

California State University, San Bernardino

CSUSB ScholarWorks

Theses Digitization Project

John M. Pfau Library

2013

Quantum cryptography

Razvan Augustin Dinu

Follow this and additional works at: <https://scholarworks.lib.csusb.edu/etd-project>



Part of the [Information Security Commons](#)

Recommended Citation

Dinu, Razvan Augustin, "Quantum cryptography" (2013). *Theses Digitization Project*. 4047.
<https://scholarworks.lib.csusb.edu/etd-project/4047>

This Thesis is brought to you for free and open access by the John M. Pfau Library at CSUSB ScholarWorks. It has been accepted for inclusion in Theses Digitization Project by an authorized administrator of CSUSB ScholarWorks. For more information, please contact scholarworks@csusb.edu.

QUANTUM CRYPTOGRAPHY

A Thesis

Presented to the

Faculty of

California State University,

San Bernardino

In Partial Fulfillment

of the Requirements for the Degree

Master of Arts

in

Mathematics

by

Razvan Augustin Dinu

March 2013

QUANTUM CRYPTOGRAPHY

A Thesis

Presented to the

Faculty of

California State University,

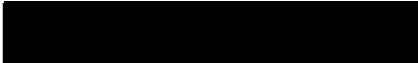
San Bernardino

by

Razvan Augustin Dinu

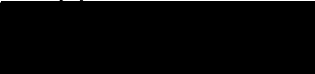
March 2013


Approved by:

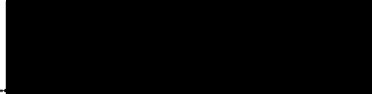

Dr. Chris Freiling, Committee Chair

3/12/2012
Date


Dr. Corey Dunn, Committee Member


Dr. Ronald Trapp, Committee Member


Dr. Peter Williams, Chair,
Department of Mathematics


Dr. Charles Stanton
Graduate Coordinator,
Department of Mathematics

ABSTRACT

The Thesis starts by presenting RSA, the most used cryptosystem in the world. Then the Thesis presents the basis of quantum information theory, while also making a succinct comparison with the equivalent classical counterparts and presenting briefly how a quantum computer works. The Thesis then builds a case for using a quantum computer for solving cryptographic problems. It looks at the quantum turing machine concept, explores why use quantum computers and presents Deutsch's problem which allows one to select from amongst the parallel paths a quantum computer calculates. After the Thesis briefly presents the classes of quantum algorithms, it describes in detail the algorithm stages a quantum computer would have to go through in order to break an RSA code: Shor's factoring algorithm, quantum fourier transform (QFT), tensor product factorization, a circuit for QFT, and Simon's algorithm for phase estimation.

ACKNOWLEDGEMENTS

I wish to thank Dr. Chris Freiling for all his help and support in the development and formation of this paper. I also want to thank Dr. Corey Dunn and Dr. Ronald Trapp for their comments on the paper as well. Your assistance was very much appreciated.

Table of Contents

Abstract	iii
Acknowledgements	iv
List of Figures	vi
1 Introduction	1
2 Quantum Information	3
2.1 Basic Quantum Information Theory	3
2.2 How a Quantum Computer Works	12
2.3 Shor's Algorithm	18
3 The Rationale for Using a Quantum Computer	19
3.1 The Quantum Turing Machine	19
3.2 Why Use Quantum Computers	21
3.3 Deutsch's Problem	22
4 Quantum Algorithms	28
4.1 Classes of Quantum Algorithms	28
4.2 Shor's Factoring Algorithm and Order Finding	29
4.3 Quantum Fourier Transform	34
4.4 Tensor Product Factorization	37
4.5 A Circuit for Quantum Fourier Transform	38
4.6 Simon's Algorithm for Phase Estimation	41
5 Conclusion	43
Bibliography	47

List of Figures

2.1	The quantum Fredkin gate.	8
2.2	A quantum circuit black-box model	15
2.3	A quantum circuit with the second qubit set to $ 0\rangle$	16
2.4	A qubit in state $ 0\rangle$, applied to a Hadamard gate	16
2.5	The transformation $U_f : xy\rangle \rightarrow x(y \oplus f(x))\rangle$ performed by the quantum gate array.	17
3.1	Classical and quantum parallelism.	23
3.2	A quantum circuit for solving Deutsch's problem.	24
4.1	The relationship between integer factorization, order finding, phase estimation, and Quantum Fourier Transform.	29
4.2	The cycle for computing the order k for an integer $x \bmod P$	30
4.3	The multiplicative cycle for $11 \bmod 21$	31
4.4	Flowchart of the factorization algorithm for $P = 21$ and $q = 11$	34
4.5	A circuit for Quantum Fourier Transform.	40
4.6	A quantum circuit for phase estimation	42

Chapter 1

Introduction

Cryptography, in Greek meaning “code-making”, together with cryptanalysis, meaning “codebreaking”, form the science of cryptology. Cryptology, first employed by the Spartans in 400 B.C., is the science of secret communication.

Cryptography uses methods that allow for secret communication between parties. The plain text is encrypted, respectively decrypted into a cipher text using a certain key. The communication is secret as long as an eavesdropper has no information on the message.

The goal of cryptanalysis is to overcome cryptographic systems and reveal the contents of encrypted messages, even if the cryptographic key is unknown.

Communication and computer systems encoding information in the spin of electrons or polarization of photons are called quantum computers. Quantum means “some quantity” in Latin. The term has been employed in physics with the same meaning of the word “discrete” in mathematics: it refers to a quantity that can take only sharply defined values, as opposed to a continuously varying quantity.

Quantum systems have strange and, at the same time, remarkable properties. There are several concepts associated with quantum mechanics that are very difficult to accept.

One of these concepts is the non-determinism, or the Heisenberg’s Uncertainty Principle: the position of a quantum particle and its velocity cannot be measured simultaneously. This is because measuring the property of the quantum particle corresponds to applying a linear operator to the vector describing the state of the system. A linear operator in quantum mechanics has a matrix associated with it and, because the product of matrices is non-commutative, the order of measurement is important.

Another one of these concepts is the non-locality: the fact that two quantum

objects can influence one another, even when separated by a large distance, and that the change of state of one determines an instantaneous change of state of the other.

Quantum computers can have a role both in cryptography, by using quantum states as information carriers, and in cryptanalysis, by using quantum algorithms that can factor large numbers very efficiently, thus potentially being able to break the RSA, the most used cryptosystem in the world.

The RSA is based on the use of one-way functions, as proposed by Diffie and Hellman in 1976 [DH76]. The RSA uses two different keys: a public one and a private one. The public key is announced to everybody, and the private key is kept secret. The plaintext is converted in ciphertext using a one-way function. This function can be evaluated easily, but is hard to invert. The inversion can easily be done using the private key. Thus, the plaintext can be recovered from the ciphertext by the owner of the private key.

The RSA system was invented by Rivest, Shamir, and Adleman in 1978 [RSA78] and it is based on the difficulty of factoring large numbers. The RSA system works as follows: Let $M = n_1 \cdot n_2$ the product of two prime numbers n_1 and n_2 . Let $\phi(M) = (n_1-1)(n_2-1)$ be the Euler function of M , choose a such that $1 < a < \phi(M)$ and $\gcd(a, \phi(M)) = 1$, and calculate p such that $a \cdot p = 1 \bmod \phi(M)$. The numbers M and a then form the public key, whereas p is the private key. A message S gets encoded into $C = S^a \bmod M$. Since $x^{ap} = x \bmod M$, we can decode by calculating $C^p \bmod M = S$. As an example, take $n_1 = 7$ and $n_2 = 11$, thus $M = 77$ and $\phi(M) = 6 \times 10 = 60$. Choosing the public key $a = 13$ (together with $M = 77$) we find the private key $p = 37$.

Chapter 2

Quantum Information

2.1 Basic Quantum Information Theory

The theory of quantum information is involved with the communication and processing of quantum states.

Quantum system store information using qubits. A qubit can be in any state that is a linear combination of the basis states $|0\rangle$, represented as \ominus , and $|1\rangle$, represented as \oplus , called computational basis states. For example, in case the physical implementation uses photons, the basis states can be horizontal polarization ($|0\rangle$ represented as \ominus) and vertical polarization ($|1\rangle$ represented as \oplus), or up spin and down spin in case the physical implementation uses electrons. Imagine the possible states a qubit can be in, represented as $\ominus \oplus \oplus \oplus \oplus \oplus \oplus \oplus \oplus \oplus \oplus \oplus \oplus \oplus \oplus \oplus$.

The result of measuring a qubit is non-deterministic and the measurement alters its value. In contrast, a regular bit can only be in the states 0 or 1, and we can measure its value with certainty and without affecting its value.

Definition 2.1. *Hilbert space: a complex vector space with a finite dimension n . We will note the Hilbert space by \mathcal{H}^n .*

Definition 2.2. *The qubit, or quantum bit, is the unit of quantum information.*

Consider $\{|0\rangle, |1\rangle\}$ to be an orthonormal basis of \mathcal{H}^2 , the Hilbert space we define our qubit in. Then we can represent the qubit as a normalized vector in \mathcal{H}^2 : $|\omega\rangle = m|0\rangle + n|1\rangle$. Coefficients m and n are the probability amplitudes of the state and are complex numbers, such that $|m|^2 + |n|^2 = 1$. The $|0\rangle$ can be represented as a matrix

$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and the $|1\rangle$ can be represented as a matrix $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$.

When we measure a qubit, we actually project the vector on one of the basis vectors chosen randomly depending on the amplitudes. The qubit must be normalized so that the probabilities sum to one.

The outcome of a measurement of a qubit with the state $|\psi\rangle = a_0|0\rangle + a_1|1\rangle$ is 0 with probability $|a_0|^2$ and 1 with probability $|a_1|^2$. We say the qubit is in superposition of these two states, meaning that it is both 0 and 1 at the same time.

Law 3.1: Qubit measurement (collapsing): Once we measured the state of a qubit and the result reads $|0\rangle$ (or $|1\rangle$), then repeated measurements will give the same result. We say the state collapses to $|0\rangle$ (or $|1\rangle$).

To do useful computation we need more than one qubit. If we consider a system consisting of n qubits, the possible observed states of such a system form a vector space 2^n . That is because for each qubit we have two orthogonal basis. There are 2^n basis states forming a computational basis and there are superposition states resulting from the superposition of these basis states. The state of an n -qubit system is given by the tensor product of the individual state spaces.

Definition 2.3. Quantum register: A quantum register is a grouping of n qubits $|\phi_1\rangle \dots |\phi_n\rangle$.

Its classical analogue is a processor register.

We can express the state of the register $|\psi\rangle$ as a tensor product:

$$|\psi\rangle = |\phi_1\rangle \otimes |\phi_2\rangle \dots \otimes |\phi_n\rangle$$

or as a linear combination of the basis vectors $|0\rangle, \dots, |2^n - 1\rangle$:

$$|\psi\rangle = \sum_{i=0}^{2^n-1} a_i |i\rangle$$

with the restriction that

$$\sum_{i=0}^{2^n-1} |a_i|^2 = 1$$

It is important to emphasize that the qubits in the register may be in an entangled state.

For example, if we group two qubits, then the resulting space will be $\mathcal{H}^4 = \mathcal{H}^2 \otimes \mathcal{H}^2$. So let's say we have two qubits, one represented by the vector $|\alpha\rangle = a_0|0\rangle + a_1|1\rangle$, and the other qubit represented by the vector $|\beta\rangle = b_0|0\rangle + b_1|1\rangle$. The resulting product state is

$$|\alpha\beta\rangle = a_0b_0|00\rangle + a_0b_1|01\rangle + a_1b_0|10\rangle + a_1b_1|11\rangle$$

For example, it yields 00 with probability $|a_0b_0|^2$. We can see this product state is normalized since

$$\sqrt{(a_0b_0)^2 + (a_0b_1)^2 + (a_1b_0)^2 + (a_1b_1)^2} = \sqrt{a_0^2(b_0^2 + b_1^2) + a_1^2(b_0^2 + b_1^2)} =$$

$$\sqrt{(a_0^2 + a_1^2)(b_0^2 + b_1^2)} = \sqrt{(a_0^2 + a_1^2)}\sqrt{(b_0^2 + b_1^2)} = 1$$

But if a transformation is applied to the state, then separation of the states may no longer be possible. In other words we cannot represent it by a tensor product of qubits - we say the qubits are in an entangled state and we present this concept later on in this chapter.

