MSc thesis in Mapping in Quantum Computers

Design Space Exploration For Mapping In Quantum Computers

Hossein Zarein
2020
DESIGN SPACE EXPLORATION FOR MAPPING IN QUANTUM COMPUTERS

A thesis submitted to the Polytechnic University of Catalonia in partial fulfillment of the requirements for the degree of

Master of Electronics Engineering

by

Hossein Zarein

November 2020

Affiliation: Quantum and Computer Architecture Lab

Department of Quantum and Computer Engineering
Faculty of Electrical Engineering, Mathematics and Computer Sciences
Delft University of Technology
Project duration: September, 2019 – November, 2020
Thesis committee: Prof. dr. ir. José Antonio Rubio Sola Prof. dr. ir. Marcos Postigo
Supervisors: Prof. dr. ir. Eduard Alarcon Prof. dr. ir. Carmen G. Almudever
ABSTRACT

Quantum computers have emerged as an alternative computer paradigm that will allow to solve some complex problems of large numbers that are not tractable for classical computers, e.g. factorization. Quantum computers also have the ability to simulate quantum mechanics in a real and direct way. The development of quantum computers faces many unique obstacles.

In recent years, various quantum computer prototypes have been created by companies such as Google and IBM, on which we can run some quantum algorithms. But these prototypes are far from the quantum computers we envision and can function like today’s classical computers. It is said that we are in the time of Noisy Intermediate-Scale Quantum (NISQ) computers. This means that current quantum computers are very noisy and their size is very small compared to classical computers. The challenge with small, noisy quantum devices running quantum algorithms is that due to the hardware limitations they face which must be respected, quantum algorithm needs to be modified before their execution on quantum devices.

To solve this problem in quantum computers, a mapping process is needed, which is part of the compilation step that modifies the quantum algorithms to take into account all the limitations of the targeted quantum device. After mapping process, the resulting quantum circuit usually has a higher number of gates and (execution time) latency, decreasing the algorithm’s reliability. Different mapping solutions have been already proposed. Most of them are meant for a specific quantum processor and different in methodology, approach and features. In addition, they are usually only compared in terms of added gates (gate overhead) or added circuit execution time (latency overhead). No thorough comparative analysis of the different mapping solutions performance and features has been performed so far.

In this thesis, we apply structured Design Space Exploration to the mapping of quantum circuits. By doing so, we will be able to have a deeper understanding of this process, make a thorough comparison of the mapper approaches and even develop optimized solutions for given applications and quantum devices. Moreover, we apply Design Space Exploration to the generation of quantum algorithms used as benchmarks and start to explore profiling them in a structured approach.
ACKNOWLEDGEMENTS

I would like to thank the following people, without whom I would not have been able to complete this research, and without whom I would not have made it through my masters degree! The quantum computer architecture research group in Delft University of Technology, especially to my supervisor dear Carmen G. Almudéver and dear Eduard Alarcon my supervisor at UPC who introduced me to this group, whose both insight and knowledge and their consistent support and guidance during the running of this project into the subject matter steered me through this research to achieve promising results and the final thesis.

I would like to thank my parents who supported me for everything and believed me and are always there for me. Through any difficult path, feeling them by my side gives me the confidence and power to keep on and not give up. Only I know and will know how extraordinary you are. Thank you a lot.
CONTENTS

1 INTRODUCTION .......................................................... 1
  1.1 Motivation ......................................................... 1
  1.2 Problem Statement ............................................. 1
  1.3 Structure of the Thesis ....................................... 2

2 QUANTUM COMPUTING AND THE MAPPING PROBLEM ............ 5
  2.1 Quantum Computing ............................................. 5
    2.1.1 Fundamentals of quantum computation .................. 5
    2.1.2 Quantum Circuits ........................................ 6
  2.2 The Mapping Problem .......................................... 7
  2.3 The Qmap ....................................................... 8
    2.3.1 Qmap overview ........................................... 9
    2.3.2 Initial Placement ......................................... 10
    2.3.3 Router .................................................... 10
    2.3.4 RC-Scheduler ............................................. 12
  2.4 State-of-the-art on the mapping of quantum circuits ... 12

3 APPLYING DESIGN SPACE EXPLORATION (DSE) TO QMAP ....... 15
  3.1 Design Space Exploration (DSE) ............................ 15
  3.2 Applying Design Space Exploration (DSE) to Qmap ....... 16
  3.3 Analysis Framework ........................................... 17
    3.3.1 OpenQL .................................................. 17
    3.3.2 Framework of the Qmap ................................ 18
    3.3.3 Benchmarks .............................................. 18
    3.3.4 Analysis framework ..................................... 19

4 RESULTS ..................................................................... 23
  4.1 Selecting the Options of the Qmap as the input parameters ... 23
  4.2 DSE on the Qmap using Options as the input parameters and analyzing the performance metrics ... 23
  4.3 DSE on the benchmarks ........................................ 25
  4.4 DSE on the Qmap for the new synthetic benchmarks ....... 25
  4.5 Introducing new input parameters to DSE to analyze performance metrics ................................. 27
  4.6 Graph analysis on the Benchmarks ............................ 28
    4.6.1 Extracting graph metrics to introduce to DSE ........ 29
    4.6.2 Interconnecting the input parameters ................... 30
    4.6.3 DSE on the Qmap with the new graph parameters to analyse performance metrics .................. 32

5 CONCLUSIONS AND FUTURE WORK .............................. 37
  5.1 Conclusions ..................................................... 37
  5.2 Future work ...................................................... 37
LIST OF FIGURES

Figure 1.1 Full stack implementation of a quantum computer[1] . . . . 2
Figure 2.1 Bloch sphere ........................................................... 6
Figure 2.2 An example of quantum circuit ................................. 6
Figure 2.3 An example of gate decomposition. ............................... 7
Figure 2.4 Example of possible topologies ................................. 8
Figure 2.5 An example of mapping process to map a quantum circuit to a 2D topology quantum processor. ............................... 9
Figure 2.6 An overview of Qmap which is embedded in OpenQL [2] .... 10
Figure 3.1 Complexity ranking of Qmap options ........................... 17
Figure 3.2 A quantum algorithm described in high-level language in python using OpenQL library ................................. 18
Figure 3.3 The block diagram of the framework this thesis is using in order to do Design Space Exploration on the Qmap ................. 21
Figure 4.1 $G_{overhead}$ vs Config. 1-4 and $L_{overhead}$ vs Config. 1-4 and Figure of merit vs Config. 1-4 ............................... 24
Figure 4.2 3D representation of all the benchmarks in the design space with input parameters of number of qubits, gates, and CZ ratio. ............................... 26
Figure 4.3 3D representation of new benchmarks and existing benchmarks all together in the design space with input parameters of number of qubits, gates, and CZ ratio. ............................... 27
Figure 4.4 New benchmarks $G_{overhead}$ vs Config. 1-4 and $L_{overhead}$ vs Config. 1-4 and Figure of merit vs Config. 1-4 ............................... 28
Figure 4.5 4d representation of design space with input parameters of number of qubits, gates and CZ ratio vs. $G_{overhead}$ and $L_{overhead}$ ............................... 29
Figure 4.6 The interaction graph of two benchmarks with the same number of qubits, gates, and CZ ratio. ............................... 30
Figure 4.7 Analysis on the graph structure of the benchmarks. Representing the benchmarks graph parameters vs. their Complexity. ............................... 30
Figure 4.8 2D representations of Qubits, gates, CZ ratio vs Average shortest path with color code ............................... 31
Figure 4.9 2D representations of Qubits, gates, CZ ratio vs Average clustering ............................... 32
Figure 4.10 2D representation of Qubits, gates, CZ ratio and Average Node connectivity with color code ............................... 33
Figure 4.11 4D representation of Qubits, gates, CZ ratio and Average shortest path with color code. ............................... 33
Figure 4.12 4D representation of Average shortest path length, Average clustering, and Average Node connectivity vs $G_{overhead}$ and $L_{overhead}$ ............................... 34
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Quantum Gates</td>
<td>7</td>
</tr>
<tr>
<td>2.2</td>
<td>Part of the results of mapping quantum benchmarks to the Surface-17 processor</td>
<td>13</td>
</tr>
<tr>
<td>3.1</td>
<td>Benchmarks</td>
<td>20</td>
</tr>
<tr>
<td>4.1</td>
<td>The graph metrics and number of Qubit, gates, and CZ ratio for some of the existing Benchmarks. AS: Average shortest path length, AC: Average clustering, AN: Average Node connectivity</td>
<td>35</td>
</tr>
<tr>
<td>4.2</td>
<td>The graph metrics and number of Qubit, gates, and CZ ratio for some of the new Benchmarks. AS: Average shortest path length, AC: Average clustering, AN: Average Node connectivity</td>
<td>36</td>
</tr>
</tbody>
</table>
# Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>NISQ</td>
<td>Noisy Intermediate-Scale Quantum</td>
<td>1</td>
</tr>
<tr>
<td>NN</td>
<td>nearest-neighbour</td>
<td>2</td>
</tr>
<tr>
<td>ILP</td>
<td>Integer Linear Programming</td>
<td>10</td>
</tr>
<tr>
<td>DSE</td>
<td>Design Space Exploration</td>
<td>13</td>
</tr>
</tbody>
</table>
1 INTRODUCTION

1.1 MOTIVATION

Today, the processors used in mobile phones have reached a structure of 7 nanometers, and it is expected that by 2021, processors with 5 nanometer lithography will be released. Without exaggeration, these small processors are several times more powerful than the high-end desktop processors introduced in the 2000s. This is true not only for mobile processors but also for desktop and server processors that came with 10 and 7 nanometer lithography. Aside from complex issues, a more fundamental problem in the development and fabrication of chips is the shrinkage of transistors. Today, big companies like AMD and Apple, after much effort, have acquired the technology of making 7-nanometer chips. In structures below 10 nm, the dimensions of the transistors are closer to the atomic dimensions, in which case the laws of quantum physics become more pronounced and change the rules of the game. For example, an electron can tunnel from one transistor to another, which is a drawback for a computational chip.

