs are still required in order to interact qubits in different cavities. To interact pairs of qubits located far away, we first load both qubits into the top level transmons. Then we can swap them through a path of connecting transmons at the top level. Once they are co-located in the same cavity, we can perform a two qubit interaction as usual. This architecture reduces the average distance between any pair of qubits meaning reduction in SWAP overhead. This of course comes at the cost of having a required a Load and Store for every operation.

As an example, consider an input program as in Figure 4. On other near-term devices, this program could executed as is, with no additional operations (depending on the chosen gate set the CNOT may need to be converted). On this device, we require no SWAPs but several sets of Loads and Stores.

One other notable difference is the scale of the proposed devices. Our architecture would require many cavities, which occupy a much larger volume (though still small) than transmons. This leads to the need for unique ways of connecting the transmon qubits which may result in worse interconnect error rates. We explore our sensitivity to this later.

## 3.1 MEQC Compilation: Input Program to Executable

We now present our compilation framework which takes as input a program specified as a quantum circuit and outputs a new circuit executable on our architecture, specifically maps program qubits to modes of the hardware cavities and inserts the requisite Loads and Stores as well as any SWAPs that may be necessary to move qubits between cavities. For all of the steps presented below we assume the input circuit has been decomposed into a suitable gate set (the set of gates executable on the device) consisting of only one and two qubit interactions.

From this decomposed circuit, we may produce the *interaction graph* of the circuit, where nodes in this graph correspond to the circuit qubits and the edges are weighted by the number of interactions between the pairs of qubits. Specifically, if the nodes of the interaction graph are $q_1, ..., q_n$ then $w(q_i, q_j)$ is the number of interactions between $q_i$ and $q_j$ in the input program.

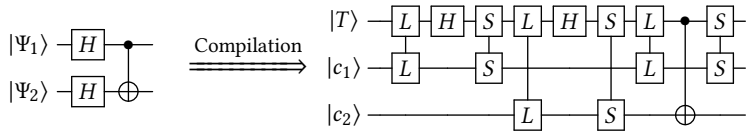


**Figure 4: Compiling a small program to a MEQC device. In this case we map the input qubits $|\Psi_1\rangle$ and $|\Psi_2\rangle$ to one of the two available cavity modes. When executing the gates, we first execute a Load to move the qubit to the connected transmon. The gate is then executed and the qubit is returned to its original mode via a Store. We represent Loads as $L-L$ and Stores as $S-S$.**

## 3.2 Initial Qubit Placement via Graph Partitioning

The first step in the compilation process is to generate a mapping from circuit level qubits to hardware qubits. In this case, we want to map the qubits to modes of the available resonator cavities. In order to avoid needing excessive SWAPs, we want to map qubits which frequently interact to the same cavity. Because there is essentially full connectivity between qubits stored in the same cavity, by placing frequently interacting qubits together we circumvent the need for SWAPs.

Because each cavity is represented as a fully connected graph in the underlying topology, we can perform initial placement by partitioning the interaction graph such that we minimize the number of edge crossings between partitions. Specifically, if we have $k$ many cavities with exactly $p$ resonant modes, we want to partition the interaction graph into $k$ clusters with maximum cluster size of $p$. We do not require a minimum size; if there are disconnected components it will usually be worse to force clusters to be full. We allow for clusters to be non-full by adding in dummy nodes which have weight zero to every real qubit. This problem is well-studied, and we adapt a version of a heuristic Overall Extreme Exchange to do this initial partitioning [17].

Alone, this is insufficient. We want to map the clusters to the physical cavities. If these cavities were fully connected (i.e. every transmon was connected to every other transmon), then every assignment of clusters to cavities is the same, that is in order to interact a pair of qubits located in different cavities we would need exactly one SWAP. However, this is not in general the case and the transmon connectivity may be much less well connected and therefore if *clusters* which interact frequently are distant we may need many more SWAPs. To avoid this, we want to locate these cavities nearby in terms of the device topology.

To handle this, we first choose any mapping of clusters to cavities. Let $P_1, ..., P_n$ be the partitions produced by the partitioning algorithm, which may be empty, and let $C_1, ..., C_n$ be the cavities of the machine. These cavities have a distance between them $d(C_i, C_j)$ give by the shortest path distance in the device topology. This initial mapping is an assignment of partitions to cavities, a bijective map $\varphi$. During the following process, the qubits in the cavities remain fixed and therefore the *weight* between partitions is fixed and is given as

$$W(P_i, P_j) = \sum_{q_i \in P_i} \sum_{q_j \in P_j} w(q_i, q_j)$$

where if $W$ is large then the partitions $P_i$ and $P_j$ should be located nearby each other.

We then repeat the following process. First we compute the *gain*, $g$, obtained by swapping pairs of adjacent partitions. We want to compute if by swapping two partitions they in general get closer to partitions they have a large weight with while simultaneously not getting too far from partitions they were already close to with high weight. Specifically, for every pair of partitions $P_i, P_j$ with $d(\varphi(P_i), \varphi(P_j)) = 1$ we compute the following "forward gain" and "backward gain"

$$g_f(P_i, P_j) = \sum_{\substack{P_k \\ d(\ell_j,\ell_k)<d(\ell_i,\ell_k)}} W(P_i, P_k) - W(P_j, P_k)$$

$$g_b(P_i, P_j) = \sum_{\substack{P_k \\ d(\ell_j,\ell_k)>d(\ell_i,\ell_k)}} W(P_j, P_k) - W(P_i, P_k)$$

where $\ell_i = \varphi(P_i)$. Then the net gain is given as

$$g(P_i, P_j) = g_f + g_b$$

We choose the pair $P_i, P_j$ with $g(P_i, P_j) > 0$ maximized and swap them. Specifically, we set $\varphi(P_i), \varphi(P_j) = \varphi(P_j), \varphi(P_i)$ and repeat until no more swaps can be made. This heuristic helps locate strongly weighted partitions together on the topology in order to reduce needed SWAPs to interact qubits between them.