Definition 2.4. *Unitary transformation: A unitary transformation is a unitary operator acting on a vector space \mathcal{H}^n .*

Unitary means that the following properties are satisfied:

$$1) U : \mathcal{H}^n \longrightarrow \mathcal{H}^n$$

$$2) \langle\alpha|\beta\rangle = \langle U\alpha|U\beta\rangle$$

$$3) \exists U^{-1} \ni UU^{-1} = I$$

where $\langle\alpha|\beta\rangle$ is the inner product in \mathcal{H}^n .

Theorem 2.5. *Unitary transformations form a group under composition.*

Indeed, if U and V are unitary transformations then UV is also surjective and

$$\langle UV\alpha|UV\beta\rangle = \langle V\alpha|V\beta\rangle = \langle\alpha|\beta\rangle$$

for every $\alpha, \beta \in \mathcal{H}^n$. Also, $(UV)^{-1} = V^{-1}U^{-1}$.

Hence UV is also a unitary transformation.

We will represent unitary transformations using unitary matrices.

Definition 2.6. *A unitary matrix is a square complex-valued matrix, A , whose inverse is equal to its conjugate transpose:*

$$A^{-1} = \bar{A}^t$$

Quantum computers are comprised of quantum circuits which are built using quantum gates. Quantum logic gates perform unitary operations and therefore are represented by unitary matrices.

Definition 2.7. *Qubit operation: A single qubit operation is the operation performed by a quantum gate on a single qubit.*

Definition 2.8. *Quantum algorithm: A quantum algorithm is a sequence of unitary operations on the vector $|\psi\rangle$.*

Definition 2.9. *Walsh-Hadamard transform: The one qubit Walsh-Hadamard transform determines a rotation on the basis vectors defined by:*

$$H : \begin{aligned} |0\rangle &\longrightarrow \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\ |1\rangle &\longrightarrow \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \end{aligned}$$

H transforms a qubit $|\psi\rangle = a_0|0\rangle + a_1|1\rangle$ in the following way:

$$H|\psi\rangle = a_0H|0\rangle + a_1H|1\rangle = \frac{a_0}{\sqrt{2}}(|0\rangle + |1\rangle) + \frac{a_1}{\sqrt{2}}(|0\rangle - |1\rangle) = \frac{a_0 + a_1}{\sqrt{2}}|0\rangle + \frac{a_0 - a_1}{\sqrt{2}}|1\rangle$$

Its corresponding unitary matrix is: $\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$. The most important role of the Walsh-Hadamard gates is the fact that it can take a qubit in any state and transform the state into a superposition state, with equal probability amplitudes for $|0\rangle$ and $|1\rangle$.

Definition 2.10. *CNOT operation: A CNOT (controlled NOT) operation is performed by a CNOT gate. The CNOT gate switches the second qubit (the target qubit) to its opposite state if and only if the first qubit (the control qubit) is $|1\rangle$. The operation performed by this gate is similar to the one performed by a classical XOR gate.*

The operation of the CNOT gate can be modeled by the following chart:

Before	After
$ 0, 0\rangle$	$ 0, 0\rangle$
$ 0, 1\rangle$	$ 0, 1\rangle$
$ 1, 0\rangle$	$ 1, 1\rangle$
$ 1, 1\rangle$	$ 1, 0\rangle$

The CNOT operation is given by the matrix:

$$C = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

CNOT is a unitary operation because the inverse of its matrix C is equal to its conjugate transpose:

$$C^{-1} = \frac{1}{\det(C)} \text{adj}(C) = \frac{1}{-1} \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

and

$$\bar{C}^t = C^t = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Let $|\psi\rangle$ be the state of a quantum register formed by the qubits $|\alpha\rangle = a_0|0\rangle + a_1|1\rangle$ and $|\beta\rangle = b_0|0\rangle + b_1|1\rangle$:

$$|\psi\rangle = |\alpha\beta\rangle = a_0b_0|00\rangle + a_0b_1|01\rangle + a_1b_0|10\rangle + a_1b_1|11\rangle =$$

$$a_{00}|00\rangle + a_{01}|01\rangle + a_{10}|10\rangle + a_{11}|11\rangle$$

By applying the CNOT operation to this register we obtain:

$$C|\psi\rangle = C|\alpha\beta\rangle = a_{00}|00\rangle + a_{01}|01\rangle + a_{11}|10\rangle + a_{10}|11\rangle$$

If the target qubit is $|\beta\rangle = |0\rangle = 1|0\rangle + 0|1\rangle$, then the quantum register is

$$|\psi\rangle = |\alpha\beta\rangle = a_01|00\rangle + a_00|01\rangle + a_11|10\rangle + a_10|11\rangle = a_0|00\rangle + a_1|10\rangle$$

and the CNOT generates the state

$$C|\psi\rangle = a_0|00\rangle + a_1|11\rangle$$

and in this way the CNOT gate copied the superposition of the control qubit to the target.

One-qubit operations and the CNOT operation are called elementary unitary transformations. Dan C. Marinescu [MM05] shows how we only need these elementary unitary transformations in order to construct a universal quantum computer.

Definition 2.11. *Fredkin gate: The Fredkin gate is a three-qubit quantum gate with two target inputs, a , b , and a control input, c . The gate has three outputs, a' , b' , and c' .*

The input of the Fredkin gate determines the output as follows:

- The control input c is transferred directly to the output, $c' = c$, see Figure 2.1(a)
- When $c = 0$, the two target inputs are transferred without modification to the output: $a' = a$ and $b' = b$, see Figure 2.1(b).
- When $c = 1$, the two target inputs are swapped: $a' = b$ and $b' = a$, see Figure 2.1(c)

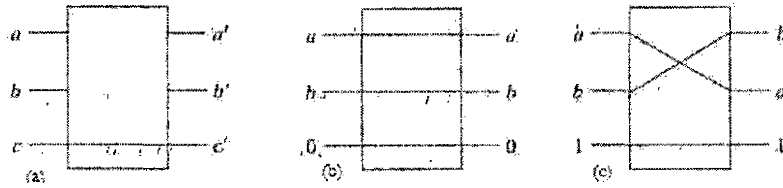


Figure 2.1: The quantum Fredkin gate.

A system of three qubits requires an eight-dimensional complex vector space with a basis consisting of eight vectors: $|000\rangle$, $|001\rangle$, $|010\rangle$, $|011\rangle$, $|100\rangle$, $|101\rangle$, $|110\rangle$, and $|111\rangle$.

Each of the basis vectors in \mathcal{H}_8 is a tensor product of three basis vectors in \mathcal{H}_2 :

$$\begin{aligned}
 |000\rangle &= \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} & |001\rangle &= \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} & |010\rangle &= \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} & |011\rangle &= \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}
 \end{aligned}$$

$$\begin{aligned}
|100\rangle &= \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} & |101\rangle &= \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} & |110\rangle &= \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} & |111\rangle &= \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}
\end{aligned}$$

when $c = |1\rangle$, a and b are switched, and the mapping of the Fredkin gate's input to output is as follows:

$$|000\rangle \mapsto |000\rangle$$

$$|001\rangle \mapsto |001\rangle$$

$$|010\rangle \mapsto |010\rangle$$

$$|011\rangle \mapsto |101\rangle$$

$$|100\rangle \mapsto |100\rangle$$

$$|101\rangle \mapsto |011\rangle$$

$$|110\rangle \mapsto |110\rangle$$

$$|111\rangle \mapsto |111\rangle$$

The transfer matrix of the Fredkin gate is obtain as follows:

$$\begin{aligned}
G_{Fredkin} &= |000\rangle\langle 000| + |001\rangle\langle 001| + |010\rangle\langle 010| + |011\rangle\langle 101| + \\
&\quad + |100\rangle\langle 100| + |101\rangle\langle 011| + |110\rangle\langle 110| + |111\rangle\langle 111|
\end{aligned}$$

And therefore

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

Let $|V_{Fredkin}\rangle$ be the input state of a Fredkin gate:

$$|V_{Fredkin}\rangle = a_{000}|000\rangle + a_{001}|001\rangle + a_{010}|010\rangle + a_{011}|011\rangle + a_{100}|100\rangle + a_{101}|101\rangle + a_{110}|110\rangle + a_{111}|111\rangle.$$

The output state of a Fredkin gate, $|W_{Fredkin}\rangle$ is given by:

$$|W_{Fredkin}\rangle = G_{Fredkin}|V_{Fredkin}\rangle$$

The Fredkin gates have the following properties:

- they are reversible;
- they conserve the number of 1s at their input making them a conservative logic gate;
- they can simulate both AND and NOT gates, making them universal gates.

In order to implement “if-then-else” constructs, there were devised quantum circuits who can execute a unitary transformation G function of the state of a “decision” qubit. The most common of these single-qubit controlled transformations are: the controlled- H , the controlled- Z , the controlled- P_α , and the controlled- R_k .

The controlled- R_k will be used later on in Chapter 4 so we are presenting its definition below. When we apply a transformation on a qubit, its angle θ about the three axes changes, as well. The controlled- R_k gate implements such a phase-shift transformation.

Definition 2.12. *Controlled- R_k gate: The controlled- R_k gate implements the operation given by:*

$$R_k = \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{pmatrix}$$

The controlled- R_k gate maps its input to the output as follows:

$$\begin{aligned} |00\rangle &\mapsto |00\rangle \\ |01\rangle &\mapsto |01\rangle \\ |10\rangle &\mapsto |10\rangle \\ |11\rangle &\mapsto e^{2\pi i/2^k} |11\rangle \end{aligned}$$

The transfer matrix of the controlled- R_k gate is given by:

$$G_{\text{controlled-}R_k} = |00\rangle\langle 00| + |01\rangle\langle 01| + |10\rangle\langle 10| + e^{2\pi i/2^k} |11\rangle\langle 11|$$

And therefore

$$G_{\text{controlled-}R_k} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{2\pi i/2^k} \end{pmatrix}$$

Definition 2.13. *Entanglement: Two or more qubits are entangled if the state of the register formed by them cannot be expressed as a tensor product.*

For example, consider a CNOT gate with the control qubit $|\alpha\rangle = 1|0\rangle + 0|1\rangle$ and the target qubit $|\beta\rangle = b_0|0\rangle + b_1|1\rangle$. Then state of the CNOT gate will be $b_0|00\rangle + b_1|01\rangle = |0\rangle \otimes (b_0|0\rangle + b_1|1\rangle)$. This state is separable and the two qubits are not entangled because the state can be written as a tensor product.

If we consider a similar situation but with the control qubit $|\alpha\rangle = a_0|0\rangle + a_1|1\rangle$ and the target qubit $|\beta\rangle = 1|0\rangle + 0|1\rangle$, then the state of the CNOT gate will be $a_0|00\rangle + a_1|11\rangle$ which cannot be written as a tensor product and therefore is entangled.

We will prove that $a_0|00\rangle + a_1|11\rangle$ with $a_0, a_1 \neq 0$ cannot be written as a tensor product.

Proof. Assume $\exists |\varphi_1\rangle, |\varphi_2\rangle \ni |\varphi_1\rangle \otimes |\varphi_2\rangle = \alpha|00\rangle + \beta|11\rangle, \alpha, \beta \neq 0$

Let $|\varphi_1\rangle = a_0|0\rangle + a_1|1\rangle$ and $|\varphi_2\rangle = b_0|0\rangle + b_1|1\rangle$.

Then $|\varphi_1\rangle \otimes |\varphi_2\rangle = a_0b_0|00\rangle + a_0b_1|01\rangle + a_1b_0|10\rangle + a_1b_1|11\rangle$.

