UNIVERSIDADE ESTADUAL PAULISTA “JÚLIO DE MESQUITA FILHO”

FACULDADE DE ENGENHARIA

CAMPUS DE ILHA SOLTEIRA

ALEXANDRE ARAUJO AMARAL DE ALMEIDA

Methodology for Mapping Quantum and Reversible Circuits to IBM Q Architectures

Metodologia para o Mapeamento de Circuitos Quânticos e Reversíveis para Arquiteturas IBM Q

Ilha Solteira

2019

ALEXANDRE ARAUJO AMARAL DE ALMEIDA

Methodology for Mapping Quantum and Reversible Circuits to IBM Q Architectures
Metodologia para o Mapeamento de Circuitos Quânticos e Reversíveis para Arquiteturas IBM Q

Thesis presented to the Faculdade de Engenharia – UNESP – Campus de Ilha Solteira as part of the requirements to obtain the PhD title in Electrical Engineering.
Knowledge Area: Automation.

Prof. Dr. Alexandre César Rodrigues da Silva
Supervisor

Prof. Dr. Gerhard W. Dueck
Co-Supervisor

Ilha Solteira
2019

FICHA CATALOGRÁFICA
Desenvolvido pelo Serviço Técnico de Biblioteca e Documentação

A447m

Almeida, Alexandre Araujo Amaral de.
Methodology for mapping quantum and reversible circuits to IBM Q architectures / Alexandre Araujo Amaral de Almeida. -- Ilha Solteira: [s.n.], 2019
132 f. : il.

Tese (doutorado) - Universidade Estadual Paulista. Faculdade de Engenharia de Ilha Solteira. Área de conhecimento: Automação, 2019

Orientador: Alexandre César Rodrigues da Silva
Coorientador: Gerhard W. Dueck
Inclui bibliografia

1. IBM Q. 2. Quantum computation. 3. Linear programming. 4. Reversible logic.

Raiane da Silva Santos
Supervisora Técnica de Seção
Seção Técnica de Referência, Atendimento ao usuário e Documentação
Diretoria Técnica de Biblioteca e Documentação
CRB/8 - 9999

UNIVERSIDADE ESTADUAL PAULISTA

Câmpus de Ilha Solteira

CERTIFICADO DE APROVAÇÃO

TÍTULO DA TESE: Methodology for Mapping Quantum and Reversible Circuits to IBM Q Architectures

AUTOR: ALEXANDRE ARAUJO AMARAL DE ALMEIDA
ORIENTADOR: ALEXANDRE CESAR RODRIGUES DA SILVA

Aprovado como parte das exigências para obtenção do Título de Doutor em ENGENHARIA ELÉTRICA, área: Automação pela Comissão Examinadora:

Prof. Dr. ALEXANDRE CESAR RODRIGUES DA SILVA
Departamento de Engenharia Elétrica / Faculdade de Engenharia de Ilha Solteira

Prof. Dr. RUBEN AUGUSTO ROMERO LAZARO
Departamento de Engenharia Elétrica / Faculdade de Engenharia de Ilha Solteira

Prof. Dr. EDSON DONIZETE DE CARVALHO
Departamento de Matemática / Faculdade de Engenharia de Ilha Solteira

Prof. Dr. FRANKLIN DE LIMA MARQUEZINO
Núcleo Multidisciplinar de Pesquisas em Computação / Universidade Federal do Rio de Janeiro

Prof. Dr. FERNANDO MAGNO QUINTÃO PEREIRA
Departamento de Ciência da Computação / Universidade Federal de Minas Gerais

Ilha Solteira, 23 de agosto de 2019

ACKNOWLEDGMENTS

I thank God for giving me health over the years.

I thank my wife Graziela and my daughter Sofia for the company and moments of joy.

I thank my parents, Antonio and Marileide, my brother, Alessandro and his wife, Camila, my father-in-law and mother-in-law, Antonio and Ivone, for all their support.

Thanks to my supervisor Prof. Alexandre for all the teachings and advice during this time working together.

Thanks to Prof. Gerhard for the opportunity to have visited and worked at the University of New Brunswick, Canada.

Thanks to the laboratory members for all the moments of work and relaxation.

I thank all those who, directly or indirectly, contributed to the development of this work.

I thank the Graduate Program in Electrical Engineering of Department of Electrical Engineering of Ilha Solteira, São Paulo State University (Unesp).

I am grateful for the support of the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Financing Code 001 and PDSE process n° 88881.189547/2018-01.

ABSTRACT

Research in the field of quantum circuits has increased as technology advances in the development of quantum computers. IBM offers access to quantum computers via the cloud service called IBM Q. However, these architectures have some restrictions regarding the types of quantum gates that can be realized. This work proposes a methodology for the mapping of quantum and reversible circuits to the architectures made available by the IBM Q project. The methodology consists in finding CNOT mappings using a set of defined qubits movements to satisfy the architectures constraints by adding as few gates as possible. In order to reduce the number of CNOT gates needing mapping, the permutation of the circuit can be changed. One alternative to find this permutation is trough exhaustive search. However, is not feasible as the number of qubit increases. To solve this problem, the permutation problem was formulated as an Integer Linear Programming problem. The mapping of quantum circuits realized with non-implementable gates and reversible Toffoli circuits to the IBM quantum architectures were proposed in this work as well. This was done by adapting the developed CNOT mappings along with the Integer Linear Programming formulation. The proposed methodology was evaluated by mapping quantum and reversible circuits to an IBM quantum architectures with 5 and 16 qubits. The results were compared with two algorithms that map quantum circuits to IBM architectures. The cost metric used in the evaluation were the number of quantum gates and the number of levels (depth) of the circuits. Experimental results have shown that the proposed methodology outperforms the other approaches regarding the circuits' size. The results of the methodology used to implement quantum (with non-implementable gates) and Toffoli circuits to IBM architectures have shown that the mapping targeting a specific architecture constraints results in a smaller mapped circuit. The methodology developed can be easily adapted to any quantum architecture. Also, the proposed Integer Linear Programming formulation can be used in any reversible circuit and can be applied in other research areas.

Keywords: IBM Q. Quantum computation. Linear programming. Reversible logic.

RESUMO

Pesquisa no campo de circuitos quânticos tem alavancado conforme a tecnologia avança no desenvolvimento de computadores quânticos. Atualmente, a IBM oferece acesso a computadores quânticos através do serviço em nuvem chamado IBM Q. No entanto, essas arquiteturas têm algumas restrições com relação aos tipos de portas quânticas e qubits em que uma porta CNOT pode ser implementada. Neste trabalho foi proposta uma metodologia para o mapeamento de circuitos quânticos e reversíveis para as arquiteturas disponibilizadas pelo projeto IBM Q. A metodologia consiste em mapear as portas CNOT utilizando uma série de movimentos de qubits, mantendo a permutação do circuito inalterada. A fim de reduzir o número de portas CNOT não implementáveis, a permutação do circuito pode ser alterada. Uma alternativa para encontrar essa permutação é a busca exaustiva. No entanto, é inviável conforme o número de qubits aumenta. Para resolver este problema, o problema de permutação foi formulado como um problema de Programação Linear Inteira. Como a metodologia é facilmente adaptável, o mapeamento de circuitos quânticos utilizando portas quânticas não implementáveis e circuitos reversíveis Toffoli também foram propostas neste trabalho. A avaliação da metodologia proposta foi feita com a realização do mapeamento de circuitos quânticos e reversíveis para arquiteturas quânticas com 5 e 16 qubits. Os resultados foram comparados com dois algoritmos que mapeiam circuitos quânticos para arquiteturas IBM. A métrica de custo utilizada na avaliação foram o número portas quânticas e o número de níveis do circuito. Os resultados experimentais mostraram que a metodologia proposta obteve melhor desempenho comparado com as outras abordagens, mesmo mantendo a permutação inalterada. Com relação à implementação dos circuitos quânticos não implementáveis e circuitos Toffoli para arquiteturas da IBM, os resultados mostraram que o mapeamento visando uma arquitetura específica resulta em circuitos mais otimizados. Uma outra característica da metodologia é que pode ser adaptada para qualquer arquitetura quântica. E também, a Programação Linear Inteira pode ser utilizada em qualquer tipo de circuito reversível, podendo ser aplicada em outras áreas de pesquisa.

Palavras-chave: IBM Q. Computação quântica. Programação linear. Lógica reversível.

LIST OF FIGURES

Figure 1 – Reversible gates. 28

Figure 2 – Example of a reversible circuit with three variables. 29

Figure 3 – Circuit generated with MDM algorithm. 30

Figure 4 – Reversible gates equivalences. 31

Figure 5 – Representation of the classical states in *Dirac* notation. 31

Figure 6 – Bloch sphere. 33

Figure 7 – Example of a quantum circuit realized with Clifford+T gates. 36

Figure 8 – Identity equivalences of quantum gates. 36

Figure 9 – Optimization equivalences for quantum gates. 37

Figure 10 – CNOT gates movements. 37

Figure 11 – IBM QX4 and IBM QX5 architectures from project IBM Q. 40

Figure 12 – SWAP gate. 41

Figure 13 – Swapping control and target qubits of a CNOT. 41

Figure 14 – SWAP gate with changed CNOT. 42

Figure 15 – Possible CNOT(Q_0, Q_2) mapping. 42

Figure 16 – Movement of control qubit using the CNOT(a, b). 43

Figure 17 – Movement of control qubit using the CNOT(b, a). 43

Figure 18 – Movement of target qubit using the CNOT(a, b). 44

Figure 19 – Movement of target qubit using the CNOT(b, a). 45

Figure 20 – CNOT3(a, b, c) mapping. 45

Figure 21 – CNOT3(a, b, c) mapping expanded. 46

Figure 22 – CNOT3†(a, b, c) mapping. 46

Figure 23 – Path $\{Q_0, Q_{15}, Q_{14}, Q_3\}$ to map CNOT(Q_0, Q_3). 48

Figure 24 – Path $\{Q_0, Q_{15}, Q_2, Q_3\}$ to map CNOT(Q_0, Q_3). 49

Figure 25 – Path $\{Q_0, Q_1, Q_2, Q_3\}$ to map CNOT(Q_0, Q_3). 51

Figure 26 – IBM QX4 architecture cost matrix. 53

Figure 27 – Example circuit for the application of the CNOT mappings. 54

Figure 28 – CNOT gates matrix of example circuit. 54

Figure 29 – Example circuit mapped to $\{Q_3, Q_2, Q_1, Q_4, Q_0\}$. 55

Figure 30 – Original circuit for the first evaluation of the CNOT mappings. 61

Figure 31 – Qubits permutation of Standalone algorithm for the first evaluation of the CNOT mappings. 61

Figure 32 – Standalone algorithm mapping for the first evaluation of the CNOT mappings. 61

Figure 33 – Qiskit mapping for the first evaluation of the CNOT mappings. 62

Figure 34 – Qubits permutation of proposed approach for the first evaluation of the CNOT mappings. 63

Figure 35 – Proposed approach for the first evaluation of the CNOT mappings. 63

Figure 36 – Example circuit to present the permutation problem. 66

Figure 37 – Example circuit mapped to IBM QX4 with trivial permutation. 66

Figure 38 – Example circuit of the permutation problem permuted. 66

Figure 39 – Mapping of the permuted circuit targeting the IBM QX4. 67

Figure 40 – Example circuit to describe the ILP formulation. 68

Figure 41 – Example circuit for the application of the ILP formulation. 74

Figure 42 – IBM QX5 architecture cost matrix. 74

Figure 43 – Example circuit for the ILP application permuted. 76

Figure 44 – Example circuit for the ILP application mapped. 77

Figure 45 – Circuit to test the impact of the CNOT gates in the ILP solving time. 78

Figure 46 – Solution time for circuits with 5 qubits in IBM QX4 architecture. 78

Figure 47 – V gate realized with Clifford+T library. 86

Figure 48 – $V^\dagger$ gate realized with Clifford+T library. 86

Figure 49 – Property of CNOT gates with T gate. 87

Figure 50 – Property of CNOT gates with $T^\dagger$ gate. 87

Figure 51 – Example of the V gate targeting the IBM QX4. 87

Figure 52 – Example of the V gate targeting the IBM QX4 using the property. 88

Figure 53 – Clifford+T mapping and NCV mapping cost matrices. 88

Figure 54 – Example to map NCV circuit to Clifford+T. 89

Figure 55 – Example of NCV circuit mapped to Clifford+T. 89

Figure 56 – Example of NCV circuit mapped after ILP formulation. 90

Figure 57 – Example of NCV circuit mapped to Clifford+T after ILP formulation. 91

Figure 58 – Toffoli implementation with Clifford+T gates using 17 gates. 91

Figure 59 – Toffoli implementation with Clifford+T gates using 15 gates. 92

Figure 60 – Another Toffoli implementation with Clifford+T gates using 15 gates. 92

Figure 61 – Another Toffoli implementation with Clifford+T gates using 15 gates and controls swapped. 92

Figure 62 – Toffoli with positive-negative controls to Clifford+T with 17 gates. 93

Figure 63 – Positive-negative controls Toffoli mapped to Clifford+T with 15 gates. 93

Figure 64 – Positive-negative controls Toffoli mapped to Clifford+T with 15 gates and CNOT changed. 93

Figure 65 – Toffoli with negative-positive controls to Clifford+T with 17 gates. 93

Figure 66 – Negative-positive controls Toffoli realized with 15 Clifford+T gates. 94

Figure 67 – Another negative-positive controls Toffoli realized with 15 Clifford+T gates. 94

Figure 68 – Toffoli with negative polarity to Clifford+T with 17 gates. 94

Figure 69 – Negative controls Toffoli mapped to Clifford+T with 15 gates. 94

Figure 70 – Another negative controls Toffoli mapped to Clifford+T with 15 gates. 95

Figure 71 – Another negative controls Toffoli mapped to Clifford+T with 15 gates and controls swapped. 95

Figure 72 – Example circuit to find the lowest cost permutation of a Toffoli circuit. 97

Figure 73 – Example circuit mapped to the lowest cost permutation. 99

Figure 74 – Lowest cost permutation mapped to Clifford+T. 99

Figure 75 – Lowest cost permutation mapped to IBM QX4 architecture. 99

Figure 76 – Circuit 3_17a. 100

Figure 77 – Circuit 3_17a mapped to Clifford+T (3_17e). 100

Figure 78 – Circuit 3_17e mapped to IBM QX4 using the CNOT mapping. 101

Figure 79 – Circuit 3_17e mapped to IBM QX4 using the ILP formulation. 101

Figure 80 – Mapped 3_17 circuit using NCV mapping approach. 101

Figure 81 – Toffoli circuit a2x_a. 104

Figure 82 – Clifford+T circuit a2x_e. 104

Figure 83 – Circuit a2x_e mapped with default permutation to IBM QX4. 104

Figure 84 – Circuit a2x_e mapped with best permutation to IBM QX4. 105

Figure 85 – Circuit a2x mapped to IBM QX4 with proposed approach. 105

Figure 86 – NOT gate acting on three different qubits. 122

Figure 87 – Example circuit to find a matrix representation of controlled gates. 125

Figure 88 – V gate mapped to Clifford+T. 127

Figure 89 – Toffoli to Clifford+T circuit with 17 gates and 13 levels. 128

Figure 90 – Property to change CNOT gates order. 128

Figure 91 – Toffoli circuit with order of CNOT gates changed. 129

Figure 92 – Toffoli circuit after adjacent gates removed. 129

Figure 93 – Swap of CNOT gates with $T^\dagger$ gate. 129

Figure 94 – Final circuit for mapping a Toffoli to Clifford+T. 129

Figure 95 – Toffoli to Clifford+T mapping with 17 gates and 13 levels. 130

Figure 96 – Toffoli mapping with gates moved. 130

Figure 97 – Toffoli mapping with gates swapped on qubits a and c. 130

Figure 98 – Toffoli mapping with adjacent CNOT gates to be removed. 130

Figure 99 – Final circuit for alternatively mapping a Toffoli to Clifford+T. 131

LIST OF TABLES

Table 1 – Basic functions of the Boolean algebra. 25

Table 2 – Truth table of XOR and XNOR functions. 26

Table 3 – Truth table of the adder. 27

Table 4 – Example of irreversible and reversible functions. 27

Table 5 – Truth table of the circuit from Figure 2. 29

Table 6 – Pauli gates representation. 34

Table 7 – Hadamard gate representation. 34

Table 8 – S and $S^\dagger$ gates representation. 35

Table 9 – CNOT gate representation. 35

Table 10 – V and $V^\dagger$ gates representation. 35

Table 11 – CNOT Mappings strategy compared with Standalone without Qiskit optimization. 57

Table 12 – CNOT Mappings strategy compared with Standalone with Qiskit optimization. 58

Table 13 – CNOT mappings strategy compared with Qiskit. 60

Table 14 – Benchmarks to compare CNOT Mappings with Qiskit in IBM QX5. 80

Table 15 – Benchmarks to compare CNOT Mappings with Standalone without optimization in IBM QX5. 82

Table 16 – Benchmarks to compare CNOT Mappings with Standalone optimized in IBM QX5. 83

Table 17 – Mapping NCV circuits compared with Qiskit in IBM QX5. 102

Table 18 – Mapping NCV circuits compared with Standalone optimized in IBM QX5. 103

Table 19 – Mapping Toffoli circuits compared with Qiskit in IBM QX4. 106

Table 20 – Mapping Toffoli circuits compared with Standalone optimized in IBM QX4. 107

Table 21 – Mapping Toffoli circuits compared with Qiskit in IBM QX5. 108

Table 22 – Mapping Toffoli circuits compared with Standalone optimized in IBM QX5. 110

Table 23 – Matrix representation of quantum gates. 123

Table 24 – ILP gap for Chapter 4. 132

Table 25 – ILP gap for Chapter 5. 132

NOMENCLATURE

CAD	Computer-Aided Design
CNOT	Controlled-NOT
DAG	Direct Acyclic Graph
IBM	International Business Machines Corporation
IC	Integrated Circuit
ILP	Integer Linear Programming
LNN	Linear Nearest Neighbor
MIP	Mixed Integer Programming
NISQ	Noise Intermediate-Scale Quantum
QASM	Quantum Assembly Language
QECC	Quantum Error Correction Codes
QISA	Quantum Instruction Set Architecture
QuMA	Quantum Microarchitecture
RHS	Right-Hand-Side
SAT	Satisfiability
TQEC	Topological quantum error correction
XNOR	Exclusive-NOR
XOR	Exclusive-OR

CONTENTS

1 INTRODUCTION 17

1.1 CONTRIBUTIONS AND RELATED WORK 21

2 BACKGROUND ON REVERSIBLE LOGIC AND QUANTUM COMPUTATION 25

2.1 REVERSIBLE LOGIC 25

2.2 QUANTUM COMPUTATION 31

2.3 COMPUTATIONAL TOOLS 37

3 CNOT MAPPINGS FOR IBM QUANTUM COMPUTERS 39

3.1 IBM QX ARCHITECTURES 39

3.2 CNOT GATES MAPPINGS 41

3.2.1 Types of movements 42

3.2.1.1 Control movements 43

3.2.1.2 Target movements 44

3.2.2 Template based mappings 45

3.2.3 Movements optimizations 47

3.2.4 Finding CNOT gates mappings 48

3.2.5 Generating architecture cost matrix 52

3.2.6 Example of CNOT gates mappings application 54

3.3 EXPERIMENTAL RESULTS OF CNOT GATES MAPPINGS 55

3.3.1 Benchmarks 56

3.3.2 Case of study 59

4 INTEGER LINEAR PROGRAMMING MODEL 65

4.1 THE PERMUTATION PROBLEM 65

4.2 THE INTEGER LINEAR PROGRAMMING MODEL 67

4.2.1 Objective function 69

4.2.2 First set of constraints 70

4.2.3 Second set of constraints 71

4.2.4 Third set of constraints 72

4.3 EXAMPLE OF APPLICATION OF THE ILP FORMULATION 74

4.4 EXPERIMENTAL RESULTS OF THE ILP FORMULATION 77

5 MAPPING OF NCV AND TOFFOLI CIRCUITS TO IBM QX 86

5.1 NCV TO CLIFFORD+T CIRCUITS 86

5.1.1 NCV Example 88

5.2 TOFFOLI TO CLIFFORD+T CIRCUITS 91

5.2.1 Toffoli Example 96

5.3 EXPERIMENTAL RESULTS OF NCV AND TOFFOLI MAPPINGS 99

5.3.1 NCV mapping evaluation 100

5.3.2 Toffoli mapping evaluation 103

6 CONCLUSIONS AND FUTURE WORK 112

6.1 FUTURE WORK 113

REFERENCES 115

APPENDIX A – MATRIX REPRESENTATION OF QUANTUM GATES AND PROOF OF CIRCUITS EQUIVALENCE 122

A.1 FINDING MATRIX REPRESENTATION FOR SINGLE QUBIT GATE 122

A.2 FINDING MATRIX REPRESENTATION FOR CONTROLLED GATE 124

A.3 PROVING THE CIRCUIT EQUIVALENCE 126

APPENDIX B – TOFFOLI OPTIMIZATIONS 128

B.1 FIRST POSSIBLE MAPPING 128

B.2 SECOND POSSIBLE MAPPING 129

APPENDIX C – GAP OF THE CIRCUITS WITH SOLUTION TIME EXCEEDED 132

1 INTRODUCTION

The increased scale of integration of electronic circuits has aided the development of ever smaller and more powerful circuits. Miniaturization of integrated circuits (IC) employing more advanced technologies at the nanometer scale faces some problems with the ever decreasing size of transistors and increasing power consumption. Consequently, these technologies will reach their limit in the very near future. This shows that the design of electronic circuits has become an increasingly complex problem.

There is, therefore, a great interest in developing technologies that are free from the limitations arising from the exponential decrease of the transistors that currently approaches the critical dimensions inherent in the fundamental physics limits.

An alternative for the traditional computing is the quantum computation. Since quantum computing is reversible, reversible logic has become an essential aspect of describing the physical reality of quantum computing (WILLE; CHATTOPADHYAY; DRECHSLER, 2016).

The main difference is the representation of the information unity, while in the traditional computation there are only two bit states 0 and 1, in quantum computation the information is represented using the quantum bit, or qubit. The qubit allows to represent the basis states 0 and 1, also to represent a combination of both states at the same time (NIELSEN; NIELSEN; CHUANG, 2000).

It has been shown that with quantum computation, problems can be solved much more efficiently, e.g., exponential problems can be solved in polynomial time (SHOR, 1997). This efficiency in solving problem is due to the fact that quantum computing is reversible (AL-RABADI, 2004). A characteristic of the reversible circuit is that no energy is dissipated, since no information is lost (LANDAUER, 1961). This may lead to a low power consumption in quantum computation, although, it requires more research, since the quantum computers are still in development.

A quantum computer works by using specific quantum particles (e.g, a photon) as a qubit and controls it by giving a sequence of pulses to run a specific operation and change its state (spin rotation). These pulses are executed in parallel affecting all the qubits at

the same time. However, the quantum computer still cannot isolate the qubits from the noises making it susceptible to errors (GOMES, 2018).

The research in quantum computation has increased with the progress of physical implementation of quantum computers (ZULEHNER *et al.*, 2019). Although several theoretical algorithms have been developed (AMY *et al.*, 2013; WILLE; LYE; DRECHSLER, 2014), the physical realization of the quantum circuits was not yet available to the researchers. In 2011, the D-Wave Systems developed the first *commercial* quantum computer (D-Wave one) with 128-qubit chipset applying the metaheuristic quantum annealing targeting optimization problems. However, only in early 2017 IBM made available the project IBM Q to give public access to quantum processors (IBM, e). At the time of this writing, there are quantum processors with 5, 16 and 20 qubits, IBM QX4, IBM QX5 and QS1_1, respectively (IBM, a). Other companies like Microsoft, Rigetti and Google also announced the development of quantum computers (COURTLAND, 2017).

These quantum computers are still in early development and the operations have low fidelity, i.e., errors that can change the desire output state. In order to keep the error to a minimum and increase the fidelity of the quantum circuit, some researches proposed different fault-tolerant algorithms (DING *et al.*, 2018; PALER, 2018) and verification codes (NAVEH *et al.*, 2018). Other researches have been doing some work to address the topologies of the quantum computation. The topological quantum error correction (TQEC) techniques are used to prevent the error in 1-D and 2-D architectures (LIN *et al.*, 2018).

A different approach to reduce the error, hence, improving the fidelity is the optimization of quantum circuits, by reducing its size (LIU *et al.*, 2019). A combination of Linear Nearest Neighbor and Quantum Error Correction Codes (QECC) was proposed to optimize quantum circuits (DAS; CHATTOPADHYAY; RAHAMAN, 2019). Another work that applies the same LNN technique to optimize quantum circuits is presented by Das, Chattopadhyay and Rahaman (2019), which explores the optimization of quantum circuits targeting the modular exponentiation in order to achieve large-scale applications.

More work has been done in the search for a scalable quantum computation (MORELLO *et al.*, 2018). A new architecture was proposed by Sargaran and Mohammadzadeh (2019) called SAQIP, which reduces the architecture's latency. Other work proposed a new methodology to scale the design of quantum systems (VAN DIJK *et al.*, 2018). However, the Computer-Aided Design (CAD) for Quantum Computation has several challenges underway (WILLE; FOWLER; NAVEH, 2018).

In order to help the development of quantum circuits, high level languages to describe and simulate a quantum circuit are being developed (SVORE *et al.*, 2018). The authors in Fu *et al.* (2019) developed the eQASM, a quantum instruction set architecture (QISA) that can be translated from the quantum assembly language (QASM) (CROSS *et al.*, 2017). Analogously, the work of Fu *et al.* (2018) proposes a quantum microarchitecture (QuMA), which works in a low-level between quantum software and quantum hardware.

The simulation of quantum computers is not a trivial task for classical computers, since it requires a considerable computer power (memory and processor). However, some works have proposed simulation platforms, such as the one called QX (KHAMMASSI *et al.*, 2017). In recent work, Zulehner and Wille (2019a) presented a simulation for quantum circuits based on graphs, exploiting the redundancies of the quantum state in order to increase the performance of the simulation.

Some computational tools for the generation and simulation of quantum circuits have been developed, for example, the tool Qiskit (ALEKSANDROWICZ; ALEXANDER; BARKOUTSOS,) used in the IBM Q project (WILLE; VAN METER; NAVEH, 2019). In the literature, there are some other tools with the same purpose (PROUSALIS; KONOFAOS, 2016). Among the computational tools, the one used in this work is RevKit which is a framework developed using the C ++ programming language (SOEKEN *et al.*, 2008). RevKit is widely used for the development of new algorithms for the synthesis and optimization of reversible circuits and quantum circuits.

Another research area targeting the improvement of quantum computation is regarding the decoherence, i.e., loss of information of a quantum circuit (O'CONNELL, 2005). In order to address that, some authors proposed an improvement of the depth of a quantum circuit, such as the one proposed by Bhattacharjee and Chattopadhyay (2017) in which uses Integer Linear Programming to achieve depth-optimal quantum circuits for arbitrary topologies. Similar work was proposed by Amy *et al.* (2013), in which an algorithm to synthesize depth-optimal quantum circuits, consisting in parallelizing the T gates was presented.

The realization of a quantum circuit has some restrictions regarding the *position* of the gates in different topologies (architectures) (JAVADI-ABHARI *et al.*, 2017). Different approaches have been developed, such as the one presented by Bhattacharjee *et al.* (2018) which realizes a quantum circuit for 2-D architectures using the Linear Nearest Neighbor (LNN) technique. The work from Kole, Datta and Sengupta (2018) presented a heuristic to generalize the realization of quantum circuits for different number of architectures

dimension.

These techniques can be used to map a quantum circuit to available quantum computers. These quantum computers have restrictions in the type of quantum gates that can be used and the qubits that can be used to implement them. For example, quantum circuits in the IBM quantum processors can be realized with a set of quantum gates acting on a single qubit, e.g., T, $T^\dagger$ and the Controlled-NOT (CNOT) gates, acting on two qubits. The available single qubit gates can be realized without restriction, however, the CNOT gates have some restrictions that are defined by a coupling graph. This coupling graph describes the available qubits which can be used to implement a CNOT gate. It means that not all CNOT gates can be used and need some transformation.

These restrictions of IBM architectures raised a problem to be solved, i.e., realize a quantum circuit satisfying the coupling graph. This is not a trivial task, since satisfying the restrictions increases the circuit due the need to implement the CNOT gate using only the available qubits. Therefore, the goal is to realize a quantum circuit in an IBM architecture in such way that result in a smaller circuit as possible (regarding the number of gates) in order to increase its fidelity. In this thesis this problem is addressed.

This thesis proposes a methodology to map quantum circuits to any quantum architectures available. Also, the methodology is adapted to map reversible circuits as well. The approach used is to define a CNOT mapping for each possible pair of qubits in a given architecture with a respective cost regarding the number of additional gates required to do the mapping. The goal of these CNOT mappings is to map a quantum circuit complying with the architecture constraints in such way that the permutation at the input and output of the circuit remains the same. After the CNOT gates defined, the problem was to find a good mapping for the qubits in order to reduce the number of CNOT gates not satisfying the constraints. To address this problem, an Integer Linear Programming (ILP) formulation was proposed. Combining the CNOT mappings along with the ILP formulation the reversible and quantum circuits can be mapped to any quantum architecture regarding the CNOT gates restrictions.

The next subsection presents the main contributions of this thesis along with some related work.

1.1 CONTRIBUTIONS AND RELATED WORK

One of the contributions of this thesis is to propose an adaptable methodology to map quantum and reversible circuits to quantum processors. The architectures used to describe this methodology are the ones available by the project IBM Q. The quantum processors have different topologies (architectures) regarding the coupling architectures constraints, i.e., the pair of the CNOT gates can be applied. Some works have also addressed the mapping problem regarding the constrains of quantum architectures.

Siraichi *et al.* (2018) proposed an exact approach using Dynamic Programming to find the optimal qubits permutation to quantum architectures. However, the exact solution only worked for circuits with up to 8 qubits. In order to scale the approach, a heuristic solution was proposed. The approach was divided into two stages, the first stage tries to satisfy as many constraints as possible to find the initial mapping. Followed by the second stage, satisfying the remaining constraints, solving each CNOT gate at time. The results presented in the paper showed that the proposed heuristic outperforms other strategies. However, for the 16 qubits architectures the IBM's algorithm found better results.

Zulehner, Paler and Wille (2019) proposed a methodology to map quantum circuits to the IBM QX quantum architectures by partitioning the circuit into sub-circuits. The methodology consists in use the search algorithm A^* to find a low cost permutation trough the sub-circuits while satisfying the architecture constraints adding as few gates as possible. The evaluation was regarding the size of the circuits and its depth (number of levels). Results have shown that the proposed approach outperforms the IBM's algorithm. This work is the continuation of Zulehner, Paler and Wille (2018). The same authors took advantage of the previous methodology and changed the supported gates to propose a mapping using SU(4) quantum gates to the IBM QX architectures (ZULEHNER; WILLE, 2019b). The approach was compared with IBM algorithm and the evaluation was done regarding the fidelity of the mapped circuit, i.e., different gates has different costs.

Aiming the fidelity of the quantum circuit, the authors Ash-Saki, Alam and Ghosh (2019) proposed a mapping strategy to increase the fidelity of the circuit in quantum architectures. The approach targets the mapping of gates that have lower error probabilities to find a circuit with better fidelity. The circuit permutation is found trying all possible permutation and then swapping the qubits targeting the operations with higher fidelity, although the scalability is compromised due the exhaustive search. Therefore, a greedy approach is used. The results have shown that the proposed approach can find circuits with higher fidelity, however the improvement decreases as the number of qubit

increases.

Soeken *et al.* (2019) presented an algorithm that translates single-target gates into a quantum circuit composed of the elementary quantum gates. The algorithm has two steps. In the first step a circuit with a sequence of single-target gates using an adaptation of the synthesis Young-subgroup is realized. Then an optimization of the CNOT gates have been done by taking into account the architecture coupling constraints. Finally, the single gates are rewritten implementing H gates, CNOT gates and R_z gates. The presented results showed that the proposed approaches targeting a quantum computer architectures reduce the number of gates and depth of the final circuit.

Wille, Burgholzer and Zulehner (2019) presented a method to map quantum circuits to IBM QX architectures with a minimal number of SWAP and H operations by using Boolean satisfiability solvers. In order to accomplish that, all possible applications of the SWAP and H operations are calculated using the Boolean satisfiability solvers (SAT). This search is computational expensive and only circuits with few CNOT gates could be found. The results have shown that the heuristic used in IBM algorithm exceeds the lower bound of gates by more than 100% on average.

Following the goal of reducing the number of necessary SWAP gates in the quantum circuits, the authors Matsuo, Hattori and Yamashita (2019) proposed a strategy to reduce the number of SWAP gates into IBM QX architectures. The proposed methodology consists in partitioning the circuit in sub-circuits finding the best qubit placement for the adjacent sub-circuit. To do that, a SAT solver is used, i.e., the qubit placement is done based on Boolean satisfiability problem. Also, the order of quantum gates are changed when possible. SWAP gates are used between two sub-circuits to permute the circuit as needed using the search algorithm $A*$. Results have shown that the proposed algorithm reduces the number of sub-circuits by moving the gates, thus, reducing the SWAP gates needed.

Li, Ding and Xie (2019) proposed an algorithm to map quantum circuits to Noise Intermediate-Scale Quantum (NISQ) devices. The proposed algorithm applies a bidirectional heuristic search algorithm using SWAP gates, optimizing every search. Also, the algorithm globally optimizes the initial mapping using a reverse transversal technique. Furthermore, the algorithm also evaluate, whenever possible to trade the number of gates with the depth of the circuit. The constraints between two qubits are represented by a Direct Acyclic Graph (DAG) that will be used to guide the heuristic search algorithm. The heuristic has three design techniques to ensure a better final circuit mapping, which

is a reduction in the search space for the SWAP gate insertion, reversing the circuit to find a better initial mapping and a evaluation between the circuit depth or the number of additional gates. Experimental results were compared with Zulehner, Paler and Wille (2019) and showed that the proposed approach outperforms the compared one.

The main difference, in the mapping of quantum circuits targeting the IBM architectures, between the others approaches and the proposed in this thesis is that the permutation at the input and at the output of the circuit are the same. This is done by finding a mapping for each possible CNOT in the target architecture using some qubits movements and optimizations. The others approaches change the circuits' permutation with the mapping, while in the proposed approach the mapping keeps the same permutation.

The other contribution of this thesis is the Integer Linear Programming formulation for reversible circuits. This formulation can be applied to any reversible circuit, with reduced number of constraints (it changes with the type of the circuit). Another work proposed an ILP approach to solve the qubits permutation.

Shafaei, Saeedi and Pedram (2014) proposed a Mixed Integer Programming (MIP) model to allocate qubits in 2-D grid architectures. The formulation used binary and real variables to represent the mapping of qubits to a respective location in the grid. Let n be the number of grid nodes, the number of binary variables is n^2, n^2 real variables and n^2+2n constraints. In the formulation proposed in this thesis, the CNOT gates were used as integer variables and the number of constraints can change with the characteristic of the circuit. Let n be the number of qubits of the architecture, x be the number of qubits with at least two CNOT gates and y the number of different CNOT gates in the circuit. The number of variables are $(n^2-n)*y$ and the number of constraints are $(x*n)+(n+y)$. This is the main difference from the formulation proposed by Shafaei, Saeedi and Pedram (2014). By changing the number of constraints with the type of circuit, the formulation can be more scalable and requires less time to be solved. The formulation is used to find the mapping that will require the fewer additional gates regarding the CNOT gates mappings.

Taking advantage of the CNOT mappings and of the proposed ILP formulation, the methodology were applied to map NCV circuits and Toffoli circuits into the quantum architectures.

The thesis is structured as follows. Chapter 2 presents some background regarding reversible logic and the quantum computation. The IBM quantum processors and

architectures along with defined qubits movements and the CNOT mappings are shown in Chapter 3. This chapter also shows how the permutation of a circuit can change the final cost and the problem to find this permutation. Then, Chapter 4 describes the permutation problem formulated as an Integer Linear Programming model and its applications. Chapter 5 shows an application of the defined CNOT mappings to map NCV and Toffoli circuits to IBM architectures along with the ILP formulation, followed by Chapter 6 concluding the thesis and presenting some future works.

2 BACKGROUND ON REVERSIBLE LOGIC AND QUANTUM COMPUTATION

This chapter presents a brief theory regarding reversible logic and quantum computation. Section 2.1 shows the theory about reversible logic. Section 2.2 presents quantum computation theory along with quantum gates and circuits. It also be briefly present two computational tools, in Section 2.3, that are used to support the research in the area of reversible logic and quantum computation.

2.1 REVERSIBLE LOGIC

The traditional computation has the bit as the information unity and uses the binaries (Boolean, $\mathbb{B}$) variables to represent it, assuming two values, 0 or 1, with $\mathbb{B} \in \{0, 1\}$. The traditional computation is realized trough functions with binary variables defined in Definition 2.1 (TOCCI; WIDMER, 1997).

Definition 2.1. *A Boolean function* $f : \mathbb{B}^n \rightarrow \mathbb{B}$ *with* $n \in \mathbb{N}$ *is a set of variables* $(x_1, x_2, x_3, ..., x_n)$ *mapped to an output* $f(x_1, x_2, x_3, ..., x_n)$.

To realize this mapping from input to output are utilized the Boolean operators (gates). The basic operators of the Boolean algebra are AND ($\wedge$), OR ($\vee$) and NOT ($^{-}$). Table 1 shows the truth table of the AND, OR and NOT, respectively.

Table 1 – Basic functions of the Boolean algebra.

(a) AND.

x_1	x_2	$f(x_1, x_2)$
0	0	0
0	1	0
1	0	0
1	1	1

(b) OR.

x_1	x_2	$f(x_1, x_2)$
0	0	0
0	1	1
1	0	1
1	1	1

(c) NOT.

x_1	$f(x_1, x_2)$
0	1
1	0

Source: Prepared by the author.

From Table 1, it can be noted that the output of AND will be 1 only when both inputs have logic value are 1. For the OR, the output will be 1 if at least one of the inputs

has value logic 1 and the NOT always invert the logical value of its variable, acting as a negation.

An example of a function realized with the basic operators is the Exclusive-OR (XOR) and Exclusive-NOR (XNOR) functions. This function is presented in Formula 1 and is realized with the basic operators. Also, Table 2 shows the respective truth table.

$$f(x_1, x_2) = x_1 \oplus x_2 = (x_1 \wedge \overline{x_2}) \vee (\overline{x_1} \wedge x_2) \tag{1}$$

Table 2 – Truth table of XOR and XNOR functions.

(a) XOR.

x_1	x_2	$f(x_1, x_2)$
0	0	0
0	1	1
1	0	1
1	1	0

(b) XNOR.

x_1	x_2	$f(x_1, x_2)$
0	0	1
0	1	0
1	0	0
1	1	1

Source: Prepared by the author.

Note that the output of the XOR function will be 1 only when the inputs have different logic values. The XNOR is the negated XOR. Some Boolean functions can have more than one output as defined in Definition 2.2 (WILLE; DRECHSLER, 2010).

Definition 2.2. *Boolean functions with more than one output are called multiple output Boolean function where* $f : \mathbb{B}^n \rightarrow \mathbb{B}^m$ *where* $n, m \in \mathbb{N}$ *are a set of variables* $(x_1, x_2, x_3, ..., x_n)$ *mapped to multiple output functions* $(f_1, f_2, f_3, ..., f_m)$.

An example of a multiple output Boolean function is an adder, with three input variables and two outputs. Table 3 shows the truth table of this adder.

There is a special type of multiple output Boolean function which has the capability of obtain the input value of the circuit from an output. This function is called reversible function and is defined in Definition 2.3 (VINOKUROV; FRANTSEVA, 2016).

Definition 2.3. *A reversible function is a bijective function, i. e., each output has only one input. Also, a reversible function must have the same number of inputs and outputs* $(\mathbb{B}^n \rightarrow \mathbb{B}^n$ *com* $n \in \mathbb{N})$.