According to Moore’s law, the number of complex problems which have been unsolvable has reduced since the 1960s, as the number of transistors increased, resulting in increased memory and computer speed. But there are still some problems that even the supercomputers that have been introduced to the world in recent years do not have the power and ability to solve. Apart from the classical solutions to problems, we can examine the approaches of quantum physics and introduce the science of computing into the quantum world.

That is how quantum computers arise in order to help us solve diverse problems that are not solvable by classical computers such as factoring large numbers and simulating quantum systems. Quantum computing is now in the Noisy Intermediate-Scale Quantum (NISQ) era in which small-scale prototypes of quantum computers already exist. Figure 1.1 shows a full-stack implementation of a quantum computer proposed by the quantum computer architecture research group in Delft University of Technology[1]. This layered full-stack allows to run quantum algorithms on a quantum device as followings: Quantum algorithms are expressed in high-level programming languages and then compiled to a set of instructions that are further translated to specific signals so they can operate on the physical qubits.

Current quantum devices face some constraints which should be respected while implementing these algorithms. These algorithms, expressed as quantum circuits, need to be modified to respect quantum processor constraints. This task is performed by the mapper which is part of the compiler.

This thesis focuses on the mapping of quantum circuits. More precisely, it proposes in-depth and structured analysis of mapping in quantum computers. It proposes to apply structured Design Space Exploration methodology to mapping process, so for the first time, we can have a complete, structured, and in-depth analysis of the mapping process. Moreover, this allows in turn to have a thorough comparison of different solutions which have been proposed for mapping process. In this thesis, we also propose to apply Design Space Exploration methodology to the benchmarks (quantum circuits) which are commonly used as data-sets to analyze mappers performance metrics.
Quantum computers and quantum algorithms have acquired a lot of interest within the past few years, principally driven by their ability to resolve specific tasks considerably quicker than classical algorithms. These quantum algorithms are usually described by quantum circuits, a sequence of gates that are applied to the qubits of a quantum processor. However, in order to execute these circuits on physical quantum devices they have to be modified, so that the physical constraints the quantum devices might have are respected. This is known as the mapping problem. One of the most important constraints of current quantum devices is the reduced connectivity between the qubits (e.g., qubits are usually aligned in a 1D or 2D array with only nearest-neighbour (NN) connectivity). To overcome this restriction, additional gates are added to the circuit. This increase in the number of gates affects the reliability of the quantum circuit. Therefore, the number of additional gates must be the least possible.

The process of modifying and adapting the quantum algorithms to meet the quantum chip requirements is known as mapping process. So far different, disparate and heterogeneous mappers have been proposed and most of them aim specific processors and vary in methodology, features and approach. Therefore, there is a need for analyzing in a structured systematic manner and comparing the performance of a mapper. Normally, mappers are compared either by the number of gates, and latency (overall execution time) they add to the circuit. Comparing the performance of mappers only using these two metrics is not a complete and thorough analysis, because their performances depend on the characteristics of the algorithms used as benchmarks and, of course, of the underlying physical constraints of the quantum computer. Therefore, in this thesis, we propose to use a structured design space exploration. It can also help us find new metrics and variables which makes it possible to find the optimum design point of a mapper. This methodology creates a multidimensional design space according to the variables and metrics that we introduce in order to carry out in-depth and structured analysis of performance. Therefore, we can analyze the design space and gain valuable information such as design trends, optimum point, complete comparison, and an exploratory foresight of incomplete areas for future designs (Gap analysis).

The structure of the thesis is the following:

In Chapter 2, we propose an introduction on quantum computing and then an overview on mapping. In the last part of this chapter the Qmap which is the target mapper to instantiate all the analysis in this thesis is introduced.
Chapter 3 starts with an introduction on Design Space Exploration methodology. Then, the ideas on how this methodology is applied are presented. Moreover, there is a brief explanation of the created framework module and the sub modules used in this thesis.

In Chapter 4 all the results are presented. In Chapter 5 we conclude the thesis and we explain possible future works after this thesis.
Quantum computers will be able to solve complex problems which are not solvable by classical computers. Quantum computers make use of quantum computation to solve intractable problems for classical computers. In this section, we learn about the fundamentals of quantum computing to see what makes quantum computers special.

2.1 Quantum Computing

Quantum computers will be able to solve complex problems which are not solvable by classical computers. Quantum computers make use of quantum computation to solve intractable problems for classical computers. In this section, we learn about the fundamentals of quantum computing to see what makes quantum computers special.

2.1.1 Fundamentals of quantum computation

Bit is the smallest unit of processing in classical information and can only have two constant values 0 or 1. Bits are processed by gates, many of which we are familiar with, such as AND, OR, NOT, XOR, and so on. A bit can only have one of the specified states 0 or 1 at a time. Now suppose we have a new entity instead of a bit that can have both values 0 and 1 at the same time. This entity is named quantum bit or qubit and it is the basic unity of information in quantum computers.

Because in the quantum world, quantum states (such as the spin state of an electron) are represented by vectors, qubits are a state vector in Hilbert’s two-dimensional space. Similar to a classical bit, qubits can have two basic states |0⟩ and |1⟩ in the form of |ψ⟩ = α|0⟩ + β|1⟩. Note that the coefficients α and β are the so-called probability amplitudes and generally complex numbers. So, the state of a qubit can be both |0⟩ and |1⟩. However, this superposition cannot be read. When reading (measuring) a qubit state, a binary value, 0 or 1 is biased more specifically.

The probability setting 0 or 1 is shown by |α|^2 and |β|^2, respectively. We know that the sum of the probabilities of a system or phenomenon must be 1, so this condition requires that |α|^2 + |β|^2 = 1. However that qubits can collapse to |0⟩ when measuring 0 and to |1⟩ when measuring a 1. This process destroys the superposition and therefore the qubit entanglement is lost.

For an intuitive understanding of the state of a qubit, a qubit state can be represented as a vector like |ψ⟩ = [α, β] and as the vector is 2-dimensional, complex, and unitary, by defining the coefficients α and β in the form of α = cos(θ/2) and β = e^(iφ)sin(θ/2), we can display the state of a qubit on the Bloch sphere (see Figure 2.1). In fact, a qubit can only be in the position of the Bloch sphere. A classical bit, on the other hand, has only two values, 0 and 1, on the poles of the Bloch sphere. Note that before measuring the system, the location of the qubit can be at any point on the surface of the Bloch sphere.

N qubits can be combined and its state can be represented as |ψ⟩ = α₀|0..0⟩ + α₁|0..1⟩ + ... + αᵣ|1..1⟩. These qubits are correlated and this property is called entanglement which is the power of quantum computers.

Quantum states can be changed by measuring them as we explained. However, they can also be modified by applying quantum gates that can act on a single qubit or there are also two qubit gates. These gates make it possible to do calculations. Quantum gates are unitary operations and they are reversible (not like many classical logic gates). They are depicted by unitary matrices. The most common quantum
gates operate on 1 or 2 qubits. This suggests that as matrices, quantum gates may be represented by $2 \times 2$ or more generally by $2^n \times 2^n$, where $n$ is the number of qubits that act on, with orthonormal rows. In Table 2.1 we can see the most common quantum gates. Single qubit gates can be represented as rotations in Bloch sphere. For instance, X gate is a rotation of 180° around x-axis and it changes the qubit state from $|0\rangle$ to $|1\rangle$ or the other way around. For two-qubit gates, we can mention CNOT and SWAP gates as examples which are also essential to the next chapters. The CNOT gate is a Controlled-NOT operation. Its execution on the target qubit depends on the state of the control qubit. If the control qubit is 1 and it operates as a X gate on the target qubit, if not the Controlled-NOT operation does not execute. And the SWAP gate exchanges the state of two qubits.