Therefore $\begin{cases} a_0b_0 = \alpha, a_0b_1 = 0 \Rightarrow b_1 = 0 \\ a_1b_1 = \beta, a_1b_0 = 0 \Rightarrow b_0 = 0 \end{cases} \Rightarrow \alpha = 0 \text{ and } \beta = 0$ which is a contradiction of our hypothesis.

□

We can conclude that, by applying the CNOT operation to the quantum register $|\psi\rangle$, returned as an output the entangled superposition of the control and the target. In other words, the output of the CNOT gate cannot be expressed as a tensor product.

The fact that a quantum system is entangled means there are reciprocal relationships between the subsystems. For example, suppose that $|\Psi\rangle$ represents the state of two qubits, one of which is in Los Angeles and the other one is in New York. Since the qubits are entangled, their individual states cannot be written as vectors in \mathcal{H}^2 . If we measure the qubit in Los Angeles, we will obtain a random answer (0 or 1), but we will be sure that if someone measures the other qubit, he will obtain the opposite answer. Although the answer we receive is random, the other answer is always its opposite. What we do with one part of the system influences the other part. This will prove very useful for computation.

Law 2: Evolution: The evolution in time of a quantum system is unitary. In other words, in a quantum system, for any event, the sum of probabilities of all possible outcomes is equal to 1.

2.2 How a Quantum Computer Works

A classical computer evolves from an input state, goes through a number of intermediate states, and arrives at an output state. Throughout its evolution, these states are all known. For example, a server might be in a state of IDLE, CONNECTED, BUSY, ERROR, etc. This classical computing engine always returns the same result any time a specific set of input values is present. We call this behaviour “deterministic”.

A quantum computer is a quantum system. As we have seen earlier, the quantum system is a generalization of the qubit to the n -states and its state is represented by:

$$|Y\rangle = a_0|0\rangle + a_1|1\rangle \dots + a_i|i\rangle \dots + a_{n1}|n1\rangle$$

, such that $\sum_{i=0}^{n-1} |a_i|^2 = 1$

This means the states of the quantum computer are not known. We call this behaviour “stochastic”.

The output states of a stochastic engine are random and the value of the output state cannot be discovered. All we can do is label a set of pairs consisting of an output state of an observable and a measured value of that observable.

Then the quantum computation works as follows: we prepare a quantum register in a known state, we apply quantum gates on the register, i.e. we apply unitary operations

on some qubits in a precise order, and we measure the final state of the register to learn its content. It is important to note that a quantum computation is probabilistic in nature, because although the evolution of the states is deterministic, the measurement step gives probabilistic answers.

One important result is that every classical computable function can be implemented by unitary transformations[MM05]. This means a quantum computer can perform any function a classical computer does, and more. As we will see in Chapter 3.1, there is no significant increase in the time (time complexity) or number of operations required to compute those functions with a quantum computer. An example of something a quantum computer can do and a classical one cannot would be the implementation of a fair coin toss using a one qubit computation. First we prepare our qubit in state $|0\rangle$. We then input this in a Walsh-Hadamard gate:

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

which yields the superposition. A measurement on this resulting superposition state of the the qubit in the standard basis would produce a $|0\rangle$ or a $|1\rangle$ with equal probability. The result we obtain is a true random number. It would be impossible to generate such a number using a classical computer.

Quantum mechanics imposes limitations on what we can do with qubits. Two important theorems follow:

Theorem 2.14. *There is no perfectly accurate measurement of amplitudes. Let $|\psi\rangle = a_0|0\rangle + a_1|1\rangle$ an arbitrary qubit. Measuring the qubit will not allow us to tell its probability amplitudes a_0 and a_1 .*

In other words measuring the qubit will not allow us to tell that it is in the state, say $\frac{1}{2}|0\rangle + \frac{\sqrt{3}}{2}|1\rangle$. We can only distinguish between orthogonal states: if we know the qubit is in one of the orthogonal states $|\alpha\rangle$ or $|\beta\rangle$ (such that $\langle\alpha|\beta\rangle = 0$); then we could apply a measurement to tell in which one of those two states the qubit is.

Theorem 2.15. *No cloning: Given an arbitrary qubit $|\psi\rangle = a_0|0\rangle + a_1|1\rangle$, there does not exist an operator A and a state $|\alpha\rangle \ni A(|\psi\rangle \otimes |\alpha\rangle) = |\psi\rangle \otimes |\psi\rangle$. This implies that we can't clone arbitrary quantum states.*

Definition 2.16. *Quantum interference: In quantum computing different parallel computations can interfere, making the path of certain computations stronger, while the path of others weaker. This can cause some answers to become more likely than others.*

For example, suppose we have the unitary transformation U :

$$U = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}$$

and we apply this transformation to a qubit in the state $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$.

We obtain

$$U|1\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \frac{1}{\sqrt{2}} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle)$$

meaning the results $|0\rangle$ or $|1\rangle$ have an equal probability. But applying this transformation twice:

$$UU|1\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 2 \\ 0 \end{pmatrix} = |0\rangle$$

and this means the output will certainly be $|0\rangle$.

Similarly

$$UU|0\rangle = |1\rangle$$

The conclusion is that when we apply U once we obtain a stochastic answer but applying U twice gives a deterministic answer: the negation of the input.

In many quantum algorithms, interference plays a crucial role. The “good” computational paths interfere constructively and the “bad” ones interfere destructively so they will not be encountered.

Interference, together with entanglement and parallelism play a major role in quantum computing.

Let x be a binary string of length n . Then there are 2^n possible values of x . In order to compute a function $f(x)$ using a classical circuit we can either :

1. Use a sequence of the same circuit and obtain all the values of the function after 2^n time steps (exponential time), or
2. Or, use 2^n copies of the circuit, each with a different input x , and obtain all the values in a single time step (exponential resources).

For example, let us consider a function $f : \{0, 1\} \rightarrow \{0, 1\}$. To calculate its output in a standard fashion we would have to perform $f(0)$ and $f(1)$, separately.

Definition 2.17. *Quantum parallelism: Quantum circuits can perform $f(|0\rangle + |1\rangle) = |f(0)\rangle + |f(1)\rangle$ in one step, using what is called quantum parallelism, a behavior based on the superposition principle and entanglement.*

We will see below how quantum circuits have the ability to calculate the all the values of the function $f(x)$ for all possible values of the input x , in one time increment and using only one instance of a circuit.

Let us see an example for a two qubit computation.

We will use a black-box approach.

Let $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$ be a function we would like to obtain information about. In order to do so f is associated with a black-box U_f .

Definition 2.18. *The black-box model: In this model a gate is considered a black-box in the sense that we don't know how it works internally. All we know is that it takes $x \in \{0, 1\}^n$ as input, and outputs $f(x)$.*

As it is, this function is not generally reversible (consider for instance the all-0 function). A quantum circuit needs to be reversible. In order to make the function reversible we use another input $y \in \{0, 1\}^m$. In this way the black-box U_f outputs x and $f(x) \oplus y$. Using this extra input U_f is then a permutation of all $(n + m)$ -bit strings, and therefore reversible[BFFP10] Now consider the circuit shown in the Figure 2.2 below used by Peter Shor as a quantum circuit black-box model[Sho02]. This circuit takes an input not in the computational basis and applies U_f to it. As a note, the function f is hardwired into the circuit using Fredkin gates. The circuit takes as input the two qubits $|x\rangle$ and $|y\rangle$ and outputs $|x\rangle$ and $|y \oplus f(x)\rangle$.

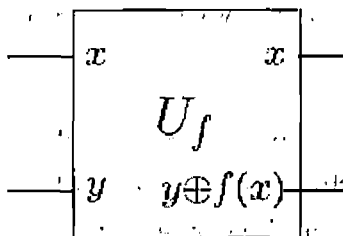


Figure 2.2: A quantum circuit black-box model

Therefore $U_f : |xy\rangle \rightarrow |x(y \oplus f(x))\rangle$

If the second qubit is set to $|0\rangle$, as seen in Figure 2.3, then the transformation carried out by the circuit is

$$|x0\rangle \rightarrow |xf(x)\rangle.$$

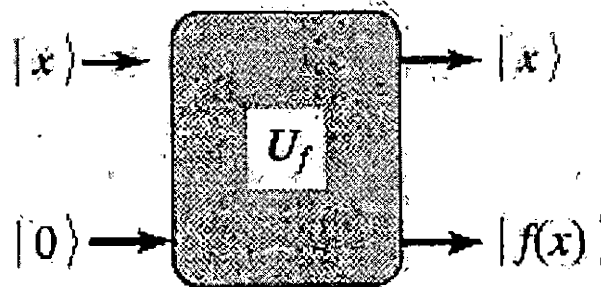


Figure 2.3: A quantum circuit with the second qubit set to $|0\rangle$

Let us take a qubit in state $|0\rangle$, apply it to a Hadamard gate, which will output the superposition state $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, and apply this to the input $|x\rangle$, all while the input $|y\rangle$ is set to $|0\rangle$, as in Figure 2.4.

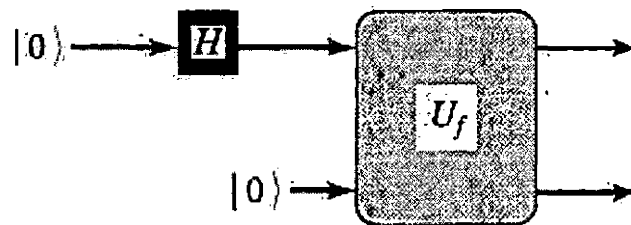


Figure 2.4: A qubit in state $|0\rangle$, applied to a Hadamard gate

As a result, the two output qubits of the circuit are:

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

and

$$f\left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) = \frac{|f(0)\rangle + |f(1)\rangle}{\sqrt{2}}$$

The readout state of the quantum black-box is:

$$\left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) \otimes f\left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) = \frac{|0f(0)\rangle + |0f(1)\rangle + |1f(0)\rangle + |1f(1)\rangle}{2}$$

As you can see, we have evaluated $f(x)$ for both $f(0)$ and $f(1)$ concurrently. The problem is that we cannot extract both answers. If we measure the resulting qubit, we will obtain the answer $f(|0\rangle)$ or $f(|1\rangle)$ at random because $f(|0\rangle)$ and $f(|1\rangle)$ are orthogonal.

Let us see an example with three qubits, where we have a two qubit register $|x\rangle$, a one qubit $|y\rangle = |0\rangle$, and would like to compute $f(|x\rangle)$. We prepare $|x\rangle$ formed by the 2 qubits initially set at $|0\rangle$ using Hadamard gates. This gives us $H|x\rangle = \left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right) \left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right) = \frac{|00\rangle+|01\rangle+|10\rangle+|11\rangle}{2}$ as output from the Hadamard gates. Then $f(|x\rangle) = f\left(\frac{|00\rangle+|01\rangle+|10\rangle+|11\rangle}{2}\right) = \frac{|f(00)\rangle}{2} + \frac{|f(01)\rangle}{2} + \frac{|f(10)\rangle}{2} + \frac{|f(11)\rangle}{2}$.

This result can be extended to an input consisting of say m qubits. For this we need a quantum circuit to transform an m -dimensional vector $|x\rangle$ into $|f(x)\rangle$.

Figure 2.5 illustrates a quantum gate array performing a linear transformation U_f to the input $|x\rangle \in \mathcal{H}_{2^m}$, a 2^m -dimensional vector acting as a control input, and $|y\rangle \in \mathcal{H}_{2^k}$ a 2^k -dimensional vector acting as a work qubit.

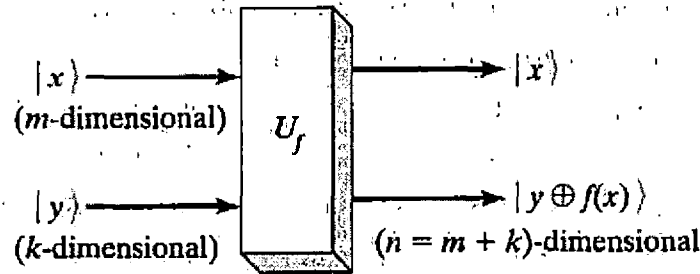


Figure 2.5: The transformation $U_f : |xy\rangle \rightarrow |x(y \oplus f(x))\rangle$ performed by the quantum gate array.

