SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Robotics, Cognition, Intelligence

Hardware-aware Optimal Quantum Circuit Cutting and Knitting

Thang Tran

SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Robotics, Cognition, Intelligence

Hardware-aware Optimal Quantum Circuit Cutting and Knitting

Hardware-gestütztes optimales Schneiden und Stricken von Quantenschaltungen

Author:	Thang Tran
Supervisor:	Prof. Dr.-Ing. Pramod Bhatotia
Advisor:	Emmanouil Giortamis, Francisco Romão
Submission Date:	15.03.2024

I confirm that this master's thesis is my own work and I have documented all sources and material used.

Munich, 15.03.2024 Thang Tran

Acknowledgments

I would like to take the opportunity to express my gratitude to Professor Pramod Bhatotia for offering me the opportunity to dive deep into the exciting research field of Quantum Computing.

Furthermore, this thesis would not be possible without my advisors: Emmanouil Giortamis together with Francisco Romão. I am greatly thankful for their time, guidance, constructive feedback and advice. Their support has a great impact on the quality of my work and motivates me greatly.

Abstract

Over the past decade, Quantum Computer (QCer)s have experienced a resurgence in interest, largely driven by their potential for commercial applications. By harnessing the mechanisms of quantum physics, QCers offer the potential prospect of tackling computational problems that are currently beyond the reach of Classical Computer (CCer)s.

Despite the theoretical promise, the practical realization of QCers faces great challenges. While QCers with relatively modest qubit counts have demonstrated promising capabilities, achieving the scale that is necessary for real-world applications remains a daunting engineering obstacle. Moreover, the presence of noise in quantum hardware poses a significant hurdle, rendering the current systems incapable of scalable and fault-tolerant operations. This is known as the Noisy Intermediate-Scale Quantum (NISQ) QCer era.

To address these challenges, researchers are exploring a wide variety of approaches to optimize quantum circuits and mitigate the effects of noise. One such approach is Gate Virtualization, which seeks to minimize noise-induced errors by partitioning circuits into smaller, more manageable segments. Additionally, techniques like wire cutting, which utilizes swap gates, *Move Circuit* or quantum teleportation, aim to reduce qubit dependencies and enhance circuit performance. However, these advancements come with trade-offs. For instance, the sampling overhead required for circuit cuts and partitions can quickly become intractable for larger circuits, highlighting the need for more efficient optimization strategies.

In response to these challenges, this thesis proposes a novel hardware-aware algorithm that partitions circuits across multiple Quantum Processing Unit (QPU)s. By leveraging Gate Virtualization and Virtual Move Circuit techniques, coupled with advanced optimization strategies, the algorithm aims to maximize circuit performance while minimizing computational overheads.

The approach utilizes a Satisfiability Modulo Theories (SMT) solver, specifically a Boolean Satisfiability Problem (SAT) solver, to identify the optimal cuts and partitions, thereby paving the way for the practical realization of quantum applications.

This work shows that it is possible to partition a circuit into smaller subcircuits without losing accuracy. Furthermore, by combining different cuts and cut techniques, the fidelity of the circuit could be improved by up to 16225%. The geometric mean of fidelity improvements from this work is 369%.

In summary, while the road to scalable and fault-tolerant quantum computing faces many challenges, ongoing research efforts hold the promise of unlocking the full potential of QCers in the near future.

Contents

1 Introduction

In the last decade, interest in QCers has made a strong comeback, especially with its commercial potential. By leveraging the mechanisms of quantum physics, QCers could potentially be able to solve problems that are currently intractable with state-of-the-art CCers. This is known as "Quantum Advantage" [1]. In theory, QC has demonstrated advantages in solving problems in different areas such as: machine learning [2], factorization [3], optimization [4] and others [5, 6].

According to Preskill et al. [7], a QPU with 50-100 qubits could already show the superior capabilities of QC compared to Classical Computing (CC). However, a Quantum Circuit for any real world Quantum Application would typically require millions of qubits [8]. A QPU would need to have preferably also as much number of qubits as the circuit requires to accommodate it. To have such a large QPU is a challenge that engineers are facing at the moment [9]. The IBM Condor, introduced in December 2023, is one of the largest QCer with 1121 qubits [9]. A QPU with hundreds thousands of qubits is expected to become reality soon [9], however, this is still far away from any useful purposes.

Another problem that current hardware for quantum faces is noise. The hardware for QCers is still in its infancy and is said to be in the NISQ era. This is because they're currently very noisy and not fault-tolerant enough to be scaled up for meaningful applications. As the circuit size gets bigger, the effect of noises will also grow and render the end signal to be unusable [7]. There exists quantum error correction procedures to mitigate the effect of noise. However, the correction procedures use lots of qubits and hence increase the overhead computation significantly [10].

Due to these hardware limitations, researchers have been focusing on different techniques to solve these issues. These approaches aim to optimize quantum circuits so that the effect of noise is as minimal as possible. Gate virtualization has been a notable method to minimize the effect of noise [11]. This technique works based on the fact that quantum operations are similar to matrix operations, and hence can be operated on independently. Using gate virtualization, a binary qubit gate can be virtualized by using a gate with single-qubit operation instead [11–13]. With this method, binary gates can be "cut" and the quantum circuit can be partitioned into multiple smaller circuits that can be run on a smaller QPU that requires fewer qubits. A hybrid approach can then be used to run the cut circuit on QCer, and then have a post processing step to merge the results together to obtain the result that would be produced by the original circuit [11, 12]. Another tactic to reduce the dependencies between qubits is using wire cuts [13–15]. There are different ways to do this: using a swap gate, quantum teleportation [16] or using a Move Circuit [15]. For each method, there are different advantages and disadvantages. For instance, the teleportation method would require two ancilla qubits, while the Virtual Move circuit only needs one.

Gate Virtualization sounds promising, however, they come with a huge trade-off: Gate Virtualization suffers greatly from overhead sampling, which requires lots of computational cost [11, 12]. Cutting a circuit using n gate cuts and m wire cuts would need an overhead sampling of $O(6^n * 8^m)$ [11]. For this reason, the number of virtualizations should preferably be kept as low as possible. For instance, a circuit with five wire cuts will incur an overhead sampling of $8^5 = 32768$ times and require 5 ancilla qubits. As the circuit size grows, the number of sampling overhead grows intractably.

To reduce noise and utilize qubits more efficiently, this thesis proposes the following solutions:

- Based on previous work of Brandhofer et al. [15], the algorithm is made to be hardware-aware and supports partitioning the circuit to different QPUs.
- Partition the circuit according to the solutions using Gate Virtualization and Virtual Move Circuit.

This thesis uses a SMT solver, a specific type of SAT solver, to find the optimal cuts and partitions. By breaking the quantum application's circuit into a directed graph, each vertex and edge could be formulated into constraints and objective functions to be fed into the solver. If an optimal solution is found, the solver will return the edges that should be cut, and the type of cut that should be used: either teleportation or virtualization. The original circuit will then be cut and partitioned according to the model's solutions.

2 Background

This chapter is intended to provide the theoretical foundations that are necessary for understanding the developed algorithm.

2.1 Quantum Computing (QC)

2.1.1 Foundational principles

Different from CCers, QCers work based on quantum physics. Two important concepts in quantum physics that make it different from classical physics are interference and entanglement. These concepts are essential for quantum computers [17]:

- **Quantum interference**: Similar to classical physics, quantum states could be seen as waves where different quantum states can be added or subtracted. The resulting quantum state would be another valid quantum state [17]. This is vital for QC since it allows information to be processed in parallel. With the speed-up in information processing, the efficiency of the QCer would grow exponentially.
- **Entanglement**: Entanglement is a state when multiple objects share the same state even when they're physically far apart [18]. Changing the state of one entangled object would have an immediate effect on its counterpart. This is the key element behind the impactfully computational capabilities of QCers [19].

2.1.2 Qubit

In CC, information is represented as bits. Each bit has exactly one value at any point in time, either a 1 or 0. Similarly, in QC information is represented as qubits. Different from classical bits, qubits could represent two values at the same time. This is known as superposition. A state of qubit is typically represented as

$$|\psi\rangle = \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} = a_0 |0\rangle + a_1 |1\rangle \tag{2.1}$$

where $a_0, a_1 \in \mathbb{C}$ and $|a_0|^2 + |a_1|^2 = 1$.

Analog to classical bits of 0 and 1, qubit has the state of $|0\rangle$ and $|1\rangle$, where

$$|0\rangle = \begin{pmatrix}1\\0\end{pmatrix} \quad \text{and} \quad |1\rangle = \begin{pmatrix}0\\1\end{pmatrix} \tag{2.2}$$

To simplify the visualization of a qubit state, a *Bloch sphere* is used. Figure 2.1 is an example of a *Bloch sphere* representing $|\psi\rangle = cos(\theta/2)\,|0\rangle + sin(\theta/2)e^{i\phi}\,|1\rangle$.

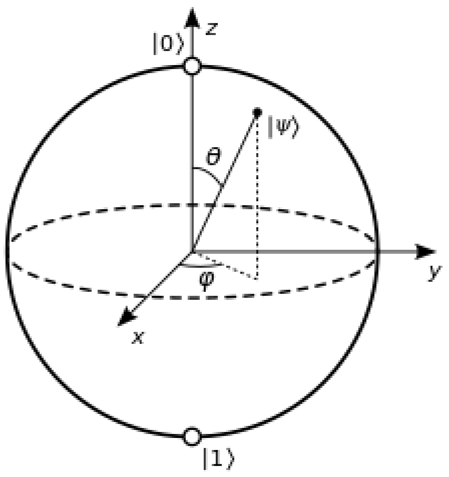


Figure 2.1: Bloch sphere represents a qubit state [20]

Conventionally, *Bloch sphere* is represented in a 3-dimensional coordinate system of X, Y and Z where $(x, y, z) = (sin(\theta)cos(\phi), sin(\theta)sin(\phi), cos(\theta))$

2.1.3 Unary gates

In CCers, logical gates are used to modify bits. Analogously in QCers, qubit gates are used. In QC, gates can be represented as any unitary matrices. To apply qubit gates to qubits, one can simply multiply the qubits state's vector with the gate matrix. The result would be the qubits' new states after applying the qubit gate. The most basic example of an unary qubit gate is the *NOT* gate (also known as *X* gate). Similar to its counterpart in CC, this gate would flip the value of its input (e.g. $|0\rangle \rightarrow X \rightarrow |1\rangle$):

$$X = \begin{pmatrix}0 & 1\\1 & 0\end{pmatrix} \tag{2.3}$$

Another well-known unary qubit gate is the *Hadamard gate* (or *H gate*).

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\1 & -1\end{pmatrix} \tag{2.4}$$

H gate creates an equal superposition state. i.e. $|0\rangle \rightarrow H \rightarrow |+\rangle$ and $|1\rangle \rightarrow H \rightarrow |-\rangle$, where

$$|+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \tag{2.5}$$

$$|-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}} \tag{2.6}$$

Both the *X gate* and *H gate* when being applied, will rotate the quantum state by a fixed angle. A more general description for this kind of "rotating" gate is the $R_Z(\lambda)$ gate that rotates around the Z axis by λ angle:

$$R_Z(\lambda) = \begin{pmatrix} e^{-i\frac{\lambda}{2}} & 0 \\ 0 & e^{i\frac{\lambda}{2}} \end{pmatrix} \tag{2.7}$$

2.1.4 Binary gates

For multiple qubits, their quantum states could be denoted as the tensor product combined from all single qubit states. A system with two qubits q_0 and q_1 could be denoted as $|q_0\rangle \otimes |q_1\rangle$ where $\otimes$ denotes the tensor product. For simplicity, this would usually be written as $|q_0 q_1\rangle$. Where:

$$|\psi\rangle = \begin{pmatrix} a_{00} \\ a_{01} \\ a_{10} \\ a_{11} \end{pmatrix} = a_{00}|00\rangle + a_{01}|01\rangle + a_{10}|10\rangle + a_{11}|11\rangle \tag{2.8}$$

given that $a_{00}, a_{01}, a_{10}, a_{11} \in \mathbb{C}$ and $|a_{00}|^2 + |a_{01}|^2 + |a_{10}|^2 + |a_{11}|^2 = 1$.