## 3.3 Scheduling and Routing: Inserting Loads, Stores, and SWAPs

In this section, we will briefly describe the baseline method for compilation to other current proposed scalable architectures specifically the LNN and 2D mesh, the benchmarks we test on, and methods of evaluation.

In other common quantum architectures, we only need to insert SWAPs in order to make interacting qubits adjacent. Here, we must also insert Loads and Stores. For an input circuit, we define a *moment* as the maximal set of operations which can be performed simultaneously, or in parallel. We assume all input programs have operations occurring as soon as possible. In each moment, there are operations which can already be executed, that is all one-qubit operations and all two-qubit operations for which both qubits are co-located in the same cavity. We execute these first, in an arbitrary order.

For each single qubit operation we insert a Load, moving the qubit into the cavity's transmon, execute the gate on the transmon, then Store the qubit in it's original location in the resonator. For a two qubit controlled gate, we insert a Load to move the control into the transmon, we execute the gate, and then Store the control in its original location.

| | QAOA | | | QFT-Adder | | | Generalized-Toffoli | | | Rand-0.4 | | | Rand-0.6 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Num. Qubits | 10 | 20 | 50 | 10 | 20 | 50 | 9 | 19 | 49 | 10 | 20 | 50 | 10 | 20 | 50 |
| Num. of Gates | 290 | 580 | 1450 | 45 | 165 | 975 | 119 | 289 | 799 | 19 | 72 | 483 | 24 | 127 | 757 |
| Circuit Depth | 125 | 203 | 276 | 14 | 29 | 74 | 52 | 73 | 95 | 6 | 11 | 27 | 7 | 17 | 37 |

**Table 1: Benchmarks and some of their properties.**

| Model | $p_1$ | $p_2$ | $\Delta_1$ | $\Delta_2$ | $\Delta_\ell$ | $T_1$ | $T_{1,t}$ | $T_{1,c}$ |
|---|---|---|---|---|---|---|---|---|
| SC-Current | 0.001 | 0.01 | 100 ns | 300 ns | - | 73 $\mu$s | - | - |
| MEQC-Current | 0.001 | 0.01 | 100 ns | 250 ns | 150 ns | - | 150 $\mu$s | 900 $\mu$s |

**Table 2: Error model details for current systems [15, 20].**

Operations which cannot be executed in this moment are ones in which the two qubits are located in different cavities, call them $q_0$ and $q_1$. SWAPs are inserted which modify the qubit mapping. We determine the sequence of SWAPs based on a heuristic algorithm with two goals. The first is to minimize the total number of SWAPs inserted. The second is to displace qubits in the cavities along the path from $q_0$ to $q_1$ which are least attached to their current location, i.e will interact the least in the future with qubits in its current cavity.

We begin by computing an updated interaction graph for the program, considering only future moments, updating edge weights of the original interaction graph to be only the number of future interactions between pairs of qubits, $w(q_i, q_j)$. Let $d(C_i, C_j)$ be the distances between cavities on the device, as before. Let $\lambda$ be a map from qubits to cavities. The distance between qubits $q_0, q_1$ is then given as $d(\lambda(q_0), \lambda(q_1))$. Then, for every cavity $C_i$ with $d(\lambda(q_0), C_i) = 1$ and $d(C_i, \lambda(q_1) < d(\lambda(q_0), \lambda(q_1))$ we compute the following *gain* to swap $q_0$ with some element $q_t \in C_i$

$$gain(q_0, q_t) = \sum_{q_i \in C_i \setminus q_t} [w(q_0, q_i) - w(q_t, q_i)] + \sum_{q_i \in \lambda(q_0) \setminus q_0} w(q_t, q_i)$$

In the same way we compute $gain(q_1, q_t)$ for $q_t \in C_i$ for every $C_i$ with $d(\lambda(q_1), C_i) = 1$ and $d(C_i, \lambda(q_0)) < d(\lambda(q_1), \lambda(q_0))$. This captures how much is gained (or lost) by making a particular swap of qubits between cavities while maintaining we only consider cavities strictly closer to the target qubit. This ensures we always get closer to our target. We choose to swap the qubits with the greatest gain, and then repeat. Notice we allow either $q_0$ or $q_1$ to be moved at each step, so the qubits may meet in some cavity between them. The SWAPs are executed in order by Loading both qubits to be swapped into their transmons, execute the SWAP between transmons and Store the qubits back into their new cavity. This may introduce redundant Loads and Stores, but they can be eliminated via an optimization procedure (Sec 3.4).

The gain value above can be modified to favor swapping with cavities which are closest to the target, however, we choose this method because it allows us to sometimes anticipate future interactions of qubits eliminating the need for some future SWAPs. Once all the operations of a moment have been executed, we move to the next moment and perform the same procedure.

