Study of
Reversible Circuits

THESIS SUBMITTED IN FULFILMENT OF THE
REQUIREMENTS FOR THE AWARD OF THE
DEGREE OF

MASTER OF ENGINEERING

IN

ELECTRONICS & TELECOMMUNICATION ENGINEERING

By

RANJAN KUMAR MANDAL

EXAM ROLL: M4ETC24008

REGISTRATION NO.: 163534 OF 2022-2023

Under the Guidance of

Prof. Mrinal Kanti Naskar

Department of
Electronics & Telecommunication Engineering

Jadavpur University
Kolkata 700032, West Bengal, India

May 2024

Faculty of Engineering and Technology

Jadavpur University

Certificate of Recommendation

This is to certify that the thesis entitled “Study of Reversible Circuits” has been carried out by Ranjan Kumar Mandal (Class Roll No: 002210702010, Examination Roll No.: M4ETC24008 and Registration No: 163534 OF 2022-2023) under my guidance and supervision and be accepted in partial fulfilment of the requirement for the degree of Master of Electronics & Telecommunication Engineering.

_____

Prof. Mrinal Kanti Naskar

Supervisor

Department of Electronics & Telecommunication Engineering

Jadavpur University

Kolkata – 700032

_____

Prof. Monotosh Biswas

Head of the Department

Electronics & Telecommunication Engineering

Jadavpur University

Kolkata – 700032

_____

Prof. Dipak Laha

Dean

Faculty Council of Engineering & Technology

Jadavpur University

Kolkata – 700032

Faculty of Engineering and Technology

Jadavpur University

Certificate of Approval

The forgoing thesis titled “Study of Reversible Circuits” is hereby approved as a creditable study of an engineering subject carried out and presented in a manner satisfactory to warrant its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval, the undersigned do not necessarily endorse or accept every statement made, opinion expressed, or conclusion drawn therein but only approve the thesis only for the purpose of which it has been submitted.

Committee on Final Examination for Evaluation
of the Thesis:

_____

Additional Examiner

_____

Supervisor

Declaration of Originality and Compliance of Academic Ethics

I hereby declare that this thesis contains a literature survey and original work by the undersigned candidate, as part of his Master of Electronics and Telecommunication Engineering studies.

All information in this document has been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name: Ranjan Kumar Mandal

Exam Roll No: M4ETC24008

Thesis Title: “Study of Reversible Circuits”

_____

Signature of Candidate

Acknowledgement

The successful completion of this project required guidance and support from a number of people and I am privileged to have this during the entire duration of the project.

I express my sincere respect and thanks to my project guide Prof. Mrinal Kanti Naskar for providing me an opportunity to do the project and providing me with all the guidance and support to complete the project.

I am thankful to the teachers of the Department of Electronics and Telecommunication Engineering for their encouragement and support.

I am also thankful to my family members and my friends for their support during my tenure at the project.

Ranjan Kumar Mandal

Abstract

Field of reversible circuits has a lot of research attention because of its potential connect to Low Energy Computing and Quantum Computing. In this work, we establish the mathematical basis for the analysis / synthesis of a reversible circuit and its components (the reversible gates). We study reversible gates and the circuits developed with these components. We develop a program which can convert any function / truth table (for up to 10 input variables) to a reversible circuit with Multi-Controlled X (MCX) gates.

Keywords: Reversible circuits, Reversible gates, Reversible logic, Quantum cost, Quantum computing

Preface

About this document

This document is the thesis for the project entitled “Study of Reversible Circuits” done in the Department of Electronics and Telecommunication Engineering.

Table of Contents

ACKNOWLEDGEMENT ..... **I**

ABSTRACT ..... **II**

PREFACE ..... **III**
About this document iii
Table of Contents iii
List of figures v
List of tables vi

CHAPTER 1 INTRODUCTION **1**
1.1 Reversible circuits 1
1.2 Motivation 3
1.2.1 Low energy computing 3
1.2.2 Relation to quantum computing 4
1.3 History and Literature Review 4
1.3.1 History 4
1.3.2 Literature review 4
1.3.2.1 Importance 5
1.3.2.2 Mathematical background 5
1.3.2.3 Reversible circuits and quantum computing 5
1.3.2.4 Reversible gates 5
1.3.2.5 Circuits synthesis based on truth table 5
1.3.2.6 Programming (Python, Qiskit) 5
1.3.2.7 Optimization 5

CHAPTER 2 MATHEMATICAL APPROACH TO REVERSIBILITY **6**
2.1 Boolean logic 6
2.1.1 Boolean function with one input (A) 6
2.1.2 Boolean function with two inputs (A, B) 6
2.1.3 Boolean function with n inputs 8
2.1.4 Reed Muller Form of a Boolean function 9
2.1.5 General observation of Boolean function and reversibility 10
2.2 Matrices and reversibility 10
2.2.1 Operation represented by a matrix 11
2.2.2 General observation on matrices and reversibility 12
2.3 Cycles and groups 13
2.4 Reversibility with Boolean function, matrices and cycles 14
2.4.1 Reversibility with Boolean function 15
2.4.2 Reversibility with operator matrices 15
2.4.3 Reversibility with groups / cycles 16

CHAPTER 3 QUANTUM COMPUTING – THE PARADIGM **17**
3.1 Classical computing paradigm 17
3.2 Quantum computing paradigm 17

3.2.1 Operations on a qubit 18
3.2.1.1 X gate or Pauli-X gate: 18
3.2.1.2 Hadamard Gate 19
3.2.2 Measurement of a qubit 19
3.2.3 Computing with quantum circuits 19

CHAPTER 4 REVERSIBLE LOGIC GATES 20
4.1 NOT gate 20
4.2 CNOT gate 20
4.3 Toffoli gate 22
4.4 Fredkin gate 24
4.5 Peres gate 26
4.6 Hadamard gate 28
4.7 Pauli operator gates 29
4.7.1 Pauli X gate 29
4.7.2 V gate 30
4.7.3 Controlled V and controlled V dagger gates 30
4.7.4 Pauli Y gate 32
4.7.5 Pauli Z gate 33
4.8 MCX gate 34
4.8.1 CX gate 34
4.8.2 CCX gate 34
4.8.3 CCCX gate 35
4.8.4 CnX gate 36
4.9 Quantum cost of a gate 36

CHAPTER 5 BUILDING REVERSIBLE CIRCUITS 38
5.1 Circuits with reversible gates 38
5.1.1 Toffoli gate as a complete logic 39
5.1.2 Fredkin gate as a complete logic 40
5.1.3 Interconnections among the reversible gates 40
5.2 Implementation of a function with Bennett trick 41
5.2.1 Reversible AND gate 42
5.2.2 Reversible OR gate 43
5.2.3 Reversible Full adder Sum 43
5.2.3.1 Classical circuit view of full adder Sum 43
5.2.3.2 Block diagram view of full adder Sum 43
5.2.3.3 Quantum circuit view of full adder Sum 44
5.2.4 Reversible Full adder Carry (Toffoli, NOT, CNOT) 44
5.2.4.1 Block diagram view of full adder Carry 44
5.2.4.2 Quantum circuit view of full adder Carry 45

CHAPTER 6 REVERSIBLE CIRCUIT FOR A FUNCTION 46
6.1 The approach 46
6.2 Program development 48
6.2.1 Input 48
6.2.2 Positive Polarity Reed Muller form (PPRM) 49
6.2.3 Processing with MCX gate 49
6.2.4 Drawing the circuit 49
6.3 Circuits developed through the program 49
6.3.1 Full adder (2 bit) 50
6.3.2 Mux (2 to 1) 51

CHAPTER 7 OPTIMIZATION OF A REVERSIBLE CIRCUIT 53

CHAPTER 8 RESULTS AND DISCUSSIONS 57

CHAPTER 9 CONCLUSION AND FUTURE SCOPE 60
9.1 Conclusion 60

Mathematical view of the reversibility in circuits 60
Reversible gates 60
Building reversible circuits 60
Program to convert a truth table to a circuit 60
Discussion on optimization 60
9.2 Future scope 60

BIBLIOGRAPHY 61

APPENDIX 63
Bijective relation 63
Relation in general 63
Function in general 63
Injective function 64
Surjective function 65
Bijective function 65
Program to generate reversible circuit for a given truth table 66

List of figures

Fig 1: Boolean logic / circuit (AND and XOR) 1
Fig 2: Matrix representation of Toffoli gate 11
Fig 3: Input output permutation (bijective) 15
Fig 4: Truth table of NOT gate with symbol 20
Fig 5: CNOT gate and the equivalent circuit 20
Fig 6: Toffoli gate – classical equivalent, block diagram and quantum circuit 22
Fig 7: Fredkin gate – Classical equivalent circuit 24
Fig 8: Fredkin gate – Block diagram and quantum circuit 24
Fig 9: Peres gate – Classical equivalent, block diagram and quantum circuit 26
Fig 10: Peres gate – Classical equivalent, block diagram and quantum circuit 26
Fig 11: The Pauli X gate 29
Fig 12: The V gate 30
Fig 13: The V dagger gate 30
Fig 14: Quantum equivalent circuit for Controlled V gate 31
Fig 15: Quantum equivalent circuit for Controlled V Dagger gate 31
Fig 16: Pauli Y gate 32
Fig 17: Pauli Z gate 33
Fig 18: CX gate / C1X gate 34
Fig 19: CCX gate / C2X gate 34
Fig 20: CCCX gate – classical and quantum equivalent circuits 35
Fig 21: Decomposition of Toffoli gate 36
Fig 22: Getting the AND logic from a Toffoli gate 39
Fig 23: CNOT gate as a fanout 41
Fig 24: Irreversible circuit to reversible circuit 41
Fig 25: Reversible circuit with garbage handled 42
Fig 26: Reversible AND gate with Bennett trick 42
Fig 27: Reversible OR gate with Bennett trick 43
Fig 28: Full adder Sum (classical circuit view) 43
Fig 29: Full adder Sum (Toffoli, NOT, CNOT) with Bennett trick 43
Fig 30: Full adder Sum – the quantum circuit view 44
Fig 31: Full adder Carry using Bennett Trick 44
Fig 32: Full adder Carry – quantum circuit view 45
Fig 33: MCX sample configuration and output 47
Fig 34: The program approach 48

Fig 35: Circuit for 2 input full adder 51
Fig 36: Circuit for 2 to 1 mux 52
Fig 37: Circuit for the given truth table 59
Fig 38: Relation 63
Fig 39: Example of a function 63
Fig 40: A relation which is not a function 64
Fig 41: Example of an injective function 64
Fig 42: Example of a surjective function 65
Fig 43: Example of a bijective function 65

List of tables

Table 1: Truth tables for AND Gate and XOR Gate 1
Table 2: Reverse truth tables for XOR gate 1
Table 3: Reverse truth tables for AND gate 2
Table 4: Truth tables for single input Boolean function 6
Table 5: Truth tables for two input Boolean function 6
Table 6: Truth tables for 2 input OR and XOR gates 7
Table 7: Logic gates with two inputs and two outputs 8
Table 8: Truth table for a reversible circuit 8
Table 9: Possible 2 input reversible functions 9
Table 10: Truth table for Boolean function 9
Table 11: PPRM example with 2 variables 10
Table 12: PPRM example with 3 variables 10
Table 13: Truth table of a 3 input reversible circuit (Toffoli gate) 11
Table 14: A simple truth table for a 3 input 3 output circuit 13
Table 15: Truth table of a function 14
Table 16: The truth table, matrix and cycles for CNOT gate 20
Table 17: The truth table, matrix and cycles for Toffoli gate 22
Table 18: The truth table, matrix and cycles for Fredkin gate 24
Table 19: Truth table for CCCX gate 35
Table 20: Truth table for AND gate and XOR gate 38
Table 21: Truth table for Toffoli gate with rows for C=0 shaded 38
Table 22: Truth table for Toffoli gate with C = 0 39
Table 23: Full adder sum and carry 39
Table 24: Truth table for a 2 input 2 output function 46
Table 25: Truth table for 2 input full adder 50
Table 26: Truth table for 2 to 1 mux 51
Table 27: Truth table for a reversible function 57
Table 28: A truth table to be implemented 58

Chapter 1 Introduction

1.1 Reversible circuits

A binary logic circuit implements a Boolean function. For a function taking n input variables and producing m output variables, we denote the same as: $\{0, 1\}^n \rightarrow \{0, 1\}^m$. For example, in a function with 2 input variables (the input vector) producing 2 output variables (the output vector), we can uniquely determine the output vector for a given input vector. However, is the reverse also applicable? That is, given the output vector, would it be possible to determine the input vector uniquely? We examine the following circuits and corresponding truth tables for AND Gate and XOR Gate.

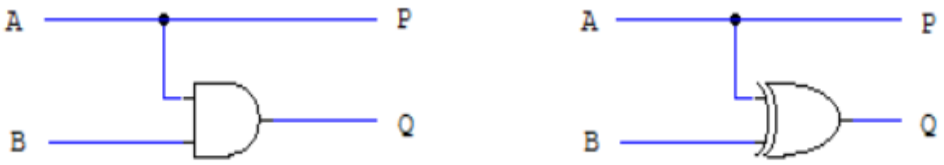


Fig 1: Boolean logic / circuit (AND and XOR)

Circuit with AND gate				
A	B	P=A	Q=A.B	Remarks
0	0	0	0	From two output lines, we cannot determine A and B
0	1	0	0	
1	0	1	0	
1	1	1	1	

Circuit with XOR Gate				
A	B	P=A	Q=A⊕B	Remarks
0	0	0	0	From two output lines, we can determine A and B
0	1	0	1	
1	0	1	1	
1	1	1	0	

Table 1: Truth tables for AND Gate and XOR Gate

In both the truth tables, number of inputs (2) is same as number of outputs ($\{0, 1\}^2 \rightarrow \{0, 1\}^2$). For both gates, if the input vector (A, B) is provided, the output vector (P, Q) can be determined uniquely. For example, in case of the AND Gate, if A is 0 and B is 1, then P=0 and Q=0. Also, in case of the XOR Gate, if A=0 and B=1, then P=0 and Q=1.

We take a closer look on the path of the reverse for the XOR gate. Next tables are the truth tables for the reverse path (determining A and B from P and Q) with two situations (case 1 and case 2). In case 1, we have the input as (Q) and we expect the output to be (A, B). In case 2, we have the inputs as (P, Q) and we expect the outputs to be (A, B). Also, P = A and Q = A ⊕ B

XOR, Reverse truth table, Q = A ⊕ B (one output)		
Q = A ⊕ B	**A**	**B**
0	0	0
1	0	1
1	1	0
0	1	1

Case 1 (number of outputs: 1)

XOR, Reverse truth table, P = A, Q = A ⊕ B (two outputs)			
P = A	**Q = A ⊕ B**	**A**	**B**
0	0	0	0
0	1	0	1
1	1	1	0
1	0	1	1

Case 2 (number of outputs: 2)

Table 2: Reverse truth tables for XOR gate

In the case with one output (case 1):

- Q = 0, (A, B) = (0, 0) or (A, B) = (1, 1)
- Q = 1, (A, B) = (0, 1) or (A, B) = (1, 0)

For Q = 0, there are two options, (A, B) = (0, 0) or (A, B) = (1, 1). Hence, it would not be possible to uniquely determine (A, B).

In the case with two outputs (case 2):

- (P, Q) = (0, 0), (A, B) = (0, 0)
- (P, Q) = (0, 1), (A, B) = (0, 1)
- (P, Q) = (1, 1), (A, B) = (1, 0)
- (P, Q) = (1, 0), (A, B) = (1, 1)

For each combination of (P, Q) values, there is a unique combination of (A, B) values. Hence, in this case (P = A, Q = A $\oplus$ B), it is possible to uniquely determine (A, B) if (P, Q) is given. This is an example of a reversible circuit.

Now we take a look on the path of the reverse for the AND gate. We have the next truth tables for the reverse path (determining A and B from P and Q) with two situations (case 1 and case 2). In case 1, we have the input as (Q) and we expect the output to be (A, B). In case 2, we have the inputs as (P, Q) and we expect the outputs to be (A, B). Also, P = A and Q = A . B

AND, Reverse truth table, Q = A . B (one output)		
Q = A . B	A	B
0	0	0
0	0	1
0	1	0
1	1	1

Case 1 (number of outputs: 1)

AND, Reverse truth table, P = A, Q = A . B (two outputs)			
P = A	Q = A. B	A	B
0	0	0	0
0	0	0	1
1	0	1	0
1	1	1	1

Case 2 (number of outputs: 2)

Table 3: Reverse truth tables for AND gate

In the case with one output (case 1):
Q = 0, (A, B) = (0, 0), or (A, B) = (0, 1), or (A, B) = (1, 0)
Q = 1, (A, B) = (1, 1)
For Q = 1, we can infer that the combination (A, B) = (1, 1). However, for Q = 0, we cannot uniquely infer about (A, B), because there are three options for (A, B) – these are (0, 0), (0, 1) and (1, 0). Hence, it is not possible to uniquely determine (A, B).

In the case with two outputs (case 2):
(P, Q) = (0, 0), (A, B) = (0, 0) or (A, B) = (0, 1)
(P, Q) = (1, 0), (A, B) = (1, 0)
(P, Q) = (1, 1), (A, B) = (1, 1)

For (P, Q) = (0, 0), (A, B) = (0, 0) or (A, B) = (0, 1). In this case (P = A, Q = A . B), it is not possible to uniquely determine (A, B) if (P, Q) is given. This is an example of a circuit which is not reversible.

Hence, there exists a category of circuits where number of inputs equals number of outputs and a given output vector uniquely points to a particular input vector. These circuits are called reversible circuits. The above circuit with XOR gate is an example of reversible circuit.

Circuits which do not satisfy the above criteria are not reversible circuits. For the circuit with AND gate, number of inputs equals number of outputs (2 inputs, 2 outputs) and there is an output vector ((P, Q) = (0, 0)) for which the input vector cannot be uniquely determined ((A, B) = (0, 0), or (0, 1)). Hence this circuit with AND gate is not a reversible circuit.

Hence, based on the above, a computation is reversible if it can be "undone" in the sense that the output contains sufficient information to reconstruct the input, that is, no input information is erased, and it is also common to require that no information is duplicated (if duplicated, the reverse path will not remain unique). A reversible logic implementation will be an n-input n-output set up with one-to-one mapping. This enables getting the outputs from the inputs and also the inputs from the outputs. In the synthesis of reversible circuits direct fan-out is not allowed as 'one to many' connection is not reversible.

A reversible circuit is an n-input n-output logic device with one-to-one mapping. Reversible circuits are those circuits which can uncompute whatever was computed. This enables getting the outputs from the inputs and also the inputs from the outputs [1] [2].

Also in the synthesis of reversible circuits direct fan-out is not allowed as one-to-many concept is not reversible[1] [3]. However fan-out in reversible circuits is achieved using additional gates [3]. A reversible circuit should be designed using minimum number of reversible logic gates. From the point of view of reversible circuit design, there are different parameters for determining the complexity and performance of circuits.

The basic properties are:

- $\{0, 1\}^n = \{0, 1\}^n$
- The transformation is bijective in nature [1] [2] providing one-to-one mapping (refer to Appendix, Bijective relation)

As discussed earlier, fan-out is not permitted because this will cause loss of uniqueness of return path. We have to use another gate to incorporate a fan-out connection.

This field of reversible circuits finds its application in many areas which include quantum computing and low power computing.

1.2 Motivation

The computer industry has gone through major improvements in a number of areas. These include the computing power in an integrated circuit (IC) in low energy computing and the field of quantum computing. We consider these two areas from the perspective of reversible circuits.

1.2.1 Low energy computing

Two propositions out of many caused a substantial amount of focus in reversible circuits.

Landauer (in 1961) stated that the information loss (0 to 1 and 1 to 0) in the Boolean circuits cause generation of heat or loss of energy [4]. These transitions are inevitable because we want the circuits to have controlled transitions to give us results. Quantitatively, loss of one bit of information dissipates KT ln(2) Joules of heat energy (K is Boltzmann's constant ≈ 1.380658×10^{-23} J/K, T is the absolute temperature in Kelvins) [5] [6]. While it is not very significant for one bit of information in room temperature (≈ 2.9×10^{-21} J), it would be substantial for all bits operating in the universe.

Bennett (in 1973) stated that with reversible circuits (where the information is not lost in processing), the energy dissipation due to information loss can be avoided, hence the technology of tomorrow would have to move towards reversible circuits.

With the advent of technology for integrated circuits, as the demand has been 'smaller', 'faster' and 'more and more in number', the IC's became faster, smaller and packed more and more number of transistors. This resulted in uncontrolled heat generation (more frequency → more heat, less volume → more chance of melting, more components → more energy dissipation) [5]. This was not to be stopped because our planet required more and more computing resources.

Moore's law (Gordan Moore, 1965) stated that in an IC, the number of transistors doubles in 18-24 months. This actually happened in general till 1960's, gave us more computing power while size of the components moving towards the size of an atom!

The research is on for the solution to this issue – trying to handle challenges like connectivity and reversibility at all levels (down to the level of manufacturing with silicon / substrate).

1.2.2 Relation to quantum computing

The field of quantum computing has seen a lot of focus especially in the past five or six decades. This comes for its potential to bring in sea change in computing paradigm for areas including cryptography, cyber security, algorithm (like p versus np problems, factorization, handling numbers, searches) etc [7]. A small example of its disruptive nature would be the security system for our bank transactions – which has to be relooked once quantum computing comes to its full form.

A feature inherent to quantum computing is that all transformations / operations (except measurement) under its purview are reversible operations, following the rule of unitary matrices and the related properties [7] [2] [8].

Reversible circuits follow the same principles of unitary matrices. Study of reversible circuits would greatly help in manipulating the implementation of quantum algorithms [2] [7].

This work is mainly on the reversible circuit's inherent relation to quantum computing.

1.3 History and Literature Review

Substantial work has been done in a number of areas related to reversible circuits.

In this section, we describe the work done and relevant background material on reversible circuits, especially in connection to quantum computing paradigm.

1.3.1 History

The work on reversibility of circuits especially connected to quantum computing has started in late 1940s. From physics, Pauli and Feynman, Landauer and Bennett from Engineering and other significant personalities like Toffoli and Fredkin worked on the reversibility to come up with gates, circuit implementation and methods [7].

Currently large and relevant teams are working across the globe on quantum computing and its reversibility in terms of harnessing its power to change the way computing is thought of.