When both qubits $|q_0 q_1\rangle$ are measured in the Z-basis, the basis states that can be observed are:

$$|00\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad , \quad |01\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \quad , \quad |10\rangle = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \quad , \quad |11\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \tag{2.9}$$

Nowadays, on multi-qubit systems, the states of two qubits can be altered simultaneously with binary gates. Analog to unary qubit gates, binary gates could also be presented as unitary matrices. One of the most commonly used binary qubit gates is the Controlled NOT (CNOT) gate. The CNOT gate will flip the target qubit's state if the control qubit is in state 1. There are also qubit gates that act on three or more qubits. However, since each of these gates could be decomposed down to unary and binary gates, this work will be limited to those.

2.1.5 Quantum Processing Unit (QPU)

QPUs currently find themselves in the NISQ era. While showing potential superiority compared to its CCers counterpart, the QCers still suffer greatly from noise. The noise is caused by multiple different factors such as: errors from gates, errors during the measurement, etc. and this prevents researchers from creating error-free, fault-tolerant QPUs. This in turn makes it difficult for QPUs to grow bigger and have any meaningful applications.

As of the present, the most advanced QCer is Condor, developed by IBM, boasting 1121 qubits [9]. Although the QPU with hundreds of thousands of qubits is anticipated to materialize soon [9], it remains distant from practical applications.

The hardware of QCer is still under development and is known to be called NISQ. Since the hardware is noisy, they are not sufficiently fault-tolerant to be scaled up for any meaningful applications. Furthermore, the effects of noise also grow as the size of QCer's circuit increases [7]. It's possible to mitigate the noise effect using quantum error correction techniques. The downside of this approach is that it requires lots of qubits and will increase the overhead computation greatly [10].

Moreover, there is not a common consensus on how to build a QCers. Different vendors use different platforms to store quantum information for their QCers such as: superconducting circuits, quantum dots, color centers, trapped ions, topologically protected systems, etc. [21]. It is worth noticing that each of these platforms has their own problems that affect the characteristics of the QPU including but not limited to: the number of qubits in the circuit, qubit topology, gate error characteristics, measurement error characteristics, gate length, and numerous other parameters.

Different from CCers, one can not simply do an operation between two arbitrary qubits. For different QPUs, there exists a different qubit topology: a map of connections between qubits. If n qubits are directly connected to each other, a n-gate could be applied directly. Otherwise, if they are not directly connected, some extra operations such as swapping, qubit teleportation, etc. need to be used.

Furthermore, the measurement and gate errors are not homogenous among QPUs. These errors have a direct impact on the accuracy and reliability of the quantum circuit. Another parameter that also plays a role in the overall performance is the gate length, which describes the duration of a quantum operation. Moreover, the NISQ era QPUs are generally not stable and need to be recalibrated often. QPUs are dynamic systems that are easily affected by their environment and hence the noise characteristics would also change over time.

2.1.6 Transpilation and Execution

In CC, for a program to be executed, it needs to be compiled into machine code. Similarly, a quantum application circuit would need to be prepared for a specific QPU so that this circuit could be executed on that QPU. This preparation process is known as the *transpilation* process. In QC, the transpilation means to transform the quantum application circuit from its high level implementation into a form that is suitable for the characteristics and topology of that specific QPU.

The transpilation steps are required since each QPU has its own unique characteristics and topology for the qubits. A quantum application circuit is typically abstracted away from the QPU hardware and can not be executed directly. Without this transpilation process, each of the quantum applications would need to be rewritten for each of the QPUs that this circuit needs to be run on. These differences include but are not limited to qubit layouts, gate sets, connectivity constraints, etc. Transpilation process will read the original quantum application's circuit and adapt this circuit to match the constraints and properties of the target QPU. The *transpilation* process typically involves several steps:

1. **Gate Decomposition**: a QPU only supports a limited number of gates. However, it is not always the case that the quantum application's circuit only uses these supported gates. The *Gate Decomposition* step transforms the natively-not-available gates into the gates that this QPU supports.

2. **Qubit Remapping**: similar to gates, each target QPU has a different qubit topology. The connectivity of qubit in the circuit, also called logical qubit mapping, is not compatible with the actual qubit's connectivity on the QPU. This step transforms the logical connection to reflect the physical connection.

3. **Optimization**: To reduce the effects of noise, this step will find a way to reduce the number of gates required by the circuit if possible.

Transpilation is the key step that enables researchers to develop the quantum algorithms independent from the actual QPU. The researchers could then focus their work more on a higher level of abstraction.

After the *Transpilation* process is done, the circuit can now be fed to the target QPU for execution. To retrieve the result from a circuit, it is not enough to execute the transpiled circuit once. Due to the noisy hardware, quantum circuits need to be run repeatedly with thousands of executions (also known as shots) to mitigate the effect of noise. Using a probabilistic approach, the results of execution can be obtained.

Running a circuit on NISQ QPU's results in a noisy probability distribution. By running the circuit with CCer via simulation, a perfect noiseless result can be obtained. To measure how noisy the result is, *Hellinger fidelity* is used. This metric shows the quality of the result when executing the circuit on NISQ HW to a noiseless result from a simulated experiment. In general, the *fidelity* formula is:

$$F = \frac{\text{number of correct results}}{\text{number of shots}} \tag{2.10}$$

In Equation 2.10, the "number of correct results" is the total number of correct results. This correct result is typically obtained from the perfect simulation. The variable "number of shots" is the total number of executions.

2.2 Gate cuts and wire cuts with Quantum Circuit Knitting

One optimization technique that has been recently proposed is circuit *cutting and knitting* [13]. It is a combination of different divide-and-conquer methods. The first step is partitioning, which aims to group qubits and gates into smaller partitions that can be executed independently. After partitioning, the gate and wire dependencies still remain. The circuit knitting step is then used to resolve this violation [15]. Afterward, it is possible to cut a large circuit into smaller subcircuits. These cut circuits would be easier to execute since they are smaller and have fewer constraints, results in less noise on the same target QPUs. However, it is worth noticing that with the circuit knitting algorithms, the number of times these circuits need to be executed, i.e. the *sampling overhead*, will grow exponentially with the number of cuts, increasing runtime of the quantum circuit significantly. After all of the subcircuits are executed, the results will be merged using a classical algorithm. This merged result will be approximately equal to that of the original circuit as it was executed on a much larger and less noisy QPU.

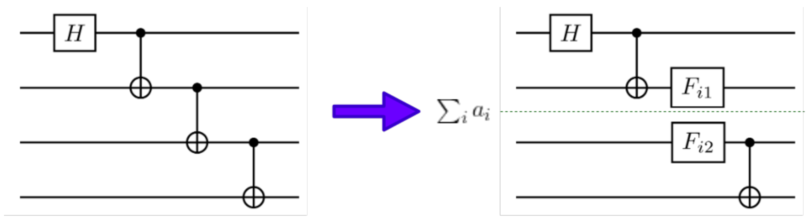


Figure 2.2: Cutting a 4-qubit GHZ circuit into two partitions. Gate dependency is resolved using quantum circuit knitting.

Figure 2.2 is a general example of quantum circuit knitting resolving a gate dependency. In quantum circuit knitting, a gate g is cut by transforming an unitary channel

$\mathcal{U}$ representing a quantum gate g by a Quasiprobability distribution (QPD) [11, 13, 22]:

$$\mathcal{U} = \sum_i a_i F_i \tag{2.11}$$

In Equation 2.11, $\mathcal{F}_i$ represents an operation that can be done locally together with potential classical communication. Each $\mathcal{F}_i$ is weighted by a parameter a_i that signifies how the computation of $\mathcal{U}$ should be approximated. Each time a partition that has gate g is executed, the gate g will be randomly replaced by the operation $\mathcal{F}_i$ with a probability that is tied to a_i. Each of these replacements is called a *subcircuit* and the resulting sampling overhead scales with K^2 for each gate cut. The K-factor is defined as $K := \sum_i |a_i|$. For each quantum gate, there could be multiple QPDs with different K-factor [13]. K should preferably be minimized. The minimum K that is achievable is known as the *γ-factor* [15].

Generally, *γ-factor* for quantum circuit knitting is directly correlated with the entangling degree of a quantum gate [15]. After all subcircuits are generated, the expected result of the original uncut gate is obtained by summing all of the subcircuits' results together with their corresponding a_i factor [13].

The sampling overhead of multiple, *simultaneous* cuts could be reduced by using Dependency resolution techniques. These techniques could be implemented using quantum gate teleportation protocol and will require ancilla qubits together with classical communication [13, 23, 24].

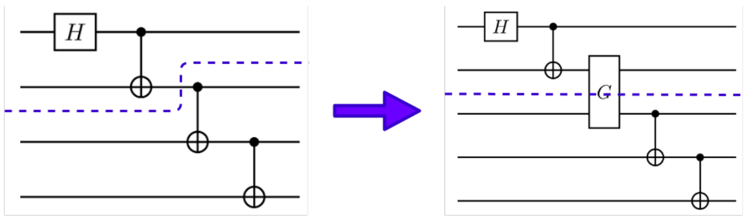


Figure 2.3: Cutting a 4-qubit GHZ circuit into two partitions. Wire dependency is resolved using gate cut.

A Wire Dependency can be resolved by transforming it to a Gate Dependency as shown in Figure 2.3. The newly created Gate Dependency can then be cut using the regular Gate Dependency resolving technique as mentioned above. At the wire cut location, a set of quantum operations is injected to transfer the quantum state of the left and cut position to an ancilla qubit. This set of operations is represented as subsystem G in Figure 2.3.

In the following steps, to resolve the newly added gate dependencies introduced by subsystem G, the gate cut technique can be applied. It is worth mentioning that for each wire cut, one ancilla qubit is required. Reusing qubits technique, proposed in 2023 could be used [25, 26]. This technique could be applied to wire cuts to reduce the overall requirements for ancilla qubits.

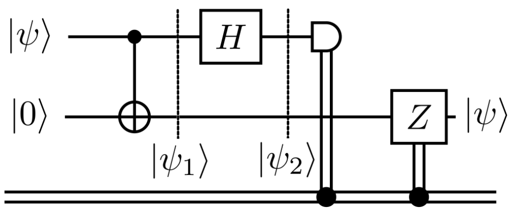


Figure 2.4: Move Circuit. Qubit state $|\psi\rangle$ is moved from the first qubit to the second qubit

There are multiple options to implement subsystem G: using a quantum teleportation circuit [16], using a swap gate or a *Move Circuit*. Each methods has its own advantages and disadvantages regarding sampling overhead, ancilla qubit requirements, classical communication and latency. The *Move Circuit* and teleportation method have the lowest incurred sampling overhead and both require classical communication. The *Move Circuit* however, requires one less ancilla qubit compared to the teleportation method [15].

Figure 2.4 shows an example of the *Move Circuit*. Following is the proof that the *Move Circuit* moves the qubit state $|\psi\rangle$ from the first qubit to the second qubit. The *Move Circuit* is made of a Bell measurement followed by a phase correction [27]. Given

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{2.12}$$

First, a *CNOT gate* is applied between two qubits resulting in

$$|\psi_1\rangle = \alpha\,|00\rangle + \beta\,|11\rangle \tag{2.13}$$

afterward, a *Hadamard gate* is applied to the first qubit, resulting in

$$|\psi_2\rangle = \alpha(|0\rangle + |1\rangle)\,|0\rangle + \beta(|0\rangle - |1\rangle)\,|1\rangle \tag{2.14}$$

Now the measurement of the first qubit has the same probability for "zero" and "one". The states of both qubits is dependent on this measurement.

