Qurzon: A Prototype for a Divide and Conquer Based Quantum Compiler for Distributed Quantum Systems

Preprint, compiled November 25, 2021

Turbasu Chatterjee
Maulana Abul Kalam Azad University of Technology
turbasu.chatterjee@gmail.com

Arnav Das
Kazi Nazrul University
arnav.das88@gmail.com

Shah Ishmam Mohtashim
University of Dhaka
sishmam51@gmail.com

Amit Saha
Atos
abamitsaha@gmail.com

Amlan Chakrabarti
University of Calcutta
acakcs@caluniv.ac.in

Abstract

When working with algorithms on quantum devices, quantum memory becomes a crucial bottleneck due to low qubit count in NISQ-era devices. In this context, the concept of ‘divide and compute’, wherein a quantum circuit is broken into several subcircuits and executed separately, while stitching the results of the circuits via classical post-processing, becomes a viable option, especially in NISQ-era devices. This paper introduces Qurzon, a proposed novel quantum compiler that incorporates the marriage of techniques of divide and compute with the state-of-the-art algorithms of optimal qubit placement for executing on real quantum devices. A scheduling algorithm is also introduced within the compiler that can explore the power of distributed quantum computing while paving the way for quantum parallelism for large algorithms. Several benchmark circuits have been executed using the compiler, thereby demonstrating the power of the divide and compute when working with real NISQ-era quantum devices.

1 Introduction

Quantum Computing is the future promise of computing with far-reaching impacts in diverse scientific fields [1] [2] [3] [4]. It should deliver speed up in many problems of those fields ranging from machine learning [5] [6] to natural sciences [7]. These absolute speedups will occur when large-scale, universal, fault-tolerant quantum computers come into being. Till then, the challenge of the research community is to utilize currently available noisy intermediate-scale quantum computers (NISQ-era devices) to their full potential [2]. These NISQ-era devices are bound by noise levels, limited coherence times of the qubits, scalability, etc [8]. Thus, researchers are bound to only using small-scale quantum computers with limited and differing connectivity among qubits of different quantum devices.

For this reason, this paper proposes Qurzon: a divide and conquer based quantum compiler. The proposed quantum compiler attempts to make many exponential size problems tractable using current NISQ-era quantum computers by cutting the quantum circuits for those problems to subcircuits that fit the present device architectures. Extending that notion, Qurzon also implements optimal qubit routing for these NISQ-era devices. The nature of NISQ-era devices is that each quantum device is characterized by its inherent qubit connectivity or so-called qubit topology. Qurzon considers these qubit topologies and implements an optimal qubit routing scheme for the subcircuits formed by the cut algorithm.

For this paper, the engineering of Qurzon, a combination of recursive circuit cutting and optimal qubit routing [9], is explained and implemented against existing circuit optimization methods. A significant quantitative advantage is shown that makes Qurzon the state-of-the-art compiler amongst all existing ones found in literature [10] [11] [12] [13], thus improving on, and paving the way for future work on quantum multiprocessor computing.

The key contributions of this paper are as follows:

- Proposing the use of the divide-and-compute framework for execution on actual quantum devices.
- Proposal for a scheduling algorithm for distributed parallel quantum multiprocessing.
- Demonstrating the use of optimal qubit routing and evaluating its performance for implementation on actual quantum devices.
- Runtime evaluation for the parallel and distributed execution of subcircuits on quantum devices.

The organization of the paper is as follows. A few preliminaries of NISQ-era quantum computation are presented in Section 2. Section 3 illustrates the overview of the proposed compiler. In Section 4, experimental results are exhibited with brief discussion. Section 5 depicts our concluding remarks with future scope.

2 Background

The circuit-based quantum computing model is one in which operations are performed by a series of quantum gates that are always reversible. The quantum gates act on the prepared initial qubit states sequentially, transforming the initial states through unitary transformation into the final state as dictated by the quantum operations. The final state is then measured to complete the computation procedure. This reversibility property makes it unique from classical computing. All quantum gates are by-design unitary to ensure reversibility.
Currently, the world has functional but small scale and imperfect quantum computers that are called Noisy intermediate scale Quantum Computers (NISQ). Large-scale quantum computation is impossible in NISQ-era quantum computers due to the increase of errors as the number of qubits is scaled up. The error propagates throughout the circuits as the number of qubits increases till it reaches a point when measurement on quantum computing becomes hugely unreliable to the point of being untenable. The current research trend is to improve the existing faulty NISQ devices so that they become viable for usage in the passing time. Till the era of fault-tolerant computing starts, NISQ devices are the only machines to test and build quantum algorithms.

Universal fault-tolerant quantum computing (FTQC) [3] [14] [15] still requires years of research till they are realizable in real life. FTQCs are large-scale quantum computers with a minimal error rate and significant coherent time, ensuring a significant quantum advantage over classical computing when they arrive. Till then, NISQ devices are currently the best working model of quantum computing we have, and all quantum algorithms are designed keeping the limitations of NISQ in mind. NISQ-era computers are not changing the world right away, but it is a promise of the quantum advantage that is coming. These quantum computers have already some real-life applications that give somewhat exploratory results for some problems [16] [17] [18] and a few quantum supremacy [19] [20] [21] examples have also been shown in recent years.