As we have seen in the examples above, the m qubits in the register $|x\rangle$ have to be first brought to a superposition state using m Hadamard gates. Typically, we start with these m qubits in state $|0\rangle$. Then, each $|0\rangle$ qubit is transformed by a Hadamard gate as follows:

$$H : |0\rangle \rightarrow \frac{|0\rangle + |1\rangle}{\sqrt{2}}$$

By applying an array of m Hadamard gates we obtain:

$$\begin{aligned} (H \otimes H \dots \otimes H) |00 \dots 0\rangle &= \left[\frac{(|0\rangle + |1\rangle) \otimes (|0\rangle + |1\rangle) \dots \otimes (|0\rangle + |1\rangle)}{(\sqrt{2})^n} \right] = \\ &= \frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} |j\rangle \end{aligned}$$

and this result is expressed as a linear combination of 2^n vectors forming an orthonormal basis in \mathcal{H}_{2^n} . The gate array performs a linear transformation and it generates a result in the form of a superposition state. The outputs are $|x\rangle$ and $|y \oplus f(x)\rangle \in \mathcal{H}_{2^n}$, with $n = m + k$. When $|y\rangle = |0\rangle$, the second output becomes

$$\begin{aligned} |y \oplus f(x)\rangle &= |f(x)\rangle = \\ &= U_f \left(\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x 0\rangle \right) = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} U_f(|x 0\rangle) = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} (|x f(x)\rangle) \end{aligned}$$

In this way we calculated the function $f(x)$ for its entire 2^n -dimension domain in a single step and this is what is called quantum parallelism.

However, this parallel computation does not seem to be useful because when we measure the output of the quantum gate array we can only observe one value. We need the capability to extract from the superposition state $\sum_x |x, f(x)\rangle$ more than one value of $f(x)$. We will see how this is done using the Deutsch's Algorithm in Chapter 3.

2.3 Shor's Algorithm

In 1994, Peter Shor[Sho97] published a quantum algorithm that factors integers in polynomial time. This discovery drew a lot of attention to the field of quantum computation because the factoring problem is at the core of modern cryptography. Many cryptographic systems base their security on the assumption that the factoring problem is difficult. Shor's algorithm could break such systems very effectively. The subject of this Thesis is this algorithm and we will see how it functions in Chapter 4.

Chapter 3

The Rationale for Using a Quantum Computer

3.1 The Quantum Turing Machine

The Church-Turing principle is stating that all computing devices can be simulated by a Turing Machine. This principle has major implications for the computability theory. It means that in order to study computability, instead of investigating a potentially infinite set of physical computing devices, it is sufficient to restrict ourselves to a single abstract model, the Turing Machine.

Definition 3.1. *Turing Machine: A Turing Machine (TM) is an abstract computing device with a finite number of internal states, a read-write head (the modern memory bus), and an infinite moving tape (the modern memory) organized in individual cells with each cell containing a symbol.*

The Turing Machine starts in a certain state, scans the symbol currently under the head and, function of the internal state of the machine and the current symbol, it may either erase that symbol and replace it with another symbol, or leave it as is, and then move the tape left or right and change its internal state. Let Q_i and S_i denote the state and the symbol read at time t . Let D the direction of the movement and Q_j , S_j the new state and the symbol written. Then the relationship between these can be expressed as

$$Q_j = F(Q_i, S_i),$$

$$S_j = G(Q_i, S_i)$$

$$D = D(Q_i, S_i)$$

The Turing Machine is completely defined by the original tape and the set of quintuples (Q_i, S_i, Q_j, S_j, D) .

As Alan Turing describes it in 1936[Tur], there is a Universal Turing Machine (UTM) able to simulate any other Turing Machine.

In order to do so, we input into the Universal Turing Machine the set of quintuples and the tape describing the original Turing Machine together with the indication where to start and where to end.

To establish if a function $f(x)$ is Turing-computable or not, we have to find a TM able to carry out the computation prescribed by the function $f(x)$. If such a machine exists we can be assured that the function is computable.

Definition 3.2. *Turing-computable functions: A function $f(x)$ is Turing-computable if there is a Turing Machine T_f with a tape containing a description of x and which will eventually halt with a description of $f(x)$ on tape.*

Being able to automate the solving of a mathematical problem consists in finding a Turing Machine able to carry out that same computation.

Definition 3.3. *A Probabilistic Turing Machine (PTM) is a Turing Machine with an extra instruction that allows it to randomly choose an execution path. A PTM can have stochastic results such that a computation performed several times on a given input may have different run times, or it may not go through at all.*

A classical computation performed by a PTM can be represented by a probability tree diagram with nodes for each state, and branches connecting parent and child states, each branch with its probability. Because of the stochastic nature of PTM, the same state may appear in the probability tree multiple times, which we will call state instances. The probability of a state instance is given by the product of probabilities assigned to all branches leading from the root to the node associated with that instance state. The probability of a state is the sum of the probabilities of all the instances of that state. The sum of the probabilities of all states at any given level of the tree is equal to one.

Definition 3.4. *Quantum Turing Machine: A Quantum Turing Machine (QTM) is a PTM in which the classical laws of probabilities are substituted by quantum systems laws [Sim97].*

A computation on a QTM is represented by a computational tree where each branch has an associated probability amplitude. The probability amplitude of an instance

state is given by the product of the probability amplitudes assigned to all branches leading from the root to the node associated with that instance state. The probability amplitude of a state at any level is the sum of the probability amplitudes of all the instances of that state at that level. The probability of an instance of a state is the square of the probability amplitude of the corresponding node. For example, the probability of a particular final state is the square of the sum (not the sum of the squares) of all leaf nodes corresponding to that state [Sim97]. The evolution of a QTM must be unitary and reversible.

3.2 Why Use Quantum Computers

There are two reasons why we would want to use quantum computers instead of classical ones in solving certain types of mathematical problems. One reason has to do with computational complexity. The other reason is the possibility of using quantum parallelism.

Computational complexity classifies problems into classes of complexity. Computational complexity takes into account two major complexities: resource complexity and time complexity. The time complexity of an algorithm, expressed as \mathcal{O} , depends on the size of the input to the problem and represents the run time of that algorithm. Thus, time complexity represents a measure of the performance of an algorithm and represents the totality of elementary operations performed by the algorithm. In a quantum computer each elementary unitary transformation is counted as one operation [1]. On a conventional computer performing a unitary operation requires more than one arithmetic operation. Time complexity has an asymptotic representation. For example, if an algorithm presented with inputs of size n requires a run time of $T(n) = 7n^5 + 2n$, the asymptotic time complexity is $\mathcal{O}(n^5)$.

Depending on the type of the function $T(n)$, an algorithm with time complexity $\mathcal{O}(n)$ is called a polynomial time algorithm, and an algorithm with time complexity $\mathcal{O}(2^n)$ is called an exponential time algorithm. A problem is considered relatively simple if it uses polynomial time and difficult if it uses exponential time.

Computational complexity theory considers several complexity classes. For our thesis we will consider just two of these classes:

\mathcal{P} : The complexity class of decision problems that can be solved on a deterministic Turing machine in polynomial time.

\mathcal{NP} : The complexity class of decision problems that can be solved on a non-deterministic Turing machine in polynomial time.

The complexity class $\mathcal{P} \subset \mathcal{NP}$. \mathcal{NP} contains something called \mathcal{NP} -complete problems. There are no known polynomial-time algorithms capable of solving this type of problems.

C.M.Papadimitriou launched the hypothesis that any physical computing device can be simulated by a Turing Machine in a number of steps polynomial in the resources used by the computing device[Pap94]. No counter-examples have been found amongst the classical mechanics-based systems. However, counter-examples seem to exist amongst quantum-based systems [121]. According to this theory, \mathcal{NP} problems such as factoring integers, or finding discrete logarithms could be solved in polynomial time by a quantum computing system. This means that we are well motivated to investigate quantum computing devices and quantum algorithms. It is mainly the phenomenons of entanglement and quantum parallelism, characteristic to quantum systems, that can make all of this computing power possible [Joz98].

There is also the question if we can emulate the behavior of a quantum system using a classical system. Research [Joz98] shows that it may be possible for the 2^n modes of a classical system to emulate the behavior of n qubits and exhibit entangled states, but we would need to expend an exponential amount of energy to emulate the behavior of entangled quantum particles.

3.3 Deutsch's Problem

Imagine a black box circuit performing a hardwired function $f(x)$ with the possible inputs to the blackbox being $x = 0$ or $x = 1$.

We have four possible mappings:

$$f(0) = 0, f(0) = 1, f(1) = 0, f(1) = 1$$

with the assumption that the black box spends T time computing any of the mappings and that it takes no time to compare the results.

The problem posed by David Deutsch is the ability to discriminate between $f(0) = f(1)$ and $f(0) \neq f(1)$ using only one circuit and in the minimum time possible.

In order to do so we have three options.

- Two classical computing options:
 - a sequential solution: use the same circuit to calculate $f(0)$, then $f(1)$ and then compare the results, for a total time $2T$, see Figure 3.1(a).

- a parallel solution: use two copies of the circuit, with one of the circuits receiving 0 as input, the other circuit receiving 1 as input, and then compare the results for a total time T , see Figure 3.1(b).
- One quantum computing option: a quantum circuit implementing the function U_f , which has as input qubits $|x\rangle$ (control) and $|y\rangle$ (target) and produces as outputs $|x\rangle$ and $|y \oplus f(x)\rangle$, see Figure 3.1(c).

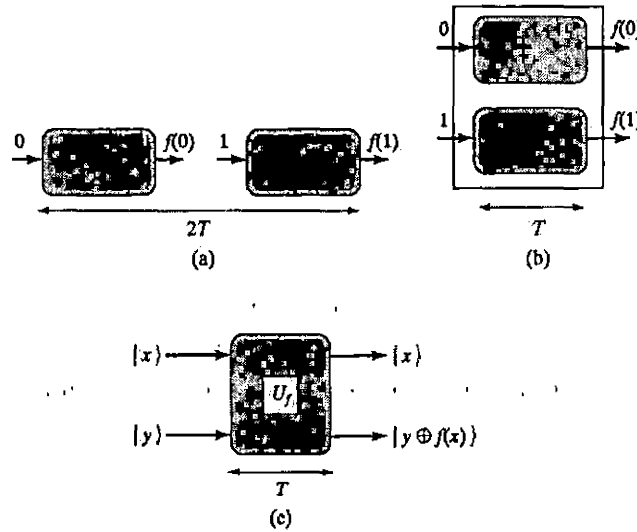


Figure 3.1: Classical and quantum parallelism.

You can see in Figure 3.2 how $|x\rangle$ is the result of preparing a $|0\rangle$ through a Haddamard gate and $|y\rangle$ is the result of preparing a $|1\rangle$ through another Haddamard thus $|x\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and $|y\rangle = \frac{|0\rangle-|1\rangle}{\sqrt{2}}$.

At each stage, the condition of the system is described by the following vectors :

- $|\omega_0\rangle$ represents the input state,
- $|\omega_1\rangle$ represents the input after it was prepared by Hadamard gates,
- $|\omega_2\rangle$ represents the output state of the black box implementing U_f ,
- $|\omega_3\rangle$ represents the output state of the circuit.