If the result is "zero", the new state of the second qubit is:

$$\langle 0| \cdot |\psi_2\rangle = \alpha(0+1)\,|0\rangle + \beta(0+1)\,|1\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{2.15}$$

If the result is "one", the new state of the second qubit is:

$$\langle 1| \cdot |\psi_2\rangle = \alpha(0+1)\,|0\rangle + \beta(0-1)\,|1\rangle = \alpha\,|0\rangle - \beta\,|1\rangle \tag{2.16}$$

Depending on the measurement of the first qubit, a phase correction with a Z *gate* is applied on the second qubit. After the phase correction, the state $|\psi\rangle$ is obtained from the second qubit.

2.3 Gate Virtualization

Gate Virtualization is a technique that lies under *cutting and knitting*. Using *Gate Virtualization*, the qubit binary gates can be virtualized using only unary gates instead. A simple knitting formula can then be applied to get the result's probability distribution of the original circuit [11, 12]. The virtualization technique can be applied to any binary gates. In the following, a virtual Controlled Z (CZ) gate will be used to explain and demonstrate the *Gate Virtualization* technique as described by Mitarai at al [11].

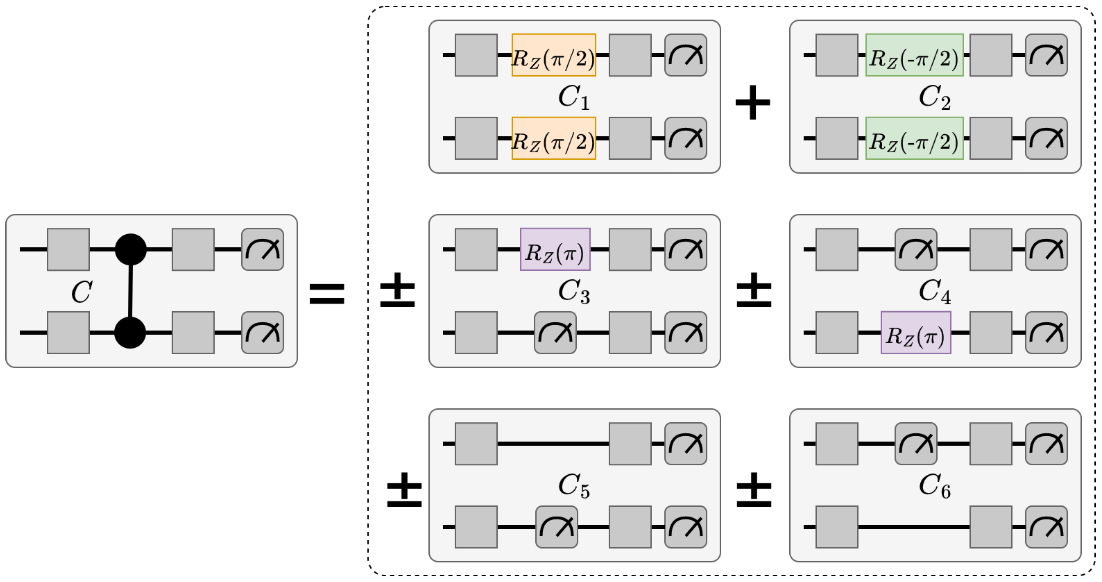


Figure 2.5: Schematic formula for CZ gate's virtualization [28]

Figure 2.5 is a schematic formula for the virtualization steps of a CZ gate. Using different unary gates and sampling them, together with the mid-circuit measurements from each subcircuit C_i, the original circuit probability distribution P can be calculated using the probability distribution of each subcircuit P_i.

2.3.1 Creating independent fragments via virtualization

In the virtualization technique, the virtualized gate is sampled with unary gates and measurements. For the case of the virtualized CZ gate, the sampling step will result in 6 different subcircuit configurations C_1 - C_6 as shown in Figure 2.5. Each of these subcircuits results in their own probability distributions P_1 - P_6. The probability distributions of the subcircuits are transferred to the knitting step. As demonstrated in Figure 2.5, in each of the subcircuits C_1 - C_6, the virtualized gate CZ is replaced with either a unitary gate, a mid-circuit measurement, or both. After the virtualization step, the qubit dependency from the CZ gate in circuit C_i is removed, creating 2 independent fragments. Each of these fragments can then be executed individually as C_i^1 and C_i^2, resulting in the probability distribution of each fragment: P_i^1 and P_i^2. To obtain the probability distribution P_i of configuration C_i, rules of stochastically independent events will be applied to P_i^1 and P_i^2. The final result is the multiplication of each result state $|x\rangle : P_i(|x\rangle) = P_i^1(|x\rangle) \cdot P_i^2(|x\rangle)$.

2.3.2 Knitting step

To obtain the original circuit's final probabilities $P(|x\rangle)$, the following formula is used:

$$\begin{aligned}
p_1 &= P_1(|x\rangle) \\
p_2 &= P_2(|x\rangle) \\
p_3 &= P_3(|x\rangle, |1\rangle) - P_3(|x\rangle, |0\rangle) \\
p_4 &= P_4(|x\rangle, |1\rangle) - P_4(|x\rangle, |0\rangle) \\
p_5 &= P_5(|x\rangle, |0\rangle) - P_5(|x\rangle, |1\rangle) \\
p_6 &= P_6(|x\rangle, |0\rangle) - P_6(|x\rangle, |1\rangle)
\end{aligned} \tag{2.17}$$

where $|x\rangle \in S_C$ is in the sample space of C and the term $P_i(|x\rangle, |n\rangle)$ expresses the probability of the state $|x\rangle$ with a mid-circuit measurement of $|n\rangle$.

The final probability distribution of the original circuit can then be reconstructed in the classical post-processing step using:

$$P(|x\rangle) = \frac{1}{2}(p_1 + p_2 + p_3 + p_4 + p_5 + p_6) \tag{2.18}$$

2.3.3 Simultaneous Virtualization of Multiple Binary Gates

Multiple binary gates can be virtualized simply by recursively sampling the circuit configurations. With this, the knitting step must also be done recursively. For instance, in the case of virtualizing two CZ gates, denoted cz_1 and cz_2, the virtualization step starts with generating all configurations C_1^1 - C_6^1 of cz_1. This step is then repeated for cz_2, resulting in C_1^2 - C_6^2. It is worth noticing that if cz_1 and cz_2 are cut simultaneously, each of C_1^1 configurations would have all six configurations of cz_2 C_1^2 - C_6^2. This results in a total of $6^2 = 36$ configurations. The total number of configurations would grow exponentially for each additional virtualized gate. In other words, to virtualize k gates, 6^k configurations need to be created and executed.

In the knitting step, the probability distribution of each combined configuration is used. Running the knitting step recursively results in an exponential overhead of 6^{k-1} calculations for k virtual gates and a recursion depth of k.

In conclusion, cutting k binary gates would result in an exponential computational cost of $O(6^k)$ for both classical and quantum computation.

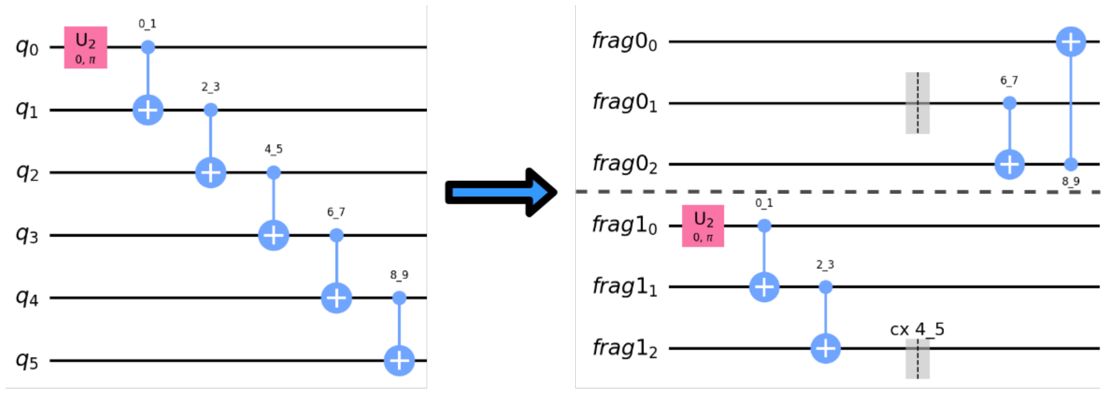


Figure 2.6: Demonstration of gate cut using binary gate virtualization

Figure 2.6 is an example of a Gate Cut using Gate Virtualization. The *CNOT gate* between q_4 and q_5 is virtualized and cut, resulting in two partitions with three qubits each.

Figure 2.7 is an example of a Wire Cut applied to the same circuit using Gate Virtualization together with Move circuit. The wire cut position is marked with a barrier labeled *WC 3_4*. *WC 3_4* is then replaced with a *Move Circuit* and is cut subsequently using *Gate Virtualization*. This results in two partitions with the requirement of three and four qubits each.

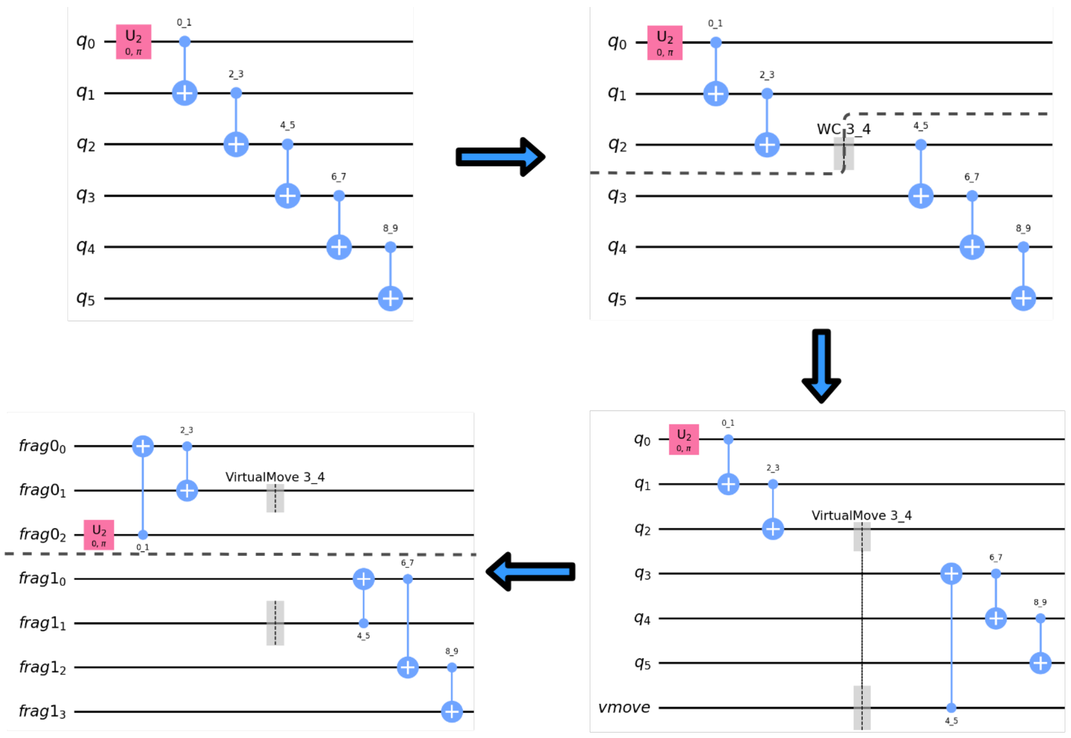


Figure 2.7: Demonstration of wire cut using Move circuit and binary gate virtualization

2.4 Quantum Teleportation

2.4.1 Measurement postulate of Quantum Mechanics

Quantum teleportation is a technique used to solve the following problem: if Alice has a qubit $|\psi\rangle$ with an unknown state, how could she send this qubit to Bob. This problem is trivial in CC. Alice could simply look at the state of $|\psi\rangle$, then transmit that to Bob. However, in QC, this is not possible. Because once Alice measures $|\psi\rangle$, the state of $|\psi\rangle$ will be disturbed due to the following measurement postulate of quantum mechanics [29]. According to the measurement postulate of Quantum Mechanics, the following completeness equation needs to be satisfied:

$$\sum_m M_m^\dagger M_m = \mathbb{I} \tag{2.19}$$

M_m represents a collection of *measurement operators* acting on the space $\mathbb{C}^2$ of the quantum system that we measure. The collection M_m describes the Quantum measurements [29].