If one of the criteria is not fulfilled, the function is called irreversible. The adder function, presented in Table 3, is an irreversible function due the different numbers of

Table 3 – Truth table of the adder.

x_1	x_2	x_3	f_1	f_2
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Source: Prepared by the author.

inputs and outputs. Table 4 shows an example of an irreversible and a reversible function, respectively.

Table 4 – Example of irreversible and reversible functions.

(a) Irreversible.

x_1	x_2	x_3	f_1	f_2	f_3
0	0	0	0	0	0
0	0	1	1	1	1
0	1	0	0	1	1
0	1	1	0	1	0
1	0	0	0	0	1
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	0

(b) Reversible.

x_1	x_2	x_3	f_1	f_2	f_3
0	0	0	1	0	0
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	0	0	1
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0

Source: Prepared by the author.

The function in Table 4(a) has the same number of inputs and outputs, but two outputs are repeated (111). Therefore, it is not a bijective function. In the case of the Table 4(b), the function is reversible, since the number of inputs and outputs are equal and each input has only one output, without repetition.

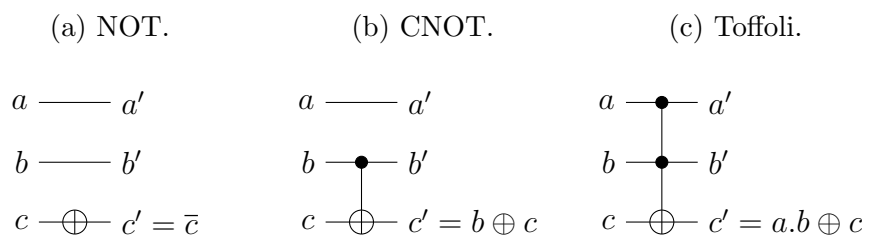
There are some methodology that realize the transformation of an irreversible function into a reversible one. Maslov and Dueck (2004) verified that at least $log_2(\mu)$ output functions must be added to an irreversible function in order to become reversible. The value of μ is the number of times an output was repeated in the truth table. If the number of outputs results in a larger number than the input, some variables must be added to the input to match the quantities. This added variables are called *constants* and the added output variables are called *garbage*.

As in the conventional logic, any function in the reversible logic can be designed by

a set of basic reversible gates. This basic gates are the NOT, Controlled-NOT (CNOT) and Toffoli.

The CNOT and Toffoli consist of a target, represented by the symbol $\oplus$ and one or more controls, represented by the symbols $\bullet$ (positive control) or $\circ$ (negative control). The reversible gates will act on the target qubit only if the control are "active", i. e., the positive control has a logic value 1 and the negative control has a logic value 0. The only gate that do not have a control is the NOT gate. Figure 1(a), 1(b), 1(c) show the basic reversible gates NOT, CNOT and Toffoli, respectively.

Figure 1 – Reversible gates.



Source: Prepared by the author.

Each line in Figure 1 represents a variable. The NOT gate has only the target qubit and inverts the logic value of the variable on the qubit that is acting.

The CNOT gate is a controlled NOT with a control qubit and a target qubit. This gate inverts the logic value on the target qubit when the control qubit is "active", i. e., the control qubit has the same logic value (polarity) of the control. For example, the logic value to activate the positive control is 1 and for the negative control is 0.

The Toffoli gate is controlled NOT with two positive controls. This gate inverts the logic value on the target qubit when both controls are "active". Also, there is the mixed Toffoli, in which the controls have different polarities and the negative controlled Toffoli with both negative controls. These Toffoli gates acts on the target qubits when the controls have the same polarities of the variables on the controls qubits.

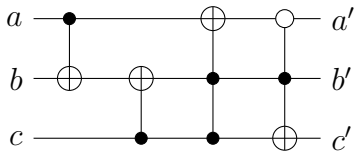
Note that the CNOT gate with positive controls acts like an XOR and if the control has negative polarity, the gate acts like a XNOR. For example, the CNOT presented in Figure 1(b), will invert the logic value of the variable c when the variable b has logic value 1. This fact can be seen in Table 2 with the variable $b = x_1$, $c = x_2$ and $c' = f(x_1, x_2)$.

By grouping different reversible gates next of each other a reversible circuit is created, defined in Definition 2.4.

Definition 2.4. *A reversible circuit is a cascade of n reversible gates* $(g_0, g_1, g_2, ..., g_n)$*, where* $C=\{c_0, c_1, c_2, ..., c_n\}$ *is a set of controls and* $T=\{t_0, t_1, t_2, ..., t_n\}$ *is a set o targets where,* $g_{c_i} \neq g_{t_i}$ *with* $i \leq n$*.*

In the reversible circuit is not allowed to have feedback and fanout. Figure 2 shows an example of a reversible circuit with three variables.

Figure 2 – Example of a reversible circuit with three variables.



Source: Prepared by the author.

The reversible circuit is bidirectional, it means that the circuit can be executed from input to output or from the output to input. For the remainder of this work, the left side of the circuit will be defined as input and the right side as output. Thus, in Figure 2, the variables a, b and c are the inputs and the a', b' and c' are the outputs. Table 5 shows the truth table of the circuit from Figure 2.

Table 5 – Truth table of the circuit from Figure 2.

a	b	c	a'	b'	c'
0	0	0	0	0	0
0	0	1	1	1	1
0	1	0	0	1	1
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	1	0	1
1	1	0	1	0	0
1	1	1	0	1	0

Source: Prepared by the author.

Applying the $\{a,b,c\} = \{1,1,1\}$ as input, the output will be $\{a',b',c'\} = \{0,1,0\}$. Since the circuit is reversible, applying $\{a',b',c'\} = \{0,1,0\}$ to the "output" (acting as input), the input variables will be $\{a,b,c\} = \{1,1,1\}$ (acting as output). That is the main characteristic of this type of circuits, the reversibility. Due the reversibility of the circuits, the feedback and the fanout is not allowed, i. e., the output of a reversible gate is connected to only one input of other reversible gate.

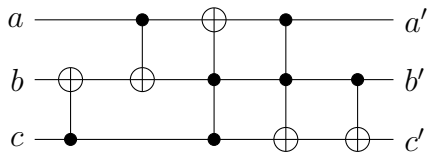
There are some methodologies to transform a reversible specification into a reversible circuit. This transformation process is called synthesis of a reversible

circuit. The synthesis of a reversible circuit can be done from a reversible function completely specified (YANG *et al.*, 2015), from incompletely specified reversible functions (SCHAEFFER, 2017) and from irreversible functions (ZULEHNER; WILLE, 2017).

One of the synthesis algorithm for a reversible circuit completely specified is the MDM (DUECK; MASLOV; MILLER, 2003). In this algorithm the truth table of the reversible function is transversed applying Toffoli gates in order to transform the truth table into an identity (the input is the same as the output). After the truth table was transformed into an identity, the reversible circuit is generated with the cascade of Toffoli gates.

Applying the MDM algorithm to the truth table from Table 5, the circuit presented in Figure 3 is obtained.

Figure 3 – Circuit generated with MDM algorithm.



Source: Prepared by the author.

It can be noticed that the circuit presented in Figure 2 is different from the circuit from Figure 3. The circuit from Figure 2 has 4 gates, while the circuit from Figure 3 has 5 gates. However, both circuits has the same specification (truth table).

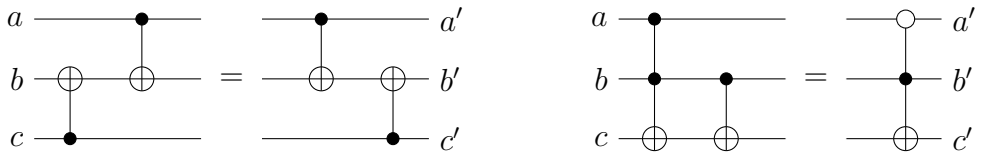
This difference can occur due the optimization techniques of reversible circuits. The optimization is used to reduce the number of gates of a circuit without changing the specification. Soeken and Thomsen (2013) showed some rules that can be used to optimize the reversible circuits. For example, regarding the presented circuits, the equivalences in Figure 4 can be used.

Applying the equivalences from Figure 4, the circuit from Figure 3 can be optimized and obtain the same circuit from Figure 2.

One of the criteria used to evaluate a reversible circuit is the number of gates. However, the quantum cost can be used as well. The goal of this cost is to inform how many gates will be necessary to transform a reversible circuit into a quantum circuit. The table cost of this transformation depends on the architecture and configuration of the quantum circuit to be realized.

Figure 4 – Reversible gates equivalences.

(a) Equivalence 1. (b) Equivalence 2.



Source: Prepared by the author.

One application of the reversible logic is the quantum computation (NIELSEN; NIELSEN; CHUANG, 2000). Some concepts about quantum computation are presented in the next section.

2.2 QUANTUM COMPUTATION

In contrast with the conventional logic, quantum computation uses a different type of information unity, the quantum bit (qubit). The representation of a qubit state is done using the *Dirac* notation, with the symbol $|\rangle$.

Geometrically, the state of a qubit is represented by a unit vector in the complex space of two dimensions Hilbert space. Figure 5 shows the representation of the classical states, 0 and 1, in *Dirac* notation.

Figure 5 – Representation of the classical states in *Dirac* notation.

(a) Bit 0. (b) Bit 1.

$$|0\rangle \equiv \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad |1\rangle \equiv \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Source: Prepared by the author.

The difference between a bit and a qubit lies on the possibility of a qubit representing other states beyond the classical ones, $|0\rangle$ and $|1\rangle$, defined in Definition 2.5.

Definition 2.5. *Superposition is the linear combination of classical states, i.e., representing both states at the same time.*

The superposition of two qubits is presented in Equation 2.

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{2}$$

The α and β numbers are *amplitudes* that represents the relation between the qubit and the basis states. These numbers obey the relation of Equation 3.

$$|\alpha|^2 + |\beta|^2 = 1 \tag{3}$$

The relation from Equation 3 represents the probability of a qubit of being in state $|0\rangle$ or in state $|1\rangle$. The quantum mechanics claim that is not possible to obtain information regarding the state of qubit with precise values of α and β. Therefore, the probability of the qubit being in the state 0, $|\alpha|^2$, and the probability of the qubit being in the state 1, $|\beta|^2$. When the superposition is measured the qubit collapses to one of states with respective probability. The term collapse is used to express this *change* of the qubit state.

The concept of superposition of states also applies for multiple qubits. For example, for two qubits the possible states are $|00\rangle$, $|01\rangle$, $|10\rangle$, $|11\rangle$ and the superposition of a pair of these qubits. Equation 4 shows the vector that describes the linear combination of these two qubits.

$$|\psi\rangle = \alpha_{00}\,|00\rangle + \alpha_{01}\,|01\rangle + \alpha_{10}\,|10\rangle + \alpha_{11}\,|11\rangle \tag{4}$$

The measurement x of these qubits can result in the values 00, 01, 10, or 11 with probability $|\alpha_x|^2$, being the state qubit measured $|x\rangle$. The values of α_x must obey the relation in Equation 5. The notation $\{0,1\}^2$ represents the set of all possible combinations for two qubits.

$$\sum_{x\in\{0,1\}^2} |\alpha_x|^2 = 1 \tag{5}$$

Beyond the superposition, other characteristic that differs the classical computation of the quantum computation is the entanglement defined in Definition 2.6.

Definition 2.6. *The entanglement is the exchange of information between two qubits, i.e., the state of one relies on the state of the other.*

In the entanglement is not possible to describe each qubits state independently of the other qubits, due the measurements of one qubit be correlated to the measurement of the other qubit.

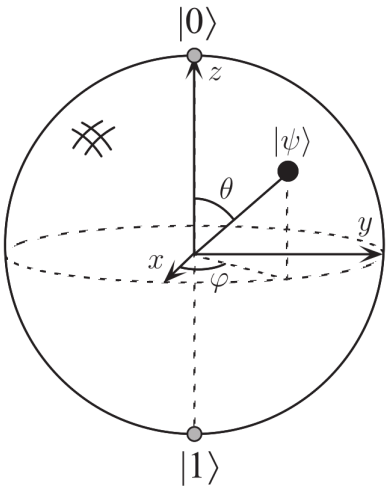
A qubit state can be represented in three dimensional space using the *Bloch* sphere. To do this the Equation 2 must be rewritten and the Equation 6 is obtained (NIELSEN;

NIELSEN; CHUANG, 2000).

$$|\psi\rangle = cos\frac{\theta}{2}|0\rangle + e^{i\varphi}sen\frac{\theta}{2}|1\rangle \tag{6}$$

With Equation 6 is possible to represent the qubit in the Bloch sphere. Figure 6 shows this representation. The numbers θ and φ define a point in the three dimension sphere, where $0 \leq \theta \leq \pi$ and $0 \leq \varphi \leq 2\pi$. Most of the operations on qubits can be represented using this sphere, making it useful for quantum computing and quantum information. The Bloch sphere is used to represent only a single qubit, since there is no generalization of the sphere representation for multiple qubits. However, some researches have a representation for a model for two qubits (WIE, 2014).

Figure 6 – Bloch sphere.



Source: Nielsen, Nielsen and Chuang (2000).

The difference between classical computation and quantum computation can be seen on the Bloch sphere. The state of the classical bits 0 and 1 are represented by the north and south poles of Bloch sphere. However, a qubit state can be represented by any points on the surface of the Bloch sphere.

The transformation of one qubit state into another can be done by a quantum gate (BARENCO *et al.*, 1995). These quantum gates are represented by matrices with size $2^n \times 2^n$, where n is the number of qubits of the quantum gate. The representation of a gate on different qubits can be seen in Appendix A. One characteristic of the quantum gates is that all the matrices of quantum gates are unitary. Moreover, any unitary matrix can be a quantum gate.

A set of quantum gates acting on a single qubit which has the characteristic of apply a π radians rotation around the main axes of the Bloch sphere are called Pauli gates. The

Pauli-X, Pauli-Y and Pauli-Z rotates the qubit around the x, y and z axes, respectively. Table 6 shows these gates representation. Note that the Pauli-X is the NOT gate and can be represented by the symbol $\oplus$.

Table 6 – Pauli gates representation.

Gate	Matrix	Representation	Gate	Matrix	Representation	Gate	Matrix	Representation
Pauli-X	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	$-\boxed{X}-$	Pauli-Y	$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$	$-\boxed{Y}-$	Pauli-Z	$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$	$-\boxed{Z}-$

Source: Prepared by the author.

The Pauli-X (NOT) only changes the state of the acting qubit by applying a π rotation around the x axis. The Pauli-Y gates change the state and the phase of the acting qubit by applying a π rotation around the y axis. The Pauli-Z qubit state remains the same, only changing its phase (φ) by applying a π rotation around the z axis. The rows of the matrix represents the input states and the columns represent the outputs for the respective input, e.g., the first row of the Pauli-X matrix from Table 6 represents the input state $|0\rangle$ and the second row the input state $|1\rangle$. Therefore, for the input $|0\rangle$, the output will be $|1\rangle$, i. e., the input state will be inverted.

Another quantum gate is the Hadamard gate. This gate acts on a single qubit and creates an equal superposition of the basis states. Table 7 presents the Hadamard gate and its representation.

Table 7 – Hadamard gate representation.

Gate	Matrix	Representation
H	$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$	$-\boxed{H}-$

Source: Prepared by the author.

Applying the Hadamard gate takes the qubit state to the (x, y) plan, i.e., the chances of the qubit collapses in $|0\rangle$ or $|1\rangle$ are the same.

Another quantum gates are the S, $S^\dagger$ and T, $T^\dagger$. The symbol $\dagger$ represents the matrix adjoint, e.g., the matrix of $S^\dagger$ is the adjoint matrix of S. Table 8 shows the gates.

The S and T do not change the states of the qubits. However change its phase, i.e., the S and T gates represent a $\pi/2$ and $\pi/4$ rotation around the z axis, respectively.

The quantum computation also have controlled gates. One of the most important gate in the quantum computation is the CNOT. The importance of the CNOT gate is due its entanglement between two qubits. Table 9 shows the CNOT representation.

Table 8 – S and $S^\dagger$ gates representation.

Gate	Matrix	Representation	Gate	Matrix	Representation
S	$\begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$		$S^\dagger$	$\begin{pmatrix} 1 & 0 \\ 0 & -i \end{pmatrix}$	
T	$\begin{pmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{4}} \end{pmatrix}$		$T^\dagger$	$\begin{pmatrix} 1 & 0 \\ 0 & e^{\frac{-i\pi}{4}} \end{pmatrix}$	

Source: Prepared by the author.

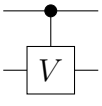
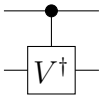
Table 9 – CNOT gate representation.

Gate	Matrix	Representation
CNOT	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$	

Source: Prepared by the author.

The CNOT is the same used in the classical reversible computation, the gate will act on the target qubit when the control is the state $|1\rangle$. Another controlled gate that is used in this work is the Controlled-V and Controlled-$V^\dagger$ gate, that is shown in Table 10. For the remainder of this text, the Controlled-V and Controlled-$V^\dagger$ gates will be referred as V and $V^\dagger$ gates.

Table 10 – V and $V^\dagger$ gates representation.

Gate	Matrix	Representation	Gate	Matrix	Representation
V	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1+i}{2} & \frac{1-i}{2} \\ 0 & 0 & \frac{1-i}{2} & \frac{1+i}{2} \end{pmatrix}$		$V^\dagger$	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1-i}{2} & \frac{1+i}{2} \\ 0 & 0 & \frac{1+i}{2} & \frac{1-i}{2} \end{pmatrix}$	

Source: Prepared by the author.

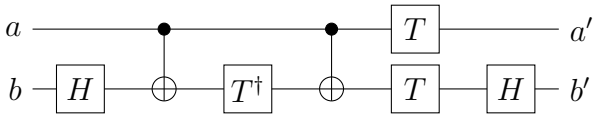
There are other gates in the quantum computation. However, only the ones used in this work were presented. These gates can be joined into libraries. The Clifford+T consists of the gates NOT, CNOT, H, S, $S^\dagger$, T and $T^\dagger$. Another one is the NCV, which is composed by the gates NOT, CNOT, V and $V^\dagger$ gates.

Using the gates previously presented, it is possible design a quantum circuit, defined in Definition 2.7.

Definition 2.7. *Let n be the number of gates, the quantum circuit can be defined as a cascade of quantum gates $g_0, g_1, g_2, ..., g_n$, where g_i with $i \leq n$ is a gate represented by a unitary matrix acting on a single qubit q_i or a controlled gate acting on qubits (q_i, q_j) where $i \neq j$.*

Figure 7 shows an example of a quantum circuit realized with Clifford+T gates.

Figure 7 – Example of a quantum circuit realized with Clifford+T gates.



Source: Prepared by the author.

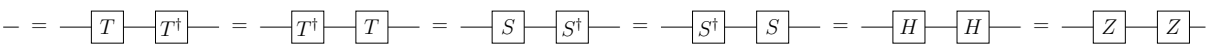
Multiplying the matrices of each gate will result in the matrix of the Controlled-V gate (Table 10, as can be seen in Section A.3 of Appendix A). For example, applying the input $|11\rangle$ the output will be $\frac{1-i}{2}|10\rangle + \frac{1+i}{2}|11\rangle$. The circuit has 7 gates and 6 levels (depth). The term level has the same meaning that the depth and it is defined in Definition 2.8.

Definition 2.8. *The level of a circuit can be defined as the number of times a set of gates are applied in the circuit, i. e., gates can be parallelized when acting on different qubits, thus, the gates are being applied at the same time (counting as a single level).*

There are different methodologies to synthesize a quantum circuit. In the research of Soeken *et al.* (2016) the quantum circuit is realized from the specification. However, other approach is to map the reversible circuit to a quantum circuit using some equivalences between the reversible and quantum gates, as presented in Miller, Soeken and Drechsler (2014). Note that a quantum circuit is reversible, thus, every reversible circuit can be mapped to a quantum circuit. This mapping is done by decomposing each reversible gate in the circuit into a cascade of quantum gates.

After synthesize a quantum circuit, optimizations can be used in order to reduce its size. Some equivalences that acts on single qubits can be used to accomplish that. Figure 8 shows some identities equivalences.

Figure 8 – Identity equivalences of quantum gates.

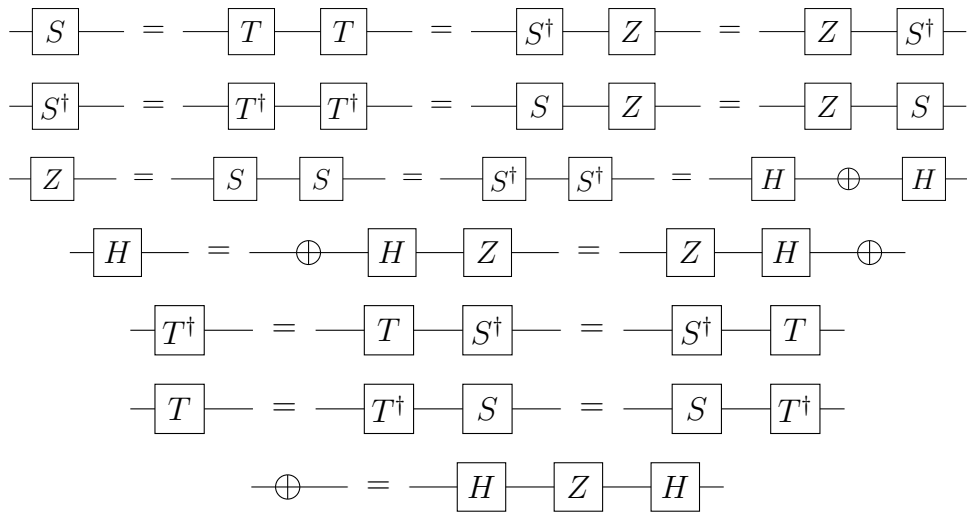


Source: Prepared by the author.

Other optimizations that can be used are presented in Figure 9. Since the gates T, $T^\dagger$, S, $S^\dagger$, Z can commute, the goal is try to move these gates together and remove them when possible. These equivalences can be checked using the approach presented in Appendix A.

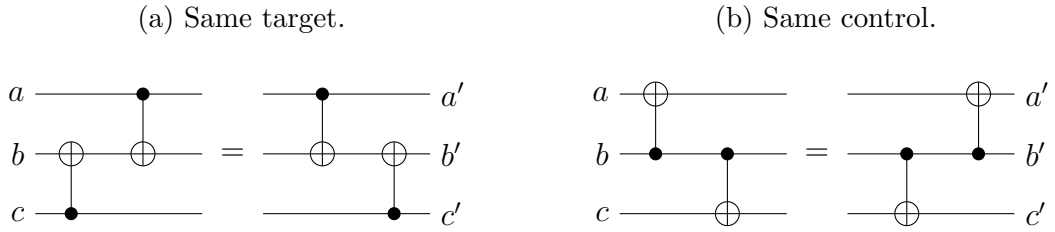
Some CNOT gates movements also can be used to optimize the circuits. The CNOT gates can commute if they have same target with different controls or same controls with different targets. These movements are presented in Figure 10.

Figure 9 – Optimization equivalences for quantum gates.



Source: Prepared by the author.

Figure 10 – CNOT gates movements.



Source: Prepared by the author.

The algorithms developed in the research area of reversible logic and quantum computation can be programed on top of a framework called RevKit. The next section RevKit and a framework developed by IBM, called Qiskit, will be briefly presented.

2.3 COMPUTATIONAL TOOLS

RevKit is modular framework developed with programming language C++ that is used to program algorithms regarding reversible logic and quantum computation. This framework is open source and has features of synthesis, optimization, circuit verification, among others (SOEKEN *et al.*, 2008).

The structure provided by RevKit helps in the implementation of new algorithms, due to its modular structure, simplifying the addition of new features. The framework works via command line interface, in which the available commands can be used. To evaluate the algorithms developed, a series of benchmarks were created, called RevLib, and became

available online (WILLE *et al.*, 2008).

Another framework used in this work was made available by IBM and is called QISKit (Quantum Information Software Kit). QISKit aids in the development of quantum circuits making it possible to execute them on quantum computers made available in the cloud by IBM. Another feature of this tool is the mapping of quantum circuits to the architecture of the project IBM Q (IBM, e).

This chapter presented the background about the reversible logic and quantum computation. The covered theory will be used to map reversible and quantum circuits to quantum processors available by IBM in the project IBM Q. The next chapter presents the characteristics and challenges of the IBM Q project. Also, the proposed approach to solve the problems raised by the constraints in the IBM quantum processors.

3 CNOT MAPPINGS FOR IBM QUANTUM COMPUTERS

IBM has a project, named IBM Q, that provides access to quantum computers, which have different architectures with different set of constraints. This chapter presents these quantum computers and how the architecture constraints are addressed. Section 3.1 describes the IBM quantum computers along with the characteristics and restrictions. Section 3.2 shows the proposed approach to address these restrictions followed by Section 3.3 with the experimental results of the proposed approach.

3.1 IBM QX ARCHITECTURES

IBM has a project that provides access to quantum computers. The purpose of this access is to *experiment* with in real devices. There are architectures with 5, 16, and 20 qubits available (IBM, e).

The architectures are still in development and have limitations regarding the number of qubits and the fidelity, i.e., the gates in the real devices are susceptible to errors, thus, the output may not be the expected one.

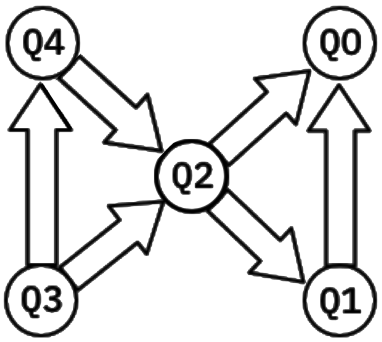
Another limitation of the architectures is the types of quantum gates that can be used in the circuits. All architectures support quantum circuits realized with the universal single qubit gate $U(\theta, \phi, \lambda) = R_z(\phi)R_y(\theta)R_z(\lambda)$ which is a rotation around the y-axis and two rotations around z-axis. Therefore, any single qubit gate can be realized by the parameters (θ, ϕ, λ), e.g., the NOT, H, S, $S^\dagger$, T and $T^\dagger$, Pauli-X, Pauli-Y, Pauli-Z (IBM, a).

The IBM QX architectures also support the CNOT gate, although, the availability is limited. The architectures restrict the interaction of the physical qubits, i.e., it is only possible to apply a CNOT gate to a defined set of qubits. These CNOT restrictions are expressed in a coupling graph. Figure 11 shows the 5 and 16 qubits IBM architectures (coupling graphs), named IBM QX4 and IBM QX5, respectively (IBM, c, d).

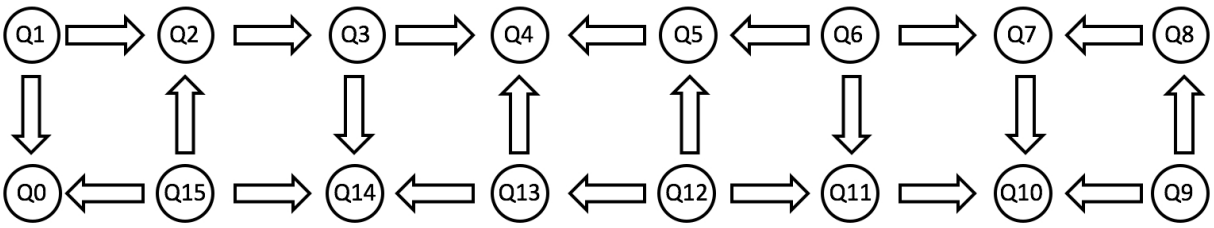
The circles in Figure 11 represent the physical qubits $(Q_0, Q_1, Q_2, \ldots, Q_{15})$ and the arrows indicate a possible implementation of a CNOT gate. The base of the arrow represents the control qubit and the tip of the arrow represents the target qubit. Therefore,

Figure 11 – IBM QX4 and IBM QX5 architectures from project IBM Q.

(a) IBM QX4.



(b) IBM QX5.



Source: Adapted from IBM (a).

the CNOT must be placed in these qubits in order to be implemented. To represent a CNOT will be used the notation CNOT(a, b), where a is the control qubit and b is the target. For example, a possible implementation of a CNOT in the IBM QX5 has the control in qubit Q_1 and the target in qubit Q_0, represented by CNOT(Q_1, Q_0). The architecture constraints regarding the CNOT gates is defined in Definition 3.1.

Definition 3.1. *Let $Q = \{q_0, q_1, q_2, ..., q_n\}$ be a set of the available physical qubits of n qubits architecture. Also, let $Cp = \{(q_a, q_b), (q_c, q_d), ..., (q_k, q_l)\}$ be the coupling graph of available CNOT gates in the architecture. The CNOT(q_i,q_j) is possible if (q_i,q_j) $\in$ Cp.*

The CNOT that is not implementable in the architecture can be referred as misplaced. Note that the number of possible CNOT gates in a architecture with 5 qubits is 20. However, only 6 CNOT gates are available in IBM QX4 (Figure 11(a)). For an architecture with 16 qubits, there are 240 possible CNOT gates and only 22 available in IBM QX5 (Figure 11(b)). This shows the challenges to realize a quantum circuit in a real device and raise a problem to be solved defined in Definition 3.2.

Definition 3.2. *Let $Q = \{q_0, q_1, q_2, ..., q_n\}$ be a set of n physical qubits and $Cp = \{(q_a, q_b), (q_c, q_d), ..., (q_k, q_l)\}$ be a set of a coupling graph of available CNOT gates. Then, let*

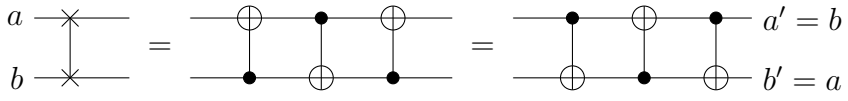
$G = \{g_0, g_1, g_2, ..., g_m\}$ be a set of m logical qubits of a circuit with $m \leq n$. A CNOT(i,j) acts on the logical qubits (i,j) and must be mapped to a pair of physical qubits (r,s).

The challenge is to find an efficient way to map the misplaced CNOT gates, i.e., to transform the non-available CNOT gates to be used in one of these architectures. The goal is to map the logical qubits to the physical ones in such way that the number of CNOT gates available in the target architecture is maximized. This can be done by mapping the logical qubits to the available physical ones in a way that reduces the number of misplaced gates. However, the remaining misplaced gates must be mapped in order to obey the architectures constraints. The next section presents a mapping strategy for these CNOT gates.

3.2 CNOT GATES MAPPINGS

Since not all CNOT gates are available in the IBM QX architectures (IBM QX refers to a generic architecture), it is necessary to find alternatives to map them using the available gates. One of the alternatives is to use SWAP gates, which interchange the values of two qubits. Figure 12 shows the SWAP gate.

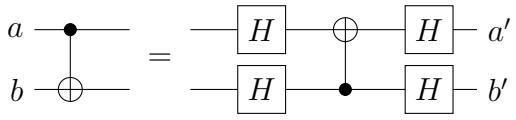
Figure 12 – SWAP gate.



Source: Prepared by the author.

It is possible to move the control or the target qubit to a given qubit in order to map a CNOT gate by using a SWAP gate. However, it must be ensured that the CNOT gates used in the SWAP satisfies the constraints of the respective target architecture. This can be done by flipping a CNOT adding 4 Hadamard gates as shown in Figure 13.

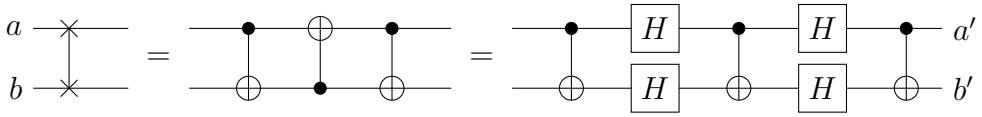
Figure 13 – Swapping control and target qubits of a CNOT.



Source: Prepared by the author.

Applying the operation from Figure 13 to the SWAP gate from Figure 12, the CNOT gates can be realized with the control and targets in the same qubits, as shown in Figure 14.

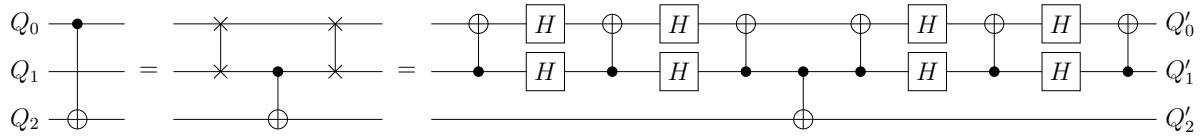
Figure 14 – SWAP gate with changed CNOT.



Source: Prepared by the author.

For example, to map the CNOT(Q_0, Q_2), which is not available in IBM QX5, Q_0 can be moved to Q_1. This movement results in the circuit shown in Figure 15.

Figure 15 – Possible CNOT(Q_0, Q_2) mapping.



Source: Prepared by the author.

The SWAP gate will be applied again after the CNOT gate to ensure that the qubits are not permuted. The inclusion of the SWAP gates causes the circuit to increase in size. In this case, 14 gates and 10 levels must be added to the circuit to map the CNOT(Q_0, Q_2) into IBM QX5.

In order to find a circuit with as few gates as possible, which satisfies the IBM QX architecture constrains, some qubits movements are defined and optimized to be used in the CNOT gates mappings.

3.2.1 Types of movements

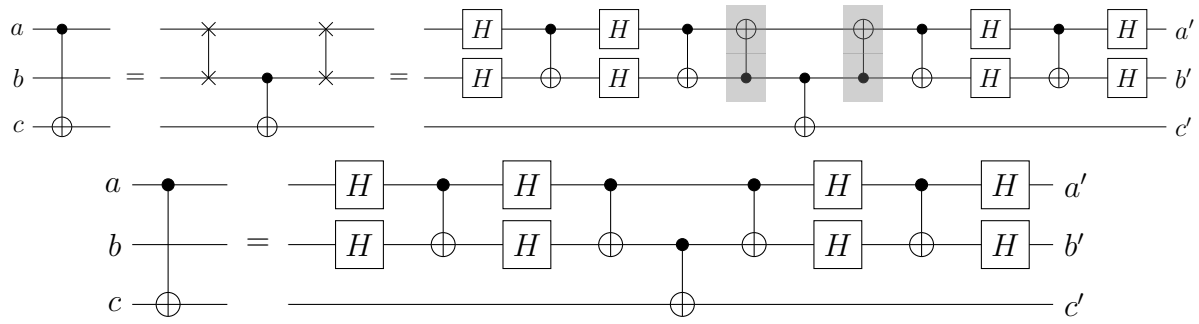
To evaluate the mappings, two types of cost metrics are used: the number of gates and levels in the circuit. To map an arbitrary CNOT(a, c) to an IBM QX architecture, there must be a path linking qubits from a to c. Note that in the graph (see Figure 11(b)) the path does not have to follow the direction of the arrows.

There are two choices: move the control a towards the target c, or move the target c towards the control a. Assume qubit a has to move to b, then there are two cases: either CNOT(a, b) or CNOT(b, a) exists. From this idea we can define the following movements.

3.2.1.1 Control movements

For this movement the CNOT(a, b) must be available in the IBM QX architecture and the control qubit need to be moved from qubit a to qubit b. Figure 16 shows the movement. The highlighted CNOT gates can be moved together and removed.

Figure 16 – Movement of control qubit using the CNOT(a, b).



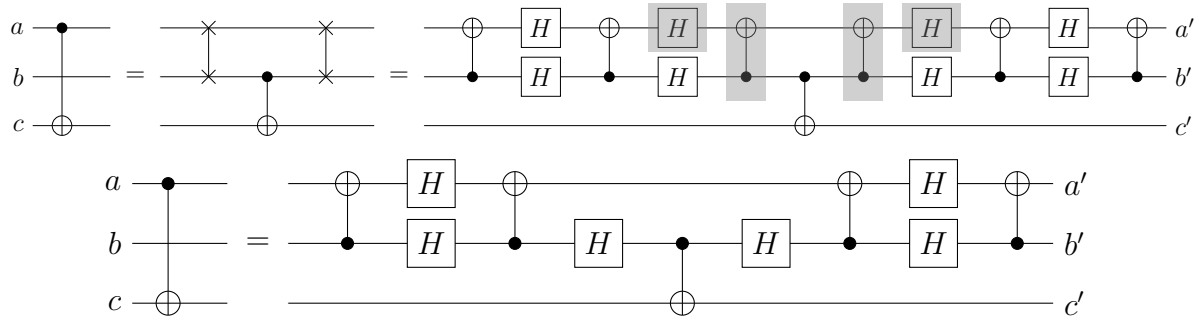
Source: Prepared by the author.

From the operation performed by the gates in Figure 16 two control movements are defined in Definition 3.3.

Definition 3.3. *The operation performed by the first six gates in Figure 16 is called CAB(a, b) with cost 6 and 4 levels. Similarly, from the operation performed by the last six gates in Figure 16 is called CAB$^\dagger(a, b)$ with the same cost.*

The † symbol represents the reverse movement. Note that the cost is the number of additional gates and levels. Therefore, an available CNOT has cost 0. The other possible control movement is to move the control from a to b, however the CNOT(b, a) must be available in the IBM QX architecture. Figure 17 shows the movement. The highlighted gates can be moved together and eliminated.

Figure 17 – Movement of control qubit using the CNOT(b, a).



Source: Prepared by the author.

From the operations performed by the gates in Figure 17 two control movements are defined in Definition 3.4.

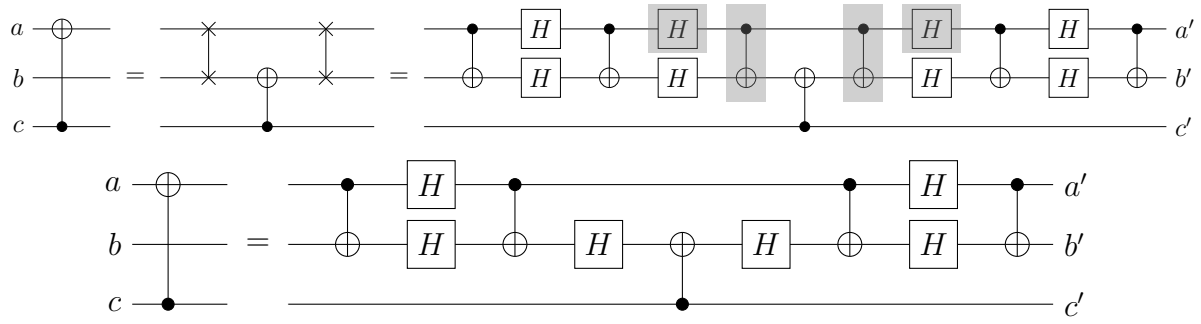
Definition 3.4. *The operation performed by the first five gates in Figure 17 is called CBA(a, b) with cost 5 and 4 levels. In the same way, the operation performed by the last five gates in Figure 17 is called CBA$^\dagger(a, b)$ with the same cost.*

The next set of possible movements is to move the target qubit.

3.2.1.2 Target movements

For this movement the CNOT(a, b) must be available in the IBM QX architecture and the target qubit need to be moved from a to b. The movement is shown in Figure 18. The highlighted gates of the SWAP gate decomposition can be moved together and removed.

Figure 18 – Movement of target qubit using the CNOT(a, b).



Source: Prepared by the author.

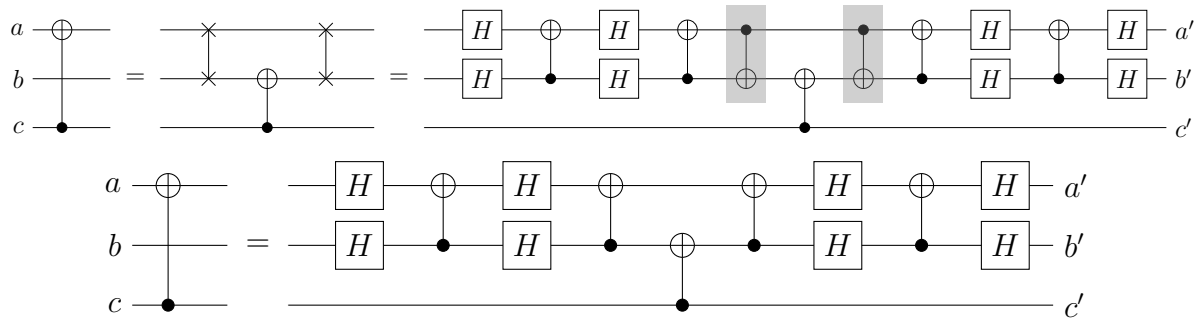
From the operations performed by the gates in Figure 18 two target movements are defined in Definition 3.5.