### 2.1.2 Quantum Circuits

Once we have qubit and quantum state, we can build quantum circuits. A quantum circuit is a set of operations executed coherently and concurrently on multiple qubits. It is in fact a computational routine. These operations can be universal quantum gates, measurements, or initialization. Any quantum program or quantum algorithm can be represented by quantum circuits. In Figure 2.2, an example of a quantum circuit is represented. In the first part, we see that a single qubit gate called Hadamard (H) is executed on q1 and then a two-qubit gate CNOT is executed in which the control qubit is q1 and the target qubit is q2. In step 2 q1 is measured. Concurrently, another CNOT is executed with the control of q2 and with the target q3. Then in step 4, a Hadamard gate is executed on q2.
### Table 2.1: Quantum Gates

<table>
<thead>
<tr>
<th>Operator</th>
<th>Gate(s)</th>
<th>Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pauli-X (X)</td>
<td><img src="image" alt="X gate" /></td>
<td>$\begin{bmatrix} 0 &amp; 1 \ 1 &amp; 0 \end{bmatrix}$</td>
</tr>
<tr>
<td>Pauli-Y (Y)</td>
<td><img src="image" alt="Y gate" /></td>
<td>$\begin{bmatrix} 0 &amp; i \ -i &amp; 0 \end{bmatrix}$</td>
</tr>
<tr>
<td>Pauli-Z (Z)</td>
<td><img src="image" alt="Z gate" /></td>
<td>$\begin{bmatrix} 1 &amp; 0 \ 0 &amp; -1 \end{bmatrix}$</td>
</tr>
<tr>
<td>Hadamard (H)</td>
<td><img src="image" alt="H gate" /></td>
<td>$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 &amp; 1 \ 1 &amp; -1 \end{bmatrix}$</td>
</tr>
<tr>
<td>Controlled-Not (CNOT)</td>
<td><img src="image" alt="CNOT gate" /></td>
<td>$\begin{bmatrix} 1 &amp; 0 &amp; 0 &amp; 0 \ 0 &amp; 1 &amp; 0 &amp; 0 \ 0 &amp; 0 &amp; 0 &amp; 1 \ 0 &amp; 0 &amp; 1 &amp; 0 \end{bmatrix}$</td>
</tr>
<tr>
<td>SWAP</td>
<td><img src="image" alt="SWAP gate" /></td>
<td>$\begin{bmatrix} 1 &amp; 0 &amp; 0 &amp; 0 \ 0 &amp; 0 &amp; 1 &amp; 0 \ 0 &amp; 1 &amp; 0 &amp; 0 \ 0 &amp; 0 &amp; 0 &amp; 1 \end{bmatrix}$</td>
</tr>
</tbody>
</table>

#### 2.2 THE MAPPING PROBLEM

When we want to run a quantum algorithm on a physical quantum device, we need to modify that algorithm so that the the constraints of the quantum processor are met. These constraints are different depending on the architecture of the quantum processor and the technology. Therefore, any quantum device has its own specific constraints that must be respected. One common constraint is that two-qubit gates must operate on adjacent qubits or so-called nearest-neighbours (NN), because for instance in superconducting quantum computers, the qubits are placed in a 2D topology and only adjacent qubits can interact. Moreover, the elementary gates that a processor support must be considered. Therefore, any gate not supported by the processor must be decomposed to elementary gates supported by the processor. Therefore these constraints make new requirements that for any algorithm a mapping process step is required. Generally, mappers consist of the following steps:

1. The mapper has to decompose the circuit gates to primitive gates which are supported by the quantum processor. In Figure 2.3 we can see an example of gates decomposition in which a Z gate which is not supported by the device, and it is decomposed into X and Y gates.

   ![Figure 2.3](image)

   **Figure 2.3:** An example of gate decomposition.

2. Next, the qubits in the circuit, also called virtual or logical qubits have to be mapped to physicals qubits (qubits in the quantum processor). This process is also known as initial-placement.
3. The mapper has to schedule all the gates so all the dependencies of the operations are respected.

4. Finally mapper must add gates in order to meet the constraints of the quantum processor. Like moving the qubits to a place, so the qubits are adjacent or the execution of that gate is possible. In Figure 2.4, two different and very common topologies, 2D and 1D array, are shown. The nodes represent the qubits and the links show whcih qubits can interact. Only the qubits which are connected by a line can interact and perform a two-qubit gate.

![Figure 2.4: Example of possible typologies (a) 2d array. (b) 1d array](image)

In Figure 2.5 we see an example of mapping a simple circuit to a 2D topology quantum processor. On the top of the figure, the topology of the processor is represented. Each link connecting the two qubits represents that these two qubits can interact. When executing the algorithm on this processor, each virtual qubit should be placed into a physical qubit. Then, when executing each two-qubit-gate, the qubits must be NN and this means that they should be connected. Therefore, while running the algorithm, if we reach a two-qubit-gate and the qubits are non-NN, they must be moved to places in which they can interact. In Figure 2.5, it is shown that first, each virtual qubit is placed in a physical qubit. The intital placement of this example is done by assigning virtual qubit one to physical qubit one and q2 to Q2 and so on. In this example there are two two-qubit-gates which their qubits are not NN, therefore, the mapper has to move them to make them adjacent. As we can see when we reach step 3 and step 4, the two-qubit gates between q2 and q3 and also q1 and q4 cannot be executed. Therefore, a swap gate between q4 and q3 can solve the problem. Adding the swap gate increased the number of quantum gates and consequently it increases circuit depth.

Moving the qubits to NN positions and make them adjacent does not have a single solution specially when the circuit size (number of gates and qubits) tends to be larger. However, for situations like that shortest path can be used. Then using different approaches differentiates between mappers. In general Mappers try to minimize the number of gates or the execution time or time steps added to the circuit known as gates overhead and latency or circuit depth overhead respectively. Normally, the performance of a mapper is examined using these two metrics.

Now days quantum processors are know as NISQ. Which refers to quantum processors consisting a few tens to hundred of noisy qubits with short coherent time and operations with errors. This means that increasing the number of gates (gate overhead) and the execution time (latency overhead) compromises the algorithm reliability. Therefore, the mapper must minimize these two metrics as much as possible.

2.3 THE QMAP

This thesis is based on the Qmap [2] mapper. In this section, this mapper is briefly explained . Further information can be found in [2] and [4].
Figure 2.5: An example of mapping process to map a quantum circuit to a 2D topology quantum processor.

2.3.1 Qmap overview

Qmap decomposes any quantum gate into the primitive gates supported by the processor. It starts with an initial placement of qubits and finds the set of movement operations to place non-NN qubits to adjacent positions when they have to interact. It yields a valid sequence of instructions with scheduled operations respecting not only the data dependencies but also the classical control constraints. Moreover, minimizing the gate and latency overhead for implementing the algorithm on NISQ processors is essential.

In Figure 2.6, an overview of Qmap mapping process is represented. Quantum algorithms are given to Qmap as inputs and also a configuration file consisting of primitive gates supported by the processor and their execution time, the processor topology, and classical control constraints of the chip. Classical constraint means that in quantum processors, qubits are controlled by classical electronic controls. However it is not scalable to dedicate one control to each qubit. Therefore, they share them. This limits the possibility of parallelism in some operations. Then these inputs go through the mapping process consisting three main steps: 1-Initial placement 2- Routing 3- RC-scheduler. Finally, the output file is an executable QASM code which can be executed on the target quantum processor.
2.3.2 Initial Placement

Initial placement is the process of mapping virtual qubits to physical qubits. Which means the qubits from the circuit will be mapped to the actual qubits of the quantum chip. In the initialization it is preferable to place highly interacting qubits next to each other such that less movement operations will be added for performing two-qubit gates. In Qmap there are two alternatives for this process. Either each qubit is mapped to the same qubit as in its quantum circuit or trying to find the placement in which the non-NN interactions are reduced. In Qmap, the solution for finding the best placement is named Integer Linear Programming (ILP). The initial placement options are the followings:

Inital Placement: Random (q1 – Q1, ..., qn – Qn) or ‘optimal’ (ILP)

2.3.3 Router

When a two-qubit gate is to be executed and the two qubits are non-NN, they have to be moved to adjacent positions. Routing refers to the task of finding a series of movement operations that enables the execution of two-qubit gates on a given processor topology and trying to chooses the path which need the least number of movements. The difference is there are different strategies for finding the shortest path. Some mappers try to find the shortest path in which the number of gates added (Gates overhead) is minimized and the others try to minimize the execution time overhead (Latency overhead) in which the path is selected based on how well the added movement operations interleave with other operations. In Qmap there are different routing strategies and the following three have been evaluated:

- **MinExtendRCRouter**: This router evaluates all movement sets by looking back to the previously mapped gates and interleaving each set of movements with those gates using an as-soon-as-possible (ASAP) scheduling policy. Then, it selects the one(s) which minimally extend(s) the circuit latency. That is, one of the shortest path is selected.

- **MinPathRouter**: It just randomly selects one of the movement sets that are generated as described above without evaluating them for their extension of the circuit latency or depth.

- **TrivialRouter**: When there is a non-NN two-qubit gate, only the first shortest path that is found, is taken. In addition, a single movement set is generated for it; the one moving the control qubit until it is near to the target.

There are more options in the router and using them we can enhance its performance. These options are:
1. **Maplookahead**: This option controls which gate we select to map first when the mapper reaches set of gates which are independent (e.g. not operating on the same qubit) and can be mapped to one time slot.
   - **no**: The mapper takes the gates to be mapped one by one from the circuit
   - **critical**: Gates that by definition do not need routing, are mapped first like single qubit quantum gates. Of the remaining (two-qubit) quantum gates the most critical gate is selected first, i.e. the one which mostly affects on the overhead of the circuit after being mapped
   - **noroutingfirst**: The mapper selects the two qubit quantum gates of which their qubits are **NN**. Then it chooses the critical gates. Afterwards, the single-qubit gates are selected.
   - **all**: As with **noroutingfirst**, but does not select the most critical one; instead, for all remaining (two qubit non-NN) gates generate variations and find the best from these according to the strategy of the router (e.g. less latency or gate overhead).

2. **Mappathselect**: When generating variations of shortest paths between two physical qubits these paths are different in the sense that considering a rectangle in which the two qubits are placed in the nodes and there are different paths which either pass through the square or some other pass the borders of this square.
   - **all**: select all possible variations
   - **borders**: only select those variations that correspond to the borders of the rectangle spanning between the two physical qubits.