1.3.2 Literature review

The major areas taken up in relevant field are:

- Relevance of reversible circuits
- Mathematical background
    - Boolean algebra
    - Matrices
    - Cycles / group theory
- Reversible circuits and its direct relation to quantum computing
- Reversible gates
- Circuits synthesis based on truth table
- Programming (Python [9], Qiskit [10])
- Optimization

1.3.2.1 Importance

Importance of study of reversible circuits has been highlighted [5] [1]. This is mainly in the areas of:

- Quantum computing paradigm [8]
- Potential solutions dissipation of low energy for the industry [11]

1.3.2.2 Mathematical background

Reversible circuits have been considered from various perspectives including Boolean algebra, cycles / group theory and matrices [1] [2]. The seemingly different approaches finally culminate into certain set of properties of reversible circuits [7].

This work mainly covers the relationship of reversible circuits in the field of quantum computing.

1.3.2.3 Reversible circuits and quantum computing

The quantum computing operations on qubits (unitary operations) and their equivalence in reversible circuits have been covered [8] [12] [13] [7].

1.3.2.4 Reversible gates

Various gates in classical domain and in quantum domain have been illustrated [14] [15]. Truth tables [16] and their corresponding matrices are indicated. The functions realized by the gates are referenced [17] [11]. Unitary nature in a circuit of gates has been taken up.

1.3.2.5 Circuits synthesis based on truth table

Using the gates, various functions are realized – with circuit ideas for larger circuits [6]. These realizations are unitary in nature. Bennet trick has been used as one of the means to achieve the reversibility [16].

1.3.2.6 Programming (Python, Qiskit)

Program has been written to generate reversible circuit for realizing any arbitrary function [18] [9][10]. The idea of automated circuit realization is discussed [19].

1.3.2.7 Optimization

Various circuits and optimization for some of them have been discussed [20] [21] [22] [4] [23] [24].

Chapter 2 Mathematical approach to reversibility

This section describes the mathematical approach to the operations which are being performed in relation to the reversible circuits.

2.1 Boolean logic

A Boolean function is expressed as $\{0, 1\}^n \rightarrow \{0, 1\}^m$.

2.1.1 Boolean function with one input (A)

We consider the case where n = m = 1, a single input single output scenario. The input can be either 0 or 1 and the output also can be either 0 or 1, i.e. there are two different inputs [11].

There can be four functions of A. Refer to the following truth tables.

A	f(A) = 0
0	0
1	0

(a)

A	f(A) = A
0	0
1	1

(b)

A	f(A) = $\overline{A}$
0	1
1	0

(c)

A	f(A) = 1
0	1
1	1

(d)

Table 4: Truth tables for single input Boolean function

- In above truth table (a), the output is 0 irrespective of value of A, i.e. f(A) = 0, a constant output
- In truth table (d), the output is 1 irrespective of value of A, i.e. f(A) = 1, a constant output
- In truth table (b), if A = 0, f(A) = 0; if A = 1, f(A) = 1. Here, f(A) = A, this is an identity function where output is equal to the input
- In truth table (c), if A = 0, f(A) = 1; if A = 1, f(A) = 0. Here, f(A) = $\overline{A}$, this is NOT function where output is equal to the inverse of the input

2.1.2 Boolean function with two inputs (A, B)

We now examine the scenario with two input bits (A and B). The inputs can have four values (00, 01, 10, 11) [11] [2]. The possible outputs / truth tables can be of the form listed in the table below ($\{0, 1\}^2 \rightarrow \{0, 1\}^1$).

A	**B**		**f0**	**f1**	**f2**	**f3**	**f4**	**f5**	**f6**	**f7**	**f8**	**f9**	**f10**	**f11**	**f12**	**f13**	**f14**	**f15**
0	0		0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1		0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0		0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1		0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Table 5: Truth tables for two input Boolean function

- It is observed that the functions (including canonical form) are:
- f0 = 0
- f1 = A . B
- f2 = A . $\overline{B}$
- f3 = A . B + A . $\overline{B}$ = A
- f4 = $\overline{A}$. B
- f5 = $\overline{A}$. B + A . B = B

- f6 = $\overline{A}$. B + A . $\overline{B}$
- f7 = $\overline{A}$. B + A . $\overline{B}$ + A .B
- f8 = $\overline{A}$. $\overline{B}$
- f9 = $\overline{A}$. $\overline{B}$ + A . B
- f10 = $\overline{A}$. $\overline{B}$ + A . $\overline{B}$ = $\overline{B}$
- f11 = $\overline{A}$. $\overline{B}$ + A . $\overline{B}$ + A . B
- f12 = $\overline{A}$. $\overline{B}$ + $\overline{A}$. B = $\overline{A}$
- f13 = $\overline{A}$. $\overline{B}$ + $\overline{A}$. B + A . B
- f14 = $\overline{A}$. $\overline{B}$ + $\overline{A}$. B + A . $\overline{B}$
- f15 = $\overline{A}$. $\overline{B}$ + $\overline{A}$. B + A . $\overline{B}$ + A . B = 1

It is noticed that the two variables A and B give 2^2 = 4 input combinations and number of functions is 4^2 = 16 (denoted here as f0 to f15). It is also observed that f0 = 0 and f15 = 1 are constant output functions. Also, f3 (= A), f5 (= B), f10 (=$\overline{B}$) and f12 (=$\overline{A}$) are functions of one variable only.

We now take two examples of the function $\{0, 1\}^2 \rightarrow \{0, 1\}^2$ with A and B as input and P and Q as output to check the reversibility (number of inputs equals number of outputs and bijective relation).

A	B		P = A	Q = A + B
0	0		0	0
0	1	→	0	1
1	0		1	1
1	1		1	1

(a)

A	B		P = A	Q = A ⊕ B
0	0		0	0
0	1	→	0	1
1	0		1	1
1	1		1	0

(b)

Table 6: Truth tables for 2 input OR and XOR gates

Table (a) depicts a circuit with two inputs (A, B) and two outputs (P = A, Q = A + B). Table (b) depicts a circuit with two inputs (A, B) and two outputs (P = A, Q = A ⊕ B).

In table (a), for the combination (P = 1, Q = 1), the input cannot be found out uniquely ((A, B) = (1, 0) or (A, B) = (1, 1)). This function (OR) is not reversible. Output vector (1, 0) does not exist for this function. The output has a repeated vector (1, 1) which is appearing twice. Since both input and output have two variables, there would be 2^2 = 4 combinations on both sides. For the relation to be bijective, all combinations should exist and no combination should get repeated.

In table (b), each row / combination of input (input vector) gives a unique output vector. For the reverse, each row / combination of output (output vector) gives a unique input vector.

For (P, Q) = (0, 0), (A, B) = (0, 0)
For (P, Q) = (0, 1), (A, B) = (0, 1)
For (P, Q) = (1, 0), (A, B) = (1, 1)
For (P, Q) = (1, 1), (A, B) = (1, 0)

This function (XOR) is reversible [11] [7] [2].

Both input and output have two variables with 2^2 = 4 combinations on both sides. Also, all inputs are to have a corresponding output and all outputs are to have a corresponding input – relation would be both injective and surjective (giving a bijective relation). For the relation to be bijective, all combinations should exist and no combination should get repeated – one to one and onto nature.

For reversible circuits of two variables (A, B), we take two functions (P, Q). Refer to the following tables:

A	B		P=A	Q=A.B
0	0		0	0
0	1	→	0	1
1	0		1	1
1	1		1	1

(a)

A	B		P=A	Q=A⊕B
0	0		0	0
0	1	→	0	1
1	0		1	1
1	1		1	1

(b)

A	B		P=A⊕B	Q=$\bar{A}.\bar{B}$+A.B
0	0		0	1
0	1	→	1	0
1	0		1	0
1	1		0	1

(c)

Table 7: Logic gates with two inputs and two outputs

For table (a), output vector (0, 0) is repeated. Hence AND is not a reversible operation.

Observing table (b), we infer that for a reversible circuit, all the input vectors appear as output vector, with a different sequence [11]. This also necessitates that for a reversible circuit, each function (P, Q), there are equal number of 0's and 1's i.e. the reversible functions are balanced functions.

Table (c) is an example where both functions (P, Q) are balanced, still the circuit is not reversible, because each of (P, Q) = (1, 0) and (P, Q) = (0, 1) is repeated twice. In addition to each function being balanced, the combination has to span the entire set of possible outputs.

2.1.3 Boolean function with n inputs

With n variables, we can have 2^n input combinations. With 2^n input combinations, we get 2^{2^n} different functions [1] [7] [2].

For n = 1, number of possible functions = $2^{2^1} = 2^2 = 4$.
For n = 2, number of possible functions = $2^{2^2} = 2^4 = 16$.
For n = 3, number of possible functions = $2^{2^3} = 2^8 = 256$, and so on.

For reversible circuits with n inputs, there are n outputs (n different functions). The output functions taken together should cover all possible values.

A	B		P=A	Q=A⊕B
0	0		0	0
0	1	→	0	1
1	0		1	1
1	1		1	0

Table 8: Truth table for a reversible circuit

For example, in the truth table above, P and Q together cover all possible combinations (00, 01, 10, 11). Each function (P and Q) is a balanced function.

For n inputs, there would be 2^n input combinations, there would be n outputs (balanced functions) covering all 2^n values.

These balanced functions cover all the 2^n values, there would be $2^n!$ of n bit possible output functions which are reversible. Within them n! times internal combinations are also taken. Hence the actual number of possible reversible unique functions is ($2^n!$ / n!). for 2 inputs, it becomes (4! / 2!) = 12.

These twelve are given in the following figure.

It may be noted that the combination (3, 6) is CNOT gate. Also, (3, 6) and (6, 3) would give the same circuit, hence one set is shown as a unique set. Hence there is a division (by n!) in the formula.

Function values for 2 input reversible circuits (12 nos)																							
3	5	3	6	3	9	3	10	5	6	5	9	5	12	6	10	6	12	9	10	9	12	10	12
0	0	0	0	0	1	0	1	0	0	0	1	0	1	0	1	0	1	1	1	1	1	1	1
0	1	0	1	0	0	0	0	1	1	1	0	1	1	1	0	1	1	0	0	0	1	0	1
1	0	1	1	1	0	1	1	0	1	0	0	0	0	1	1	1	0	0	1	0	0	1	0
1	1	1	0	1	1	1	0	1	0	1	1	1	0	0	0	0	0	1	0	1	0	0	0

Table 9: Possible 2 input reversible functions

In a similar way, for 3 input 3 output reversible circuit, the number of functions will be $2^{2^3} = 2^8 = 256$, and number of possible reversible circuits will be $2^3! / 3! = 8! / 3! = 6720$.

2.1.4 Reed Muller Form of a Boolean function

We have seen the Boolean function expressed in a truth table.
For example, in the following truth table, the function (P = f(A, B, C)) is shown.

A	B	C		P
0	0	0		0
0	0	1		1
0	1	0		0
0	1	1	→	1
1	0	0		0
1	0	1		1
1	1	0		1
1	1	1		0

Table 10: Truth table for Boolean function

To get the sum of product (SoP) in canonical form (with ORing of minterms or product / ANDed terms), we write the function as OR of the AND of the terms which give the output as 1.

In this case, P = $\overline{A}.\overline{B}.C + \overline{A}.B.C + A.\overline{B}.C + A.B.\overline{C}$

We have the following identities:
- $A + B = A \oplus B \oplus A . B$
- $A \oplus A = 0$
- $1 \oplus A = \overline{A}$
- $0 \oplus A = A$

Reed Muller form is another representation [25] of the Boolean function. It is of the form as given below [7] [2].

- For 2 input function, Reed Muller form is:
  - $f(A, B) = c_{00} \oplus c_{01}B \oplus c_{10}A \oplus c_{11}AB$
- For 3 input function, Reed Muller form is:
  - $f(A, B, C) = c_{000} \oplus c_{001}C \oplus c_{010}B \oplus c_{011}BC \oplus c_{100}A \oplus c_{101}AC \oplus c_{110}AB \oplus c_{111}ABC$

The main properties of the above Reed Muller form are [25]:

- Reed Muller expression is a complete logic, any function can be expressed this way
- Reed Muller coefficients (c terms, c_{ijk}) can be either 0 or 1
- Using only positive polarity Reed Muller (PPRM) terms - no inverse terms of the variables
- For n variables, there are 2^n terms at the maximum

Few examples of Reed Muller (PPRM) expression with 2 variables:

A	B		P
0	0		1
0	1	→	1
1	0		0
1	1		1

(a) $\overline{A}.\overline{B} + \overline{A}.B + A.B = 1 \oplus A \oplus A.B$

A	B		P
0	0		0
0	1	→	1
1	0		1
1	1		1

(b) $\overline{A}.B + A.\overline{B} + A.B = A \oplus B \oplus A.B$

Table 11: PPRM example with 2 variables

Few examples of Reed Muller (PPRM) expression with 3 variables:

A	B	C		P
0	0	0		0
0	0	1		1
0	1	0		0
0	1	1	→	1
1	0	0		0
1	0	1		0
1	1	0		1
1	1	1		0

(a) $\overline{A}.\overline{B}.C + \overline{A}.B.C + A.B.\overline{C}$

$= C \oplus A.B \oplus A.C \oplus A.B.C$

A	B	C		P
0	0	0		1
0	0	1		1
0	1	0		0
0	1	1	→	0
1	0	0		0
1	0	1		0
1	1	0		1
1	1	1		0

(b) $\overline{A}.\overline{B}.\overline{C} + \overline{A}.\overline{B}.C + A.B.\overline{C}$

$= 1 \oplus A \oplus B \oplus A.B.C$

A	B	C		P
0	0	0		1
0	0	1		1
0	1	0		0
0	1	1	→	0
1	0	0		1
1	0	1		0
1	1	0		1
1	1	1		0

(c) $\overline{A}.\overline{B}.\overline{C} + \overline{A}.\overline{B}.C + A.\overline{B}.\overline{C} + A.B.\overline{C}$

$= 1 \oplus B \oplus AB \oplus AC$

Table 12: PPRM example with 3 variables

2.1.5 General observation of Boolean function and reversibility

- With n variables, there are 2^n input combinations and 2^{2^n} different functions [1] [2]
- For n inputs, there would be n outputs (balanced functions) covering all 2^n values
- For n inputs, there would be $2^n!$ of n bit possible output functions which are reversible. Within them n! times internal combinations are also taken. Hence the actual number of possible reversible unique functions is $(2^n! / n!)$ [2]
- A Boolean function can be expressed in AND – XOR terms (Reed Muller terms) [25]

2.2 Matrices and reversibility

A circuit can be viewed as a box which takes a set of inputs (input vector), processes the same and produces a set of outputs (output vector). For example, we consider the following truth table:

A	B	C		P=A	Q=B	R=AB⊕C
0	0	0		0	0	0
0	0	1		0	0	1
0	1	0		0	1	0
0	1	1	→	0	1	1
1	0	0		1	0	0
1	0	1		1	0	1
1	1	0		1	1	1
1	1	1		1	1	0

Table 13: Truth table of a 3 input reversible circuit (Toffoli gate)

One can see that this is a function of ($\{0, 1\}^3 \rightarrow \{0, 1\}^3$). The output covers all possible combinations. Input vector can be obtained from output vector. Hence this is a reversible circuit (a Toffoli gate) [7].

2.2.1 Operation represented by a matrix

An operation is performed on (A, B, C) yielding (P, Q, R). We can represent the operation through a matrix (T x Input = Output). It is represented as:

$$\begin{pmatrix} 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&1\\ 0&0&0&0&0&0&1&0 \end{pmatrix} \text{x} \begin{pmatrix} 0&0&0\\ 0&0&1\\ 0&1&0\\ 0&1&1\\ 1&0&0\\ 1&0&1\\ 1&1&0\\ 1&1&1 \end{pmatrix} = \begin{pmatrix} 0&0&0\\ 0&0&1\\ 0&1&0\\ 0&1&1\\ 1&0&0\\ 1&0&1\\ 1&1&1\\ 1&1&0 \end{pmatrix}$$

Fig 2: Matrix representation of Toffoli gate

The truth table can be represented this way through a matrix [2].

If U is a matrix, then $U^\dagger$, (called U dagger) is the adjoint (complex conjugate transpose) of U. For a matrix with real values, the adjoint is the transpose itself.

If $UU^\dagger = U^\dagger U = I$ (identity matrix), then the matrix U is a unitary matrix.

All operations (except measurement) in quantum computing are reversible and unitary operations. We would see that all unitary operations in classical domain are also reversible. Hence the study of reversible circuits in classical domain has turned out to be very important.

If T = Toffoli Matrix given above, $T^\dagger$ = adjoint of Toffoli Matrix

$$= \begin{pmatrix} 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&1\\ 0&0&0&0&0&0&1&0 \end{pmatrix}$$ which is same as T. So, in this case, $T = T^\dagger$.

Now, $TT^{\dagger} = \begin{pmatrix} 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&1\\ 0&0&0&0&0&0&1&0 \end{pmatrix} \times \begin{pmatrix} 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&1\\ 0&0&0&0&0&0&1&0 \end{pmatrix} = \begin{pmatrix} 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&1 \end{pmatrix} = I = T^{\dagger}T$

We find that for the matrix of Toffoli gate, $TT^{\dagger} = T^{\dagger}T = I$ (identity matrix) and it is reversible operation.

We take another example. For Peres gate, the matrix is:

, say $P = \begin{pmatrix} 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&1\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&1&0&0&0 \end{pmatrix}$; Then $P^{\dagger} = \begin{pmatrix} 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&0&0&0&1\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0 \end{pmatrix}$

Now, $PP^{\dagger} = \begin{pmatrix} 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&1\\ 0&0&0&0&0&0&1&0 \end{pmatrix} \times \begin{pmatrix} 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&1\\ 0&0&0&0&0&0&1&0 \end{pmatrix} = \begin{pmatrix} 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&1 \end{pmatrix}$

We find that for the matrix of Peres gate, $PP^{\dagger} = P^{\dagger}P = I$ (identity matrix) and it is reversible operation.

It has also been observed that $T^{\dagger}$ equals T, where as $P^{\dagger}$ does not equal P. however, both give identity matrix when multiplied by the adjoint [2].

The matrices which are equal to their own adjoint, are called Hermitian matrices. Hermitian matrices have a special feature – they are equal to their own inverse. This has an interesting consequence, when one such circuit is placed in series (one just after the other), it gives back the original input.

While unitary operation has all the information of the input, putting two such circuits in sequence may not give back the input – for Peres gate, we need to connect four in sequence to get back the input.

However, unitary operation satisfying Hermitian property would give back the input when two such circuits are placed in series, Toffoli gate is an example of a unitary matrix with Hermitian property.

2.2.2 General observation on matrices and reversibility

- Transformation / operation can be represented by a matrix of size (2^n by 2^n) [7] [2]
- The operation U on the variables (Input matrix) is a matrix multiplication: Input matrix is multiplied by U (U is acting as an operator)

- If the operation is represented through a matrix U and U is a unitary matrix, then $UU^{\dagger} = U^{\dagger}U = I$ (identity matrix) and the operation is reversible, $U^{\dagger}$ being the adjoint (complex conjugate transpose) of the unitary matrix U
- If the adjoint of a unitary matrix U equals itself ($U^{\dagger} = U$), then it is called a Hermitian matrix.
- For a unitary matrix which is also Hermitian, ($UU^{\dagger} = U^{\dagger}U = I$) and ($UU = I$), and this means $U = U^{-1}$. A Hermitian matrix then, it is its own inverse. In this case, two such circuits in series will result in an operation as $UU = UU^{-1} = I$ which is an identity operation – no change! It gives back the input. Toffoli gate is an example of that. Two Toffoli gates in series will give the original inputs back.
- If the matrix is unitary but not Hermitian, then also the operation is reversible, the information on original input is carried along, however, it will require more than two such circuits in series to get back the inputs. Peres gate is an example of that – four such circuits in series will give the input back

2.3 Cycles and groups

Consider the simple truth table in the next figure.

No.	A	B	C		P	Q	R	No.
0	0	0	0		0	1	0	2
1	0	0	1		1	0	1	5
2	0	1	0		1	1	0	6
3	0	1	1	→	0	0	0	0
4	1	0	0		0	1	1	3
5	1	0	1		0	0	1	1
6	1	1	0		1	1	1	7
7	1	1	1		1	0	0	4

Table 14: A simple truth table for a 3 input 3 output circuit

We see that the input vectors span from 0 to 7 and so do the output vectors. This yields one-to-one and onto nature that is bijective and the circuit for this will be reversible, because from each output, we can reach the particular input vector which generates the output vector. This happens for n input – n output circuit when input vectors span {0, 1, … $2^n - 1$} and the output vectors also span {0, 1, … $2^n - 1$} (all numbers exist in both input side and output side). Input side is sequential and sequence may be different in the output side. The set of output vectors or the permutations on the finite set {0, 1, … $2^n - 1$} forms a symmetric group [1] [7] [2].

We can represent this 3 input truth table in the form (0, 1, 2, 3, 4, 5, 6, 7) → (2, 5, 6, 0, 3, 1, 7, 4).
Another such function could be (0, 1, 2, 3, 4, 5, 6, 7) → (2, 1, 0, 7, 6, 3, 4, 5).

Since the left side is same (0 to 2^n - 1 in sequence) in all such expressions, each function can be shown with the right hand side part only. Hence the first function can be shown as (2, 5, 6, 0, 3, 1, 7, 4) and the second function can be shown as (2, 1, 0, 7, 6, 3, 4, 5).

The function say h = (2, 5, 6, 0, 3, 1, 7, 4), this is a permutation spanning {0, 1, … 7}. Here, h(0)=2, h(1)=5, h(3)=6 and so on. This is a permutation. With A as a finite set and set of permutations on A = {0, 1, … 2^n-1} form a symmetric group on A [1] [2].

A cycle for a function f in the context of reversible circuit is expressed as terms like (a_1, a_2, … a_k) which is a permutation with the property that $f(a_1) = a_2$, $f(a_2) = a_3$, … and $f(a_k) = a_1$. In a way if one moves along the terms as per the function, one would repeat the sequence [1].

The function g = (2, 5, 6, 0, 3, 1, 7, 4) can be written as (0, 2, 6, 7, 4, 3) (1, 5). The first part here is a 6 cycle with 6 members and the second part is a 2 cycle with 2 members.