Definition 3.5. *The operation performed by the first five gates in Figure 18 is called TAB(a, b) with cost 5 and 4 levels. Similarly, the operation performed by the last five gates in Figure 18 is called TAB$^\dagger(a, b)$ with the same cost.*

Another possible target movement is to move the target from a to b, however CNOT(b, a) must be available in the IBM QX architecture. The movement is shown in Figure 19. The highlighted CNOT gates can be moved together and removed.

From the operations performed by the gates in Figure 19 two target movements are defined in Definition 3.6.

Figure 19 – Movement of target qubit using the CNOT(b, a).



Source: Prepared by the author.

Definition 3.6. *The operation performed by the first six gates in Figure 19 is called TBA(a, b) with cost 6 and 4 levels. Also, from the operation performed by the last six gates in Figure 19 is called TBA$^\dagger(a, b)$ with the same cost.*

Applying the defined qubits movements it possible to find a mapping for the non-available CNOT gates in the IBM QX architectures. However, there are some cases that a movement is not required and a mapping with reduced cost can be applied.

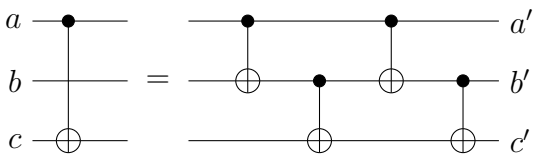
3.2.2 Template based mappings

If CNOT(a, b) must be mapped and the CNOT(b, a) is available in the IBM QX architecture, then the control and target qubits can be swapped. This can be done with the operation from Figure 13. From this operation is defined a mapping in Definition 3.7.

Definition 3.7. *The operation of flipping a CNOT gate with 4 Hadamard is called FLIP(a, b), which has cost 4 and 2 levels.*

For the next case assume that a given CNOT(a, c) needs to be mapped. A possible mapping for this CNOT is shown in Figure 20 and defined in Definition 3.8. For this mapping the CNOT(a, b) and CNOT(b, c) must be available in the IBM QX architecture.

Figure 20 – CNOT3(a, b, c) mapping.

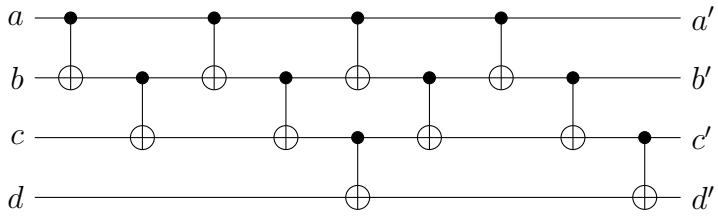


Source: Prepared by the author.

Definition 3.8. *The mapping from Figure 20 is called CNOT3(a, b, c) which has cost 3 and 3 levels.*

The movement CNOT3(a, b, c) will always be possible when two CNOT gates in sequence are available (the two arrows in the coupling graph are pointing to the same direction, from control to target qubits) in the architecture. Also, it can be expanded if a path of more CNOT gates are available in sequence. Figure 21 shows an example of this expansion for two CNOT gates in sequence, for that the CNOT(Q_a, Q_b), CNOT(Q_b, Q_c), CNOT(Q_c, Q_d) exists.

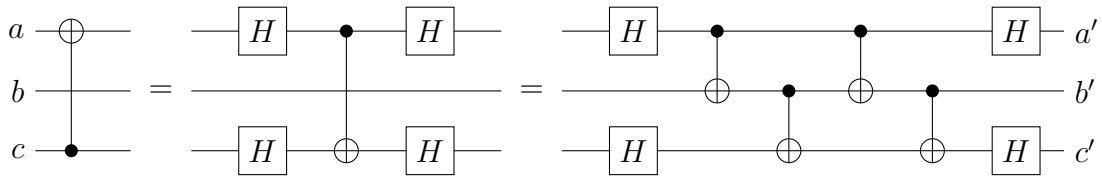
Figure 21 – CNOT3(a, b, c) mapping expanded.



Source: Prepared by the author.

The last case is when the CNOT(c, a) needs to be mapped. If in this case the CNOT(a, b) and CNOT(b, c) are available in the IBM QX architecture, it is possible to swap the CNOT with 4 Hadamard gates and use the CNOT3(a, b, c) to map it. The mapping is shown in Figure 22 and defined in Definition 3.9. From this operation a mapping called CNOT3$^\dagger(a, b, c)$ with cost 7 and 5 levels is defined.

Figure 22 – CNOT3$^\dagger(a, b, c)$ mapping.



Source: Prepared by the author.

Definition 3.9. *The mapping from Figure 22 is called CNOT3$^\dagger(a, b, c)$ which has cost 3 and 3 levels.*

The idea of expanding the mapping if CNOT gates in sequence are found can be used with this mapping as well. In this case, the difference is that the sequence of CNOT gates are swapped in the coupling graph (the opposite direction of arrows, from the target qubit to control qubit).

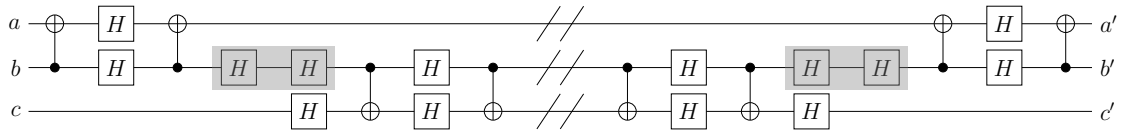
The qubits movements and template based mappings defined above can be used to find the mappings of the CNOT gates that not satisfies the IBM QX architectures constraints. However, some combinations of movements in the mappings can be optimized.

3.2.3 Movements optimizations

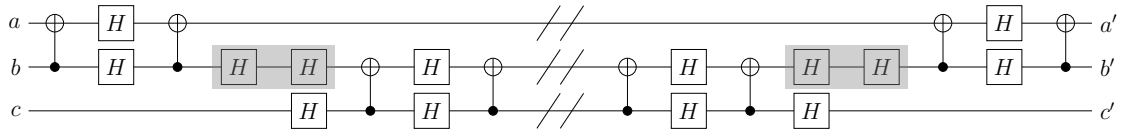
The movements defined in this previous subsection consist of a combination of CNOT and Hadamard gates. Since the Hadamard is self-inverse, two adjacent Hadamard gates, acting on the same qubit can be removed.

Thus, the combination of movements where one ends with a Hadamard and the next one begins with a Hadamard can be optimized. The following combinations of movements exhibit this characteristic and can be optimized:

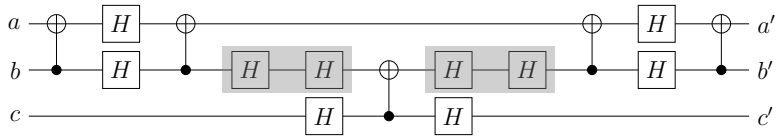
1. CBA(a, b), CAB(b, c) ... CAB$^\dagger(b, c)$, CBA$^\dagger(a, b)$



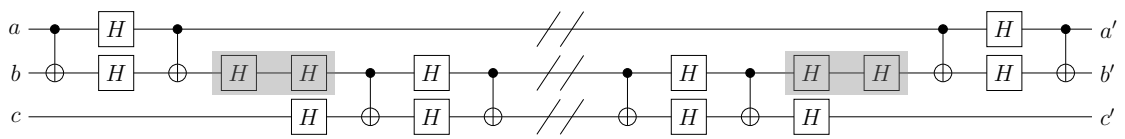
2. CBA(a, b), TBA(b, c) ... TBA$^\dagger(b, c)$, CBA$^\dagger(a, b)$



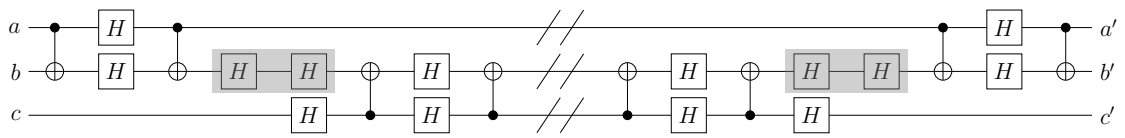
3. CBA(a, b), FLIP(b, c), CBA$^\dagger(a, b)$



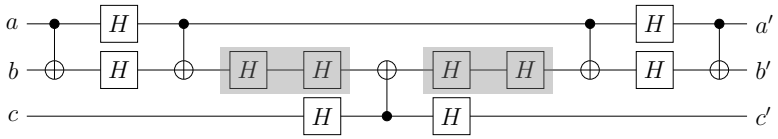
4. TAB(a, b), CAB(b, c) ... CAB$^\dagger(b, c)$, TAB$^\dagger(a, b)$



5. TAB(a, b), TBA(b, c) ... TBA$^\dagger(b, c)$, TAB$^\dagger(a, b)$



6. TAB(a, b), FLIP(b, c), TAB$^{\dagger}(a, b)$



Every time one of these 6 combinations of movements occurs, 4 Hadamard gates can be removed from the circuit and the number of levels can be reduced by 2. Therefore, if a transformation has 3 combinations, 12 gates and 6 levels can be removed.

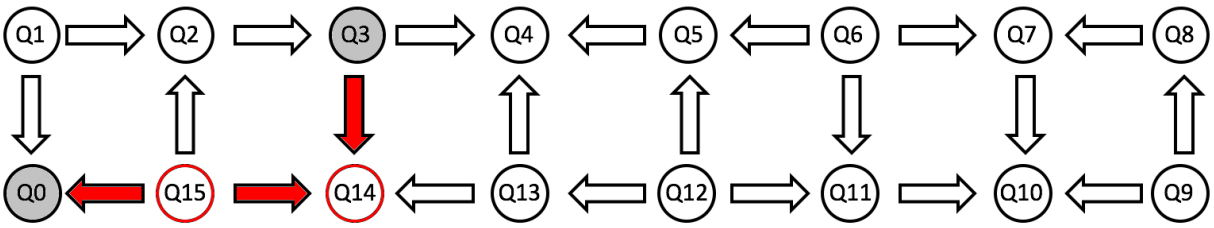
The goal of the qubits movements is to find lower cost mappings, i.e., to find the path between the control and target qubits of a CNOT with fewer additional gates.

3.2.4 Finding CNOT gates mappings

The CNOT(Q_0, Q_3) will be used as an example to demonstrate the process that determines the lower cost mapping for each CNOT using the qubits movements. This mapping has the IBM QX5 as target architecture. The CNOT(Q_0, Q_3) can be mapped with different costs and the goal is to find the mapping with the lowest one.

The first path to be considered is $\{Q_0, Q_{15}, Q_{14}, Q_3\}$ and it is shown in Figure 23.

Figure 23 – Path $\{Q_0, Q_{15}, Q_{14}, Q_3\}$ to map CNOT(Q_0, Q_3).



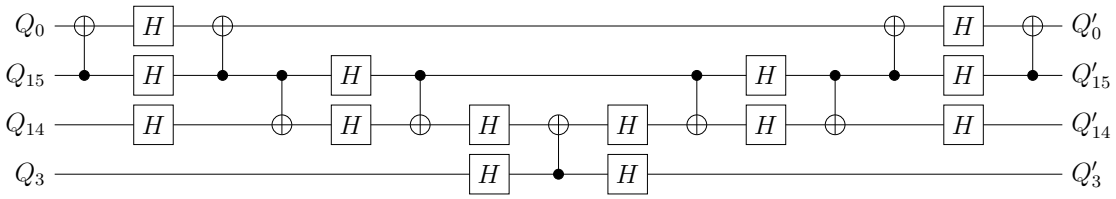
Source: Prepared by the author.

There are different possible mappings for the CNOT(Q_0, Q_3) using this path. Move the control towards the target, move the target towards the control or move the control and target towards each other. These possible mappings for this path are:

1. The control qubit is moved towards the target qubit;

   CBA$(0, 15)$, CAB$(15, 14)$, FLIP$(14, 3)$, CAB$^{\dagger}(15, 14)$, CBA$^{\dagger}(0, 15)$

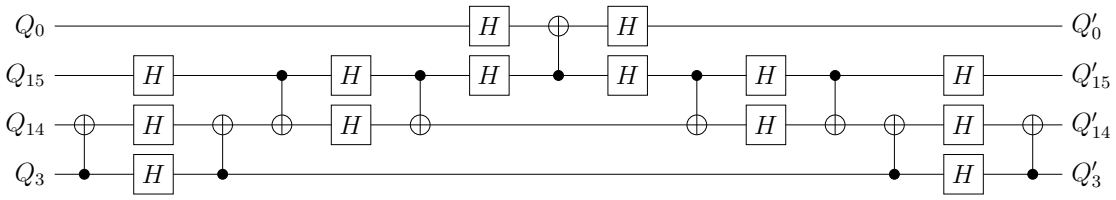
   - Cost: 22 gates and 14 levels.



2. The target qubit is moved towards the control qubit;

   TAB$(3, 14)$, TBA$(14, 15)$, FLIP$(0, 15)$, TBA$^\dagger(14, 15)$, TAB$^\dagger(3, 14)$

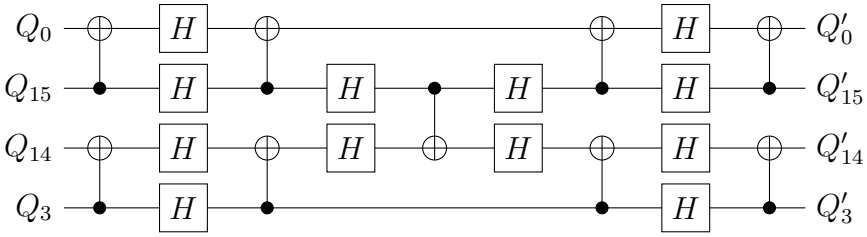
   - Cost: 22 gates and 14 levels.



3. The control and the target qubits are moved towards each other.

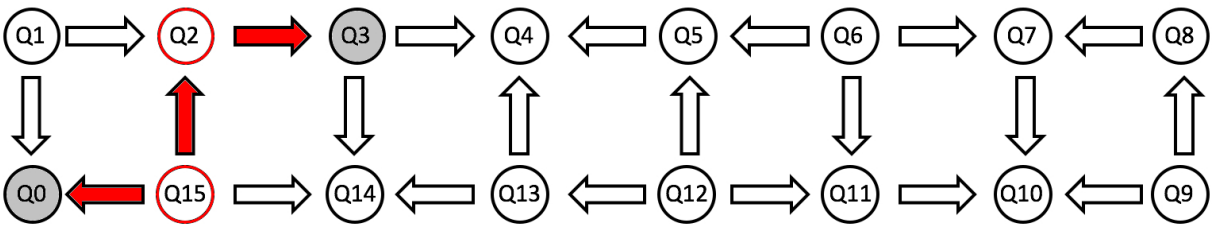
   CBA$(0, 15)$, TAB$(3, 14)$, CNOT$(15, 14)$, TAB$^\dagger(3, 14)$, CBA$^\dagger(0, 15)$

   - Cost: 20 gates and 8 levels.



The second path to be considered is $\{Q_0, Q_{15}, Q_2, Q_3\}$. Figure 24 shows this path.

Figure 24 – Path $\{Q_0, Q_{15}, Q_2, Q_3\}$ to map CNOT(Q_0, Q_3).



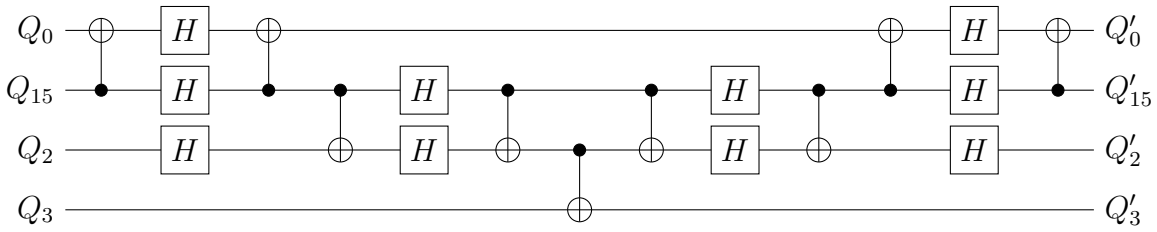
Source: Prepared by the author.

The same idea from the last path is used. Try all combinations moving the control and target. The possible mappings for this path are:

4. The control qubit is moved towards the target qubit;

   CBA$(0, 15)$, CAB$(15, 2)$, CNOT$(2, 3)$, CAB$^\dagger(15, 2)$, CBA$^\dagger(0, 15)$

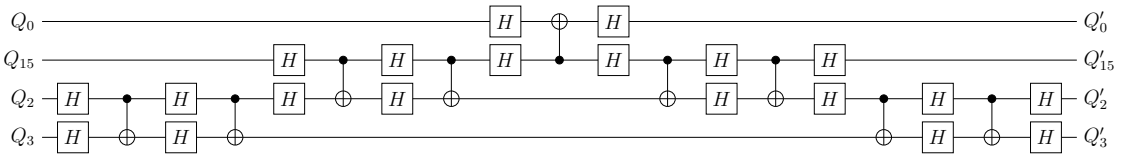
   - Cost: 18 gates and 12 levels.



5. The target qubit is moved towards the control qubit;

   TBA$(3, 2)$, TBA$(2, 15)$, FLIP$(0, 15)$, TBA$^\dagger(2, 15)$, TBA$^\dagger(3, 2)$

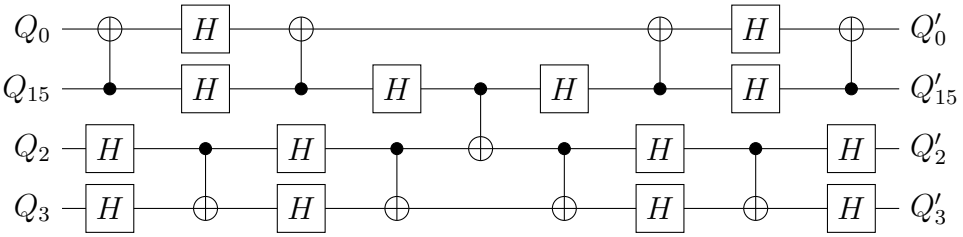
   - Cost: 28 gates and 18 levels.



6. The control and the target qubits are moved towards each other.

   CBA$(0, 15)$, TBA$(3, 2)$, CNOT$(15, 2)$, TBA$^\dagger(3, 2)$, CBA$^\dagger(0, 15)$

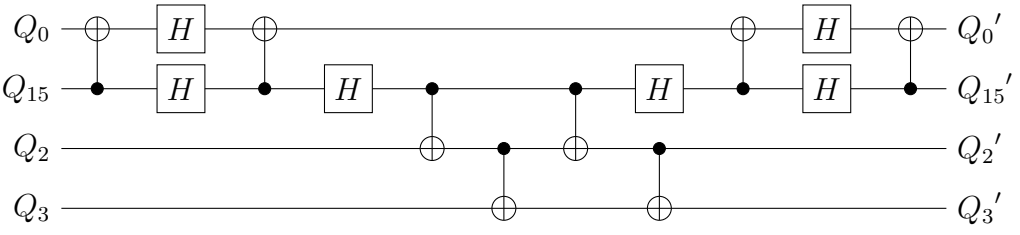
   - Cost: 22 gates and 8 levels.



7. The control is moved to Q_{15} and the CNOT3$(15, 2, 3)$ mapping is used.

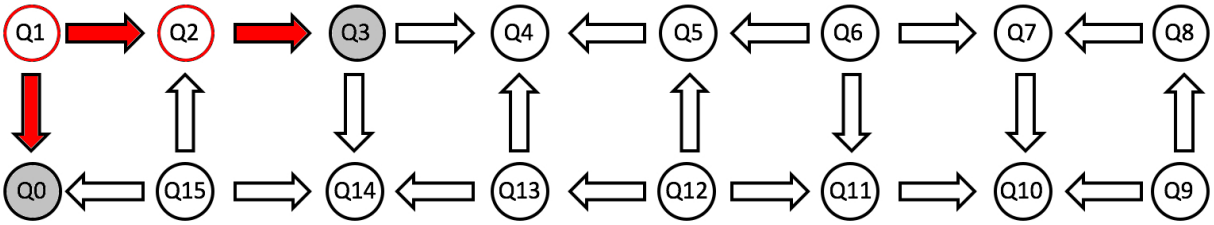
   CBA$(0, 15)$, CNOT3$(15, 2, 3)$, CBA$^\dagger(0, 15)$

   - Cost: 13 gates and 10 levels.



The last path with the same length of the other two is $\{Q_0, Q_1, Q_2, Q_3\}$. This path is shown in Figure 25.

Figure 25 – Path $\{Q_0, Q_1, Q_2, Q_3\}$ to map CNOT(Q_0, Q_3).



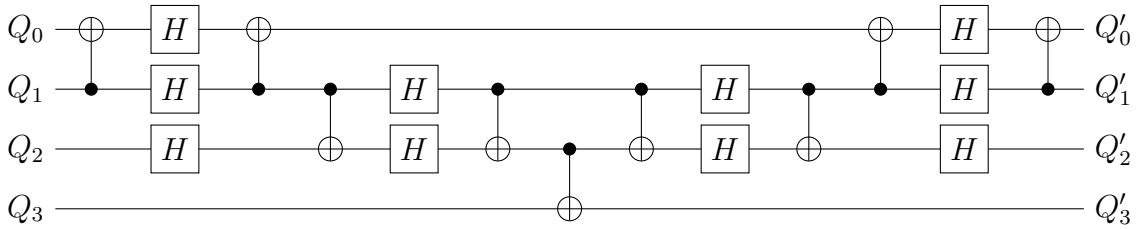
Source: Prepared by the author.

Trying all the possible mappings for this path:

8. The control qubit is moved towards the target qubit;

   CBA$(0, 1)$, CAB$(1, 2)$, CNOT$(2, 3)$, CAB$^\dagger(1, 2)$(1,2), CBA$^\dagger(0, 1)$

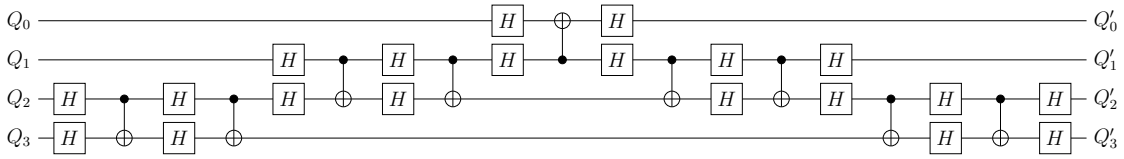
   - Cost: 18 gates and 12 levels.



9. The target qubit is moved towards the control qubit;

   TBA$(3, 2)$, TBA$(2, 1)$, FLIP$(0, 1)$, TBA$^\dagger(2, 1)$, TBA$^\dagger(3, 2)$

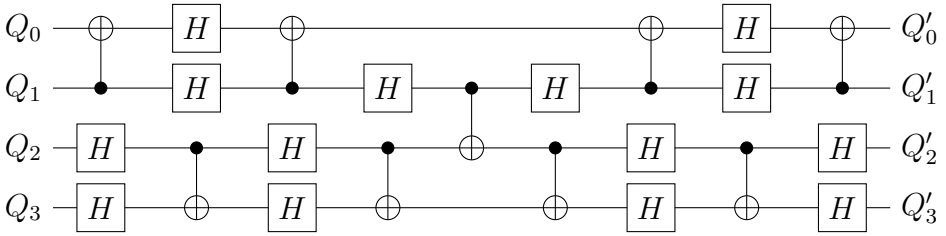
   - Cost: 28 gates and 18 levels.



10. The control and the target qubits are moved towards each other.

    CBA$(0, 1)$, TBA$(3, 2)$, CNOT$(1, 2)$, TBA$^\dagger(3, 2)$, CBA$^\dagger(0, 1)$

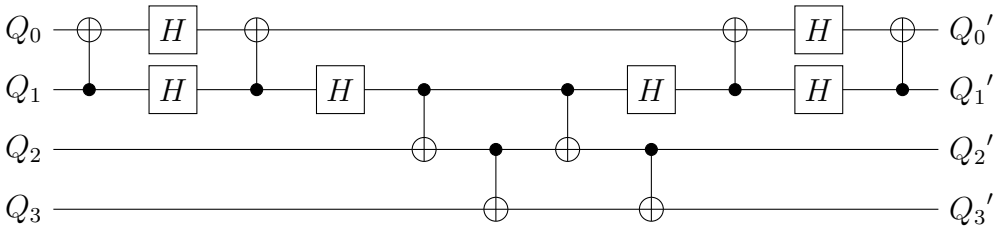
    - Cost: 22 gates and 8 levels.



11. The control is moved to Q_1 and the CNOT3$(1, 2, 3)$ mapping is used.

    CBA$(0, 1)$, CNOT3$(1, 2, 3)$, CBA$^{\dagger}(0, 1)$

    - Cost: 13 gates and 10 levels.



There are other paths that can be used to do the mapping of the CNOT(Q_0, Q_3), but it is clear that those paths will result in a higher cost. This is due to the longer path to be mapped.

The mappings with the lowest number of gates were the ones using the CNOT3(a, b, c), numbers 7 and 11, with cost 13. The mappings 3, 6 and 10 have the lowest number of levels, with 8 levels. However, these mappings costs more than 13. Since the mapping has the priority regarding the number of gates, the mappings 7 or 11 can be used to map the CNOT(Q_0, Q_3).

From the strategy presented, a matrix with the cost of each possible CNOT in an IBM QX architecture can be generated.

3.2.5 Generating architecture cost matrix

As an example, it will be shown how the cost matrix can be generated for the IBM QX4 (Figure 11(a)). Since the architecture has only 5 qubits (20 possible CNOT gates), it is possible to enumerate them all.

There are CNOT gates that do not need to be mapped. They are already available in the architecture and have cost 0. In the IBM QX4, these CNOT gates are:

- CNOT(Q_1, Q_0); CNOT(Q_2, Q_0); CNOT(Q_2, Q_1);

- CNOT(Q_3, Q_2); CNOT(Q_3, Q_4); CNOT(Q_4, Q_2).

There are CNOT gates that only need to swap the control and target qubits to be mapped. In these cases the movement that will be applied is the FLIP(a, b). The cost of these mappings is 4. The CNOT gates that can be mapped with this movement are:

- CNOT(Q_0, Q_1); CNOT(Q_0, Q_2); CNOT(Q_1, Q_2);
- CNOT(Q_2, Q_3); CNOT(Q_2, Q_4); CNOT(Q_4, Q_3).

There are CNOT gates that can be mapped using the CNOT3(a, b, c). The cost of these mappings is 3. These CNOT gates are:

- CNOT(Q_3, Q_0); CNOT(Q_3, Q_1);
- CNOT(Q_4, Q_0); CNOT(Q_4, Q_1).

The next set of CNOT mappings are the ones that are mapped with CNOT3†(a, b, c). These mappings have cost 7. The CNOT gates that have these characteristics are:

- CNOT(Q_0, Q_3); CNOT(Q_0, Q_4);
- CNOT(Q_1, Q_3); CNOT(Q_1, Q_4).

A matrix with the costs of each mapping can be generated from the mappings. The cost matrix for IBM QX4 is shown in Figure 26. The "-" in the matrix indicates that CNOT(i, i) is not possible.

Figure 26 – IBM QX4 architecture cost matrix.

$$T = \begin{bmatrix} - & 4 & 4 & 7 & 7 \\ 0 & - & 4 & 7 & 7 \\ 0 & 0 & - & 4 & 4 \\ 3 & 3 & 0 & - & 0 \\ 3 & 3 & 0 & 4 & - \end{bmatrix}$$

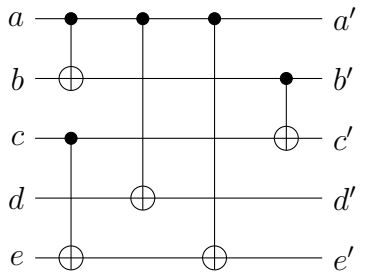
Source: Prepared by the author.

In order to automate the search of the best mappings of each CNOT, the strategy presented was programmed on top of RevKit (SOEKEN *et al.*, 2008). This strategy can be easily adaptable to other architectures with different coupling graphs. The architecture is read as a graph into RevKit and the matrix cost generation can be done only once per architecture. The next subsection presents an example of the proposed CNOT mapping in a circuit.

3.2.6 Example of CNOT gates mappings application

To illustrate the application of the CNOT mapping, the circuit of Figure 27 will be used. The goal is to map this circuit to the IBM QX4 architecture. The original circuit has 5 gates and 3 levels.

Figure 27 – Example circuit for the application of the CNOT mappings.



Source: Prepared by the author.

The trivial permutation is to map the logical qubits $\{a, b, c, d, e\}$ to the physical ones $\{Q_0, Q_1, Q_2, Q_3, Q_4\}$. It is possible to calculate the number of gates for this circuit using the matrix cost in Figure 26. To do this a matrix with the CNOT gates of the circuit can be generated. This matrix is presented in Figure 28.

Figure 28 – CNOT gates matrix of example circuit.

$$T = \begin{bmatrix} - & 1 & 0 & 1 & 1 \\ 0 & - & 1 & 0 & 0 \\ 0 & 0 & - & 0 & 1 \\ 0 & 0 & 0 & - & 0 \\ 0 & 0 & 0 & 0 & - \end{bmatrix}$$

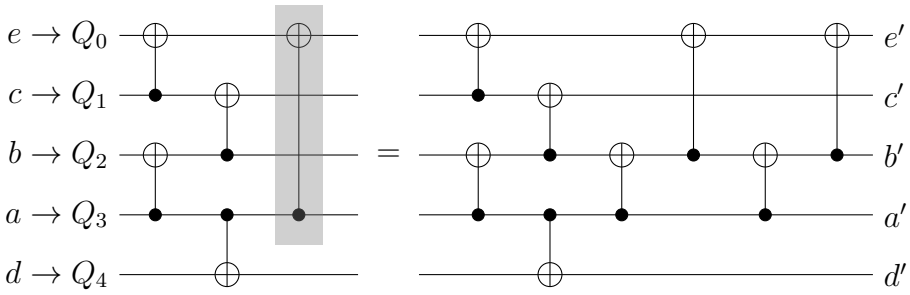
Source: Prepared by the author.

Multiplying the matrix cost with the matrix of CNOT is possible to calculate the final cost of the circuit for this allocation. Note that the reduction with post-mapping optimization is not taken into consideration. The final cost for this permutation will be 26 additional gates.

In order to find the lowest cost permutation, all permutations must be found. IBM QX4 has 5 qubits, hence, it will be 120 (5!) permutations. The best permutation for this circuit will be the qubits $\{a, b, c, d, e\}$ mapped to $\{Q_3, Q_2, Q_1, Q_4, Q_0\}$. This permutation will add only 3 gates to the circuit. The permuted circuit is presented in Figure 29.

The highlighted CNOT is the only misplaced gate regarding the IBM QX4 architecture. The cost for this gate is 3, i.e., 3 additional gates are required to map the

Figure 29 – Example circuit mapped to $\{Q_3, Q_2, Q_1, Q_4, Q_0\}$.



Source: Prepared by the author.

respective CNOT to the IBM QX4. The final circuit has 8 gates and 6 levels, compared with the trivial permutation is a reduction of 88% in the additional gates.

In order to evaluate the proposed mapping strategy, some benchmarks were used and compared with other existing approaches. Results of this evaluation are presented in the next section.

3.3 EXPERIMENTAL RESULTS OF CNOT GATES MAPPINGS

To evaluate the proposed mapping strategy two existing approaches were used. One is the algorithm proposed by Zulehner, Paler and Wille (2019) that also realizes the mapping of Clifford+T circuits to IBM QX architectures. The other algorithm is Qiskit, the framework for quantum computing available from IBM (ALEKSANDROWICZ; ALEXANDER; BARKOUTSOS,). Qiskit uses the *Open Quantum Assembly Language* to describe the quantum circuits (CROSS *et al.*, 2017). From now on, the algorithm from Zulehner, Paler and Wille (2019) will be referred to as Standalone algorithm.

The first evaluation was done with some benchmarks targeting the IBM QX4. As Qiskit optimizes the circuit and realize it in terms of CNOT and U gates, this evaluation was done in two steps. In the first step the circuits found with the proposed mapping strategy and with the Standalone algorithm were not optimized and compared. In the second step the circuits were optimized and compared with each other. This evaluation in two steps is due the work of Zulehner, Paler and Wille (2019) was presented with Qiskit to fully optimize the algorithm. Also, the proposed mapping strategy was compared with Qiskit.

The second evaluation was done with one circuit as case of study comparing the proposed mapping strategy with the two approaches targeting the IBM QX5.

3.3.1 Benchmarks

To evaluate the proposed mappings the IBM QX4 was used as target architecture. The benchmark circuits were taken from Zulehner, Paler and Wille (2019) and from Revlib in Wille *et al.* (2008).

The first test was to compare the proposed mapping strategy with the Standalone algorithm. The number of gates and the number of levels were used as comparison criteria. Since the IBM QX4 has 5 qubits, it is feasible to find all the 120 permutations and choose the lowest cost one. The priority in the cost is the number of gates of the circuit. Table 11 shows the benchmarks results. The first column, **Benchmarks**, presents the name of each circuit used. The columns **Original** show the characteristics of the original circuit. The column, **Qubits**, presents the number of qubits. The columns **Gates** is the number of gates of the respective circuits and the columns **Levels** is the number of levels. The columns **Standalone** present the circuits characteristics obtained with the Standalone algorithm. Similarly, the columns **CNOT Mappings** show the circuits characteristics obtained with the proposed mapping strategy. The columns the **%Improvement** show, in percentage, the improvement of the mapping strategy compared with the Standalone algorithm.

From Table 11 it can be seen that the proposed mapping strategy found circuits with fewer gates than the Standalone algorithm. However, in some circuits the number of levels was worse. For example, the Standalone algorithm found a mapping with fewer levels in the circuit *4gt5_75*.

The proposed mapping strategy obtained circuits with up to 50% fewer gates and 45.98% fewer levels than the Standalone algorithm. This may be due the fact that the architecture has only 5 qubits and the best permutation was found for each circuit. Along with that, the cost to map the CNOT gates, in the worse case, was to move only one qubit (due to the IBM QX4 architecture).

On the other hand, the Standalone algorithm finds a permutation for the first *layer* (set of parallelized CNOT gates) of the circuit and then go through the circuit mapping the CNOT gates using a look-ahead scheme searching all SWAP combinations. This may result in a circuit with more misplaced CNOT gates as the permutation is changing.

Another difference in the algorithms is the permutation at the output of the circuit. In the proposed mapping strategy, the permutation is the same at the input and output of the circuit. However, in the Standalone that is not true. The CNOT mapping through

Table 11 – CNOT Mappings strategy compared with Standalone without Qiskit optimization.

	Original			Standalone		CNOT Mappings		%Improvement	
Benchmarks	**Qubits**	**Gates**	**Levels**	**Gates**	**Levels**	**Gates**	**Levels**	**Gates**	**Levels**
mod5d1_63	5	22	12	76	49	38	27	50.00	44.90
alu-v3_34	5	52	29	154	87	83	47	46.10	45.98
4mod5-v1_22	5	21	12	61	38	34	22	44.26	42.11
4mod5-v0_18	5	69	39	202	116	124	73	38.61	37.07
4mod5-v1_24	5	36	21	96	56	59	34	38.54	39.29
rd32-v0_66	4	34	20	94	56	58	33	38.30	41.07
rd32-v1_68	4	36	20	96	57	60	35	37.50	38.60
4mod5-v1_23	5	69	37	187	107	117	68	37.43	36.45
4mod5-v0_20	5	20	12	49	31	31	19	36.73	38.71
one-two-three-v3_101	5	70	37	196	117	130	70	33.67	40.17
4mod5-v0_19	5	35	21	87	53	59	35	32.18	33.96
miller_11	3	50	29	106	56	72	41	32.08	26.79
alu-v0_26	5	84	46	188	104	128	71	31.91	31.73
mod5d2_64	5	53	32	142	86	97	62	31.69	27.91
one-two-three-v1_99	5	132	72	347	198	239	132	31.12	33.33
4gt13_92	5	66	36	136	72	94	49	30.88	31.94
rd32_270	5	84	46	214	126	148	88	30.84	30.16
ex-1_166	3	19	12	39	22	27	16	30.77	27.27
4gt13-v1_93	5	68	37	138	67	96	51	30.43	23.88
alu-v1_28	5	37	21	89	51	62	37	30.34	27.45
4gt11_83	5	23	16	66	38	47	29	28.79	23.68
ham3_102	3	20	13	36	20	26	15	27.78	25.00
4gt5_77	5	131	70	300	155	217	124	27.67	20.00
alu-v0_27	5	36	20	84	48	61	37	27.38	22.92
aj-e11_165	5	151	83	408	219	297	163	27.21	25.57
4gt11_82	5	27	19	81	48	59	38	27.16	20.83
alu-v4_37	5	37	20	85	50	62	35	27.06	30.00
mod5mils_65	5	35	21	80	48	59	34	26.25	29.17
decod24-v2_43	4	52	29	116	66	88	50	24.14	24.24
decod24-v0_38	4	51	29	105	55	81	45	22.86	18.18
4_49_16	5	217	121	523	290	418	249	20.08	14.14
one-two-three-v2_100	5	69	37	142	73	115	64	19.01	12.33
decod24-v3_45	5	150	79	316	162	258	149	18.35	8.02
one-two-three-v0_98	5	146	77	345	177	286	175	17.10	1.13
4gt10-v1_81	5	148	80	351	184	293	161	16.52	12.50
4mod7-v1_96	5	164	88	330	175	276	162	16.36	7.43
hwb4_49	5	233	127	569	314	476	280	16.34	10.83
4mod7-v0_94	5	162	88	344	183	288	166	16.28	9.29
alu-v2_32	5	163	87	338	167	286	157	15.38	5.99
alu-v2_31	5	451	239	976	506	828	466	15.16	7.91
one-two-three-v0_97	5	290	152	655	338	558	312	14.81	7.69
4gt5_76	5	91	52	200	106	172	99	14.00	6.60
alu-v4_36	5	115	63	247	130	213	124	13.77	4.62
decod24-v1_41	5	85	48	175	94	151	81	13.71	13.83
4gt13_90	5	107	60	233	134	205	119	12.02	11.19
4gt11_84	5	18	11	34	19	30	15	11.76	21.05
4gt5_75	5	83	44	177	86	157	89	11.30	-3.49
mini-alu_167	5	288	152	604	320	540	310	10.60	3.13
alu-v2_33	5	37	20	67	38	60	31	10.45	18.42
mod10_171	5	244	132	530	258	478	267	9.81	-3.49
mod10_176	5	178	95	371	197	350	193	5.66	2.03
3_17_13	3	36	22	60	33	58	34	3.33	-3.03
4gt13_91	5	103	57	210	119	205	113	2.38	5.04

Source: Prepared by the author.

the circuit changes the permutation, resulting in different permutations at the input and output of the circuit.

The next test was to optimize the circuits from the first test using the Qiskit and

compare the number of gates and levels of each circuit. Table 12 shows the results of this comparison. The structure of the Table is the same used in the first test with the addition of two columns. The added columns within the cell *%NoOptimization* show the results obtained in the first test. This was done to make it easier to compare the results before and after the Qiskit optimization.

Table 12 – CNOT Mappings strategy compared with Standalone with Qiskit optimization.