3. **Mapselectmaxlevel**: This is the option to benefit from look-ahead strategy. The mapper not only maps to minimize overhead of the circuit, but also maps in order to be as compatible as possible for mapping the next set of gates, so all together can minimize the overall overhead of the circuit. These numbers indicate how many two-qubit gates from circuit ahead should be added to the evaluation process.
   - **Mapselectmaxlevel**: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, inf

4. **Mapselectmaxwidth**: How many gates we select from current mapping to evaluate mapping of the next set of gates. For instance minimum is, only strategies that has minimum extension will enter the look ahead process to be evaluated.
   - **Mapselectmaxwidth**: min, minplusone, minplushalf, min, minplusmin, all

5. **Maptiebreak**: When the mapper is to select the gate to map and the shortest path to make the corresponding qubits adjacent, it tries to select the variation in which the overhead is lower. But if multiple variations (the selected gate to be mapped and its corresponding shortest path) are considered to be equal regarding overhead then mapper selects the strategy:
   - **first**: select the first of the set
   - **last**: select the last of the set
   - **random**: select in a random way from the set
   - **critical**: In which the most critical gate is routed first.

6. **Mapusemoves**: The mapper can use MOVE instead of SWAP where possible when moving two qubits to adjacent positions. The difference between MOVE and SWAP gates is that sometimes instead of swapping the value of two qubits, when the number of physical qubits are higher than the virtual qubits, the qubit can be moved to positions which are not occupied by
any other qubit. The advantage is that a SWAP gate is decomposed to three CNOTs but a MOVE is decomposed to two CNOTs resulting in less overhead.

- No
- Yes

### 2.3.4 RC-Scheduler

After routing, the circuit adheres to the processor topology constraint for two-qubit interactions, and has been scheduled in an as-soon-as-possible (ASAP) way, taking the classical constraints into account only in the case of the MinExtendRC router. The RC-scheduler reschedules the routed circuit to achieve the shortest circuit latency and the highest instruction-level parallelism. It does this in an as-late-as-possible (ALAP) way to minimize the required life-time and thus the decoherence error of each qubit, while taking the resource constraints into account. The main option to control the scheduling is the following:

**Scheduler commute:** When the scheduler is to schedule set of gates, it can either schedule them one by one as the order they appeared in the quantum circuit, or it can commute them meaning that it can swap them if this process can result in less overhead.

- **no:** don’t allow two-qubit gates to commute (CZ/CNOT); they are kept in original circuit order and presented to the mapper in this order
- **yes:** allow commutation of two-qubit CZ/CNOT gates; e.g. when one later one is already nearest-neighbor, map it before an earlier one which is not nearest-neighbor

### 2.4 State-of-the-Art on the Mapping of Quantum Circuits

Different mappers have been proposed in order to modify the algorithms to make them meet the constraints of different quantum processors. Many of these Solutions focus on NISQ devices like IBM [5] or Rigetti [6] chips because they are accessible on the cloud. The mappers vary and are classified based on different characteristics. One of the most important constraints of current quantum devices is the limited connectivity between the qubits. Many quantum processors follow a 1D or 2D topologies with only NN interactions [7], [8], [9], [10], [11]. Or they may have other topologies [12], [13], [14], [15], [16]. There are also other constraints which originate from the classical control part, as the parallel control of all qubits is not feasible regarding scalability, as the classical controls must be shared by scaling up. This reduces the possibility of parallel execution of quantum gates [17], [18].

Exact approaches [8], [14], [19], are only feasible in small number of qubits and they are not scalable. Therefore, for large quantum circuits approximate solutions using heuristics are often used [20], [21], [16]. Some used strategies are (Mixed)Integer Linear Programming ((M)ILP) solvers [22], [2], [23], Satisfiability Modulo Theory (SMT) solvers [14], [24], [25], heuristic (search) algorithms [22], [26], [8], [27], [28], [29], [30], decision diagrams [31], or reinforcement learning [32], [33].

Most of the mappers as explained before try to minimize gates and latency overhead. However, there are recent mappers which try to optimize for circuit reliability [24]–[34], [35]. Some mappers use the strategy of look-back and consider the already scheduled and mapped gates when choosing the routing path [2] or look-ahead strategy [30] [28], [16], which also consider some of the future two-qubit gates when selecting the shortest path to route.
2.5 PROBLEM STATEMENT

In all the work on mapping proposed, the mappers try to optimize for either gates or latency overhead or reliability in recent works. Therefore, they compare the mapper performance based on their chosen metrics. In Table 2.2 some part of the results of mapping several quantum benchmarks to the Surface-17 processor are shown. In this case the performance of the mapper is evaluated using metrics like latency and number of gates after the quantum circuit is mapped to the target quantum processor. This is how currently mappers are analyzed. But what is missing and has never been done is in-depth and structured analysis of the mappers performance. Current comparisons are not complete and deep. They are only compared by reduced selection of benchmarks and limited metrics. Using Design Space Exploration (DSE) we can structure a complete design space to make a complete, structured and in depth comparison possible.

The second identified problem is that the current benchmarks to test and compare the mappers are not complete set. Using DSE we can make new benchmarks in a structured manner to fill the design space for more complete analysis.

<table>
<thead>
<tr>
<th></th>
<th>The trivial router</th>
<th>The minpath router</th>
<th>The minextendrc router</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Latency</td>
<td>Gates</td>
<td>CZs</td>
</tr>
<tr>
<td>alu bdd 288</td>
<td>335</td>
<td>393</td>
<td>113</td>
</tr>
<tr>
<td>alu v0 27</td>
<td>166</td>
<td>188</td>
<td>56</td>
</tr>
<tr>
<td>benstein vazirani</td>
<td>36</td>
<td>45</td>
<td>10</td>
</tr>
<tr>
<td>4gt12 v1 89</td>
<td>931</td>
<td>1191</td>
<td>346</td>
</tr>
<tr>
<td>4gt4 v0 72</td>
<td>1124</td>
<td>1416</td>
<td>413</td>
</tr>
</tbody>
</table>

Table 2.2: Part of the results of mapping quantum benchmarks to the Surface-17 processor.
3.1 DESIGN SPACE EXPLORATION (DSE)

Different methodologies are used to analyze the overall quality and optimality of system design, for example, in order to find the optimum or to compare the quality of different processes according to different metrics. One of these methodologies which is used in many fields is called DSE. This methodology has been used in microelectronics [36], computer architecture and various fields, and the results are promising.

DSE refers to the methodologically consistent activity of exploring design alternatives before implementation. The ability to work on the area of potential design candidates renders this methods the most compelling approach for structured optimization. DSE is helpful for many engineering tasks, together with fast prototyping, improvement, and system integration. The most challenge in DSE are from the sheer size of the planning space that has to be explored. Typically, an oversized system has millions, if not billions, of prospects, so enumerating each purpose in the design space is prohibitory [37].

DSE is a structured design approach based on inter-dependencies of design parameters, design variables and performance metrics of some system, used for optimizing it. The goal of DSE is to enhance some predefined performance metrics or a combination of them in the form of a cost function or figure of merit, by concurrently sweeping over a large range of all input variables (multidimensional sweep). Therefore, for applying DSE methodology we have to:

1. Outline and state the design space which is represented in terms of input variables and a range of values or design points that must be swept.

2. Choose the performance metrics that are represented as functions depending on the previous group of various input variables.

3. Select a global cost function as a figure of merit that consists of the combination and compression of the metric space encompassing all different performance metrics and permits to identify overall optimum points.

4. Model the inter-dependencies between performance metrics and input parameters and variables. This will be done by exploitation analytical models, computer-based simulation or experimental information which will be interpolated. Note that input variables will take continuous as well as distinct values or perhaps to be knobs that may be turned on and off in order to outline different application scenarios.

What we achieve using DSE are the followings:

* By performing a qualitative analysis, we can outline performance metrics and figure of merit and identify design trends across a wide-range design space.

* We can divide and partition the design space and outline different design areas. Therefore, optimum points for different areas can be identified.

* Finding the optimum design and related parameters.
• Perform gap analysis and trying to extrapolate design space to predict future design possibilities and their performance, to provide a compelling answer to the application-pull technology-push tension.

We hereby propose to use this methodology and thereby carry out a complete analysis on our target mapper which is the Qmap mapper in order to analyze the performance metrics of this mapper with its existing input parameters and even discover more input parameters. This analysis results can be a guideline for all other mappers to be able to do a full, in-depth and structured analysis. Moreover, by defining a new figure of merit encompassing all performance metrics, mappers can be compared in a more meaningful strategy.

3.2 APPLYING DESIGN SPACE EXPLORATION (DSE) TO QMAP

The primary goal of this thesis was to analyze the performance of the Qmap mapper in a structured manner. Some prior analysis has been done but not in-depth and definitely not in a structured manner. As the very first step, we started by identifying and studying the available alternative options in the mapper. The options of the mapper are to control over three main parts of the Qmap mapping process, Initial placement, Routing, and Scheduling. These options are already explained in The Qmap. We started studying these options and how they affect the resulting circuit overhead and tried to make scientific guesses about their impact. More precisely, we analyzed the Qmap mapper complexity in a structured manner and analyzed what was the resulting latency and gate overhead. This is illustrated in Figure 3.1. We defined the “configuration” which includes the selected options. Based on our expertise and knowledge on the subject, we ordered the options and their alternatives. For instance config2 is more complex than config1 and it is expected to perform better. Having applied 4 different configurations, we started the in-depth and structured analysis of Qmap performance. All the comparisons in previous works, were using the metrics introduced in Benchmarks. Therefore, in the first design space we create in order to explore performance, the input parameters are the Qmap options in which we sweep and try different alternatives by applying different configurations. The performance metrics of the design space are gate and latency overhead. Moreover, in our work, for the first time we defined a function as a figure of merit to analyze and compare the performance using a unique value in pursuit of future optimization. This new metric includes both gate and latency overhead and it is defined as $\frac{1}{G_{\text{overhead}} \times L_{\text{overhead}}}$. 