NISQ devices are made of qubits that are erroneous, unstable, and have prepared states that decay over short periods [8] [22]. Even the gate operations are faulty and prone to deviations from actual results. Qubit stability error is called decoherence whereas gate operation error is known as fidelity. Due to these errors, the quantum computing hardware cannot scale. Hence, the term intermediate scale came into being. Each of these NISQ-era devices have particular topologies or connectivities that acts as a characterization of these devices.

The qubit topology graph of the device represents the qubit layout of the hardware. It shows how the physical qubits are connected on the real device. The qubit topology for some devices are shown in Figure 1 [23]. This graph is important when a compiler tries to map a circuit to the quantum device, because it shows how qubits are connected and this is something that the compiler has to consider when transforming circuits to run in devices. For example, if two qubits are not physically connected, a CNOT gate cannot be executed between the two qubits. It is not possible to run any circuit on a real device without rewriting the circuit to match the device topology of the intended device to run on. This is part of a process called transpilation. Hence, topology graphs play an important role when designing circuits for algorithms in the NISQ-era.

The qubits on IBM hardware are fixed. IBM uses superconducting transmon qubits rather than trapped-ion qubits. They don’t move around as trapped-ion qubit hardware. Therefore, IBM Q quantum devices have limited connectivity among the qubits of the device. This property gives rise to the optimal qubit mapping problem as elaborated in the subsequent sections.

Due to noise, the depth and width of quantum circuits are limited. The low depth and width of quantum circuits mean that the capabilities of today’s NISQ machines are dependent on the realization of hybrid algorithms. Hybrid algorithms contain both quantum and classical computing parts. At present all quantum algorithms follow this generic hybrid structure. The quantum part is like any quantum algorithm: the initial state is prepared, followed by unitary quantum gate operations, followed by measurements of the final states. The final state is measured by varying the input parameters of the quantum gate operations through classical post-processing till the results improve to the desired level.

Larger devices have significantly more noise than smaller ones. Hence, cutting a large circuit into smaller subcircuits seems to be a way to realize better fidelity. Many seemingly intractable problems for current NISQ-era devices are made possible by dividing the humongous problem-specific circuit into manageable ones that the current architectures can run. This improves [24] on the design of distributed quantum systems.

In a distributed quantum computing, multiple smaller non-local quantum devices are connected coherently, with all the nodes together working as a single large ecosystem of quantum computation. The number of qubits scales linearly with the number of interconnected devices. Hence, distributed Quantum Computing can act as an intermediate solution to the scaling-up problem.

3 Compiler Overview

In the previous sections, we have observed that the NISQ-era devices are ones with low qubit count and high inherent device noise. The depth of a circuit is the number of time-steps required, assuming that gates acting on distinct bits can operate simultaneously (that is, the depth is the maximum length of a directed path from the input to the output of the circuit) [25]. For NISQ-era devices, as the depth of the quantum circuit increases, the cumulative noise in the circuit also increases. The primary objective of our proposed compiler architecture is to reduce the depth of the quantum circuit by employing a recursive circuit cutting algorithm. This technique drastically reduces the size of the circuit to be executed. This, however, comes at a cost: The circuit reconstruction done as of now employs random shot sampling, thereby inducing finite sampling noise in the circuits. Qurzon tries to mitigate this noise by employing an optimal cut level algorithm, which finds the optimal level of recursive circuit cuts where the finite sampling noise is minimized, but maximizing the number of fragmented circuits, for ease of execution.

A diagrammatic overview of Qurzon’s components can be found in Figure 2, which shall be briefly introduced here and elaborated on later in the subsequent subsections.

1. Circuit Cutting Algorithm: One of the primary components of the Qurzon is the circuit cutting algorithm. This takes a large multiqubit (>2) circuit and uses the CutQC [26] algorithm to find the optimal cut. The optimal cut finding is done using the Gurobi Solver with a mixed-integer programming formulation of the circuit’s directed acyclic graph (DAG). This circuit cutting is employed recursively in order to break the circuit into multiple parts. Once done, it is passed on to the next part of the compiler.

2. Scheduling Algorithm: This part of the algorithm takes the cut circuits from the previous step and ar-
Figure 1: The qubit topology and connectivities of (a) IBM Kolkata: A 27-qubit superconducting qubit quantum computer (b) Google Sycamore: A 53-qubit superconducting qubit quantum computer (c) An 11-qubit IonQ trapped-ion quantum computer. Note that the IonQ device has full connectivity of all the qubits in its system, but some of these connectivities have high noise levels.

Figure 2: A high level overview of the Qurzon compiler, showing the different components and workflow

ranges them in a queue. It then takes into account the available quantum devices in the device pool. Once done, it allocates the subcircuits as jobs to the available quantum devices based on a greedy algorithm and passes it down to the next stage of the compilation process.

3. **Optimal Qubit Routing:** Once the scheduling algorithm zeros in on a target quantum device to run the subcircuit, the optimal qubit routing algorithm takes into account the current device qubit topology. Once done, it optimally places the quantum gates associated with the logical qubits of the subcircuit onto the physical qubits of the device, minimizing the number of SWAP gates required in the process.