The input vector is

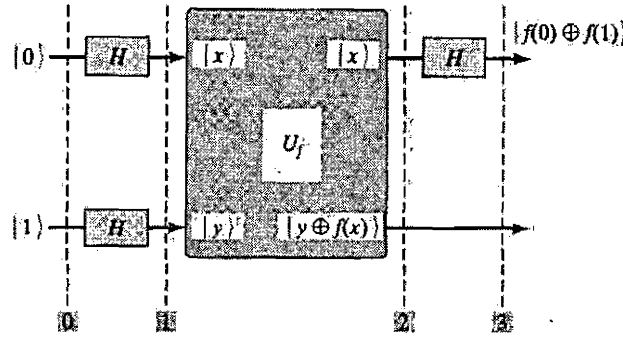


Figure 3.2: A quantum circuit for solving Deutsch's problem.

$$|\omega_0\rangle = |01\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}.$$

The transfer matrix of the Hadamard gates is

$$G_1 = H \otimes H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \otimes \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}$$

And if we apply this transfer matrix to $|\xi_0\rangle$ we obtain,

$$|\omega_1\rangle = G_1|\omega_0\rangle = \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \end{pmatrix}$$

or

$$|\omega_1\rangle = \frac{1}{2} (|00\rangle - |01\rangle + |10\rangle - |11\rangle) = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

Therefore the two qubits applied to the input of the black box are

$$|x\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}, \text{ and } |y\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

We know that $|0 \oplus f(x)\rangle = |f(x)\rangle$ thus:

$$|y \oplus f(x)\rangle = \frac{|f(x)\rangle - |1 \oplus f(x)\rangle}{\sqrt{2}}$$

But $|1 \oplus f(x)\rangle = |0\rangle$ when $f(x) = 1$ and $|1 \oplus f(x)\rangle = |1\rangle$ when $f(x) = 0$ thus

$$|y \oplus f(x)\rangle = \begin{cases} \frac{|0\rangle - |1\rangle}{\sqrt{2}} & \text{if } f(x) = 0 \\ -\frac{|0\rangle - |1\rangle}{\sqrt{2}} & \text{if } f(x) = 1. \end{cases}$$

And we can write this as,

$$|y \oplus f(x)\rangle = (-1)^{f(x)} \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

When $f(0) = f(1)$, the state $|\omega_2\rangle$ of the black box is:

$$|\omega_2\rangle = |x\rangle \otimes |y \oplus f(x)\rangle = \begin{cases} \left[\frac{|0\rangle + |1\rangle}{\sqrt{2}} \right] \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right] \rightarrow \frac{1}{2} \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \end{pmatrix} & \text{if } f(0) = f(1) = 0 \\ -\left[\frac{|0\rangle + |1\rangle}{\sqrt{2}} \right] \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right] \rightarrow -\frac{1}{2} \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \end{pmatrix} & \text{if } f(0) = f(1) = 1. \end{cases}$$

When $f(0) \neq f(1)$, the state $|\omega_2\rangle$ of the black box is:

$$|\omega_2\rangle = |x\rangle \otimes |y \oplus f(x)\rangle = \begin{cases} \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right] \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right] \rightarrow \frac{1}{2} \begin{pmatrix} 1 \\ -1 \\ -1 \\ 1 \end{pmatrix} & \text{if } f(0) = 0 \text{ and } f(1) = 1 \\ - \left[\frac{|0\rangle + |1\rangle}{\sqrt{2}} \right] \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right] \rightarrow \frac{1}{2} \begin{pmatrix} 1 \\ -1 \\ -1 \\ 1 \end{pmatrix} & \text{if } f(0) = 1 \text{ and } f(1) = 0. \end{cases}$$

Combining these two results we have

$$|\omega_2\rangle = \begin{cases} \pm \left[\frac{|0\rangle + |1\rangle}{\sqrt{2}} \right] \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right] \rightarrow \pm \frac{1}{2} \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \end{pmatrix} & \text{if } f(0) = f(1) \\ \pm \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right] \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right] \rightarrow \pm \frac{1}{2} \begin{pmatrix} 1 \\ -1 \\ -1 \\ 1 \end{pmatrix} & \text{if } f(0) \neq f(1). \end{cases}$$

The transfer matrix of the third stage of the quantum circuit in Figure 3.2 is

$$G_3 = H \otimes 1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix}.$$

If $f(0) = f(1)$ then

$$|\xi_3\rangle = \pm \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix} \frac{1}{2} \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \end{pmatrix} = \pm \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \\ 0 \\ 0 \end{pmatrix} = \pm |0\rangle \frac{|0\rangle - |1\rangle}{\sqrt{2}}.$$

If $f(0) \neq f(1)$ then

$$|\xi_3\rangle = \pm \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix} \frac{1}{2} \begin{pmatrix} 1 \\ -1 \\ -1 \\ 1 \end{pmatrix} = \pm \frac{1}{\sqrt{2}} \begin{pmatrix} 0 \\ 0 \\ 1 \\ -1 \end{pmatrix} = \pm |1\rangle \frac{|0\rangle - |1\rangle}{\sqrt{2}}.$$

Therefore, if the first output qubit of the circuit is $|0\rangle$, then we can rule that $f(0) = f(1)$ because $|0\rangle \oplus |0\rangle = |0\rangle$ and $|1\rangle \oplus |1\rangle = |0\rangle$ and when the first output qubit of the circuit is $|1\rangle$, then we can rule that $f(0) \neq f(1)$ because $|0\rangle \oplus |1\rangle = |1\rangle$ and $|1\rangle \oplus |0\rangle = |1\rangle$.

We observe that

$$f(0) \oplus f(1) = \begin{cases} 0 & \text{if } f(0) = f(1) \\ 1 & \text{if } f(0) \neq f(1). \end{cases}$$

Finally, we rewrite $|\omega_3\rangle$

$$|\omega_3\rangle = \pm |f(0) \oplus f(1)\rangle \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right].$$

Therefore, we are able to establish $f(0) \oplus f(1)$ after performing a single computation of the function $f(x)$, just by measuring the first output qubit of the circuit.

Chapter 4

Quantum Algorithms

The purpose of this chapter is to help us understand Shor's factoring algorithm and order finding.

4.1 Classes of Quantum Algorithms

In a recent paper [Sho03], Peter Shor classifies the quantum algorithms proved to exhibit a significant speed-up over their standard counterparts into three broad categories:

1. Algorithms that find the periodicity of a function using Fourier Transforms, Simon's algorithm [127], Shor's algorithms for factoring and for computing discrete logarithms [123], and Hallgren's algorithm to solve Bell's Equation are all members of this class.
2. Search algorithms which can perform an exhaustive search of N items in \sqrt{N} . An example of this class are Grover's algorithms [67, 68, 69].
3. Algorithms for simulating quantum systems, as suggested by Feynman.

The focus of this paper is the first class of algorithms. Peter Shor speculates [Sho03] that quantum algorithms may offer a substantial speedup over classical algorithms, but may be very limited. This means we have to concentrate on problems not in the classical computational class \mathcal{P} – see Section 3.2.

One of these algorithms is the one for integer factorization. The security of cryptographic protocols, in general, is based on the fact that large integers are difficult to factor using classic computers. In 1994, Peter Shor found a factorization quantum algorithm operating in polynomial time [Sho94]. This algorithm performs the factorization by determining

the period of a function. By using quantum parallelism, the algorithm produces a superposition of all values of the function, in one iteration. Then, the algorithm computes the Quantum Fourier Transform (QFT) of the function. By using QFT, the algorithm organizes the amplitudes into multiples of the fundamental frequency, the reciprocal of the period.

The Figure 4.1 below summarizes the stages of the algorithm: integer factorization reduces to order finding, which requires phase (period) estimation, which requires Quantum Fourier Transform.

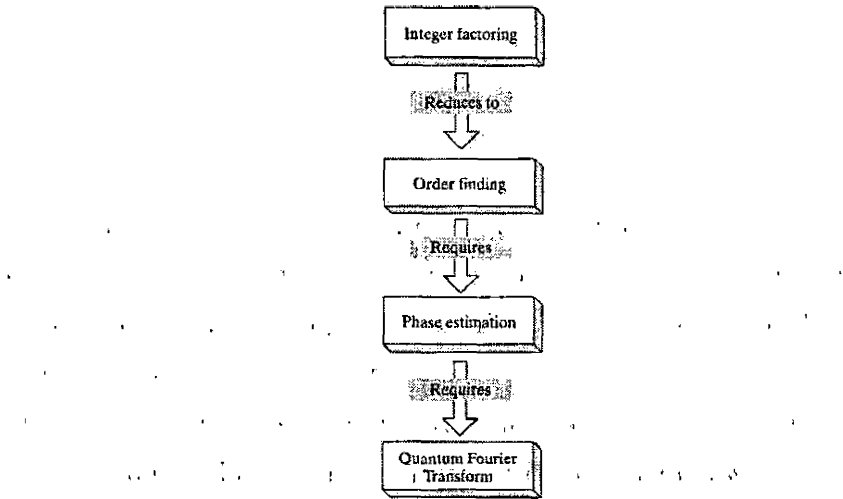


Figure 4.1: The relationship between integer factorization, order finding, phase estimation, and Quantum Fourier Transform.

4.2 Shor's Factoring Algorithm and Order Finding

We will construct an algorithm that allows us to determine the prime factors of P . This algorithm will be based on determining the order of integers $x \bmod P$ for $x < P$.

But for our factorization algorithm we cannot simply try all $x < P$. That is why we will determine the x integers using period estimation, better known as phase estimation. Phase estimation is an algorithm that allows us to estimate the eigenvalue associated with an eigenvector of a unitary operator. The phase estimation algorithm uses the Quantum Fourier Transform.

Definition 4.1. To factor an integer P means to write $N = l \times q_1 \times q_2 \times \dots \times q_n$, with l, q_1, \dots, q_n prime numbers.

Definition 4.2. The integer l is said to be a proper factor of P if three conditions are satisfied:

1. another integer q exists such that $P = lq$,
2. $l \neq 1$, and
3. $l \neq P$.

For example, the proper factors of 14 are $l = 2$ and $q = 7$, while the proper factors of 39 are $l = 3$ and $q = 13$. To factor small integers is trivial. The problem becomes increasingly difficult when the integers to be factored increase in size. Take for example factoring the integer 2841877. By trial and error we could find that $l = 19$ is a proper factor, thus $2841877 = 19 \times 149573$ but then we have to factor the integer 149573.

Definition 4.3. The order of an integer x modulo P is the smallest positive integer k , such that $x^k = 1 \pmod{P}$ with two additional conditions:

1. $x^1 \neq 1 \pmod{P}$, and
2. $x^{k-1} + x^{k-2} + \dots + x^2 + x + 1 \neq 1 \pmod{P}$

We can represent graphically the computation for the order k modulo P of an integer q as a cycle of length $k - 1$, as shown in Figure 4.2.

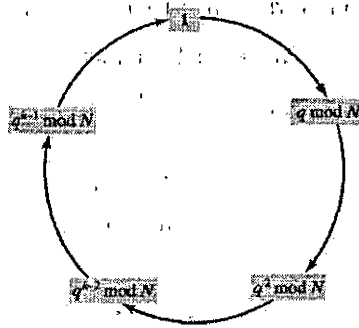


Figure 4.2: The cycle for computing the order k for an integer $x \pmod{P}$.

Several examples of finding the order of an integer modulo P by direct search are given below. First, we compute the order of 11 modulo 21 (i.e., we compute the smallest integer k such that $11^k = 1 \pmod{21}$). We start with $k = 2$ and continue with $k = 3, 4, 5, 6$.

$$k = 2 : 11^2 = 121 = 16 \bmod 21 = 2^4 \bmod 21$$

$$k = 3 : 11^3 = 176 = 8 \bmod 21 = 2^3 \bmod 21$$

$$k = 4 : 11^4 = 14641 = 4 \bmod 21 = 2^2 \bmod 21$$

$$k = 5 : 11^5 = 161051 = 2 \bmod 21 = 2^1 \bmod 21$$

$$k = 6 : 11^6 = 1771561 = 1 \bmod 21 = 2^0 \bmod 21$$

As you can see in Figure 4.3, the length of the cycle is $k - 1 = 5$.