Next phase of the work focused on studying and analyzing the results from previous part and understanding the reasons behind the results. In this phase, we found out the benchmarks were not as diverse as expected like number of qubits, gates, and CZ ratio. To fill the gaps we created synthetic quantum circuits.

Having the new benchmarks, we did the same analysis as the first phase on the new benchmarks. We did the full analysis on the performance of mapper by sweeping configuration. After this complete analysis, we encountered some unclear aspects. As the results were not conclusive, a more in-depth profiling of the algorithms was made in which we looked at the qubits interaction graph. The intended interaction graph is a graph whereby the nodes are representing the qubits and its links are representing if those qubits share a two-qubit gate.

Therefore, we analyzed and compared all the benchmarks and we discovered that the graph structure of the new and existing benchmarks are qualitatively different. These ‘hidden’ variables that proxy the internal circuit structure should also be taken into account in the future when evaluating and for understating the performance of the mapper.
In this thesis we use OpenQL [38] as the framework. OpenQL is a high-level programming language designed by the quantum computer architecture research group in Delft University of Technology. With this framework we can describe quantum algorithms and compile them. This framework is an open-source library and accessible in C++ and Python. The Qmap [2] is embedded in OpenQL as one of target platforms. Using OpenQL and Qammp we do all the analysis in this thesis.

3.3 ANALYSIS FRAMEWORK

In this thesis we use OpenQL [38] as the framework. OpenQL is a high-level programming language designed by the quantum computer architecture research group in Delft University of Technology. With this framework we can describe quantum algorithms and compile them. This framework is an open-source library and accessible in C++ and Python. The Qmap [2] is embedded in OpenQL as one of target platforms. Using OpenQL and Qammp we do all the analysis in this thesis.

3.3.1 OpenQL

OpenQL is an open-source high-level quantum programming framework. In this framework we can describe quantum algorithms independent from the target platform and OpenQL is able to compile the algorithm and translate it to executable codes for different target platforms with superconducting or semiconducting qubits.

The OpenQL consists of different layers. First is the high-level programming interface. We can describe the target quantum algorithm as a program either in C++ or Python. The next layer transforms the quantum program to a quantum circuit which is optimized, scheduled and mapped to the target quantum physical processor. This process is in compliance with constraints of the target quantum device, such as limited connectivity. The next stage is responsible for producing a technology-independent Common Quantum Assembly code (cQASM) [39] which describes the final quantum circuit but still independent of some low-level hardware details like the architecture of instructions in the target quantum processor. Using this code we can simulate the circuit and analyze it and check its reliability. The last stage which is the lowest-level layer, compiles the QASM codes and translate them to an executable code considering all and very detailed configuration and constraints of the target quantum processor. This final code consists of low-level instructions executable on the target quantum processor. In Figure 3.2, an example of a quantum algorithm described in python is represented. As it is shown, the
OpenQSL is an accessible library and using this library we can describe the algorithm and compile it.

### 3.3.2 Framework of the Qmap

The Qmap is embedded in OpenQSL. Quantum circuits are written in OpenQSL in c++ or python and then compiled by OpenQSL. After the compilation, Qmap performs mapping on the quantum circuit using the configuring of quantum processor which contains the constraints of the target quantum processor like connectivity and number of qubits, the primary gates, how to decompose the gates, and the execution time of gate. After the compilation cQASM and eQASM [40] are generated. These files can be used to simulate and analyze the circuits after mapping in which we can access the circuit mapped, optimized, scheduled and in full compliance with the target processor. The target processor for the whole analysis in this thesis is the Surface-17 processor [42].

```python
from openql import openql as ql
import os
import argparse

def circuit(config_file, new_scheduler=yes, mapper=no, maptiebreak=no, scheduler_post179=no, scheduler_uniform=no, scheduler_commute=no, initialplacement=random, loglevel=LOG_WARNING):
    curdir = os.path.dirname(config_file)
    config_fn = os.path.join(curdir, config_file)
    numqubits = 1
    numcircuits = 1
    p = ql.Program("sort 254", platform, numqubits)
    k = ql.Kernel("cnot", 2, 0)
    k.gate("cnot", [0, 0])
    k.gate("cnot", [1, 0])
    k.gate("cnot", [0, 1])
    k.gate("cnot", [0, 0])
    k.gate("cnot", [0, 1])
    k.gate("cnot", [1, 1])
    k.add()  # gate: extra gate added

    p.add(k)
    p.set_sweep_points(sweep_points, numcircuits)
    p.set_option("optimise", optimise)
    p.set_option("scheduler", scheduler)
    p.set_option("mapper", mapper)
    p.set_option("initialplacement", initialplacement)
    p.set_option("loglevel", loglevel)
    p.set_option("initialplace", initialplace)
    p.set_option("mapper", mapper)
    p.set_option("optimizer", optimizer)
    p.set_option("scheduler", scheduler)
    p.set_option("maptiebreak", maptiebreak)
    p.set_option("schedulerpost179", scheduler_post179)
    p.set_option("scheduleruniform", scheduler_uniform)
    p.set_option("schedulercommute", scheduler_commute)
    p.set_option("mapper", mapper)
    p.set_option("mapper", mapper)
    p.set_option("mapper", mapper)
    p.set_option("mapper", mapper)

    p.compile()

    for q in range(numqubits):
        p.add_kernel(q)

    p.compile()
```

Figure 3.2: A quantum algorithm described in high-level language in python using OpenQSL library.

### 3.3.3 Benchmarks

Some of the benchmarks we use in this thesis and their characteristics are show in Table 3.1. These benchmarks are common algorithms and are normally used among and across mappers to test and analyze their performance. The benchmarks we use are accessible and described in python using OpenQSL library. The metrics shown in the Table 3.1 such as number of gates and latency have been already presented. Circuit depth is equivalent to the total number of time-steps for executing the circuit assuming each of the gates takes one time-step. Latency and gate duration are expressed in cycles. The existing available benchmarks are not diverse and biased.
especially in the percentage of CNOT gates in all number of gates. Therefore, we synthetically create an extended set of benchmarks in a structured manner based on DSE and add them to the existing benchmarks.

3.3.4 Analysis framework

In Figure 3.3, it is represented how we implemented a framework to carry out the steps we presented in order to apply DSE into the Qmap. The main part of this framework is a module using Qmap embedded in OpenQL which maps all the input benchmarks and controls the input options of the Qmap mapper, while it changes them according to each target configuration. Accordingly all the benchmarks are given as inputs to this module and they are all mapped for 4 different configurations and for 3 different routers. After each mapping process, a description file is created by the Qmap mapper in which the quantum circuit after being mapped is described. This file is processed by the framework module and the gate and latency of these quantum circuits are extracted so that the corresponding latency and gate overhead and the figure of merit values are stored in a database. At the last step, the figures are exported from this database in order to be analyzed.

The other main part of this framework is the module to carry out the analysis on the benchmarks. In this module, firstly all the benchmarks are analyzed based on the number of qubits, gates, and CZ ratio. After finding a gap in the design space, a framework is used to generate synthetic benchmarks which fill the gaps in a structured manner. This framework is a random quantum circuit generator in which the number of qubits, gates, and CZ ratio are controlled but how these gates are distributed among the qubits is generated by a random process. It is noticed that the results are not conclusive and the interaction graph might play an important role in the performance. Then, all the benchmarks are given to a block to be analyzed based on their interaction graph structure. From this analysis, new input parameters are discovered and thus introduced to the main module.