4. **Parallel Execution on Distributed Quantum Devices:** The subcircuits placed optimally on the quantum devices are then executed in parallel from the quantum device pool. As of now the device pool is considered to be a homogenous collection of superconducting qubit devices.

5. **Subsequent Reconstruction:** The entire circuit is consequentially reconstructed using a shot-based probabilistic sampling of conditional probabilities of the subcircuit executions. This gives us the final output of the final quantum circuit.

3.1 **Circuit Cutter**

The theory of circuit cutting has been first proposed in the paper by Peng et.al. [27]. This section will be containing the mathematical framework for the circuit cutting algorithm, as proposed by Ayral et. al. [28] and thereafter, will be elaborating on the CutQC Algorithm [26] that Qurzon uses.

**Circuit Bipartitioning**

The calculation is a summary of the calculation presented in [28]. For further elaboration on this paper, the authors suggest referring to the original paper.

An $m$ qubit quantum circuit can be expressed as a composition of superoperators as follows:

$$
\mathcal{C} = C_A^{after} \circ C_B^{after} \circ C_A^{before} \circ C_B^{before}
$$

Here, the support of the super operators $C_A^{after}$ and $C_B^{after}$ is a bipartition of the quantum circuit. Similarly, the support of the superoperators $C_A^{before}$ and $C_B^{before}$ also forms a bipartition of
The final state of the circuit is given by the density matrix:

\[ \rho = C(\rho) = C_A^{\text{after}} \circ C_B^{\text{after}} \circ C_A^{\text{before}} \circ C_B^{\text{before}}(\rho_0) \]

where \( \rho_0 \) is the density matrix of the initial state. The probability of measuring a state \( i \) with binary representation \( i = (\hat{b}_0, \hat{b}_1, \ldots, \hat{b}_{m-1}) \) is given by:

\[ p(i) = Tr[\Pi_i \cdot \rho] \]

where \( \Pi_i \) is the projector on the state \( i \) \((i = 0, \ldots, 2^m)\).

Therefore, the final expression for the probabilities is given by:

\[ p(\hat{b}_0 \ldots \hat{b}_{m-1}) = \frac{1}{2} \sum_{a=X,Y,Z} \sum_{bbr \in \{0,1\}} \tilde{\gamma}^{bbr}_a p^a_B \]

\[ (\hat{b}_0 \ldots \hat{b}_{n-1}; b') \times p^a_B(\hat{b}_n \ldots \hat{b}_{m-1}) \]

where, \( p^a_B(\hat{b}_0 \ldots \hat{b}_{n-1}; b') \equiv 2^{n+1} \langle \langle \Pi_{\hat{b}_n \ldots \hat{b}_{m-1}}(\sigma^{b'}_a | b) R_b^a(\rho_0) \rangle \rangle_{q_0 \ldots q_n} \)

and,

\[ p^a_B(\hat{b}_n \ldots \hat{b}_{m-1}) \equiv 2^{m-n} \langle \langle \Pi_{\hat{b}_n \ldots \hat{b}_{m-1}}(R_B^a | b) \rangle \rangle_{q_{n+1} \ldots q_{m-1}}. \]

The circuit cutting algorithm automatically locates optimal positions to cut a large quantum circuit into smaller subcircuits. The cutting algorithm is processed classically. This technique makes it possible to simulate quantum algorithms for large problems, which would be otherwise impossible considering the current device restrictions with scalability. There are many advantages to circuit cutting. The solution state remains the same even as the number of cuts increases. It is also relevant to tensor network contraction methods.

### 3.2 Scheduling Algorithm

For this work, the scheduling algorithm takes into account the list of available devices the subcircuits can be executed upon. The scheduling algorithm allocates subcircuits to different devices (depending on their size and availability) while minimizing the loss of computational power and loss of time by making sure that most of the devices are used. The only function of the scheduler algorithm is to find the optimal distribution of the subcircuits over the list of devices.

The salient features of the algorithm proposed herein consist of the following:

- The subcircuits that are maintained in a priority queue, with a user-defined priority assigned to them before enqueueing.
- By default the subcircuits are to have a priority zero.
- The priorities assigned to each subcircuits add to Qurzon’s distributed nature as given in Section 3.4.
- Circuits with higher priority get executed earlier than the circuits with lower priority.
- The initial allocation of the circuits to the respective quantum devices are to be done based on a priority queue.
- The estimated time in the execution queue, post initial allocation is calculated and corresponding wait times are to be calculated.
- These wait times and their corresponding devices are arranged in increasing order of wait times.
- Any circuit that hasn’t been allocated to a quantum device post initial allocation are to be allocate thereafter, using a round-robin algorithm, starting from the device with the least wait time.