## 3.4 Optimizations

The above algorithms produce a valid executable of an input circuit, however with more gates than necessary due to an invariant which requires every operation to be initiated with a Load and terminated with a Store. For example, if we wanted to execute two single qubit gates on the same qubit in sequence we would need to insert two Loads and two Stores resulting in a total of 6 gates. We can eliminate redundant Loads and Stores by simply checking if a Store from a transmon to a cavity mode is followed immediately by a Load from the same mode to the same transmon.

Another important optimization is to ensure that qubits are loaded from memory only immediately when they are being used. Because T1 times are much lower in the transmon, we do not want any qubit to idle there and instead want to make sure it persists in a cavity for as long as its unused to protect it from decoherence errors. We perform both of these optimizations in our compilation process.

## 3.5 Fundamental Trade Offs

The proposed architecture has several fundamental trade offs distinct from currently operational architectures. The first is the requirement for qubits to be loaded from memory in order to be operated on. This forces a greater degree of sequentialism since only a single operation per cavity can be performed at once. This also means, for every operation we need approximately 3 times as many gates, the Load, the Store, and the gate itself.

This architecture has several benefits which outweigh these costs as we scale to larger devices. When qubits are not being operated on they are stored in memory with coherence times substantially longer than the computational qubits, the transmons. The cavities are random access for the transmon, providing full connectivity between qubits in the same cavity and reducing the average distance between qubits on the device meaning many fewer SWAPs required. Parallelism is still achievable in this architecture. Each transmon-cavity pair operates independently; a well-partitioned algorithm with some degree of parallelism will still be executed in parallel.

### 3.6 Limitations and Potential

The underlying technology which serves as a basis for our proposed architecture is less developed than currently available commercial hardware like that of IBM or Rigetti. However, the current limits such as greater errors, are not fundamental and the technology is expected to evolve with similar trajectory of other current hardware. Our goal is to demonstrate the power of the unique advantages provided by equipping transmons with localized memory. In order to evaluate this, we experiment with more speculative error rates and coherence times which are believed to reflect the potential of the underlying hardware. One possible important advantage not evaluated here is consistency. Current manufacturers have struggled to scale current designs beyond handfuls of transmon qubits with consistently low error rates. With MEQC, obtaining the same number of logical qubits requires fewer total transmons which may help in reducing overall variability by manufacturing a smaller number but higher quality set of transmons. MEQC, while currently in developmental stages, is not fundamentally limited.

## 4 EXPERIMENTAL SETUP

### 4.1 Compiling to LNN and 2D Mesh Architectures

The compilation procedure for these devices closely reflects the compilation procedure for our proposed MEQC architecture. For each of these, we utilize the heuristic found in [19] to give an initial mapping of circuit qubits to hardware qubits. Operations on these architectures are assumed to be done in parallel if possible and no loading or storing is required. If an operation cannot be done given the initial mapping, the interacting qubits are moved via SWAPs to each other, the operation is performed, and then the qubits are moved back to their original position. This common SWAP strategy assumes the initial qubit mapping is a good one, minimizing the average total distance between qubits which interact frequently and consequently number of required SWAPs.

In order to limit the number of gates applied to specific qubits on the 2D mesh architectures, we look at all shortest paths between interacting qubits given by the bounding rectangle between them. For each path in this rectangle, we choose the path which uses qubits with the fewest number of past and future uses. SWAP paths are done in parallel if possible.

These devices are denoted as LNN-x and 2D-n-m where $x$ is the number of qubits in the chain, and $n, m$ are the dimensions of the mesh. The average distance between qubits in 2D and LNN architectures scales much worse than an MEQC device, as noted in Figure 1, meaning on average to interact a pair of qubits many more SWAPs will be required even with MEQC transmons connected poorly.

### 4.2 Benchmarks

We evaluate our proposed architecture compared to other competing designs by compiling a range of both parallel and serial NISQ applications, see Table 1.

*4.2.1 Arithmetic Circuits.* Many important quantum algorithms like Shor's algorithm make use of arithmetic circuits. Many of these circuits like modular exponentiation are beyond the NISQ era. Other smaller arithmetics like addition are much more practical. There are several efficient implementation of these circuits like the Cuccaro Adder [9] and the QFT (Quantum Fourier Transform) Adder [24]. These circuits are highly sequential for a majority of their execution. We focus on the QFT Adder as a representative for this class of circuits. Unlike the Cuccaro adder, the QFT adder has a highly parallel section in the middle of its execution.

*4.2.2 The Generalized Toffoli.* The Toffoli gate, and its generalized form, is well studied as well as its decompositions. It has many practical uses in a number of applications over a wide range of sizes for example in Grover's search algorithm, larger arithmetic circuits, etc. We consider a generalized decomposition using ancilla given by [4] which allows this operation to be highly parallel, specifically in logarithmic depth.

*4.2.3 QAOA - MAX-CUT.* One of the most promising algorithms for NISQ era devices is optimization problems. One such problem is QAOA, quantum approximate optimization algorithm [10], which can be used to find approximation solutions to combinatorial optimization problems such as MAX-CUT. The parallelism in this circuit is dependent on the underlying graph in the input problem. We run our QAOA on 4-regular graphs with $n$ nodes, $n$ the number of qubits.

*4.2.4 Random Circuits.* Finally, we generate a number of random circuits which tend to have very low amounts of parallelism. To generate these circuits, over $n$ trials we select with probability $p$ if an interaction exists between two qubits. If it does, we insert a two qubit interaction between them. We explore $p = 0.4$ and $p = 0.6$ for more sparse and more dense random circuits, respectively.