The 2 cycle parts are called transpositions, because these get transposed with each occurrence.

The function g = (2, 1, 0, 7, 6, 3, 4, 5) is equivalent to (0, 2)(1)(3, 7, 5)(4, 6). The 2 cycles are called transpositions [1]. The 1 cycles, being obvious (output=input), are normally omitted in the display, and it is shown as (0, 2)(3, 7, 5)(4, 6).

Observing the function g = (2, 1, 0, 7, 6, 3, 4, 5) being equivalent to (0, 2)(3, 7, 5)(4, 6), we note the following [1]:

- There are two 2 cycles (0, 2) and (4, 6). For 2 cycles, the output repeats on occurrence of the circuit 2 times
- There is one 3 cycle (3, 7, 5). For 3 cycles, the output repeats on occurrence of the circuit 3 times
- There is one 1 cycle (1). For 1 cycles, the output repeats on every occurrence of the circuit
- On occurrence of the circuit LCM number of times, the circuit will be reversible. For g = (2, 1, 0, 7, 6, 3, 4, 5) the cycles are 1, 2, 2 and 3. The LCM of 1, 2 and 3 is 6. After 6 occurrences of the circuit, the input is received back. The sequence (with 0, 1, 2, 3, 4, 5, 6, 7) would be as follows [1]:
    - Occurrence 1: (2, 1, 0, 7, 6, 3, 4, 5)
    - Occurrence 2: (0, 1, 2, 5, 4, 7, 6, 3)
    - Occurrence 3: (2, 1, 0, 3, 6, 5, 4, 7)
    - Occurrence 4: (0, 1, 2, 7, 4, 3, 6, 5)
    - Occurrence 5: (2, 1, 0, 5, 6, 7, 4, 3)
    - Occurrence 6: (0, 1, 2, 3, 4, 5, 6, 7)

We take another example with g = (1, 0, 2, 3, 4, 7, 6, 5). We observe that g is equivalent to (1, 0)(7, 5).

- There are two 2 cycles and four 1 cycles.
- LCM of 1 and 2 is 2
- After 2 occurrences of the circuit, we should get back input. The sequence with (with 0, 1, 2, 3, 4, 5, 6, 7) is:
    - Occurrence 1: (1, 0, 2, 3, 4, 7, 6, 5)
    - Occurrence 2: (0, 1, 2, 3, 4, 5, 6, 7) [1]

2.4 Reversibility with Boolean function, matrices and cycles

It has been noticed that a Boolean function can be checked for reversibility. When we have n inputs, we get $2^n - 1$ input combinations. We have same number of bits in the output and there also, we get $2^n - 1$ combinations. All entries must appear in the truth table of output – then we get the exact input combination for a particular output combination. If there is any entry missing or any entry appears more than once, we lose reversibility. We would analyze the reversibility status of the function depicted in the next truth table.

A	B	C		P	Q	R
0	0	0		0	0	0
0	0	1		0	1	1
0	1	0		1	0	0
0	1	1	→	1	0	1
1	0	0		0	1	0
1	0	1		1	1	0
1	1	0		0	0	1
1	1	1		1	1	1

Table 15: Truth table of a function

2.4.1 Reversibility with Boolean function

The above function is equivalent to (0, 1, 2, 3, 4, 5, 6, 7) → (0, 3, 4, 5, 2, 6, 1, 7).
It is observed that input output relation is both injective and surjective (which is a bijective relation). The circuit is reversible [1]. The output covers all permutation of three bits (P, Q, R). The next figure depicts the reversible function.

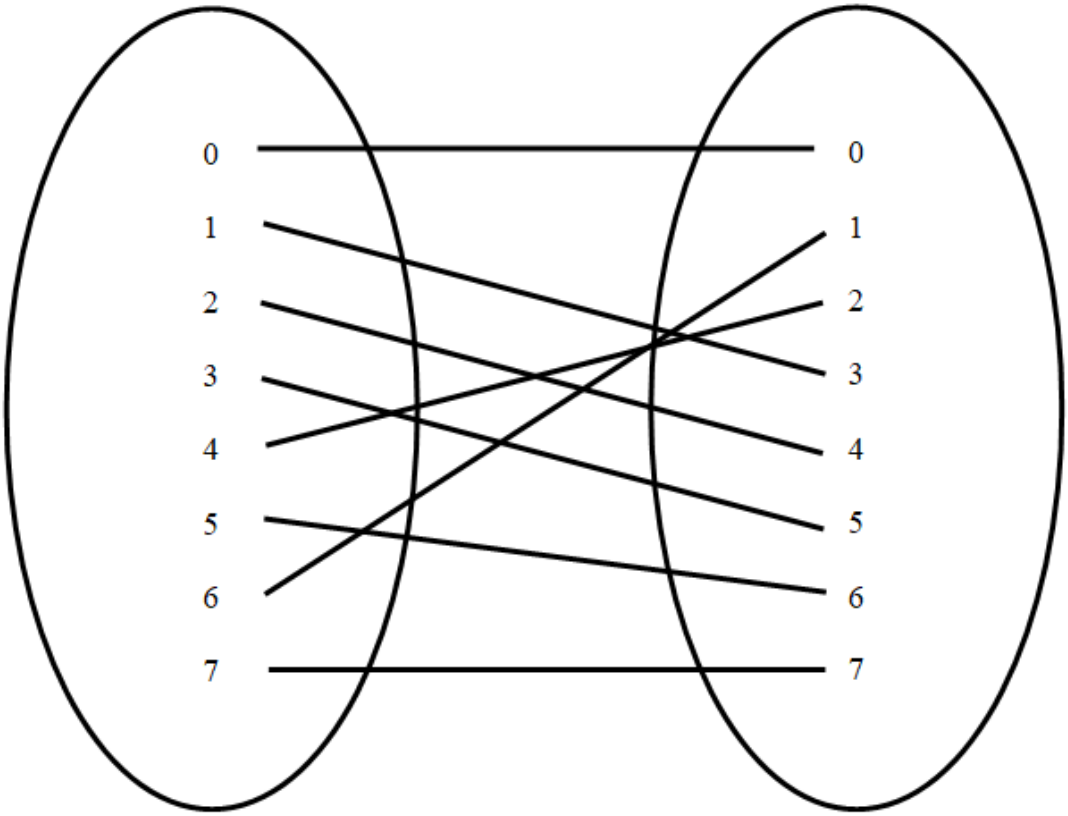


Fig 3: Input output permutation (bijective)

2.4.2 Reversibility with operator matrices

From the matrix point of view, we can express a function as an operator working on an input vector, producing an output. The operator is a matrix.

Let us consider the operator matrix M_3 working on the above input matrix ABC. We get the output as PQR matrix.

$$M_3 \times ABC = \begin{pmatrix} 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&0&1 \\ 0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&1&0&0&0&0 \end{pmatrix} \times \begin{pmatrix} 0&0&0 \\ 0&0&1 \\ 0&1&0 \\ 0&1&1 \\ 1&0&0 \\ 1&0&1 \\ 1&1&0 \\ 1&1&1 \end{pmatrix} = \begin{pmatrix} 0&0&0 \\ 0&0&1 \\ 0&1&0 \\ 1&1&1 \\ 1&0&0 \\ 1&0&1 \\ 1&1&0 \\ 0&1&1 \end{pmatrix} = PQR$$

It is found that adjoint of $M_3 = M_3^\dagger = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$ (it is same as M_3)

$$M_3 \times M_3^\dagger = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} = I$$

Therefore, $M_3 \times M_3^\dagger = I$. Hence, M_3 is a unitary matrix.
Also, $M_3^\dagger$ is seen to be equal to M_3. M_3 is Unitary and Hermitian, hence M_3 is inverse of itself. When M_3 operates on an input matrix twice in sequence, we get back the original input.
M_3 is representing a reversible operation.

Let us now consider the matrix M_2 working on working on a 2 input matrix AB, producing matrix PQ.

$$M_2 \times AB = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 1 \\ 1 & 1 \end{pmatrix} = PQ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad 2.4.1$$

Observation on equation 2.4.1 reveals that:
- The first column of the output = P = A
- The second column of the output is = Q = A + B

It is found that adjoint of $M_2 = M_2^\dagger = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$, $M_2 \times M_2^\dagger = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

We see that $M_2 \times M_2^\dagger$ is not an identity matrix. M_2 does not represent a unitary operation.

Hence, OR gate is not a reversible gate [2].

2.4.3 Reversibility with groups / cycles

The function is (0, 1, 2, 3, 4, 5, 6, 7) → (0, 1, 2, 7, 4, 5, 6, 3) and equivalent to (7, 3) because all others are 1 cycle cases. There is a permutation of input at the output. The maximum cycle length is 2, so two such circuits will give us our input back.

- LCM of 2 and 1 is 2. After 2 occurrences of the circuit, we should get back input. The sequence with (with 0, 1, 2, 3, 4, 5, 6, 7) is:
    - Occurrence 1: (0, 1, 2, 7, 4, 5, 6, 3)
    - Occurrence 2: (0, 1, 2, 3, 4, 5, 6, 7)

Chapter 3 Quantum Computing – the paradigm

Quantum computing happens to be a vast emerging field of research with a potential to disrupt many of the computing processes which are in use today. With quantum computing in place, some of the existing processes may be strengthened while some of them may be rendered useless. We would discuss some basic introductory areas of quantum computing to have the perspective in place for reversible circuits.

A small part on qubit will be taken up in this chapter. A qubit in quantum computing is in some way equivalent to a bit in classical computing [8] [7] [2].

3.1 Classical computing paradigm

Let us consider the universe of classical computing. We are aware that in classical computing, we have a bit as the basic unit of computing. The state of a bit can be either 0 or 1. The state cannot be a combination of 0 and 1. They exist separately, not like some part of a state is 0 and another part of the same state is 1. They can be measured with certainty. A circuit will maintain the input – output relationship in a consistent manner. The individual components, the parts thereof, the environment and the methods are all conforming to the laws of classical physics.

All our computing components (discrete components and transistors → digital components with binary bits → gates → integrated circuits → computing units / computers with von-Neumann stored program architecture) are based on this approach.

3.2 Quantum computing paradigm

In quantum computing, the basic computing unit (like a bit in classical computing) is a qubit [2] [7].

The state of the object is largely the configuration under consideration. As an example, for a coin, the two sides (head and tail) are the states, the possibilities we may have as the outcome of tossing the coin. The coin (or for that matter, any object in the classical world) can be in one of the possible states at any point of time. In quantum domain, however, the state can be a superposition of the possibilities.

The state of an arbitrary qubit can be represented as [8] [7] [2]:
$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$ 3.2.1

The above expression comes with a few conditions, these are [8] [7]:

$\alpha, \beta \in \mathbb{C}$ ($\mathbb{C}$ is the set of all complex numbers) 3.2.2
$|\alpha|^2 + |\beta|^2 = 1$ 3.2.3

Here, with $|\alpha|^2 + |\beta|^2 = 1$, normalization is achieved. It is, in a way, total probability (sum of $|\alpha|^2$ and $|\beta|^2$) being 1 for the expected outcomes.

In equation 3.2.1, states $|\psi\rangle$, $|0\rangle$ and $|1\rangle$ are in Ket notation. Bra-Ket notation (in line of a bracket enclosing items) was introduced by Paul Dirac. The Bra (written as $\langle\varphi|$, for example) is a row vector and Ket (written as $|\psi\rangle$ for example) is a column vector.

In equation 3.2.1, the state is shown to be a superposition of Ket 0 ($|0\rangle$) and Ket 1 ($|1\rangle$). The state of the qubit is a linear combination of the basis states, with conditions 3.2.2 and 3.2.3 followed. Ket 0 ($|1\rangle$) and Ket 1 ($|1\rangle$) are the basis states for the single qubit system. In addition to expression of superposition of states, the basis states will individually provide the measurement of states (discussed later in this chapter. These are the building blocks for more complex qubits (like $|00\rangle$ or $|01\rangle$ etc. for example).

The state of a qubit can also be expressed through 2 dimensional complex unit vector satisfying the required properties 3.2.2 and 3.2.3.

The vector representation of the states $|0\rangle$ and $|1\rangle$ are as follows [8] [7] [2]:

$|0\rangle \equiv \begin{pmatrix}1\\0\end{pmatrix}$ and $|1\rangle \equiv \begin{pmatrix}0\\1\end{pmatrix}$ ……3.2.4

Now, based on expression 3.2.1, we have the following expressions:

$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \equiv \alpha\begin{pmatrix}1\\0\end{pmatrix} + \beta\begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix}\alpha\\0\end{pmatrix} + \begin{pmatrix}0\\\beta\end{pmatrix} = \begin{pmatrix}\alpha\\\beta\end{pmatrix}$ ……3.2.5

So, we have $|\psi\rangle = \begin{pmatrix}\alpha\\\beta\end{pmatrix}$, where $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$ ……3.2.6

The operators / gates working on the vectors represented with the notation as above will cause evolution / transformation of the vectors. These operators are normally unitary matrices and on operation on the vectors transformation occurs on the input resulting in transformed vectors.

3.2.1 Operations on a qubit

When a qubit is passed through a quantum gate, it is transformed and it results in a new state.

These transformations / evolutions in quantum domain are reversible. In some cases of evolution, the new state is same as the old one (identity operation). The transformed state (through quantum gates) has enough information to go back to the old state [8] [2].

If it is looked at from matrix point of view, the gate is working like a matrix, the matrix can be thought of as an operator which works on the input and creates an output matrix. To enable the reversible operation, the matrix must be unitary. We assume M is a unitary matrix that works on input matrix ABC and creates an output matrix PQR. Hence: M x ABC = PQR and M x $M^\dagger$ = $M^\dagger$ x M = I (identity matrix) [8] [7].

We can take a closer look at the evolution of a qubit with the following examples (of X Gate and Hadamard Gate).

3.2.1.1 X gate or Pauli-X gate:

In classical computing, we have the inverter (NOT gate). X gate or Pauli-X gate is the equivalent of NOT gate in the quantum computing domain [7].

The matrix is of the form: $X \equiv \begin{pmatrix}0 & 1\\1 & 0\end{pmatrix}$.

When the state $|\psi\rangle = \begin{pmatrix}\alpha\\\beta\end{pmatrix}$ (ref equation 3.2.5 and 3.2.6) is passed through X gate, the following happens:

Output = (X gate) x ($|\psi\rangle$) = $\begin{pmatrix}0 & 1\\1 & 0\end{pmatrix} \times \begin{pmatrix}\alpha\\\beta\end{pmatrix} = \begin{pmatrix}\beta\\\alpha\end{pmatrix}$

For $|0\rangle$, the output = $\begin{pmatrix}0 & 1\\1 & 0\end{pmatrix} \times |0\rangle = \begin{pmatrix}0 & 1\\1 & 0\end{pmatrix} \times \begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}0\\1\end{pmatrix} = |1\rangle$

For $|1\rangle$, the output = $\begin{pmatrix}0 & 1\\1 & 0\end{pmatrix} \times |1\rangle = \begin{pmatrix}0 & 1\\1 & 0\end{pmatrix} \times \begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix}1\\0\end{pmatrix} = |0\rangle$

3.2.1.2 Hadamard Gate

Hadamard gate converts a clustered state to a superposed state with equal proportions. This gate works in the quantum computing domain [7] [2].

The matrix for Hadamard gate is: $H \equiv \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$.

When the state $|\psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$ (ref equation 3.2.5 and 3.2.6) is passed through H gate, the following happens:

Output = (H gate) x ($|\psi\rangle$) = $\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ x $\begin{pmatrix} \alpha \\ \beta \end{pmatrix}$ = $\frac{1}{\sqrt{2}} \begin{pmatrix} \alpha + \beta \\ \alpha - \beta \end{pmatrix}$

3.2.2 Measurement of a qubit

When a qubit is measured, the square of the absolute value of the coefficient / amplitude of a basis state $|x\rangle$ is the probability of getting $|x\rangle$. Hence, a state expressed as $|\psi\rangle = \alpha |0\rangle + \beta |1\rangle$, when measured, we would get $|0\rangle$ with probability of $|\alpha|^2$ and $|1\rangle$ with probability of $|\beta|^2$ [8].

We consider, for example, the state $\psi\rangle = \frac{1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} |1\rangle$. Here, the basis states are $|0\rangle$ and $|1\rangle$.

Hence, when measured, the result will be either $|0\rangle$ with probability $\frac{1}{\sqrt{2}}$ x $\frac{1}{\sqrt{2}} = \frac{1}{2}$, or $|1\rangle$ with probability $\frac{1}{\sqrt{2}}$ x $\frac{1}{\sqrt{2}} = \frac{1}{2}$.

One way to look at this measurement is that if n number of such states are measured, then for approximately $\frac{n}{2}$ number of such states [7], the result will be $|0\rangle$ and for approximately $\frac{n}{2}$ number of such states, the result will be $|1\rangle$.

Measurement in quantum computing domain is essentially an observation on the qubit. This is also a matrix operation. However, the measurement operation is not a reversible operation (matrix is not unitary). On measurement, the state collapses to one of the basis states, and is left there. Hence, after an observation is executed, we cannot go back to the original state of the qubit and the information on the state is lost.

3.2.3 Computing with quantum circuits

The quantum computing happens with quantum circuits. These circuits are assembled with different reversible components. A quantum algorithm is normally implemented to harness the potential of superposition [7] and other inherent properties of quantum world.

Chapter 4 Reversible logic gates

In this section, we will consider the some of the widely used reversible logic gates and the details thereof. These gates are assembled to build the reversible circuits.

4.1 NOT gate

A	$P = \overline{A}$
0	1
1	0

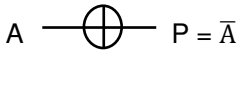


Fig 4: Truth table of NOT gate with symbol

It can be noted from the truth table that the transformation is bijective, hence NOT function is reversible. If the output is 1, we know that the corresponding input is 0 and vice versa [17] [15].

4.2 CNOT gate

CNOT gate (also known as Feynman gate, after Richard Feynman) has the input vector with two inputs (A, B) [11]. It has output vector with two outputs (P, Q). It derives its name from the phrase 'Controlled NOT gate' [1] [15].

The outputs are [23] [11]:

- P = A
- Q = A ⊕ B

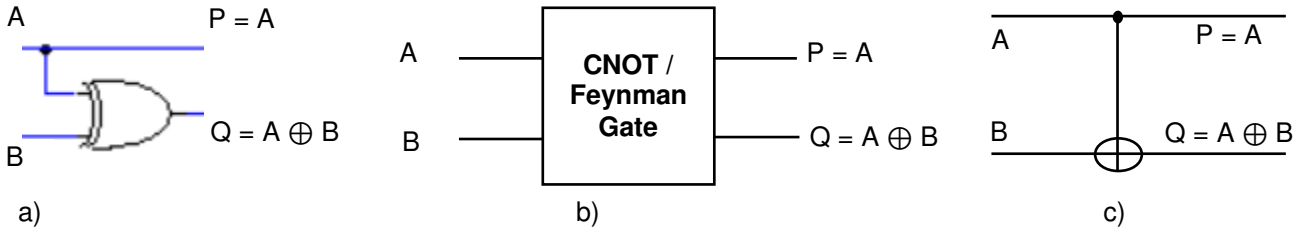


Fig 5: CNOT gate and the equivalent circuit

The truth table of CNOT gate is in the next figure [17].

A	B	P = A	Q = A ⊕ B	Matrix M_{CNOT}	CNOT gate cycles
0	0	0	0	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$	(2, 3)
0	1	0	1		
1	0	1	1		
1	1	1	0		

Table 16: The truth table, matrix and cycles for CNOT gate

So, when A = 0, Q = B and when A = 1, Q = $\overline{B}$, so A is controlling if Q would be B or $\overline{B}$. It is acting as a controlled inverter. Hence the name 'Controlled NOT gate' or CX gate (X indicates inverter). Equivalently, the NOT is applied when control line is active, the input is passed through when the control line is not active and the control line is passed to the output unchanged (additional input for control, hence another output).

The implementation is as in the figure. One output P is equal to the input A and the other output is logical XOR of the two inputs A and B.

In the next figure, a) is the logical equivalent of CNOT gate in the classical circuit, b) is the block diagram and c) is the CNOT gate symbol in the quantum circuit.

The main usage of CNOT gate is to copy a bit / qubit and implementing a fan out (a direct fan out is not allowed in reversible circuit) along with use of a NOT gate activated by a control signal (CX) gate [1].

From the matrix point of view,

$M_{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$, hence, $M_{CNOT}^{\dagger} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$

$M_{CNOT}^{\dagger}$ is the adjoint of M_{CNOT} and in this case, equal to M_{CNOT}.

$M_{CNOT} \times M_{CNOT}^{\dagger} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = I$

We have, $M_{CNOT} \times M_{CNOT}^{\dagger} = I$ and also $M_{CNOT}^{\dagger} = M_{CNOT}$, so it is a Hermitian matrix (inverse of itself) – two CNOT gates in series would return the input.

Checking two gates in series: $M_{CNOT} \times M_{CNOT} \times \begin{pmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{pmatrix}$

$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \times \begin{pmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \times \begin{pmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{pmatrix}$, it gives the input back.

Observation:

- Truth table is bijective
- Each column of the output is balanced
- $M_{CNOT} \times M_{CNOT}^{\dagger} = I$
- $M_{CNOT}^{\dagger} = M_{CNOT}$, so Hermitian, → inverse of itself, so two such gates in series would give the original input back
- Cycle representation is (2, 3). The transformation with two CNOT gates will be:
    - (0, 1, 2, 3) → (0, 1, 3, 2)
    - (0, 1, 3, 2) → (0, 1, 2, 3)

4.3 Toffoli gate

Toffoli gate has the input vector with three inputs (A, B, C). It has output vector with three outputs (P, Q, R). It is also called CCX / CCN (Controlled Controlled NOT) gate or C2N gate or C2X gate [11] [4] [1] [23].

The outputs are [15]:

- P = A
- Q = B
- R = AB ⊕ C

Refer to the figure.