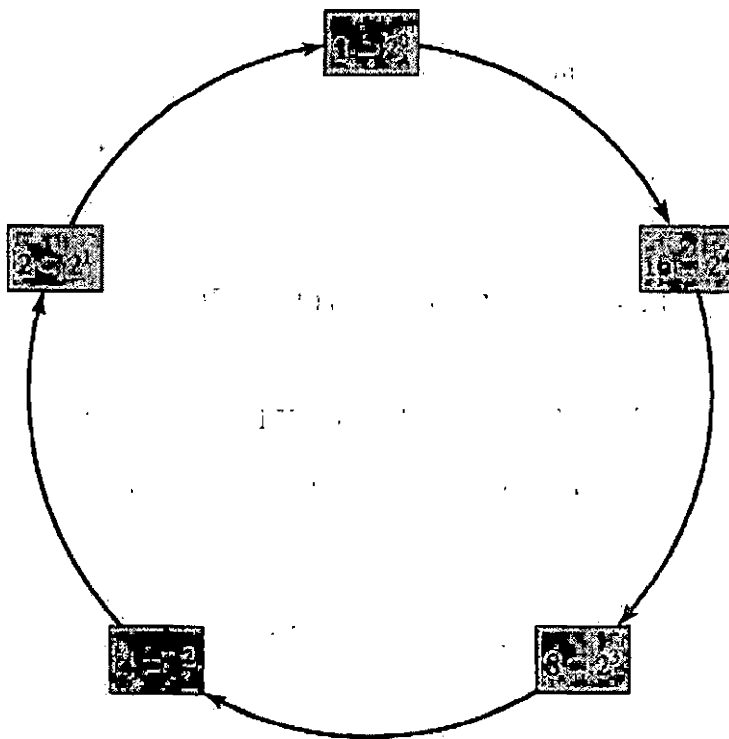


Figure 4.3: The multiplicative cycle for 11 mod 21.

Another example for the order of 5 modulo 21 :

$$5^2 = 25 = 4 \bmod 21$$

$$5^3 = 5^{2+1} = 5 \cdot 5^2 = 5 \cdot 4 \bmod 21 = 20 \bmod 21$$

$$5^4 = 5^2 \cdot 5^2 = (4 \bmod 21) \cdot (4 \bmod 21) = 16 \bmod 21$$

$$5^5 = 5^{4+1} = 5 \cdot 16 \bmod 21 = 17 \bmod 21$$

$$5^6 = 5^{4+2+1} = 5 \cdot (16 \bmod 21) \cdot (4 \bmod 21) = 1 \bmod 21$$

So the length of the cycle is also $k - 1 = 5$.

You can see that we only need several powers of x to compute x^k for any value of k . Indeed, we can express any integer k as

$$k = k_{m-1}2^{m-1} + k_{m-2}2^{m-2} + \dots + k_i2^i + k_12^1 + k_0$$

with $k_i = 0, 1$, $0 \leq i \leq m-1$. Then,

$$x^k = x^{k_{m-1}2^{m-1}} \times x^{k_{m-2}2^{m-2}} \times \dots \times x^{k_22^2} \times x^{k_12^1} \times x^{k_0}.$$

To compute x^k we need at most $m-1$ exponentiations

$$x^{2^1}, x^{2^2}, x^{2^3}, \dots, x^{2^{m-1}}$$

For example, when we wish to compute 17^{29} we can write:

$$29 = 16 + 8 + 4 + 1 = 2^4 + 2^3 + 2^2 + 2^0.$$

Thus,

$$17^{29} = 17^{16} \times 17^8 \times 17^4 \times 17^1.$$

The pseudo code to carry out this computation is

power := 1

for i=0 to m-1

 if $(x^{k_i} = 1)$ then

 power := power $\times x^{2^i} \bmod P$

endif

endfor

The condition $x^k = 1 \pmod{P}$ implies that $x^k - 1$ is divisible by P . Equivalently, this means that

$$\exists m \in \mathbb{Z} \mid x^k - 1 = (x - 1)(x^{k-1} + x^{k-2} + \dots + x^2 + x + 1) = mP.$$

This equation shows that either $(x - 1)$ or $(x^{k-1} + x^{k-2} + \dots + x^2 + x + 1)$ share a common factor with P . Assume that $x - 1$ is such a factor, or shares a common factor with P .

We show that $\gcd(P, x - 1) \neq 1$ and $\gcd(P, x - 1) \neq P$. In other words the $\gcd(P, x - 1)$ is a proper factor of P . Let us consider these two cases:

1. if $x - 1 < P$, then the common factor of $x - 1$ and P cannot be P .
2. if $x - 1 \geq P$, then $\gcd(P, x - 1)$ could be P if $x - 1$ is a multiple of P . But $x - 1$ is not a multiple of P due to the condition $x^1 \neq 1 \pmod{P}$ that can also be written as $(x - 1) \neq 0 \pmod{P}$.

Thus $l = \gcd(P, x - 1)$ is a proper factor of P and it is calculated with the Euclidean Algorithm.

Therefore finding the factors of P becomes a quest for order finding. To clarify this idea, let us give several examples.

First, we consider the easier case when P is neither even, nor the power of a prime. In this case we are looking for an integer x with the property that $x^2 - 1$ is a multiple of P , but neither $x - 1$ nor $x + 1$ are multiples of P (i.e., $x \not\equiv 1 \pmod{P}$ and $x \not\equiv -1 \pmod{P}$).

For example, when $P = 21$ a possible choice for x is $x = 8$. Indeed, $x^2 - 1 = 64 - 1 = 63 = 3 \times 21$. As expected, $x - 1 = 8 - 1 = 7$ is a proper factor of $P = 21$. When $q = 5$, we find again that $k = 6$. Now $x = 5^6 \pmod{21} = 20$, $x - 1 = 19$ and $x + 1 = 21$. We see that 21 is a factor, but not a proper factor of 21.

The pseudocode for finding a proper factor l for an integer P , based upon the order finding algorithm is:

Step 1. If P is even then return $l = 2$.

Step 2. If P is a power k of a prime integer l then return l .

Step 3. Randomly pick q , $1 < q < P - 1$.

If $l = \gcd(q, P) > 1$ then return l . Else go to Step 4.

Step 4. Determine the order k of $q \pmod{P}$.

If k is not even then go to Step 3.

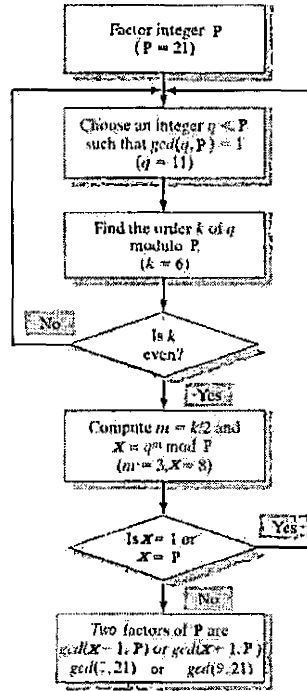


Figure 4.4: Flowchart of the factorization algorithm for $P = 21$ and $q = 11$.

Step 5. Let $k = 2m$ and determine x , the m -th power of $q \bmod N$ with $1 < x < P$.

If $1 < l = \gcd(x - 1, P) < P$ then return l .

If $1 < l = \gcd(x + 1, P) < P$ then return l .

Else (if we fail to find a proper factor of q) go to Step 3.

It is easy to see that when $k = 2m$ the condition $q^k = 1 \bmod P$ implies that the factors of P are either $\gcd(x - 1, P)$ or $\gcd(x + 1, P)$ with $x = q^m \bmod P$. Indeed,

$q^{2m} = 1 \bmod P \implies q^{2m} - 1 = (q^m - 1)(q^m + 1)$ must be divisible by P . The flowchart of the algorithm is shown in Figure ???. In our example the order of $11 \bmod 21$ is $k = 6$. But $6 = 2 \times 3$. Thus, $m = 3$ and $x = q^m \bmod P = 11^3 \bmod 21 = 8$. Then, $x - 1 = 8 - 1 = 7$ and $l = \gcd(x - 1, P) = \gcd(7, 21) = 7 < 21$ is a factor of $P = 21$. As we pointed out previously, the key to the efficiency of the algorithm is the choice of q .

4.3 Quantum Fourier Transform

Let

$$|v\rangle = \sum_{j=0}^{N-1} v_j |j\rangle$$

and

$$|w\rangle = \sum_{k=0}^{N-1} w_k |k\rangle$$

two states of a quantum system, where $|j\rangle$ and $|k\rangle$ are part of an orthonormal basis.

Then the Quantum Fourier Transform (QFT) takes the state $|v\rangle$ to the state $|w\rangle$:

$$|v\rangle \rightarrow |w\rangle$$

in such a way that the amplitudes w_k are the Discrete Fourier Transforms (DFT) of the amplitudes v_j :

$$w_k = DFT(v_j) = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} v_j e^{i2\pi jk/N}$$

When $N = 2^n$, we consider the binary representation of integers j and k

$$j = j_0 2^{n-1} + j_1 2^{n-2} + \dots + j_{n-2} 2 + j_{n-1} 2^0$$

$$k = k_0 2^{n-1} + k_1 2^{n-2} + \dots + k_{n-2} 2 + k_{n-1} 2^0$$

We use this reversed notation because in Section 4.5 the QFT circuit performs a bit reversal.

Thus, we obtain another expression for the QFT:

$$|j_0 j_1 \dots j_{n-1}\rangle \rightarrow \frac{1}{2^{\frac{n}{2}}} \sum_{k_0=(0,1)} \sum_{k_1=(0,1)} \dots \sum_{k_{n-1}=(0,1)} e^{i2\pi j \sum_{m=0}^{n-1} k_m 2^{-m}} |k_0 k_1 \dots k_{n-1}\rangle,$$

$$|j_0 j_1 \dots j_{n-1}\rangle \rightarrow \frac{1}{2^{\frac{n}{2}}} \sum_{k_0=(0,1)} \sum_{k_1=(0,1)} \dots \sum_{k_{n-1}=(0,1)} \otimes_{m=0}^{n-1} e^{i2\pi j k_m 2^{-m}} |k_m\rangle,$$

$$|j_0 j_1 \dots j_{n-1}\rangle \rightarrow \frac{1}{2^{\frac{n}{2}}} \otimes_{m=0}^{n-1} \left\{ \sum_{k_m=(0,1)} e^{i2\pi j k_m 2^{-m}} |k_m\rangle \right\}$$

where k_m can only be a 0 or a 1.

Because $e^{i2\pi j k_m} = 1$ when $k_m = 0$, then

$$|j_0 j_1 \dots j_{n-1}\rangle \rightarrow \frac{1}{2^{\frac{n}{2}}} \otimes_{m=0}^{n-1} \{ |0\rangle + e^{i2\pi j 2^{-m}} |1\rangle \}.$$

But

$$|0\rangle + e^{i2\pi j2^{-m}}|1\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ e^{i2\pi j2^{-m}} \end{pmatrix} = \begin{pmatrix} 1 \\ e^{i2\pi j2^{-m}} \end{pmatrix}$$

Therefore $|j\rangle$ has been converted to:

$$|j\rangle \rightarrow \frac{1}{2^{\frac{n}{2}}} \begin{pmatrix} 1 \\ e^{i2\pi j2^0} \end{pmatrix} \otimes \begin{pmatrix} 1 \\ e^{i2\pi j2^{-1}} \end{pmatrix} \otimes \begin{pmatrix} 1 \\ e^{i2\pi j2^{-2}} \end{pmatrix} \cdots \begin{pmatrix} 1 \\ e^{i2\pi j2^{-(n-1)}} \end{pmatrix}$$

Let us see an example for $n = 3$:

For $j = 0$, the basis vector $|000\rangle$ is transformed as follows:

$$|000\rangle \rightarrow \frac{1}{2^{\frac{3}{2}}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{1}{(\sqrt{2})^3} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

or

$$|000\rangle \rightarrow \frac{1}{2^{\frac{3}{2}}} [|000\rangle + |001\rangle + |010\rangle + |011\rangle + |100\rangle + |101\rangle + |110\rangle + |111\rangle]$$

For $j = 1$, the basis vector $|011\rangle$ is transformed as:

$$\begin{aligned} |011\rangle &\rightarrow \frac{1}{2^{\frac{3}{2}}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ e^{i2\pi(\frac{1}{2})} \end{pmatrix} \otimes \begin{pmatrix} 1 \\ e^{i2\pi(\frac{1}{4})} \end{pmatrix} = \\ &= \frac{1}{2^{\frac{3}{2}}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ -1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ i \end{pmatrix} \end{aligned}$$

or

$$|011\rangle \rightarrow \frac{1}{(\sqrt{2})^3} \begin{pmatrix} 1 \\ i \\ -1 \\ -i \\ 1 \\ i \\ -1 \\ -i \end{pmatrix}$$

Thus,

$$|011\rangle \rightarrow \frac{1}{(\sqrt{2})^3} [|000\rangle - |010\rangle + |100\rangle - |110\rangle + i(|001\rangle - |011\rangle + |101\rangle - |111\rangle)]$$

The transformations for the other basis vectors can be computed using the same procedure.