### 4.3 Evaluation

When evaluating systems, we use the error model details laid out in Table 2, where $p_1$ be the probability of an error occurring on a single qubit operation, $p_2$ be the probability of an error occurring on a two qubit operation, $\Delta_1$ the duration of a one qubit gate, $\Delta_2$ the duration of a two qubit gate, $\Delta_\ell$ the duration of a Load or Store, $T_1$ the coherence time of a superconducting qubit in a traditional architecture, $T_{1,c}$ the coherence time of a qubit located in the cavity, and $T_{1,t}$ the coherence time of a qubit located in the transmon of a MEQC device. A dash indicates the value is not relevant to the architecture.

For small circuits and devices, we are able to perform full density matrix simulations and compute the fidelity of circuits compiled for different architectures with noise channels based on various error models. For these simulations, we use typical superconducting error rates for one and two qubit gate errors. Similarly, we use the T1 times provided from [15] for current SC devices and T1 from [20] for the proposed MEQC architecture as well as gate times. For these simulations, we use the Kraus operator formalism [21] for noise simulation in which coherent error channels (bit-flip and phase-flip operators) are symmetric and amplitude damping probability is a function of the T1 times. For our architecture, we have two different T1 times and we apply amplitude damping as a function of the $T_{1,c}$ when qubits are present in the cavity and $T_{1,t}$ when being operated on.

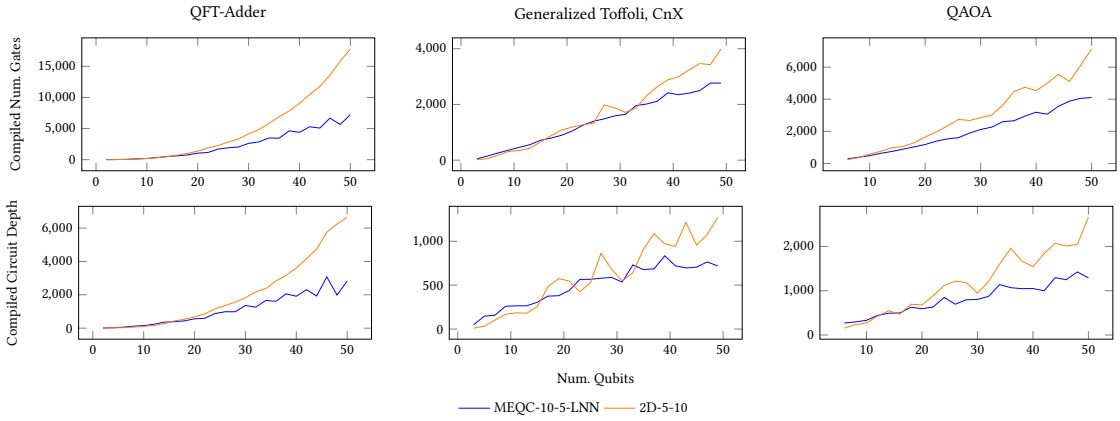


**Figure 5: The scaling of depth and gate count in a subset of our benchmarks. LNN-50 is omitted because it adds substantially more gates than both of these architectures and as such is not competitive. In many cases, 2D-5-10 is competitive with the proposed architecture, however, clear separation emerges in all cases.**

Unfortunately, simulation for even moderate sized programs is extremely hard. For larger circuits we will use an approximation method to determine an estimate for the probability of success of a circuit. For each gate in the circuit, with given probability we draw if an error occurs. If *any* error occurs during the entire circuit, we consider the program to have failed. This is only a rough approximation and will be directly related to the number of gates in the circuit. In this model, we are unable to quantify the effect of amplitude damping as a result of different T1 times, however, it should be noted in general longer coherence times mean qubits are well isolated from the environment and more protected from decoherence errors.

A more general metric we will use is gate count and depth of the compiled circuit. Usually, as the number of gates increases the probability of success drops and as the depth increases our computation will approach the coherence limit and so it is best to keep both as small as possible.

We explore a variety of different MEQC arrangements, specifically exploring different cavity sizes as well as different arrangements of the transmons. We will abbreviate these machines as MEQC-x-y-z with $x$ the number of transmons, $y$ the number of modes per cavity, and $z$ the arrangement of the transmons, for example LNN for an arrangement in a chain, 2D to refer to a mesh, and Full for full connectivity between transmons. For example, MEQC-2-5-LNN means an MEQC design with 2 transmons connected in a chain each with a single attached cavity containing 5 resonant modes. This device would be able to store 10 qubits in memory. A 2D MEQC can be built by building the chain of cavities in the 3rd dimension [8] and any effect on communication error in the mesh is discussed in Section 5.4.