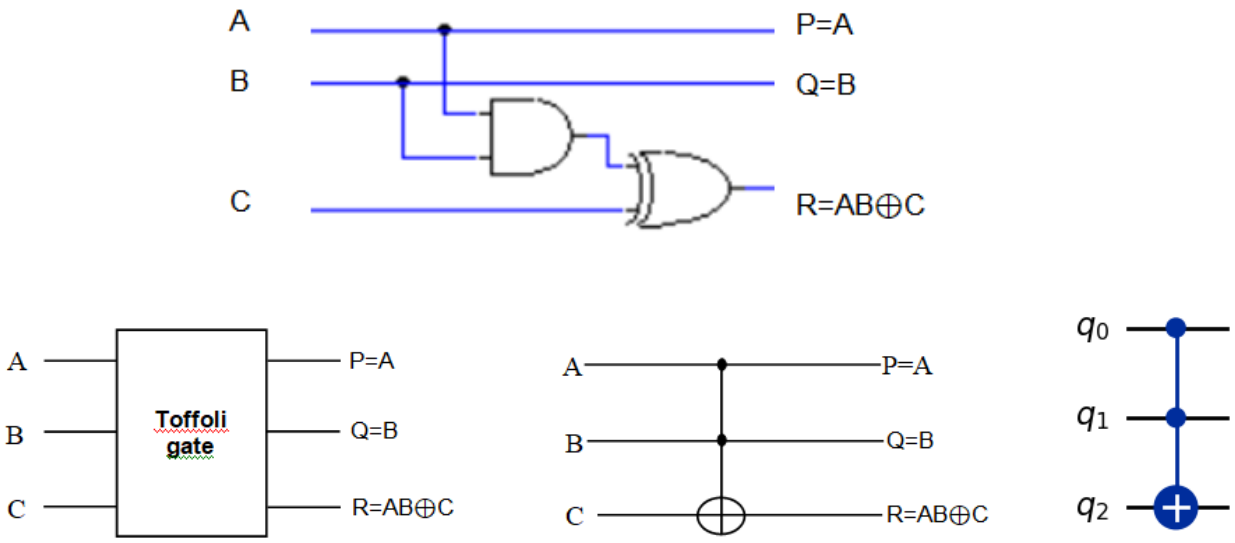


Fig 6: Toffoli gate – classical equivalent, block diagram and quantum circuit

A	B	C	P=A	Q=B	R=AB⊕C	Matrix $M_{Toffoli}$	Cycles
0	0	0	0	0	0	$\begin{pmatrix} 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&1&0 \end{pmatrix}$	(6, 7)
0	0	1	0	0	1		
0	1	0	0	1	0		
0	1	1	0	1	1		
1	0	0	1	0	0		
1	0	1	1	0	1		
1	1	0	1	1	1		
1	1	1	1	1	0		

Table 17: The truth table, matrix and cycles for Toffoli gate

We have seen that for CNOT gate, input A was the control line, (hence CNOT or CX) which determines whether the inverter for B will be activated (A=1) or for B, it will just be a pass through (A=0) [21].

Similarly, for Toffoli gate (CCNOT / CCX), AB is the control line [17].

- If AB=0, R = C
- If AB=1, R = $\bar{C}$

Now, $M_{Toffoli} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$, hence, $M_{Toffoli}^{\dagger} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$ (it is same as $M_{Toffoli}$)

$$M_{Toffoli} \times M_{Toffoli}^{\dagger} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} = I$$

So it is a unitary and Hermitian Matrix. It is reversible and two in series would get us the input vector. Toffoli gate is widely used in realization of reversible circuits.

Checking two Toffoli gates in series:

$$M_{Toffoli} \times M_{Toffoli} \times \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \times \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \times \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} = \text{original input vector}$$

Observation:

- Truth table is bijective [1]
- Each column of the output is balanced
- $M_{Toffoli} \times M_{Toffoli}^{\dagger} = I$
- $M_{Toffoli}^{\dagger} = M_{Toffoli}$, so Hermitian, → inverse of itself, so two such gates in series would give the original input back
- Cycle representation is (6, 7). The transformation with two Toffoli gates will be [1]:
    - (0, 1, 2, 3, 4, 5, 6, 7) → (0, 1, 2, 3, 4, 5, 7, 6)
    - (0, 1, 2, 3, 4, 5, 7, 6) → (0, 1, 2, 3, 4, 5, 6, 7)

4.4 Fredkin gate

Fredkin gate has the input vector with three inputs (A, B, C). It has output vector with three outputs (P, Q, R) [1]. It is also called Controlled SWAP gate / CSWAP gate also [15]. The outputs are [1] [11]:

- P = A
- Q = $\bar{A}B \oplus AC$
- R = $\bar{A}C \oplus AB$

Refer to the figure.

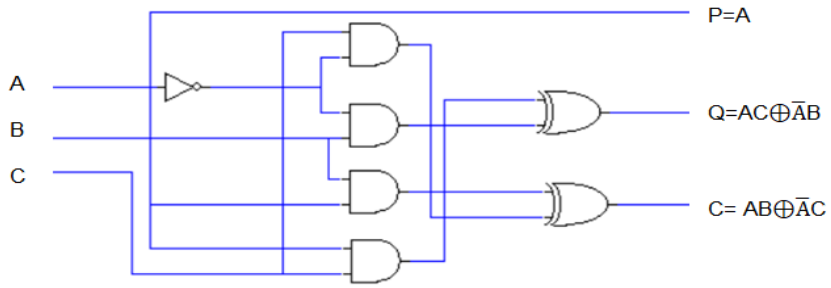


Fig 7: Fredkin gate – Classical equivalent circuit

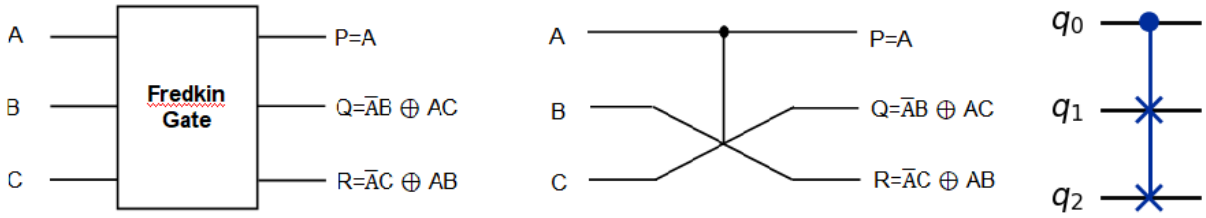


Fig 8: Fredkin gate – Block diagram and quantum circuit

A	B	C	P=A	Q=$\bar{A}B \oplus AC$	R=$\bar{A}C \oplus AB$	Matrix $M_{Fredkin}$	Cycles
0	0	0	0	0	0	$\begin{pmatrix} 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&1 \end{pmatrix}$	(5, 6)
0	0	1	0	0	1		
0	1	0	0	1	0		
0	1	1	0	1	1		
1	0	0	1	0	0		
1	0	1	1	1	0		
1	1	0	1	0	1		
1	1	1	1	1	1		

Table 18: The truth table, matrix and cycles for Fredkin gate

For Fredkin gate (Controlled Swap or CSwap), A is the control line [17].

- When A = 0, Q = B and R = C (passthrough)
- When A = 1, then the output is swapped, Q = C and R = B

Now, $M_{Fredkin} = \begin{pmatrix} 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&1 \end{pmatrix}$, hence, $M_{Fredkin}^{\dagger} = \begin{pmatrix} 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&1 \end{pmatrix}$ (it is same as $M_{Fredkin}$)

$$M_{Fredkin} \times M_{Fredkin}^{\dagger} = \begin{pmatrix} 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&1 \end{pmatrix} \times \begin{pmatrix} 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&1 \end{pmatrix} = \begin{pmatrix} 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&1 \end{pmatrix} = I$$

So it is a unitary and Hermitian Matrix. It is reversible and two in series would get us the input vector. Fredkin gate is widely used in realization of reversible circuits.

Checking two Fredkin gates in series:

$$M_{Fredkin} \times M_{Fredkin} \times \begin{pmatrix} 0&0&0\\ 0&0&1\\ 0&1&0\\ 0&1&1\\ 1&0&0\\ 1&0&1\\ 1&1&0\\ 1&1&1 \end{pmatrix} = \begin{pmatrix} 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&1 \end{pmatrix} \times \begin{pmatrix} 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&1 \end{pmatrix} \times \begin{pmatrix} 0&0&0\\ 0&0&1\\ 0&1&0\\ 0&1&1\\ 1&0&0\\ 1&0&1\\ 1&1&0\\ 1&1&1 \end{pmatrix}$$

$$= \begin{pmatrix} 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&1 \end{pmatrix} \times \begin{pmatrix} 0&0&0\\ 0&0&1\\ 0&1&0\\ 0&1&1\\ 1&0&0\\ 1&1&0\\ 1&0&1\\ 1&1&1 \end{pmatrix} = \begin{pmatrix} 0&0&0\\ 0&0&1\\ 0&1&0\\ 0&1&1\\ 1&0&0\\ 1&0&1\\ 1&1&0\\ 1&1&1 \end{pmatrix} = \text{original input vector}$$

Observation:
- Truth table is bijective
- Each column of the output is balanced
- $M_{Fredkin} \times M_{Fredkin}^{\dagger} = I$
- $M_{Fredkin}^{\dagger} = M_{Fredkin}$, so Hermitian, → inverse of itself, so two such gates in series would give the original input back
- Cycle representation is (6, 7). The transformation with two Fredkin gates will be [1]:
    - (0, 1, 2, 3, 4, 5, 6, 7) → (0, 1, 2, 3, 4, 6, 5, 7)
    - (0, 1, 2, 3, 4, 6, 5, 7) → (0, 1, 2, 3, 4, 5, 6, 7)

4.5 Peres gate

Peres gate has the input vector with three inputs (A, B, C). It has output vector with three outputs (P, Q, R) [1] [11] [15].

- P = A
- Q = A ⊕ B
- R = AB ⊕ C

Refer to the figure.

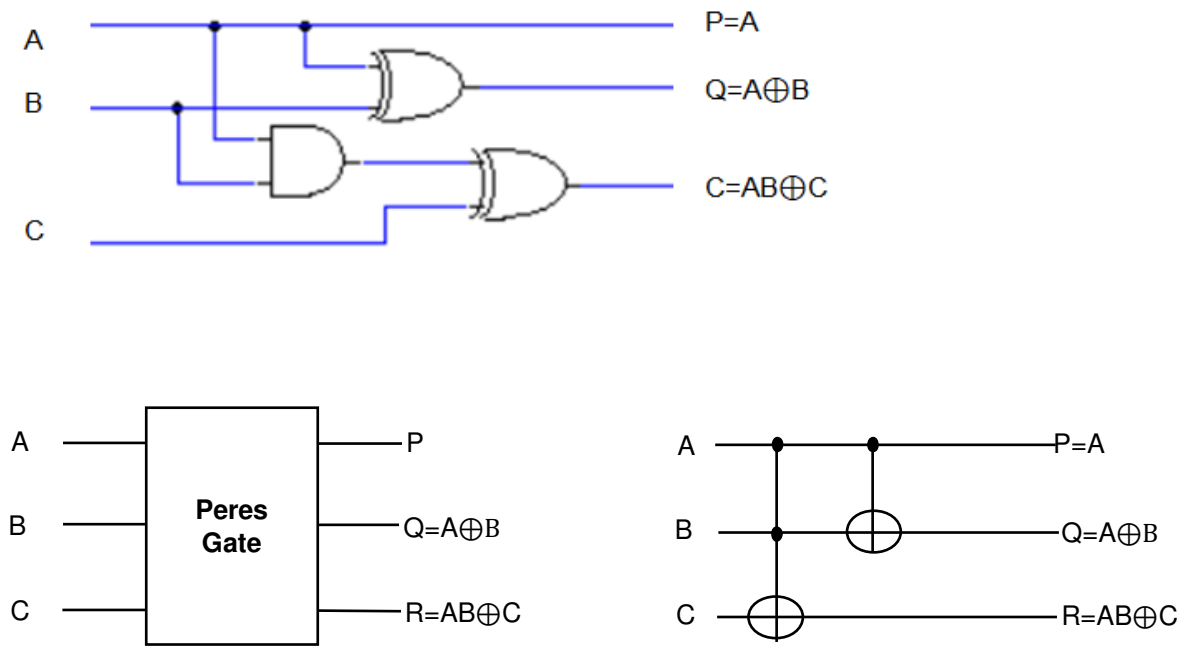


Fig 9: Peres gate – Classical equivalent, block diagram and quantum circuit

A	B	C	P=A	Q=A ⊕ B	R=AB ⊕ C	Matrix M_{Peres}	Cycles
0	0	0	0	0	0	$\begin{pmatrix} 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&1&0&0&0 \end{pmatrix}$	(6, 5, 7, 4)
0	0	1	0	0	1		
0	1	0	0	1	0		
0	1	1	0	1	1		
1	0	0	1	0	0		
1	0	1	1	1	0		
1	1	0	1	0	1		
1	1	1	1	1	1		

Fig 10: Peres gate – Classical equivalent, block diagram and quantum circuit

Now, $M_{Peres} = \begin{pmatrix} 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&1&0&0&0 \end{pmatrix}$, hence, $M_{Peres}^{\dagger} = \begin{pmatrix} 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&1&0&0 \end{pmatrix}$ (it is not same as M_{Peres})

$$M_{Peres} \times M_{Peres}^{\dagger} = \begin{pmatrix} 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&1&0&0&0 \end{pmatrix} \times \begin{pmatrix} 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&1&0&0 \end{pmatrix} = \begin{pmatrix} 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&1 \end{pmatrix} = I$$

So it is a unitary matrix. The operation is reversible. However, M_{Peres} is not having the Hermitian property. Hence two Peres gates in series will not get us the input vector.

Looking at the cycle representation, the cycle lengths are 1, 1, 1, 1, and 4. LCM of 1, 1, 1, 1, and 4 is 4. Hence, 4 Peres gates in series would give us back the original input.

Peres gate causes a transformation where the input information is carried forward and after the fourth cycle, the input is received back. The following check confirms this.

We will now check what happens to the input when we connect four Peres gates in series.

$$M_{Peres} \times M_{Peres} \times M_{Peres} \times M_{Peres} \times \begin{pmatrix} 0&0&0 \\ 0&0&1 \\ 0&1&0 \\ 0&1&1 \\ 1&0&0 \\ 1&0&1 \\ 1&1&0 \\ 1&1&1 \end{pmatrix} = M_{Peres} \times M_{Peres} \times M_{Peres} \times \begin{pmatrix} 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&1&0&0&0 \end{pmatrix} \times \begin{pmatrix} 0&0&0 \\ 0&0&1 \\ 0&1&0 \\ 0&1&1 \\ 1&0&0 \\ 1&0&1 \\ 1&1&0 \\ 1&1&1 \end{pmatrix}$$

$$= M_{Peres} \times M_{Peres} \times M_{Peres} \times \begin{pmatrix} 0&0&0 \\ 0&0&1 \\ 0&1&0 \\ 0&1&1 \\ 1&1&0 \\ 1&1&1 \\ 1&0&1 \\ 1&0&0 \end{pmatrix} = M_{Peres} \times M_{Peres} \times \begin{pmatrix} 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&1&0&0&0 \end{pmatrix} \times \begin{pmatrix} 0&0&0 \\ 0&0&1 \\ 0&1&0 \\ 0&1&1 \\ 1&1&0 \\ 1&1&1 \\ 1&0&1 \\ 1&0&0 \end{pmatrix}$$

$= M_{Peres} \times M_{Peres} \times \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix} = M_{Peres} \times \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \times \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix}$

$= M_{Peres} \times \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \times \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}$ = our original input

Observation:

- Truth table is bijective [1]
- Each column of the output is balanced
- $M_{Peres} \times M_{Peres}^{\dagger} = I$, hence unitary
- $M_{Peres} \times M_{Peres} \times M_{Peres} \times M_{Peres} = I$, four Peres gates in series gives back the original input
- Cycle representation is (4, 6, 5, 7). The transformation with four Peres gates will be [1]:
    - (0, 1, 2, 3, 4, 5, 6, 7) → (0, 1, 2, 3, 6, 7, 5, 4)
    - (0, 1, 2, 3, 6, 7, 5, 4) → (0, 1, 2, 3, 5, 4, 7, 6)
    - (0, 1, 2, 3, 5, 4, 7, 6) → (0, 1, 2, 3, 7, 6, 4, 5)
    - (0, 1, 2, 3, 7, 6, 4, 5) → (0, 1, 2, 3, 4, 5, 6, 7)

4.6 Hadamard gate

As indicated earlier, Hadamard gate converts a clustered state to a superposed state with equal proportions. This gate works in the quantum computing domain [7] [17].

The matrix for Hadamard gate is: $H \equiv \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$.

When the state $|\psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$ (ref equation 3.2.5 and 3.2.6) is passed through H gate, the following happens:

Output = (H gate) x ($|\psi\rangle$) = $\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \times \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} \alpha + \beta \\ \alpha - \beta \end{pmatrix}$

For $|0\rangle$, the output = $\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \times |0\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \times \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$

For $|0\rangle$, the output = $\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \times |1\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \times \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix}$

It can be observed that when Hadamard gate operates on a qubit:

- For arbitrary state $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$, F output = $\frac{1}{\sqrt{2}}\begin{pmatrix}\alpha+\beta\\ \alpha-\beta\end{pmatrix}$
- When the input is $|0\rangle$, output becomes $\frac{1}{\sqrt{2}}\begin{pmatrix}1\\ 1\end{pmatrix}$
- When the input is $|1\rangle$, output becomes $\frac{1}{\sqrt{2}}\begin{pmatrix}1\\ -1\end{pmatrix}$

4.7 Pauli operator gates

This set of gates is widely used in quantum circuits. The discussion mainly includes Pauli X gate, Pauli Y gate and Pauli Z gate. Some of the derivatives of X gate are also taken up (like V, $V^{\dagger}$, CV, $CV^{\dagger}$ gates) [7] [17].

4.7.1 Pauli X gate

As indicated earlier, in classical computing, we have the inverter (NOT gate), the X gate or Pauli-X gate is the equivalent of classical NOT gate in the quantum computing domain [13] [20].



Fig 11: The Pauli X gate

The matrix is of the form: X ≡ $\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}$.

When the state $|\psi\rangle = \begin{pmatrix}\alpha\\ \beta\end{pmatrix}$ (ref equation 3.2.5 and 3.2.6) is passed through X gate, the following happens:

Output = (X gate) x ($|\psi\rangle$) = $\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}$ x $\begin{pmatrix}\alpha\\ \beta\end{pmatrix}$ = $\begin{pmatrix}\beta\\ \alpha\end{pmatrix}$

For $|0\rangle$, the output = $\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}$ x $|0\rangle$ = $\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}$ x $\begin{pmatrix}1\\ 0\end{pmatrix}$ = $\begin{pmatrix}0\\ 1\end{pmatrix}$ = $|1\rangle$

For $|1\rangle$, the output = $\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}$ x $|1\rangle$ = $\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}$ x $\begin{pmatrix}0\\ 1\end{pmatrix}$ = $\begin{pmatrix}1\\ 0\end{pmatrix}$ = $|0\rangle$

It can be observed that when the Pauli X gate operates on a qubit:

- The coefficients of $|0\rangle$ and $|1\rangle$ are interchanged (values of α and β)
- When the input is $|0\rangle$, output becomes $|1\rangle$
- When the input is $|1\rangle$, output becomes $|0\rangle$

Hence the NOT operation is in place through Pauli X gate.

4.7.2 V gate

The V gate is the square root of Pauli X gate in the quantum computing domain [1] [13]. It is also considered and written as $\sqrt{X}$ gate. V gate can be used as one of the basic gates for construction of different other gates [12] [20] [14].



Fig 12: The V gate

The matrix for V = V = $\frac{1}{2}\begin{pmatrix} 1+j & 1-j \\ 1-j & 1+j \end{pmatrix}$

Now, V x V = $\frac{1}{4}\begin{pmatrix} 2+2j.j & 2-2j.j \\ 2-2j.j & 2+2j.j \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ = X

Also, $V^{\dagger} = \frac{1}{2}\begin{pmatrix} 1-j & 1+j \\ 1+j & 1-j \end{pmatrix}$, is the complex conjugate transpose of V [7] [12] [13] [20].



Fig 13: The V dagger gate

V x $V^{\dagger}$ = $\frac{1}{2}\begin{pmatrix} 1+j & 1-j \\ 1-j & 1+j \end{pmatrix}$ x $\frac{1}{2}\begin{pmatrix} 1-j & 1+j \\ 1+j & 1-j \end{pmatrix} = \frac{1}{4}\begin{pmatrix} 2-2j.j & 2+2j.j \\ 2+2j.j & 2-2j.j \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ = I

$V^{\dagger}$ x $V^{\dagger}$ = $\frac{1}{4}\begin{pmatrix} 2+2j.j & 2-2j.j \\ 2-2j.j & 2+2j.j \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ = X

Observation:

- V is equivalent to $\sqrt{X}$, V x V = X
- $V^{\dagger}$ is equivalent to $\sqrt{X}$, $V^{\dagger}$ x $V^{\dagger}$ = X
- V x $V^{\dagger}$ = $V^{\dagger}$ x V = I, hence unitary
- V is unitary, but not Hermitian
- Two V gates in series will be equivalent to a Pauli X gate [14]
- Two $V^{\dagger}$ gates in series will be equivalent to a Pauli X gate [14]

4.7.3 Controlled V and controlled V dagger gates

We have V = $\frac{1}{2}\begin{pmatrix} 1+j & 1-j \\ 1-j & 1+j \end{pmatrix}$. We would now consider a controlled V gate (adding a control line to the V gate) [12] [13]. The V gate will be applied when the control is active. When the control is not active, the input will be passed through. Also, since there is control input now, there would be an additional output, the control line itself will be the output [20] [15].