	Original			Standalone		OurApproach		%Improvements		%NoOptimization	
Benchmarks	**Qubits**	**Gates**	**Levels**	**Gates**	**Levels**	**Gates**	**Levels**	**Gates**	**Levels**	**Gates**	**Levels**
mod5d1_63	5	22	12	72	47	35	22	51.39	53.19	50.00	44.90
4mod5-v1_22	5	21	12	56	36	30	19	46.43	47.22	44.26	42.11
alu-v3_34	5	52	29	109	69	64	36	41.28	47.83	46.10	45.98
rd32-v0_66	4	34	20	69	42	43	26	37.68	38.10	38.30	41.07
rd32-v1_68	4	36	20	69	42	43	26	37.68	38.10	37.50	38.60
4mod5-v0_20	5	20	12	44	29	28	16	36.36	44.83	36.73	38.71
one-two-three-v3_101	5	70	37	163	102	104	55	36.20	46.08	33.67	40.17
alu-v1_28	5	37	21	75	44	49	25	34.67	43.18	30.34	27.45
alu-v0_27	5	36	20	69	40	46	26	33.33	35.00	27.38	22.92
alu-v4_37	5	37	20	70	41	47	27	32.86	34.15	27.06	30.00
4mod5-v0_18	5	69	39	146	88	101	59	30.82	32.95	38.61	37.07
4mod5-v1_23	5	69	37	137	78	96	56	29.93	28.21	37.43	36.45
4mod5-v0_19	5	35	21	69	42	49	29	28.99	30.95	32.18	33.96
mod5d2_64	5	53	32	118	71	84	53	28.81	25.35	31.69	27.91
4mod5-v1_24	5	36	21	70	44	50	29	28.57	34.09	38.54	39.29
rd32_270	5	84	46	172	105	124	73	27.91	30.48	30.84	30.16
mod5mils_65	5	35	21	63	38	47	27	25.40	28.95	26.25	29.17
alu-v0_26	5	84	46	135	71	102	56	24.44	21.13	31.91	31.73
alu-v2_33	5	37	20	62	35	47	26	24.19	25.71	10.45	18.42
decod24-v2_43	4	52	29	87	50	66	39	24.14	22.00	24.14	24.24
one-two-three-v1_99	5	132	72	257	151	198	110	22.96	27.15	31.12	33.33
aj-e11_165	5	151	83	300	174	239	131	20.33	24.71	27.21	25.57
4gt13-v1_93	5	68	37	97	49	79	42	18.56	14.29	30.43	23.88
one-two-three-v2_100	5	69	37	109	58	89	51	18.35	12.07	19.01	12.33
4gt13_92	5	66	36	95	50	78	40	17.89	20.00	30.88	31.94
4gt5_77	5	131	70	213	111	176	107	17.37	3.60	27.67	20.00
decod24-v0_38	4	51	29	71	39	62	35	12.68	10.26	22.86	18.18
4_49_16	5	217	121	392	224	345	207	11.99	7.59	20.08	14.14
4gt10-v1_81	5	148	80	252	136	225	124	10.71	8.82	16.52	12.50
hwb4_49	5	233	127	409	232	380	231	7.09	0.43	16.34	10.83
4gt13_90	5	107	60	179	103	167	97	6.70	5.83	12.02	11.19
4gt11_83	5	23	16	46	24	44	25	4.35	-4.17	28.79	23.68
one-two-three-v0_98	5	146	77	249	137	240	150	3.61	-9.49	17.10	1.13
one-two-three-v0_97	5	290	152	450	251	435	253	3.33	-0.80	14.81	7.69
4mod7-v0_94	5	162	88	239	127	232	140	2.93	-10.24	16.28	9.29
decod24-v3_45	5	150	79	217	115	212	129	2.30	-12.17	18.35	8.02
miller_11	3	50	29	57	31	56	30	1.75	3.23	32.08	26.79
4mod7-v1_96	5	164	88	231	123	228	137	1.30	-11.38	16.36	7.43
mini-alu_167	5	288	152	423	225	418	248	1.18	-10.22	10.60	3.13
4gt11_82	5	27	19	55	34	55	33	0.00	2.94	27.16	20.83
3_17_13	3	36	22	44	24	44	24	0.00	0.00	3.33	-3.03
4gt11_84	5	18	11	24	12	24	12	0.00	0.00	11.76	21.05
ex-1_166	3	19	12	21	12	21	12	0.00	0.00	30.77	27.27
alu-v2_32	5	163	87	228	118	229	125	-0.44	-5.93	15.38	5.99
mod10_171	5	244	132	363	193	365	205	-0.55	-6.22	9.81	-3.49
4gt5_75	5	83	44	120	60	121	65	-0.83	-8.33	11.30	-3.49
4gt13_91	5	103	57	158	91	161	88	-1.90	3.30	2.38	5.04
alu-v2_31	5	451	239	673	355	690	414	-2.53	-16.62	15.16	7.91
ham3_102	3	20	13	22	12	23	14	-4.55	-16.67	27.78	25.00
mod10_176	5	178	95	266	138	280	160	-5.26	-15.94	5.66	2.03
decod24-v1_41	5	85	48	119	64	126	66	-5.88	-3.13	13.71	13.83
alu-v4_36	5	115	63	163	86	174	103	-6.75	-19.77	13.77	4.62
4gt5_76	5	91	52	134	68	145	88	-8.21	-29.41	14.00	6.60

Source: Prepared by the author.

The Qiskit output a circuit designed with the U and CNOT gates. It means that single qubit adjacent gates are transformed in one of these U gates. From Table 12, in some circuits the percentage of improvement increased in number of gates and levels, e.g.,

the circuits *alu-v4_37* and *mod5d1_63*. In other cases, it found worse results. For example, the circuits *4gt5_76* and *alu-v2_31*. It may occur due the number of adjacent gates acting on a single qubit. Since the Standalone algorithm changes the permutation of the circuit, it may cause gates acting on a single qubit to became adjacent, being optimized by Qiskit algorithm. In the proposed mapping strategy, this adjacency of gates acting on a single qubit is harder to happen due the characteristic of keeping the permutation unchanged.

The next test was to compare the proposed mapping strategy with Qiskit. To do that, the optimized circuits from the proposed mapping strategy were used. Table 13 shows the results of this comparison. The structure of the Table is the same used in the other benchmarks. The columns within *Qiskit* cell show the results obtained with Qiskit algorithm.

The proposed mapping strategy found circuits with fewer gates than Qiskit for all the benchmarks evaluated. However, for the number of levels, in two circuits both approaches found the same number of levels, *decod24-v3_45* and *miller_11*. In other two circuits the Qiskit found better results regarding the number of levels, *4gt13_92* and *ham3_102*. Qiskit is guided by a heuristic to map the misplaced CNOT gates. This mapping is done by swapping the qubits as needed in order to comply with the architecture restrictions. Hence, the Qiskit algorithm needs more additional gates to map a circuit to a target architecture. Due to this, the proposed mapping strategy found circuits with fewer gates up to 73.60% and 70.65% fewer levels (*alu-v4_37*) when both approaches are compared.

The next step is to evaluate the proposed mapping strategy in the IBM QX5 with 16 qubits. To do that, a case of study will be used.

3.3.2 Case of study

In order to evaluate the proposed mappings, the circuit shown in Figure 30 will be used. The circuit has 7 gates and 5 levels. The goal is to map the circuit to IBM's QX5 architecture with the fewest number of gates.

With a trivial permutation the qubits $\{a, b, c, d, e, f\}$ are mapped to $\{Q_0, Q_1, Q_2, Q_3, Q_4, Q_5\}$. In this way, none of the CNOT gates can be directly mapped (see Figure 11(b)). In order to reduce the number of CNOT gates that must be mapped, different permutations can be used.

The Standalone algorithm initially find a permutation of the qubits in a way that satisfies the CNOT constraints of the first few levels of the circuit. Then, the algorithm

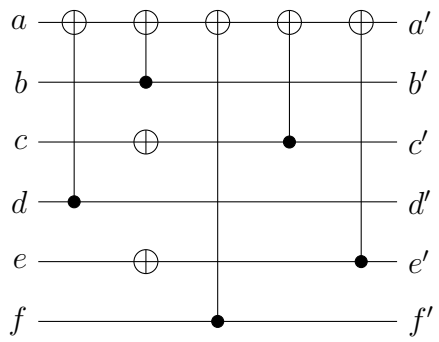
Table 13 – CNOT mappings strategy compared with Qiskit.

	Original			Qiskit		OurApproach		%Improvements	
Benchmarks	**Qubits**	**Gates**	**Levels**	**Gates**	**Levels**	**Gates**	**Levels**	**Gates**	**Levels**
alu-v4_37	5	37	20	178	92	47	27	73.60	70.65
4mod5-v1_24	5	36	21	154	76	50	29	67.53	61.84
4gt13-v1_93	5	68	37	232	108	79	42	65.95	61.11
alu-v3_34	5	52	29	184	106	64	36	65.22	66.04
decod24-v1_41	5	85	48	360	179	126	66	65.00	63.13
alu-v1_29	5	37	21	128	61	46	26	64.06	57.38
4mod5-v1_22	5	21	12	82	47	30	19	63.41	59.57
mod5d1_63	5	22	12	92	47	35	22	61.96	53.19
decod24-v2_43	4	52	29	173	93	66	39	61.85	58.06
mod5d2_64	5	53	32	210	113	84	53	60.00	53.10
rd32-v1_68	4	36	20	105	52	43	26	59.05	50.00
rd32-v0_66	4	34	20	104	53	43	26	58.65	50.94
mod5mils_65	5	35	21	112	56	47	27	58.04	51.79
alu-v3_35	5	37	20	110	53	47	27	57.27	49.06
decod24-v0_38	4	51	29	145	78	62	35	57.24	55.13
one-two-three-v2_100	5	69	37	207	108	89	51	57.00	52.78
4gt5_75	5	83	44	272	137	121	65	55.51	52.55
4mod5-v0_19	5	35	21	110	49	49	29	55.45	40.82
mini-alu_167	5	288	152	864	444	418	248	51.62	44.14
4_49_16	5	217	121	710	375	345	207	51.41	44.80
alu-v0_27	5	36	20	94	52	46	26	51.06	50.00
4mod5-v1_23	5	69	37	196	109	96	56	51.02	48.62
4gt13_91	5	103	57	326	159	161	88	50.61	44.65
4mod7-v1_96	5	164	88	443	227	228	137	48.53	39.65
aj-e11_165	5	151	83	451	223	239	131	47.01	41.26
alu-v1_28	5	37	21	92	51	49	25	46.74	50.98
one-two-three-v0_98	5	146	77	450	223	240	150	46.67	32.74
alu-v2_33	5	37	20	88	48	47	26	46.59	45.83
4gt11_82	5	27	19	102	56	55	33	46.08	41.07
alu-v2_31	5	451	239	1266	629	690	414	45.50	34.18
4gt5_77	5	131	70	313	162	176	107	43.77	33.95
4gt13_90	5	107	60	292	145	167	97	42.81	33.10
one-two-three-v1_99	5	132	72	340	188	198	110	41.76	41.49
one-two-three-v0_97	5	290	152	746	395	435	253	41.69	35.95
4mod5-v0_20	5	20	12	48	29	28	16	41.67	44.83
mod10_171	5	244	132	615	316	365	205	40.65	35.13
4mod7-v0_94	5	162	88	380	197	232	140	38.95	28.93
4gt11_83	5	23	16	72	45	44	25	38.89	44.44
rd32_270	5	84	46	199	115	124	73	37.69	36.52
one-two-three-v3_101	5	70	37	164	82	104	55	36.59	32.93
4mod5-v0_18	5	69	39	157	77	101	59	35.67	23.38
mod10_176	5	178	95	424	209	280	160	33.96	23.44
hwb4_49	5	233	127	551	302	380	231	31.03	23.51
4gt11_84	5	18	11	34	19	24	12	29.41	36.84
4gt10-v1_81	5	148	80	308	162	225	124	26.95	23.46
alu-v0_26	5	84	46	138	72	102	56	26.09	22.22
alu-v4_36	5	115	63	223	113	174	103	21.97	8.85
ex-1_166	3	19	12	26	15	21	12	19.23	20.00
decod24-v3_45	5	150	79	258	129	212	129	17.83	0.00
alu-v2_32	5	163	87	278	143	229	125	17.63	12.59
4gt5_76	5	91	52	172	90	145	88	15.70	2.22
miller_11	3	50	29	60	30	56	30	6.67	0.00
3_17_13	3	36	22	47	25	44	24	6.38	4.00
ham3_102	3	20	13	24	13	23	14	4.17	-7.69
4gt13_92	5	66	36	80	39	78	40	2.50	-2.56

Source: Prepared by the author.

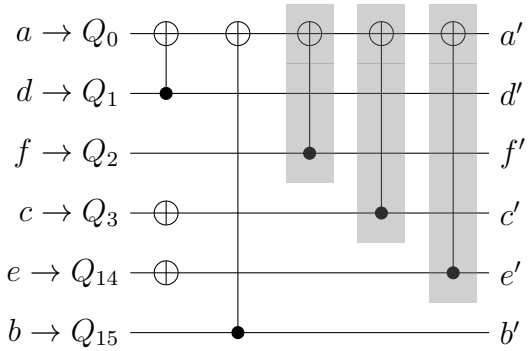
maps the CNOT gates by swapping qubits, changing the permutation of the circuit as needed. For the circuit in Figure 30 the algorithm initially finds the permutation of the qubits $\{a, b, c, d, e, f\}$ to $\{Q_0, Q_{15}, Q_3, Q_1, Q_{14}, Q_2\}$. Figure 31 shows the circuit for the given qubits permutation.

Figure 30 – Original circuit for the first evaluation of the CNOT mappings.



Source: Prepared by the author.

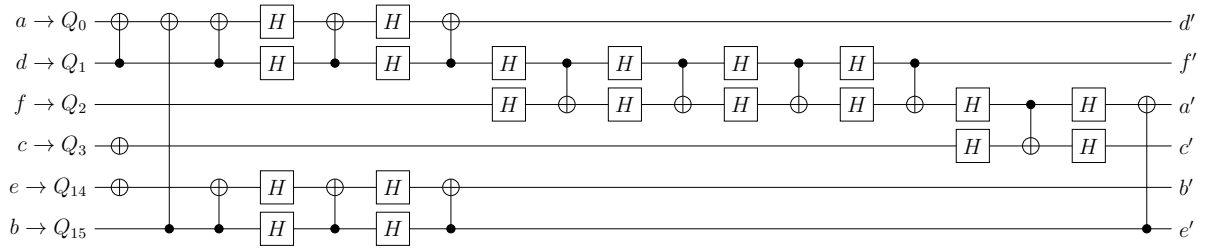
Figure 31 – Qubits permutation of Standalone algorithm for the first evaluation of the CNOT mappings.



Source: Prepared by the author.

The first two CNOT gates complies with the IBM QX5 constraints, but the last three (highlighted) must be mapped. Thus, the circuit found with the Standalone algorithm is shown in Figure 32. The order of qubits was changed for a better presentation of the circuit. The algorithm found a mapping with 36 gates and 19 levels.

Figure 32 – Standalone algorithm mapping for the first evaluation of the CNOT mappings.



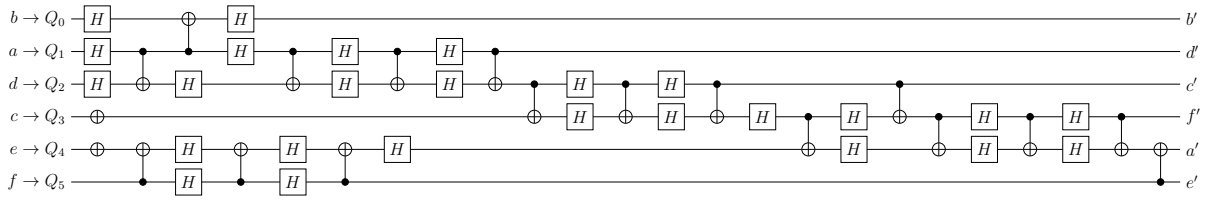
Source: Prepared by the author.

To map the three highlighted CNOT in Figure 31, the algorithm added 32 gates and 17 levels to the circuit. This was done by changing the permutation of the circuit, swapping the qubits of the CNOT that needed to be mapped. The algorithm changes the

permutation of the circuit as necessary, resulting in a different permutation at the end of the circuit. Note that at end of the circuit the logical qubit a is swapped with d, then swapped with f. Also, logical qubit e is swapped with b.

The other approach to map the circuit in Figure 30 is the Qiskit, which changes the permutation of the circuit as needed. The mappings are found with heuristics and realized using SWAP gates. Figure 33 shows the mapped circuit found with Qiskit.

Figure 33 – Qiskit mapping for the first evaluation of the CNOT mappings.



Source: Prepared by the author.

The mapping found with Qiskit has 45 gates and 24 levels. Note that the qubits used in the Qiskit mapping are different from the Standalone algorithm. This permutation resulted in a circuit with 25% more gates and 26% more levels than the Standalone algorithm. It shows that the right choice of permutation is important along with a good mapping strategy for the CNOT gates.

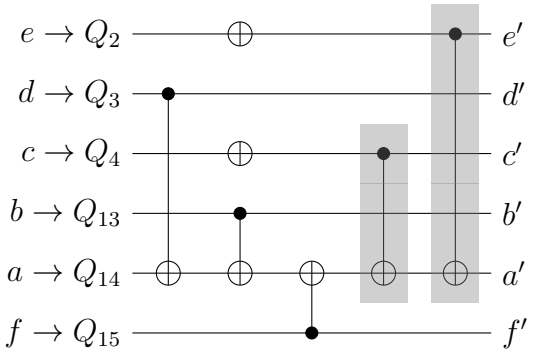
Different from both approaches, the proposed mapping strategy does not changes the permutation of the circuit, i.e., the permutation is the same at the beginning and at the end of the circuit. Therefore, the goal is to find a permutation that reduces the number of misplaced CNOT.

The circuit has only five CNOT gates with all the targets in the same qubit, thus, it is possible to find a good permutation by hand. The qubits that has the higher number of possible targets in the IBM QX5 are the qubits Q_{14}, Q_4 and Q_{10} with three possible CNOT gates. Then, the logical qubit a can be mapped to Q_{14}.

In order to map each control qubit, the adjacent qubits of Q_{14} can be used. As each gate appears only one time in the circuit, the order of the mapping will not affect the cost of the permutation. In this way, the control qubits were mapped from left to right. The first three logical qubits with controls d, b and f can be mapped to Q_3, Q_{13} and Q_{15}, respectively. The last two controls must be mapped as close as possible of the target qubit (Q_{14}). In this way, the logical qubits c and e are mapped to Q_4 and Q_2, respectively. The permutation found is shown in Figure 34.

Comparing with the permutation from Standalone algorithm which requires the

Figure 34 – Qubits permutation of proposed approach for the first evaluation of the CNOT mappings.

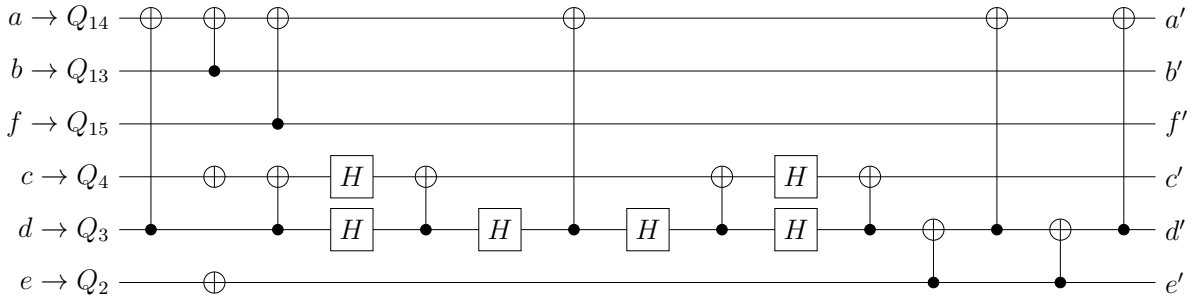


Source: Prepared by the author.

mapping of three CNOT, this permutation only needs to map two CNOT gates (highlighted) for the entire circuit to comply with the IBM QX5 constraints. Also, as those two CNOT are placed as close as possible of the target qubit, the mapping will be performed at fewer cost.

The CNOT(Q_4, Q_{14}) was replaced by the mapping CBA$(4, 3)$, CNOT$(3, 14)$, CBA$^\dagger(3, 4)$, that swaps the control in Q_4 to Q_3. Then, the CNOT(Q_2, Q_{14}) was replaced by the mapping CNOT3$(2, 3, 14)$. Figure 35 shows the mapping of the circuit. The lines were rearranged for a better presentation.

Figure 35 – Proposed approach for the first evaluation of the CNOT mappings.



Source: Prepared by the author.

Note that the permutation does not change, the qubits are the same at the beginning and at the end of the circuit. The proposed mapping strategy along with a good permutation found a circuit with 20 gates and 14 levels. That means that the proposed approach reduced the mapping by 44% in the number of gates and by 26% in the number of levels compared to the Standalone algorithm. Compared to Qiskit, the reduction was of 55% in the number of gates and 42% in the number of levels. No optimization were performed in this test.

The case of study showed that a good permutation along with the mapping strategy can find smaller circuits than other approaches. This is due the fact that a good permutation can reduce the number of CNOT gates needing mapping, so fewer gates need to be added to the circuit to comply with an IBM QX architecture.

The next chapter address the problem of finding a good permutation. It is well known that is a NP-complete problem (BOTEA; KISHIMOTO; MARINESCU, 2018; SIRAICHI *et al.*, 2018). The proposed solution for the permutation is a formulation of the problem using the concepts of Integer Linear Programming.

4 INTEGER LINEAR PROGRAMMING MODEL

This chapter presents the formulation of the Integer Linear Programming (ILP) model to solve the permutation problem. Section 4.1 describes the permutation problem. Section 4.2 shows the formulation of the permutation problem as an ILP problem followed by Section 4.3 in which an example of the Integer Linear Programming formulation is presented. Section 4.4 shows the experimental results applying the proposed formulation.

4.1 THE PERMUTATION PROBLEM

One way to reduce the number of non-available CNOT gates in an IBM architecture is to find an allocation (permutation) of the qubits. The permutation problem is defined in Definition 4.1.

Definition 4.1. *Given a circuit of n qubits and a quantum architecture QX, the problem consists in mapping the logical qubits* $\{a, b, c, ..., n\}$ *to the physical ones* $\{Q_0, Q_1, Q_2, ..., Q_n\}$ *in such way that minimizes the cost of the CNOT mappings in the circuit n in respect of the quantum architecture QX.*

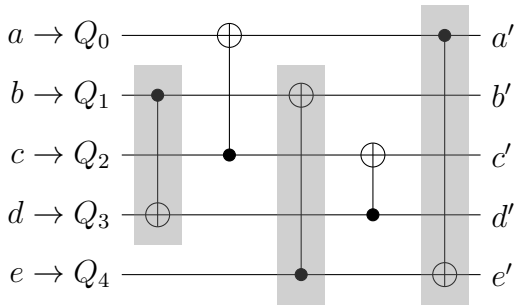
In order to show the permutation problem, an example will be given. In this example the goal is to find a permutation which will result in a circuit with fewer CNOT gates needing mapping. Thus, it will result in a smaller circuit.

Consider the circuit presented in Figure 36. The goal is to map this circuit targeting the IBM QX4. The trivial permutation is the qubits $\{a, b, c, d, e\}$ mapped to $\{Q_0, Q_1, Q_2, Q_3, Q_4\}$.

Note that only CNOT(Q_2, Q_0) and CNOT(Q_3, Q_2) can be directly used on IBM QX4 restrictions. The highlighted gates, CNOT(Q_1, Q_3), CNOT(Q_4, Q_1) and CNOT(Q_0, Q_4) must be mapped in order to comply with the architecture restrictions.

Given the CNOT costs in Figure 26, mapping all the misplaced CNOT gates to the IBM QX4 will result in a circuit with 22 gates. That means, 17 gates must be added to the original circuit. Only the number of gates will be evaluated in the permutation problem, since it is the priority in the movements defined in this work. Figure 37 shows

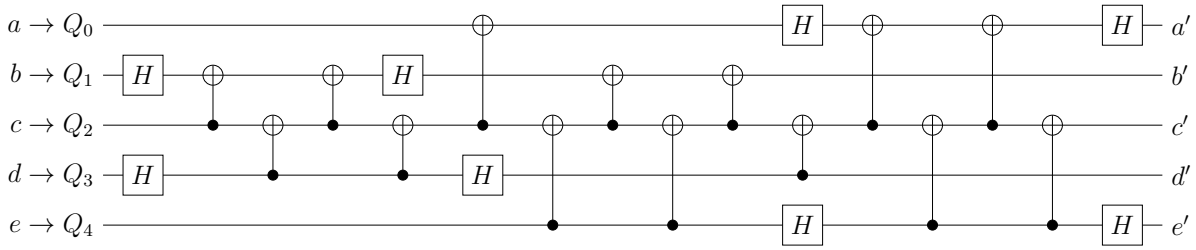
Figure 36 – Example circuit to present the permutation problem.



Source: Prepared by the author.

the mapped circuit.

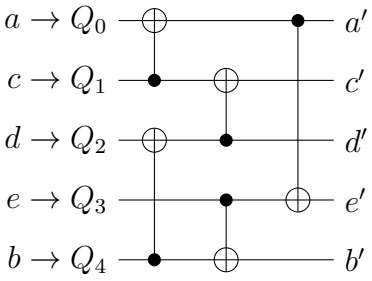
Figure 37 – Example circuit mapped to IBM QX4 with trivial permutation.



Source: Prepared by the author.

However, the circuit permutation can be changed, i.e., the logical qubits can be mapped to different physical qubits. One possible permutation for this circuit is the one shown in Figure 38. In this permutation the qubits $\{a, b, c, d, e\}$ are mapped to $\{Q_0, Q_4, Q_1, Q_2, Q_3\}$.

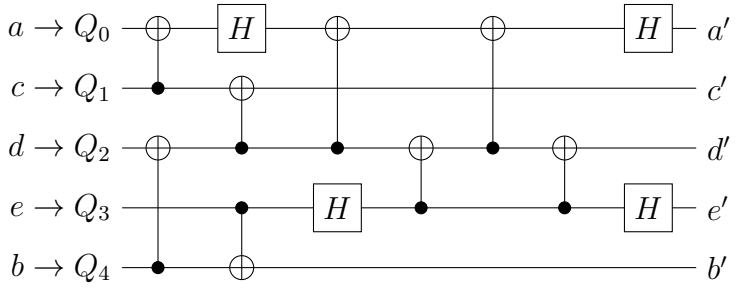
Figure 38 – Example circuit of the permutation problem permuted.



Source: Prepared by the author.

Note that with this permutation only the gate CNOT(Q_0, Q_3) do not comply with the IBM QX4 architecture. Therefore, the number of gates that must be added to the circuit is reduced. The mapped circuit for this permutation is shown in Figure 39. This circuit has 12 gates, it means that only 7 gates must be added to the circuit.

Figure 39 – Mapping of the permuted circuit targeting the IBM QX4.



Source: Prepared by the author.

Compared with the trivial permutation it was possible to reduce the circuit from 22 to 12 gates, i.e., a reduction of 45% in the circuit's size. This example shows that a good permutation can considerably reduce the number of gates. However, finding a good permutation is not a trivial task. For architectures with 5 qubits an exhaustive search can solve the problem, yet for architectures with 16 qubits this approach is infeasible due the number of possibilities (16!).

In order to address the problem of finding a good permutation, the problem was formulated as an Integer Linear Programming problem (WILLIAMS, 2013). Solving the ILP, the lowest cost permutation regarding the CNOT mappings defined in the previous chapter is guaranteed. The next section describes the formulation of the problem.

4.2 THE INTEGER LINEAR PROGRAMMING MODEL

As shown in the previous section, permuting the qubits for a given circuit can have significant cost implications. In some cases, due to the large number of possible permutations, it is not feasible to enumerate them all. A possible solution is proposed, to formulate for the permutation problem as an ILP problem. A general Integer Linear Programming is presented in Equation 7 (BAZARAA; JARVIS; SHERALI, 1990).

$$\begin{aligned} &Minimize\ cx \\ &\quad Subject\,To \\ &Ax = b \\ &x \geq 0 \\ &x\ are\ integers \end{aligned} \tag{7}$$

Where, c is $(c_1, c_2, c_3, ..., c_n)$, x, b and A are:

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \quad b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} \quad A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}$$

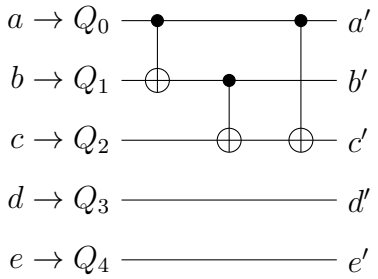
The cx is the *objective function* to be minimized, the vector c is the *cost* of each *integer variable* x. The vector b is the *right-hand-side (RHS)* vector representing the requirements to be satisfied. The matrix A represents the numbers of the constraints' problem.

The ILP is a mathematical model which the variables are all integers and has a linear objective function. It is known to be a NP-complete problem which provides a good quality solution (KUENDEE; JANJARASSUK, 2018). However, it has an exponential execution time (SIMON; TAKEFUJI, 1988).

The permutation ILP formulation consists of an objective function and three different set of constraints. The first set is to ensure that the operation in each logical qubit is not changed, *i.e.*, each "line" of the circuit must remain with the same gates after the permutation. The second set prevents two logical qubits to be mapped to the same physical qubit, meaning that two different gates from different qubits cannot be mapped to the same pair of qubits and the last set ensures that each gate of the circuit will have only one mapping.

In order to present the ILP formulation, the circuit shown in Figure 40 will be used. To generate the ILP formulation the logical qubits will be mapped with the trivial permutation.

Figure 40 – Example circuit to describe the ILP formulation.



Source: Prepared by the author.

The variables of the ILP are the CNOT gates of the given circuit and are all integers. Each CNOT in the circuit will generate a set of variables regarding all the possible mappings for the respective gate. These variables are represented by $G_{ij}C_{mn}$ and are

all integers. The G_{ij} represents the current position of the gate in the circuit, where i, j are the control and target qubits of the gate, respectively. The C_{mn} represents all the possible mappings for the gate, where m, n are the control and target qubits of the mappings, respectively.

For example, the first CNOT from the circuit in Figure 40 will generate the variables $G_{01}C_{mn}$, where $0 \leq m < 5$ and $0 \leq n < 5$ with $m \neq n$. Note that the values of m, n vary according to the number of qubits of the circuit.

4.2.1 Objective function

In an ILP model the objective function is the weighted sum of the variables. As the variables are the possible mappings for the gates in the circuit, it will be the sum of the cost of each variable (mapping) generated from the circuit. The cost of each variable will change regarding the goal of the ILP formulation. Equation 8 shows the generalization of the objective function.

$$minimize \sum_{CNOT(i,j)} \sum_{m=0}^{q} \sum_{n=0}^{q} N_{G_{ij}} * T_{ij} * G_{ij}C_{mn}, \ \forall\, m \neq n \tag{8}$$

Where,

- $CNOT(i, j)$ is each CNOT of the circuit with control qubit i and target qubit j;
- q is the number of qubits of the circuit minus 1;
- $N_{G_{ij}}$ is the number of times gate G_{ij} is used in the circuit;
- T_{ij} is the cost of each C_{mn} mapping;
- $G_{ij}C_{mn}$ is the variable representing each possible mapping.

For example, the goal regarding the permutation in IBM architectures is to find the lowest cost permutation which will result in a smaller circuit, hence, the objective function is to be minimized. The cost of each variable will be the cost of the respective gate mapping. Applying this to the circuit in Figure 40 and considering that the mapping is targeting the IBM QX4, the objective function is presented in Equation 9.

$$minimize \tag{9}$$
$$\sum_{m=0}^{4}\sum_{n=0}^{4} 1 * T_{mn}G_{01}C_{mn} + \sum_{m=0}^{4}\sum_{n=0}^{4} 1 * T_{mn}G_{12}C_{mn} + \sum_{m=0}^{4}\sum_{n=0}^{4} 1 * T_{mn}G_{02}C_{mn}, \forall m \neq n$$

Where,

- T_{mn} is the cost of the C_{mn} mapping gate (see matrix cost in Figure 26).

4.2.2 First set of constraints

The first set of constraints of the ILP model are generated by each qubit in the circuit that has at least two different gates. This is due the fact that the number of gates in a qubit cannot be changed, regardless of the permutation. These constraints represent the dependencies between the controls and/or targets of the gates that are in the same qubit. Equation 10 shows the generation of this set of constraints. This set must cover all the possible permutations, thus the constraints are generated for all possible qubits.

For each qubit l:

$$\begin{cases} For\ each\ qubit\ q: \\ (N_{G_l} - 1) * \sum_{n=0}^{q} G_{ij}C_{mn} = \sum_{CNOT(x,y)} \sum_{n=0}^{q} G_{xy}C_{rs},\ \ \forall\, m \neq n, r \neq s \end{cases} \tag{10}$$

Where,

- N_{G_l} is the number of different gates in the qubit Q_l;
- $G_{ij}C_{mn}$ is one gate present in the qubit Q_l with $m = q$ or $n = q$;
- $G_{xy}C_{rs}$ is all the other gates in the qubit Q_l with $r = q$ or $s = q$.

The idea is to keep the controls and targets in the qubits unchanged, no matter which permutation occurs. For example, in the qubit Q_1 in Figure 40, there is a target of the CNOT(Q_0, Q_1) and a control of the CNOT(Q_1, Q_2). Regardless of the permutation, these control and target will always be in the same qubit.

Another characteristic is that one of the gate must be multiplied by the number of different gates minus 1 and all the other gates must be on the other side of the equality. It will ensure that the number of gates in the qubit do not change.

This constraint needs to cover all the possible permutations for the gates in Q_1. In order to do that, this constraint must be repeated for every qubit of the circuit. As these two gates must be in the same qubit, the mappings must be for the same qubit. Therefore, the target of the mapping for CNOT(Q_0, Q_1) has to be in the same qubit of the control of the mapping for CNOT(Q_1, Q_2). Equation 11 shows the set of constraints generated for the qubit Q_1. Note that these constraints must be generated for all qubits that has at least two different gates, thus (in this case) these constraints must be generated for the qubit Q_0 and Q_2 as well.

For each qubit q:

$$(N_1 - 1) * \sum_{n=0}^{4} G_{01} C_{nq} = \sum_{n=0}^{4} G_{12} C_{qn}, \forall\, n \neq q \tag{11}$$

Where,

- N_1 is the number of different gates in the qubit Q_1.

Since CNOT(Q_0, Q_1) has the target in Q_1, the mapping target is fixed in q. Also, CNOT(Q_1, Q_2) has the control in Q_1, then the mapping control is fixed in q, making sure that the qubit Q_1 is mapped to the same qubit.

4.2.3 Second set of constraints

The next set of constraints is to prevent that two logical qubits are mapped to the same physical qubit, it means, gates from different qubits be mapped together. This is done by adding one gate of each qubit (any gate of the respective qubit) and ensuring that only one of them is present in the solution, preventing two different gates mapped to the same qubit. Equation 12 shows this constraint.

For each qubit q:

$$\sum_{CNOT(x,y)} \sum_{n=0}^{q} G_{ij} C_{mn} \;\leq\; 1, \forall\, m \neq n \tag{12}$$

Where,

- $CNOT(x, y)$ are the gates present in each qubit;
- q is the number of qubits of the circuit minus 1;
- $G_{ij}C_{mn}$ is the possible mapping for the qubit q with $m = q$ or $n = q$.

For example, in Figure 40 the control of the CNOT(Q_0, Q_1) cannot be in the same qubit that the target of the CNOT(Q_1, Q_2). Equation 13 shows an example of this constraint for the qubit Q_0. Note that for qubit Q_0 was used the control of the CNOT(Q_0, Q_1), for the qubit Q_1 was used the control of the CNOT(Q_1, Q_2) and for the qubit Q_3 was used the target of the CNOT(Q_0, Q_2).

For each qubit q:

$$\sum_{n=0}^{4} G_{01}C_{qn} + \sum_{n=0}^{4} G_{12}C_{qn} + \sum_{n=0}^{4} G_{02}C_{nq} \leq 1, \forall\, n \neq q \tag{13}$$

As all the variables are integers, the ≤ 1 ensures that all the variables in this constraint will be 0 or only one will be 1. To cover all the possible permutations, this constraint is generated for each qubit.

4.2.4 Third set of constraints

The last set of constraints is to ensure that only one of the variables representing a mapping for each gate is in the solution. For this, a constraint adding all the variables generated for a specific gate equaling 1 is used. It means that for each different gate of the circuit a respective constraint will be generated, i.e., repeated gates generate only one constraint. Equation 14 shows the equation to generate the third set of constraints.

For each CNOT(i,j):

$$\sum_{m=0}^{q} \sum_{n=0}^{q} G_{ij}C_{mn} = 1,\ \forall\, m \neq n \tag{14}$$

Where,

- $CNOT(i, j)$ is each CNOT of the circuit with control qubit i and target qubit j;
- q is the number of qubits of the circuit minus 1;
- $G_{ij}C_{mn}$ is the variable representing each possible mapping.

Equation 15 shows the equation that generates this constraint for the CNOT(Q_0, Q_1) gate of the circuit in Figure 40.

$$\sum_{m=0}^{4}\sum_{n=0}^{4} G_{01}C_{mn} = 1,\ \forall\, m \neq n \tag{15}$$

Since all the variables are integers, necessarily only one of the variables will be in the solution. It ensures that each gate has only one mapping.

The objective function along with all the constraints generates an Integer Linear Programming problem. The constraints generated in the formulation will ensure that the operation of the circuit will not be changed.

The complexity of the ILP formulation varies with the characteristics of the circuit. Each different gate in the circuit will generate a set of variables in the ILP formulation and the number of constraints varies with the number of CNOT gates in the same qubit. Let n be the number of qubits, y the number of different CNOT gates in the circuit and x be the number of qubits with at least two different acting CNOT gates. The number of variables v_c of the ILP is presented in Equation 16 and the number of constraints c_c is shown in Equation 17.

$$v_c = (n^2 - n) * y,\ \ \forall\, n > 1,\ y \geq 1 \tag{16}$$

$$\begin{cases} c_c = (n * x) + (n + y),\ \ \forall\, n > 1,\ y > 1,\ x \geq 0 \\ c_c = 1,\ \ \forall\, n > 1,\ y = 1,\ x = 0 \end{cases} \tag{17}$$

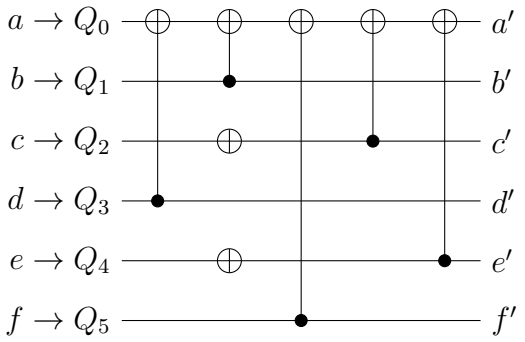
If there is only one gate in the circuit, only one constraint needs to be generated. This constraint will be the all the possible mappings for the respective gate.

The goal of the formulation can be adapted to different applications only by changing the objective function. The formulation will keep the structure of the circuit unchanged, thus, it can be applied in reversible circuits with any type of gate, no matter the number of controls or targets. In the following, an example will show an application of the proposed formulation.

4.3 EXAMPLE OF APPLICATION OF THE ILP FORMULATION

One of the applications of the proposed formulation is to find the lowest cost permutation for a Clifford+T circuit to be used in an IBM QX architecture. An example will be used to show the ILP formulation of a circuit. The example circuit is shown in Figure 41. Note that is the same circuit of Figure 30. The qubits $\{a, b, c, d, e, f\}$ are mapped to $\{Q_0, Q_1, Q_2, Q_3, Q_4, Q_5\}$.

Figure 41 – Example circuit for the application of the ILP formulation.



Source: Prepared by the author.

The goal is to find the lowest cost permutation for this circuit to be used in the IBM QX5. To do that the mapping cost of each possible CNOT needs to be calculated. Figure 42 shows the matrix with the cost of each possible CNOT in the IBM QX5 architecture. This matrix was generated with the command created on top of RevKit.