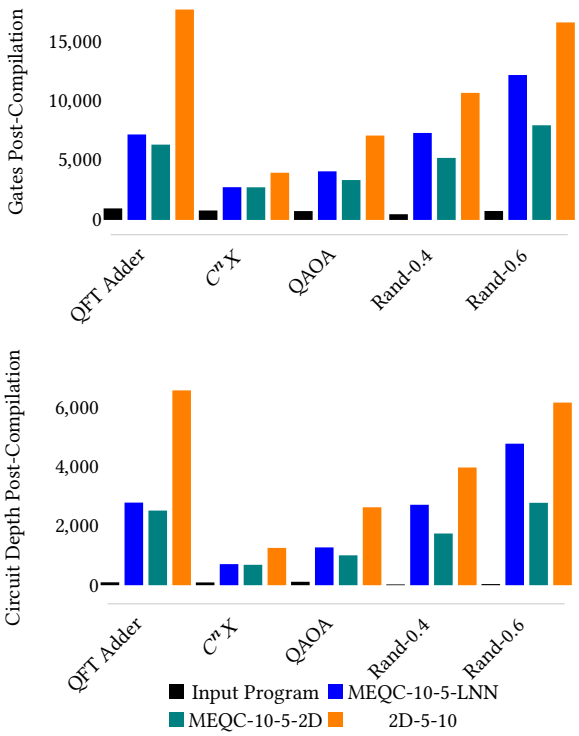


**Figure 6: Post-compilation number of gates and circuit depth for 50 qubit input programs on all benchmarks. In every case, MEQC with transmons arranged with as a chain or a mesh shows improvement over a more standard 2D mesh qubit arrangement. The increase in gate count in MEQC architectures is approximately 60% due to loads and stores and the rest from SWAPs. By requiring fewer gates, we reduce the possibility of gate-induced error.**

## 5 RESULTS AND DISCUSSION

### 5.1 Depth and Gate Count Scaling

As noted before, in the proposed MEQC architecture every gate requires approximately two additional operations, the Load and

| Benchmark | Factor Gate Improvement | Factor Depth Improvement |
|---|---|---|
| QFT-Adder | 2.46x | 2.35x |
| $C^nX$ | 1.44x | 1.76x |
| QAOA | 1.74x | 2.06x |
| Rand-0.4 | 1.46x | 1.46x |
| Rand-0.6 | 1.36x | 1.29x |
| Harmonic Mean | 1.62x | 1.70x |

**Table 3: Summary of the improvements on 50 qubit benchmarks for MEQC-10-5-2D over 2D-5-10. In all cases, we see strict improvement.**

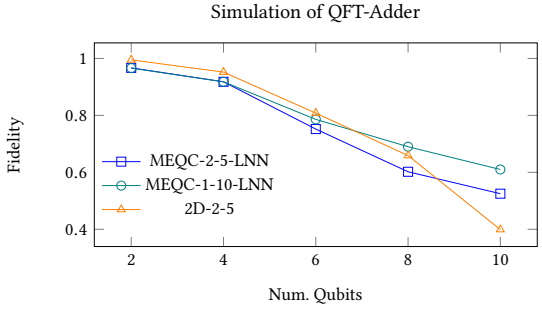


**Figure 7: Output fidelity for full density simulations of the QFT Adder on 2-10 qubits. Even with more gates in these small instances, programs compiled to MEQC devices are competitive, and at 10 qubits we see the start of advantage.**

the Store. Furthermore, this approach induces a large degree serialization, specifically when multiple operations are scheduled to be done on qubits in the same cavity they cannot be executed in parallel. However, what we lose in some serialization and required extra operations, we make up for in full connectivity within cavity and reduced average distance between qubits.

In Figure 5 we compare the scaling of gate counts and depth in a subset of the benchmarks to a 2D and MEQC device. We note all LNN devices with 50 qubits insert dramatically more SWAPs and are completely infeasible, and as such they are omitted. In Figure 5, 2D-5-10 and MEQC-5-10-LNN are competitive in both gate count and depth, with a clear separation emerging as we reach the limits of the devices. In Figure 6, we compare circuit depth and gate count post compilation for 50 qubit inputs for all benchmarks. In all cases, the number of gates required for execution is much greater than the input program. However, MEQC with transmons connected as either LNN or 2D both improve over a traditional 2D architecture. We expect this separation to be even more pronounced as the devices scale up. For reference, in MEQC architectures approximately 60% of added gates are loads and stores with the remainder due to added SWAPs.

While we might first anticipate architectures which permit every qubit to be operated on in parallel if needed to outperform our proposed architecture which forces only a small subset of qubits to be operated on at a time, it turns out communication limitations on other near-term devices eliminates this advantage. The number of SWAPs inserted to make a program executable scales extremely poorly. A greater degree of connectivity in near-term devices, and therefore reduced average distance between qubits, is critical to the performance of a program. The insertion of SWAPs itself induces a degree of serialization even in 2D or LNN architectures. Even if a program is written maximizing parallel operations, compilation procedures to transform a program into one satisfying connectivity constraints can evaporate this advantage.

Even though more traditional architectures provide the mechanism for large amounts of parallelism, it may be extraneous. In this case, qubits are still subject to the same low T1 times even when not being operated on. There is no mechanism to protect these qubits while being unused. Initially, MEQC was appealing because it avoided this issue. When qubits are not needed, they can be stored in memory with long T1 times. Random access memory and full connectivity within cavity provides a much more realizable advantage. These architectural details were previously unexplored by physical experimentalists, but MEQC indicates these new memory technologies are a path towards scalability in the near-term.

## 5.2 Effect On Probability of Success: An Estimation

In non-error corrected devices an input program is run thousands of times to obtain a distribution of answers and if run without error we expect the correct answer to appear with the highest probability. These systems have moderate errors and intuitively we expect as the number of gates required for execution increases the probability with with a program succeeds diminishes. Similarly, as the depth of a program increases, perhaps because parallelism is sequentialized or communication operations like SWAPs delay input program's execution, the qubits are more and more likely to be subject to decoherence errors. This also leads to a decreased chance of the correct answer appearing with highest probability in the output distribution.