A brief outline of the algorithm is given as follows:
Algorithm 1 Proposed Scheduling Algorithm

Require: List of subcircuits $C_i(p_i)$ to be executed, $C_list$, where $p_i$ is the user-defined priority assigned to each circuit,
List of available devices, $D_list$

$A_list = []$

while $A_list \neq D_list$ do
    Sort($C_list$) in decreasing order of $p_i$
    Assign $C_i$ to quantum device $D_i$
    Append $D_i$ to $A_list$
    Remove $C_i$ from $C_list$
    if $C_list$ is not empty then
        Get estimated times $\tau_i$ for all $D_i$
        Append $D_i(\tau_i)$ to list WD
        Sort WD in ascending order
        Assign remaining circuits to each device in WD using a round-robin algorithm
    end if
end while

3.3 Optimal Qubit Routing

As introduced in section 2, NISQ-era devices are inherently susceptible to noise. NISQ devices also have limited qubit connectivity; this is particularly prevalent not only in superconducting qubit architectures but also in trapped-ion systems, as shown in Figure 1. Although the qubits in the trapped-ion systems appear to be fully connected, something to keep in mind is that in such architectures, the connectivity between the qubits is also prone to noise. This translates to the fact that not all connections have the same fidelity. This limited connectivity in devices gives birth to the optimal qubit routing problem.

The qubit routing problem is fundamental for circuit execution in NISQ-era devices. The motivation for the problem is to map a logical qubit, a qubit allocated in the circuit or the algorithm must translate directly to a physical qubit, a qubit on the actual quantum device. In doing so, the quantum circuit must go through transpilation or a source-to-source translator. What this stage does is, taking a quantum circuit into account, the transpiler decomposes the same into an equivalent circuit with a set of gates that can be run on the quantum device. Something to note is that the set of gates or the gate set is different for every quantum device.

Once the transpiler does its job, a set of gates is left behind to be executed on the device. These include single-qubit as well as two-qubit gates. The single-qubit gates aren’t much of a concern since they can be mapped to the individual physical qubits with ease. However, to execute the two-qubit gates, the qubit connectivity of the devices are accounted for. This means that any two logical qubits that are adjacent may not be physically adjacent to each other.

A viable solution to the problem of having non-adjacent interacting qubits is to insert SWAP gates to exchange it with a neighbouring qubit, thereby moving it closer to the desired qubit position. As this method introduces new gates into the quantum circuit, the depth of the circuit increases thereby compromising
the algorithm’s performance. This problem is called the qubit routing problem, wherein two desired qubits are made to interact with one another by minimizing the number of SWAP gates necessary to do so. This problem is shown to be equivalent to the token-swapping problem and is proven to be at least NP-hard and possibly PSPACE-complete [29].

However, efforts have been made to design an exact solution to the qubit routing problem. Siraichi et al. [29] proposed a dynamic programming method, resulting in a complexity that is exponential to the number of qubits, and a heuristic solution that approximates solutions to small circuits reasonably well. Zulehner et al. [30] proposed an algorithm that takes into account partitions based on the depth of the circuit and using the A* algorithm to search for optimal solutions, specializing in IBM devices. There is considerable literature towards finding an optimal solution to this problem [31] [32] [33], but here the approach proposed by t(ket) has been incorporated in Qurzon as outlined below:

Optimal Layout Synthesis with t(ket)

\( t(ket) \) [9] [34] [35] is a language-agnostic optimising compiler package that manages executable circuits and optimizes for physical qubit layout while also reducing the number of required operations. The system consists of two main components: a powerful optimizing compiler written in C++, and a lightweight user interface and runtime system written in Python. The Python Interface is used to interact with a wide range of hardware, handle the full cycle of compilation, dispatch and results retrieval. The compilation steps can be controlled via the interface to suit the nature of the quantum circuit being run. \( t(ket) \) deals with the qubit routing problem by using a heuristic method that produces optimal mapping systems in terms of depth, total gate count, and low run times. Circuit optimisation is especially pertinent on so-called noisy intermediate-scale quantum (NISQ) devices. The longer the computation runs, the more noise builds up. It is specifically designed for NISQ devices, and includes features that minimise the influence of device errors on computation. \( t(ket) \) routing algorithm ensures the compilation of any quantum circuit to any existing device as long as its architecture can be represented as a simple connected graph. The algorithm works in four parts: decomposing the input circuit into timesteps; initial mapping; routing across timesteps; and swap synthesis and final rewriting based on device architecture constraints.

3.4 Parallel Execution on Distributed Quantum Systems

As outlined in the scheduling algorithm, Qurzon is to be equipped with the power of distributed quantum computing, with the subcircuits being executed in parallel. The quantum devices used herein are assumed to be superconducting qubit devices, taking into account their respective decoherence noise, device noise, readout error rates, etc. With the ability to assign a user-defined priority to each subcircuit, Qurzon offers some degree of control over which circuits are assigned to which quantum device. In future iterations of this research, the authors of this paper aim to provide a robust abstraction to assign each subcircuit manually to each quantum device. Furthermore, the authors of this paper aim to extend the cross-platform functionality of this compiler to trapped-ion and photonic systems, thereby allowing researchers to execute different parts of a quantum algorithm on different device architectures. The authors predict that the significant challenges faced herein are calibrating the measurement readouts to construct a full algorithm. The authors hope to address this problem through subsequent research in the field.