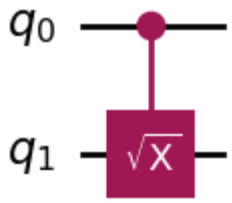


Fig 14: Quantum equivalent circuit for Controlled V gate

Matrix for Controlled V gate = CV = $\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & .5+.5j & .5-.5j \\ 0 & 0 & .5-.5j & .5+.5j \end{pmatrix}$

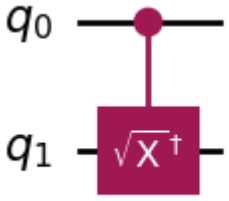


Fig 15: Quantum equivalent circuit for Controlled V Dagger gate

Hence $CV^\dagger$ [12] [13] = $\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & .5-.5j & .5+.5j \\ 0 & 0 & .5+.5j & .5-.5j \end{pmatrix}$

Two CV gates in series will result in: CV x CV

$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & .5+.5j & .5-.5j \\ 0 & 0 & .5-.5j & .5+.5j \end{pmatrix} x \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & .5+.5j & .5-.5j \\ 0 & 0 & .5-.5j & .5+.5j \end{pmatrix}$

$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$ = CNOT gate. Hence CV x CV = CNOT

Hence $CV^\dagger$ = $\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & .5-.5j & .5+.5j \\ 0 & 0 & .5+.5j & .5-.5j \end{pmatrix}$

Two $CV^{\dagger}$ gates in series will result in: $CV^{\dagger}$ x $CV^{\dagger}$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & .5-.5j & .5+.5j \\ 0 & 0 & .5+.5j & .5-.5j \end{pmatrix} x \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & .5-.5j & .5+.5j \\ 0 & 0 & .5+.5j & .5-.5j \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$ = CNOT gate. Hence $CV^{\dagger}$ x $CV^{\dagger}$ = CNOT

We also see that:

$$CV \times CV^{\dagger} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & .5+.5j & .5-.5j \\ 0 & 0 & .5-.5j & .5+.5j \end{pmatrix} x \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & .5-.5j & .5+.5j \\ 0 & 0 & .5+.5j & .5-.5j \end{pmatrix} = I$$

$$CV^{\dagger} \times CV = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & .5-.5j & .5+.5j \\ 0 & 0 & .5+.5j & .5-.5j \end{pmatrix} x \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & .5+.5j & .5-.5j \\ 0 & 0 & .5-.5j & .5+.5j \end{pmatrix} = I$$

Observation:

- CV is equivalent to $C\sqrt{X}$, CV x CV = X
- $CV^{\dagger}$ is equivalent to $C\sqrt{X}$, $CV^{\dagger}$ x $CV^{\dagger}$ = X
- CV x $CV^{\dagger}$ = $CV^{\dagger}$ x CV = I, hence unitary
- CV is unitary, but not Hermitian
- Two CV gates in series will be equivalent to a CNOT gate
- Two $CV^{\dagger}$ gates in series will be equivalent to a CNOT gate

4.7.4 Pauli Y gate

The Y gate matrix is $Y = \begin{pmatrix} 0 & -j \\ j & 0 \end{pmatrix}$. Single qubit quantum state $|\psi\rangle = \alpha |0\rangle + \beta |1 \equiv \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$



Fig 16: Pauli Y gate

When Y gate works on a state ($|\psi\rangle = \alpha |0\rangle + \beta |1\rangle$, the result is:

$Y \times \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} 0 & -j \\ j & 0 \end{pmatrix} \times \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} -j\beta \\ j\alpha \end{pmatrix}$

Also, $Y^{\dagger} = \begin{pmatrix} 0 & -j \\ j & 0 \end{pmatrix}$, which is equal to Y. Hence Y is a Hermitian matrix.

We see:

$Y \times Y^{\dagger} = \begin{pmatrix} 0 & -j \\ j & 0 \end{pmatrix} \times \begin{pmatrix} 0 & -j \\ j & 0 \end{pmatrix} = I$

$Y^{\dagger} \times Y = \begin{pmatrix} 0 & -j \\ j & 0 \end{pmatrix} \times \begin{pmatrix} 0 & -j \\ j & 0 \end{pmatrix} = I$

$Y \times Y = Y^{\dagger} \times Y^{\dagger} = \begin{pmatrix} 0 & -j \\ j & 0 \end{pmatrix} \times \begin{pmatrix} 0 & -j \\ j & 0 \end{pmatrix} = I$

Observation:

- Y is a quantum gate
- $Y \times Y^{\dagger} = Y^{\dagger} \times Y = I$, hence unitary
- $Y^{\dagger} = Y$, hence Y is Hermitian
- Two Y gates will return the same input state

4.7.5 Pauli Z gate

The Z gate matrix is $Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$. Single qubit quantum state $|\psi\rangle = \alpha |0\rangle + \beta |1 \equiv \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$

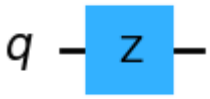


Fig 17: Pauli Z gate

When Z gate works on a state ($|\psi\rangle = \alpha |0\rangle + \beta |1\rangle$, the result is:

$Z \times \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \times \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \alpha \\ -\beta \end{pmatrix}$

Also, $Z^{\dagger} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$, which is equal to Z. Hence Z is a Hermitian matrix.

We see:

$Z \times Z^{\dagger} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = I$

$Z^{\dagger} \times Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = I$

$Z \times Z = Z^{\dagger} \times Z^{\dagger} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = I$

Observation:

- $Z \times Z^{\dagger} = Z^{\dagger} \times Z = I$, hence unitary
- $Z^{\dagger} = Z$, hence Y is Hermitian
- Two Z gates will return the same input state

Considering Pauli X gate, Pauli Y gate and Pauli Z gate, these transformations are interrelated. One of the relations is ZX = jY.

It can be noted that:

$ZX = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \times \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$, Also, $jY = j \begin{pmatrix} 0 & -j \\ j & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$

Therefore ZX = jY

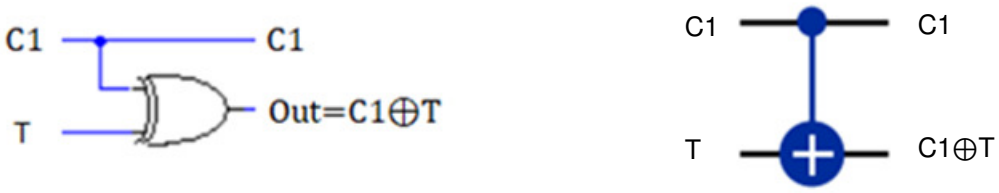
4.8 MCX gate

This is a specific set of gates derived as an extension of CNOT gate or CX gate [19] [17]. MCX stands for multiple controlled X gate. This is also known as multiple controlled Toffoli gate [1] [24] [21].

This has two main units, viz. control and controlled XOR. Refer to the next figure. This is a CNOT or CX gate. Here, the state of the single control line C1 decides if the output will be T or $\overline{T}$.

4.8.1 CX gate

If the control to the XOR unit (C1 in this case) is 0, T is passed as T (no action). If the control is 1, then T is inversed, output is $\overline{T}$ [23] [24] [19].



a) b)

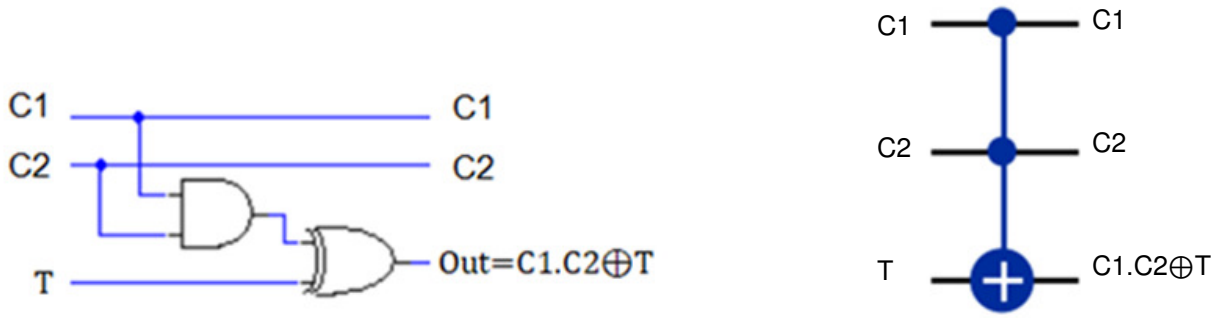
Fig 18: CX gate / C1X gate

a) is the classical circuit equivalent of CX gate
b) is the quantum circuit equivalent of CX gate

4.8.2 CCX gate

Now, we keep the controlled XOR unit [23] [24] as it is, modify the control line as C1.C2 (C1 AND C2), it becomes equivalent to the next figure, which is a Toffoli gate. If C1.C2 is 0, T is passed as T (no action). If the control is 1, then T is inversed, output is $\overline{T}$. This gate e gate has now become CCX gate or C^2X gate (there are two control lines generating the control signal [23].

The figure below is for CCX gate..



a) Classical circuit equivalent of CCX b) Quantum circuit equivalent of CCX gate

Fig 19: CCX gate / C2X gate

Referring to the truth table of Toffoli, it may be noted that when all control lines are 1, the control is activated and XOR is functional (i.e. T is inverted)

4.8.3 CCCX gate

Furthermore, keeping the XOR unit same, we add another line C3 to the control, making the control signal C1.C2.C3 (C1 AND C2 AND C3) [1] [24].

If C1.C2.C3 = 0, T is passed as T (no action). If the control is 1, then T is inversed, output is $\overline{T}$.

Refer to the following figure.

All control lines are replicated at the output. These are C1, C2 and C3

T is inverted when C1 = C2 = C3 = 1.

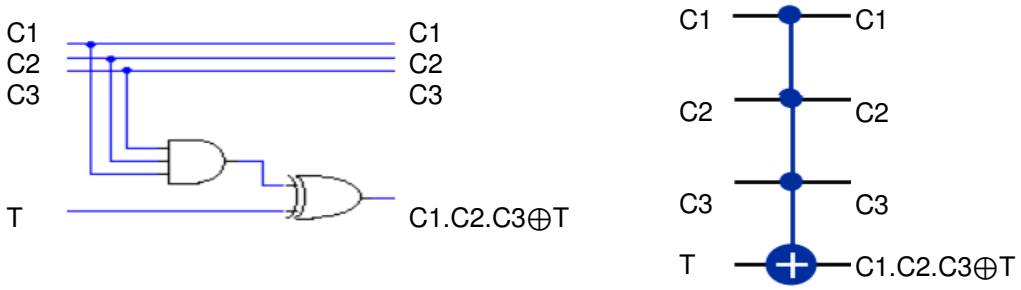


Fig 20: CCCX gate – classical and quantum equivalent circuits

C1	C2	C3	T	C1.C2.C3⊕T
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Table 19: Truth table for CCCX gate

Like CX or CCX, in CCCX gate, T is inverted in the last two rows (C1=C2=C3=1).

4.8.4 CnX gate

CnX gate is equivalent to: 'CC…n times X' gate or C^nX gate or C^nNOT gate. There are n control lines and 1 target line. The target line flips when all n control lines are 1 [1] [24] [19].

The input output mapping is as follows [17]:

- Inputs (n+1)
  - Control inputs (n in number): C1, C2….Cn
  - Target input (1 in number): T
- Outputs (n+1)
  - All control signals: C1, C2….Cn; n of them, unchanged
  - Transformed target signal = (C1 . C2 ……Cn) ⊕ T

We will see in later sections that MCX gate comes to great use in building circuits.

4.9 Quantum cost of a gate

We will now take up the parameter quantum cost for a gate.

Quantum cost of a gate is the number of elementary gates connected inside the gate to provide the functionality.

The elementary gates include:

- V gate (√X gate)
- $V^\dagger$ gate ($\sqrt{X}^\dagger$ gate)
- CNOT gate

Each of the gates has a quantum cost of 1.

Consider the Toffoli gate decomposition in the figure. Number of gates is 5, so the cost of Toffoli gate is 5 [24] [19].

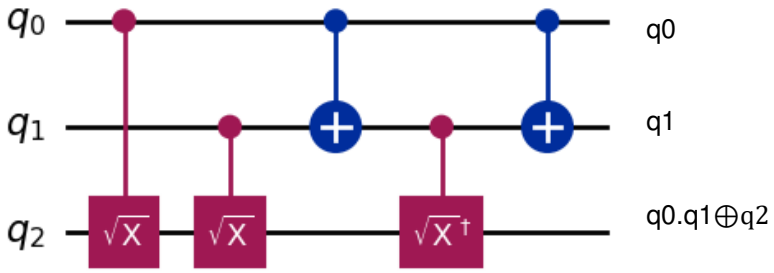


Fig 21: Decomposition of Toffoli gate

Quantum costs of some of the gates are listed below [24].

Gate	Quantum cost	Ancilla input
X gate	0	
CSWAP gate (Fredkin gate)	5	
Peres gate	4	
CX gate (Feynman gate)	1	
CCX gate (Toffoli gate)	5	
CCCX gate	13	1
CCCCX gate	19	1
CCCCCX gate	27	2

Chapter 5 Building reversible circuits

Reversible circuits are built using the reversible gates and there would be interconnections among them. The endeavor would be to identify a set of gates enabling a complete logic (like NAND and NOR) and establish inter connections to realize the logic required.

5.1 Circuits with reversible gates

We have noticed that inverter (NOT gate or X gate) is a reversible function by default. We can surely say that for an output of 0, the input was 1 and for an output of 1, the input was 0.

We have seen XOR is a reversible function, whereas AND is not a reversible function.
We take a fresh look at the truth table of AND gate and XOR gate.

Circuit with AND gate				
A	B	P=A	Q=A.B	Remarks
0	0	0	0	From two output lines, we cannot determine A and B
0	1	0	0	
1	0	1	0	
1	1	1	1	

Circuit with XOR Gate				
A	B	P=A	Q=A⊕B	Remarks
0	0	0	0	From two output lines, we can determine A and B
0	1	0	1	
1	0	1	1	
1	1	1	0	

Table 20: Truth table for AND gate and XOR gate

The XOR function has the bijective nature, hence reversible.

The AND function has two occurrences of 00 combination, hence it is not possible to figure out which input combination caused the 00 condition at the output (in this case, the possible input combinations are 00 and 01), hence not reversible.

To make them unique, we would need another line – so that the outputs are 000 and 100. If could have a truth table to accommodate this, then we do not have any issue in having a reversible circuit for AND, however we had to bear additional cost for the additional line which does not come to any other use.

Consider Toffoli gate truth table below with shaded rows where C = 0.

A	B	C	P=A	Q=B	R=AB⊕C
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	0

Table 21: Truth table for Toffoli gate with rows for C=0 shaded

Now, we make another truth table with the shaded rows of the above truth table.

A	B	C	P=A	Q=B	R=AB⊕C
0	0	0	0	0	0
0	1	0	0	1	0
1	0	0	1	0	0
1	1	0	1	1	1

Table 22: Truth table for Toffoli gate with C = 0

The above table is a clean truth table of an AND gate! The implementation is as follows:

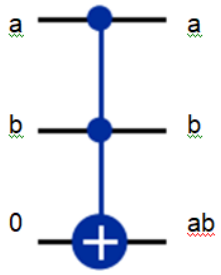


Fig 22: Getting the AND logic from a Toffoli gate

We see that the AND logic is implemented, however the cost has been another line and a 3 input gate has been used to realize a 2 input function.

Here we have added a bit to accommodate the occurrence of 00 more than once.

If the maximum number of a combination recurring in the output truth table is r, then the number of extra bits to be added is Ceiling($\log_2(r)$).

Refer to the following figure for full adder Sum and Carry.

X	Y	Cin	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Table 23: Full adder sum and carry

It may be observed that 10 combination in the output recurs 3 times, also 01 combination recurs 3 times. The existing two outputs, we have to have Ceiling($\log_2(3)$) number of bits added at the output at the minimum. Here, minimum number of bits to be added at the output = Ceiling($\log_2(3)$) = 2.

5.1.1 Toffoli gate as a complete logic

From the Boolean logic perspective, as a particular configuration, for Toffoli gate [1] [23]:

- Inputs are:
  - A = a
  - B = b
  - C = 0
- Outputs are:
  - P = A = a
  - Q = B = b
  - R = AB $\oplus$ 0 = AB = ab (an AND gate)

Now that we have a reversible NOT gate (X gate) and a reversible AND gate (with a Toffoli gate), we have in effect, a reversible NAND gate. Since NAND is a complete logic, we can implement any function in a reversible manner [1].

Another point to be noted here is that in a Toffoli gate, if we have inputs (A = 1, B = 1 and C = c) then the outputs are (P = 1, Q = 1, R = $\bar{c}$. This (output R) gives us a NOT gate implemented with Toffoli gate (though two lines are wasted - not used).

Ancilla and Garbage [1]: The extra input bit is called the ancilla bit. This just helps in reversibility and increases the cost and circuit complexity. The lesser the number of ancilla bits for a reversible circuit – the better. Since a reversible circuit has same number of inputs and outputs, when an ancilla bit is added, it generates an output which is not important. In the terminology of reversible circuits, this extra output bit is called a garbage bit. In the above example, C can be considered as ancilla bit and Q can be considered as a garbage bit.

Hence, Toffoli gate can be configured to realize NOT gate and AND gate. Hence Toffoli gate itself provides a complete logic and can be used to realize any function.

5.1.2 Fredkin gate as a complete logic

From the Boolean logic perspective, as a particular configuration, for Fredkin gate:
- Inputs are:
  - A = a
  - B = b
  - C = 0
- Outputs are:
  - P = A = a
  - Q = $\bar{A}$B $\oplus$ AC = $\bar{a}$b
  - R = $\bar{A}$C $\oplus$ AB = ab

Now that we have a reversible NOT gate (X gate) and a reversible AND gate (with a Fredkin gate), we have in effect, a reversible NAND gate. Since NAND is a complete logic, we can implement any function in a reversible manner.

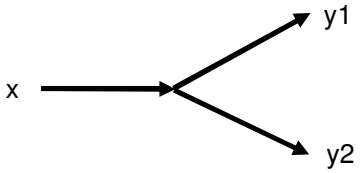
Another point to be noted here is that in a Fredkin gate, if we have inputs (A = a, B = 0 and C = 1) then the outputs are (P = a, Q =.a, R = $\bar{a}$. This (output R) gives us a NOT gate implemented with Fredkin gate (though two lines are wasted - not used).

The extra bit C is the ancilla bit here. Also, Q is the garbage output.

Hence, Fredkin gate can be configured to realize NOT gate and AND gate. Hence Toffoli gate itself provides a complete logic and can be used to realize any function.

5.1.3 Interconnections among the reversible gates

Consider the fanout (direct connection of a signal to two or more input points). Refer to the figure.



Truth table equivalent of fanout:

x	y1	y2
0	0	0
1	1	1

This is a necessity in any circuit for interconnection of components. The truth table equivalent is not surjective (for example output combination of 01 cannot happen with any input) [3]. The mitigation to this is initiated through a CNOT gate with the following configuration:

x	c	y1	y2
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

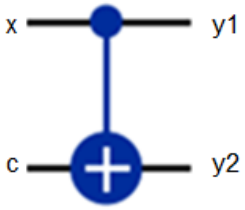


Fig 23: CNOT gate as a fanout

In the above figure, the shaded part (with c=0) of the truth table provides the required fanout equivalent [3]. This is widely used configuration for a fanout.

Fanout reversibility issue has a far reaching effect on reversible circuits – it increases number of components and complexity of a circuit to a great extent.

5.2 Implementation of a function with Bennett trick

The earlier analysis pointed to the fact that among others, Toffoli gate or Fredkin gate is sufficient to have a classical circuit implemented.

We would now move to implementation of any Boolean function (reversible or irreversible) as a reversible circuit.

The approach / methodology: [16]

- Have the irreversible gate f(x) with n input bits producing m output bits
- Construct a reversible gate with n + m bits input (with k garbage bits)
- Have this to provide output as $(x, y) \rightarrow (x, y \oplus f(x))$ where the $\oplus$ is the bitwise XOR
- If applied twice, it becomes: $(x, y) \rightarrow (x, y \oplus f(x)) \rightarrow (x, (y \oplus f(x)) \oplus f(x)) = (x, y)$ – reversible!
- With input as (x, 0), it gives (x, f(x)) – which is the required output
- The function is its own inverse and is defined over all possible inputs

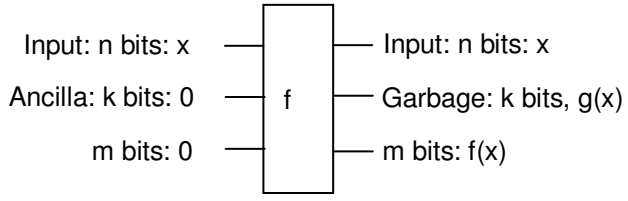


Fig 24: Irreversible circuit to reversible circuit

- Use Bennett Trick [16]: f being reversible, f^{-1} can be implemented if we reverse the order of all the circuit elements (implementing the inverse of each of these elements – since these are Hermitian and unitary / reversible)
- CNOT is used, with B = 0 to copy the value of a bit to the result (refer to the following circuit) [16]

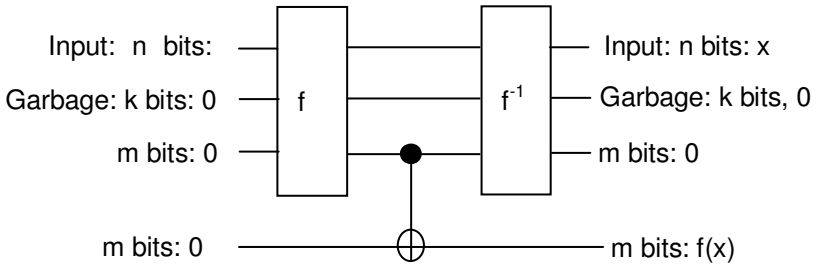


Fig 25: Reversible circuit with garbage handled

So, even an irreversible circuit can be turned into a reversible circuit [16]. The circuit through Bennett Trick may not be optimal [22]. It may use a larger number of circuit elements at the cost of garbage bits which will require optimization.

A few examples of Bennett trick are provided below. Starting with classical circuit, we move to quantum circuit representation.