All measurement outcomes that might occur are represented by the index m. For instance, if the quantum system's state is $|\psi\rangle$ right before the measurement, then the probability that the system will output m is shown by:

$$p(m) := \langle\psi| M_m^\dagger M_m |\psi\rangle \tag{2.20}$$

and the system's state right after the measurement is represented by:

$$\frac{M_m |\psi\rangle}{\sqrt{p(m)}} \tag{2.21}$$

Expression 2.21 shows that the state of a qubit will change after the measurement. This proves the previous statement, that in QC, Alice cannot simply read the information and transfer that to Bob.

2.4.2 No-cloning theorem

Is it possible to just simply copy the state of $|\psi\rangle$ and send it to Bob? According to the *no-cloning theorem*, this is not possible either in quantum mechanics. The *no-cloning theorem* states that arbitrary quantum states cannot be cloned by unitary evolution [29]. This is shown in the following theorem: Given a "quantum copy machine" that can be represented by an unitary operator, i.e. U is two-qubit gate that clones one qubit to another one. Given $|s\rangle$ is an arbitrary initial qubit, U can only clone qubits that are orthogonal or parallel to $|\psi\rangle$.

$$U|\psi\rangle|s\rangle = |\psi\rangle|\psi\rangle \tag{2.22}$$

To prove this theorem, another state will be cloned following Equation 2.22:

$$U|\phi\rangle|s\rangle = |\phi\rangle|\phi\rangle \tag{2.23}$$

To compare 2.22 and 2.23, first the inner products between the terms' left-hand sides are calculated:

$$\langle\phi| \otimes \langle s| U^\dagger U |\psi\rangle \otimes |s\rangle = \langle\phi|\psi\rangle \langle s|s\rangle = \langle\phi|\psi\rangle \tag{2.24}$$

now consider the right-hand sides:

$$((\langle\phi| \otimes \langle\phi|)(|\psi\rangle \otimes |\psi\rangle)) = \langle\phi|\psi\rangle \langle\phi|\psi\rangle = \langle\phi|\psi\rangle^2 \tag{2.25}$$

to prove this theorem, the equality of 2.24 and 2.25 is checked:

$$\langle \phi | \psi \rangle = \langle \phi | \psi \rangle^2 \tag{2.26}$$

consider $x = \langle \phi | \psi \rangle$, the equation 2.26 can be seen as $x = x^2$ with only two possible solutions:

- $x = 0$: $|\psi\rangle$ and $|\phi\rangle$ are orthogonal
- $x = 1$: $|\psi\rangle$ and $|\phi\rangle$ are parallel

With this, it is proven that U can only clone qubits that are parallel or orthogonal to $|\psi\rangle$.

2.4.3 Cutting with teleportation

Gate cut: To simultaneously cut multiple gates, quantum gate teleportation protocols could be used [13]. With this protocol, a quantum gate that belongs to two different partitions can be cut by utilizing a preexisting Bell state shared between these two partitions and using a simple Local Operation Classical Communication (LOCC) protocol [13]. In general, the sampling overhead of gate teleportation is lower than that of gate virtualization [15]. However, gate teleportation requires one extra ancilla qubit for each partition. Furthermore, classical communication between partitions is required which could result in extra latency.

2.4.4 Teleportation Protocol

To transfer a qubit with an unknown state $|\psi\rangle$, *Quantum Teleportation Protocol* can be used. To use the *Quantum Teleportation Protocol*, an extra ancilla qubit is required since this protocol depends on a pre-shared qubit pair with Bell's state (i.e. an Einstein-Podolsky-Rosen (EPR) pair) between Alice and Bob as shown in Figure 2.8.

In Figure 2.8, the qubit $|\psi\rangle$ is the one that Alice wants to teleport to Bob. The bottom two qubits are the EPR pair that has been distributed to Alice and Bob. Alice could then perform a CNOT operation between $|\psi\rangle$ and her EPR pair, followed by a Hadamard gate on $|\psi\rangle$. Now Alice can measure her $|\psi\rangle$ qubit and EPR pair, then transfer the result to Bob via classical mean. Bob then can use this result and apply the operation X and Z if necessary. The teleportation procedure is then completed and Bob holds the $|\psi\rangle$ qubit [29].

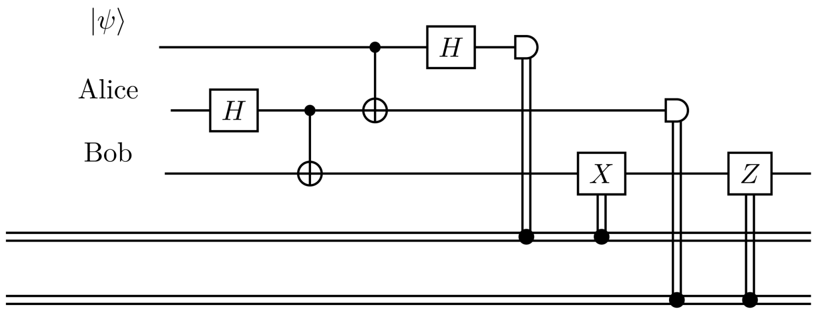


Figure 2.8: Quantum Teleportation Protocol

3 System Design

In this section, the important preprocessing steps for the circuit will be described. Afterward, the detailed design of the SMT model including model variables, model constraints, and objective functions will be studied further [15].

3.1 Overview

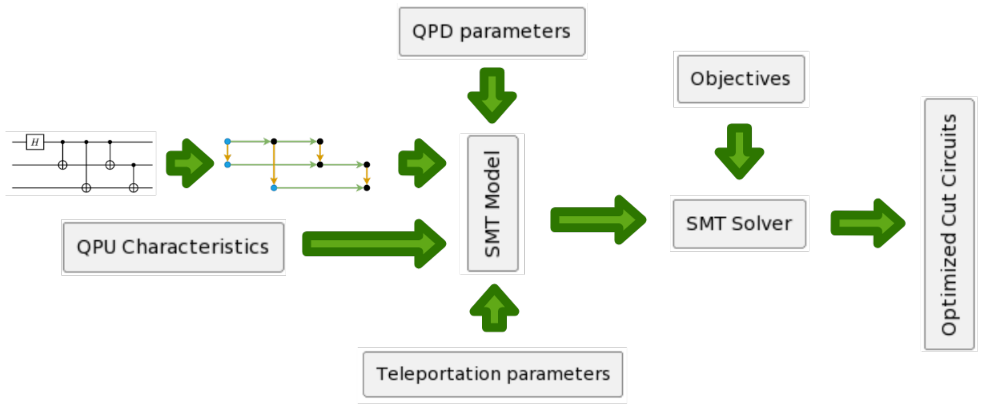


Figure 3.1: System overview

Figure 3.1 shows an overview of the system. First the *input circuit* is preprocessed and the *cut graph* is obtained. Afterward, the *QPU characteristics*, *QPD parameters*, *teleportation parameters*, together with the *cut graph* will be used to create the *SMT model*. This model will then be passed to the *SMT solver* together with the objective functions. The solver will try to find the best cut positions possible. If there exists any solutions, the cut edges will be virtualized or replaced with Move Circuit and the qubit assignments will be used to create new optimized subcircuits.

3.2 Preprocess the circuit down to the graph problem

Quantum circuits typically contain different types of gates including e.g. unary gates, binary gates and n-qubit gates. First, the circuit will be transpiled so it will only contain either unary gates or binary gates. After the transpilation step, the circuit will be converted into a *cutting graph*:

$$\xi = (V, E) \tag{3.1}$$

$$E := G \cup W \tag{3.2}$$

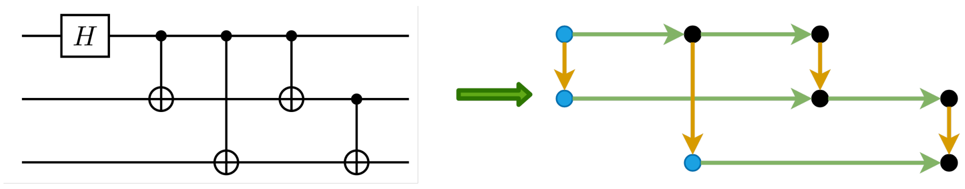


Figure 3.2: Transform circuit (left) into a cutting graph (right). Horizontal edges (green) are wire dependencies and represent potential wire cuts. Vertical edges (orange) are gate dependencies and represent potential gate cuts.

Figure 3.2 shows an example of how the preprocessing steps are done. For each binary gate in the circuit, two ends of the binary gate create two vertices. V is the set containing all vertices in the circuit and are displayed in Figure 3.2 as the blue and black dots. W is the set with all the edges that have wire dependencies. All edges that belong to W are the green edges in Figure 3.2. Analogously G is the set with all gate dependencies edges. G edges are shown in Figure 3.2 as orange edges.

The preprocessing steps are as follows: Firstly, all unary gates are removed from the original circuit since they do not have any impact on our model. Afterward, the binary gates will be processed. For each gate g between qubit q_u and q_v, two vertices u and v are created in the cutting graph ξ. An edge $s_g = (u, v) \in G$ is created to represent a gate dependency i.e. potential-gate-cut edge. For each pair of binary gates (h, k), where h is an immediate predecessor of k, an edge w is created for each overlapping qubit. This newly created edge w represents a wire dependency i.e. potential-wire-cut edge and belongs to W. Edge $w = (a, b) \in W$ is created for each vertex where $a \in s_h$ and $b \in s_k$ with $q_a = q_b$. Furthermore, given that $I \subseteq V$, I is the set of the first vertices from each qubit, denoted as a blue dot in Figure 3.2. After the circuit is preprocessed and all of the edges and vertices are identified, they can be used to design the model.

3.3 Designing the model

3.3.1 Model Variables

From the preprocessing steps, the cutting graph $\xi = (V, E)$ with $E := (G \cup W)$ is obtained. P is the set that contains all of the partitions p. The following model variables are defined:

- Set $O = \{o_{v,p} \mid v \in V, p \in P\}$ - The variable $o_{v,p}$ is evaluated to *True* if vertex v from the binary gate is assigned to partition p.
- Set $C = \{c_e \mid e \in E\}$ - The variable c_e is evaluated to *True* if the edge e should be cut using virtualization.
- Set $B = \{b_e \mid e \in E\}$ = The variable b_e is evaluated to *True* if the edge e should be cut using teleportation. In some other literature, this is known as *simultaneous cut* [15].

Furthermore, the following constants are defined in the algorithm:

- $|Q|$ defines the maximum number of qubits each partition can have
- $|C|$ is the maximum number of QPD cuts that a partition can have.

Optionally, the maximum number of partitions $|P|$ can be defined. Otherwise $|P|$ should be constrained by the number of vertices in the cutting graph $|V|$. This constraint is based on the worst case that each vertex v will be assigned to a unique partition p.