3.5 Stitching Quantum Circuits Post Execution

Circuit reconstruction/stitching is a classical post processing method that can reconstruct the output of the original circuit. It reconstructs the probability distribution over measurement outcomes for the original quantum circuit from probability distributions associated with the subcircuits. The circuit reconstruction process has a relatively low computational cost that makes the whole algorithm a practical alternative strategy for large quantum simulations.

However that being said, the reconstruction algorithm used herein uses CutQC’s [26] Dynamic Definition Query which still uses exponential time and resource complexity. This leaves room for improvement in subsequent iterations of this research wherein, attempts shall be made to reduce it to sub-exponential costs.

4 Experimental Results

4.1 Experimental Setup

The workflow for the "control system" i.e., running the circuits without using optimal layout synthesis is as follows: The circuits were generated using the code for the helper function in Wei Tang’s CutQC paper [26]. Mock backends were taken from the IBM Qiskit Aer simulator library [36]. The backends used were Fake Tokyo, Fake Tenerife, Fake Melbourne, Fake Vigo, Fake Rueschlikon and Fake Poughkeepsie. Noise models for the respective backends were used and subjected to the scheduling algorithm. The circuit cutter used Gurobi’s mixed integer programming solver and used 8192 shots for the reconstruction phase. Once the cut circuits have been run on the respective backends, they are reconstructed and their fidelities are noted. For the "experiment workflow" all the steps in the previous paragraph were done except when using the backends. Herein, the \( t(ket) \) backends were used from \( t(ket) \) IBM extension package. Here, \( t(ket) \) implements optimal layout synthesis while running the cut circuits and uses CutQC’s reconstruction methods and generates the respective fidelities.

4.2 Case Study: The Bernstein-Vazirani Algorithm

The Bernstein Vazirani algorithm was first introduced in [37]. This algorithm is an extension to the Deutsch Josza algorithm and demonstrated the use of quantum computers to solve more problems more complex in nature than that of the Deutsch Josza. Here, we are given a oracle that outputs a 0 or a 1 i.e. \( f(x_0, x_1, x_2, . . . ) \rightarrow 0 \) or 1 where \( x_i \) is 0 or 1 along with a input string \( s \). That means the oracle looks like \( f(x) = s \cdot x \pmod{2} \). The oracle returns with certainty with just one run of the function. The Bernstein Vazirani algorithm is used to find the secret input string \( s \).

For an elementary application of the Bernstein Vazirani algorithm, the circuit is given by:
Demonstrated herein is the present state of the compiler and its inner workings traced as it passes through each step in the compiler.

**Circuit Cutting Phase**

Using Mixed Integer Programming model this circuit is optimally subdivided into several subcircuits which are given below. These subcircuits are then into scheduler algorithm which determines the architecture of the devices on which the subcircuits are to be run.

**Scheduling Phase**

During the scheduling phase of the compiler, the authors have used a naive scheduling algorithm which allows for the execution of the quantum circuit on one of the devices in the device pool. The naive scheduler used herein has scheduled the subcircuits to be executed on the IBM FakeTenerife. The qubit topology of the device is given as follows:

**Qubit Mapping and Optimal Layout Synthesis**

Using the qubit topology of IBM FakeTenerife as shown in figure 4, the optimal layout synthesis was carried out using $|\text{ket}\rangle$. $|\text{ket}\rangle$ takes into account the connectivities between the qubits and have accordingly placed the quantum gates such that the execution is optimal and the number of SWAP gates required is minimized.
Circuit Reconstruction Phase

The subcircuits thus formed and executed on quantum device are consequently reconstructed using Dynamic Definition Query as outlined in [26].

The measurement thus done used 1024 shots to reconstruct the probability landscape of the entire quantum circuit. This gives rise to finite sampling error which can be quantified in the results using the following parameters and their corresponding values as outlined in the table below.

| Metric           | Value          |
|------------------|----------------|
| Cross Entropy    | 2.402524222   |
| Mean Absolute Percentage Error | 8.88194E+14 |
| Mean Squared Error | 0.000815706 |
| χ² Error         | 1.668078342   |
| Heavy output probability | 0.09048925 |

Table 1: Table showing different errors for the 10 Qubit Bernstein-Vazirani Algorithm

4.3 Results and Discussions

The results published herein are based on a limited number of qubits, with limited depth, due to computational power constraints. However, keeping in mind, the prototypical nature of this project, the results are sufficient to draw meaningful inference, paving the way for future work in the field.

On benchmarking several circuits, namely the Bernstein Vazirani algorithm, Grover’s algorithm, the AFQT, the QFT, the supremacy circuit, the linear supremacy circuit and random circuits, through the pipeline proposed in this paper, a general trend can be seen. The results of these benchmarks can be observed in Figure 3 and 5. The y-axis consists of the difference in the fidelities δF = (E − E′), where E is the mean squared error of the circuit without subjecting it to a qubit mapping algorithm and E′ is the mean squared error of the circuit after t(ket) was applied. The x-axis consists of the number of qubits in the input circuits.

When the size of the input circuits is small, the δF tends to stay negative. This is because, when circuit size is small and the size of the cut circuits are small, the number of cuts in the circuits are minimal. The cut circuits are then optimally routed using the t(ket) module. On the control side of these experiments, the optimal layout synthesis module was not used. It was noted that at these levels, optimal qubit mapping plays a significant role in mitigating the finite sampling noise, found as a result of circuit reconstruction. At large qubit counts, with cut size remaining constant, the reconstruction noise is great enough to nullify the effects of the qubit mapping.