5.2.1 Reversible AND gate

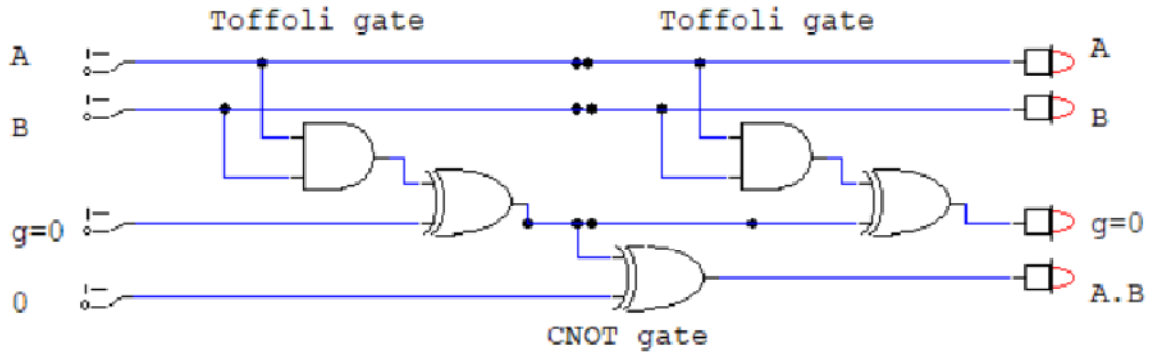


Fig 26: Reversible AND gate with Bennett trick

5.2.2 Reversible OR gate

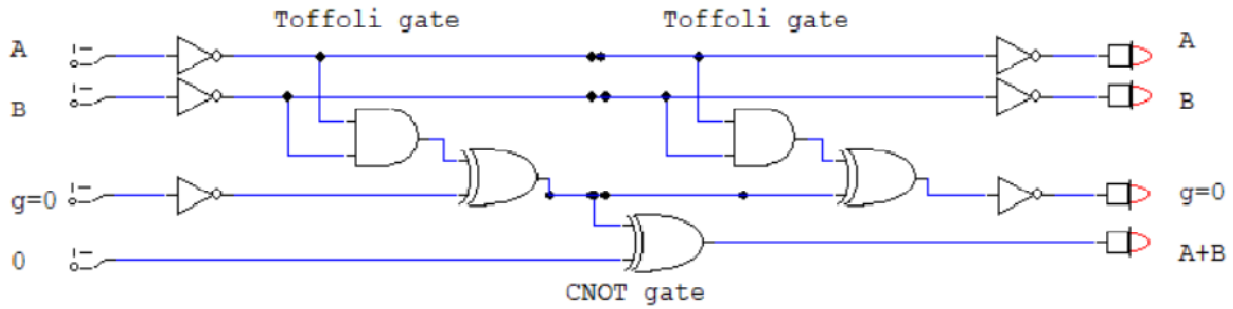


Fig 27: Reversible OR gate with Bennett trick

5.2.3 Reversible Full adder Sum

5.2.3.1 Classical circuit view of full adder Sum

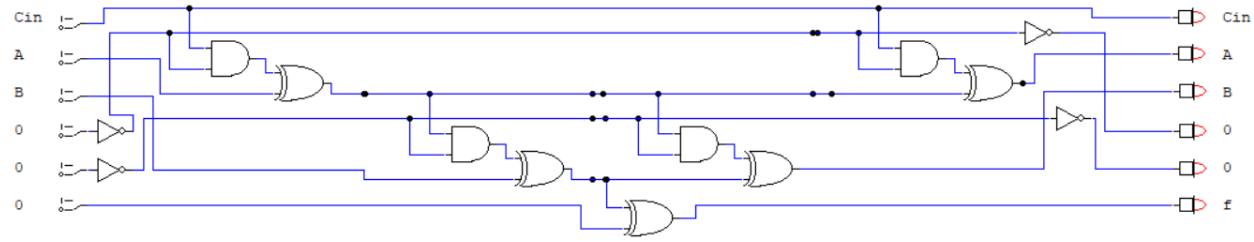


Fig 28: Full adder Sum (classical circuit view)

5.2.3.2 Block diagram view of full adder Sum

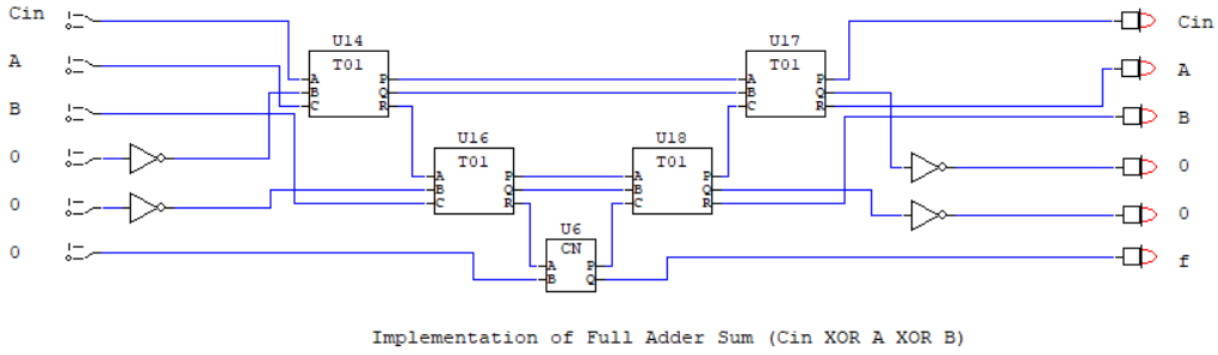


Fig 29: Full adder Sum (Toffoli, NOT, CNOT) with Bennett trick

5.2.3.3 Quantum circuit view of full adder Sum

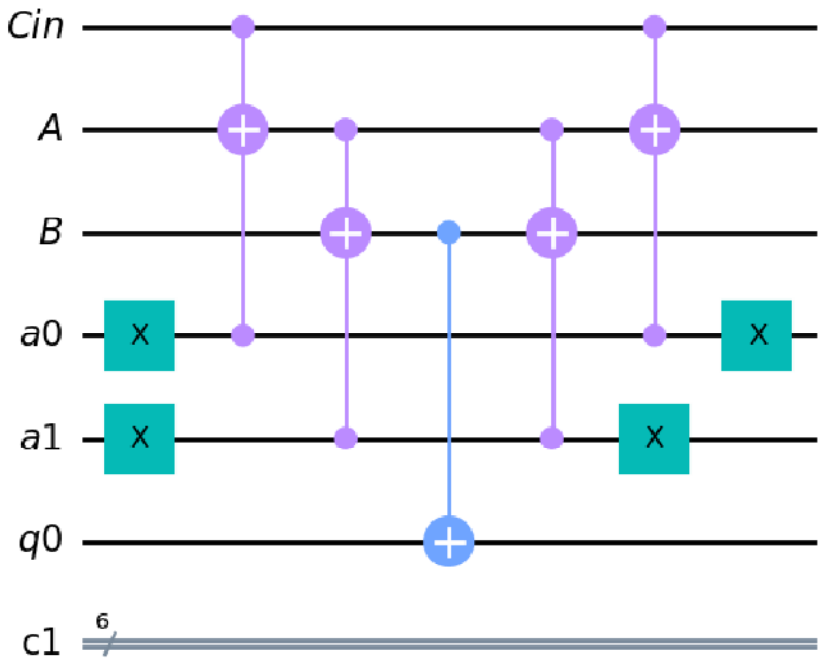


Fig 30: Full adder Sum – the quantum circuit view

5.2.4 Reversible Full adder Carry (Toffoli, NOT, CNOT)

5.2.4.1 Block diagram view of full adder Carry

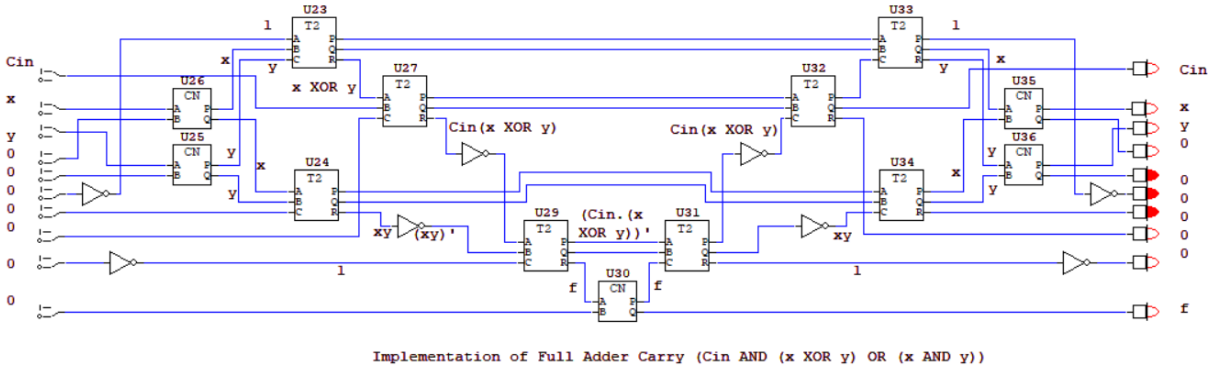


Fig 31: Full adder Carry using Bennett Trick

5.2.4.2 Quantum circuit view of full adder Carry

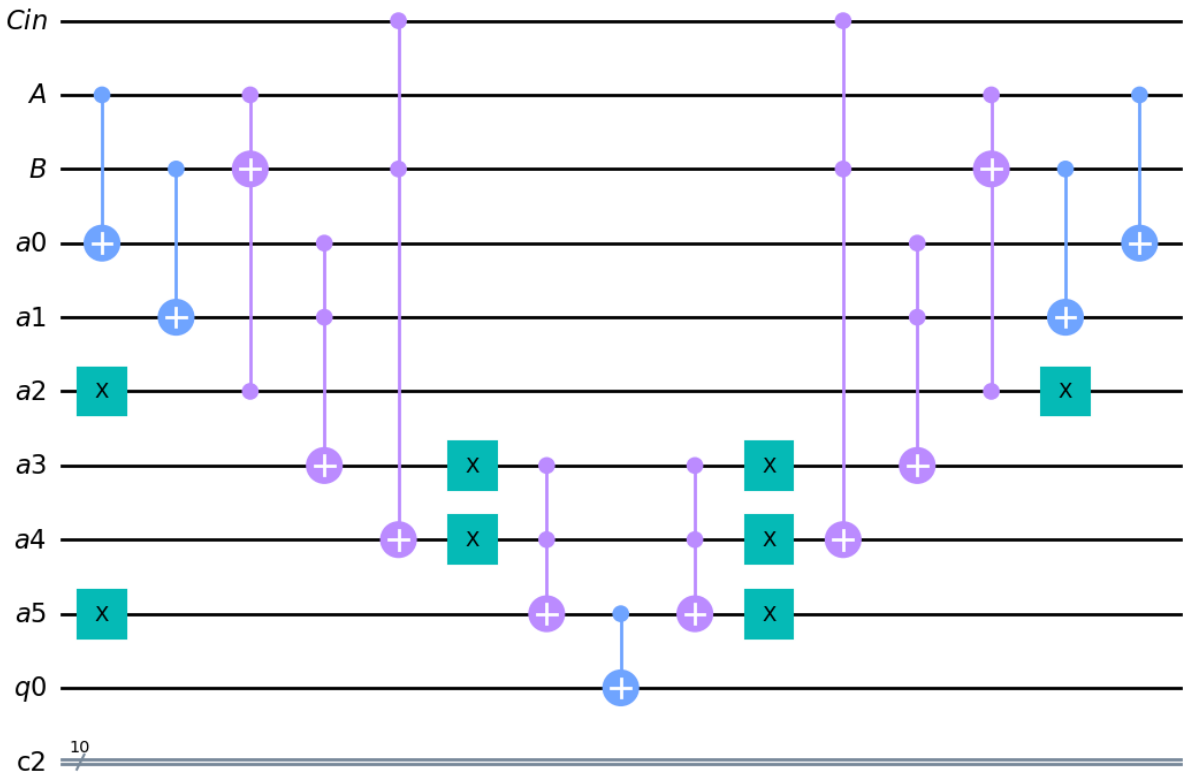


Fig 32: Full adder Carry – quantum circuit view

In this section, we discussed Bennett trick and built up some reversible circuits. While the process gives a circuit for any function, there remains a scope for optimization (in terms of reduction in number of ancilla bits / garbage bits and number of gates).

Chapter 6 Reversible circuit for a function

In this section, we would develop a program to convert any truth table (of up to 10 variables) to a reversible circuit. We would use Python (Jupyter Notebook) [9] with IBM Qiskit [10] library.

6.1 The approach

It was noted that a Boolean function can be represented as Positive Polarity Reed Muller expression. Hence a function of n variables can be written as:

- For 2 input function, Reed Muller form is:

$f(x_1, x_2) = c_{00} \oplus c_{01}x_2 \oplus c_{10}x_1 \oplus c_{11}x_1x_2$

- For 3 input function, Reed Muller form is:

$f(x_1, x_2, x_3) = c_{000} \oplus c_{001}x_3 \oplus c_{010}x_2 \oplus c_{011}x_2x_3 \oplus c_{100}x_1 \oplus c_{101}x_1x_3 \oplus c_{110}x_1x_2 \oplus c_{111}x_1x_2x_3$ … ….. 6.1.1

Same way one can expand for any number of variables.

The main properties of the above Reed Muller form are [25] [19]:
- Reed Muller expression is a complete logic, any function can be expressed this way
- Reed Muller coefficients (c terms, c_{ijk}) can be either 0 or 1
- Using only positive polarity Reed Muller (PPRM) terms - no inverse terms of the variables
- For n variables, there are 2^n terms at the maximum (if all c terms are 1)

We note that the PPRM expression is XOR of multiple terms, these terms being AND of individual variables. Hence this is AND XOR logic.

Let us consider the truth table:

Inp1	Inp0	out1	out0
0	0	1	0
0	1	0	1
1	0	1	0
1	1	1	0

Output lines:

out1 = $\overline{inp1}$. $\overline{inp0}$ + inp1 . $\overline{inp0}$ + inp1 . inp0
out0 = $\overline{inp1}$. inp0

Table 24: Truth table for a 2 input 2 output function

The Positive polarity Reed Muller equivalent of the output lines [19]:

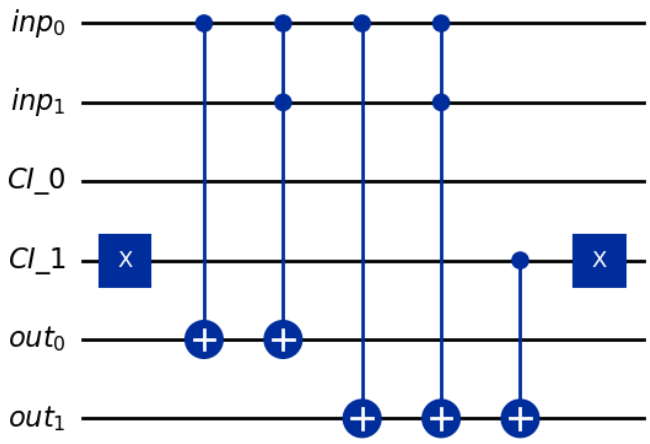
out1 = $\overline{inp1}$. $\overline{inp0}$ + inp1 . $\overline{inp0}$ + inp1 . inp0, converting this to Reed Muller form, we get,
out1 = 1 ⊕ inp0 ⊕ inp1 . inp0 …… 6.1.1

out0 = $\overline{inp1}$. inp0, converting this to Reed Muller form, we get,
out1 = inp0 ⊕ inp1 . inp0 …… ….. 6.1.2

We now consider MCX gate configuration as detailed below. In general, for the MCX circuits, the convention that would be followed with the signals:

- Input: inp0 and inp1 are input lines. Similarly, inpn would be the (n+1)th input
- Output: out0 and out1 are output lines. Similarly, outn would be the (n+1)th output. The output lines start with 0 state, get evolved with the gates and we get the final output at the right
- CI_0 is a constant input of 0

- Cl_1 is a constant input of 1. Since constant input starts with a 0, there is an X gate to generate a 1. X gate on the right gives back the original state which is a 0.



Output lines:

For out1, 1 ⊕ inp0 ⊕ inp1 . inp0
For out0, inp0 ⊕ inp1 . inp0

Fig 33: MCX sample configuration and output

The signals out0 and out1 are 0 initially.
Hence, the final output:

- For out1, inp0 ⊕ inp1 . inp0 ⊕ 1 = 1 ⊕ inp0 ⊕ inp1 . inp0
- For out0, inp0 ⊕ inp1 . inp0

Now we see that the set of output signals are equivalent to the set of output signals of the truth table.

The truth table and the MCX gate circuit are equivalent [19].

PPRM conversion provides us with:

- ANDed components
- The XORing of these ANDed components

On the MCX configuration [19]:

- Each vertical component contributing to the out line is the ANDed component
- Comonents are XORed to get the desired output

We would now transform this approach to a program.

6.2 Program development

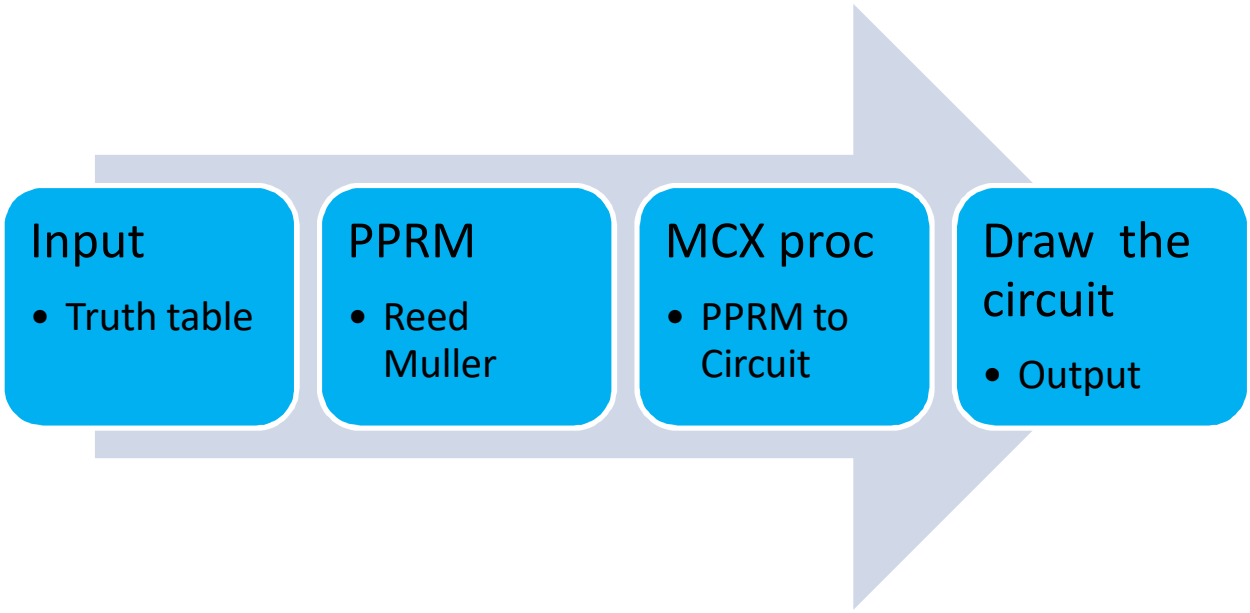


Fig 34: The program approach

The programming environment is Python [9] Jupyter Notebook with Qiskit library [10] for quantum circuit generation.

The program modules are detailed below.

6.2.1 Input

This module provides the function specification to the program in the form of a truth table.

With n input variables and m output variables, a functions would be represented as $\{0, 1\}^n \rightarrow \{0, 1\}^m$. For the function of n variables, there would be 2^n entries in the truth table. We take a list of 2^n elements (like an array of one dimension). The maximum value of each element will be $(2^m - 1)$.

We take an example as before.

inp1	inp0	out2	out1	out0	Value
0	0	1	0	0	4
0	1	0	1	1	3
1	0	1	0	1	5
1	1	1	1	1	7

Here, n = number of inputs = 2 and m = number of outputs = 3.
Hence there are $2^2 = 4$ terms (refer to the Value column). The maximum value entry would be $(2^3 - 1 = 7)$, this happens when value is 1 for all the m output functions)
The truth table will be [4, 3, 5, 7].

One discipline is to be maintained, if there is a function value which is 0 always, the same has to be kept on the right (LSB side), because 0 on the left does not change a number.

The truth table values are now in the program as lists [9] for further processing.

6.2.2 Positive Polarity Reed Muller form (PPRM)

In the previous stage, the program has received a truth table.

From the number of terms, it figures out the number of input lines ($\log_2$(number of elements)) [9]. Taking the number of elements = 4, the number of inputs = 2.

From the maximum value of the elements, it figures out the number of output lines ceiling($\log_2$(maximum element value)) [9]. In the example, taking the max value as 7, the number of outputs = 3.

The PPRM terms are found out which are multiple control terms. These ANDed terms, when XORed, give the function output. This is done using ANFform function in Python [9]

The PPRM terms are:

The term out0 = inp0 $\oplus$ inp1 $\oplus$ inp1 . inp0
The term out1 = inp0
The term out2 = 1 $\oplus$ inp0 $\oplus$ inp0 . inp1

6.2.3 Processing with MCX gate

In the previous stage, the program has received the number of inputs, the number of outputs and the PPRM form [9] [10].

It now uses the IBM Qiskit Library and generates the components of the circuit:
- Input lines (inp0, inp1….)
- Output lines (out0, out1….). these lines are 0 at the initial stage, based on the circuit, they evolve
- Constant 0
- Constant 1 – this line is 0 initially, hence an X gate / inverter is placed to get the 1. Another inverter is placed on the right side to get back the 0
- The XOR boxes – the X part of the MCX gates
- The interconnections

6.2.4 Drawing the circuit

In the previous stage, the program has completed generation of the components of the circuit.

It now uses IBM Qiskit Library [10] on Python [9] to draw the circuit from the components.

6.3 Circuits developed through the program

The program can convert any truth table to a reversible circuit.
We are taking a few examples here.

6.3.1 Full adder (2 bit)

The inputs are Cin (ino4), x0 (inp3), y0 (inp2), x1 (inp1), y1 (inp0). The processed outputs are Cout (out2), r1 (out1), r0 (out0).