Figure 42 – IBM QX5 architecture cost matrix.

$$T = \begin{bmatrix}
- & 4 & 10 & 13 & 19 & 29 & 39 & 51 & 61 & 55 & 45 & 41 & 29 & 20 & 10 & 4 \\
0 & - & 0 & 3 & 9 & 19 & 29 & 41 & 51 & 61 & 53 & 41 & 29 & 19 & 9 & 10 \\
10 & 4 & - & 0 & 3 & 13 & 23 & 35 & 45 & 55 & 45 & 35 & 23 & 13 & 3 & 4 \\
19 & 7 & 4 & - & 0 & 10 & 19 & 31 & 41 & 45 & 35 & 31 & 19 & 10 & 0 & 7 \\
25 & 13 & 7 & 4 & - & 4 & 7 & 19 & 29 & 33 & 23 & 19 & 7 & 4 & 10 & 13 \\
35 & 25 & 19 & 10 & 0 & - & 4 & 10 & 20 & 23 & 13 & 10 & 4 & 10 & 13 & 23 \\
45 & 33 & 23 & 13 & 3 & 0 & - & 0 & 10 & 13 & 3 & 0 & 10 & 13 & 25 & 33 \\
55 & 43 & 33 & 23 & 13 & 10 & 4 & - & 4 & 7 & 0 & 10 & 19 & 23 & 35 & 43 \\
67 & 55 & 45 & 35 & 25 & 22 & 10 & 0 & - & 4 & 3 & 13 & 23 & 35 & 45 & 55 \\
57 & 65 & 55 & 45 & 35 & 25 & 13 & 3 & 0 & - & 0 & 10 & 19 & 31 & 35 & 45 \\
45 & 53 & 43 & 33 & 23 & 19 & 7 & 4 & 7 & 4 & - & 4 & 7 & 19 & 23 & 33 \\
35 & 43 & 33 & 23 & 13 & 10 & 4 & 10 & 19 & 10 & 0 & - & 4 & 10 & 13 & 23 \\
25 & 33 & 23 & 13 & 3 & 0 & 10 & 13 & 23 & 13 & 3 & 0 & - & 0 & 3 & 13 \\
22 & 25 & 19 & 10 & 0 & 10 & 19 & 23 & 33 & 23 & 13 & 10 & 4 & - & 0 & 10 \\
10 & 13 & 7 & 4 & 10 & 19 & 29 & 33 & 43 & 33 & 23 & 19 & 7 & 4 & - & 4 \\
0 & 10 & 0 & 3 & 9 & 19 & 29 & 41 & 51 & 45 & 35 & 31 & 19 & 10 & 0 & -
\end{bmatrix}$$

Source: Prepared by the author.

From the matrix cost in Figure 42 and the circuit mapped to the trivial permutation, the ILP problem can be generated. The number of variables and constraints are shown in Equation 18 and Equation 19, respectively.

$$v_c = (n^2 - n) * y = (16^2 - 16) * 5 = 1200 \tag{18}$$

$$c_c = (n * x) + (n + y) = (16 * 1) + (16 + 5) = 37 \tag{19}$$

The ILP problem is presented in Equation 20 with 1200 variables and 37 constraints.

$$\begin{aligned}
&minimize \\
&\sum_{m=0}^{15}\sum_{n=0}^{15} 1.T_{mn}G_{30}C_{mn} + \sum_{m=0}^{15}\sum_{n=0}^{15} 1.T_{mn}G_{10}C_{mn} + \sum_{m=0}^{15}\sum_{n=0}^{15} 1.T_{mn}G_{50}C_{mn} + \\
&+ \sum_{m=0}^{15}\sum_{n=0}^{15} 1.T_{mn}G_{20}C_{mn} + \sum_{m=0}^{15}\sum_{n=0}^{15} 1.T_{mn}G_{40}C_{mn}, \ \forall\, m \neq n \\
&Subject\,To \\
&\begin{cases}
For\ each\ qubit\ i: \\
4 * (\sum_{n=0}^{15} G_{30}C_{in}) = \sum_{n=0}^{15} G_{10}C_{in} + \sum_{n=0}^{15} G_{50}C_{in} + \sum_{n=0}^{15} G_{20}C_{in} + \sum_{n=0}^{15} G_{40}C_{in} \\
\quad \sum_{n=0}^{15} G_{30}C_{in} + \sum_{n=0}^{15} G_{10}C_{ni} + \sum_{n=0}^{15} G_{20}C_{ni} + \\
\qquad + \sum_{n=0}^{15} G_{30}C_{ni} + \sum_{n=0}^{15} G_{40}C_{ni} + \sum_{n=0}^{15} G_{50}C_{ni} \leq 1 \\
\quad \forall\, n \neq i
\end{cases}
\end{aligned} \tag{20}$$

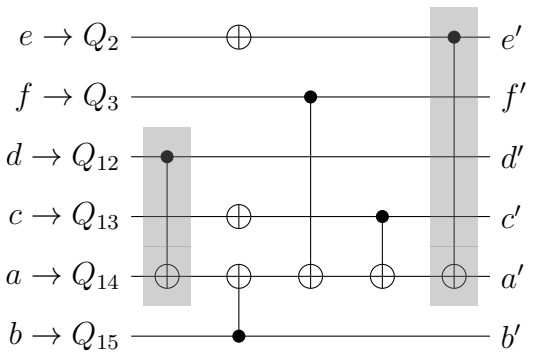
$$\sum_{m=0}^{15}\sum_{n=0}^{15} G_{30}C_{mn} = 1,\ \forall\, m \neq n$$
$$\sum_{m=0}^{15}\sum_{n=0}^{15} G_{10}C_{mn} = 1,\ \forall\, m \neq n$$
$$\sum_{m=0}^{15}\sum_{n=0}^{15} G_{50}C_{mn} = 1,\ \forall\, m \neq n$$
$$\sum_{m=0}^{15}\sum_{n=0}^{15} G_{20}C_{mn} = 1,\ \forall\, m \neq n$$
$$\sum_{m=0}^{15}\sum_{n=0}^{15} G_{40}C_{mn} = 1,\ \forall\, m \neq n$$

All variables are integers

Solving this ILP problem the lowest cost permutation for the circuit in Figure 41 is found. One solution is $G_{1_0}C_{15_14} = 1$, $G_{2_0}C_{13_14} = 1$, $G_{3_0}C_{12_14} = 1$, $G_{4_0}C_{2_14} = 1$, $G_{5_0}C_{3_14} = 1$ and all the other variables are 0. For this solution the objective function has cost 6, it means, that with this permutation 6 gates must be added to the circuit to comply with the IBM QX5 constraints.

From this solution the permutation can be found. The variable $G_{1_0}C_{15_14}$ shows that the qubit b must be mapped to Q_{15} and the qubit a to Q_{14}. Remember that $a = 0, b = 1, c = 2, d = 3, e = 4$ and $f = 5$. In the same way, the $G_{2_0}C_{13_14}$ shows that the qubit c must be mapped to Q_{13} (Note that it also shows that the qubit a must be mapped to Q_{14}). From $G_{3_0}C_{12_14}$, $G_{4_0}C_{2_14}$ and $G_{5_0}C_{3_14}$, the qubits $\{d, e, f\}$ must be mapped to $\{Q_{12}, Q_2, Q_3\}$, respectively. The mapped circuit is presented in Figure 43.

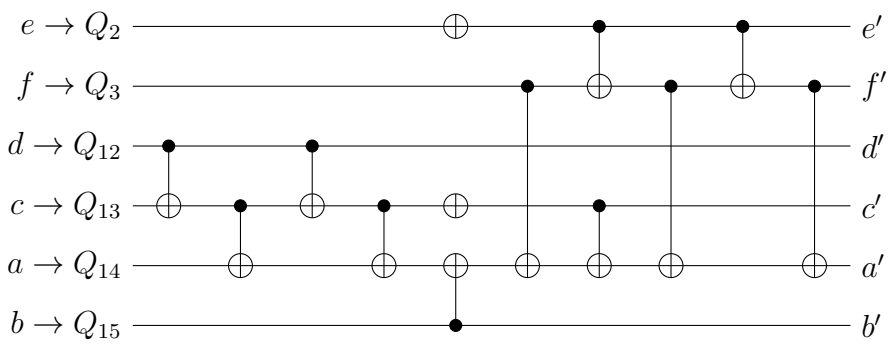
Figure 43 – Example circuit for the ILP application permuted.



Source: Prepared by the author.

The highlighted CNOT gates in the circuit are the misplaced gates for the IBM QX5. Thus, they need to be mapped. Note that from the matrix in Figure 42, the cost of each gate is 3, adding 6, the solution of the ILP problem. Figure 44 shows the mapped circuit.

Figure 44 – Example circuit for the ILP application mapped.



Source: Prepared by the author.

The mapped circuit has 13 gates and 10 levels. Compared with the circuit found in Figure 35 with 20 gates and 14 levels, the lowest cost permutation reduced the circuit in 35% and the levels in 40%. This example shows that the permutation can significantly change the size of the circuit. Therefore, the lowest cost permutation needs to be found. To solve this problem, the ILP formulation proposed can be used.

The permutation problem as an ILP problem targeting the IBM QX architectures restrictions was implemented on top of RevKit.

The program generates a file with the ILP formulation of a circuit. The ILP can be solved using different solvers. For example, the open source LP_Solve (LP_SOLVE,) or the proprietary ones, CPLEX (IBM, b) and Gurobi (GUROBI,).

In order to use the program with different quantum architectures, only the matrix with the mapping cost of each CNOT must be added to the code. The next section presents the evaluation of the proposed formulation to solve the IBM QX architectures restrictions.

4.4 EXPERIMENTAL RESULTS OF THE ILP FORMULATION

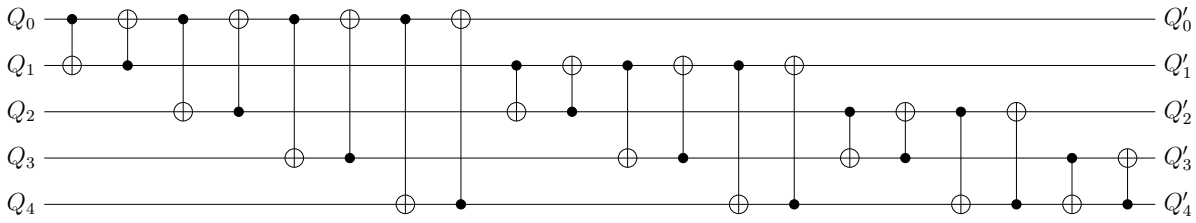
In order to evaluate the proposed formulation a series of tests were executed. The evaluations were executed on a computer using the Ubuntu operating system, Intel i5 processor, with 8 GB of ram memory. The software Gurobi (student version) with all the parameters in default mode was used to solve the ILP problems.

The first test was to compare the permutations found with the ILP formulation with the ones found with exhaustive search. To do that, the architecture IBM QX4 of 5 qubits was used. This is due the fact that is possible to find the lowest cost

permutation by exhaustive search for this architecture (5! = 120 permutations). To find all possible permutations, a program has been developed on top of RevKit that realizes all the permutations of a given circuit using the CNOT mappings defined in the previous chapter. The tests were performed with more than 20,000 circuits from 2 to 1,000,000 gates targeting the IBM QX4 architecture. For all the tests circuits, the ILP solution found the lowest cost permutation, the same as the exhaustive search.

The second test was to evaluate the impact of the number of different CNOT gates in the ILP formulation. IBM QX4 of 5 qubits was used in this evaluation, which consists in adding a CNOT one by one to the circuit until it reaches 100 gates. The first 20 CNOT gates are presented in the circuit shown in Figure 45.

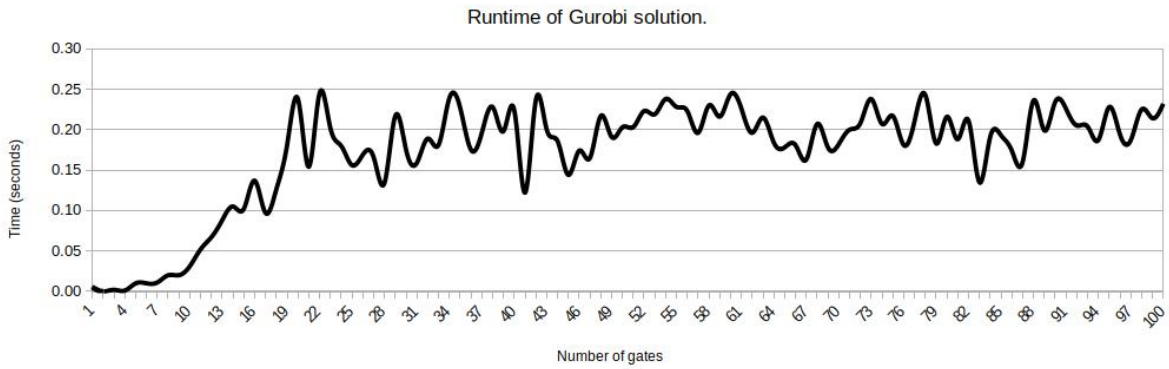
Figure 45 – Circuit to test the impact of the CNOT gates in the ILP solving time.



Source: Prepared by the author.

For each CNOT added an ILP problem was generated and solved. Since an architecture with 5 qubits has 20 different possible CNOT gates, is expected that the highest complexity of this problem will occur when all the 20 CNOT gates were used in the circuit. This same structure is repeated for 5 times, adding the CNOT one by one and generating the ILP formulation to be solved. Every ILP formulation was solved 10 times and the mean was calculated. The mean solution time found is shown in Figure 46.

Figure 46 – Solution time for circuits with 5 qubits in IBM QX4 architecture.



Source: Prepared by the author.

Figure 46 showed the impact of the solution time while the number of variables in the

ILP problem increase. After the first 20 gates, the ILP formulation will only change in the objective function due the maximum number of variables and constraints were reached. Therefore, the solution time will "stabilize", as can be seen in the graph from Figure 46. Note that the difference between the solution times is around 0.1 seconds. Also, the solution time can change if other software (or computer) is used to solve the problem.

The third test was to evaluate the CNOT mappings with Qiskit (ALEKSANDROWICZ; ALEXANDER; BARKOUTSOS,) and Standalone algorithm (ZULEHNER; PALER; WILLE, 2019). For this, some benchmarks from Zulehner, Paler and Wille (2019) were used. The goal is to map the circuits into IBM QX5 architecture of 16 qubits. To do this, the lowest cost permutation will be found with the ILP formulation and the CNOT mapping will be applied. The ILP was generated with RevKit and the software Gurobi was used to solve it. The Gurobi runtime was set to an hour (3,600 seconds). If the solution was not found after this time, the incumbent solution were used to find the permutation.

Table 14 shows the benchmarks results. The first column, **Circuits**, shows the name of the circuits. The columns below **Original** cell are the characteristics of the original circuit. The column **Qubits** presents the number of qubits of each circuit. The columns **Gates** and **Levels** show the number of gates and levels of the circuits, respectively. The column **T(sec)** is the time, in seconds, to found the mapping. For the CNOT mapping is the time to solve the ILP problem. The columns below **Qiskit** and **CNOT Mappings** cells are the characteristics of the circuits found with Qiskit and CNOT mappings, respectively. The column, **%Improvements**, is the percentage of reduction when both approaches are compared. The gap of the circuits with the solution time exceeded are shown in Appendix C.

Qiskit outputs a circuit realized with CNOT and the general unitary U gates (ALEKSANDROWICZ; ALEXANDER; BARKOUTSOS,). In order to realize the CNOT mappings circuits with the same gates, the circuits were transformed using Qiskit.

Comparing the circuits found with both algorithms, it can be seen that the CNOT mapping along with the lower cost permutation found circuits with fewer number of gates for the most of the circuits. For example, in the circuit *0410184_169* the percentage of improvement was 57% in the number of gates and 52% in the number of levels. However, in some cases Qiskit found circuits with fewer levels.

In the circuit *z4_268*, the CNOT mapping found a realization with 15% fewer gates, yet Qiskit circuit has 23% fewer levels. This is an example of the priority regarding the

Table 14 – Benchmarks to compare CNOT Mappings with Qiskit in IBM QX5.

	Original			Qiskit			CNOT Mappings			%Improvements	
Circuits	**Qubits**	**Gates**	**Levels**	**Gates**	**Levels**	**T(sec)**	**Gates**	**Levels**	**T(sec)**	**Gates**	**Levels**
4mod5-bdd_287	7	70	41	326	99	4	114	65	3	65.03	34.34
0410184_169	14	211	97	848	187	10	364	88	172	57.08	52.94
cnt3-5_179	16	175	28	739	119	8	340	79	1,235	53.99	33.61
rd53_138	8	132	56	580	228	7	270	129	114	53.45	43.42
alu-bdd_288	7	84	44	338	136	4	158	88	4	53.25	35.29
rd73_140	10	230	92	1128	283	11	539	252	407	52.22	10.95
mini_alu_305	10	173	64	849	245	8	406	124	568	52.18	49.39
rd53_133	7	580	309	2,187	822	25	1,138	560	105	47.97	31.87
cm82a_208	8	650	323	2,564	891	29	1,336	567	395	47.89	36.36
ham7_104	7	320	172	1,379	481	16	727	374	110	47.28	22.25
ex2_227	7	631	336	2,399	984	27	1,280	642	361	46.64	34.76
sys6-v0_111	10	215	74	918	257	8	494	239	455	46.19	7.00
majority_239	7	612	325	2,268	841	26	1,222	607	79	46.12	27.82
cm152a_212	12	1,221	645	4,697	1,670	52	2,549	1,183	1,017	45.73	29.16
f2_232	8	1,206	630	4,518	1,754	51	2,491	1,149	150	44.86	34.49
square_root_7	15	7,630	3,132	30,363	7,948	307	16,883	4,673	3,600	44.40	41.21
rd53_251	8	1,291	674	5,194	2,044	58	2,891	1,584	418	44.34	22.50
decod24-bdd_294	6	73	40	195	117	4	110	73	1	43.59	37.61
sym6_316	14	270	129	1,263	382	14	716	189	3,600	43.31	50.52
rd53_135	7	296	149	1,120	388	14	648	305	32	42.14	21.39
C17_204	7	467	236	1,848	663	21	1,070	505	327	42.10	23.83
rd53_131	7	469	248	1,679	627	19	1,032	540	58	38.53	13.88
dc1_220	11	1,914	984	7,317	2,369	78	4,620	2,138	3,600	36.86	9.75
rd53_311	13	275	117	1,180	383	12	753	259	2,768	36.19	32.38
sym9_146	12	328	127	1,427	369	15	915	360	3,600	35.88	2.44
wim_266	11	986	480	3,767	1,041	40	2,424	1,054	3,460	35.65	-1.25
pm1_249	14	1,776	888	6,915	1,931	73	4,604	2,034	3,600	33.42	-5.33
max46_240	10	27,126	13,415	107,933	31,159	1124	74,450	24,862	3,600	31.02	20.21
sym6_145	7	3,888	2,066	14,323	5,786	161	10,760	5,519	584	24.88	4.61
sqn_258	10	10,223	5,159	39,983	13,819	425	31,430	13,100	3,600	21.39	5.20
adr4_197	13	3,439	1,728	14,055	3,304	143	11,109	4,857	3,600	20.96	-47.00
co14_215	15	17,936	7,853	84,792	19,114	764	68,239	19,097	3,600	19.52	0.09
squar5_261	13	1,993	986	8,289	1,953	86	6,711	2,884	3,600	19.04	-47.67
z4_268	11	3,073	1,542	11,881	3,962	127	10,028	4,894	3,600	15.60	-23.52
sym9_148	10	21,504	11,415	81,574	28,384	889	69,730	27,395	3,600	14.52	3.48
misex1_241	15	4,813	2,528	19,301	5,442	208	16,782	5,902	3,600	13.05	-8.45
life_238	11	22,445	11,810	92,949	28,614	953	83,264	31,971	3,600	10.42	-11.73
radd_250	13	3,213	1,678	13,097	3,805	135	11,970	4,142	3,600	8.61	-8.86
sqrt8_260	12	3,009	1,567	12,841	3,793	132	11,756	3,832	3,600	8.45	-1.03
rd84_253	12	13,658	6,841	56,694	12,159	570	51,922	14,237	3,600	8.42	-17.09
cycle10_2_110	12	6,050	3,192	25,964	7,826	266	24,268	11,243	3,600	6.53	-43.66
root_255	13	17,159	8,318	72,496	21,074	727	68,368	20,942	3,600	5.69	0.63
inc_237	16	10,619	5,510	43,845	10,490	458	45,701	15,405	3,600	-4.23	-46.85
ham15_107	15	8,763	4,541	34,959	10,704	363	36,450	9,030	3,600	-4.26	15.64

Source: Prepared by the author.

number of gates of the CNOT mappings. Another possibility of this difference in the number of levels is that Qiskit algorithm swaps qubits trough the circuit changing its permutation while the CNOT mappings swaps the qubits back to its original position to keep the permutation unchanged.

Note that in some cases the lowest cost permutation is not guaranteed, due the solver was not able to find the exact solution in the runtime, hence, the incumbent solution was used. In some of these cases, a circuit could be realized with fewer gates.

The runtime varies with the complexity of the circuit, i.e., number of gates and the dependency between them. For example, the circuit *0410184_169* has 14 qubits and 211 gates was solved in 172 seconds. In other hand, the circuit *sym6_316* with the same

number of qubits and 270 gates was not able to be solved in the runtime.

For some circuits with more than 10 qubits, the Gurobi was not able to find the optimal solution after one hour of runtime. However, for some cases it was enough to find a permutation resulting in a circuit with fewer gates than Qiskit, such as *square_root_7* with 44% fewer gates and 41% fewer levels. In two circuits Qiskit found a smaller circuit, *inc_237* and *ham15_107*, with 4% fewer gates in both cases (in *ham15_107* the CNOT mappings a circuit with fewer levels). The CNOT mapping may find a circuit with fewer gates if the optimal solution of the problem was found.

The next evaluation was to compare the CNOT mapping with the Standalone before the Qiskit optimizations. Table 15 shows the results found with this evaluation. The structure of the Table is the same of the previous one. The columns below **Standalone** show the characteristics of the circuits found with Standalone algorithm, in which column **T(sec)** shows the time for the Standalone algorithm to find a circuit mapping. The gap of the circuits with the solution time exceeded are shown in Appendix C.

Results have shown that CNOT mapping proposed in this work found circuits with fewer gates that Standalone algorithm without the Qiskit optimizations. For example, the circuit *0410184_169* has 14 qubits, which CNOT mapping found a circuit with 41% fewer gates and 33% fewer gates. Another example is the circuit *cnt3-5_179* with 16 qubits that had an improvement of 40% and 41% in the number of gates and levels, respectively.

Due the heuristic algorithm the time to find a mapping is the main difference compared with the proposed approach of this work. For almost all the circuits the Standalone algorithm took less than a second to map the circuit. One circuit that took longer than that was the *square_root_7*, which needed 30 seconds to finish. The ILP formulation did not found the exact permutation for this circuit, yet found an improvement of 28% regarding the gates and 18% the levels.

The Standalone algorithm has the same characteristic of Qiskit regarding the change in the permutation trough the circuit. Partitioning of the circuit in *layers* can result in a circuit optimized regarding the number of levels. This can be seen in the circuit *sym9_146* and *z4_268* with 14% and 22% fewer levels.

All the circuits with worse results compared with Standalone have timeout. Since the solver could not found the optimal solution, the incumbent solution was used, hence, may be the cause of the circuit with higher size.

The scalability of an exact approach is limited, however, the formulation has a variable

Table 15 – Benchmarks to compare CNOT Mappings with Standalone without optimization in IBM QX5.

	Original			Standalone			CNOT Mappings			%Improvements	
Circuits	**Qubits**	**Gates**	**Levels**	**Gates**	**Levels**	**T(sec)**	**Gates**	**Levels**	**T(sec)**	**Gates**	**Levels**
4mod5-bdd_287	7	70	41	240	145	1	134	79	3	44.17	45.52
0410184_169	14	211	97	764	292	1	449	194	172	41.23	33.56
cnt3-5_179	16	175	28	669	130	1	401	76	1,235	40.06	41.54
alu-bdd_288	7	84	44	290	126	1	179	116	4	38.28	7.94
cm152a_212	12	1,221	645	4,255	1,989	1	2,893	1,516	1,017	32.01	23.78
rd53_133	7	580	309	1,985	950	1	1,366	768	105	31.18	19.16
f2_232	8	1,206	630	4,070	1,977	1	2,883	1,501	150	29.16	24.08
square_root_7	15	7,630	3,132	28,122	11,988	30	20,031	9,730	3,600	28.77	18.84
ex2_227	7	631	336	2,099	955	1	1,518	851	361	27.68	10.89
majority_239	7	612	325	1,961	967	1	1,441	811	80	26.52	16.13
cm82a_208	8	650	323	2,120	957	1	1,560	732	395	26.42	23.51
rd53_131	7	469	248	1,599	789	1	1,199	712	58	25.02	9.76
rd53_135	7	296	149	980	451	1	738	396	32	24.69	12.20
ham7_104	7	320	172	1,071	505	1	807	445	110	24.65	11.88
rd53_138	8	132	56	415	149	1	314	157	114	24.34	-5.37
rd53_251	8	1,291	674	4,238	2,019	1	3,261	1,842	418	23.05	8.77
C17_204	7	467	236	1,561	770	1	1,224	604	327	21.59	21.56
rd53_311	13	275	117	1,044	480	1	842	420	2,768	19.35	12.50
dc1_220	11	1,914	984	6,650	3,116	1	5,367	2,814	3,600	19.29	9.69
sys6-v0_111	10	215	74	694	235	1	567	276	455	18.30	-17.45
pm1_249	14	1,776	888	6,360	2,887	1	5,316	2,660	3,600	16.42	7.86
rd73_140	10	230	92	737	300	1	620	316	407	15.88	-5.33
sym6_316	14	270	129	954	421	1	804	411	3,600	15.72	2.38
wim_266	11	986	480	3,310	1,517	1	2,800	1,397	3,460	15.41	7.91
mini_alu_305	10	173	64	540	165	1	461	176	568	14.63	-6.67
max46_240	10	27,126	13,415	96,592	42,243	1	84,118	39,660	3,600	12.91	6.11
sym6_145	7	3,888	2,066	13,030	6,449	1	12,031	6,738	584	7.67	-4.48
sqn_258	10	10,223	5,159	36,234	16,846	1	35,220	16,996	3,600	2.80	-0.89
adr4_197	13	3,439	1,728	12,661	5,761	1	12,345	6,175	3,600	2.50	-7.19
sym9_146	12	328	127	1,051	397	1	1,036	454	3,600	1.43	-14.36
sym9_148	10	21,504	11,415	75,454	36,481	1	77,336	38,738	3,600	-2.49	-6.19
z4_268	11	3,073	1,542	10,768	4,800	1	11,185	5,866	3,600	-3.87	-22.21
squar5_261	13	1,993	986	7,222	3,274	1	7,545	3,984	3,600	-4.47	-21.69
misex1_241	15	4,813	2,528	17,554	8,362	1	18,465	8,453	3,600	-5.19	-1.09
co14_215	15	17,936	7,853	70,567	26,809	5	75,142	31,834	3,600	-6.48	-18.74
life_238	11	22,445	11,810	82,393	38,314	6	91,615	48,521	3,600	-11.19	-26.64
rd84_253	12	13,658	6,841	50,352	22,007	6	56,745	28,023	3,600	-12.70	-27.34
radd_250	13	3,213	1,678	11,563	5,150	1	13,154	6,494	3,600	-13.76	-26.10
root_255	13	17,159	8,318	64,030	26,726	3	74,627	31,915	3,600	-16.55	-19.42
sqrt8_260	12	3,009	1,567	11,020	5,217	1	12,874	6,712	3,600	-16.82	-28.66
cycle10_2_110	12	6,050	3,192	21,916	10,081	1	26,604	14,634	3,600	-21.39	-45.16
ham15_107	15	8,763	4,541	31,378	14,635	1	39,718	18,235	3,600	-26.58	-24.60
inc_237	16	10,619	5,510	38,337	17,413	1	49,242	22,606	3,600	-28.45	-29.82

Source: Prepared by the author.

number of constraints. Therefore, it would be interesting to run the tests on a server with more computer power to check how scalable the ILP formulation is and how the runtime is impacted. It can be seen that for a 16 qubits circuits, a computer with i5 processor (quad core) and 8 GB of ram memory found circuits with good results.

The next evaluation is to map all circuits with Qiskit and compare them again. The goal is to reduce the number of gates and levels by optimizing the sequence of gates acting on a single qubit. Table 16 shows the circuits characteristics after the optimizations. The Table has the same structure of the previous one.

The optimizations of the circuits had the most impact in the number of levels of the circuits. The circuit *square_root_7* before the optimization had 11,988 and 9,730 levels in

Table 16 – Benchmarks to compare CNOT Mappings with Standalone optimized in IBM QX5.

	Original			Standard			CNOT Mappings			%Improvements	
Circuits	**Qubits**	**Gates**	**Levels**	**Gates**	**Levels**	**T(sec)**	**Gates**	**Levels**	**T(sec)**	**Gates**	**Levels**
0410184_169	14	211	97	848	187	10	364	88	172	57.08	52.94
rd73_140	10	230	92	1128	283	11	539	252	407	52.22	10.95
mini_alu_305	10	173	64	849	245	8	406	124	568	52.18	49.39
sys6-v0_111	10	215	74	918	257	8	494	239	455	46.19	7.00
cnt3-5_179	16	175	28	607	87	4	340	79	1235	43.99	9.20
4mod5-bdd_287	7	70	41	190	118	2	114	65	4	40.00	44.92
alu-bdd_288	7	84	44	258	81	3	158	88	1	38.76	-8.64
rd53_133	7	580	309	1,758	545	11	1,138	560	105	35.27	-2.75
square_root_7	15	7,630	3,132	25,296	6,866	156	16,883	4,673	3,600	33.26	31.94
ex2_227	7	631	336	1,887	686	11	1,280	642	15	32.17	6.41
f2_232	8	1,206	630	3,646	1,375	20	2,491	1,149	150	31.68	16.44
cm152a_212	12	1,221	645	3,707	1,367	21	2,549	1,183	1017	31.24	13.46
majority_239	7	612	325	1,735	593	10	1,222	607	395	29.57	-2.36
cm82a_208	8	650	323	1,880	660	11	1,336	567	58	28.94	14.09
rd53_138	8	132	56	373	115	3	270	129	20	27.61	-12.17
rd53_131	7	469	248	1,388	525	9	1,032	540	22	25.65	-2.86
rd53_135	7	296	149	867	277	6	648	305	19	25.26	-10.11
ham7_104	7	320	172	963	378	7	727	374	23	24.51	1.06
rd53_251	8	1,291	674	3,733	1,323	21	2,891	1,584	327	22.56	-19.73
C17_204	7	467	236	1,371	508	8	1,070	505	22	21.95	0.59
dc1_220	11	1,914	984	5,891	2,152	32	4,620	2,138	3,600	21.58	0.65
rd53_311	13	275	117	942	361	6	753	259	2768	20.06	28.25
pm1_249	14	1,776	888	5,636	2,079	30	4,604	2,034	3,600	18.31	2.16
sym6_316	14	270	129	852	301	5	716	189	3,600	15.96	37.21
wim_266	11	986	480	2,880	1,052	17	2,424	1,054	3460	15.83	-0.19
max46_240	10	27,126	13,415	86,197	28,996	439	74,450	24,862	3,600	13.63	14.26
sym6_145	7	3,888	2,066	11,510	3,962	62	10,760	5,519	584	6.52	-39.30
sqn_258	10	10,223	5,159	32,226	11,212	164	31,430	13,100	3,600	2.47	-16.84
sym9_146	12	328	127	927	315	5	915	360	3,600	1.29	-14.29
adr4_197	13	3,439	1,728	11,201	3,831	58	11,109	4,857	3,600	0.82	-26.78
z4_268	11	3,073	1,542	9,637	3,278	49	10,028	4,894	3,600	-4.06	-49.30
sym9_148	10	21,504	11,415	66,821	25,600	353	69,730	27,395	3,600	-4.35	-7.01
squar5_261	13	1,993	986	6,385	2,124	34	6,711	2,884	3,600	-5.11	-35.78
co14_215	15	17,936	7,853	63,572	14,917	307	68,239	19,097	3,600	-7.34	-28.02
misex1_241	15	4,813	2,528	15,599	5,244	83	16,782	5,902	3,600	-7.58	-12.55
life_238	11	22,445	11,810	73,578	22,955	385	83,264	31,971	3,600	-13.16	-39.28
rd84_253	12	13,658	6,841	45,014	14,431	232	51,922	14,237	3,600	-15.35	1.34
radd_250	13	3,213	1,678	10,333	3,648	53	11,970	4,142	3,600	-15.84	-13.54
root_255	13	17,159	8,318	57,261	17,593	287	68,368	20,942	3,600	-19.40	-19.04
sqrt8_260	12	3,009	1,567	9,843	3,371	51	11,756	3,832	3,600	-19.44	-13.68
cycle10_2_110	12	6,050	3,192	19,472	6,541	103	24,268	11,243	3,600	-24.63	-71.89
ham15_107	15	8,763	4,541	27,900	9,477	145	36,450	9,030	3,600	-30.65	4.72
inc_237	16	10,619	5,510	33,967	9,636	177	45,701	15,405	3,600	-34.55	-59.87

Source: Prepared by the author.

Standalone and CNOT Mappings, respectively. However, after the optimizations it has 6,866 and 4,673, respectively. The same happened with the circuit *sqn_258*, a reduction of around 16,000 gates to around 11,000 and 13,000 levels for Standalone and CNOT Mappings, respectively.

This improvement in the number of levels is due the fact that Qiskit transform a sequence of adjacent gates acting on a single qubit in one gate, having an impact in the number of levels.

Since the CNOT mappings have priority in the number of gates, worse results regarding the number of levels were expected. A possible approach to find better circuits regarding the number of levels is to find CNOT mappings taking into account the number

of gates and the number of levels with the same weight. It would be interesting to change the focus of the CNOT mappings and defined the mappings having the levels as priority, then check the tests to see what will happen with the number of levels in the comparisons.

The improvements regarding the number of gates kept at the same percentage, changing in some circuits. For example, the *0410184_169* increased from 44% to 57% regarding the number of gates. On the other hand, some circuits the improvement decreased, as in the circuit *4mod5-bdd_287*, a reduction of 4%, from 44% to 40%.

Again, all the circuits with worse results compared with the Standalone had the solver timeout. It would be interesting to see if it would change with the optimal solution along with the optimizations.

Another interesting change would be the change in the costs of the circuits, i.e., taking the probability of errors, the time to run the circuit (TANNU; QURESHI, 2019). Due the characteristic of the proposed methodologies all these changes are easily adaptable. After defining each mapping for a respective CNOT, the ILP formulation can be used in the same way to find a lower cost implementation to apply the mappings. The characteristic of the ILP model is that the variables are the gates used in the circuit. In this way, it can be adaptable to any gate and any reversible circuit.

Since the ILP formulation has a variable number of constraints, it is possible to scale it for architectures that will implement circuits with reduced number of qubits (the circuit has fewer qubits than available in the architecture). The problem with the ILP formulation is the complexity of the problem is exponential, making it harder to be used as the number of qubits and the size of circuit increases. A possible solution is to use the incumbent solution. It would be interesting to check an incumbent solution and a metaheuristic to find a good permutation for the CNOT movements. How long will it take for the incumbent solution to reach the metaheuristic solution? Another possibility is to reduce the search area of the ILP by reducing the number of variables in the formulation, i.e., mapping the most used CNOT in the circuit to a low cost position.

Overall, the results of the evaluations have shown that the CNOT mappings proposed in this work can be used to find circuit with fewer number of gates and (most of the time) fewer number of levels. The main difference between the circuits found with the CNOT mapping and the other approaches used as comparison is the final permutation of the circuit. It means, the proposed CNOT mapping keeps the permutation unchanged at the beginning and at the end of mapped circuit. This is not guaranteed by the other algorithms, since the qubits are swapped trough the circuit, changing its permutation.

The methodology to map the CNOT gates to the IBM quantum architectures can be easily adaptable to any quantum architecture regarding the CNOT constraints. Furthermore, it can be adaptable to solve any constraints. The idea is to define a set of operations that solve the quantum architecture constrains and apply the formulation to find a good permutation to realize it.

The application of the CNOT mappings can be improved by applying sub-circuits, i.e., develop an algorithm to find a good way to slice the circuit in sub-circuits and apply the ILP formulation along with the CNOT mappings, keeping the permutation changed until it reaches the end of the circuit. This may reduce the number of gates, since some CNOT qubits are distant of each other, adding to many gates to the circuit – needing to be moved back to the original positions to keep the permutation unchanged.

The next chapter will present a methodology to map the NCV and Toffoli circuits to IBM QX architectures. This methodology shows an adaptation of the ILP formulation and an application of the defined CNOT mappings to direct map non-implementable gates to IBM quantum processors.

5 MAPPING OF NCV AND TOFFOLI CIRCUITS TO IBM QX

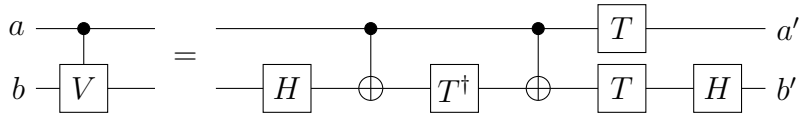
This chapter presents an approach for mapping NCV and reversible circuits to the IBM QX architectures. Both approaches will take advantage of the CNOT mappings defined in this work and of the Integer Linear Programming presented in the previous chapter. Section 5.1 shows the mapping of NCV circuits to Clifford+T targeting an IBM quantum architecture. Similarly, Section 5.2 shows the mapping of Toffoli circuits to Clifford+T complying with the IBM architecture constraints. Results of the developed approaches are presented in Section 5.3.

5.1 NCV TO CLIFFORD+T CIRCUITS

Since the Clifford+T circuits can be used in IBM QX architectures, it is necessary to map NCV into Clifford+T. The only restriction in this mapping is to ensure that all the CNOT gates used in the mapped circuit complies with the target IBM QX architecture.

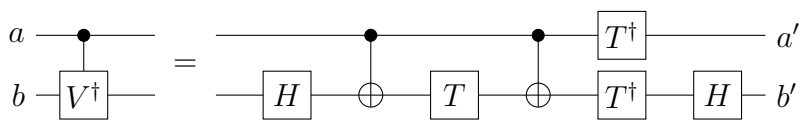
The NCV library consists in the gates NOT, CNOT, V and $V^\dagger$ gates. The NOT and CNOT gates already are in the Clifford+T library, hence only the V and $V^\dagger$ need to be mapped. The mapping of the V and $V^\dagger$ gates are shown in Figure 47 and in Figure 48, respectively.

Figure 47 – V gate realized with Clifford+T library.



Source: Adapted from Soeken, Miller and Drechsler (2013).

Figure 48 – $V^\dagger$ gate realized with Clifford+T library.



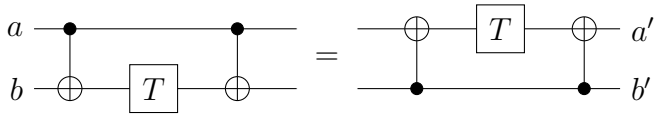
Source: Adapted from Soeken, Miller and Drechsler (2013).

Both mappings are realized with 7 gates and 6 levels, with the only difference between the two are the T and $T^\dagger$ gates. As the only restriction in the IBM QX architectures are

the CNOT gates, it can be noted that the CNOT gates after the mapping are in the same qubits that the V and $V^\dagger$ gates are.

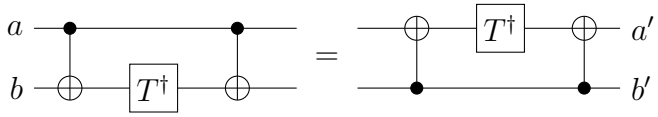
In order to use the mapping strategy presented in this work, the cost of the V and $V^\dagger$ must be calculated using the CNOT mappings described in Chapter 3. The V gate has two CNOT gates in the mapped Clifford+T circuit, thus the cost of the V gate will be the cost of these two CNOT. However, Miller, Soeken and Drechsler (2014) presented a property that can be used in order to reduce this cost. This property is shown in Figure 49 and in Figure 50, respectively.