This is an important observation because although due to computational constraints large circuits could not be simulated, a hypothesis can be reached, and can be explored further in future iterations of this research. The hypothesis is that when large circuits can be made to run through the compiler, with large enough cut circuit size (say, equal to the size of device register), qubit routing is to play a pivotal role in mitigating circuit reconstruction noise, thereby significantly improving the fidelity of the circuits formed after reconstruction.

Furthermore, the depth of the quantum circuit plays minimal impact on the fidelity, with or without the qubit routing scheme. This is due to the depth-agnostic nature of the t(ket) algorithm. However depth-aware mapping techniques maybe used in future iterations of this research in order to study the depths therein. Furthermore, it remains to be seen if the depth can be reduced effectively by using techniques similar to CutQC [26]. Such to-
mography inspired techniques have been implemented by Perlin et al. [38] but remains to be studied for general circuits.

Running time of the circuits

This section attempts to provide a formalism regarding the running times of the circuits on Qurzon, as it stands, in the latest iteration. This formalism is heavily dependent on the CutQC algorithm and the availability of quantum devices along with classical resources.

For this analysis, n quantum subcircuits, post circuit cutting, are to be considered to be allocated to run on k quantum devices. Instead of queuing the quantum circuits, the authors herein propose running the circuits parallelly on the quantum devices, subject to device availability. Let $S_k$ denote the $k^{th}$ quantum processor, that is available to be used and $t(S_k)$ denote the time taken to run the circuit on the $k^{th}$ quantum processor. Without loss of generality, it is assumed that $n > k$. The time taken for classical post processing is considered as $C(S_i)$ for the $i^{th}$ subcircuit.

The time, therefore, taken for the classical post processing of the $n$ subcircuits is given by $T_c = \sum_{i=1}^{n} C(S_i)$.

The time taken for the circuits to be executed on the quantum processor be given by $T_q = \max(t(S_1), t(S_2), \ldots, t(S_n))$. This is due to Qurzon’s proposed scheduling algorithm that shall allow for the parallel execution of these subcircuits.

Since $n > k$, it might be that some subcircuit might require some wait time before some quantum device is allocated. This wait time is characterised by $W(S_i)$ for the $i^{th}$ subcircuit in waiting. Note that $W(S_i)$ can be zero, for the circuits that can be associated with a quantum processor. Hence, the total wait time is given by $T_w = \sum_{i=1}^{n} W(S_i)$.

Therefore, the total time taken for the end to end execution of a quantum circuit, that has been divided into $n$ subcircuits associated with $k$ quantum devices is given by:

$$T = T_q + T_c + T_w$$

$$T = \max(t(S_1), t(S_2), \ldots, t(S_n)) + \sum_{i=1}^{n} C(S_i) + \sum_{i=1}^{n} W(S_i)$$

Circuit cutting to reduce multiqubit gates

Every quantum circuit consists of unitary operators that operate over either one qubit or multiple qubits. In most quantum devices, multi-qubit gates are decomposed into several two-qubit gates in order to be accommodated in the gate-set for a quantum device. The gate-set for a quantum device is defined to be the set of operations that a quantum processor is to physically run on itself, in order to output the results of a circuit. This process is called transpilation and is handled by a piece of software called the transpiler. At the hardware level, therefore, the fidelity of the circuit is dependent on the types of gates that the circuit uses. Specifically, at the juxtaposition of logic synthesis and execution, the two-qubit gates are responsible [39] [40] [41] [42] [43] for the fidelity of the quantum circuits. The two-qubit gates are also the motivation behind the optimal qubit mapping problem for NISQ-era devices.

The beauty of Qurzon is that due to the circuit cutting techniques, the number of two-qubit gates in the circuit is highly diminished when the size of the qubit register of the quantum device is small. The tradeoff, however, lies in the fact that whenever the size of the qubit register for the cut circuits is low, the time taken for classical post-processing, exponentially increases. This has been shown by Tang et. al. in the CutQC paper [26]. Not only that, the reconstruction noise after execution for several of these minuscule circuits exceeds the noise induced by these two-qubit gates. The authors of this paper, therefore, feel, that it is imperative for the reconstruction methodology for these subcircuits to be improved upon such that the probability landscape of large circuits can be calculated effectively, swiftly and reliably. This point also remains to be worked on by the authors, in the successive iterations of this project.

5 Conclusion and Future Scope

This piece of work introduces Qurzon, a novel approach to compiling and executing circuits on NISQ-era devices. It also studies the effects of qubit routing on the quantum divide and compute approach. It is noted that for sufficiently large circuits, qubit routing plays a major role in determining the circuit’s fidelity after being subjected to the cut and execute methodology. Furthermore, this work proposes the use of parallel scheduling algorithms for quantum multi-computing. The work proposed herein is to be extended to include devices from multiple vendors such as Google, IBM, Rigetti, Xanadu, IonQ i.e. on photonic and trapped ion hardware. The authors of this paper are currently investigating methods to cut down circuit depth that is to be integrated in future iterations of this work. Additionally this work paves the way for future research into areas wherein different parts of a quantum circuit maybe executed on different hardware and calibrated accordingly. Another major scope for future research in this field is to reduce the classical post-processing to a sub-exponential computational cost.