The truth table:

C0	x0	y0	x1	y1	Cout	r1	r0	Value
0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	1	0	2
0	0	0	1	0	0	1	0	2
0	0	0	1	1	1	0	0	4
0	0	1	0	0	0	0	1	1
0	0	1	0	1	0	1	1	3
0	0	1	1	0	0	1	1	3
0	0	1	1	1	1	0	1	5
0	1	0	0	0	0	0	1	1
0	1	0	0	1	0	1	1	3
0	1	0	1	0	0	1	1	3
0	1	0	1	1	1	0	1	5
0	1	1	0	0	0	1	0	2
0	1	1	0	1	1	0	0	4
0	1	1	1	0	1	0	0	4
0	1	1	1	1	1	1	0	6
1	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	1	3
1	0	0	1	0	0	1	1	3
1	0	0	1	1	1	0	1	5
1	0	1	0	0	0	1	0	2
1	0	1	0	1	1	0	0	4
1	0	1	1	0	1	0	0	4
1	0	1	1	1	1	1	0	6
1	1	0	0	0	0	1	0	2
1	1	0	0	1	1	0	0	4
1	1	0	1	0	1	0	0	4
1	1	0	1	1	1	1	0	6
1	1	1	0	0	0	1	1	3
1	1	1	0	1	1	0	1	5
1	1	1	1	0	1	0	1	5
1	1	1	1	1	1	1	1	7

The truth table input is:

[0, 2, 2, 4,
1, 3, 3, 5,
1, 3, 3, 5,
2, 4, 4, 6,
1, 3, 3, 5,
2, 4, 4, 6,
2, 4, 4, 6,
3, 5, 5, 7]

Table 25: Truth table for 2 input full adder

This results in the following circuit

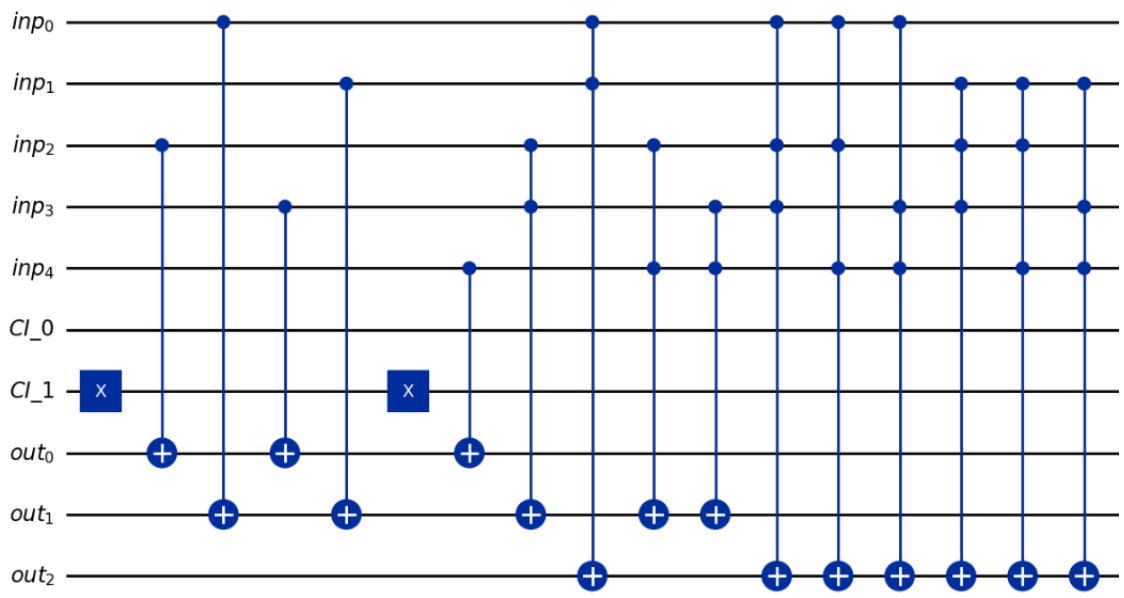


Fig 35: Circuit for 2 input full adder

6.3.2 Mux (2 to 1)

Truth table is given below. Inputs: A0 (inp0), A1 (inp1), S0 (inp2). Output: out0

S0	A1	A0	Out
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

The truth table input is:

[0, 1, 0, 1, 0, 0, 1, 1]

Table 26: Truth table for 2 to 1 mux

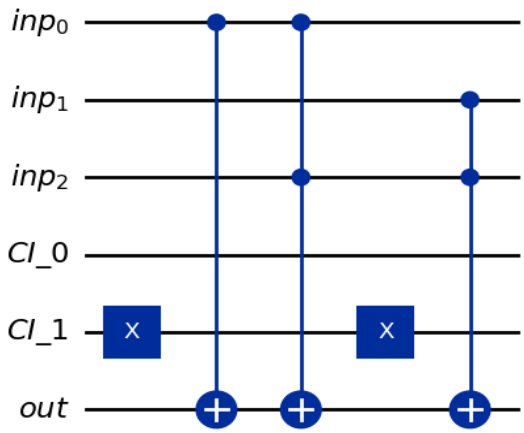


Fig 36: Circuit for 2 to 1 mux

In this section, we have developed the program and used the same to generate the circuits.

Currently it is done for up to 10 input functions. This can be extended for more inputs. However, in the current version of the program, there is no limit for number of outputs.

Chapter 7 Optimization of a reversible circuit

We have developed the circuits through the programs. It may be possible to have the same circuit through less number of components or with lesser complexity.

Getting a circuit in efficient form is the optimization.

We will now have a look at a parameter called quantum cost of a gate.

For reversible circuits, certain parameters are standardized to get it assembled or to work more efficiently. The optimization parameters include:

- Gate count: this is defined by number of gates used in the circuit
- Quantum cost (refer to the section on logic gates) [1]. This is based on how many elementary gates have been used to realize a particular function.

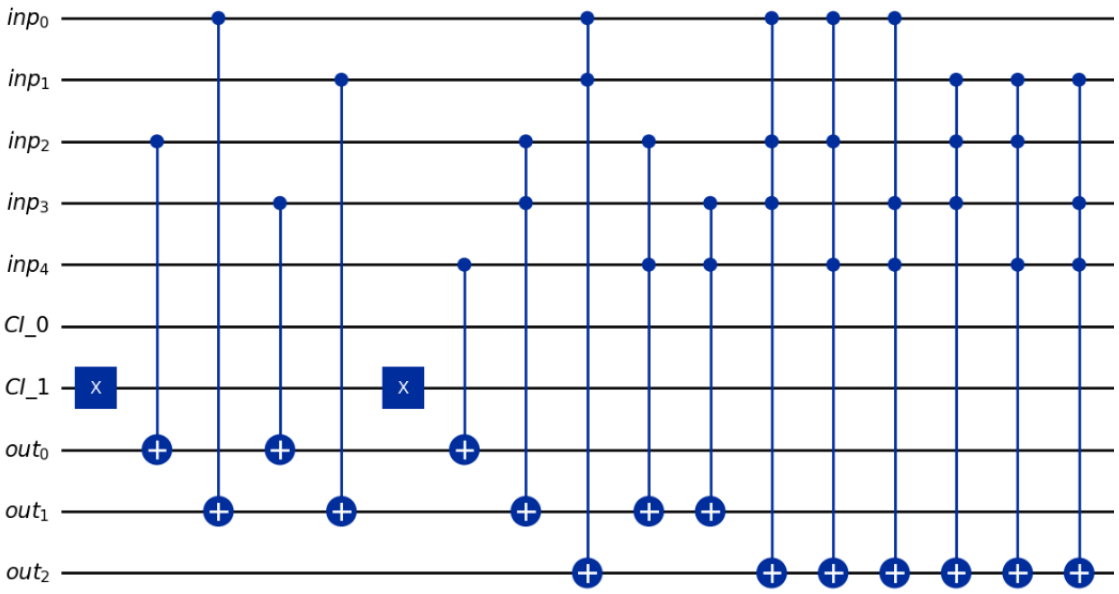
The cost table is shown once again:

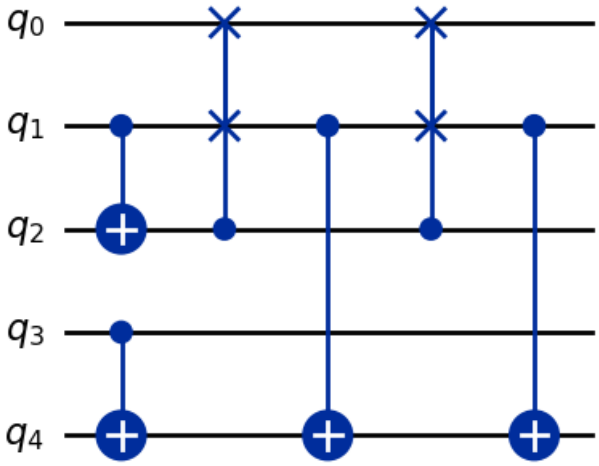
Gate	Quantum cost	Ancilla input
X gate	0	
CSWAP gate (Fredkin gate)	5	
Peres gate	4	
CX gate (Feynman gate)	1	
CCX gate (Toffoli gate)	5	
CCCX gate	13	1
CCCCX gate	19	1
CCCCCX gate	27	2

The approaches to optimization are mainly of two kinds – one at the gate level and the other is at the decomposed gate level [23] with a bend towards algorithmic analysis [19].

We will now see a few examples.

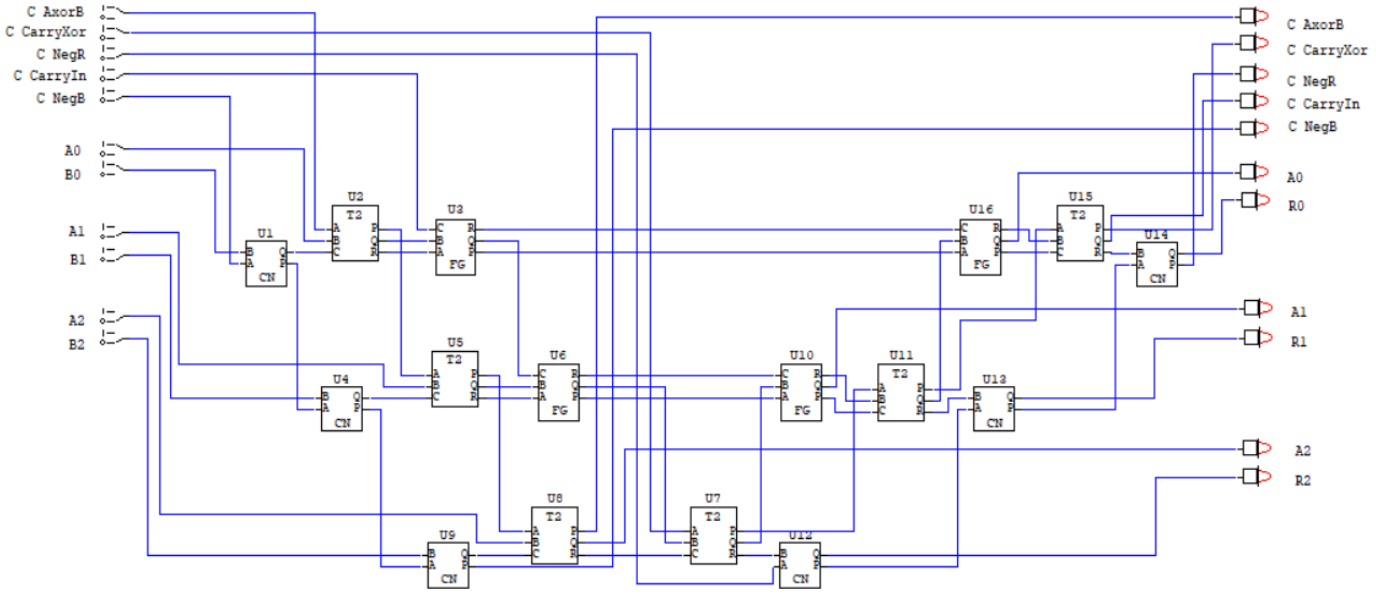
Consider the two circuits for 2 bit full adder given below.





If all the inputs are to be made available at the output, then the first circuit is a necessity. However, if the result bits are enough then the second circuit suffices. The quantum cost of first circuit is more than that of the second. Both circuits are reversible.

This was based on a particular circuit and its implementation.

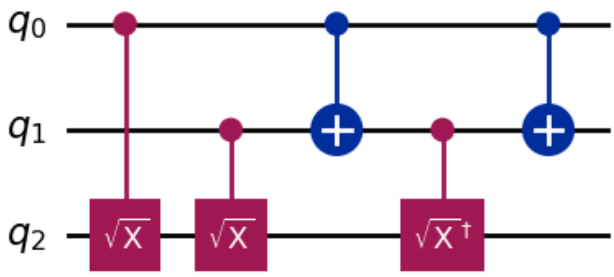


C NegB	C CarryIn	C AxorB	C CarryXor	C NegR	ALU OP
0	0	1	1	0	B +n A (ADD)
1	0	1	1	1	B -n A (SUB)
1	1	1	1	0	A -n B (NSUB)
0	0	1	0	0	B xor A (XOR)
0	0	0	0	0	B (NOP)
0	1	1	1	0	B +n A +n 1
1	1	1	1	1	B -n A -n 1
1	0	1	1	0	A -n B -n 1
0	0	1	0	1	(B xor A)'
0	0	0	0	1	B'

The above circuit is towards a reversible ALU supporting addition, subtraction XOR and NOP [22]. It is a highly optimized circuit with a V shape design [22]. Here the functions and the optimization matched very well.

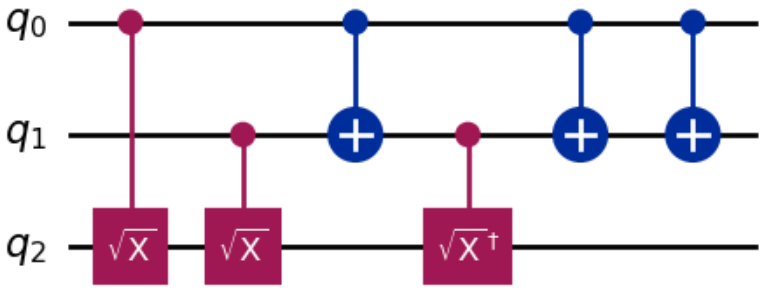
The optimization sometimes also depends on how much a signal can be copied and reused. Copying can be done with a CX gate (cost = 1)

Another approach is to use the decomposed version of the gate and check.
For example, the decomposed Toffoli gate is:



We may not require getting the q1 as it is, say. In that case put a CX gate with q0 as control and q1 as target. Then the two CX gates will cancel each other and the cost will reduce from 5 to 4. The two CX gates annihilate each other (Hermitian).

Earlier, the outputs were (q0, q1, q0.q1⊕q2), now the outputs are (q0, q0 ⊕ q1, q0.q1⊕q2). the transformation still remains reversible.



While there are processes for specific types of optimization, there is a scope for a general solution for efficient synthesis of reversible circuits.

Chapter 8 Results and discussions

Reversible circuits have potential in low energy computing [1] [5]. Also, in quantum computing, the transformations are reversible (unitary) and reversible circuits (both synthasis and analysis) are used widely in quantum computing.

Theoretically, reversibility will reduce energy consumption; however the reversibility has to go down the silicon / substrate level of manufacturing.

In relation to quantum computing, we established the different approaches to the circuits developed circuits – including:

- Boolean algebra
- Matrices
- Cycles / groups

With n variables, we can have 2^n input combinations [7]. With 2^n input combinations, we get 2^{2^n} different functions [1].

Value of n	$\mathbf{2^{2^n}}$
1	4
2	16
3	256

For reversible circuits with n inputs, there are n outputs (n different functions). The output functions taken together should cover all possible values.

A	**B**		**P=A**	**Q=f(A)**
0	0		0	1
0	1	→	0	0
1	0		1	0
1	1		1	1

Table 27: Truth table for a reversible function

The function is of the form $\{0, 1\}^n = \{0, 1\}^n$

We have the same number of inputs and outputs.

The function transforming the input to the output is to be a bijective function (surjective and injective at the same time). These should be both one to one and noto function. In other words, both input and output should cover all possible combinations and no repeat (or no missing combination) should be there. The functions will be balanced function (where number of 1's and number of 0's should be same – this is a necessary condition, not sufficient)

For each gate, the unitary nature was confirmed. The dagger (complex conjugate transpose) matrix was studied, applied to the gates and results were verified.

For each gate, we could see that the matrix M and matrix $M^\dagger$ are related in a way that:
M is unitary if :

$M \times M^\dagger = M^\dagger \times M = I$ (identity matrix).

Additionally, if $M = M^{\dagger}$, then the matrix is Hermitian. In this case, two consecutive gates would return the original input.

The cycle / group point of view: if the function has cycles of 3 or more, then LCM (of cycle lengths) number of times the gate has to be repeated to get the original input back.

Reversible gates have been discussed. The use and behavior of some of the gates in quantum computing has been checked.

Process of building reversible circuits has been discussed. The inverse behavior of some of the reversible gates (conforming to Hermitian property) have been exploited to get the input state back. The copying of the result through a CNOT / CX gate and using the Hermitian property of the components, Bennett trick has successfully realized functions (may not be optimized though) [22].

Requirement of ancilla bits which finally leads to garbage generation has been studied.

A program has been developed to synthesize any function (of up to 10 input bits). This programming was on Jupyter Notebook (on Python) and uses the IBM Qiskit Library.

It is able to convert a truth table to a standard quantum circuit – using MCX gates.

An example is presented below.

inp2	inp1	inp0	out2	out1	out0	Value
0	0	0	0	0	1	1
0	0	1	1	1	0	6
0	1	0	0	1	0	2
0	1	1	1	1	1	7
1	0	0	0	0	1	1
1	0	1	0	1	0	2
1	1	0	1	1	1	7
1	1	1	1	0	1	5

The truth table input is:

[1, 6, 2, 7, 1, 2, 7, 5]

Note: Not a reversible function (2 and 7 are repeated, 3 and 4 are not there)

Table 28: A truth table to be implemented

From the truth table entered, the program could identify:

- The number of input bits
- The number of output bits
- PPRM form (Positive Polarity REED Muller form / AND EXOR form)
    - 1 ⊕ Inp0 ⊕ inp1 ⊕ inp1 . inp2 ⊕ inp0 . inp1 . inp2
    - Inp0 ⊕ inp1 ⊕ inp0 . inp1 ⊕ inp0 . inp1 . inp2
    - Inp0 ⊕ inp0 . inp2 ⊕ inp1 . inp2

With the configuration of MCX gates the function is realized.

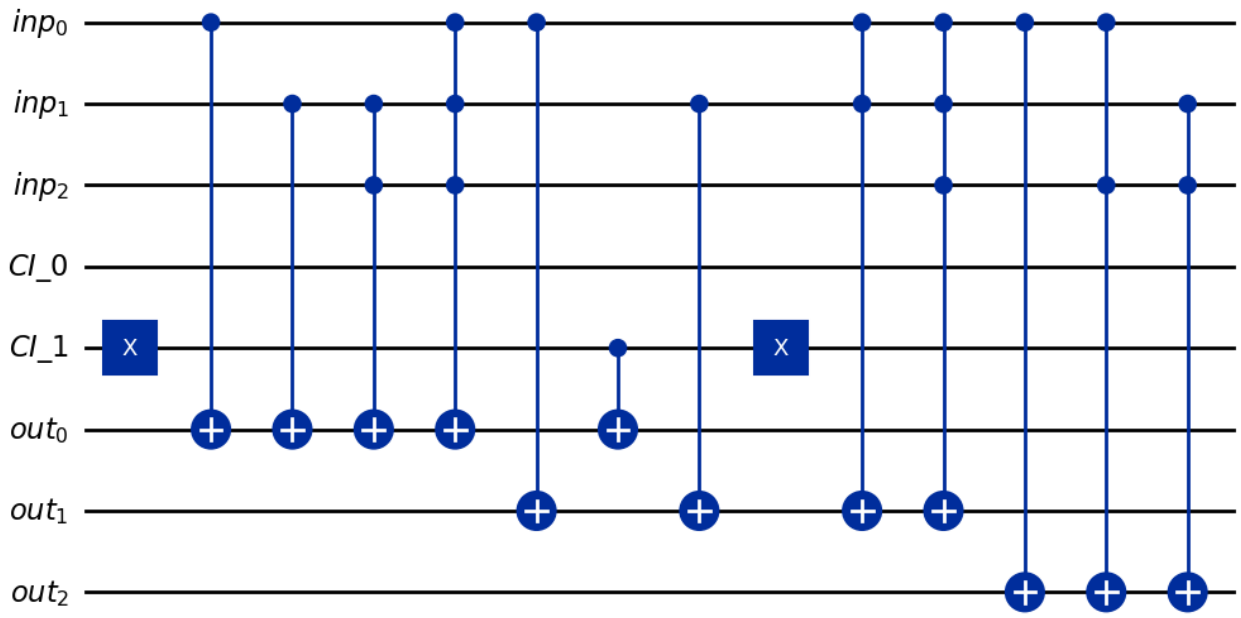


Fig 37: Circuit for the given truth table

Likewise, any function can be converted into a reversible circuit.

After generation of circuit, we discussed the optimization of these circuits.

Different approaches were discussed.

One way is to reduce the number of gates using different gates or using the intermediate results so that same function does not have to be generated more than once. Copying can be done with CX gate which has a cost of 1.

However, research work is going on for generalized algorithm for optimization.

Chapter 9 Conclusion and future scope

9.1 Conclusion

Field of reversible circuits is comparatively new. It started off in the first half of 1900, but picked up speed after the 1960's. There were two special reasons:

- Low energy computing became, and till date – is very important especially when chip manufacturing has reached nano level and the heat dissipation issue is very prominent
- Quantum computing picked up the speed in 1900's. Quantum computing operations (except measurement) are all unitary. In classical circuits the area having the unitary property is reversible circuits. A number of areas of quantum computing will need the study of unitary operators and the analysis would require study of reversible circuits.

Based on the above, in this project, we have taken up the following:

Mathematical view of the reversibility in circuits

Mathematical view has been discussed. Relation of the reversible circuits are related to Boolean algebra, matrices and cycles/groups have been discussed

Reversible gates

Some of the reversible gates [1] have been studied and the mathematical connect have been checked and established. This can be used for further study of reversible circuits.

Building reversible circuits

Building reversible circuits have been studied. Processes such as Bennett Trick have been used to build circuits and the reversibility has been established.

Program to convert a truth table to a circuit

Program has been written to convert a truth table of a function to a reversible circuit using MCX gate. Up to 10 variables can be taken as inputs; there is no restriction on number of outputs.

Discussion on optimization

Different approaches have been discussed for optimization – main parameters being [1] quantum cost, number of gates, number of ancilla / garbage bits etc.

9.2 Future scope

The main areas of future works include:

- Generalized optimization technique for reversible circuits
- Quantum computing
- Possibility of a reversible CPU to be made practically

Bibliography

[1] M. Saeedi and I. Markov, “Synthesis and Optimization of Reversible Circuits - A Survey,” *ACM Comput. Surv.*, vol. 45, Oct. 2011, doi: 10.1145/2431211.2431220.