Figure 49 – Property of CNOT gates with T gate.



Source: Prepared by the author.

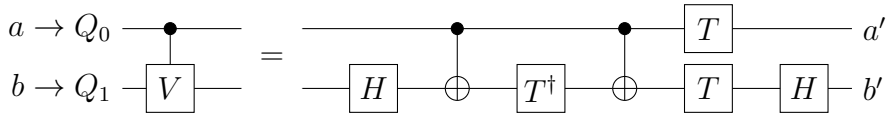
Figure 50 – Property of CNOT gates with $T^\dagger$ gate.



Source: Prepared by the author.

Using this property the V and $V^\dagger$ gates can be mapped to Clifford+T with the CNOT in both directions without an additional cost. It means that the mapping can be done using the CNOT with lower cost. For example, consider one V gate in a circuit targeting the IBM QX4 with the control in qubit Q_0 and target in qubit Q_1. Figure 51 shows this circuit.

Figure 51 – Example of the V gate targeting the IBM QX4.

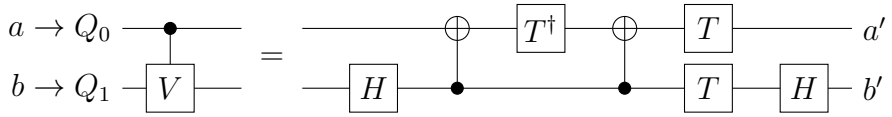


Source: Prepared by the author.

Note that in this circuit the CNOT gates are misplaced regarding the IBM QX4 and must be mapped. This mapping will add 8 gates to this circuit (see Figure 26). However, if the property is used, no additional gates are required. Figure 52 shows this circuit.

Note that this strategy can be used for the $V^\dagger$ as well. Using this strategy, a matrix with the cost of each possible V and $V^\dagger$ can be generated. Only one matrix needs to be

Figure 52 – Example of the V gate targeting the IBM QX4 using the property.



Source: Prepared by the author.

generated, since the CNOT in both gates are in the same qubits. The matrix cost for NCV circuits can be generated from the matrix cost for Clifford+T circuits. For example, for a V gate with control in Q_0 and target in Q_1, the CNOT(Q_0, Q_1) or the CNOT(Q_1, Q_0) can be used in the mapping. Therefore, the lower cost CNOT is used. In this way, the matrix cost for NCV circuits can be generated. Figure 53 shows the Clifford+T and the NCV cost matrices regarding the IBM QX4 architecture. Again, the cost of the matrices are the number of additional gates required to map a respective gate.

Figure 53 – Clifford+T mapping and NCV mapping cost matrices.

(a) Clifford+T.

$$T = \begin{bmatrix} - & 4 & 4 & 7 & 7 \\ 0 & - & 4 & 7 & 7 \\ 0 & 0 & - & 4 & 4 \\ 3 & 3 & 0 & - & 0 \\ 3 & 3 & 0 & 4 & - \end{bmatrix}$$

(b) NCV.

$$P = \begin{bmatrix} - & 0 & 0 & 6 & 6 \\ 0 & - & 0 & 6 & 6 \\ 0 & 0 & - & 0 & 0 \\ 6 & 6 & 0 & - & 0 \\ 6 & 6 & 0 & 0 & - \end{bmatrix}$$

Source: Prepared by the author.

Note that due the property of CNOT with T gate the cost matrix for NCV circuits is symmetric. To generate an element a_{ij} and a_{ji} of the NCV matrix the lower cost element between a_{ij} and a_{ji} from Clifford+T matrix is chosen and multiplied by 2, since there are two CNOT gates in the mapped circuit.

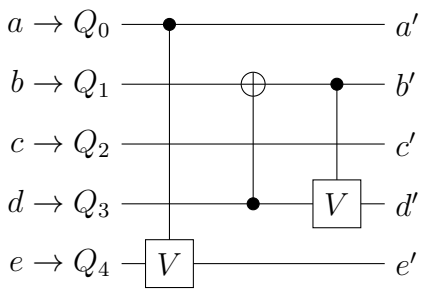
The matrix cost of NCV circuits can be used to find the best permutation trough the Integer Linear Programming formulation, hence, finding a lower cost Clifford+T circuit to be used in IBM QX architectures.

5.1.1 NCV Example

An example will be used to present an application of the ILP formulation in a NCV circuit. The goal of this example is to map the NCV circuit to the IBM QX4. Figure 54 shows the example circuit with the trivial mapping.

From the matrix with the costs of the Clifford+T and NCV gates in Figure 53, the

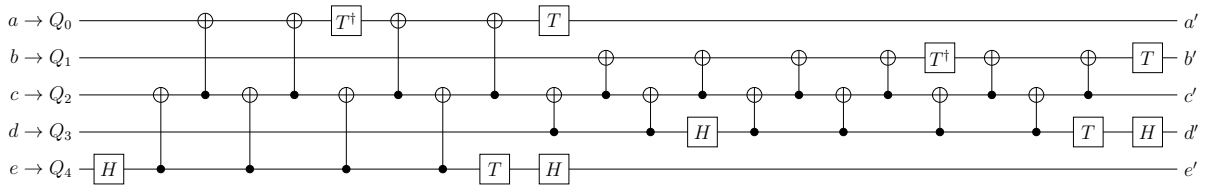
Figure 54 – Example to map NCV circuit to Clifford+T.



Source: Prepared by the author.

trivial mapping has cost 15. Thus, the Clifford+T mapping will result in a circuit with 30 gates and 22 levels. Figure 55 shows the Clifford+T mapped circuit.

Figure 55 – Example of NCV circuit mapped to Clifford+T.



Source: Prepared by the author.

In order to find the best permutation to map the circuit from Figure 54, the ILP formulation is used. Note that the variables for the CNOT and for the V and $V^\dagger$ gates are represented with different letters in the formulation. Also, since the V and $V^\dagger$ gates have a single control and a target (same as CNOT), the formulation will be realized similarly of the CNOT. Note that the Equation 16 and Equation 17 can be used as well. Also, the matrix cost, T and P, used for each gate is from Figure 53. The formulation for this circuit is presented in Equation 21.

$$
\begin{aligned}
&minimize \\
&\sum_{m=0}^{4}\sum_{n=0}^{4} 1.P_{mn}V_{04}C_{mn} \; + \sum_{m=0}^{4}\sum_{n=0}^{4} 1.T_{mn}G_{31}C_{mn} \; + \sum_{m=0}^{4}\sum_{n=0}^{4} 1.P_{mn}V_{13}C_{mn}, \; \forall\, m \neq n \\
&Subject\,To
\end{aligned}
\tag{21}
$$

$$
\begin{cases}
For\ each\ qubit\ i: \\
1 * (\sum_{n=0}^{4} V_{13} C_{in}) = \sum_{n=0}^{4} G_{31} C_{ni} \\
1 * (\sum_{n=0}^{4} V_{13} C_{ni}) = \sum_{n=0}^{4} G_{31} C_{in} \\
\quad \sum_{n=0}^{4} V_{04} C_{in} + \sum_{n=0}^{4} G_{31} C_{ni} + \sum_{n=0}^{4} G_{31} C_{in} + \sum_{n=0}^{4} V_{04} C_{ni} \leq 1 \\
\quad \forall\, n \neq i
\end{cases}
$$

$$\sum_{m=0}^{4} \sum_{n=0}^{4} V_{04} C_{mn} = 1,\ \forall\, m \neq n$$

$$\sum_{m=0}^{4} \sum_{n=0}^{4} G_{31} C_{mn} = 1,\ \forall\, m \neq n$$

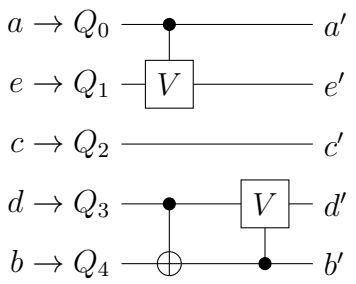
$$\sum_{m=0}^{4} \sum_{n=0}^{4} V_{31} C_{mn} = 1,\ \forall\, m \neq n$$

$$All\ variables\ are\ integers$$

Solving this ILP problem a permutation for the circuit in Figure 54 is found. One solution is $V_{0_4}C_{0_1} = 1$, $G_{3_1}C_{3_4} = 1$, $V_{1_3}C_{4_3} = 1$, and all the other variables are 0. For this solution the objective function has cost 0, meaning that with this permutation no additional gates are needed for the circuit to comply with the IBM QX4 constraints.

From this solution the permutation can be found. The variable $V_{0_4}C_{0_1}$ shows that the qubit a must be mapped to Q_0 and the qubit e to Q_1. In the same way, the $G_{3_1}C_{3_4}$ shows that the qubit d must be mapped to Q_3 and the qubit b must be mapped to Q_4. The last variable, $V_{1_3}C_{4_3}$ confirms the mapped qubits. The mapped circuit is presented in Figure 56.

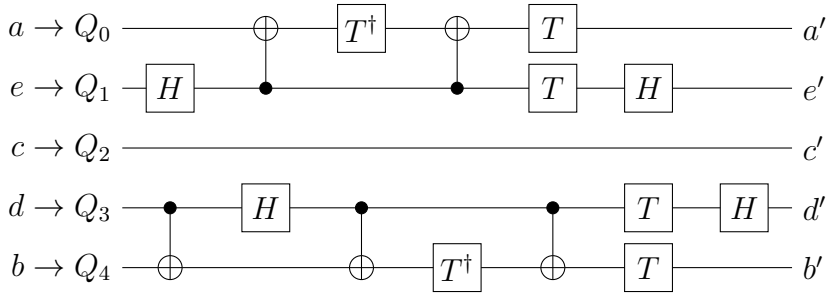
Figure 56 – Example of NCV circuit mapped after ILP formulation.



Source: Prepared by the author.

In order to map the NCV circuit to Clifford+T, the V gates must be mapped to a Clifford+T circuit. The resulting circuit is shown in Figure 57.

Figure 57 – Example of NCV circuit mapped to Clifford+T after ILP formulation.



Source: Prepared by the author.

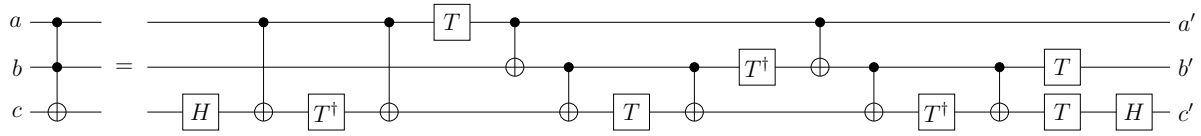
The final circuit has 15 gates and 7 levels. It showed that permuting the qubits can reduce the need of additional gates in the circuit. In this case, it even eliminated the need of additional gates.

The next section will explore the mapping of Toffoli circuits to Clifford+T circuits. After the mapping of Toffoli circuits, the correspondent Clifford+T circuit can be mapped to an IBM QX architecture with reduced cost.

5.2 TOFFOLI TO CLIFFORD+T CIRCUITS

The CNOT mappings and the ILP formulation will be used to map a Toffoli gate to an IBM QX architecture. This mapping will show how the ILP formulation can be easily adaptable to work with Toffoli circuits as well. In order to use Toffoli circuits in the IBM architectures it must be mapped to Clifford+T circuits. There are alternatives to map a Toffoli to a Clifford+T circuit. A possible mapping with 17 gates is shown in Figure 58.

Figure 58 – Toffoli implementation with Clifford+T gates using 17 gates.

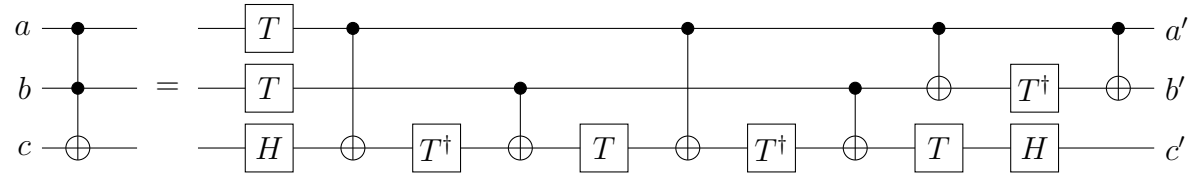


Source: Adapted from (MILLER; SOEKEN; DRECHSLER, 2014).

This mapping can be optimized as the authors Nam *et al.* (2018) presented a smaller Clifford+T equivalence. Applying some optimization rules from Soeken and Thomsen (2013) to this mapping is possible to reduce its number of CNOT gates. The detailed optimization process can be seen in Appendix B.1. Figure 59 shows the optimized mapping.

The mapping has 15 gates and 11 levels, with 6 CNOT, two less compared with the

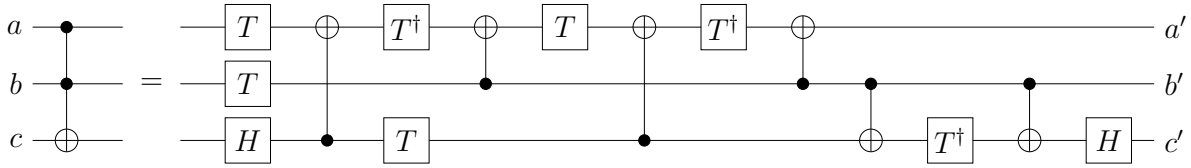
Figure 59 – Toffoli implementation with Clifford+T gates using 15 gates.



Source: Prepared by the author.

previous one. That means that the mapped circuit complying with IBM architectures will be smaller. However, another possible mapping with the same number of gates can be found in Figure 58, as presented by Amy *et al.* (2013). Applying a different sequence of optimizations, the circuit presented in Figure 60 can be found. The step-by-step optimization can be seen in Appendix B.2.

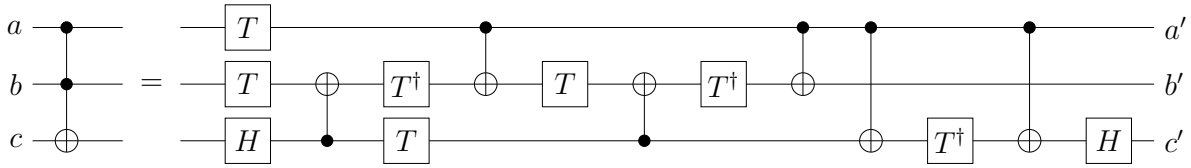
Figure 60 – Another Toffoli implementation with Clifford+T gates using 15 gates.



Source: Prepared by the author.

This mapping has 15 gates and 12 levels. The position of the CNOT gates in this circuit can reduce the number of misplaced gates depending on the architecture constraints. Another change that can reduce the number of misplaced CNOT gates is to swap the controls qubits of the Toffoli gate. Note that the controls can be interchanged. Figure 61 shows the Toffoli with the controls swapped.

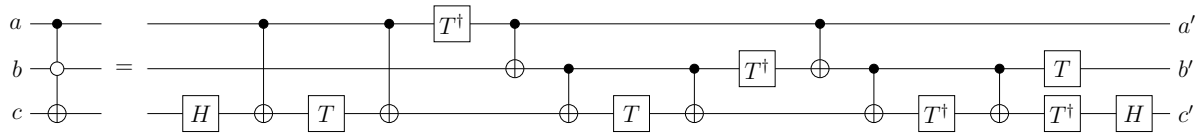
Figure 61 – Another Toffoli implementation with Clifford+T gates using 15 gates and controls swapped.



Source: Prepared by the author.

There are three possible mappings for a Toffoli to Clifford+T with 15 gates. The idea is to use the lower cost one with the CNOT mappings depending on the target architecture. The next step is to find the mappings with the same structures for the Toffoli with mixed polarities and with both negative controls. Figure 62 shows the Toffoli to Clifford+T with 17 gates.

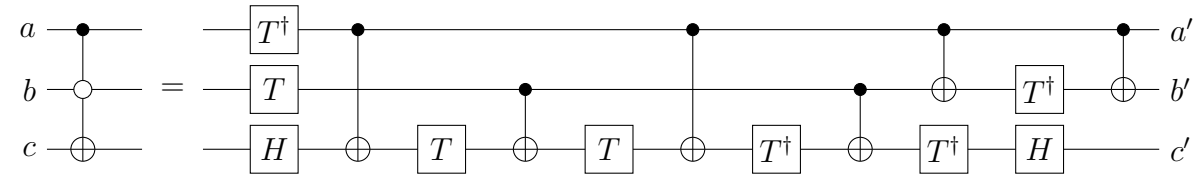
Figure 62 – Toffoli with positive-negative controls to Clifford+T with 17 gates.



Source: Prepared by the author.

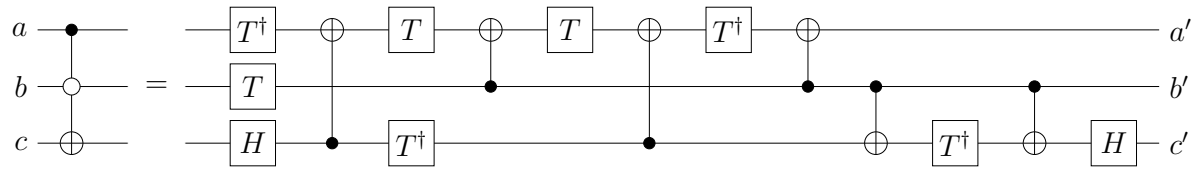
Applying the same sequence of optimizations for both structures it was possible to find the mappings for the Toffoli with positive-negative controls. Figure 63 and Figure 64 show the mappings.

Figure 63 – Positive-negative controls Toffoli mapped to Clifford+T with 15 gates.



Source: Prepared by the author.

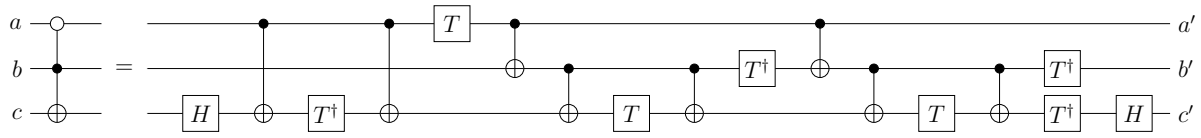
Figure 64 – Positive-negative controls Toffoli mapped to Clifford+T with 15 gates and CNOT changed.



Source: Prepared by the author.

In the mapping of Toffoli with mixed polarities swapping the controls will result in a different circuit. To solve this problem, the mapping of Toffoli with negative-positive controls must be found. Figure 65 presents the negative-positive Toffoli with 17 gates to be optimized.

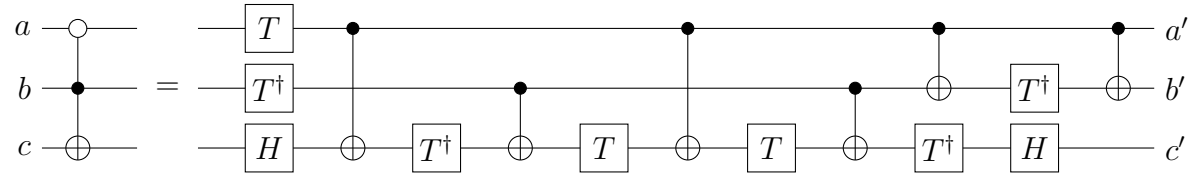
Figure 65 – Toffoli with negative-positive controls to Clifford+T with 17 gates.



Source: Prepared by the author.

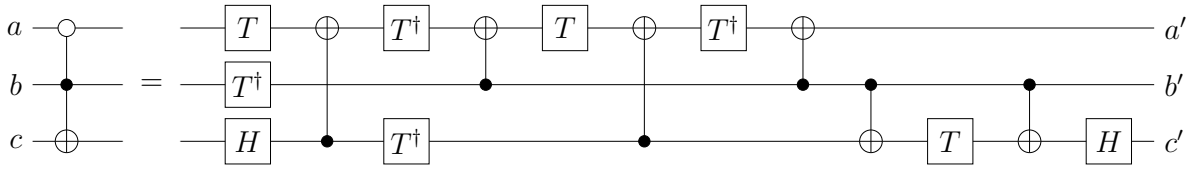
Optimizing the circuit, both structures can be found. Figure 66 and Figure 67 show the mapping of the Toffoli with negative-positive controls.

Figure 66 – Negative-positive controls Toffoli realized with 15 Clifford+T gates.



Source: Prepared by the author.

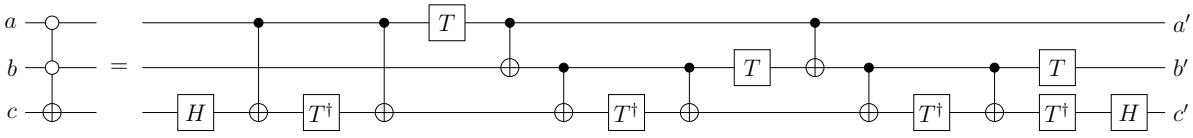
Figure 67 – Another negative-positive controls Toffoli realized with 15 Clifford+T gates.



Source: Prepared by the author.

With both circuits of the Toffoli with mixed polarities is possible to swap the controls in order to reduce the need of additional gates. The remaining Toffoli is the one with both negative controls that is shown in Figure 68.

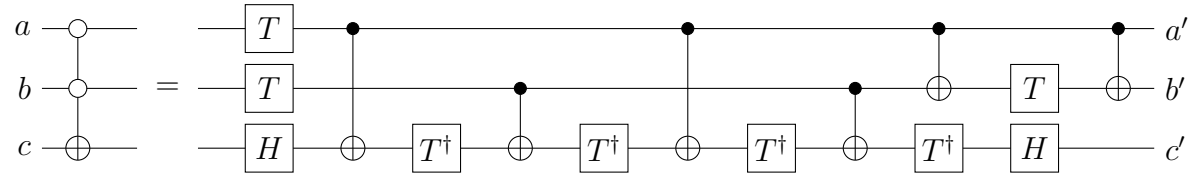
Figure 68 – Toffoli with negative polarity to Clifford+T with 17 gates.



Source: Prepared by the author.

Applying the sequence of optimizations the resulted circuits are shown in Figure 69 and Figure 70.

Figure 69 – Negative controls Toffoli mapped to Clifford+T with 15 gates.

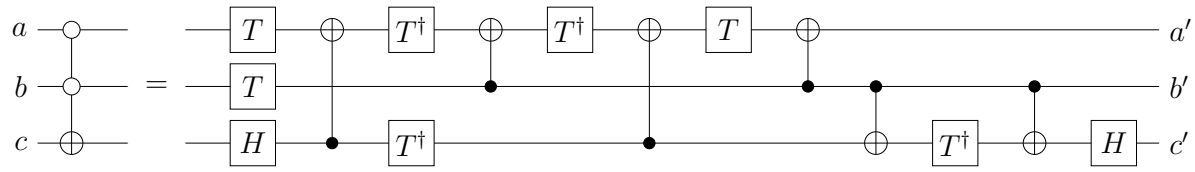


Source: Prepared by the author.

The Toffoli controls of the mapping from Figure 70 can be swapped without changing the circuit equivalence. The circuit with the controls swapped is shown in Figure 71.

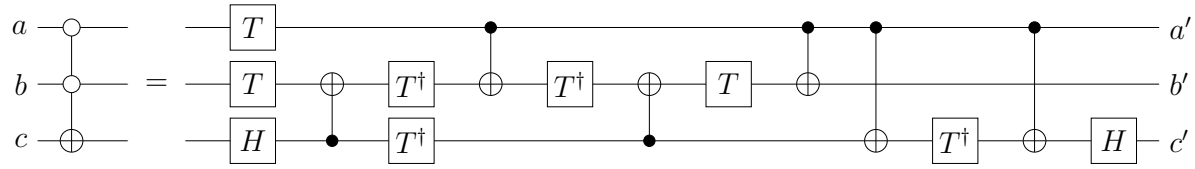
With the mapping of each Toffoli and the costs of each CNOT, it is possible to map the Toffoli using the lower cost CNOT gates. For example, consider the Toffoli

Figure 70 – Another negative controls Toffoli mapped to Clifford+T with 15 gates.



Source: Prepared by the author.

Figure 71 – Another negative controls Toffoli mapped to Clifford+T with 15 gates and controls swapped.



Source: Prepared by the author.

with both negative controls with the qubits $\{a, b, c\}$ mapped to $\{Q_0, Q_1, Q_2\}$ of the IBM QX4 (see matrix cost in Figure 26). For the mapping from Figure 69 the cost is presented in Equation 22. Note that the CNOT(Q_1, Q_0) was used due the lower cost (the CNOT(Q_0, Q_1) can be swapped without additional cost).

$$2 * CNOT(Q_0, Q_2) + 2 * CNOT(Q_1, Q_2) + 2 * CNOT(Q_1, Q_0) = 2 * 4 + 2 * 4 + 2 * 0 = 16 \quad (22)$$

Using this Toffoli mapping will require 16 additional gates to the circuit. The cost for the mapping from Figure 70 is shown in Equation 23.

$$2 * CNOT(Q_2, Q_0) + 2 * CNOT(Q_1, Q_0) + 2 * CNOT(Q_2, Q_1) = 2 * 0 + 2 * 0 + 2 * 0 = 0 \quad (23)$$

This Toffoli mapping can be used without additional gates. The third possible mapping is the one from Figure 71. The respective cost is presented in Equation 24.

$$2 * CNOT(Q_2, Q_1) + 2 * CNOT(Q_0, Q_1) + 2 * CNOT(Q_2, Q_0) = 2 * 0 + 2 * 4 + 2 * 0 = 4 \quad (24)$$

The cost of this mapping is 4, i.e., four additional gates are required to map the

Toffoli. In this case, the mapping that must be used is the one from Figure 70, since it will not add any additional gate to the circuit.

However, the problem is to find a good permutation to apply the mapping. To solve this permutation problem, the ILP formulation presented in the previous chapter can be used. Note that the matrix cost for a Toffoli gate must be three dimensional. To avoid the generation of this matrix, the cost of each Toffoli gate is calculated from the CNOT matrix while the ILP formulation is generated. The next subsection presents an example of this mapping.

5.2.1 Toffoli Example

Calculating the costs of each mapping, it is possible to use the ILP formulation to find the lowest cost permutation for a Toffoli circuit. Note that the Toffoli has two controls and a target, hence, the number of variables will increase. For example, for a 5 bit circuit, the number of possible Toffoli gates is 60. Therefore, the Equation 16 and the Equation 17 need to be updated. The updated equations are presented in Equation 25 and Equation 26, where v_t and c_t are the numbers of variables and constraints of a Toffoli circuit, n is the number of qubits, y is the number of different Toffoli gates in the circuit and x is the number of qubits with at least two acting Toffoli gates.

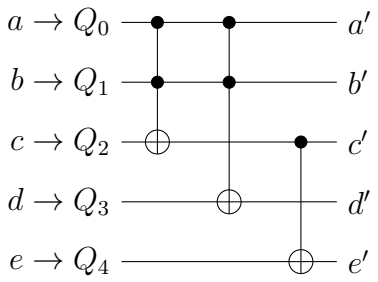
$$v_t = (n^2 - n) * (n - 2) * y, \ \forall\, n > 2, \ y > 0 \tag{25}$$

$$\begin{cases} c_t = (n * x) + (n + y), \ \forall\, n > 1, \ y > 1, \ x \geq 0 \\ c_t = 1, \ \forall\, n > 1, \ y = 1, \ x = 0 \end{cases} \tag{26}$$

In the case of a circuit consisting of Toffoli and CNOT gates, the number of variables will be the sum of Equation 16 and Equation 25. However, the number of constraints will be found making the variable x equal the number of qubits with at least two acting controlled gates and y the number of different Toffoli and CNOT gates in the circuit.

The application of the ILP formulation is shown with the circuit in Figure 72. The goal is to map this circuit to the IBM QX4 architecture. The number of variables and constraints are presented in Equation 27 and in Equation 28, respectively.

Figure 72 – Example circuit to find the lowest cost permutation of a Toffoli circuit.



Source: Prepared by the author.

$$
\begin{aligned}
v_{circuit} &= v_c + v_t = (n^2 - n) * y + (n^2 - n) * (n - 2) * y \\
v_{circuit} &= (5^2 - 5) * 1 + (5^2 - 5) * (5 - 3) * 2 \\
v_{circuit} &= 20 + 120 = 140
\end{aligned}
\tag{27}
$$

$$
c_{circuit} = (n * x) + (n + y) = (5 * 3) + (5 + 3) = 23 \tag{28}
$$

The ILP problem for the circuit in Figure 72 will have 120 variables and 23 constraints and is presented in Equation 29.

$$
\begin{aligned}
&minimize \\
&\quad \sum_{m=0}^{4}\sum_{n=0}^{4}\sum_{k=0}^{4} 1.X_{mnk}T_{012}C_{mnk} + \\
&\quad \sum_{m=0}^{4}\sum_{n=0}^{4}\sum_{k=0}^{4} 1.X_{mnk}T_{013}C_{mnk} + \sum_{m=0}^{4}\sum_{n=0}^{4} 1.M_{mn}G_{24}C_{mn},\ \forall\, m \neq n \neq k \\
&Subject\ To
\end{aligned}
\tag{29}
$$

$$
\begin{cases}
For\ each\ qubit\ i: \\
1 * (\sum_{m=0}^{4}\sum_{n=0}^{4} T_{012}C_{imn}) = \sum_{m=0}^{4}\sum_{n=0}^{4} T_{013}C_{imn} \\
1 * (\sum_{m=0}^{4}\sum_{n=0}^{4} T_{012}C_{min}) = \sum_{m=0}^{4}\sum_{n=0}^{4} T_{013}C_{min} \\
1 * (\sum_{n=0}^{4} G_{24}C_{in}\ ) = \sum_{m=0}^{4}\sum_{n=0}^{4} T_{012}C_{mni} \\
\sum_{m=0}^{4}\sum_{n=0}^{4} T_{012}C_{imn} + \sum_{m=0}^{4}\sum_{n=0}^{4} T_{012}C_{min} + \sum_{m=0}^{4}\sum_{n=0}^{4} T_{012}C_{mni} + \sum_{n=0}^{4} G_{24}C_{in}\ \leq 1 \\
\forall\, m \neq n \neq i
\end{cases}
$$

$$
\sum_{m=0}^{4}\sum_{n=0}^{4}\sum_{k=0}^{4} T_{012}C_{mnk}\ = 1,\ \forall\, m \neq n \neq k
$$

$$
\sum_{m=0}^{4}\sum_{n=0}^{4}\sum_{k=0}^{4} T_{013}C_{mnk}\ = 1,\ \forall\, m \neq n \neq k
$$

$$
\sum_{m=0}^{4}\sum_{n=0}^{4} G_{24}C_{mn} = 1,\ \forall\, m \neq n
$$

$$
All\ variables\ are\ integers
$$

The X_{mnk} are the costs of each Toffoli gate that is calculated using the CNOT gates costs. The M_{mn} are the CNOT gates costs. The T_{012} and T_{013} are the Toffoli gates and the G_{24} is the CNOT gate from the circuit.

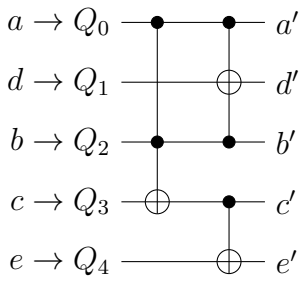
Solving this ILP problem result in an objective value of 6, meaning that 6 gates must be added to the circuit to comply with the IBM QX4 constraints. One possible solution for this ILP problem is $T_{0_1_2}C_{0_2_3} = 1$, $T_{0_1_3}C_{0_2_1} = 1$, $G_{2_4}C_{3_4} = 1$ and all the other variables are 0.

From this solution the permutation can be found, hence, $\{a, b, c, d, e\}$ are mapped to $\{Q_0, Q_2, Q_3, Q_1, Q_4\}$, respectively. The permuted circuit is shown in Figure 73.

After the permutation is found, the Toffoli gates must be mapped to a Clifford+T circuit. As the circuit is already permuted to the lowest cost permutation, the Toffoli gates can be mapped to a Clifford+T circuit. Figure 74 shows the Clifford+T circuit. The circuit has 31 gates (15 gates from each Toffoli plus the CNOT gate).

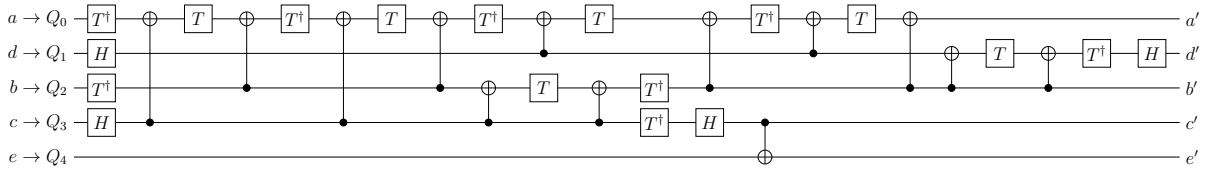
The circuit is targeting the IBM QX4, therefore both CNOT(Q_3, Q_0) gates must be mapped to comply with the architecture restrictions. This mapping will add 6 gates to the circuit, as expected with the result of the ILP problem. The final circuit is shown in

Figure 73 – Example circuit mapped to the lowest cost permutation.



Source: Prepared by the author.

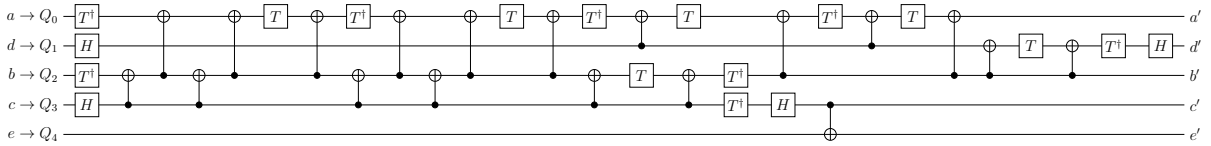
Figure 74 – Lowest cost permutation mapped to Clifford+T.



Source: Prepared by the author.

Figure 75.

Figure 75 – Lowest cost permutation mapped to IBM QX4 architecture.



Source: Prepared by the author.

The circuit has 37 gates with 25 levels and all the CNOT gates comply with the IBM QX4 restrictions. With this approach the Toffoli circuits can be mapped to Clifford+T targeting the IBM QX architectures a lower cost. The next section presents some experiment of the NCV and Toffoli approaches.

5.3 EXPERIMENTAL RESULTS OF NCV AND TOFFOLI MAPPINGS

In order to evaluate the proposed approach in this chapter, a series of tests were performed. To perform the tests, Qiskit and the Standalone algorithm are used. However, both of these algorithms do not target the mapping of NCV or Toffoli circuits to IBM architectures.

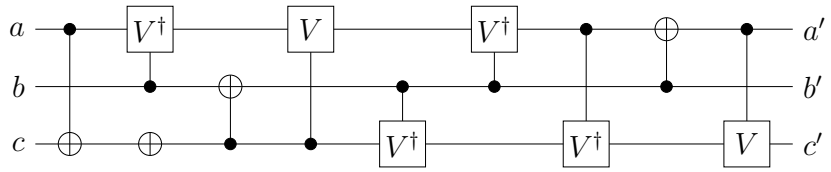
Since the algorithms used to evaluate the developed approaches do not realize the mapping of NCV and Toffoli circuits to IBM architectures, the circuits will be mapped to

Clifford+T and then the algorithms are used. All the tests were executed in an Ubuntu operating system, Intel i5 processor and 8 GB of RAM.

5.3.1 NCV mapping evaluation

For the first test the mapping strategy presented by Miller, Soeken and Drechsler (2014). This mapping does not target the IBM architecture, but it will be used to verify how the proposed approach performs. The goal is to map the circuit 3_17 (MILLER; SOEKEN; DRECHSLER, 2014) to the IBM QX4. This NCV circuit, with 10 gates and 9 levels, is shown in Figure 76. The goal of this test is to check if a mapping using the approach presented by Miller, Soeken and Drechsler (2014) combined with the CNOT mappings will result in a circuit with fewer gates than the approach presented in this chapter.

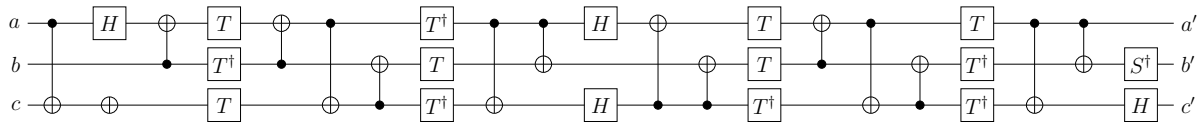
Figure 76 – Circuit 3_17a.



Source: Adapted from (MILLER; SOEKEN; DRECHSLER, 2014).

The authors Miller, Soeken and Drechsler (2014) mapped the circuit of Figure 76 to Clifford+T without taking in account the constraints of any quantum architecture. The goal was to reduce the T-depth of the circuit, i.e, the number of T gates in parallel. The mapped circuit (3_17e) with 32 gates and 20 levels is presented in Figure 77.

Figure 77 – Circuit 3_17a mapped to Clifford+T (3_17e).

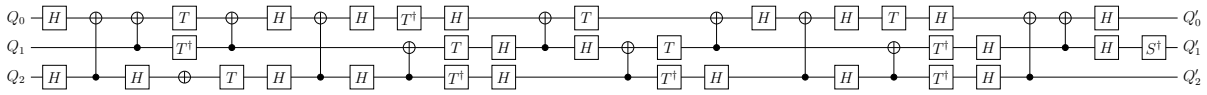


Source: Adapted from (MILLER; SOEKEN; DRECHSLER, 2014).

Some CNOT gates from Figure 77 must be changed to comply with the IBM's QX4 constrains. Mapping the misplaced CNOT gates and optimizing it will result in a circuit with 46 gates and 24 levels.

Another approach is to apply the ILP formulation to the circuit of Figure 77. Solving the ILP, the lowest cost permutation for this circuit is the qubits $\{a, b, c\}$ mapped to

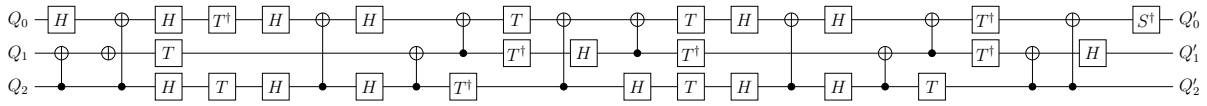
Figure 78 – Circuit 3_17e mapped to IBM QX4 using the CNOT mapping.



Source: Prepared by the author.

$\{Q_2, Q_0, Q_1\}$. Applying the CNOT mappings to the permuted circuit and then optimizing it results in a circuit with 40 gates and 22 levels. The circuit is shown in Figure 79.

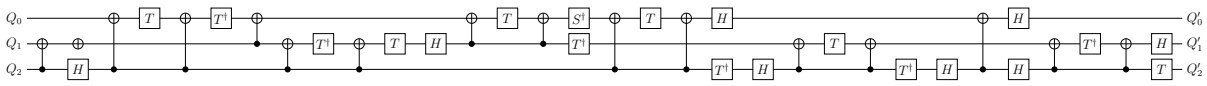
Figure 79 – Circuit 3_17e mapped to IBM QX4 using the ILP formulation.



Source: Prepared by the author.

To evaluate the NCV mapping, the circuit from Figure 76 was formulated as an ILP problem. The solution of the ILP problem ensures the lowest cost permutation for the NCV mapping regarding the CNOT mappings. The permutation found is the qubits $\{a, b, c\}$ mapped to $\{Q_2, Q_0, Q_1\}$. The resulting mapped circuit has 37 gates and 30 levels. This circuit is shown in Figure 80.

Figure 80 – Mapped 3_17 circuit using NCV mapping approach.



Source: Prepared by the author.

The NCV mapping approach found a circuit with 19.5% fewer gates compared with the mapped 3_17e circuit. However, in the number of levels it was worse, the mapped 3_17e had 20% fewer levels. It is due the fact that the circuit 3_17e was mapped to Clifford+T targeting the *T-depth*. Applying the ILP formulation to the circuit 3_17e reduced the number of gates in 13% and the number of levels in 8.3% compared with the CNOT mapping without changing the permutation. The NCV mapping found a smaller circuit compared with the circuit 3_17e mapped using ILP formulation regarding the number of gates. The difference was 7.5% fewer gates. As for the number of levels, the difference increased with the use of the ILP formulation in the circuit 3_17e. Compared both ILP formulations, the circuit 3_17e found 26.6% fewer levels. As the CNOT mapping was done targeting the number of gates, it was expected to get worse results regarding the number of levels.