After receiving the new input parameters for each benchmark, the main module in the framework, using these new input parameters conducts a new full, structured analysis on all the benchmarks and exports the same metrics and stores them into the database from which the figures can be extracted.
<table>
<thead>
<tr>
<th>Benchmarks</th>
<th>Qubits</th>
<th>Gates</th>
<th>CNOTs</th>
<th>Depth</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>alu bdd 288</td>
<td>7</td>
<td>84</td>
<td>38</td>
<td>48</td>
<td>169</td>
</tr>
<tr>
<td>alu v0 27</td>
<td>5</td>
<td>36</td>
<td>17</td>
<td>21</td>
<td>72</td>
</tr>
<tr>
<td>benstein vazirani</td>
<td>16</td>
<td>35</td>
<td>1</td>
<td>5</td>
<td>40</td>
</tr>
<tr>
<td>4gt12 v1 89</td>
<td>6</td>
<td>228</td>
<td>100</td>
<td>130</td>
<td>448</td>
</tr>
<tr>
<td>4gt14 v0 72</td>
<td>6</td>
<td>258</td>
<td>113</td>
<td>137</td>
<td>478</td>
</tr>
<tr>
<td>4mod5 bdd 287</td>
<td>7</td>
<td>70</td>
<td>31</td>
<td>40</td>
<td>140</td>
</tr>
<tr>
<td>cm42a 207</td>
<td>14</td>
<td>1776</td>
<td>771</td>
<td>940</td>
<td>3249</td>
</tr>
<tr>
<td>cn13 5 180</td>
<td>16</td>
<td>485</td>
<td>215</td>
<td>207</td>
<td>729</td>
</tr>
<tr>
<td>cuccaroAdder 1b</td>
<td>4</td>
<td>73</td>
<td>17</td>
<td>25</td>
<td>58</td>
</tr>
<tr>
<td>cuccaroMultiply</td>
<td>6</td>
<td>176</td>
<td>32</td>
<td>55</td>
<td>133</td>
</tr>
<tr>
<td>decod24 bdd 294</td>
<td>6</td>
<td>73</td>
<td>32</td>
<td>40</td>
<td>143</td>
</tr>
<tr>
<td>decod24 enable</td>
<td>6</td>
<td>338</td>
<td>149</td>
<td>190</td>
<td>669</td>
</tr>
<tr>
<td>graycode6 47</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>20</td>
</tr>
<tr>
<td>ham3 102</td>
<td>3</td>
<td>20</td>
<td>11</td>
<td>11</td>
<td>41</td>
</tr>
<tr>
<td>miller 11</td>
<td>3</td>
<td>50</td>
<td>23</td>
<td>29</td>
<td>105</td>
</tr>
<tr>
<td>mini alu 167</td>
<td>5</td>
<td>288</td>
<td>126</td>
<td>162</td>
<td>564</td>
</tr>
<tr>
<td>mod5adder 127</td>
<td>6</td>
<td>555</td>
<td>239</td>
<td>302</td>
<td>1048</td>
</tr>
<tr>
<td>mod8 10 177</td>
<td>6</td>
<td>440</td>
<td>196</td>
<td>248</td>
<td>872</td>
</tr>
<tr>
<td>one two three</td>
<td>5</td>
<td>70</td>
<td>32</td>
<td>40</td>
<td>141</td>
</tr>
<tr>
<td>rd32 v0 66</td>
<td>4</td>
<td>34</td>
<td>16</td>
<td>18</td>
<td>66</td>
</tr>
<tr>
<td>rd53 311</td>
<td>13</td>
<td>275</td>
<td>124</td>
<td>124</td>
<td>441</td>
</tr>
<tr>
<td>rd73 140</td>
<td>10</td>
<td>230</td>
<td>104</td>
<td>92</td>
<td>330</td>
</tr>
<tr>
<td>rd84 142</td>
<td>15</td>
<td>343</td>
<td>154</td>
<td>110</td>
<td>394</td>
</tr>
<tr>
<td>sf 274</td>
<td>6</td>
<td>781</td>
<td>336</td>
<td>436</td>
<td>1516</td>
</tr>
<tr>
<td>shor 15</td>
<td>11</td>
<td>4792</td>
<td>1788</td>
<td>2268</td>
<td>7731</td>
</tr>
<tr>
<td>sqrt8 260</td>
<td>12</td>
<td>3009</td>
<td>1314</td>
<td>1659</td>
<td>5740</td>
</tr>
<tr>
<td>squar5 261</td>
<td>13</td>
<td>1993</td>
<td>869</td>
<td>1048</td>
<td>3644</td>
</tr>
<tr>
<td>sym6 145</td>
<td>7</td>
<td>3888</td>
<td>1701</td>
<td>2187</td>
<td>7615</td>
</tr>
<tr>
<td>sym9 146</td>
<td>12</td>
<td>328</td>
<td>148</td>
<td>127</td>
<td>450</td>
</tr>
<tr>
<td>sys6 v0 111</td>
<td>10</td>
<td>215</td>
<td>98</td>
<td>74</td>
<td>266</td>
</tr>
<tr>
<td>vbeAdder 2b</td>
<td>7</td>
<td>210</td>
<td>42</td>
<td>52</td>
<td>116</td>
</tr>
<tr>
<td>wim 266</td>
<td>11</td>
<td>986</td>
<td>427</td>
<td>514</td>
<td>1788</td>
</tr>
<tr>
<td>xor5 254</td>
<td>6</td>
<td>7</td>
<td>5</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>z4 268</td>
<td>11</td>
<td>3073</td>
<td>1343</td>
<td>1643</td>
<td>5688</td>
</tr>
<tr>
<td>adt4 197</td>
<td>13</td>
<td>3439</td>
<td>1498</td>
<td>1839</td>
<td>6377</td>
</tr>
<tr>
<td>cm152a 212</td>
<td>12</td>
<td>1221</td>
<td>532</td>
<td>684</td>
<td>2366</td>
</tr>
</tbody>
</table>

Table 3.1: Benchmarks
Figure 3.3: The block diagram of the framework this thesis is using in order to do Design Space Exploration on the Qmap.
4 RESULTS

4.1 SELECTING THE OPTIONS OF THE QMAP AS THE INPUT PARAMETERS

In section Applying Design Space Exploration(DSE) to Qmap, it is explained how all the options available in the Qmap mapper were ranked. Out of these options the ones which proved that can increase the gate and latency overhead significantly were selected. The selected options were combined and the other options were frozen mostly to the most basic alternative or in some cases to the best alternative regarding the overhead. With combination of these selected options different configurations were created. The performance was analyzed by sweeping the configuration. Additionally, to have a better representation and understanding in our figures, the Router type was not included as a configuration parameter, and the configuration was swept for different routers and the performance was analyzed. Finally, the results of the three routers are shown in one figure to gain more insights in the figures.

The selected options to combine and sweep are the following:

- maplookahead
- maptiebreak
- mapselectmaxlevel
- mapselectmaxwidth

With the alternatives for each of these options a large set of configurations can be created. However, after a lot of testing, it is understood that some of the alternatives in these options also do not affect the gate and latency overhead as expected, so some of these alternatives were removed. Moreover, mapselectmaxlevel and mapselectmaxwidth are only available for minextendrc router. Therefore, four main configurations consisting of maplookahead and maptiebreak using their alternatives were created. By sweeping the configuration, the idea is that the complexity of the mapping process is increased too.

4.2 DSE ON THE QMAP USING OPTIONS AS THE INPUT PARAMETERS AND ANALYZING THE PERFORMANCE METRICS

For this analysis three different figures were created and each is representing one of the metrics which represents the performance of the mapper. As a first attempt to use DSE methodologies to gain insight into the possible mapper solutions and features, several internal parameters of the Qmap mapper presented were swept in a structured way. Figures 4.1a and 4.1b show how the most common metrics used to assess the mapping procedure, namely the quantum gates overhead \( G_{\text{overhead}} = \frac{G_{\text{afterMapping}} - G_{\text{beforeMapping}}}{G_{\text{beforeMapping}}} \) and circuit latency overhead \( L_{\text{overhead}} = \frac{L_{\text{afterMapping}} - L_{\text{beforeMapping}}}{L_{\text{beforeMapping}}} \)
We can derive from these figures that increasing the complexity (more or improved features) of the mapper not always monotonically leads to lower gates/latency overhead. The MinPath router, that optimizes for number of operations and therefore it just takes one of the shortest path, and the MinextendRC, that chooses the routing path that minimally extends the circuit latency, show negligible difference, and when considering both aggregated metrics in the form of the figure of merit (Figure 4.1c), it is observed that the overall goodness of the solution slightly improves, fluctuations apart, when increasing the complexity.

For better understanding of the design space, we have decided to formulate the design space as an explicit function. This function has the input parameters of the design space as its input and the performance metrics as its output. As there are three metrics to analyze the performance, three different functions were created. For the first part of the work, as we only had one variable as the input parameter to the design space which is the configuration consisting the internal options of the Qmap mapper the formulations for the first part could be as the followings:

* \( G_{overhead} = f(\text{Config}) \)
* \( L_{overhead} = f(\text{Config}) \)
In this part, after analyzing the figures, the expected trends and changes in the performance were not spotted and the results were not conclusive as expected. After studying the figures, the idea was that the benchmarks may not be comprehensive regarding their number of qubits, gates, and CNOT ratio. The benchmarks may be biased to one part of the design space and are not as diverse as they should be.

\[ FOM = \frac{1}{c_{\text{overhead}} \times t_{\text{overhead}}} = f(\text{Config}) \]

4.3 DSE ON THE BENCHMARKS

The previous explorations were not conclusive. Therefore, we decided to look at the profiling of the quantum circuits used as the benchmarks. As the overhead created by the mapping process is affected by the percentage of two-qubit gates in the quantum circuit, the percentage of two-qubit gates for all the benchmarks were analyzed. The existing benchmarks have the CZ ratio between 40% to 50%. After realizing that the design space is not comprehensive on the data-set stemming from the available benchmarks. A new analysis was defined only on the benchmarks. To do so, firstly the benchmarks were studied, and it was understood that they are biased, at least in the CNOT gate ratio, so that for most of the benchmarks they tend to be between 40% to 50%. We used our expertise and knowledge on the subject and found out that number of qubits and gates and CNOT (CZ) ratio are the best candidates to be the new parameters for the design space to analyze the input benchmarks. Therefore, in Figure 4.2a all the benchmarks are shown in a design space with new input parameters, namely number of qubits, gates, and CZ ratio. In this representation, there is no analysis on the performance, however it is a gap analysis to discover in what part of the design space the data-set is not comprehensive. As it is shown in Figure 4.2a the benchmarks are biased to some specific area and are not sufficiently diverse.