References

[1] Michael A. Nielsen and Isaac L. Chuang. Quantum Computation and Quantum Information: 10th Anniversary Edition. Cambridge University Press, 2010. doi: 10.1017/CBO9780511976667.

[2] John Preskill. Quantum Computing in the NISQ Era and Beyond. Quantum, 2:79, 2018.

[3] Andrew Steane. Quantum computing. Reports on Progress in Physics, 61(2):117, 1998.

[4] Colin P Williams. Explorations in quantum computing. Springer Science & Business Media, 2010.

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

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

[7] Yudong Cao, Jonathan Romero, Jonathan P Olson, Matthias Degroote, Peter D Johnson, Mária Kieferová, Ian D Kivlichan, Tim Menke, Borja Peropadre, Nicolas PD
Sawaya, et al. Quantum chemistry in the age of quantum computing. Chemical reviews, 119(19):10856–10915, 2019.

[8] Samudra Dasgupta and Travis S Humble. Characterizing the stability of nisq devices. In 2020 IEEE International Conference on Quantum Computing and Engineering (QCE), pages 419–429. IEEE, 2020.

[9] Alexander Cowtan, Silas Dilkes, Ross Duncan, Alexandre Krajenbrink, Will Simmons, and Seyon Sivarajah. On the qubit routing problem. arXiv preprint arXiv:1902.08091, 2019.

[10] Davide Ferrari, Angela Sara Cacciapuoti, Michele Amoretti, and Marcello Caleyfi. Compiler design for distributed quantum computing. IEEE Transactions on Quantum Engineering, 2(1–20), 2021. ISSN 2689-1808. doi: 10.1109/tqe.2021.3053921. URL http://dx.doi.org/10.1109/tqe.2021.3053921.

[11] D Ferrari, S Nasturzio, and M Amoretti. A software tool for mapping and executing distributed quantum computations on a network simulator.

[12] Daniele Cuomo, Marcello Caleyfi, and Angela Sara Cacciapuoti. Towards a distributed quantum computing ecosystem. IET Quantum Communication, 1(1):3–8, Jul 2020. ISSN 2632-8925. doi: 10.1049/iet-qtc.2020.0002. URL http://dx.doi.org/10.1049/iet-qtc.2020.0002.

[13] L Gyongyosi and S Imre. Distributed quantum computation for near-term quantum environments. In Quantum Information Science, Sensing, and Computation XIII, volume 11726, page 1172601. International Society for Optics and Photonics, 2021.

[14] John Preskill. Fault-tolerant quantum computation. In Introduction to quantum computation and information, pages 213–269. World Scientific, 1998.

[15] Peter W Shor. Fault-tolerant quantum computation. In Proceedings of 37th Conference on Foundations of Computer Science, pages 56–65. IEEE, 1996.

[16] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Journal on Computing, 26(5):1484–1509, Oct 1997. ISSN 1095-7111. doi: 10.1137/s0097539795293172. URL http://dx.doi.org/10.1137/S0097539795293172.

[17] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. Physical Review Letters, 103(15), Oct 2009. ISSN 1079-7114. doi: 10.1103/PhysRevLett.103.150502. URL http://dx.doi.org/10.1103/PhysRevLett.103.150502.

[18] Xiao Mi et al. Observation of time-crystalline eigenstate order on a quantum processor, 2021.

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

[20] Aram W Harrow and Ashley Montanaro. Quantum computational supremacy. Nature, 549(7671):203–209, 2017.

[21] Michael J Bremner, Ashley Montanaro, and Dan J Shepherd. Achieving quantum supremacy with sparse and noisy commuting quantum computations. Quantum, 1:8, 2017.

[22] Hasan Yetis and Mehmet Karakoces. Investigation of noise effects for different quantum computing architectures in ibm-q at nisq level. In 2021 25th International Conference on Information Technology (IT), pages 1–4, 2021. doi: 10.1109/IT51528.2021.9390130.

[23] IBM Quantum Experience. URL https://quantum-computing.ibm.com/.

[24] Zain H. Saleem, Teague Tomesh, Michael A. Perlin, Pranav Gokhale, and Martin Suchara. Quantum divide and conquer for combinatorial optimization and distributed computing, 2021.

[25] John Preskill. Ph219/cs219 quantum computation 2018-19, Jul 2015. URL http://theory.caltech.edu/~preskill/ph219/ph219_2018-19.

[26] Wei Tang, Teague Tomesh, Martin Suchara, Jeffrey Larson, and Margaret Martonosi. CutQC: Using small quantum computers for large quantum circuit evaluations. In Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, pages 473–486, 2021.

[27] Tianyi Peng, Aram W Harrow, Maris Ozols, and Xiaodi Wu. Simulating large quantum circuits on a small quantum computer. Physical Review Letters, 125(15):150504, 2020.