The next test was done with NCV circuits with more than 10 qubits. Since RevLib did not have NCV circuits with more than 10 qubits, the circuits were randomly created

with V and CNOT gates. The goal of this test is to map NCV circuits to the IBM QX5 architecture. To perform this test, the NCV circuits were mapped to Clifford+T circuits using the equivalence presented in Figure 47 and then applied in the mapping algorithms.

The first comparison was done with the algorithm from Qiskit. One characteristic of the mapping algorithm Qiskit is that outputs a circuit designed with parameterized U gates and CNOT gates. In order to compare the same type of circuit, the circuits found with NCV mapping were optimized to this set of gates using Qiskit. Table 17 shows the results of this comparison. The first column, **Benchmarks**, presents the name of the circuits. The columns below the **Original** cell show the characteristics of the original circuit. The number of gates and the number of levels of the circuits are shown in the columns **Gates** and **Levels**, respectively. The number of qubits of the original circuit is presented in the **Qubits** column. The characteristics of the circuits obtained with Qiskit are shown in the columns in **Qiskit**. In the same way, the columns in **NCV Mapping** shows the characteristics of the circuits obtained with the NCV mapping algorithm. The column **Time(sec)** is the time needed, in seconds, for the ILP problem to be solved. The comparison of the approaches, in percentage, is presented in the columns in **%Improvements**.

Table 17 – Mapping NCV circuits compared with Qiskit in IBM QX5.

	Original			Qiskit			NCV Mapping			%Improvements	
Circuits	**Qubits**	**Gates**	**Levels**	**Gates**	**Levels**	**T(sec)**	**Gates**	**Levels**	**T(sec)**	**Gates**	**Levels**
ncv_15	15	51	36	804	179	7	379	91	224	52.86	49.16
ncv_13_2	13	33	17	628	117	6	301	96	661	52.07	17.95
ncv_11	11	45	23	712	176	7	349	146	165	50.98	17.05
ncv_12	12	36	25	484	100	6	238	74	116	50.83	26.00
ncv_10	10	53	32	796	158	7	395	161	78	50.38	-1.90
ncv_16	16	53	27	960	168	8	485	138	944	49.48	17.86
ncv_14	14	74	33	580	101	7	310	92	73	46.55	8.91
ncv13_1	13	55	27	725	174	6	412	108	1,319	43.17	37.93

Source: Prepared by the author.

The NCV mapping found smaller circuits for all the tests. It was expected since Qiskit does not target the direct implementation of NCV circuits. However, the difference in the size of the circuit was up to 52% for the *ncv_15* with 49% fewer levels. For the circuit *ncv_14* with 14 qubits, the improvements was of 46% in the number of gates with 8% in the number of levels. However, in the case of circuit *ncv_10* with half the size, the number of levels was worse. It may occur due the optimizations done by Qiskit and the fact that the proposed NCV Mapping target the size of the circuit. These tests have shown that a direct approach to map NCV circuits targeting an IBM architecture can reduce the circuits' size. The same test was executed with Standalone algorithm.

The second comparison was done to compare the NCV mapping with the Standalone

algorithm with Qiskit optimizations. Results of this comparison is presented in Table 18. The structure of the Table is the same of the previous one. The column **Standalone** shows the characteristic of the circuits found with the Standalone algorithm.

Table 18 – Mapping NCV circuits compared with Standalone optimized in IBM QX5.

	Original			Qiskit			NCV Mapping			%Improvements	
Circuits	**Qubits**	**Gates**	**Levels**	**Gates**	**Levels**	**T(sec)**	**Gates**	**Levels**	**T(sec)**	**Gates**	**Levels**
ncv_14	14	74	33	622	145	4	310	92	73	50.16	36.55
ncv_16	16	53	27	744	199	5	485	138	944	34.81	30.65
ncv_11	11	45	23	531	144	3	349	146	165	34.27	-1.39
ncv_10	10	53	32	585	190	5	395	161	78	32.48	15.26
ncv_15	15	51	36	507	140	3	379	91	224	25.25	35.00
ncv_13_1	13	55	27	532	129	3	412	108	1,319	22.56	16.28
ncv_12	12	36	25	288	98	3	238	74	116	17.36	24.49
ncv_13_2	13	33	17	363	102	3	301	96	661	17.08	5.88

Source: Prepared by the author.

As expected, the direct mapping from NCV circuits to IBM architectures shows better results than the Standalone algorithm. The best improvement was, the circuit *ncv_14*, with 14 qubits with 50% improvement in the number of gates and 36% in the number of levels. However, for the circuit *ncv_11*, Qiskit mapped a circuit with fewer levels with 34% more gates. This resulted from the movement of qubits trough the circuit and shows that the proposed mapping only takes into account the size of the circuit.

The ideal evaluation would be comparing algorithms with the same goal. However, it shows that the proposed NCV mapping can result in smaller circuits if a mapping targeting a specific architecture direct from the NCV circuit is used. The next step to find an algorithm open source mapping the NCV algorithms to evaluate the developed mapping strategy. Another approach is to optimize the NCV circuit before the mapping instead of optimize the Clifford+T mapping.

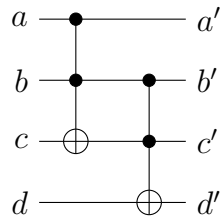
The next mapping to test it the Toffoli mappings. The other approach to be evaluated is the mapping of Toffoli circuits to IBM architectures. For the first test a mapping of Toffoli circuit to Clifford+T from Miller, Soeken and Drechsler (2014) is going to be used. For the second test, the Toffoli circuits were mapped to Clifford+T using the equivalences presented in Section 5.2.

5.3.2 Toffoli mapping evaluation

The first test was to compare the Toffoli mapping strategy with a mapping presented in (MILLER; SOEKEN; DRECHSLER, 2014). This mapping does not target the constrains of an IBM architecture. Thus, the CNOT mapping will be used. The goal of this test is to map the Toffoli circuit to IBM QX4. The circuit used for this test is the

circuit a2x from Miller, Soeken and Drechsler (2014) and is presented in Figure 81.

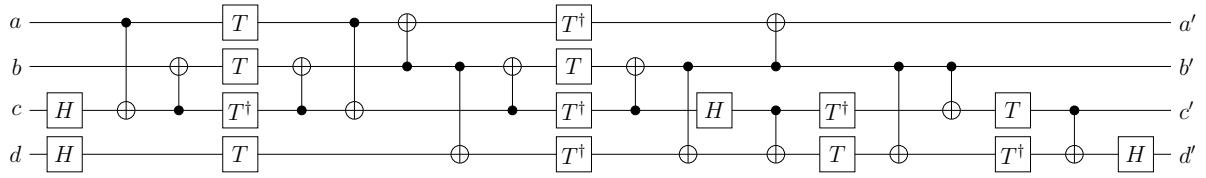
Figure 81 – Toffoli circuit a2x_a.



Source: Adapted from (MILLER; SOEKEN; DRECHSLER, 2014).

Transforming the circuit using the approach from Miller, Soeken and Drechsler (2014) result in the circuit with 30 gates and 17 levels that is shown in Figure 82. This approach targets the *T-depth* of the circuit. Hence, some CNOT must be changed to comply with the IBM QX4 constraints.

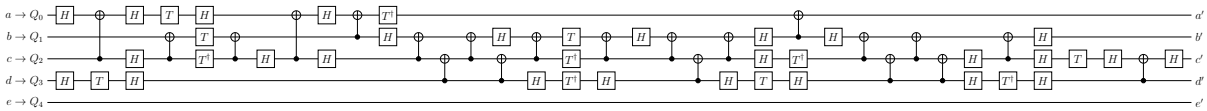
Figure 82 – Clifford+T circuit a2x_e.



Source: Adapted from (MILLER; SOEKEN; DRECHSLER, 2014).

Mapping the misplaced CNOT gates from the Figure 82 with the proposed movements results in the circuit presented in Figure 83. The circuit has 59 gates and 35 levels.

Figure 83 – Circuit a2x_e mapped with default permutation to IBM QX4.

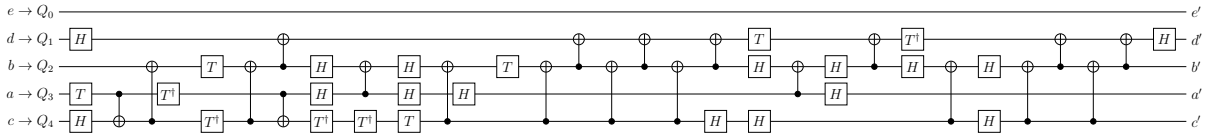


Source: Prepared by the author.

In order to find a circuit with fewer gates, the permutation can be changed. To do that, the circuit from Figure 82 was formulated with the ILP. The permutation found was {a, b, c, d, e} mapped to {3, 2, 4, 1, 0}. Figure 84 shows the circuit. The circuit has 46 gates and 29 levels, a reduction of 22% and 17% in the number of gates and levels, respectively.

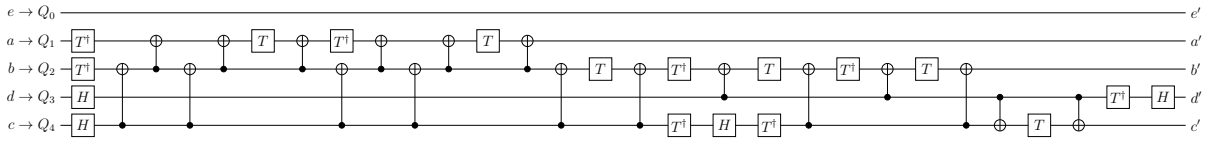
Applying the Toffoli mapping approach to the circuit from Figure 81 and solving the ILP formulation, permutation {a, b, c, d, e} mapped to {1, 2, 4, 3, 0} was found. This permutation leads to a circuit with 36 gates and 28 levels that is presented in Figure 85.

Figure 84 – Circuit a2x_e mapped with best permutation to IBM QX4.



Source: Prepared by the author.

Figure 85 – Circuit a2x mapped to IBM QX4 with proposed approach.



Source: Prepared by the author.

Comparing the Toffoli mapping approach with the default permutation from Figure 83, reduced the number of gates in 39% and 20% in the number of levels. This difference is due the fact that the approach from (MILLER; SOEKEN; DRECHSLER, 2014) target the *T-depth* of the circuit and do not take in account the IBM architecture constraints. To reduce the size of the circuit, the best permutation for the circuit from Figure 83 was found. Compared with this permuted circuit from Figure 84, the reduction was of 22% and 3% regarding the number of gates and levels, respectively. In the same way, the goal of the compared approach is the *T-depth* and it can be noted that the number of levels of the mapped circuit was almost the same.

The next evaluation was to map a set of Toffoli circuits to IBM QX4 and QX5. For this, a set of reversible circuits were used as benchmarks (WILLE *et al.*, 2008). The goal is to compare the Toffoli mapping with the Standalone and Qiskit mapping algorithms. As both algorithms only work with Clifford+T circuits, the reversible circuits were transformed using the Toffoli equivalences presented in Figure 59. Then, the mapped circuits were realized on IBM QX4 and QX5 using each algorithm.

The first evaluation was to compare the Qiskit and Standalone algorithms with the Toffoli mapping targeting the IBM QX4. The permutation of the Toffoli mapping were found using the ILP formulation. The benchmarks results are shown in Table 19. The first column, **Circuits**, is the name of each circuit used to evaluate the proposed approach. The columns below **Original** are the characteristic of the original circuit. The number of gates and the number of levels are presented in the columns **Gates** and **Levels**, respectively. The columns below **Qiskit** cell are the number of gates and levels of the mapped circuit using Qiskit. Similarly, the columns of **Toffoli Mapping** are the characteristics of the circuits after the Toffoli mapping algorithm applied. The columns of **%Improvements**

show the difference between the two approaches in percentage.

Table 19 – Mapping Toffoli circuits compared with Qiskit in IBM QX4.

	Original			Qiskit		Toffoli Mapping		%Improvements	
Circuits	**Qubits**	**Gates**	**Levels**	**Gates**	**Levels**	**Gates**	**Levels**	**Gates**	**Levels**
alu-v3_34	5	7	5	139	74	43	32	69.06	56.76
mod5d1_63	5	7	5	80	40	26	18	67.50	55.00
mod5d2_70	5	8	6	117	68	43	31	63.25	54.41
alu-v0_26	5	9	6	193	112	76	52	60.62	53.57
rd32_271	5	9	4	208	120	82	50	60.58	58.33
alu-v1_28	5	7	6	93	52	37	26	60.22	50.00
alu-v1_29	5	7	6	86	45	35	26	59.30	42.22
4mod7-v0_95	5	12	7	390	227	164	121	57.95	46.70
one-two-three-v3_101	5	8	7	182	93	79	56	56.59	39.78
4gt5_75	5	8	4	196	117	86	52	56.12	55.56
one-two-three-v2_100	5	8	7	136	77	60	43	55.88	44.16
mod5d2_64	5	8	6	116	66	57	40	50.86	39.39
4mod7-v0_94	5	12	8	300	170	153	113	49.00	33.53
4gt5_77	5	10	6	246	138	127	94	48.37	31.88
4mod5-v0_18	5	9	5	118	66	61	48	48.31	27.27
4gt13_90	5	17	13	244	121	128	85	47.54	29.75
4mod5-v1_23	5	8	8	129	75	68	53	47.29	29.33
rd32_270	5	9	4	149	95	82	50	44.97	47.37
4gt13_91	5	13	9	206	118	116	77	43.69	34.75
4gt10-v1_81	5	12	7	279	152	163	117	41.58	23.03
alu-v2_31	5	31	19	751	422	480	354	36.09	16.11
alu-v2_33	5	7	3	56	33	36	28	35.71	15.15
hwb5_55	5	39	23	669	356	432	308	35.43	13.48
one-two-three-v1_99	5	11	7	212	114	138	100	34.91	12.28
alu-v4_37	5	7	6	56	30	37	27	33.93	10.00
alu-v2_32	5	13	8	266	138	176	128	33.83	7.25
4gt13-v1_93	5	7	5	82	44	55	39	32.93	11.36
alu-v3_35	5	7	6	54	34	37	27	31.48	20.59
4mod7-v1_96	5	13	8	227	135	164	121	27.75	10.37
one-two-three-v0_98	5	11	6	224	127	163	122	27.23	3.94
one-two-three-v0_97	5	20	11	417	234	322	234	22.78	0.00
4gt11_83	5	8	7	49	23	38	25	22.45	-8.70
4gt5_76	5	16	14	135	64	107	76	20.74	-18.75
sf_275	5	17	8	267	160	212	142	20.60	11.25
rd32_273	5	44	22	574	336	504	365	12.20	-8.63
alu-v4_36	5	10	6	131	73	116	83	11.45	-13.70
4gt11_82	5	12	11	50	30	50	30	0.00	0.00

Source: Prepared by the author.

The time to solve the ILP formulation was less than a second, thus, is not presented in the Table 19. The tests showed that the Toffoli mapping found smaller circuits compared to the Qiskit algorithm. These results were expected, since the Qiskit was used in a Clifford+T circuit instead of the direct mapping from the Toffoli circuit. Circuits with up 69% fewer gates were found and 56% fewer levels, such as the *alu-v3_34*. In the circuit *4gt11_82* both approaches found a mapping with the same number of gates and levels. Some circuits the Qiskit found circuits with fewer levels, for example, *4gt5_76* with 18% difference. This is due the characteristic of the CNOT mappings, which was defined in terms of number of gates instead of levels. Also, the Qiskit algorithm has the characteristic of permuting the circuit as needed to map the CNOT gates to the target architecture, reducing the number of levels. While, in the Toffoli mapping the qubits are moved back

to the original position, increasing the number of levels. Even with the *increase* number of levels in the CNOT mappings, the Toffoli approach found smaller circuits regarding the number of gates and levels.

The next set of tests was to compare the Toffoli mapping with the Standalone algorithm optimized with Qiskit. Table 20 shows the characteristics of the circuit resulted from these tests. The structure of the Table is the same of the previous one. The columns below **Standalone** presents the results of the mapped circuits using the Standalone optimized algorithm.

Table 20 – Mapping Toffoli circuits compared with Standalone optimized in IBM QX4.

	Original			Standalone		Toffoli Mapping		%Improvements	
Circuits	**Qubits**	**Gates**	**Levels**	**Gates**	**Levels**	**Gates**	**Levels**	**Gates**	**Levels**
one-two-three-v2_100	5	8	7	127	90	60	43	52.76	52.22
alu-v0_26	5	9	6	147	97	76	52	48.30	46.39
alu-v3_35	5	7	6	69	48	37	27	46.38	43.75
alu-v4_37	5	7	6	69	48	37	27	46.38	43.75
alu-v2_33	5	7	3	65	44	36	28	44.62	36.36
alu-v1_29	5	7	6	61	42	35	26	42.62	38.10
alu-v1_28	5	7	6	62	43	37	26	40.32	39.53
4gt5_77	5	10	6	204	144	127	94	37.75	34.72
alu-v3_34	5	7	5	69	42	43	32	37.68	23.81
4mod7-v0_95	5	12	7	251	167	164	121	34.66	27.54
4mod7-v1_96	5	13	8	251	167	164	121	34.66	27.54
mod5d2_70	5	8	6	64	42	43	31	32.81	26.19
rd32_270	5	9	4	122	74	82	50	32.79	32.43
rd32_271	5	9	4	122	74	82	50	32.79	32.43
one-two-three-v0_97	5	20	11	473	331	322	234	31.92	29.31
one-two-three-v3_101	5	8	7	116	77	79	56	31.90	27.27
4gt11_83	5	8	7	55	36	38	25	30.91	30.56
4gt5_75	5	8	4	123	88	86	52	30.08	40.91
4mod5-v1_23	5	8	8	97	57	68	53	29.90	7.02
4mod5-v0_18	5	9	5	87	50	61	48	29.89	4.00
mod5d2_64	5	8	6	80	52	57	40	28.75	23.08
mod5d1_63	5	7	5	36	25	26	18	27.78	28.00
one-two-three-v0_98	5	11	6	218	142	163	122	25.23	14.08
4mod7-v0_94	5	12	8	203	128	153	113	24.63	11.72
4gt10-v1_81	5	12	7	215	142	163	117	24.19	17.61
alu-v2_31	5	31	19	620	421	480	354	22.58	15.91
4gt11_82	5	12	11	64	42	50	30	21.88	28.57
hwb5_55	5	39	23	552	347	432	308	21.74	11.24
one-two-three-v1_99	5	11	7	171	113	138	100	19.30	11.50
4gt13-v1_93	5	7	5	67	40	55	39	17.91	2.50
4gt13_91	5	13	9	139	87	116	77	16.55	11.49
alu-v4_36	5	10	6	138	92	116	83	15.94	9.78
alu-v2_32	5	13	8	204	132	176	128	13.73	3.03
4gt13_90	5	17	13	148	93	128	85	13.51	8.60
4gt5_76	5	16	14	120	69	107	76	10.83	-10.14
sf_275	5	17	8	207	132	212	142	-2.42	-7.58
rd32_273	5	44	22	450	287	504	365	-12.00	-27.18

Source: Prepared by the author.

The Standalone algorithm was used in Clifford+T without the direct mapping from Toffoli circuits. However, in some cases, found smaller circuits than the developed approach. For example, the circuit *rd32_273*, found a circuit with 12% fewer gates. This is due the transformation used was the "best" one to map the Toffoli circuit to Clifford+T,

i.e., resulted in fewer misplaced CNOT gates. Another characteristic of the Standalone algorithm that may reduced the number of gates is the look-ahead scheme combined with the change in the permutation circuit. Toffoli mapping found smaller circuits with up to 52% smaller gates and levels, keeping the permutation unchanged. In the same circuit of Qiskit evaluation, *4gt5_76*, the Standalone algorithm found a mapping with 10% fewer levels.

The same tests were repeated for the IBM QX5. Table 21 shows the results from these tests. In this case, the column **T(sec)**, was added to present the runtime (in seconds) of the algorithms to map the circuits. The gap of the circuits with the solution time exceeded are shown in Appendix C.

Table 21 – Mapping Toffoli circuits compared with Qiskit in IBM QX5.

	Original			Qiskit			Toffoli Mappings			%Improvements	
Circuits	**Qubits**	**Gates**	**Levels**	**Gates**	**Levels**	**T(sec)**	**Gates**	**Levels**	**T(sec)**	**Gates**	**Levels**
mini_alu_305	10	20	6	1,020	361	8	306	155	687	70.00	57.06
0410184_169	14	46	16	762	173	8	285	105	25	62.60	39.31
rd73_140	10	20	12	1,019	200	9	409	251	781	59.86	-25.50
cm82a_208	16	50	18	2,030	607	22	890	478	149	56.16	21.25
rd53_137	16	26	15	817	242	9	372	134	59	54.47	44.63
sys6-v0_111	10	20	10	786	172	8	364	208	441	53.69	-20.93
f2_232	16	81	37	3,524	1,271	38	1,731	833	109	50.88	34.46
ham7_105	16	27	15	787	260	9	398	252	335	49.43	3.08
mod5adder_127	16	41	15	1,480	612	17	750	425	848	49.32	30.56
ex3_229	16	24	10	923	360	10	472	304	8	48.86	15.56
rd53_133	16	40	19	1,544	569	16	802	381	149	48.06	33.04
rd53_135	16	26	16	1,025	375	12	542	311	270	47.12	17.07
rd84_142	15	28	14	1,319	263	12	701	335	3,600	46.85	-27.38
C17_204	16	32	14	1,330	492	15	708	395	696	46.77	19.72
wim_266	16	71	28	3,165	699	33	1,699	808	1,401	46.32	-15.59
mod5adder_129	16	23	14	742	345	10	410	226	73	44.74	34.49
pm1_249	16	126	54	5,304	1,711	55	2,971	1,617	3,600	43.99	5.49
cm152a_212	16	81	41	3,203	1,194	36	1,885	788	368	41.15	34.00
ham7_104	16	35	23	925	349	10	549	330	237	40.65	5.44
rd53_131	16	49	19	1,208	481	14	717	354	367	40.65	26.40
dc1_220	16	129	54	5,865	1,391	59	3,500	1,758	3,600	40.32	-26.38
rd53_311	13	34	8	954	279	10	573	238	3,600	39.94	14.70
cnt3-5_179	16	25	5	441	73	5	268	85	316	39.23	-16.44
mod5adder_128	16	27	15	927	431	11	564	385	120	39.16	10.67
rd53_130	16	81	30	3,006	1,058	33	1,884	1,085	3,600	37.33	-2.55
sym9_146	12	28	14	1,067	328	10	670	295	3,600	37.21	10.06
decod24-enable_126	16	23	11	801	325	10	566	306	164	29.34	5.85
hwb6_58	16	54	35	1,368	567	15	1,023	536	3,600	25.22	5.47
ham15_108	16	162	84	5,755	1,696	59	4,732	1,930	3,600	17.78	-13.80
ham15_109	16	128	67	2,780	750	24	2,397	855	3,600	13.78	-14.00
sym9_148	16	1,344	672	58,039	21,324	630	54,814	21,866	3,600	5.56	-2.54

Source: Prepared by the author.

The Qiskit time was the time to map the Clifford+T circuit, while the Toffoli Mapping was the time for the ILP to be solved. The runtime of the ILP solver was set to 3,600 seconds. If the problem was not solved at this time, the incumbent solution is used to find the permutation.

For the all tests the Toffoli mapping found smaller circuits compared with Qiskit. Qiskit finds the mapping guided by a heuristic without taking into account the all circuit, it means that the algorithm maps the CNOT gates as needed in the circuit. It may lead

to a mapping with higher number of gates. For example, the circuit *mini_alu_305* with 10 qubits, the Toffoli mapping found a circuit with 70% fewer gates and 57% fewer levels. Another example is the circuit *rd53_137* with 16 qubits, the developed approach found a circuit 54% smaller regarding the number of gates and 44% in the number of levels. However, for some circuits the number of levels was worse, e.g., the circuit *rd73_140* and *rd84_142* with 25% and 27% fewer levels, respectively. These results show that the goal of the proposed mappings was the size of the circuit.

In the cases in which the ILP problem could not find the exact solution due the runtime constraints, the incumbent solution was used. Even using the incumbent solution, the Toffoli mapping strategy found better circuits than Qiskit, e.g., the circuit *ham15_108*. In these cases, the lowest cost permutation is not guaranteed, however, was enough to find better mappings than the ones found with Qiskit. These cases are expected to improve the size of the circuit if the exact solution was found.

In order to improve the runtime of the ILP problem a machine with more processing power can be used. Another approach is to apply an algorithm to reduce the search area of the solution, i.e., to reduce the number of variables by removing the variables that will clearly find poor solutions. For example, a CNOT that is repeated several times in the circuit, it does not make sense to map this CNOT to a position with high mapping cost.

The next evaluation was to compare the circuits with the Standalone optimized. Table 22 shows the results. The Table structure is the same used in the previous one. The columns below **Standalone** presents the characteristic of the circuits obtained with Standalone algorithm optimized by Qiskit. The column **T(sec)** is the time, in seconds, that the circuit using the Standalone optimized was obtained and for the Toffoli Mapping was the time it took to solve the ILP formulation. The gap of the circuits with the solution time exceeded are shown in Appendix C.

Note that in the Table 22 all the cases with worse results compared with the Standalone algorithm, the ILP solver did not find the lowest cost permutation. For example, the circuits *sym9_148* and *ham15_109*, the Standalone algorithm found a circuit with 22% and 6% fewer gates, respectively. The characteristic of the Standalone algorithm is to divide the circuit into layers and apply a look-ahead scheme to find the shortest distant between the two layers. This may reduce the number of levels of the circuit by reducing the number of CNOT gates to be mapped along with the Qiskit optimizations. Another possibility is due the characteristics of the circuit, i.e., the number of misplaced CNOT gates in the circuit allocated in qubits with high cost mappings, increasing the

Table 22 – Mapping Toffoli circuits compared with Standalone optimized in IBM QX5.

	Original			Standalone			Toffoli Mappings			%Improvements	
Circuits	**Qubits**	**Gates**	**Levels**	**Gates**	**Levels**	**T(sec)**	**Gates**	**Levels**	**T(sec)**	**Gates**	**Levels**
0410184_169	14	46	16	559	107	7	285	105	25	49.02	1.87
C17_204	16	32	14	1,091	446	8	708	395	696	35.11	11.43
cm82a_208	16	50	18	1,357	428	8	890	478	149	34.41	-11.68
rd53_133	16	40	19	1,146	448	8	802	381	149	30.02	14.96
rd53_137	16	26	15	523	234	4	372	134	59	28.87	42.74
sys6-v0_111	10	20	10	510	115	3	364	208	441	28.63	-80.87
ham7_105	16	27	15	553	190	4	398	252	335	28.03	-32.63
f2_232	16	81	37	2,392	890	14	1,731	833	109	27.63	6.40
mod5adder_127	16	41	15	1,005	395	7	750	425	848	25.37	-7.59
rd53_131	16	49	19	960	337	6	717	354	367	25.31	-5.04
mod5adder_129	16	23	14	525	242	4	410	226	73	21.90	6.61
cm152a_212	16	81	41	2,411	1,110	18	1,885	788	368	21.82	29.01
ham7_104	16	35	23	688	296	5	549	330	237	20.20	-11.49
wim_266	16	71	28	2,114	797	12	1,699	808	1,401	19.63	-1.38
mini_alu_305	10	20	6	367	136	3	306	155	687	16.62	-13.97
rd53_135	16	26	16	649	200	4	542	311	270	16.49	-55.50
ex3_229	16	24	10	563	215	5	472	304	8	16.16	-41.40
pm1_249	16	126	54	3,537	1,449	21	2,971	1,617	3,600	16.00	-11.59
dc1_220	16	129	54	4,130	1,692	25	3,500	1,758	3,600	15.25	-3.90
cnt3-5_179	16	25	5	314	49	3	268	85	316	14.65	-73.47
mod5adder_128	16	27	15	652	304	6	564	385	120	13.50	-26.64
rd84_142	15	28	14	805	193	5	701	335	3,600	12.92	-73.58
rd73_140	10	20	12	465	128	4	409	251	781	12.04	-96.09
hwb6_58	16	54	35	1,112	539	8	1,023	536	3,600	8.00	0.56
rd53_130	16	81	30	2,031	646	13	1,884	1,085	3,600	7.24	-67.96
decod24-enable_126	16	23	11	608	235	5	566	306	164	6.91	-30.21
sym9_146	12	28	14	707	191	5	670	295	3,600	5.23	-54.45
rd53_311	13	34	8	586	207	4	573	238	3,600	2.22	-14.98
ham15_108	16	162	84	4,555	1,643	27	4,732	1,930	3,600	-3.89	-17.47
ham15_109	16	128	67	2,259	737	12	2,397	855	3,600	-6.11	-16.01
sym9_148	16	1,344	672	44,813	16,004	262	54,814	21,866	3,600	-22.32	-36.63

Source: Prepared by the author.

size of the circuit in the Toffoli mapping.

The cases with the ILP solution, the Toffoli mapping found smaller circuits for all the cases. For the circuit *C17_204* the difference was of 35% fewer gates and 11% fewer levels. In some cases, even with the incumbent solution the Toffoli mapping found smaller circuit, e.g., *dc1_220* with 15% fewer gates. However, the number of levels was worse, with 4% more levels. It can be noted that in the number of levels, the proposed Toffoli mapping found worse results in some circuits with almost doubling it, e.g., the circuit *rd73_140*. Another example is the circuit *rd53_130* with 68% more levels.

One possible improvement in the Toffoli mapping is to check if it is "cheaper" to order the qubits after each Toffoli than mapping the misplaced CNOT gates of the Toffoli mapping. This can reduce the number of gates in cases that the misplaced CNOT are in high cost mapping position.

Results show that targeting only the size of the circuit may result in a poor mapping regarding the number of levels. It would be interesting to change the targeting of the CNOT mappings by choosing the ones with smaller additional levels and compare with the Standalone algorithm. Another possible approach is to find the CNOT mappings

taking into account both size and depth of the circuit.

In the ILP formulation Toffoli circuits the number of variables is higher than a CNOT circuit, thus, its complexity. Even with the higher complexity, several circuits with 16 qubits could found the optimal solution for the ILP problem. It would be interesting to test the ILP formulation in architectures with more qubits to verify how scale this approach will be, even being an exact one.

Results obtained with the tests have shown that the NCV and Toffoli mappings are good approaches to map reversible and NCV circuits to quantum architectures. The time that the ILP took to found the permutations to apply the mappings is not expensive, since it will be done once for the circuit and the quantum technology still does not need a fast development of circuits. Both approaches still have plenty of room for improvements. For the NCV mappings, a good approach will be to explore the optimization of the NCV circuits and different mapping of V and $V^\dagger$ gates to Clifford+T circuits. Regarding the Toffoli mapping, it will be interesting to find different Toffoli circuits to increase the number of different CNOT positions. However, the best approach is to focus in the synthesis of Clifford+T circuit for the Toffoli circuits due the increase complexity of the mapping as the number of controls in the Toffoli gate increases.

The next chapter concludes the presented work and explore the possibilities of the continuation in this research area.

6 CONCLUSIONS AND FUTURE WORK

This work developed a methodology to map quantum and reversible circuits to IBM quantum processors. The methodology consists in a series of qubits movements used to realize the CNOT gates complying with the IBM architecture restrictions. Along with movements a good permutation must be found in order to reduce the number of misplaced CNOT gates. To solve this, the problem was formulated as an Integer Linear Programming problem. The methodology was compared with two other approaches with the same goal, Qiskit (ALEKSANDROWICZ; ALEXANDER; BARKOUTSOS,) and the approach of Zulehner, Paler and Wille (2019) (referred as Standalone algorithm). The cost metric used in the evaluation was the number of gates and levels, with the number of gates as priority.

The evaluation was done using the IBM QX4 and IBM QX5 architectures with 5 and 16 qubits, respectively. A series of benchmarks were used to compare the developed methodology with the other two approaches. Results have shown that the developed CNOT mappings applied to a low cost permutation can find circuits with fewer number of gates and levels. However, since the developed mappings prioritize the number of gates, the circuits had higher number of levels. Also, most of the circuits with worse results were the ones with no ILP optimal solution. It may change if the optimal solution was found.

The approaches are different, i.e., Qiskit and Standalone swaps the qubits trough the circuit changing its permutation while the developed CNOT mappings keep the permutation unchanged by swapping the qubits back to its original position. Results have shown that this may lead to circuits with higher number of gates when applied to a circuit with many misplaced CNOT gates in high cost positions in the architectures. One possible solution to this problem is to keep the permutation changed in the middle of the circuit and at end of the circuit return it back to the beginning permutation.

The Integer Linear Programming developed in this work has shown that is able to find the lowest cost permutation for the respective objective function when the problem is solved. Due the characteristic of the ILP formulation, it was possible to scale the formulation for architectures with 16 qubits. Furthermore, it is possible to scale even more depending on the characteristics of the circuit and the target architecture. However,

its complexity increases as the number of different CNOT gates and qubits are added to the circuit. The incumbent solution and the reduction of the search area are possible approaches to scale the ILP application. The ILP has the characteristic of being formulated regarding the gates used in the circuit. Therefore, it can be used with any reversible circuit, also, with any type of gate. One example of its adaptation is the mapping of the Toffoli circuits.

The mapping of NCV and reversible circuits showed a possible adaptation of the developed methodology to map circuits to quantum architectures. Since the approaches used to evaluate the methodology do not realize the direct mappings of NCV and Toffoli circuits, the circuits were mapped to Clifford+T. This was not the ideal evaluation, yet, it showed that the direct mapping can result in a smaller circuit. The evaluation also showed the impact in keeping the permutation unchanged, in some cases, by resulting in circuits with higher number of gates.

Overall, the methodology presented in this work showed that it is possible to find good circuits targeting quantum architectures while keeping the permutation of the circuit unchanged. The ILP formulation to solve the permutation problem can be applied to any reversible circuits with any type of gate due to its characteristic in describing the constraints in terms of the gates used in the circuit. The formulation can be scaled by exploring the characteristics of the problem that has to be solved, reducing the search area of the solution and/or using the incumbent solution at the given time.

6.1 FUTURE WORK

The methodology developed in this work can be used in different problems. As it can be adapted to any quantum architecture, it would be interesting to use it with different quantum topologies with different constraints in order to evaluate the developed CNOT mappings.

A possible improvement of this work is to swap the qubits trough the circuit, changing its permutation to satisfy as many constraints as possible in order to reduce the high number of additional gates.

Another improvement is to evaluate the possibility of transform the circuit in a sequence of sub-circuits and apply the mapping strategy in each sub-circuit separately. This may reduce the size of the circuit due the prevention of misplaced CNOT gates distant of each other.

Since the methodology can easily be changed, it is interesting to change the priority in the defined CNOT mappings to the number of levels, then evaluate the circuits to see the impact in the mapped circuit. Also, the cost can be changed to a weight sum regarding the number of gates and levels.

Another change will be in the cost of the CNOT gates regarding the decoherence of the quantum processors. Different qubits may have different errors values, hence, it is interesting to use the lowest error rate to increase the circuit's fidelity.

The Integer Linear Programming can be used as baseline for the development of heuristics and metaheuristics targeting the permutation problem. The complexity of the ILP problem rapidly increases with the number of different CNOT gates and the number of qubits in the quantum architecture. Therefore, it would be interesting to check how long it will take for the ILP solver to outperforms the metaheuristic solution.

An improvement of the ILP formulation is to use the characteristics of the problem to reduce its search area. For example, CNOT gates that are used many times in the circuit must be mapped to a low cost position with low cost adjacent gates. This may reduce the complexity of the problem, increasing the scalability.

REFERENCES

ABDESSAIED, N.; AMY, M.; SOEKEN, M.; DRECHSLER, R. Technology mapping of reversible circuits to Clifford+T quantum circuits. In: IEEE SYMPOSIUM ON MULTIPLE-VALUED LOGIC (ISMVL), 46., 2016, Sapporo. **Anais**[...]. Sapporo, 2016. p. 150–155.

AL-RABADI, A. N. **Reversible Logic Synthesis**: From fundamentals to quantum computing. London: Springer, 2004. 427 p. ISBN 978-3-540-00935-1.

ALEKSANDROWICZ, G.; ALEXANDER, T.; BARKOUTSOS, P. **Qiskit: An Open-source Framework for Quantum Computing**. Disponível em: <https://github.com/Qiskit>. Acesso em: 11 mar. 2019.

AMY, M.; MASLOV, D.; MOSCA, M.; ROETTELER, M. A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, New York, v. 32, n. 6, p. 818–830, June 2013. ISSN 0278-0070.

ASH-SAKI, A.; ALAM, M.; GHOSH, S. QURE: Qubit re-allocation in noisy intermediate-scale quantum computers. In: ANNUAL DESIGN AUTOMATION CONFERENCE 2019, Las Vegas, NV, USA. **Anais**[...]. Las Vegas, NV, USA, 2019. (DAC '19), p. 141:1–141:6. ISBN 978-1-4503-6725-7.

BARENCO, A.; BENNETT, C. H.; CLEVE, R.; DIVINCENZO, D. P.; MARGOLUS, N.; SHOR, P.; SLEATOR, T.; SMOLIN, J. A.; WEINFURTER, H. Elementary gates for quantum computation. **Phys. Rev. A**, College Park, v. 52, p. 3457–3467, Nov 1995.

BAZARAA, M. S.; JARVIS, J. J.; SHERALI, H. D. **Linear Programming and Network Flows**. 2. ed. New York: John Wiley & Sons, Inc., 1990. 574 p. ISBN 0-471-63681-9.

BHATTACHARJEE, A.; BANDYOPADHYAY, C.; WILLE, R.; DRECHSLER, R.; RAHAMAN, H. A novel approach for nearest neighbor realization of 2D quantum circuits. In: IEEE COMPUTER SOCIETY ANNUAL SYMPOSIUM ON VLSI (ISVLSI), 8., 2018, Hong Kong. **Anais**[...]. Hong Kong, 2018. p. 305–310.

BHATTACHARJEE, D.; CHATTOPADHYAY, A. Depth-optimal quantum circuit placement for arbitrary topologies. **ArXiv**, New York, v. 1703.08540, p. 1–17, 2017.

BOTEA, A.; KISHIMOTO, A.; MARINESCU, R. On the complexity of quantum circuit compilation. In: THE INTERNATIONAL SYMPOSIUM ON COMBINATORIAL SEARCH, 11., 2018, Stockholm. **Anais**[...]. Stockholm, 2018. p. 150–155.

COURTLAND, R. Google aims for quantum computing supremacy [news]. **IEEE Spectrum**, New York, v. 54, n. 6, p. 9–10, 2017. ISSN 0018-9235.

CROSS, A. W.; BISHOP, L. S.; SMOLIN, J. A.; GAMBETTA, J. M. Open quantum assembly language. **ArXiv**, New York, v. 1707.03429, p. 1–24, 2017.

DAS, R.; CHATTOPADHYAY, A.; RAHAMAN, H. Optimizing quantum circuits for modular exponentiation. In: INTERNATIONAL CONFERENCE ON VLSI DESIGN AND INTERNATIONAL CONFERENCE ON EMBEDDED SYSTEMS (VLSID), 32., 2019, Delhi. **Anais**[...]. Delhi, 2019. p. 407–412.

DING, Y.; HOLMES, A.; JAVADI-ABHARI, A.; FRANKLIN, D.; MARTONOSI, M.; CHONG, F. Magic-state functional units: Mapping and scheduling multi-level distillation circuits for fault-tolerant quantum architectures. In: ANNUAL IEEE/ACM INTERNATIONAL SYMPOSIUM ON MICROARCHITECTURE (MICRO), 51., 2018, Fukuoka. **Anais**[...]. Fukuoka, 2018. p. 828–840.