The missing areas are easily detected. Therefore, the best methodology could be water filling these missing areas. As all the design space cannot be covered, the idea is to fill this area in a structured manner in order to be both feasible and comprehensive. Therefore, new benchmarks were created which were random circuits regarding the structure of the circuit. This randomness applies to how the gates are distributed among qubits, but with desired number of qubits, gates and CZ ratio swept in a structured manner in order to fill the gaps in the design space. They were accordingly created by the number of gates of 60, 100, 1000, 3000, 6000, 10000, 20000, 30000, and 50000. The with the number of qubits consisted of 5, 11, and 17. The CZ ratio on its turn was considered of 10%, 45%, and 80%. These new synthetic benchmarks are represented in Figure 4.2b. As it is shown, the new benchmarks are both comprehensive in covering the design space and well structured. In Figure 4.3, all the benchmarks including new and existing ones are represented in the design space proving the accomplishment of covering all the design space and reaching the point to claim that the analysis is well-posed and inspired by DSE to the data-set (benchmarks).

4.4 DSE ON THE QMAP FOR THE NEW SYNTHETIC BENCHMARKS

After accomplishing a comprehensive data-set, it is proposed to do the same analysis of the first part on the new benchmarks. Therefore, to analyze the Qmap mapper performance metrics, the configuration was swept and the performance metrics...
Figure 4.2: 3D representation of the design space with input parameters of number of qubits, gates, and CZ ratio for (a) all the existing benchmarks (b) new benchmarks.

were measured and analyzed. The formulation of the design space is the same as Section 4.2 as follows:

* \( G_{\text{overhead}} = f(\text{Config}) \)

* \( L_{\text{overhead}} = f(\text{Config}) \)

* \( \text{FOM} = \frac{1}{G_{\text{overhead}} \times L_{\text{overhead}}} = f(\text{Config}) \)

The \( G_{\text{overhead}} \), \( L_{\text{overhead}} \), and FOM (figure of merit) for the new benchmarks are shown in Figures 4.4a, 4.4b, and 4.4c respectively. The mean values of performance metrics of each configuration are connected with a line to clearly see the trends while sweeping configuration complexity.

The trends and the results and insights gained from the figures are quite similar to the first part analysis. But the main difference is that the mean values of performance metrics for each configuration are different from the same figures in Section 4.2 especially for \( G_{\text{overhead}} \). Therefore, a more in-depth analysis on all the benchmarks including the new and existing ones to discover the reason, in particular why the gate and latency overhead and figure of merit are different for existing and new synthetic benchmarks, is needed. Since the current representation of these figures misses essential parameters which can clear up the gray area on how the existing and synthetic benchmarks vary, so that these mean values are different. The best and first attempt was to introduce the new input parameters consisting of number of qubits, gates, and CZ ratio to the design space and carry out a structured and in-depth analysis of the performance metrics in the design space with the new input parameters.
For this part which is aimed to have a DSE with the new parameters introduced in Section 4.3, all the benchmarks are shown in the design space with the number of qubits, gates, and CZ ratio as the input parameters, while the related performance metrics are measured and analyzed in order to detect the reason of why the mean values are different, specially for $G_{\text{overhead}}$. In this part of the analysis, only $G_{\text{overhead}}$ and $L_{\text{overhead}}$ as the performance metrics are measured and analyzed, since the FOM mean values for each configuration were quite similar for the existing and new synthetic benchmarks. For the new figures after introducing the new parameters, namely the number of qubits, gates and CZ ratio, the formulation of performance metrics would be the following:

\[ G_{\text{overhead}} = f(\text{Config}, Q, G, CZ) \]
\[ L_{\text{overhead}} = f(\text{Config}, Q, G, CZ) \]

The design space now includes 4 input parameters and the outcome performance metrics. Therefore, to visualize the full design space a 5D presentation would be needed, but to inspect and assess performance we would only be able to represent 4D data consisting of 3 dimensions and one dimension with color code. Therefore, in the design space, the configuration is frozen to the 4th configuration and the router type to the “minextendrc”, which is ultimately the best configuration regarding the performance metrics. Since this point on-wards, all the figures are frozen to this configuration, and other parameters will be swept. The formulation of the new figures would be modified as the followings:

\[ G_{\text{overhead}} = f(\text{Config}, Q, G, CZ)|_{\text{config}=4} \]
\[ L_{\text{overhead}} = f(\text{Config}, Q, G, CZ)|_{\text{config}=4} \]

In Figures 4.5a and 4.5b, all the benchmarks are represented in a 4d design space. The three dimensions are the input parameters including number of qubits, gates and CZ ratio. The performance metrics shown by color code are $G_{\text{overhead}}$ and $L_{\text{overhead}}$ respectively. The new synthetic benchmarks are shown in circular points and the existing benchmarks are shown in square points.

The conclusions we can draw from Figures 4.5a and 4.5b could be the following. As we expected, by increasing the input metrics, which means increasing the size of the quantum circuit, the Gate and Latency overhead increase and this increase
Figure 4.4: New benchmarks (a) $G_{\text{overhead}}$ vs Config. 1-4 (b) $L_{\text{overhead}}$ vs Config. 1-4 (c) Figure of merit vs Config. 1-4.

is far stronger when increasing CZ ratio. Moreover, $G_{\text{overhead}}$ and $L_{\text{overhead}}$ of existing benchmarks are higher than $G_{\text{overhead}}$ and $L_{\text{overhead}}$ of the new benchmarks. This difference is stronger in $G_{\text{overhead}}$. The most important conclusion from Figures 4.5a and 4.5b is that in the same area of the design space for which the input parameters including number of qubits, gates and CZ ratio are approximately equal, the performance metric values are different and are higher for the existing benchmarks. Therefore, we realized that the design space may not be as complete and it is missing other input parameters. The first idea was that the complexity of the benchmarks with the same number of qubits, gates and CZ ratio might be different and that results in different $G_{\text{overhead}}$ and $L_{\text{overhead}}$.

4.6 GRAPH ANALYSIS ON THE BENCHMARKS

After noticing the difference in $G_{\text{overhead}}$ and $L_{\text{overhead}}$ in benchmarks with approximately equal input parameters, the idea was to investigate their complexity. The best options to do so is to analyze their graph structures. Therefore, the interaction graph of each benchmark is created. As explained in previous chapters, adding two qubit gates increase the complexity of a quantum circuit. When the two-qubit gates are non-NN, the mapper has to add MOVE and SWAP gates to route the quantum circuit and make them NN so they can interact. The two qubit gates in the quantum circuits (benchmarks) are CNOT gates and it is also one of the input parameters, which is CZ ratio. Therefore, the intended interaction graph is a graph with its nodes representing the qubits and its links representing if those qubits share a two-qubit gate. The weights on the links show how many two-qubit gates each pair of
qubits share. In Figure 4.6 the interaction graph of two benchmarks are shown, one from the existing and one from the new synthetic benchmarks. As it can be noticed, the insight on the complexity difference in the benchmarks which have equal values in input parameters but are very different in performance metric, seems to be correct. The structure of the graphs are quite different. Therefore, to be able to analyze these two graphs, new parameters which can differentiate these two graphs structures should be extracted.

### 4.6.1 Extracting graph metrics to introduce to DSE

By resorting to graph theory, some parameters which are indicative of the complexity of a graph are considered. Therefore, using these parameters, the benchmark structures could be compared more in-depth. Moreover, these new benchmarks seem to be new input parameters which could be introduced to the design space. The parameters which are proposed are the following:

- **Average shortest path length**: The mean value of lengths of all shortest paths between all the nodes.

- **Average clustering**: The average clustering is the mean value of clustering coefficients of all nodes. The cluster coefficient shows how the neighbour nodes of a node are connected. It is a value between 0 to 1. The higher the connectivity, the higher the average clustering.
Figure 4.6: The interaction graph of two benchmarks with the same number of qubits, gates, and CZ ratio. One from the existing and one from the new benchmarks. (a) Existing benchmark. (b) New benchmark.

- Node connectivity: It shows in average, how many neighbour nodes each node has. In this work this value is also a mean value between 0 to 1. 0 means no connection between the nodes and 1 shows a full mesh.

In figure 4.7, the benchmarks are sorted in terms of their complexity meaning that benchmarks which have less diversity of interaction are less complex, and vice versa. The graph metrics tend to be 1 as the complexity of the graph increases. These results prove major difference in graph parameters of the benchmarks showing that number of qubits, gates and CZ ratio are not the only parameters which discriminate the benchmarks. Therefore, the benchmarks can also be profiled regarding to their graph structure and hence global graph parameters.

Figure 4.7: Analysis on the graph structure of the benchmarks. Representing the benchmarks graph parameters vs. their Complexity.