4.4 Tensor Product Factorization

We can compute QFT either using direct matrix multiplication or tensor product factorization.

Using the direct matrix multiplication we need a quantum system in an initial state $|V\rangle$, a quantum device with the transfer matrix G , and the final state of the system after the transformation $|W\rangle$. Both $|V\rangle$, and $|W\rangle$ will be expressed using two column vectors with 2^n elements, $|V\rangle, |W\rangle \in \mathcal{H}_{2^n}$, and G is a $2^n \times 2^n$ unitary matrix.

Therefore

$$|W\rangle = G |V\rangle$$

and performing this calculation by direct matrix multiplication we must use $\mathcal{O}((2^n)^2)$ operations.

The algorithm proposed by Cooley and Tukey [CT65] in 1965 diminishes the amount of operations from $2N^2$ to $2N \log_2 N$ and is based on tensor product factorization. The algorithm, called the Fast Fourier Transform (FFT), separates recursively the transformation for an integer $N = 2^n$ into two transformations of length $N/2$ as follows:

$$\sum_{j=0}^{N-1} a_j e^{\frac{-i2\pi jk}{N}} = \sum_{j=0}^{\frac{N}{2}-1} a_{2j} e^{\frac{-i2\pi(2j)k}{N}} + \sum_{j=0}^{\frac{N}{2}-1} a_{2j+1} e^{\frac{-i2\pi(2j+1)k}{N}} =$$

$$\sum_{j=0, j \text{ even}}^{\frac{N}{2}-1} a_j^{\text{even}} e^{\frac{-i2\pi 2jk}{N/2}} + e^{\frac{-i2\pi k}{N}} \sum_{j=0, j \text{ odd}}^{\frac{N}{2}-1} a_j^{\text{odd}} e^{\frac{-i2\pi jk}{N/2}}$$

We will use this algorithm to compute the Quantum Fourier Transform.

Consider a quantum system in state $|V\rangle$. We apply to this system a transformation with the transfer matrix G and the resulting state of the system will be $|W\rangle$, with $|V\rangle, |W\rangle \in \mathcal{H}_{2^n}$.

We can express both the space \mathcal{H}_{2^n} , as well as the transformation matrix, as tensor products:

$$\mathcal{H}_{2^n} = \mathcal{H}_2^{(1)} \otimes \mathcal{H}_2^{(2)} \dots \otimes \mathcal{H}_2^{(j)} \dots \otimes \mathcal{H}_2^{(n)}$$

$$G = G^{(1)} \otimes G^{(2)} \dots \otimes G^{(j)} \dots \otimes G^{(n)},$$

with $1 \leq j \leq n$.

Let us now examine the j -th component of $W = (W_1, W_2, \dots, W_j, \dots, W_{2^n})$ with the index j expressed in binary as $j = (j_1, j_2, \dots, j_n)$

$$W_j = \sum_{k_1, k_2, \dots, k_n} [G_{j_1 k_1}^{(1)} G_{j_2 k_2}^{(2)} \dots G_{j_n k_n}^{(n)}] V_{k_1 k_2 \dots k_n}.$$

Each application of $G^{(j)}$ requires a fixed number of operations and to compute each W_j we need $\mathcal{O}(n)$ operations because in the previous expression we have a product of n such matrices. There are 2^n components W_j thus, the total number of operations is $\mathcal{O}(2^n \cdot n)$.

We conclude that the tensor product factorization reduces the number of operations and leads to an exponential speed up compared with the direct matrix multiplication which requires $\mathcal{O}((2^n)^2)$ operations.

4.5 A Circuit for Quantum Fourier Transform

In this section we will obtain the QFT in a new format and derive the quantum circuit able to carry out the transformation based upon this new expression. We have defined in Section 4.3 that the Quantum Fourier Transform of $|j\rangle = |j_0 j_1 \dots j_{n-1}\rangle$ is

$$|j_0 j_1 \dots j_{n-1}\rangle \mapsto \frac{1}{2^{\frac{n}{2}}} \bigotimes_{m=0}^{n-1} \left\{ |0\rangle + e^{i2\pi j 2^{-m}} |1\rangle \right\}$$

We will use regular vector multiplication instead of tensor products:

$$\bigotimes_{m=0}^{n-1} \left\{ |0\rangle + e^{i2\pi j 2^{-m}} |1\rangle \right\} = \prod_{m=0}^{n-1} \left(|0\rangle + e^{i2\pi j 2^{-m}} |1\rangle \right)$$

Thus,

$$|j_0 j_1 \dots j_{n-1}\rangle \mapsto \frac{1}{2^{\frac{n}{2}}} \left(|0\rangle + e^{i2\pi(j/2^0)} |1\rangle \right) \left(|0\rangle + e^{i2\pi(j/2^1)} |1\rangle \right) \dots \left(|0\rangle + e^{i2\pi(j/2^{n-1})} |1\rangle \right)$$

Recall that we used the reversed notation for $j = j_0 2^{n-1} + j_1 2^{n-2} + \dots + j_{n-2} 2 + j_{n-1} 2^0$ and if we use the following notation [NC00]

$$0.j_m j_{m+1} \dots j_{n-1} = j_m 2^{-1} + j_{m+1} 2^{-2} + \dots + j_{n-1} 2^{-(n-m)},$$

the transformation becomes

$$|j_0 j_1 \dots j_{n-1}\rangle \mapsto \frac{1}{2^{\frac{n}{2}}} \left(|0\rangle + e^{i2\pi 0.j_{n-1}} |1\rangle \right) \left(|0\rangle + e^{i2\pi 0.j_{n-2} j_{n-1}} |1\rangle \right) \dots \left(|0\rangle + e^{i2\pi 0.j_0 j_1 \dots j_{n-1}} |1\rangle \right)$$

Figure 4.5 shows a circuit for the Quantum Fourier Transform based upon this new expression for the transformation. Each basis vector is first transformed into a superposition state by a Hadamard gate and then it goes through one or more R_k gates (see Chapter 2). An R_k gate transforms a qubit by multiplying its projection on $|1\rangle$ by $e^{i2\pi/2^k}$

$$R_k = \begin{pmatrix} 1 & 0 \\ 0 & e^{i2\pi/2^k} \end{pmatrix}$$

Under this Fourier transformation the bits of the transformed state are in reverse order.

So if

$$a = a_0 2^{n-1} + a_1 2^{n-2} + \dots + a_{n-3} 2^2 + a_{n-2} 2 + a_{n-1}$$

is presented as input to the transformation then the result will be

$$\bar{a} = a_{n-1} 2^{n-1} + a_{n-2} 2^{n-2} + a_{n-3} 2^{n-3} + \dots + a_2 2^2 + a_1 2 + a_0.$$

Let us now follow the state transformations produced by the circuit in Figure 4.5. We apply the Hadamard gate to the first qubit and obtain:

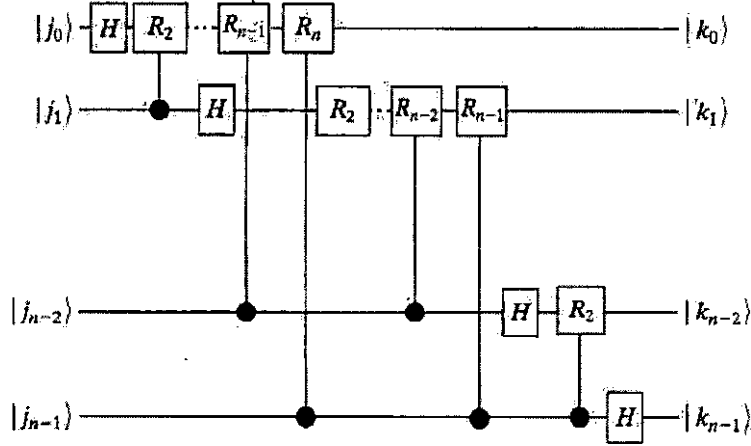


Figure 4.5: A circuit for Quantum Fourier Transform.

$$|j_0 j_1 j_2 \cdots j_{n-1}\rangle \mapsto \frac{1}{2^{\frac{1}{2}}}(|0\rangle + e^{i2\pi 0 \cdot j_0} |1\rangle) |j_1 j_2 \cdots j_{n-1}\rangle$$

with

$$e^{i2\pi 0 \cdot j_0} = e^{i2\pi j_0/2} = \begin{cases} +1 & \text{if } j_0 = 0 \\ -1 & \text{if } j_0 = 1. \end{cases}$$

Each *controlled* – R gate applied to the first qubit adds an extra bit to the phase of the projection on $|1\rangle$. As a result, we observe the successive changes of state

$$\begin{aligned} & \frac{1}{2^{\frac{1}{2}}}(|0\rangle + e^{i2\pi 0 \cdot j_0} |1\rangle) |j_1 j_2 \cdots j_{n-1}\rangle \mapsto \text{first } R_k \text{ gate} \mapsto \\ & \frac{1}{2^{\frac{1}{2}}}(|0\rangle + e^{i2\pi 0 \cdot j_0 j_1} |1\rangle) |j_1 j_2 \cdots j_{n-1}\rangle \mapsto \text{second } R_k \text{ gate} \mapsto \\ & \frac{1}{2^{\frac{1}{2}}}(|0\rangle + e^{i2\pi 0 \cdot j_0 j_1 j_2} |1\rangle) |j_1 j_2 \cdots j_{n-1}\rangle \mapsto \text{third } R_k \text{ gate} \mapsto \\ & \cdots \frac{1}{2^{\frac{1}{2}}}(|0\rangle + e^{i2\pi 0 \cdot j_0 j_1 j_2 \cdots j_{n-1}} |1\rangle) |j_1 j_2 \cdots j_{n-1}\rangle \end{aligned}$$

after the $n - 1$ -th R_k gate.

The transformations of the second qubit are

$$\frac{1}{2^{2(\frac{1}{2})}}(|0\rangle + e^{i2\pi 0 \cdot j_0} |1\rangle)(|0\rangle + e^{i2\pi 0 \cdot j_1} |1\rangle) |j_2 \cdots j_{n-1}\rangle \mapsto \text{first } R_k \text{ gate} \mapsto$$

$$\frac{1}{2^{2(\frac{1}{2})}}(|0\rangle + e^{i2\pi 0.j_0}|1\rangle)(|0\rangle + e^{i2\pi 0.j_1j_2}|1\rangle)|j_2 \dots j_{n-1}\rangle \mapsto \text{second } R_k \text{ gate} \mapsto$$

$$\dots \frac{1}{2^{2(\frac{1}{2})}}(|0\rangle + e^{i2\pi 0.j_0}|1\rangle)(|0\rangle + e^{i2\pi 0.j_1j_2 \dots j_{n-1}}|1\rangle)|j_2 \dots j_{n-1}\rangle.$$

As we can see from Figure 4.5, the total number of gates required by a QFT with $N = 2^n$ is

$$\text{Total Number of Gates}_{(QFT \text{ with } N=2^n)} = \frac{n(n+1)}{2}$$

If $N \neq 2^n$ then the QFT gives only approximate results and therefore we will round up the the next power of 2.

4.6 Simon's Algorithm for Phase Estimation

Definition 4.4. *Phase estimation is the process in which one determines the period of a periodic function.*

To solve problems such as integer factorization we need to perform repeatedly the phase estimation procedure.