[2] A. De Vos, *Reversible Computing: Fundamentals, Quantum Computing, and Applications*, 1st ed. Wiley, 2010. doi: 10.1002/9783527633999.

[3] T. Toffoli, “Reversible computing”.

[4] M. Aditya, B. Kumar, and M. Vasantha, “Reversible full/half adder with optimum power dissipation,” Jan. 2016, pp. 1–4. doi: 10.1109/ISCO.2016.7726975.

[5] M. P. Frank, “Introduction to reversible computing: motivation, progress, and challenges,” in *Proceedings of the 2nd conference on Computing frontiers*, Ischia Italy: ACM, May 2005, pp. 385–390. doi: 10.1145/1062261.1062324.

[6] M. Lukac, G. W. Dueck, M. Kameyama, and A. Pathak, “Building a Completely Reversible Computer,” Feb. 2017.

[7] T. Sapv, D. Bera, A. Maitra, and S. Maitra, *Quantum Algorithms for Cryptographically Significant Boolean Functions: An IBMQ Experience*. in SpringerBriefs in Computer Science. Singapore: Springer Singapore, 2021. doi: 10.1007/978-981-16-3061-3.

[8] E. G. Rieffel and W. H. Polak, *Quantum Computing: A Gentle Introduction*. MIT Press, 2011.

[9] “Welcome to Python.org,” Python.org. Accessed: Apr. 12, 2024. [Online]. Available: https://www.python.org/

[10] “IBM Quantum Computing | Qiskit.” Accessed: Apr. 12, 2024. [Online]. Available: https://www.ibm.com/quantum/www.ibm.com/quantum/qiskit

[11] V. Singh and K. B. Ramesh, “Introduction to Reversible Logic Gates and its Operations,” *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 10, no. 2, pp. 1148–1156, Feb. 2022, doi: 10.22214/ijraset.2022.40249.

[12] A. Alasow, P. Jin, and M. Perkowski, “Quantum Algorithm for Variant Maximum Satisfiability,” *Entropy*, vol. 24, no. 11, Art. no. 11, Nov. 2022, doi: 10.3390/e24111615.

[13] “on_gates.pdf.” Accessed: Apr. 13, 2024. [Online]. Available: https://threeplusone.com/pubs/on_gates.pdf

[14] S. Saini and R. Majumdar, “A novel design of reversible 2: 4 decoder,” Mar. 2015. doi: 10.1109/ICSPCom.2015.7150670.

[15] V. Kumar, “A Review on Reversible Logic Gates and their Implementation”, Accessed: Apr. 12, 2024. [Online]. Available: https://www.academia.edu/35317193/A_Review_on_Reversible_Logic_Gates_and_their_Implementation

[16] D. Bacon, “CSE 599d - Quantum Computing Reversible Classical Circuits and the Deutsch-Jozsa Algorithm”.

[17] “Function References · QuantumCircuitOpt.jl.” Accessed: Apr. 24, 2024. [Online]. Available: https://harshangrjn.github.io/QuantumCircuitOpt.jl/dev/function_references/

[18] “IBM Quantum,” IBM Quantum. Accessed: Apr. 13, 2024. [Online]. Available: https://quantum.ibm.com/

[19] T. Ahmed, A. Younes, and A. Elsayed, “Improving the quantum cost of reversible Boolean functions using reorder algorithm,” *Quantum Inf. Process.*, vol. 17, Mar. 2018, doi: 10.1007/s11128-018-1874-1.

[20] M. Surekha, “Efficient Approaches for Designing Quantum Costs of Various Reversible Gates,” *Int. J. Eng. Stud.*, vol. 9, no. 1, pp. 57–78, Jun. 2017, doi: 10.37622/IJES/9.1.2017.57-78.

[21] D. M. Miller, R. Wille, and Z. Sasanian, “Elementary Quantum Gate Realizations for Multiple-Control Toffoli Gates,” in *2011 41st IEEE International Symposium on Multiple-Valued Logic*, Tuusula, Finland: IEEE, May 2011, pp. 288–293. doi: 10.1109/ISMVL.2011.54.

[22] M. K. Thomsen, R. Glück, and H. B. Axelsen, “Reversible arithmetic logic unit for quantum arithmetic,” *J. Phys. Math. Theor.*, vol. 43, no. 38, p. 382002, Aug. 2010, doi: 10.1088/1751-8113/43/38/382002.

[23] V. V. Shende, A. K. Prasad, I. L. Markov, and J. P. Hayes, “Reversible Logic Circuit Synthesis,” arXiv.org. Accessed: Jan. 01, 2024. [Online]. Available: https://arxiv.org/abs/quant-ph/0207001v4

[24] M. H. A. Khan, “Primitive quantum gate realizations of multiple-controlled Toffoli gates,” in *16th Int’l Conf. Computer and Information Technology*, Khulna: IEEE, Mar. 2014, pp. 443–447. doi: 10.1109/ICCITechn.2014.6997380.

[25] S. M. R. Taha, *Reversible Logic Synthesis Methodologies with Application to Quantum Computing*, vol. 37. in Studies in Systems, Decision and Control, vol. 37. Cham: Springer International Publishing, 2016. doi: 10.1007/978-3-319-23479-3.

Appendix

Bijective relation

This section provides information on relation and function, especially for reversible circuits where the relation is bijective.

Relation in general

Relation R from set A to set B is a subset of the cartesian product A X B and is derived by describing a relationship between first and second element of the ordered pair A X B

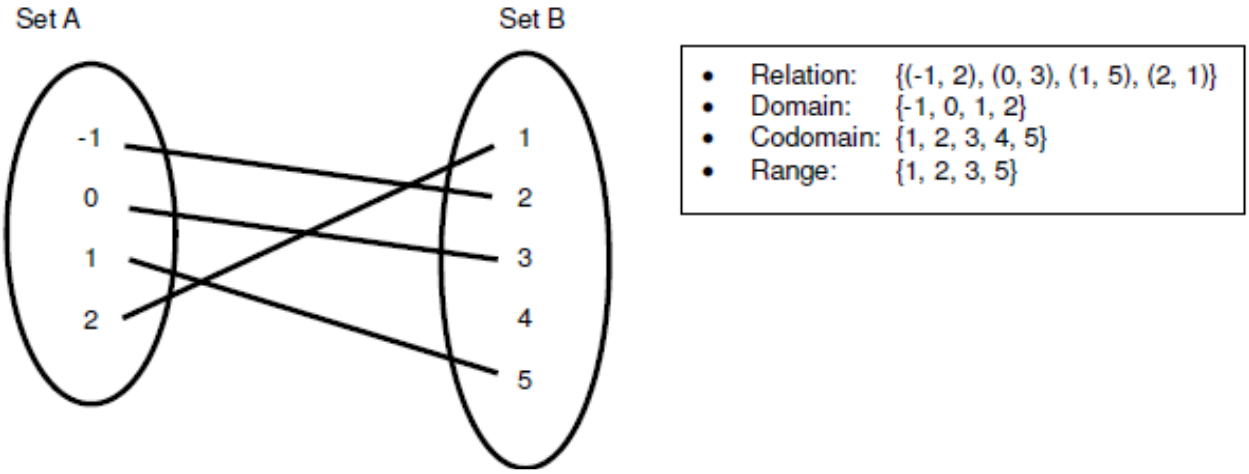


Fig 38: Relation

Function in general

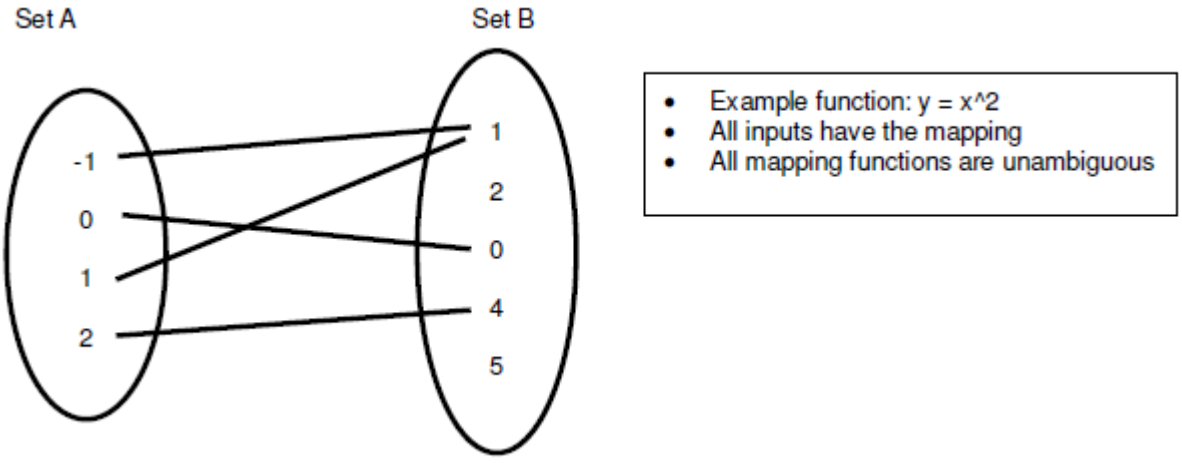


Fig 39: Example of a function

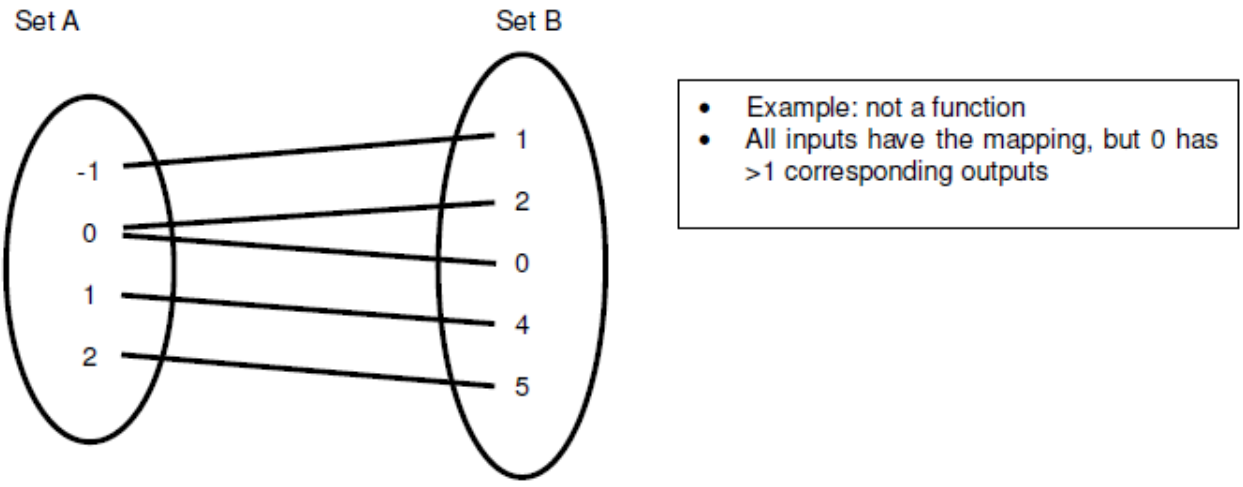


Fig 40: A relation which is not a function

Injective function

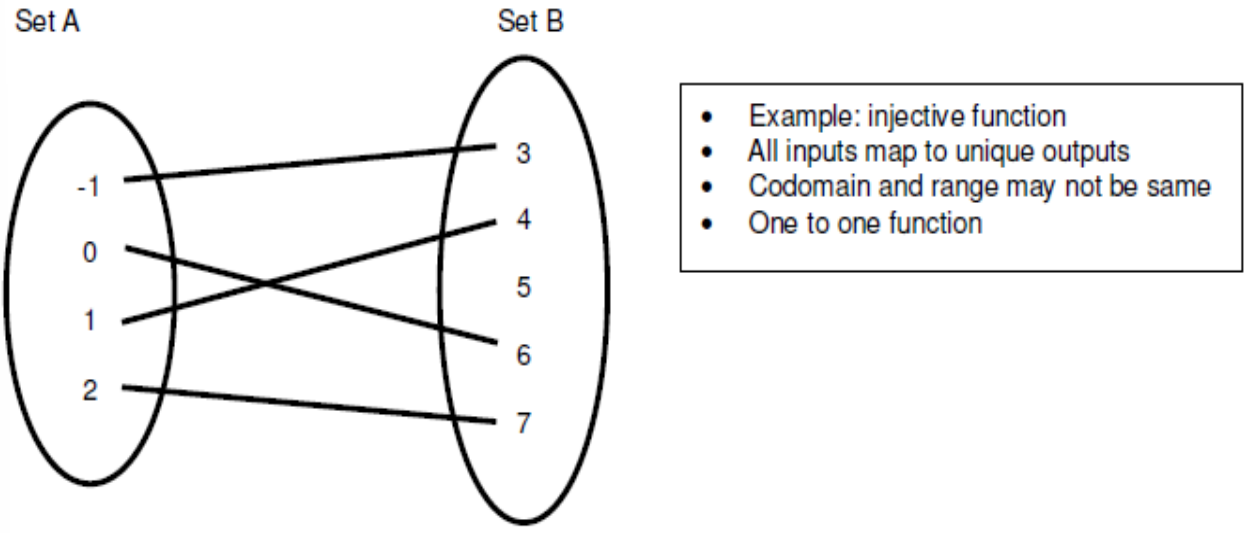


Fig 41: Example of an injective function

Surjective function

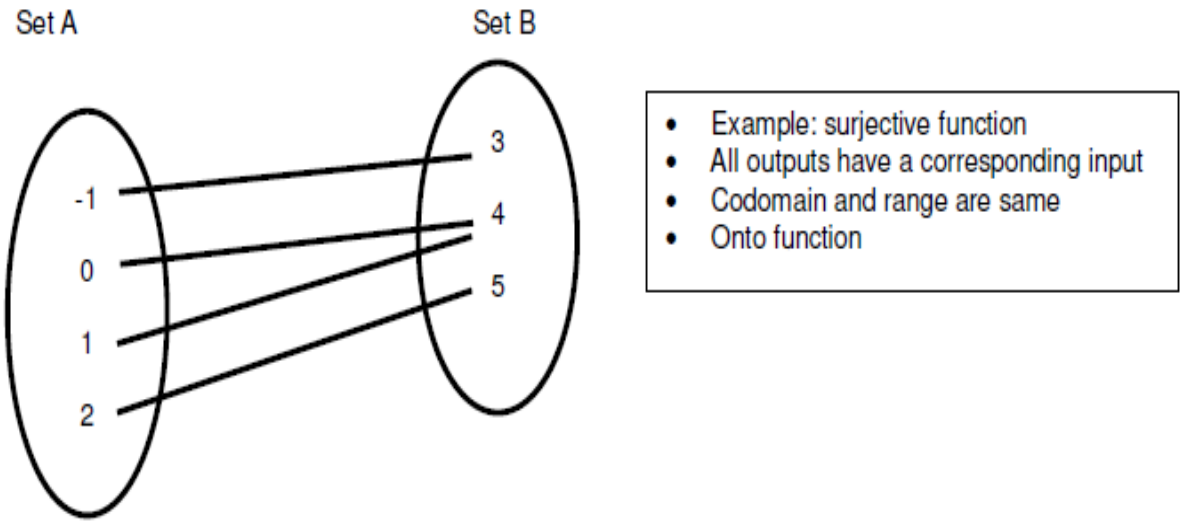


Fig 42: Example of a surjective function

Bijective function

Bijective function provides a one-to-one correspondence (both surjective and injective).

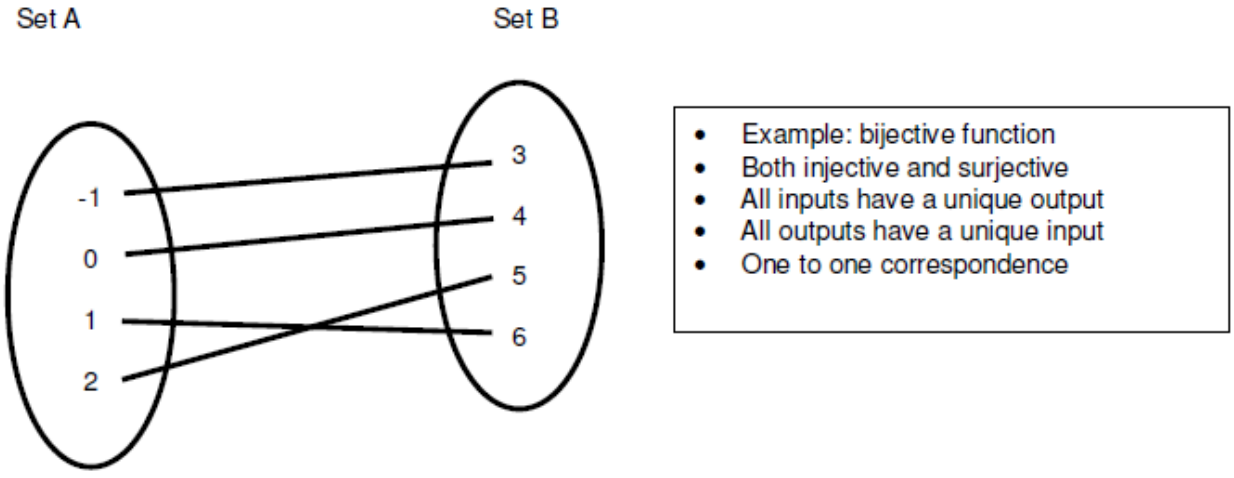


Fig 43: Example of a bijective function

Program to generate reversible circuit for a given truth table

```
from qiskit import *
from qiskit import assemble
from qiskit import QuantumCircuit
from qiskit.circuit.library import SXdgGate
from qiskit.circuit.library import C3XGate
from qiskit.circuit.library import C4XGate
from qiskit.circuit.library import XGate
from qiskit.circuit.library import MCXGate
from qiskit.circuit.library.standard_gates.swap import SwapGate
import numpy as np
import math
import array as arr
%matplotlib inline
from sympy.logic.boolalg import ANFform
from sympy.abc import a, b, c, d, e, f, g, h, i, j
#functions:
def DecimalToBinary(D2BNum, D2BNoOfBits):
    D2BL01 = []
    for D2BNo01 in range(D2BNoOfBits):
        D2BL01 = D2BL01 + [0]
    for D2BNo01 in range(D2BNoOfBits):
        if (D2BNum >= 1):
            D2BL01[D2BNoOfBits - 1 - D2BNo01] = D2BNum % 2
            D2BNum = D2BNum // 2
    return(D2BL01)
def Rev1(lst):
   new1_lst = lst[::-1]
   return new1_lst

#main:
TruthTable = [1, 6, 2, 7, 1, 2, 7, 5]
NoOfInp = int(math.log(len(TruthTable), 2))
NoOfOut = math.ceil(math.log(max(TruthTable) + 1, 2))
#target line-wise boolean expressions
TargetList = []
for Int01 in range(NoOfOut):
    TargetListRows = []
    for Int02 in range(len(TruthTable)):
        List01 = DecimalToBinary(TruthTable[Int02], NoOfOut)
        TargetListRows.append(List01[NoOfOut - 1 - Int01])
    TargetList.append(TargetListRows)
ListOfVariables = []
for Int01 in range(NoOfInp):
    ListOfVariables = ListOfVariables + [chr(97+Int01)]
ListOfVariables=Rev1(ListOfVariables)
inp=QuantumRegister(NoOfInp, 'inp')
CI_0 = QuantumRegister(1, 'CI_0')
CI_1 = QuantumRegister(1, 'CI_1')
out = QuantumRegister(NoOfOut, 'out')
qc1=QuantumCircuit(inp, CI_0, CI_1, out)
#for CI_1, input was 0, now X gate makes it 1 for use in circuit
qc1.x(CI_1)
TConnects=[]
```

```
for Int01 in range(NoOfOut):
    TmpSum=0
    for Int02 in range(len(TruthTable)):
        TmpSum=TmpSum + TargetList[Int01][Int02]
    if (TmpSum == 0): #CI_0 placed on output
        gate=MCXGate(1)
        qc1.append(gate, [CI_0, out[Int01]])
    else:
        StrANF0 = str(ANFform(ListOfVariables, TargetList[Int01]))
        StrANF1 = StrANF0.replace("True", "T")
        StrANF2 = ""
        for Int03 in range(len(StrANF1)):
            if (StrANF1[Int03] != " "):
                StrANF2 = StrANF2 + StrANF1[Int03]
        StrANF3 = ""
        for Int03 in range(len(StrANF2)):
            if (StrANF2[Int03] == "^"):
                StrANF3=StrANF3 + " "
            elif ((StrANF2[Int03] == "&") or (StrANF2[Int03] == "(") or (StrANF2[Int03] == ")")):
                StrANF3 = StrANF3
            else:
                StrANF3 = StrANF3 + StrANF2[Int03]
        ListOfTerms = StrANF3.split()
        print(ListOfTerms)
        for Int03 in range(len(ListOfTerms)): #process each term
            if (len(ListOfTerms[Int03]) == 1): #all single term
                if (ListOfTerms[Int03] == "T"): #single term is T
                    TConnects = TConnects + [Int01]
                else: #single terms other than T
                    InpIndex = ord(ListOfTerms[Int03]) - 97
                    gate = MCXGate(1)
                    qc1.append(gate, [inp[InpIndex], out[Int01]])
                    #print(ListOfTerms[Int03], ord(ListOfTerms[Int03]), InpIndex)
            else: #terms other than single terms
                Degree = len(ListOfTerms[Int03])
                ListOfCharactersInTerm = []
                for Int04 in range(Degree):
                    ListOfCharactersInTerm = ListOfCharactersInTerm + [ListOfTerms[Int03][Int04]]
                #print(ListOfCharactersInTerm)
                gate=MCXGate(Degree)
                ListOfGateComponents=[]
                for Int04 in range(Degree):
                    CharacterToIndex = ord(ListOfCharactersInTerm[Int04]) - 97
                    ListOfGateComponents = ListOfGateComponents + [inp[CharacterToIndex]]
                ListOfGateComponents = ListOfGateComponents + [out[Int01]]
                qc1.append(gate, ListOfGateComponents)
#set up TConnects
gate = MCXGate(1)
for Int01 in range(len(TConnects)):
    qc1.append(gate, [CI_1, out[TConnects[Int01]]])
#input was 0, made 1, used, back to 0 now
qc1.x(CI_1)
qc1.draw('mpl')
```