[28] Thomas Ayal, François-Marie Le Régent, Zain Saleem, Yuri Alexeev, and Martin Suchara. Quantum divide and compute: Exploring the effect of different noise sources. SN Computer Science, 2(3), Mar 2021. ISSN 2661-8907. doi: 10.1007/s42979-021-00508-9. URL http://dx.doi.org/10.1007/s42979-021-00508-9.

[29] Marcos Yukio Siraichi, Vinicius Fernandes dos Santos, Sylvain Collange, and Fernando Magno Quintão Pereira. Qubit allocation. In Proceedings of the 2018 International Symposium on Code Generation and Optimization, pages 113–125, 2018.

[30] Alwin Zulehner, Alexandru Paler, and Robert Wille. An efficient methodology for mapping quantum circuits to the ibm q architectures. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 38(7):1226–1236, 2018.

[31] Robert Wille, Lukas Burgholzer, and Alwin Zulehner. Mapping quantum circuits to ibm qx architectures using the minimal number of swap and h operations. In 2019 56th ACM/IEEE Design Automation Conference (DAC), pages 1–6. IEEE, 2019.

[32] Beatrice Nash, Vlad Gheorghiu, and Michele Mosca. Quantum circuit optimizations for nisq architectures. Quantum Science and Technology, 5(2):025010, 2020.

[33] Bochen Tan and Jason Cong. Optimal layout synthesis for quantum computing. In 2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD), pages 1–9. IEEE, 2020.

[34] Alexander Cowtan, Will Simmons, and Ross Duncan. A generic compilation strategy for the unitary coupled cluster ansatz, 2020.
[35] Alexander Cowtan, Silas Dilkes, Ross Duncan, Will Simmons, and Seyon Sivarajah. Phase gadget synthesis for shallow circuits. *Electronic Proceedings in Theoretical Computer Science*, 318:213–228, May 2020. ISSN 2075-2180. doi: 10.4204/eptcs.318.13. URL http://dx.doi.org/10.4204/EPTCS.318.13.

[36] MD SAJID ANIS et al. Qiskit: An open-source framework for quantum computing, 2021.

[37] Ethan Bernstein and Umesh Vazirani. Quantum complexity theory. *SIAM Journal on Computing*, 26(5):1411–1473, 1997. doi: 10.1137/S0097539796300921. URL https://doi.org/10.1137/S0097539796300921.

[38] Michael A. Perlin, Zain H. Saleem, Martin Suchara, and James C. Osborn. Quantum circuit cutting with maximum likelihood tomography, 2021.

[39] Yu Chen, C. Neill, P. Roushan, N. Leung, M. Fang, R. Barends, J. Kelly, B. Campbell, Z. Chen, B. Chiaro, A. Dunsworth, E. Jeffrey, A. Megrant, J. Y. Mutus, P. J. J. O’Malley, C. M. Quintana, D. Sank, A. Vainsencher, J. Wenner, T. C. White, Michael R. Geller, A. N. Cleland, and John M. Martinis. Qubit architecture with high coherence and fast tunable coupling. *Phys. Rev. Lett.*, 113:220502, Nov 2014. doi: 10.1103/PhysRevLett.113.220502. URL https://link.aps.org/doi/10.1103/PhysRevLett.113.220502.

[40] Fei Yan, Philip Krantz, Youngkyu Sung, Morten Kjaergaard, Daniel L. Campbell, Terry P. Orlando, Simon Gustavsson, and William D. Oliver. Tunable coupling scheme for implementing high-fidelity two-qubit gates. *Phys. Rev. Applied*, 10:054062, Nov 2018. doi: 10.1103/PhysRevApplied.10.054062. URL https://link.aps.org/doi/10.1103/PhysRevApplied.10.054062.

[41] Pranav Mundada, Gengyan Zhang, Thomas Hazard, and Andrew Houck. Suppression of qubit crosstalk in a tunable coupling superconducting circuit. *Phys. Rev. Applied*, 12:054023, Nov 2019. doi: 10.1103/PhysRevApplied.12.054023. URL https://link.aps.org/doi/10.1103/PhysRevApplied.12.054023.

[42] X. Li, T. Cai, H. Yan, Z. Wang, X. Pan, Y. Ma, W. Cai, J. Han, Z. Hua, X. Han, Y. Wu, H. Zhang, H. Wang, Yipu Song, Luming Duan, and Luyan Sun. Tunable coupler for realizing a controlled-phase gate with dynamically decoupled regime in a superconducting circuit. *Phys. Rev. Applied*, 14:024070, Aug 2020. doi: 10.1103/PhysRevApplied.14.024070. URL https://link.aps.org/doi/10.1103/PhysRevApplied.14.024070.

[43] X. Y. Han, T. Q. Cai, X. G. Li, Y. K. Wu, Y. W. Ma, Y. L. Ma, J. H. Wang, H. Y. Zhang, Y. P. Song, and L. M. Duan. Error analysis in suppression of unwanted qubit interactions for a parametric gate in a tunable superconducting circuit. *Phys. Rev. A*, 102:022619, Aug 2020. doi: 10.1103/PhysRevA.102.022619. URL https://link.aps.org/doi/10.1103/PhysRevA.102.022619.