In order to evaluate the effect of this system on the probability of success, we simulate some small instances of the QFT-Adder benchmark. We perform full density simulations by injecting into the final compiled circuit both coherent errors, or gate errors, with probability given in Table 2 and decoherence errors, specifically depolarizing errors with probability given as a function of the T1 times and gate durations of Table 2. Specifically, we used Google's quantum framework Cirq [1] which contains a full density simulator. We run this simulator on the compiled circuit, which results in the ideal outcome matrix, and the same circuit except with the appropriate error channels which results in a noisy outcome matrix. We compute the pseudo-metric fidelity to evaluate how close the noisy outcome is to the ideal outcome. In Figure 7 we show the resulting fidelities for one benchmark. Despite fewer gates and less depth in small circuits compiled to the traditional 2D architecture, MEQC is competitive and begins to edge out the 2D architecture beginning around 10 qubits, indicating the significantly longer T1 times of qubits stored in memory do indeed protect these qubits. MEQC-2D is roughly the same performance as MEQC-LNN in these small programs and is omitted for clarity.

Unfortunately, modeling errors via simulation is difficult to do for even moderately sized programs, requiring exponential space and

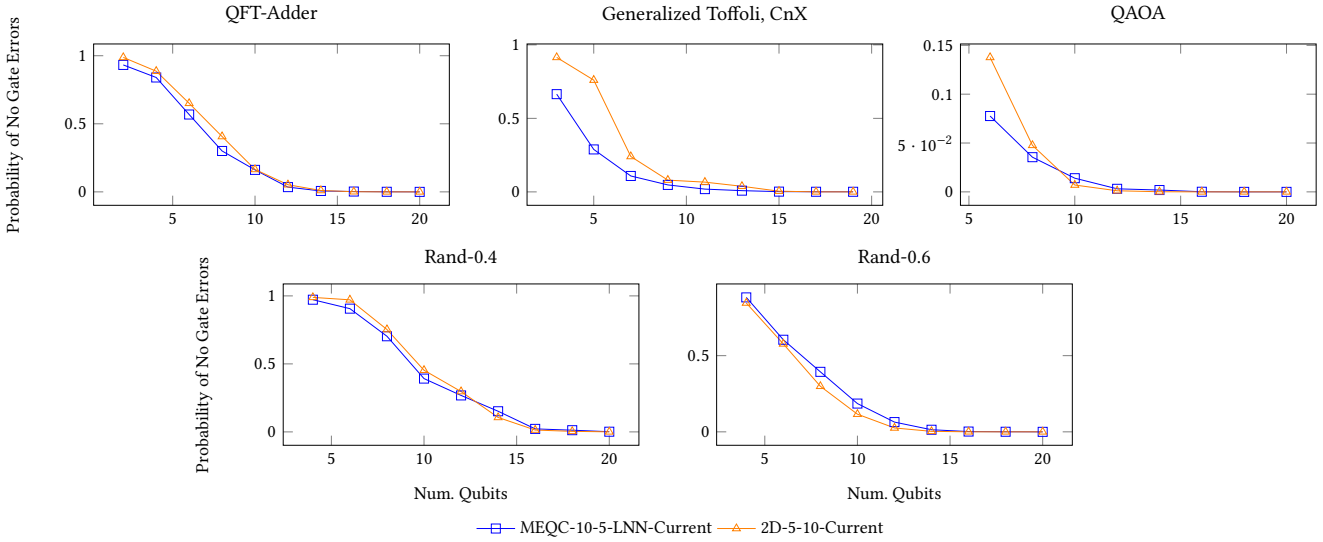


**Figure 8: An estimation of if no gate errors occur in small program instances. As noted, this only accounts for errors due to one and two qubit gate errors and is not influenced by decoherence errors. Larger is better and in general programs compiled to our proposed architecture are competitive or better than a 2D architecture. We expect with better T1 times and better gate and depth scaling, our architecture will outperform, by increasing the likelihood or successful execution, by a larger margin as programs scale and gate errors improve. All data points were obtained by running 8000 trials of the input compiled input program.**

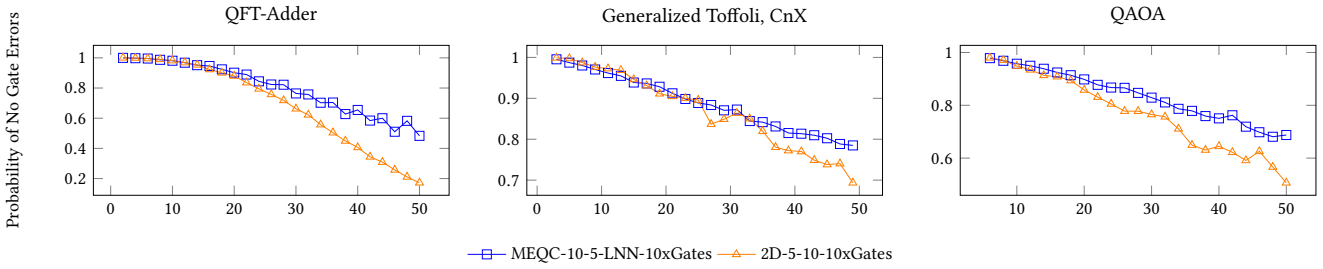


**Figure 9: With 100x better gates, we begin to see the effect of improved compilation. Specifically, by reducing the total number of gates required for execution on the proposed MEQC devices we reduce the probability of a program failing due to gate errors. Furthermore, with substantially longer T1 times in cavity, qubits stored in memory are protected from decoherence errors. All data points are obtained by running 8000 trials of the input program compiled to the two target architectures.**

time. More generally, we rely on an approximation to account for how added gates affect the output of a program. We approximate the probability of no gate errors occurring given the error rates in Table 2. This will not account for decoherence errors due to interaction with the environment. However, the T1 times of the proposed device are significantly longer than those of other competing devices. We expect with reduced depth and larger coherence times that programs compiled to MEQC architectures will be less affected by this type of error as we scale, as indicated by the small simulations.