3.3.2 Model Constraints

When the variable $o_{v,p} \in O$ evaluates to *True*, the vertex v will be assigned to partition p. Given an edge $e = (u, v)$ where $e \in E$, if each of its vertex is assigned to a different partition, the edge e will be cut. Given this, the definition of $c_{e=(u,v)}$ is as follow:

$$c_{e=(u,v)} := \bigvee_{p \in P} o_{u,p} \neq o_{v,p} \tag{3.3}$$

Furthermore, each vertex $v \in V$ needs to be constrained so that it is assigned only once to exactly one partition:

$$o_{v,p} \rightarrow \neg o_{v,p'} \quad \text{where} \quad p, p' \in P \land p \neq p' \tag{3.4}$$

Each vertex should be assigned to one partition:

$$\bigvee_{p \in P} o_{v,p} \tag{3.5}$$

To reiterate, if an edge is cut using virtualization, the variable c_e will evaluate to *True* and b_e will be *False*. If edge e is cut using *Teleportation*, both c_e and b_e will evaluate to *True*. Given this, b_e is *True* if and only if c_e is *True*:

$$b_e \rightarrow c_e \quad \forall e \in E \tag{3.6}$$

The assignment of v and p in variable $o_{v,p} \in O$ at the same time determines the number of partitions. Together with cut variables c_e and b_e, the total number of qubits Q_p of partition p is defined as:

$$Q_p := \sum_{v \in I \subseteq V} o_{v,p} + \sum_{e=(u,v) \in W} c_e \wedge o_{v,p} + \sum_{e=(u,v) \in E} b_e \wedge (o_{v,p} \vee o_{u,p}) = A + B + C \tag{3.7}$$

In the term A of Equation 3.7, I is the set that contains all the first vertex of each qubit. For each partition p, each starting vertex v that is assigned to p would require one qubit.

Term B of Equation 3.7 describes the qubit requirement for a wire cut using a *Move circuit* instead of teleportation. Given a wire cut edge e with vertices u and v. If the circuit should be cut at edge e, the partition p that the vertex v was assigned to would require one ancilla qubit. The partition that the vertex u was assigned to could reuse the existing qubit, hence no ancilla qubit is needed.

Term B of Equation 3.7 describes the qubit requirement for any cut that uses teleportation. Given an arbitrary edge e with vertices u and v. If the circuit will be cut at edge e and the vertices are assigned to p_u and p_v, both p_u and p_v will each require an extra ancilla qubit for the *Teleportation Protocol*.

The total number of QPD cuts that each partition contains can be found as follows:

$$C_p := \sum_{e=(u,v) \in E} \neg b_e \wedge c_e \wedge (o_{v,p} \vee o_{u,p}) \tag{3.8}$$

Given an edge e, the number of QPD cuts for partition p is increased by one if QPD is used instead of teleportation.

$|Q|$ is defined as the maximum number of qubits that a partition can have. This results in the following constraint:

$$Q \geq Q_p \quad \forall p \in P \tag{3.9}$$

Similarly, $|C|$ is defined as the maximum number of QPD cuts that a partition can have. This results in the following constraint:

$$C \geq C_p \quad \forall p \in P \tag{3.10}$$

The total number of QPD cuts that are currently being used is defined as:

$$totalQpdCuts = \sum_{e \in E} c_e \wedge \neg b_e \tag{3.11}$$

Teleportation reduces the overall sampling overhead of the circuit. However, this method requires two ancilla qubits and classical communication to work. Furthermore, by introducing classical communication across QPU, further latency is expected. Cutting using QPD has minimal ancilla qubit requirements and introduces insignificant latency from classical communication. However, the sampling overhead of QPD cuts is higher: six for gate cutting and eight for wire cutting. To use both methods QPD and teleportation for cutting, a constraint should be enforced to only introduce teleportation after the maximum number of allowed QPD cuts is reached. This constraint is defined as follows:

$$b_e \rightarrow (totalQpdCuts == maxQpdCuts) \quad \forall e \in E \tag{3.12}$$

The variable *maxQpdCuts* defines the maximum number of allowed QPD. For this thesis, maxQpdCuts is chosen to be 5. In the worst case where only wire cut is used, the sampling overhead for the circuit would be $8^5 = 32768$.

Furthermore, it has been shown that if teleportation happens early in the circuit, the outcome will be less useful since the noise will accumulate along the teleportation path. For this reason, it is preferable to have teleportation applied later in the circuit. This constraint is modeled as follows:

$$maxQpdEdgeIdx < minTeleportationEdgeIdx \tag{3.13}$$

For each edge $e \in E$, there is an index *idx* associated with it. These indices have a topological order based upon qubits and gates dependencies order i.e. the higher the index, the later this edge e will appear. The index *idx* of the last QPU cut could be retrieved by getting the maximum index *idx* from the list of all QPD cuts' indices. Similarly, the index *idx* of the first Teleportation cut can be obtained by getting the minimum index *idx* from the list of all teleportation cuts' indices. By enforcing the last QPD cut's index *idx* to be smaller than the first teleportation cut's index, it can ensure that teleportation will happen only after QPD cuts. The constraint in Equation 3.13 is added as a soft constraint to allow for sub-optimal solutions.

	k Gate Cuts			Wire Cuts	
	CNOT	swap	CR(ϕ)	1	k arbitrary
ancilla + CC	$(2^{k+1}-1)^2$	16^k	$(\leq 4)^k$	9	$(2^{k+1}-1)^2$
no ancilla + no CC	9^k	49^k	$(1+2\lvert sin(\phi)\rvert)^{2k}$	16	16^k

Table 3.1: γ-factor for different cut types and configurations [15]

3.3.3 Objective Functions

The incurred sampling overhead is defined as [15]:

$$S := \prod_{e \in E} \gamma_e^2 (c_e \wedge \neg b_e) \prod_{e \in E} \gamma_{e(K)}^2 b_e \tag{3.14}$$

For a cut edge $e \in E$, it is given that:

- γ_e: is the individual cut's γ-factor
- $\gamma_{e(K)}$: is the γ-factor for K simultaneous cut.

Table 3.1 specifies the γ-factor and $\gamma_{e(K)}$-factor for different configurations [13, 15]. This work however uses the Gate Virtualization from *DQS* [28] framework, in which the γ-factors are fixed. 6^k for k gate cuts and 8^t for t wire cuts respectively.

To minimize the number of qubits that each partition requires, and to distribute the qubits in a balanced manner, the following objective is applied:

$$min(Q) \tag{3.15}$$

Analogously, to distribute the number of QPD cuts in a balanced manner between partitions, the following objective is used:

$$min(C) \tag{3.16}$$

4 Evaluation

4.1 Implementation

A prototype of this thesis was implemented using the Python framework. The SMT model was implemented using Z3-prover, an open source SAT solver provided by *Microsoft Research*. Z3 was developed to solve problems that were encountered in software verification and program analysis. Z3 natively supports fixed-size bit-vectors, extensional arrays, arithmetic, data types, uninterpreted functions and quantifiers. It is also mainly used for extended static checking, test case generation and predicate abstraction.

This work also uses the *virtualization framework* derived from the work of *DQS* [28]. This framework provides a module to create and execute virtual circuits. It can also be used to fragment the cut circuit into partitions. The generation of instantiations is also handled by *DQS*.

IBM Quantum is used as the quantum cloud provider due to its robustness and open-access model. *IBM Quantum* also provides free limited access to real quantum hardware. Together with its QC toolkit, *qiskit*, a seamless integration is achieved.

The source code for this work, together with its benchmark data can be found on Github at `http://tinyurl.com/HWAwareCuttingAndKnitting`.

4.2 Technology stacks

Qiskit provides fake backends that mimic the behaviors and characteristics of real IBM QC hardware. These fake backends can be used to execute and evaluate the cut circuits. Furthermore, Qiskit also provides simulators. These simulators represent a perfect environment that is not influenced by noise. The simulators are typically used to obtain the expected results of a circuit. This result is then used to calculate the *Hellinger fidelity*.

The utilized infrastructure for the tests consists of the machine running on NixOS equipped with AMD EPYC 7713P 64-Core processors and 512 GB of RAM.

4.3 Benchmarking

4.3.1 Setup

The implemented model was executed with benchmark quantum circuits of up to 20 qubits. The quantum circuit generator from *CutQC* is used to generate benchmark circuits [14]. The following circuits were used for the benchmark and will be cut into two partitions:

- **Supremacy**: Implemented following [30], *Supremacy* circuit is a type of 2-D random circuit. This benchmark circuit has highly dense probability output and was used by Google to illustrate quantum advantage [31].
- **Syscamore**: is a different implement of *Supremacy* circuit based on the work of Arute et al. [31].
- **Approximate Quantum Fourier Transform (AQFT)**: Quantum Fourier Transform (QFT) [32] is a typical procedure that can be found in numerous quantum algorithms that provides a potential speedup compared to classical algorithms. On NISQ hardware, it has been shown that AQFT yields better results compared to that of QFT [33].
- **Adder**: Having one ancilla qubit and linear depth, *Adder* circuit is a quantum ripple-carry adder [34]. The *Adder circuit* only allows even numbers of qubits, since they do a summation of two quantum registers with the same width. *Adder circuit* is considered to be an important procedure in quantum arithmetic.
- **Hardware efficient ansatz (HWEA)**: HWEA circuit belongs to the near-term variational applications. This category of quantum algorithm has shown great potential on NISQ hardware [35].

In the following experiments, the backend from *Aer provider* was used to obtain the expected results from both the input benchmark circuit and the cut circuit. *Aer provider* offers different high performance simulator backends for different simulation methods. In this work, *AerSimulator* was used exclusively.

To observe how the circuit behaves in noisy real-world situations, the *fake provider* from *qiskit* was used to evaluate both the cut circuit and original input benchmark circuit. Specifically, the *FakeKolkataV2* is used. It is modeled according to the *ibmq_kolkata* hardware, which is a superconducting Falcon r5.11 QPU with 27 physical qubits. Figure 4.1 shows the topology of this QPU. The following gates are natively supported by this QPU: *CX*, *ID*, *RZ*, *SX*, *X*.

Each of the benchmark circuits is executed in four different configurations:

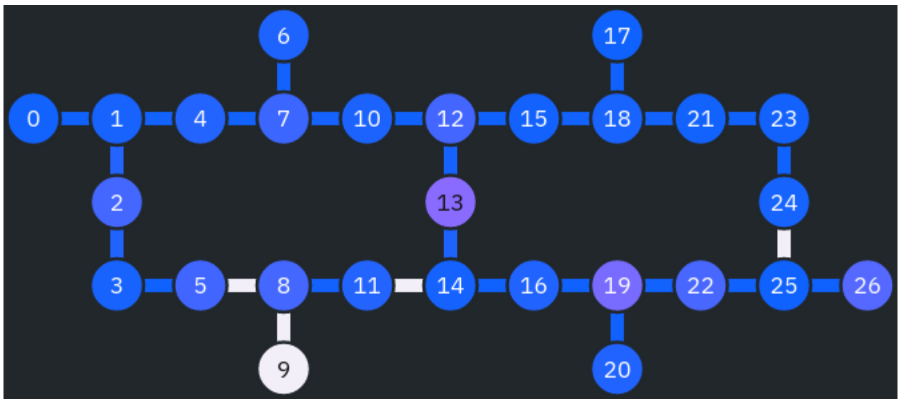


Figure 4.1: Topology of Kolkata [36]

- input circuit executed on *AerSimulator*
- input circuit executed on *FakeKolkataV2*
- cut circuit executed on *AerSimulator*
- cut circuit executed on *FakeKolkataV2*

For each benchmark circuit, all four configurations was run to compare their fidelity. All circuits was optimized for the backend that they will run on with level 3 : the highest optimization level possible. After the optimization step, each configuration was run with 1000 shots and the fidelity, number of CNOT gates and the depth between the original and the cut circuit will be compared.

4.3.2 Results

Fidelity comparison

For each original circuit and their cut counterpart, the experiment was executed three times and the average fidelities are calculated.

The benchmark circuits are cut as described in Table 4.1.

Figure 4.2 and Table 4.2 show the results of the experiment. From these results, the partitioning method has shown to improve the accuracy of quantum circuits. Depending on the size and shape of the circuit, the improvement in fidelity varies differently. The geometric mean of improvements from this work is 369%.