4.6.2 Interconnecting the input parameters

Before introducing these new parameters into the design space, as these new parameters seem to be intermediate parameters, meaning that they are internal parameters of the benchmarks, not the input constituent variables such as number of
qubits, gates and CZ ratio, before analyzing the performance by introducing these graph metrics as input parameters to the design space, we assess the interplay and relationship between the primary parameters which are number of qubits, gates and CZ ratio and the new graph parameters including AS (Average shortest path length), AC (Average clustering), AN (Average Node connectivity). In Tables 4.1 and 4.2 we see the existing benchmarks and new benchmarks respectively, showing all the parameters. In Figures 4.8, 4.9, and 4.10 it is shown the 2D representation of Qubits, Gates and CZ ratio versus the new parameters extracted from graph metrics. In these figures, it can be concluded that the existing benchmarks plus the synthetic benchmarks are homogeneous considering all the parameters. In Figures 4.11a, 4.11b, and 4.11c the 4d representations of qubits, gates, and CZ ratio with each of graph metrics are shown. The existing benchmarks are shown with square points and the new benchmarks are shown with circular points. We can clearly see that the new benchmarks tend to have higher values of average shortest path length, average clustering, and average Node connectivity. The conclusion can be drawn that these graph metrics are clearly effective and this difference in the graph metrics between existing and new benchmarks is one of the reasons of why the $G_{overhead}$ and $L_{overhead}$ are different even when number of qubits, gates, and CZ ratio are approximately equal. But when representing the benchmarks to see the relation between the primary parameters including number of qubits, gates, and CZ ratio and the new internal parameters consisting of average shortest path length, average clustering, and average Node connectivity in Figures 4.11a, 4.11b, and 4.11c, the results are not fully conclusive.

![Graph Analysis on the Benchmarks](image.png)

**Figure 4.8:** 2D representations of Qubits, gates, CZ ratio vs Average shortest path with color code. (a) Qubit vs Average shortest path (b) Gates vs Average shortest path (c) CZ ratio vs Average shortest path.
Figure 4.9: 2D representations of Qubits, gates, CZ ratio vs Average clustering. (a) Qubit vs Average clustering. (b) Gates vs Average clustering (c) CZ ratio vs Average clustering.

4.6.3 DSE on the Qmap with the new graph parameters to analyse performance metrics

After discovering the new graph metrics, we would like to see how the performance metrics behave when sweeping these parameters. Therefore, these new metrics are introduced to the design space and obtained accordingly as resulting. The formulation of the new design space will be the following:

\[ G_{\text{overhead}} = f(\text{Config}, Q, G, CZ, AS, AC, AN) |_{\text{Config}=4} \]

\[ L_{\text{overhead}} = f(\text{Config}, Q, G, CZ, AS, AC, AN) |_{\text{Config}=4} \]

In this representation it is not included the number of qubits, gates, and CZ ratio. In Figures 4.12a and 4.12b the 4D representations of graph metrics and \( G_{\text{overhead}} \) and \( L_{\text{overhead}} \) are shown.
Figure 4.10: 2D representation of Qubits, gates, CZ ratio and Average Node connectivity with color code. (a) Qubits vs Average Node connectivity (b) Gates vs Average Node connectivity (c) CZ ratio vs Average Node connectivity.

Figure 4.11: 4D representation of Qubits, gates, CZ ratio and (a) Average shortest path with color code. (b) Average clustering with color code. (c) Average Node connectivity with color code.
Figure 4.12: 4D representation of Average shortest path length, Average clustering, and Average Node connectivity and (a) $G_{overhead}$ with color code. (b) $L_{overhead}$ with color code.
The graph metrics and number of Qubit, gates, and CZ ratio for some of the existing Benchmarks. AS: Average shortest path length, AC: Average clustering, AN: Average Node connectivity.
<table>
<thead>
<tr>
<th>AS</th>
<th>AC</th>
<th>AN</th>
<th>Qubit</th>
<th>CZ ratio</th>
<th>Gates</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.8</td>
<td>0</td>
<td>0.25</td>
<td>5</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>1.1</td>
<td>0.9</td>
<td>0.75</td>
<td>5</td>
<td>0.35</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>0.816666667</td>
<td></td>
</tr>
<tr>
<td>1.3</td>
<td>0.766666667</td>
<td>0.5</td>
<td>5</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>0.48</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>0.81</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>0.099</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>0.802</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>0.468</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>0.100666667</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>0.784666667</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>0.451333333</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>0.103666667</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>0.796666667</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>0.4575</td>
<td></td>
</tr>
<tr>
<td>1.654545455</td>
<td>0.284848485</td>
<td>0.2</td>
<td>11</td>
<td>0.466666667</td>
<td></td>
</tr>
<tr>
<td>1.418181818</td>
<td>0.649494949</td>
<td>0.2</td>
<td>11</td>
<td>0.783333333</td>
<td></td>
</tr>
<tr>
<td>1.254545455</td>
<td>0.764574315</td>
<td>0.5</td>
<td>11</td>
<td>0.82</td>
<td></td>
</tr>
<tr>
<td>1.490909091</td>
<td>0.529220779</td>
<td>0.3</td>
<td>11</td>
<td>0.43</td>
<td></td>
</tr>
<tr>
<td>1.236363636</td>
<td>0.733333333</td>
<td>0.6</td>
<td>11</td>
<td>0.091</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>11</td>
<td>0.453</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>11</td>
<td>0.808</td>
<td></td>
</tr>
<tr>
<td>1.018181818</td>
<td>0.981818182</td>
<td>0.9</td>
<td>11</td>
<td>0.095</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>11</td>
<td>0.796</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>11</td>
<td>0.460666667</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>11</td>
<td>0.098166667</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>11</td>
<td>0.806</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>11</td>
<td>0.447333333</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.2: The graph metrics and number of Qubit, gates, and CZ ratio for some of the new Benchmarks. AS: Average shortest path length, AC: Average clustering, AN: Average Node connectivity
In this thesis, a comprehensive, structured, and in-depth analysis of Qmap mapper performance has been proposed and carried out. This analysis has been performed to evaluate performance metrics through the Design Space Exploration methodology. Common performance metrics include gates and latency overhead. However, it was discovered that these metrics are not enough to assess performance and, most importantly, not enough to compare the performance of different mappers. Therefore, a thorough analysis has been conducted on the input benchmarks by sweeping internal parameters of the Qmap. All the performance metrics have been measured for all the benchmarks and for different configuration of the Qmap mapper. Studying the results, as they have not been conclusive, a new structured and in-depth analysis has been performed on the input data-set of the so-called benchmarks. This analysis proved that, firstly, the benchmarks on which the map is tested are not complete and do not fill the design space. Therefore, the design space has been filled with new benchmarks that were synthetically created in a structured manner stemming from a parametrized random circuit generator. It was also understood that the input parameters of the mapper design space are not limited to its options. But also it is possible to introduce the variables of a quantum circuit such as the number of qubits, gates, and CZ ratio as input parameters to the design space in order to analyze performance. But after careful analysis of a complete set of benchmarks that fill the design space in a structured manner, we realized that the design space parameters are not limited to current variables.

After deeper research, it was extracted that the inner graph structure of a quantum circuit can remarkably affect its performance and this aspect is hence indispensable for a fair in-depth algorithm-dependent quantum circuit profiling. By carefully examining the structure of graphs of different quantum circuits, it was discovered that the differences between the structures of circuits with similar number of qubits and gates, CZ ratio. This difference led to a difference in their performance. We were able to introduce the variables which have been found from the graph structure into the mapping design space. A structured and in-depth analysis was done this time with the new parameters. This analysis has shown that the structure of a quantum graph has a significant effect on performance, but the variables which have been found are not enough and this field needs further research because it can be said that these variables are not completely independent of each other and are affected by the variables that have not been found by the study in this thesis.

In conclusion, it can be said that this thesis discovered the lack of a correct, in-depth and structured analysis in mapping of quantum circuit. With some research, it was able to introduce new variables for performance analysis for the first time. Moreover, new profiling methodology on the benchmarks regarding their graph structure has been introduce in this thesis.

5.2 FUTURE WORK

The Design Space Exploration Figure of merit, which is a cost function specifically defined in this thesis to assess the overall performance of a given mapping tech-
unique for a Quantum Computer, used to analyze performance can be studied more in depth. Since in this study both the gate and latency overhead which are the performance metrics have the same weight, this is not necessarily the case and further flexibility of the trade-off metrics space can be further studied. Since the mappers try to optimize for gate and latency overhead in order to gain reliability it implies that reducing number of gates and the execution, due to the existing errors in quantum devices, results in higher fidelity of the quantum circuit executing on the targeted quantum device. Therefore, depending on the targeted quantum device the effective weights of gate and latency overhead in the definition of figure of merit might differ. This can be obtained by correlating the latency and gate overhead upon the fidelity of quantum circuits after mapping process.

As explained in Section 2.3.3, there are different strategies to find the shortest path to move qubits to adjacent positions in order to execute two-qubit gates. These strategies optimize either for gate or latency overhead. The Qmap mapper has an internal cost function similar to the figure of merit, so that when selecting the shortest path, if the mapper optimizes for gate overhead then latency has no weight in this cost function and vice versa. However, if the correct form figure of merit is extracted from this approach which is fully correlated with fidelity, then this function can be used to find the shortest path in which the fidelity is optimum. This could constitute the basis for the ultimate optimal routing strategy in terms of overall performance.

This thesis ended its work in the graph study stage. We think this is a very promising field to study. Because in our research it showed that very valuable information can be obtained by studying the graph structure of quantum circuits and its effect on their performance in mapping. Also our idea is that with this research we can reach more specific and precise variables whose impact on performance is clearly visible. Using this valuable information, we can provide feedback to re-design the mapper that can uniquely customize mapping solutions for each quantum circuit. This means that the mapper options are specifically set for the input quantum circuit. Therefore, before performing the mapping process, the input circuit can be analyzed, then the mapping process is adjusted based on the desired variables and the mapping process is performed, which would pave the way to future domain-specific adaptive optimal mapping techniques, specially suited to the stringent specifications of the mid-term NISQ era.


“https://www.rigetti.com/forest.”