Given current error rates of typical superconducting devices, on all benchmarks up to size 20 we notice competitive if not better probability of no gate errors as in Figure 8. In order to better distinguish the effects of Figure 5 on this probability, in Figure 9 we explore a potential set of futuristic error rates, specifically 100x better gate errors. In this case, we begin to be able to identify the advantage of our proposed architecture with separation emerging on even moderate sized input programs. We expect even greater advantage when all error types are considered.

## 5.3 Sensitivity to Arrangement and Cavity Size

One notable feature of Figure 6 is that better connected transmon qubits results in fewer gates and less depth. In order to further evaluate this we study two adjustable parameters in the proposed MEQC architecture: the number of modes in resonator cavity and the top-level connectivity of the transmon qubits. Specifically, we

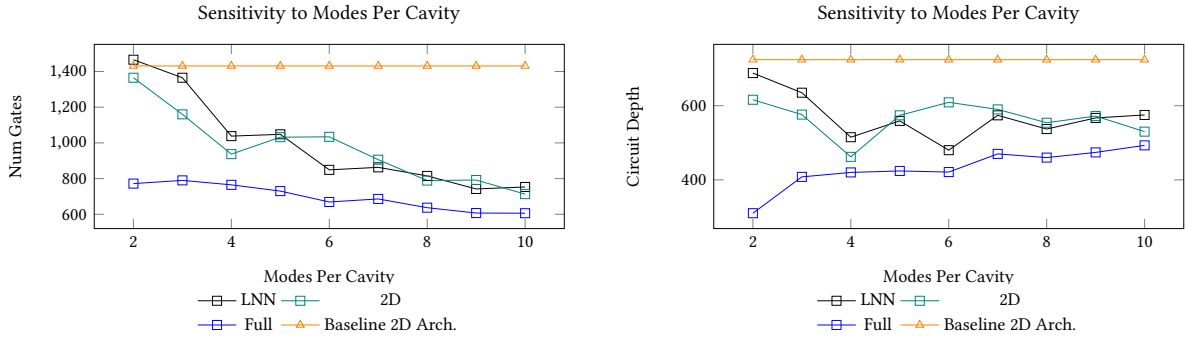


**Figure 10: Gates and depth of 20 qubit QFT Adder compiled to MEQC architectures with different transmon connectivity and varying cavity sizes. As the number of qubits per cavity increases, we expect the average qubit distance to be reduced meaning fewer SWAPs necessary. However, in MEQC devices operations on qubits in the same cavity cannot be done in parallel. Therefore, we expect lose some degree of parallelism, hence the increase in depth.**

study LNN, 2D, and Full connectivity between the transmons as well as cavities of various size.

We expect as the number of modes in the cavity increase, the number of SWAPs, and hence the total gates, required will decrease because of improved average qubit distance. The compilation procedure will prefer to place qubits in these large, well-connected regions of the machine because of this. However, as we noted previously, operations cannot be performed in parallel on qubits co-located in the same cavity. We expect this corresponds to an increase in overall program depth because of reduced parallel operations. We study this tradeoff in Figure 10, in which we study three different arrangements of 10 transmons with increasing cavity size for a 20 qubit QFT Adder input program.

While full connectivity shows consistent improvement, it is only very slight advantage. We expect as the size of the input program increases this will become more important but for near-term devices it suggests we do not need as well connected transmons and what matters most is the well connectedness of the cavity itself. The best choice of modes per cavity here is 5, a balance between number of gates and depth, though these curves are a function of the particular input program. For example, for the 50 qubit QFT Adder of Figure 6 there is only marginal improvement by moving to a 2D connectivity of the transmons. For other benchmarks, this gain is larger. This also demonstrates if we know gate errors will be much more dominant than idle errors, we can choose to favor designs with larger cavities.

## 5.4 Sensitivity to Interconnect Error Rates

In each of the previous studies, we assume the error rates of SWAPs and communication *between* transmons of MEQC devices is the same as they would be in other more traditional architectures. Demonstrations of this communication protocol have achieved less desirable fidelity, sometimes with error several factors worse than SWAPs in current devices [7, 8]. We call this interconnect error. As we've seen, the scaling of both number of gates and depth is better for the proposed MEQC device and we study the degree to which we can tolerate this greater interconnect error.