For the case of *AQFT 6 qubits*, the fidelity slightly dips. This is due to the sampling overhead. With two gate cuts, and two wire cuts, the cut circuit has a sampling over-

	Number of gate cuts	**Number of wire cuts**
Adder 10 qubits	0	2
AQFT 6 qubits	2	2
GHZ 24 qubits	1	0
HWE 10 qubits	1	0
Supremacy 12 qubits	3	0
Syscamore 12 qubits	0	0
Supremacy 20 qubits	5	0

Table 4.1: Cuts from different circuits

	Fidelity changes in %
Adder 10 qubits	+12.83%
AQFT 6 qubits	-1.5%
GHZ 24 qubits	+36.73%
HWE 10 qubits	+13.63%
Supremacy 12 qubits	+520.27%
Syscamore 12 qubits	+433.39%
Supremacy 20 qubits	+16225.34%

Table 4.2: Fidelity changes between the original circuits and the cut circuits

head of $6^2 * 8^2 = 2304$ for each partition. Hence, the state space of the cut circuit grows and needs more shots to achieve the same fidelity.

For the case of *Syscamore 12 qubits*, there is no cut involved. The circuit contains independent fragments and the model was able to determine these fragments. By simply rearranging qubits and gates into two fragments, the fidelity is improved by 433.39%.

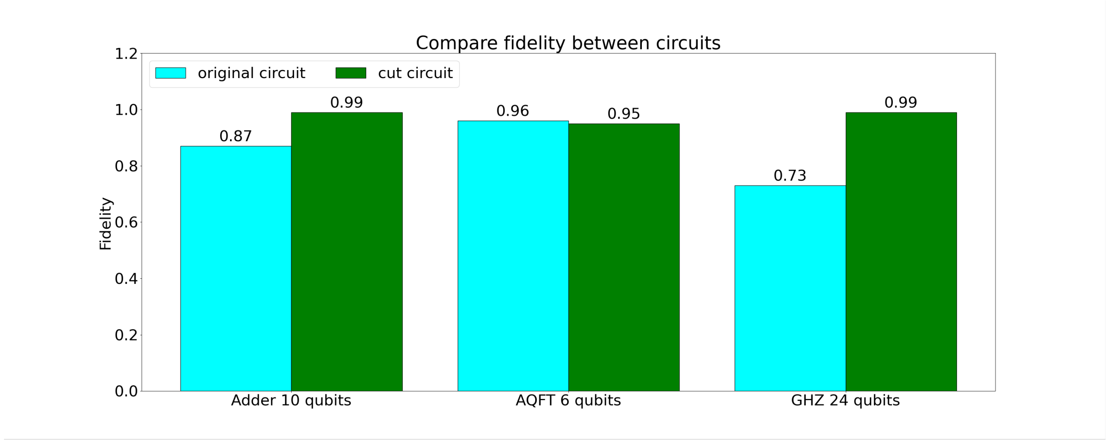
By using five gate cuts, the fidelity of *Supremacy 20 qubits* was improved significantly by 16225.34%. This is possible since the number of CNOT gates and depth of partitions is greatly reduced.

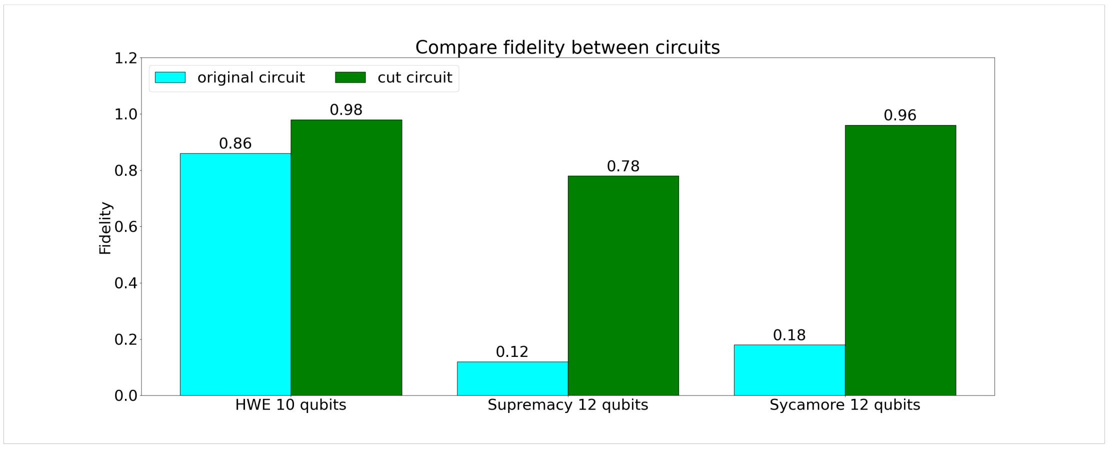
Number of CNOT gates and circuit's depth comparison

The *Transpilation* process is used to optimize both the original and the cut circuit and to make the circuits executable on the backend. It is however not guaranteed that the *Transpilation* process will also return the same transpiled circuit. Hence it will result in a different number of CNOT gates and circuit depth. For this reason, the experiment is also executed three times for each of the original circuits and their cut circuits.

In the following, the number of CNOT gates and the depth of the original circuits are shown. For the cut circuit, the maximum number of CNOT gates and maximum depth from all of the subcircuits are shown. Figure 4.3 - 4.10 shows the results of this experiment:

By partitioning the circuit, the number of CNOT gates and circuit depth could be reduced significantly. On average, the number of CNOT gates could be reduced by half. In the case of *GHZ 24 qubits* shown in Figure 4.4, the number of CNOT gates is reduced by approximately sixfold. Furthermore, this technique has shown no penalty on circuit depth. For most cases, the circuit depth is reduced considerably, up to four times in the case of *GHZ 24 qubits* shown in Figure 4.8.





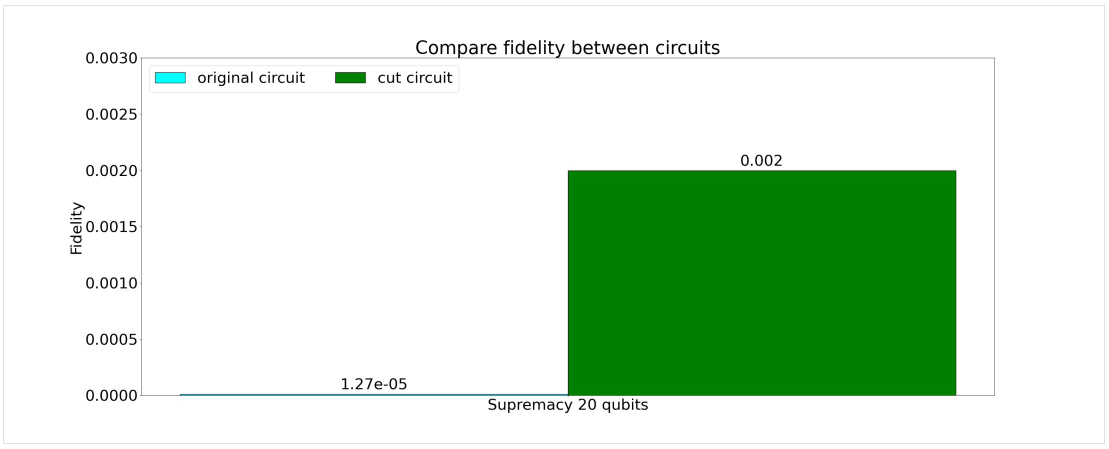


Figure 4.2: Fidelity comparison between the original and the cut circuits

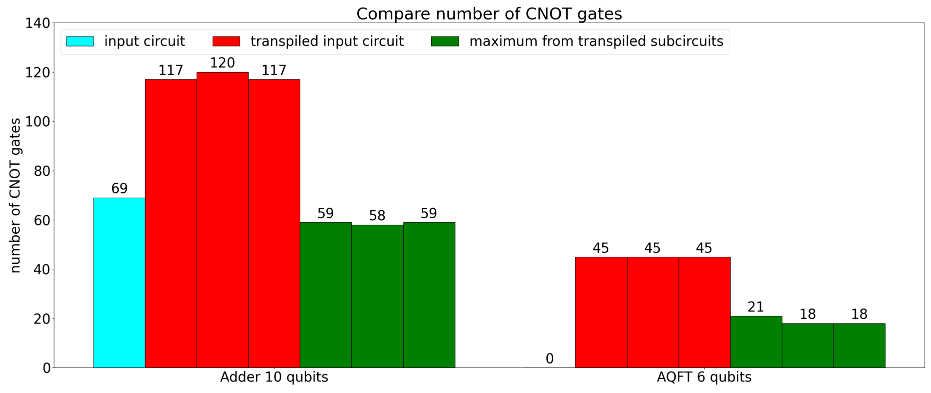


Figure 4.3: Number of CNOT gates comparison between original and cut circuit part 1

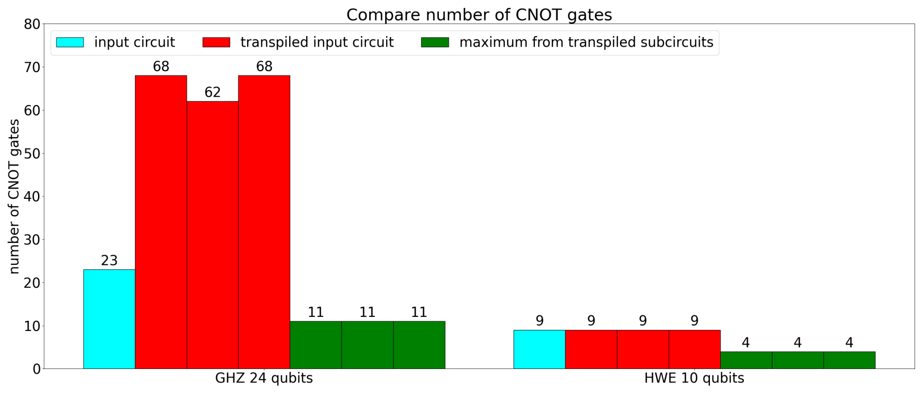


Figure 4.4: Number of CNOT gates comparison between original and cut circuit part 2

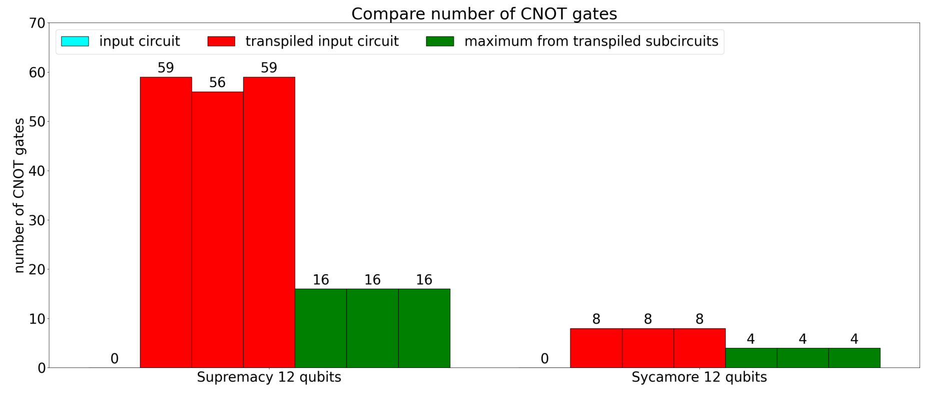


Figure 4.5: Number of CNOT gates comparison between original and cut circuit part 3