DUECK, G. W.; MASLOV, D.; MILLER, D. M. Transformation-based synthesis of networks of Toffoli/Fredkin gates. In: CANADIAN CONFERENCE ON ELECTRICAL AND COMPUTER ENGINEERING. TOWARD A CARING AND HUMANE TECHNOLOGY, 2003, Montreal. **Anais**[...]. New York, 2003. p. 211–214.

FU, X.; RIESEBOS, L.; ROL, M. A.; VAN STRATEN, J.; VAN SOMEREN, J.; KHAMMASSI, N.; ASHRAF, I.; VERMEULEN, R. F. L.; NEWSUM, V.; LOH, K. K. L.; STERKE, J. C. de; VLOTHUIZEN, W. J.; SCHOUTEN, R. N.; ALMUDEVER, C. G.; DICARLO, L.; BERTELS, K. eQASM: An executable quantum instruction set architecture. In: IEEE INTERNATIONAL SYMPOSIUM ON HIGH PERFORMANCE COMPUTER ARCHITECTURE (HPCA), 25., 2019, Washington. **Anais**[...]. Washington, 2019. p. 224–237. ISSN 2378-203X.

FU, X.; ROL, M. A.; BULTINK, C. C.; VAN SOMEREN, J.; KHAMMASSI, N.; ASHRAF, I.; VERMEULEN, R. F. L.; STERKE, J. C. de; VLOTHUIZEN, W. J.; SCHOUTEN, R. N.; ALMUDEVER, C. G.; DICARLO, L.; BERTELS, K. A microarchitecture for a superconducting quantum processor. **IEEE Micro**, New York, v. 38, n. 3, p. 40–47, May 2018. ISSN 0272-1732.

GOMES, L. Quantum computing: Both here and not here. **IEEE Spectrum**, New York, v. 55, n. 4, p. 42–47, April 2018. ISSN 0018-9235.

GUROBI. **Gurobi Optimization**. Disponível em: <http://www.gurobi.com>. Acesso em: 01 mar. 2019.

IBM. **Backend information**. Disponível em: <https://github.com/QISKit/ibmqx-backend-information>. Acesso em: 11 mar. 2019.

IBM. **IBM ILOG CPLEX Optimization Studio**. Disponível em: <https://www.ibm.com/products/ilog-cplex-optimization-studio>. Acesso em: 01 mar. 2019.

IBM. **IBM Q 16 Rueschlikon V1.x.x**. Disponível em: <https://github.com/Qiskit/ibmq-device-information/tree/master/backends/rueschlikon/V1>. Acesso em: 11 mar. 2019.

IBM. **IBM Q 5 Tenerife V1.x.x**. Disponível em: <https://github.com/QISKit/qiskit-backend-information/blob/master/backends/tenerife/V1>. Acesso em: 01 mar. 2019.

IBM. **Project IBM Q**. Disponível em: <https://www.research.ibm.com/ibm-q/>. Acesso em: 11 mar. 2019.

JAVADI-ABHARI, A.; GOKHALE, P.; HOLMES, A.; FRANKLIN, D.; BROWN, K. R.; MARTONOSI, M.; CHONG, F. T. Optimized surface code communication in superconducting quantum computers. In: IEEE/ACM INTERNATIONAL SYMPOSIUM ON MICROARCHITECTURE, 50., 2017, New York. **Anais**[...]. New York, 2017. (MICRO-50 '17), p. 692–705. ISBN 978-1-4503-4952-9.

KHAMMASSI, N.; ASHRAF, I.; FU, X.; ALMUDEVER, C. G.; BERTELS, K. QX: A high-performance quantum computer simulation platform. In: DESIGN, AUTOMATION TEST IN EUROPE CONFERENCE EXHIBITION (DATE), 2017, 9., 2017, Lausanne. **Anais**[...]. Lausanne, 2017. p. 464–469. ISSN 1558-1101.

KOLE, A.; DATTA, K.; SENGUPTA, I. A new heuristic for n-dimensional nearest neighbor realization of a quantum circuit. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, New York, v. 37, n. 1, p. 182–192, 2018. ISSN 0278-0070.

KUENDEE, P.; JANJARASSUK, U. A comparative study of mixed-integer linear programming and genetic algorithms for solving binary problems. In: INTERNATIONAL CONFERENCE ON INDUSTRIAL ENGINEERING AND APPLICATIONS (ICIEA), 5., 2018, Singapore. **Anais**[...]. Singapore, 2018. p. 284–288.

LANDAUER, R. Irreversibility and heat generation in the computing process. **IBM J. Res.**, New York, v. 5, n. 3, p. 183–191, 1961.

LI, G.; DING, Y.; XIE, Y. Tackling the qubit mapping problem for NISQ-Era quantum devices. In: INTERNATIONAL CONFERENCE ON ARCHITECTURAL SUPPORT FOR PROGRAMMING LANGUAGES AND OPERATING SYSTEMS, 24., 2019, Providence. **Anais**[...]. Providence, RI, USA, 2019. (ASPLOS '19), p. 1001–1014. ISBN 978-1-4503-6240-5.

LIN, Y.; YU, B.; LI, M.; PAN, D. Z. Layout synthesis for topological quantum circuits with 1-D and 2-D architectures. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, New York, v. 37, n. 8, p. 1574–1587, Aug 2018. ISSN 0278-0070.

LIU, P.; HU, S.; PISTOIA, M.; CHEN, C. R.; GAMBETTA, J. M. Stochastic optimization of quantum programs. **Computer**, New York, v. 52, n. 6, p. 58–67, June 2019. ISSN 0018-9162.

LP_SOLVE. **LP_Solve - Mixed Integer Linear Programming (MILP) solver**. Disponível em: <http://lpsolve.sourceforge.net/5.5/>. Acesso em: 01 mar. 2019.

MASLOV, D.; DUECK, G. W. Reversible cascades with minimal garbage. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, New York, v. 23, n. 11, p. 1497–1509, 2004. ISSN 0278-0070.

MATSUO, A.; HATTORI, W.; YAMASHITA, S. Reducing the overhead of mapping quantum circuits to ibm q system. In: IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS (ISCAS), 51., 2019, Sapporo. **Anais**[...]. Sapporo, 2019. p. 1–5. ISSN 2158-1525.

MILLER, D. M.; SOEKEN, M.; DRECHSLER, R. Mapping NCV circuits to optimized Clifford+T circuits. In: INTERNATIONAL CONFERENCE ON REVERSIBLE COMPUTATION, 6., 2014, Kyoto. **Anais**[...]. Springer: Springer, 2014. p. 163–175.

MORELLO, A.; TOSI, G.; MOHIYADDIN, F. A.; SCHMITT, V.; MOURIK, V.; BOTZEM, T.; LAUCHT, A.; PLA, J. J.; TENBERG, S.; SAVYTSKYY, R.; MADZIK, M.; HUDSON, F.; DZURAK, A. S.; ITOH, K. M.; JAKOB, A. M.; JOHNSON, B. C.; MCCALLUM, J. C.; JAMIESON, D. N. Scalable quantum computing with ion-implanted dopant atoms in silicon. In: IEEE INTERNATIONAL ELECTRON DEVICES MEETING (IEDM), 5., 2018, San Francisco. **Anais**[...]. San Francisco, 2018. p. 6.2.1–6.2.4.

NAM, Y.; ROSS, N. J.; SU, Y.; CHILDS, A. M.; MASLOV, D. Automated optimization of large quantum circuits with continuous parameters. **NPJ Quantum Information**, Cologne, v. 4, n. 1, p. 23, 2018. ISSN 2056-6387.

NAVEH, Y.; KASHEFI, E.; WOOTTON, J. R.; BERTELS, K. Theoretical and practical aspects of verification of quantum computers. In: DESIGN, AUTOMATION TEST IN EUROPE CONFERENCE EXHIBITION (DATE), 21., 2018, Dresden. **Anais**[...]. Dresden, 2018. p. 721–730. ISSN 1558-1101.

NIELSEN, M.; NIELSEN, M.; CHUANG, I. **Quantum Computation and Quantum Information**. 1. ed. New York, NY, USA: Cambridge University Press, 2000. (Cambridge Series on Information and the Natural Sciences). ISBN 9780521635035.

O'CONNELL, R. F. Decoherence in quantum systems. **IEEE Transactions on Nanotechnology**, New York, v. 4, n. 1, p. 77–82, Jan 2005. ISSN 1536-125X.

PALER, A. Controlling distilleries in fault-tolerant quantum circuits: problem statement and analysis towards a solution. In: IEEE/ACM INTERNATIONAL SYMPOSIUM ON NANOSCALE ARCHITECTURES (NANOARCH), 14., 2018, Athens. **Anais**[...]. Athens, 2018. p. 1–6. ISSN 2327-8226.

PROUSALIS, K.; KONOFAOS, N. QuCirDET: A design and simulation tool for quantum circuits. In: INTERNATIONAL CONFERENCE ON MODERN CIRCUITS AND SYSTEMS TECHNOLOGIES (MOCAST), 5., 2016, Thessaloniki. **Anais**[...]. Thessaloniki, 2016. p. 1–4.

SARGARAN, S.; MOHAMMADZADEH, N. SAQIP: A scalable architecture for quantum information processors. **ACM Transactions on Architecture and Code Optimization (TACO)**, New York, v. 16, n. 2, p. 12:1–12:21, abr. 2019. ISSN 1544-3566.

SCHAEFFER, B. Product transformation and heuristic EXOR-AND-OR logic synthesis of incompletely specified functions. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, New York, v. 36, n. 11, p. 1831–1841, 2017. ISSN 0278-0070.

SHAFAEI, A.; SAEEDI, M.; PEDRAM, M. Qubit placement to minimize communication overhead in 2D quantum architectures. In: ASIA AND SOUTH PACIFIC DESIGN AUTOMATION CONFERENCE (ASP-DAC), 19., 2014, Singapore. **Anais**[...]. Singapore, 2014. p. 495–500. ISSN 2153-6961.

SHOR, P. W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. **SIAM Journal on Computing (SICOMP)**, Philadelphia, v. 26, n. 5, p. 1484–1509, out. 1997. ISSN 0097-5397.

SIMON, F. Y.-P.; TAKEFUJI. Integer linear programming neural networks for job-shop scheduling. In: INTERNATIONAL CONFERENCE ON NEURAL NETWORKS, 1., 1988, San Diego. **Anais**[...]. San Diego: IEEE, 1988. v. 2, p. 341–348.

SIRAICHI, M. Y.; SANTOS, V. F. d.; COLLANGE, S.; PEREIRA, F. M. Q. Qubit allocation. In: INTERNATIONAL SYMPOSIUM ON CODE GENERATION AND OPTIMIZATION, 16., 2018, Vienna. **Anais**[...]. Vienna, 2018. (CGO 2018), p. 113–125. ISBN 978-1-4503-5617-6.

SOEKEN, M.; DUECK, G. W.; RAHMAN, M. M.; MILLER, D. M. An extension of transformation-based reversible and quantum circuit synthesis. In: IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS (ISCAS), 48., 2016, Montreal. **Anais**[...]. Montreal, 2016. p. 2290–2293.

SOEKEN, M.; FREHSE, S.; WILLE, R.; DRECHSLER, R. RevKit: a toolkit for reversible circuit design. In: INTERNATIONAL SYMPOSIUM ON MULTIPLE-VALUED LOGIC (ISMVL), 38., 2008, Dallas. **Anais**[...]. Dallas, 2008. p. 1–6.

SOEKEN, M.; MILLER, D. M.; DRECHSLER, R. Quantum circuits employing roots of the Pauli matrices. **Phys. Rev. A**, College Park, v. 88, p. 042322, Oct 2013.

SOEKEN, M.; MOZAFARI, F.; SCHMITT, B.; MICHELI, G. D. Compiling permutations for superconducting QPUs. In: DESIGN, AUTOMATION TEST IN EUROPE CONFERENCE EXHIBITION (DATE), 22., 2019, Florence. **Anais**[...]. Florence, 2019. p. 1349–1354. ISSN 1558-1101.

SOEKEN, M.; THOMSEN, M. K. White dots *do* matter: Rewriting reversible logic circuits. In: INTERNATIONAL CONFERENCE ON REVERSIBLE COMPUTATION, 5., 2013, Victoria, BC, Canada. **Anais**[...]. Victoria: Springer, 2013. p. 196–208.

SVORE, K.; GELLER, A.; TROYER, M.; AZARIAH, J.; GRANADE, C.; HEIM, B.; KLIUCHNIKOV, V.; MYKHAILOVA, M.; PAZ, A.; ROETTELER, M. Q#: Enabling scalable quantum computing and development with a High-level DSL. In: REAL WORLD DOMAIN SPECIFIC LANGUAGES WORKSHOP 2018, 4., 2018, Vienna. **Anais**[...]. ACM, 2018. (RWDSL2018), p. 7:1–7:10. ISBN 978-1-4503-6355-6.

TANNU, S. S.; QURESHI, M. K. Not all qubits are created equal: A case for variability-aware policies for NISQ-Era quantum computers. In: ARCHITECTURAL SUPPORT FOR PROGRAMMING LANGUAGES AND OPERATING SYSTEMS, 24., 2019, Providence. **Anais**[...]. New York, NY, USA: ACM, 2019. (ASPLOS '19), p. 987–999. ISBN 978-1-4503-6240-5.

TOCCI, R. J.; WIDMER, N. **Digital Systems: Principles and Applications**. 7th. ed. Nova Jersey: Prentice Hall Professional Technical Reference, 1997. ISBN 0137005105.

VAN DIJK, J.; VLADIMIRESCU, A.; BABAIE, M.; CHARBON, E.; SEBASTIANO, F. A co-design methodology for scalable quantum processors and their classical electronic interface. In: DESIGN, AUTOMATION TEST IN EUROPE CONFERENCE

EXHIBITION (DATE), 21., 2018, Dresden. **Anais**[...]. Dresden, 2018. p. 573–576. ISSN 1558-1101.

VINOKUROV, S. F.; FRANTSEVA, A. S. Complexity of representations of multiple-output boolean functions in the reversible logic circuits. In: IEEE INTERNATIONAL CONFERENCE ON SOFT COMPUTING AND MEASUREMENTS (SCM), 19., 2016, Saint Petersburg. **Anais**[...]. Saint Petersburg, 2016. p. 374–376.

WIE, C.-R. Bloch sphere model for two-qubit pure states. **ArXiv**, New York, v. 1403.8069, p. 1–5, 2014.

WILLE, R.; BURGHOLZER, L.; ZULEHNER, A. Mapping quantum circuits to IBM QX architectures using the minimal number of SWAP and H operations. In: ANNUAL DESIGN AUTOMATION CONFERENCE 2019, 56., 2019, Las Vegas. **Anais**[...]. Las Vegas: ACM, 2019. (DAC '19), p. 142:1–142:6. ISBN 978-1-4503-6725-7.

WILLE, R.; CHATTOPADHYAY, A.; DRECHSLER, R. From reversible logic to quantum circuits: Logic design for an emerging technology. In: INTERNATIONAL CONFERENCE ON EMBEDDED COMPUTER SYSTEMS: ARCHITECTURES, MODELING AND SIMULATION (SAMOS), 16., 2016, Samos Island. **Anais**[...]. Samos Island, 2016. p. 268–274.

WILLE, R.; DRECHSLER, R. **Towards a Design Flow for Reversible Logic**. London: Springer, 2010. 174 p. ISBN 978-90-481-9578-7.

WILLE, R.; FOWLER, A.; NAVEH, Y. Computer-aided design for quantum computation. In: IEEE/ACM INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN (ICCAD), 10., 2018, San Diego. **Anais**[...]. San Diego, 2018. p. 1–6. ISSN 1558-2434.

WILLE, R.; GROSSE, D.; TEUBER, L.; DUECK, G. W.; DRECHSLER, R. RevLib: An online resource for reversible functions and reversible circuits. In: INTERNATIONAL SYMPOSIUM ON MULTIPLE-VALUED LOGIC (ISMVL), 38., 2008, Dallas. **Anais**[...]. New York, 2008. p. 220–225.

WILLE, R.; LYE, A.; DRECHSLER, R. Exact reordering of circuit lines for nearest neighbor quantum architectures. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, New York, v. 33, n. 12, p. 1818–1831, Dec 2014. ISSN 0278-0070.

WILLE, R.; VAN METER, R.; NAVEH, Y. IBM's Qiskit tool chain: Working with and developing for real quantum computers. In: DESIGN, AUTOMATION TEST IN EUROPE CONFERENCE EXHIBITION (DATE), 23., 2019, Grenoble. **Anais**[...]. Grenoble, 2019. p. 1234–1240. ISSN 1558-1101.

WILLIAMS, H. P. **Model Building in Mathematical Programming**. 5. ed. New York: John Wiley & Sons, Inc., 2013. ISBN 978-1-118-44333-0.

YANG, Y. S.; CHEN, H. K.; KUO, S. Y.; ZENG, G. J.; CHOU, Y. H. A novel efficient optimal reversible circuit synthesis algorithm. In: IEEE INTERNATIONAL CONFERENCE ON SYSTEMS, MAN, AND CYBERNETICS, 8., 2015, Hong Kong. **Anais**[...]. Hong Kong, 2015. p. 68–73.

ZULEHNER, A.; NIEMANN, P.; DRECHSLER, R.; WILLE, R. Accuracy and compactness in decision diagrams for quantum computation. In: DESIGN, AUTOMATION TEST IN EUROPE CONFERENCE EXHIBITION (DATE), 23., 2019, Grenoble. **Anais**[...]. Grenoble, 2019. p. 280–283. ISSN 1558-1101.

ZULEHNER, A.; PALER, A.; WILLE, R. Efficient mapping of quantum circuits to the IBM QX architectures. In: DESIGN, AUTOMATION TEST IN EUROPE CONFERENCE EXHIBITION (DATE), 21., 2018, Dresden. **Anais**[...]. Dresden, 2018. p. 1135–1138. ISSN 1558-1101.

ZULEHNER, A.; PALER, A.; WILLE, R. An efficient methodology for mapping quantum circuits to the IBM QX architectures. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, New York, v. 38, n. 7, p. 1226–1236, July 2019. ISSN 0278-0070.

ZULEHNER, A.; WILLE, R. Skipping embedding in the design of reversible circuits. In: INTERNATIONAL SYMPOSIUM ON MULTIPLE-VALUED LOGIC (ISMVL), 47., 2017, Oslo. **Anais**[...]. Oslo, 2017. p. 173–178.

ZULEHNER, A.; WILLE, R. Advanced simulation of quantum computations. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, New York, v. 38, n. 5, p. 848–859, May 2019. ISSN 0278-0070.

ZULEHNER, A.; WILLE, R. Compiling SU(4) quantum circuits to IBM QX architectures. In: ASIA AND SOUTH PACIFIC DESIGN AUTOMATION CONFERENCE, 24., 2019, Tokyo. **Anais**[...]. ACM, 2019. (ASPDAC '19), p. 185–190. ISBN 978-1-4503-6007-4.

APPENDIX A – MATRIX REPRESENTATION OF QUANTUM GATES AND PROOF OF CIRCUITS EQUIVALENCE

Quantum circuits with n qubits can be represented by a $2^n \times 2^n$ matrix. To obtain this matrix, the representation (matrix) of each gate in the circuit must be multiplied in the sequence of the gates appear in the circuit. Therefore, to prove the equivalences between two circuits, the matrices that represent each circuit must be equal.

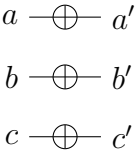
From the 2×2 matrix representation of each gate acting on a single qubit it is possible to find a $2^n \times 2^n$, where n is the number of qubits of the circuit. To do that, the Kronecker product is used.

A.1 FINDING MATRIX REPRESENTATION FOR SINGLE QUBIT GATE

The matrix representation of any gate for any number of qubits can be found starting with the representation of the respective gate acting on a single qubit. The representation of these gates is presented in Table 23. The rows and columns of the matrices represent the input and output of the circuit, respectively.

To explain the generation of the matrix the NOT gate will be used. Assume a quantum circuit with 3 qubits presented in Figure 86. The matrix representation of the NOT gate acting on the three qubits will be shown.

Figure 86 – NOT gate acting on three different qubits.



Source: Prepared by the author.

Table 23 – Matrix representation of quantum gates.

Gate	Matrix	Representation	Gate	Matrix	Representation
NOT	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$		H	$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$	
V	$\begin{pmatrix} \frac{1+i}{2} & \frac{1-i}{2} \\ \frac{1-i}{2} & \frac{1+i}{2} \end{pmatrix}$		$V^\dagger$	$\begin{pmatrix} \frac{1-i}{2} & \frac{1+i}{2} \\ \frac{1+i}{2} & \frac{1-i}{2} \end{pmatrix}$	
T	$\begin{pmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{4}} \end{pmatrix}$		$T^\dagger$	$\begin{pmatrix} 1 & 0 \\ 0 & e^{\frac{-i\pi}{4}} \end{pmatrix}$	
S	$\begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$		$S^\dagger$	$\begin{pmatrix} 1 & 0 \\ 0 & -i \end{pmatrix}$	

Source: Adapted from Abdessaied *et al.* (2016).

The matrix representation will be found using the Identity (2×2) matrix. From now on, the Identity (2×2) matrix will be represented as I. The matrix representation of the NOT acting on qubit a is presented in Equation 30.

$$NOT_a = (\text{NOT} \otimes I) \otimes I = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{30}$$

Note that the NOT $\otimes I$ is the representation of the NOT gate acting on qubit a for a circuit with two qubits. Similarly, for the NOT gate acting on qubit b, the representation is shown in Equation 31.

$$NOT_b = (I \otimes \text{NOT}) \otimes I = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \tag{31}$$

In the same way, Equation 32 shows the NOT acting on the qubit c.

$$NOT_c = (I \otimes I) \otimes \text{NOT} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \tag{32}$$

To find the matrix representation for the circuit from Figure 86 the three matrices NOT_a, NOT_b, and NOT_c must be multiplied. Let M be the matrix representation for this circuit. Equation 33 shows this matrix.

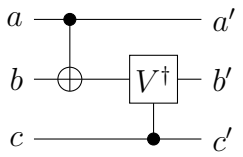
$$M = NOT_a * NOT_b * NOT_c = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{33}$$

The next step is to find the matrix representation for controlled gates. This will be presented in the next section.

A.2 FINDING MATRIX REPRESENTATION FOR CONTROLLED GATE

The strategy to find the matrix representation for controlled gates is the same used in the previous section. However, the matrix must be changed in the rows that the control qubit will not act, i.e., for a positive control, the output will be the identity when the input (on the control qubit) is 0. For a negative control, the output will be the identity when the input (on the control qubit) is 1. The idea is to keep the matrix unchanged in the rows in which the control qubit is acting and change to the identity the remaining rows. The circuit in Figure 87 will be used to exemplify this type of matrix generation.

Figure 87 – Example circuit to find a matrix representation of controlled gates.



Source: Prepared by the author.

The first step to obtain the matrix representation for a controlled gate is to find the matrix for the gate that is being used. For example, in order to find the matrix representation of the CNOT gate, first it is needed to find the matrix of the NOT gate. Since the NOT is acting on qubit b (controlled by qubit a), the matrix is the same from Equation 31.

The next step is to update the NOT_b matrix in the rows that the control is not acting in the circuit. Since the control is on qubit a, it means that the NOT will only act when the value in qubit a is 1. Thus, the rows in the matrix representing the qubit a with value 0 will act as an Identity. The matrix of this CNOT gate is presented in Equation 34.

$$CNOT(a,b) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \tag{34}$$

The same approach is applied for the controlled $V^\dagger$ gate. To find the matrix for the $V^\dagger$ acting on qubit b, the I $\otimes$ $V^\dagger$) $\otimes$ I must be used. Equation 35 shows the matrix resulted of this product.

$$V_b^\dagger = \begin{pmatrix} 0.5-0.5i & 0.0+0.0i & 0.5+0.5i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i \\ 0.0+0.0i & 0.5-0.5i & 0.0+0.0i & 0.5+0.5i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i \\ 0.5+0.5i & 0.0+0.0i & 0.5-0.5i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i \\ 0.0+0.0i & 0.5+0.5i & 0.0+0.0i & 0.5-0.5i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i \\ 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.5-0.5i & 0.0+0.0i & 0.5+0.5i & 0.0+0.0i \\ 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.5-0.5i & 0.0+0.0i & 0.5+0.5i \\ 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.5+0.5i & 0.0+0.0i & 0.5-0.5i & 0.0+0.0i \\ 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.5+0.5i & 0.0+0.0i & 0.5-0.5i \end{pmatrix} \tag{35}$$

After finding the matrix of the $V^\dagger$ acting on qubit b, the next step is to update the matrix in the rows that the control qubit is not acting. The control qubit is on qubit c, thus the matrix is presented in Equation 36.

$$V_{bc} = \begin{pmatrix} 1.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i \\ 0.0+0.0i & 0.5-0.5i & 0.0+0.0i & 0.5+0.5i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i \\ 0.0+0.0i & 0.0+0.0i & 1.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i \\ 0.0+0.0i & 0.5+0.5i & 0.0+0.0i & 0.5-0.5i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i \\ 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 1.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i \\ 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.5-0.5i & 0.0+0.0i & 0.5+0.5i \\ 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 1.0+0.0i & 0.0+0.0i \\ 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.5+0.5i & 0.0+0.0i & 0.5-0.5i \end{pmatrix} \quad (36)$$

The matrix representation of the circuit from Figure 87 can be found by multiplying the CNOT(a, b) and V_{bc} matrices. Let M be the matrix representation for this circuit. The result matrix for this circuit is: $M = CNOT(a, b) * V_{bc}$, shown in Equation 37.

$$M = \begin{pmatrix} 1.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i \\ 0.0+0.0i & 0.5-0.5i & 0.0+0.0i & 0.5+0.5i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i \\ 0.0+0.0i & 0.0+0.0i & 1.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i \\ 0.0+0.0i & 0.5+0.5i & 0.0+0.0i & 0.5-0.5i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i \\ 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 1.0+0.0i & 0.0+0.0i \\ 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.5+0.5i & 0.0+0.0i & 0.5-0.5i \\ 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 1.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i \\ 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.5-0.5i & 0.0+0.0i & 0.5+0.5i \end{pmatrix} \quad (37)$$

It is possible to find the matrix representation for Toffoli gates using the same approach presented for the controlled gates. The only difference is that more rows of the matrix will have the identity. The strategy presented can be used to prove the equivalence between two quantum circuits by finding and comparing the matrix representation of each circuit.

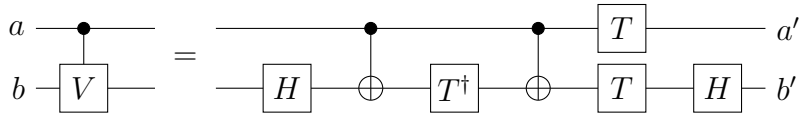
A.3 PROVING THE CIRCUIT EQUIVALENCE

In this section will be presented the proof of equivalence between two quantum circuits. Figure 88 shows the circuit that will be used as example.

To prove this circuit equivalence, both matrix representations must be found. Equation 38 shows the equation for the proof.

$$V_{ab} = H_b * CNOT(a, b) * T_b^\dagger * CNOT(a, b) * T_a * T_b * H_b \quad (38)$$

Figure 88 – V gate mapped to Clifford+T.



Source: Adapted from Soeken, Miller and Drechsler (2013).

Where:

- $V_{ab} = V$ gate controlled by qubit a;
- $H_b = I \otimes H$;
- $CNOT(a, b) =$ NOT gate controlled by qubit a;
- $T_b^\dagger = I \otimes T^\dagger$;
- $T_a = T \otimes I$;
- $T_b = I \otimes T$.

Solving the Equation 38 results in Equation 39.

$$\begin{pmatrix} 1.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i \\ 0.0+0.0i & 1.0+0.0i & 0.0+0.0i & 0.0+0.0i \\ 0.0+0.0i & 0.0+0.0i & 0.5+0.5i & 0.5-0.5i \\ 0.0+0.0i & 0.0+0.0i & 0.5-0.5i & 0.5+0.5i \end{pmatrix} = \begin{pmatrix} 1.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.0+0.0i \\ 0.0+0.0i & 1.0+0.0i & 0.0+0.0i & 0.0+0.0i \\ 0.0+0.0i & 0.0+0.0i & 0.5+0.5i & 0.5-0.5i \\ 0.0+0.0i & 0.0+0.0i & 0.5-0.5i & 0.5+0.5i \end{pmatrix} \tag{39}$$

The two matrices have the same value, meaning that the equivalence holds. This approach can be used to check if any two quantum circuits are equivalents.

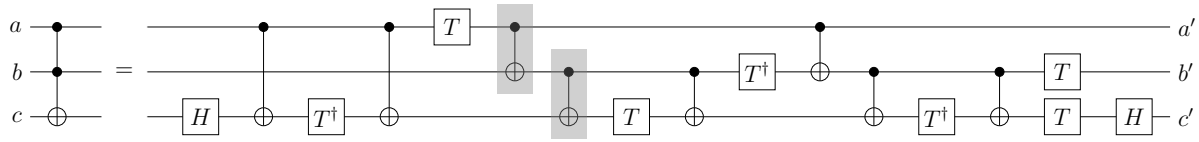
APPENDIX B - TOFFOLI OPTIMIZATIONS

In this appendix will be shown the optimization of the Toffoli to Clifford+T mapping.

B.1 FIRST POSSIBLE MAPPING

In order to reduce the number of gates and levels of the Toffoli to Clifford+T mapping, some manipulations of the gates must be done. Initially the Toffoli to Clifford+T mapping circuit has 17 gates and 13 levels, in which 8 gates are CNOT gates, as shown in Figure 89.

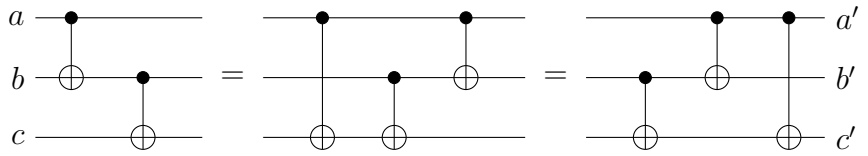
Figure 89 – Toffoli to Clifford+T circuit with 17 gates and 13 levels.



Source: Prepared by the author.

The colored CNOT gates from the circuit in Figure 89 can swap the order by using the property shown in Figure 90.

Figure 90 – Property to change CNOT gates order.

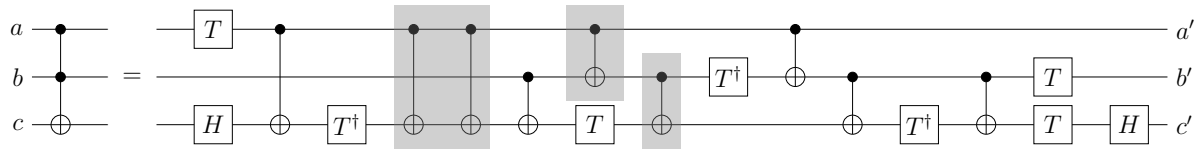


Source: Adapted from (SOEKEN; THOMSEN, 2013).

After applying the property to change the CNOT gates order and moving the T gate on qubit a to the beginning of the circuit, the updated circuit is shown in Figure 91.

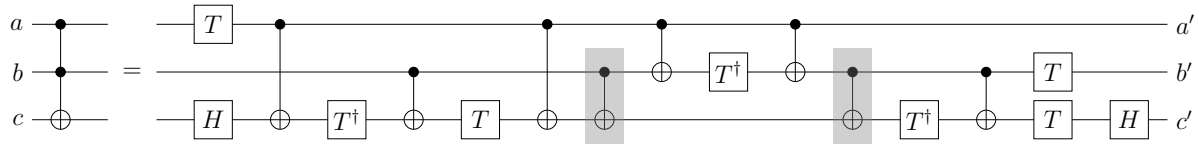
The first two adjacent CNOT gates colored in gray from the circuit in Figure 91 can be removed and on the other two colored CNOT gates can be applied the property from Figure 90. The circuit after these changes is presented in Figure 92.

Figure 91 – Toffoli circuit with order of CNOT gates changed.



Source: Prepared by the author.

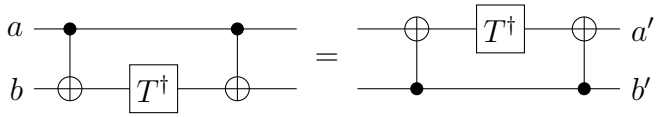
Figure 92 – Toffoli circuit after adjacent gates removed.



Source: Prepared by the author.

The colored CNOT gates from the circuit in Figure 92 can be moved together by using the property presented in Figure 93.

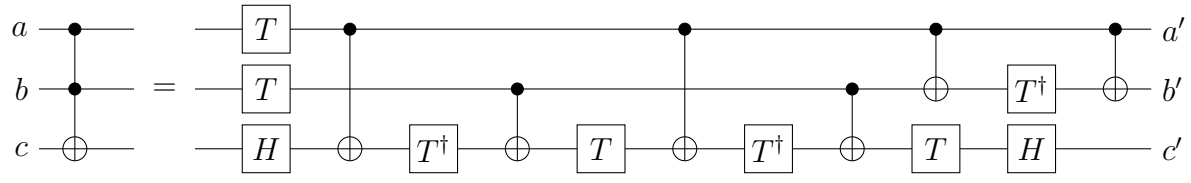
Figure 93 – Swap of CNOT gates with $T^\dagger$ gate.



Source: Prepared by the author.

After changing the position of the gates on qubits a and b with the property from Figure 93, the colored CNOT gates can be moved together and removed. Also, the T gate on qubit b can be moved to the beginning of the circuit. Then, the gates on qubits a and b can be swapped back to the original position. The final circuit is presented in Figure 94 with 15 gates and 11 levels, in which, 6 gates are CNOT gates.

Figure 94 – Final circuit for mapping a Toffoli to Clifford+T.

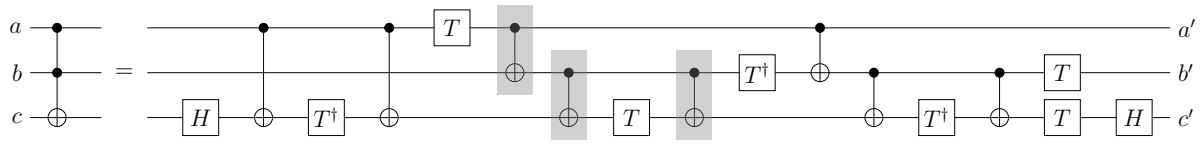


Source: Prepared by the author.

B.2 SECOND POSSIBLE MAPPING

In order to find an alternative mapping for the Toffoli gate, the manipulation of the gates must occur differently. Starting with the Toffoli mapping to Clifford+T with 17 gates and 13 levels presented in Figure 95.

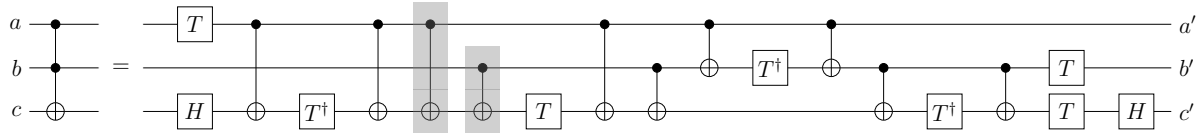
Figure 95 – Toffoli to Clifford+T mapping with 17 gates and 13 levels.



Source: Prepared by the author.

The goal in the first manipulation is to move the first colored CNOT (on qubits a and b) from the circuit in Figure 95 to the right until it reaches the $T\dagger$ gate on qubit b. To do that, the property from Figure 90 will be used to move the CNOT gate over the other colored gates in the circuit. The circuit after these movements is shown in Figure 96.

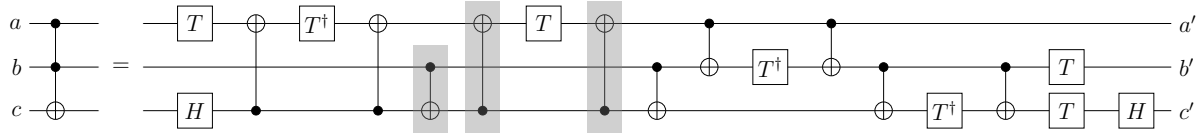
Figure 96 – Toffoli mapping with gates moved.



Source: Prepared by the author.

Instead of remove the adjacent CNOT gates, in this approach the colored CNOT gates will have the orders changed using the property from Figure 90. Then, the two set of gates can change the position of the CNOT gates using the property from Figure 93. The circuit after that is shown in Figure 97.

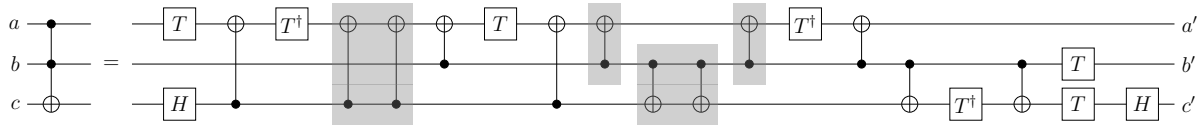
Figure 97 – Toffoli mapping with gates swapped on qubits a and c.



Source: Prepared by the author.

In order to remove the first CNOT colored in the circuit from Figure 97 on qubits b and c, it will be moved to the right until it becomes adjacent with the same CNOT. To move over the other colored CNOT in the circuit, the property from Figure 90 will be used. The updated circuit is presented in Figure 98.

Figure 98 – Toffoli mapping with adjacent CNOT gates to be removed.

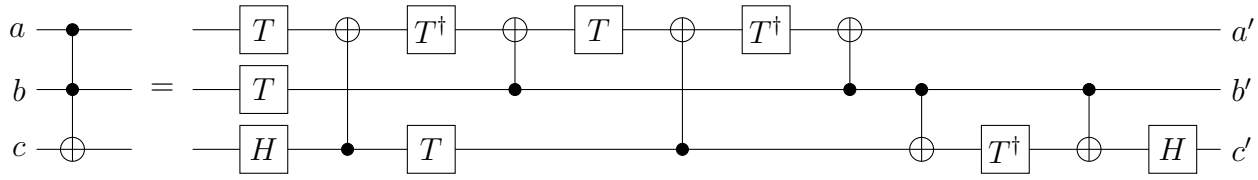


Source: Prepared by the author.

The colored CNOT gates in the circuit from Figure 98 can be removed. Also, the T

gate on qubit b and c can be moved to the left. After these changes, the final circuit with 15 gates and 10 levels. This circuit is shown in Figure 99.

Figure 99 – Final circuit for alternatively mapping a Toffoli to Clifford+T.



Source: Prepared by the author.

APPENDIX C – GAP OF THE CIRCUITS WITH SOLUTION TIME EXCEEDED

In this appendix will be shown the gap of the circuits with the Integer Linear Programming solution time exceeded. In Table 24 is presented the gap of the circuits from Chapter 4 and in Table 25 the gap of the circuits from Chapter 5. The gap is the information of how far the solver is from finding the solution of the problem.

Table 24 – ILP gap for Chapter 4.

Circuits	Time(sec)	Gap(%)
adr4_197	3,600	75.07
co14_215	3,600	81.71
cycle10_2_110	3,600	80.53
dc1_220	3,600	46.78
ham15_107	3,600	85.11
inc_237	3,600	84.18
life_238	3,600	76.97
max46_240	3,600	58.87
misex1_241	3,600	76.23
pm1_249	3,600	44.16
radd_250	3,600	73.77
rd84_253	3,600	77.19
root_255	3,600	82.88
sqn_258	3,600	61.62
sqrt8_260	3,600	76.64
squar5_261	3,600	71.45
square_root_7	3,600	37.70
sym6_316	3,600	21.77
sym9_146	3,600	33.46
sym9_148	3,600	77.81
z4_268	3,600	66.97

Table 25 – ILP gap for Chapter 5.

Circuits	Time(sec)	Gap(%)
dc1_220	3,600	47.64
ham15_108	3,600	78.28
ham15_109	3,600	84.65
hwb6_58	3,600	50.40
pm1_249	3,600	17.98
rd53_130	3,600	57.27
rd53_311	3,600	49.34
rd84_142	3,600	60.75
sym9_146	3,600	60.29
sym9_148	3,600	82.32