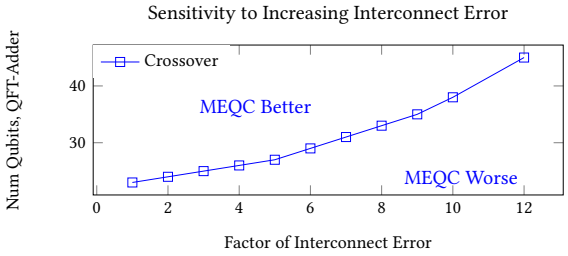


**Figure 11: Crossover points for various interconnect error rates of the QFT-Adder benchmark. Interconnect in MEQC devices may not be as good as SWAPs in traditional architectures. We study how much interconnect error we can tolerate in the NISQ target of 100 qubit devices with $10^{-5}$ two qubit error rates. We find we can tolerate up to 12x worse interconnect errors, provided programs of at least size 52.**

In Figure 11, we use fixed 100 qubit machines, 2D-10-10 and MEQC-10-10-2D, with two qubit error rates 1000x better than current error rates (e.g. two qubit errors of $10^{-5}$), target machines for the NISQ era [5]. We scale the error rate of gates occurring between transmons of the MEQC devices and use our approximation method as before to predict the probability of no gate errors occurring. We locate the crossover points, the program size where it becomes advantageous to use our architecture.

The crossover points are depicted in Figure 11 for error factors up to 12x worse interconnect error on the QFT-Adder. We find for NISQ devices up to 100 qubits, a 2D MEQC can tolerate up to 12x worse interconnect error as other 2D architectures. As the number of qubits $n$ grows, the random access advantage of MEQC grows substantially (as long as $n$ does not overly dominate the maximum memory bank size of 10 cavities per transmon).

## REFERENCES

[1] 2018. Cirq: A python framework for creating, editing, and invoking Noisy Intermediate Scale Quantum (NISQ) circuits. https://github.com/quantumlib/cirq.
[2] 2018. Quantum devices simulators. https://www.research.ibm.com/ibm-q/technology/devices/

[3] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando G. S. L. Brandao, David A. Buell, Brian Burkett, Yu Chen, Zijun Chen, Ben Chiaro, Roberto Collins, William Courtney, Andrew Dunsworth, Edward Farhi, Brooks Foxen, Austin Fowler, Craig Gidney, Marissa Giustina, Rob Graff, Keith Guerin, Steve Habegger, Matthew P. Harrigan, Michael J. Hartmann, Alan Ho, Markus Hoffmann, Trent Huang, Travis S. Humble, Sergei V. Isakov, Evan Jeffrey, Zhang Jiang, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Paul V. Klimov, Sergey Knysh, Alexander Korotkov, Fedor Kostritsa, David Landhuis, Mike Lindmark, Erik Lucero, Dmitry Lyakh, Salvatore Mandrà, Jarrod R. McClean, Matthew McEwen, Anthony Megrant, Xiao Mi, Kristel Michielsen, Masoud Mohseni, Josh Mutus, Ofer Naaman, Matthew Neeley, Charles Neill, Murphy Yuezhen Niu, Eric Ostby, Andre Petukhov, John C. Platt, Chris Quintana, Eleanor G. Rieffel, Pedram Roushan, Nicholas C. Rubin, Daniel Sank, Kevin J. Satzinger, Vadim Smelyanskiy, Kevin J. Sung, Matthew D. Trevithick, Amit Vainsencher, Benjamin Villalonga, Theodore White, Z. Jamie Yao, Ping Yeh, Adam Zalcman, Hartmut Neven, and John M. Martinis. 2019. Quantum supremacy using a programmable superconducting processor. *Nature* 574, 7779 (2019), 505–510. https://doi.org/10.1038/s41586-019-1666-5

[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] Lev S Bishop, Sergey Bravyi, Andrew Cross, Jay M Gambetta, and John Smolin. 2017. Quantum volume. *Quantum Volume. Technical Report* (2017).

[6] 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).

[7] P Campagne-Ibarcq, E Zalys-Geller, A Narla, S Shankar, P Reinhold, L Burkhart, C Axline, W Pfaff, L Frunzio, RJ Schoelkopf, and Devoret RH. 2018. Deterministic remote entanglement of superconducting circuits through microwave two-photon transitions. *Physical review letters* 120, 20 (2018), 200501.

[8] Kevin S Chou, Jacob Z Blumoff, Christopher S Wang, Philip C Reinhold, Christopher J Axline, Yvonne Y Gao, Luigi Frunzio, MH Devoret, Liang Jiang, and RJ Schoelkopf. 2018. Deterministic teleportation of a quantum gate between two logical qubits. *Nature* 561, 7723 (2018), 368.

[9] 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/0410184 [quant-ph]

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

[11] Lov K. Grover. 1996. A Fast Quantum Mechanical Algorithm for Database Search. In *Annual ACM Symposium on Theory of Computing*. ACM, 212–219.

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

[13] Thomas Häner and Damian S Steiger. 2017. 0.5 petabyte simulation of a 45-qubit quantum circuit. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM, 33.

[14] Connor T Hann, Chang-Ling Zou, Yaxing Zhang, Yiwen Chu, Robert J Schoelkopf, Steven M Girvin, and Liang Jiang. 2019. Hardware-efficient quantum random access memory with hybrid quantum acoustic systems. *arXiv preprint arXiv:1906.11340* (2019).

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

[16] Julian Kelly. 2018. A preview of Bristlecone, Google's new quantum processor. *Google Research Blog* 5 (2018).

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

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

[19] 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

[20] RK Naik, N Leung, S Chakram, Peter Groszkowski, Y Lu, N Earnest, DC McKay, Jens Koch, and DI Schuster. 2017. Random access quantum information processors using multimode circuit quantum electrodynamics. *Nature communications* 8, 1 (2017), 1904.

[21] 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.

[22] 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

[23] Chad Rigetti. 2018. The Rigetti 128-qubit chip and what it means for quantum. *Medium* (2018).

[24] 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

[25] 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

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

[27] 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

[28] 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

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