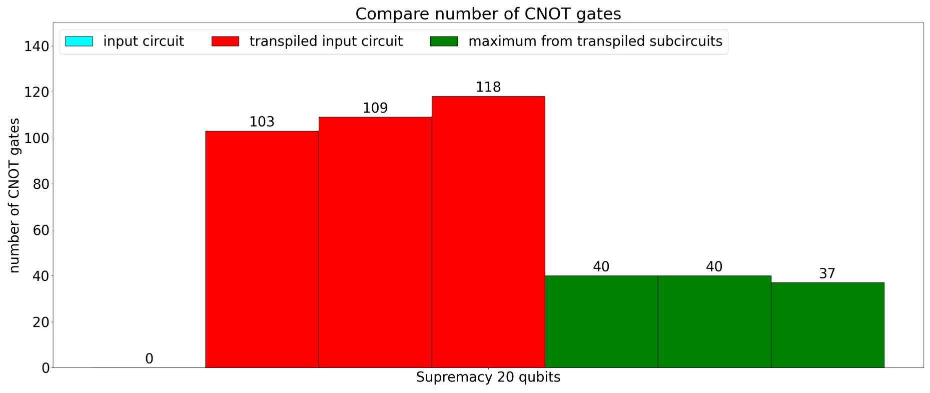


Figure 4.6: Number of CNOT gates comparison between original and cut circuit part 4

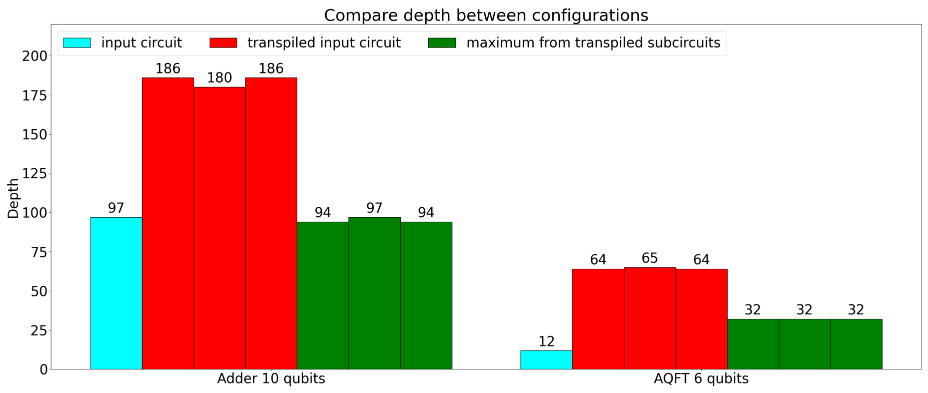


Figure 4.7: Depth comparison between original and cut circuit part 1

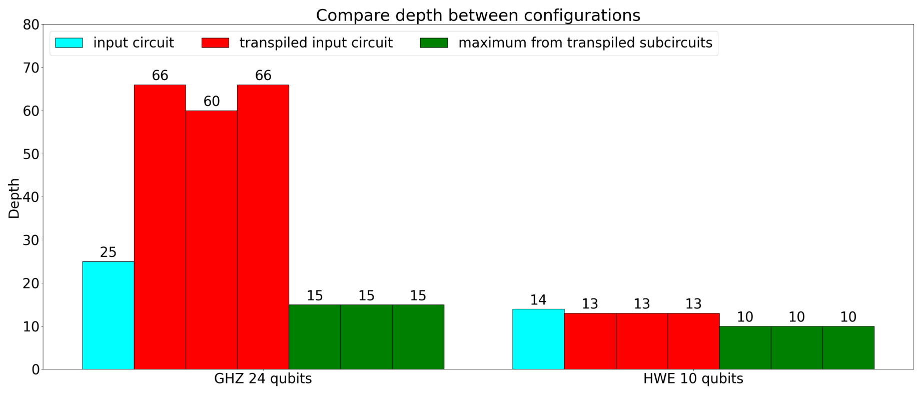


Figure 4.8: Depth comparison between original and cut circuit part 2

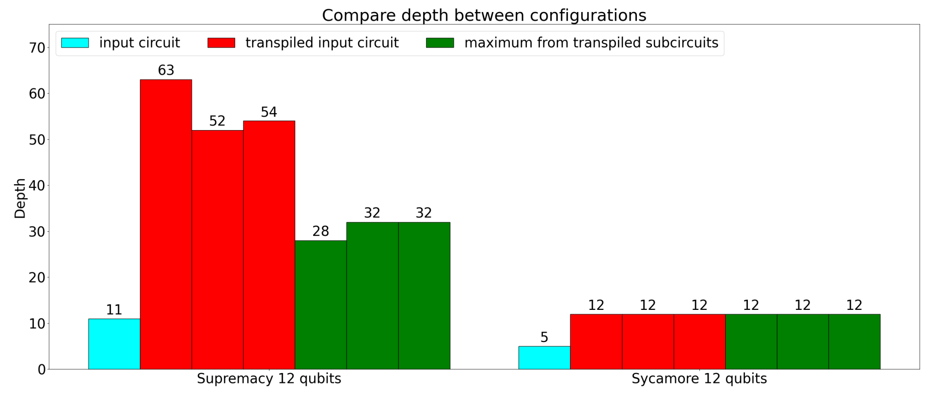


Figure 4.9: Depth comparison between original and cut circuit part 3

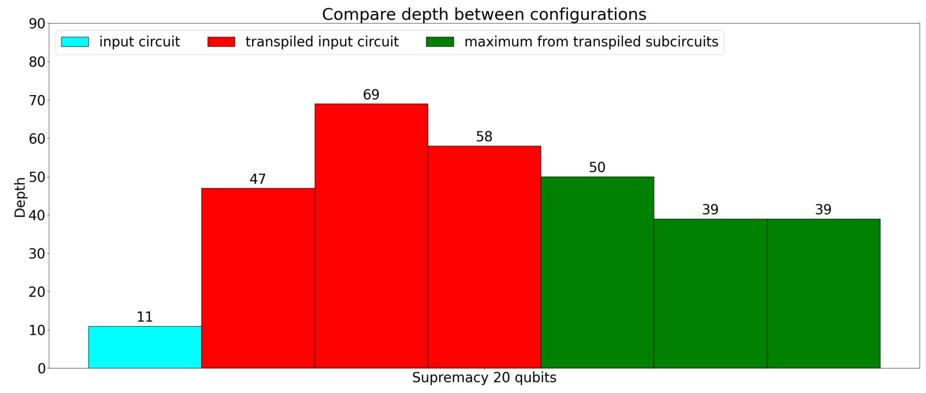


Figure 4.10: Depth comparison between original and cut circuit part 4

5 Related Works

The techniques to resolve gate dependencies have been explored by different groups [22, 37–39]. The previous works in [11, 12, 14, 16, 23, 24, 40–42] have also proposed different ways to resolve wire dependencies via wire cutting. Furthermore, the works from [11, 12, 14, 22, 41, 42] does not use classical communication. By using ancilla qubits and classical communication, the works in [13, 16, 24, 40] could reduce the sampling overhead.

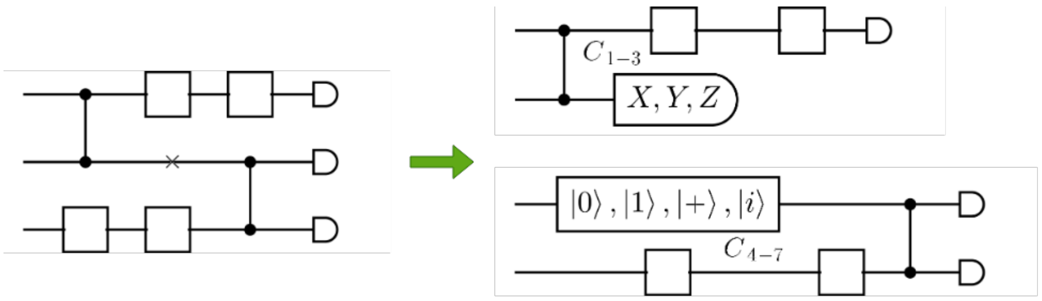


Figure 5.1: Wire cut method proposed by Tang et al. [14]

A qubit's wire describes how a qubit evolves over time. Hence wire cutting technique is a time-wise partitioning approach. The Gate Virtualization technique instead will partition the circuit in a space-wise manner.

Previous works have also tried to solve the problem of cutting a circuit into smaller partitions using only wire cuts without classical communication [12, 14, 41]. As mentioned previously in Table 3.1, wire cuts have an exponential sampling overhead of 16^k for individual and simultaneous cuts when classical communication and ancilla qubits are not used. The overhead is 4^k if classical communication and one ancilla qubit is used. Depending on the shape and size of the circuit, using only wire cuts might create larger subcircuits than using both wire cuts and gate cuts together. Since these previous works did not consider gate cuts, it is not possible to calculate the minimal cost partitionings with gate cuts using low sampling overhead gates such as CR gates. Hence, depending on the input circuit, previous works might incur a larger number of cuts with higher cost in sampling overhead. Shown in Figure 5.1 is the proposed wire cut method by Tang et al. [14]. The wire of the second qubit is cut at the cross position.

The cut results in two smaller subcircuits that can be run independently from each other. The subcircuits are then sampled and their results are merged with the *knitting process*. After the knitting step, the probability distribution of the original input circuit is obtained. The subcircuit on the left of the cut location, C_{1-3}, is obtained by measuring at the cut location in the X, Y and Z basis. To obtain the subcircuit on the right of the cut location, C_{4-7}, the four input states $|0\rangle, |1\rangle, |+\rangle, |i\rangle$ are prepared and sampled. The results are knitted using a knitting formula. The probability distribution of subcircuit C_{1-3} is used to weigh the probability distribution of subcircuit C_{4-7} in different input states. The sampling overhead of the subcircuits and the knitting overhead is 4^k for k wire cuts [14].

6 Conclusion and Outlook

In this thesis, a proof-of-concept for a quantum circuit partitioning method that uses both wire cut and gate cut was implemented. Furthermore, the model in this work also limits the number of QPD cuts and delays the use of teleportation to a later time frame in the circuit. By applying these methods and constraints, the requirement for qubit is reduced. Furthermore, the fidelity also is improved since the effects of noise are mitigated.

Moreover, this work also has made the partitioning method hardware-aware. By providing a list of backends that should be used, the model will find the best assignment that matches the partitions' qubit requirement to the number of qubits that each QPU has. This could be further improved by making the model aware of the backend's topology and noise characteristics. By doing so, the model could select a better cut location and a better cut method to create subcircuits that will have the best result possible for a given QPU.

This work has shown that it is possible to partition circuits into smaller subcircuits without losing accuracy. By doing so, the fidelity of the circuit could be improved by up to 16225%. The geometric mean of improvements from this work is 369%.

The Z3-prover has been shown to reach its limit when the circuit has more than 45 qubits. It was suggested the model should be ported to other more performant solvers. For instance, Answer Set Programming (ASP) solvers such as *clingo* [43] could be explored in future research since they have been shown to have better performance than Z3-prover.

It is believed that the field of circuit partitioning, gate cut and wire cut will grow significantly in the future. Especially during the NISQ era of the hardware, these techniques need to be further improved to create useful quantum applications and to fully unlock the potential of Quantum Computing.

In conclusion, gate cutting and wire cutting both have their advantages and disadvantages. By combining both, the circuit could be partitioned in a more flexible way with lower sampling overhead. Teleportation is a good candidate to be used in wire cutting and gate cutting. However, to apply the teleportation technique, an EPR pair needs to be generated between different QPUs and this is currently not yet realizable [44]. For this reason, more research and experiments on the teleportation technique need to be done to explore its potential.

Abbreviations

NISQ Noisy Intermediate-Scale Quantum

QPU Quantum Processing Unit

QC Quantum Computing

QCer Quantum Computer

CC Classical Computing

CCer Classical Computer

SAT Boolean Satisfiability Problem

SMT Satisfiability Modulo Theories

ASP Answer Set Programming

CNOT Controlled NOT

CZ Controlled Z

HW hardware

LOCC Local Operation Classical Communication

EPR Einstein-Podolsky-Rosen

QPD Quasiprobability distribution

QFT	Quantum Fourier Transform
AQFT	Approximate Quantum Fourier Transform
HWEA	Hardware efficient ansatz
ASP	Answer Set Programming

List of Figures

List of Tables

Bibliography

[1] "40 years of quantum computing." In: *Nature Reviews Physics* 4.1 (Jan. 2022), pp. 1–1. ISSN: 2522-5820. DOI: 10.1038/s42254-021-00410-6.

[2] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd. "Quantum machine learning." In: *Nature* 549.7671 (Sept. 2017), pp. 195–202. ISSN: 1476-4687. DOI: 10.1038/nature23474.

[3] P. W. Shor. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer." In: *SIAM Journal on Computing* 26.5 (Oct. 1997), pp. 1484–1509. ISSN: 1095-7111. DOI: 10.1137/s0097539795293172.

[4] E. Farhi, J. Goldstone, and S. Gutmann. *A Quantum Approximate Optimization Algorithm*. 2014. arXiv: 1411.4028 [quant-ph].

[5] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. OBrien. "A variational eigenvalue solver on a photonic quantum processor." In: *Nature Communications* 5.1 (July 2014). ISSN: 2041-1723. DOI: 10.1038/ncomms5213.

[6] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta. "Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets." In: *Nature* 549.7671 (Sept. 2017), pp. 242–246. ISSN: 1476-4687. DOI: 10.1038/nature23879.

[7] J. Preskill. "Quantum Computing in the NISQ era and beyond." In: *Quantum* 2 (Aug. 2018), p. 79. ISSN: 2521-327X. DOI: 10.22331/q-2018-08-06-79.

[8] M. Suchara, J. Kubiatowicz, A. Faruque, F. T. Chong, C.-Y. Lai, and G. Paz. "QuRE: The Quantum Resource Estimator toolbox." In: *2013 IEEE 31st International Conference on Computer Design (ICCD)*. 2013, pp. 419–426. DOI: 10.1109/ICCD.2013.6657074.

[9] J. Gambetta. *Expanding the IBM Quantum roadmap to anticipate the future of quantum-centric supercomputing*. https://research.ibm.com/blog/ibm-quantum-roadmap-2025. [Online; accessed 22-January-2024]. 2022.

[10] J. Roffe. "Quantum error correction: an introductory guide." In: *Contemporary Physics* 60.3 (July 2019), pp. 226–245. ISSN: 1366-5812. DOI: 10.1080/00107514.2019.1667078.

[11] K. Mitarai and K. Fujii. "Constructing a virtual two-qubit gate by sampling single-qubit operations." In: *New Journal of Physics* 23.2 (Feb. 2021), p. 023021. ISSN: 1367-2630. DOI: 10.1088/1367-2630/abd7bc.

[12] T. Peng, A. W. Harrow, M. Ozols, and X. Wu. "Simulating Large Quantum Circuits on a Small Quantum Computer." In: *Physical Review Letters* 125.15 (Oct. 2020). ISSN: 1079-7114. DOI: 10.1103/physrevlett.125.150504.

[13] C. Piveteau and D. Sutter. "Circuit knitting with classical communication." In: *IEEE Transactions on Information Theory* (2023), pp. 1–1. ISSN: 1557-9654. DOI: 10.1109/tit.2023.3310797.

[14] W. Tang, T. Tomesh, M. Suchara, J. Larson, and M. 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*. ASPLOS 21. ACM, Apr. 2021. DOI: 10.1145/3445814.3446758.

[15] S. Brandhofer, I. Polian, and K. Krsulich. *Optimal Partitioning of Quantum Circuits using Gate Cuts and Wire Cuts*. 2023. arXiv: 2308.09567 [quant-ph].

[16] L. Brenner, C. Piveteau, and D. Sutter. *Optimal wire cutting with classical communication*. 2023. arXiv: 2302.03366 [quant-ph].

[17] B. Zohuri and F. Mossavar-Rahmani. "What is Quantum Computing and How it Works." In: *Journal of Material Sciences & Manfacturing Research* (Dec. 2020), pp. 1–5. DOI: 10.47363/JMSMR/2020(2)104.

[18] D. Garisto. *What Is Quantum Entanglement?* https://spectrum.ieee.org/what-is-quantum-entanglement. [Online; accessed 22-January-2024]. 2022.

[19] D. Lugovoy. "Scalable Quantum Cloud Scheduling: Optimizing Resource Allocation for Efficient NISQ Computing." In: (2024).

[20] *Bloch sphere*. https://en.wikipedia.org/wiki/Bloch_sphere. [Online; accessed 27-February-2024].

[21] N. P. de Leon, K. M. Itoh, D. Kim, K. K. Mehta, T. E. Northup, H. Paik, B. S. Palmer, N. Samarth, S. Sangtawesin, and D. W. Steuerman. "Materials challenges and opportunities for quantum computing hardware." In: *Science* 372.6539 (2021), eabb2823. DOI: 10.1126/science.abb2823. eprint: https://www.science.org/doi/pdf/10.1126/science.abb2823.

[22] K. Mitarai and K. Fujii. "Overhead for simulating a non-local channel with local channels by quasiprobability sampling." In: *Quantum* 5 (Jan. 2021), p. 388. ISSN: 2521-327X. DOI: 10.22331/q-2021-01-28-388.

[23] E. Pednault. *An alternative approach to optimal wire cutting without ancilla qubits*. 2023. arXiv: `2303.08287 [quant-ph]`.

[24] H. Harada, K. Wada, and N. Yamamoto. *Doubly optimal parallel wire cutting without ancilla qubits*. 2023. arXiv: `2303.07340 [quant-ph]`.

[25] F. Hua, Y. Jin, Y. Chen, S. Vittal, K. Krsulich, L. S. Bishop, J. Lapeyre, A. Javadi-Abhari, and E. Z. Zhang. *Exploiting Qubit Reuse through Mid-circuit Measurement and Reset*. 2023. arXiv: `2211.01925 [quant-ph]`.

[26] S. Brandhofer, I. Polian, and K. Krsulich. *Optimal Qubit Reuse for Near-Term Quantum Computers*. 2023. arXiv: `2308.00194 [quant-ph]`.

[27] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

[28] N. Tornow. "DQS: A Framework for Efficient Distributed Simulation of Large Quantum Circuits." In: (2022).

[29] R. LaRose. "https://www.ryanlarose.com/uploads/1/1/5/8/115879647/quic02.pdf." In: ().

[30] S. Boixo, S. V. Isakov, V. N. Smelyanskiy, R. Babbush, N. Ding, Z. Jiang, M. J. Bremner, J. M. Martinis, and H. Neven. "Characterizing quantum supremacy in near-term devices." In: *Nature Physics* 14.6 (Apr. 2018), pp. 595–600. ISSN: 1745-2481. DOI: `10.1038/s41567-018-0124-x`.

[31] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. S. L. Brandao, D. A. Buell, B. Burkett, Y. Chen, Z. Chen, B. Chiaro, R. Collins, W. Courtney, A. Dunsworth, E. Farhi, B. Foxen, A. Fowler, C. Gidney, M. Giustina, R. Graff, K. Guerin, S. Habegger, M. P. Harrigan, M. J. Hartmann, A. Ho, M. Hoffmann, T. Huang, T. S. Humble, S. V. Isakov, E. Jeffrey, Z. Jiang, D. Kafri, K. Kechedzhi, J. Kelly, P. V. Klimov, S. Knysh, A. Korotkov, F. Kostritsa, D. Landhuis, M. Lindmark, E. Lucero, D. Lyakh, S. Mandrà, J. R. McClean, M. McEwen, A. Megrant, X. Mi, K. Michielsen, M. Mohseni, J. Mutus, O. Naaman, M. Neeley, C. Neill, M. Y. Niu, E. Ostby, A. Petukhov, J. C. Platt, C. Quintana, E. G. Rieffel, P. Roushan, N. C. Rubin, D. Sank, K. J. Satzinger, V. Smelyanskiy, K. J. Sung, M. D. Trevithick, A. Vainsencher, B. Villalonga, T. White, Z. J. Yao, P. Yeh, A. Zalcman, H. Neven, and J. M. Martinis. "Quantum supremacy using a programmable superconducting processor." In: *Nature* 574.7779 (Oct. 2019), pp. 505–510. ISSN: 1476-4687. DOI: `10.1038/s41586-019-1666-5`.

[32] J. W. Cooley and J. W. Tukey. "An Algorithm for the Machine Calculation of Complex Fourier Series." In: *Mathematics of Computation* 19.90 (1965), pp. 297–301. ISSN: 00255718, 10886842.

[33] A. Barenco, A. Ekert, K.-A. Suominen, and P. Törmä. "Approximate quantum Fourier transform and decoherence." In: *Phys. Rev. A* 54 (1 July 1996), pp. 139–146. DOI: 10.1103/PhysRevA.54.139.

[34] S. A. Cuccaro, T. G. Draper, S. A. Kutin, and D. P. Moulton. *A new quantum ripple-carry addition circuit*. 2004. arXiv: quant-ph/0410184 [quant-ph].

[35] N. Moll, P. Barkoutsos, L. S. Bishop, J. M. Chow, A. Cross, D. J. Egger, S. Filipp, A. Fuhrer, J. M. Gambetta, M. Ganzhorn, A. Kandala, A. Mezzacapo, P. Müller, W. Riess, G. Salis, J. Smolin, I. Tavernelli, and K. Temme. "Quantum optimization using variational algorithms on near-term quantum devices." In: *Quantum Science and Technology* 3.3 (June 2018), p. 030503. ISSN: 2058-9565. DOI: 10.1088/2058-9565/aab822.

[36] *IBM Quantum*. https://quantum.ibm.com/. [Online; accessed 27-February-2024].

[37] C. Ufrecht, M. Periyasamy, S. Rietsch, D. D. Scherer, A. Plinge, and C. Mutschler. "Cutting multi-control quantum gates with ZX calculus." In: *Quantum* 7 (Oct. 2023), p. 1147. ISSN: 2521-327X. DOI: 10.22331/q-2023-10-23-1147.

[38] S. Bravyi, G. Smith, and J. A. Smolin. "Trading Classical and Quantum Computational Resources." In: *Phys. Rev. X* 6 (2 June 2016), p. 021043. DOI: 10.1103/PhysRevX.6.021043.

[39] A. Eddins, M. Motta, T. P. Gujarati, S. Bravyi, A. Mezzacapo, C. Hadfield, and S. Sheldon. "Doubling the Size of Quantum Simulators by Entanglement Forging." In: *PRX Quantum* 3.1 (Jan. 2022). ISSN: 2691-3399. DOI: 10.1103/prxquantum.3.010309.

[40] A. Lowe, M. Medvidovic, A. Hayes, L. J. O Riordan, T. R. Bromley, J. M. Arrazola, and N. Killoran. "Fast quantum circuit cutting with randomized measurements." In: *Quantum* 7 (Mar. 2023), p. 934. ISSN: 2521-327X. DOI: 10.22331/q-2023-03-02-934.

[41] T. Ayral, F.-M. Le Regent, Z. Saleem, Y. Alexeev, and M. Suchara. "Quantum Divide and Compute: Hardware Demonstrations and Noisy Simulations." In: *2020 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, July 2020. DOI: 10.1109/isvlsi49217.2020.00034.

[42] T. Ayral, F.-M. L. Régent, Z. Saleem, Y. Alexeev, and M. Suchara. "Quantum Divide and Compute: Exploring the Effect of Different Noise Sources." In: *SN Computer Science* 2.3 (Mar. 2021). ISSN: 2661-8907. DOI: 10.1007/s42979-021-00508-9.

[43] *clingo*. https://potassco.org/clingo/. [Online; accessed 27-February-2024].

[44] M. Bechtold, J. Barzen, F. Leymann, and A. Mandl. *Circuit Cutting with Non-Maximally Entangled States*. 2023. arXiv: `2306.12084 [quant-ph]`.