In other words given a periodic function f that maps binary n -tuples to binary m -tuples, we wish to determine the period, p , of the function:

$$f(a) = f(b) \iff a \equiv b \pmod{p}$$

Classic algorithms calculate the period of a function in exponential time. This is because classic algorithms cannot compute p until it finds two inputs a and b such that $f(a) = f(b)$. In 1994, Dan Simon created a quantum algorithm for phase estimation that requires quadratic execution time.

Simon formulates the phase estimation problem as follows [Sim97]:

We are given a function $f : \{0, 1\}^n \mapsto \{0, 1\}^m$ with the prospect that there exists a $p \in \{0, 1\}^n$, $p \neq 0^n$ such that $\forall a \neq b \ f(a) = f(b) \iff b = a \oplus p$, find p .

In other words, given a function $f(a)$, we wish to establish whether $f(a)$ is a periodic function and, if so, to determine its period p .

The algorithm uses two n qubits registers A and A' , with $|A| = |A'| = 2^{n-1}$ and $A' = \{a \oplus p | a \in A\}$.

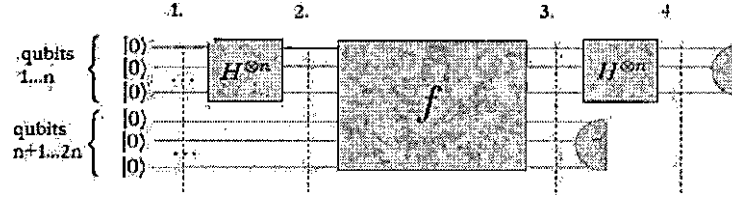


Figure 4.6: A quantum circuit for phase estimation

You can follow Figure 4.6

Step 1: We start by initializing both registers to state $|0\rangle$.

Step 2: Take the first register of n qubits in state $|0\rangle$ and apply the Hadamard transform to it obtaining the superposition:

$$2^{-\frac{n}{2}} \sum_{a \in \{0,1\}^n} |a\rangle |0 \dots 0\rangle$$

Step 3: Calculate $f(a)$ and store the result in the second register obtaining:

$$\begin{aligned} 2^{-\frac{n}{2}} \sum_a |a\rangle |f(a)\rangle &= \\ &= 2^{-\frac{n}{2}} \sum_{a \in A} \frac{1}{\sqrt{2}} (|a\rangle + |a \oplus p\rangle) |f(a)\rangle \end{aligned}$$

Measure $2^{-\frac{n}{2}} \sum_{a \in A} \frac{1}{\sqrt{2}} (|a\rangle + |a \oplus p\rangle) |f(a)\rangle$ for random $a \in A$

Step 4: Apply the Hadamard transform on the output of Step 3 to produce

$$\begin{aligned} 2^{-\frac{n+1}{2}} \sum_{b \in \{0,1\}^n} \left((-1)^{a \cdot b} + (-1)^{(a \oplus p) \cdot b} \right) |b\rangle &= \\ = 2^{-\frac{n+1}{2}} \sum_{b \in \{0,1\}^n} (-1)^{a \cdot b} (1 + (-1)^{p \cdot b}) |b\rangle &= \\ = 2^{-\frac{n+1}{2}} \sum_{b: b \cdot p = 0} (-1)^{a \cdot b} |b\rangle \end{aligned}$$

We now will start measuring randomly b_1 such that $b_1 \cdot p = 0$ and then repeat measuring randomly b_2 such that $b_2 \cdot p = 0$, b_3 such that $b_3 \cdot p = 0$. We know that p is uniquely determined when we have $n - 1$ linearly independent equations.

Repeating this process $\mathcal{O}(n)$ times will give us $n - 1$ linearly independent y 's.

Chapter 5

Conclusion

The most used cryptosystem in the world is the RSA. The RSA system was invented by Rivest, Shamir, and Adleman in 1978 [RSA78] and it is based on the difficulty of factoring large numbers. In 1994, Peter Shor found a factorization quantum algorithm operating in polynomial time [Sho94]. This algorithm performs the factorization by determining the period of a function. By using quantum parallelism, the algorithm produces a superposition of all values of the function, in one iteration. Then, the algorithm computes the Quantum Fourier Transform (QFT) of the function. By using QFT, the algorithm organizes the amplitudes into multiples of the fundamental frequency, the reciprocal of the period. Then, by using the results of Deutsch's problem, one can discriminate between the various results.

Here is an example for factoring $N=15$ using Shor's algorithm.

(1) Chose number of qubits so $2^n \geq N$. In our case, $n = 4$, $2^4 > N$

We choose an a such as $\gcd(a, N) = 1$. For example, we pick $a = 13$.

(2) Initialize two quantum registers of $n = 4$ qubits to state $|0000\rangle$

(3) Put the first register through a Hadamard gate obtaining superposition of states:

$$|0000\rangle \rightarrow \frac{1}{\sqrt{16}} (|0\rangle + |1\rangle) (|0\rangle + |1\rangle) (|0\rangle + |1\rangle) (|0\rangle + |1\rangle) = \frac{1}{\sqrt{2^4}} \sum_{k=0}^{15} |k\rangle$$

The state of the two registers after this step is:

$$|\varphi_1\rangle = \frac{1}{\sqrt{2^4}} \sum_{k=0}^{15} |k\rangle |0\rangle$$

(4) Compute the function $f(k) = 13^k \bmod 15$ on the second register:

$$|\varphi_2\rangle = \frac{1}{\sqrt{2^4}} \sum_{k=0}^{15} |k\rangle |f(k)\rangle$$

Let us calculate $f(k) = 13^k \bmod 15$.

$$k = 1, f(1) = 13^1 \bmod 15$$

$$k = 2, f(2) = 13^2 \bmod 15 = (15 \cdot 11 + 4) \bmod 15 = 4$$

$$k = 3, f(3) = 13^3 \bmod 15 = (15 \cdot 11 + 4) \cdot 13 \bmod 15 = 7$$

$$k = 4, f(4) = 13^4 \bmod 15 = (7 \cdot 13) \bmod 15 = 1$$

Therefore the period is $r = 4$.

And the state is

$$\begin{aligned} |\varphi_2\rangle &= \frac{1}{\sqrt{2^4}} \sum_{k=0}^{15} |k\rangle |f(k)\rangle = \\ &= \frac{1}{\sqrt{2^4}} (|0\rangle|1\rangle + |1\rangle|13\rangle + |2\rangle|4\rangle + |3\rangle|7\rangle + \\ &\quad + |4\rangle|1\rangle + |5\rangle|13\rangle + |6\rangle|4\rangle + |7\rangle|7\rangle + \\ &\quad + |8\rangle|1\rangle + |9\rangle|13\rangle + |10\rangle|4\rangle + |11\rangle|7\rangle + \\ &\quad + |12\rangle|1\rangle + |13\rangle|13\rangle + |14\rangle|4\rangle + |15\rangle|7\rangle) \end{aligned}$$

obtained in one operation due to quantum parallelism.

As you can see, we obtain equal probabilities for the possible results: $|1\rangle$, $|13\rangle$, $|4\rangle$, and $|7\rangle$.

(5) Operate on first four qubits by the QFT F

$$|k\rangle \longrightarrow \frac{1}{\sqrt{2^4}} \sum_{u=0}^{15} e^{\frac{2\pi i u k}{16}} |u\rangle$$

, measure the state of the first register and assume that the second register is measured, as well. Suppose we measure $|4\rangle$ in the second register. Therefore the superposition will collapse and the state will be:

$$|\varphi_3\rangle = \frac{1}{2} (|2\rangle + |6\rangle + |10\rangle + |14\rangle)$$

We now apply quantum Fourier transform (QFT)

$$|k\rangle \longrightarrow \frac{1}{\sqrt{2^4}} \sum_{u=0}^{15} e^{\frac{2\pi i u k}{16}} |u\rangle$$

and obtain:

$$|2\rangle \longrightarrow \frac{1}{\sqrt{2^4}} \sum_{u=0}^{15} e^{\frac{2\pi i u 2}{16}} |u\rangle$$

$$|6\rangle \longrightarrow \frac{1}{\sqrt{2^4}} \sum_{u=0}^{15} e^{\frac{2\pi i u 6}{16}} |u\rangle$$

$$|10\rangle \longrightarrow \frac{1}{\sqrt{2^4}} \sum_{u=0}^{15} e^{\frac{2\pi i u 10}{16}} |u\rangle$$

$$|14\rangle \longrightarrow \frac{1}{\sqrt{2^4}} \sum_{u=0}^{15} e^{\frac{2\pi i u 14}{16}} |u\rangle$$

Putting these four terms together, we get

$$\begin{aligned} |\varphi_3\rangle &= \sqrt{\frac{4}{16}} \frac{1}{\sqrt{16}} \sum_{u=0}^{15} |u\rangle \left(e^{\frac{2\pi i u 2}{16}} + e^{\frac{2\pi i u 6}{16}} + e^{\frac{2\pi i u 10}{16}} + e^{\frac{2\pi i u 14}{16}} \right) = \\ &= \frac{1}{8} \sum_{u=0}^{15} |u\rangle A_u \end{aligned}$$

The probability of getting result $|u\rangle$ after first register is measured is

$$P_u = \left| \frac{1}{8} A_u \right|^2$$

When we calculate P_u for all 16 cases we get $P_0 = P_4 = P_8 = P_{12} = \frac{1}{4}$ and all other probabilities being zero. Therefore, we can get only states $|0\rangle$, $|4\rangle$, $|8\rangle$, and $|12\rangle$ with equal probabilities.

We can show the probabilities are non zero only if $ur = 16k$.

So the probabilities to get the correct period from the first run are:

$|u\rangle = |0\rangle$ - does not give any information - rerun algorithm;

$|u\rangle = |4\rangle$ - gives $4r = 16k$, lowest $k = 1$: Period is $r = 4$;

$|u\rangle = |8\rangle$ - gives $8r = 16k$, $r = 2$ is easy to check is not correct and we rerun the algorithm

$|u\rangle = |12\rangle$ - gives $12r = 16k$, $k = 3$: Period is $r = 4$.

Therefore the algorithm has a probability of success of $\frac{1}{2}$.

Now that we have the period, the factors of N can be found using a classical computer by calculating $\gcd(a^{\frac{r}{2}} + 1, N)$ and $\gcd(a^{\frac{r}{2}} - 1, N)$:

$$\gcd(13^{\frac{4}{2}} + 1, 15) = 5$$

$$\gcd(13^{\frac{4}{2}} - 1, 15) = 3$$

Bibliography

- [BFFP10] F. Benatti, M. Fannes, R. Floreanini, and D. Petritis, editors. *Quantum information, computation and cryptography*, volume 808 of *Lecture Notes in Physics*. Springer-Verlag, Berlin, 2010. An introductory survey of theory, technology and experiments.
- [CT65] James W. Cooley and John W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Math. Comp.*, 19:297–301, 1965.
- [DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Trans. Information Theory*, IT-22(6):644–654, 1976.
- [Joz98] Richard Jozsa. Entanglement and quantum computation. In *The geometric universe (Oxford, 1996)*, pages 369–379. Oxford Univ. Press, Oxford, 1998.
- [MM05] D. C. Marinescu and G. M. Marinescu. *Approaching Quantum Computing*. Pearson Prentice Hall, Upper Saddle River, New Jersey, 2005.
- [NC00] Michael A. Nielsen and Isaac L. Chuang. *Quantum computation and quantum information*. Cambridge University Press, Cambridge, 2000.
- [Pap94] Christos H. Papadimitriou. *Computational complexity*. Addison-Wesley Publishing Company, Reading, MA, 1994.
- [RSA78] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Comm. ACM*, 21(2):120–126, 1978.
- [Sho94] Peter W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *35th Annual Symposium on Foundations of Computer Science (Santa Fe, NM, 1994)*, pages 124–134. IEEE Comput. Soc. Press, Los Alamitos, CA, 1994.

- [Sho97] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, 1997.
- [Sho02] Peter W. Shor. Introduction to quantum algorithms. In *Quantum computation: a grand mathematical challenge for the twenty-first century and the millennium (Washington, DC, 2000)*, volume 58 of *Proc. Sympos. Appl. Math.*, pages 143–159. Amer. Math. Soc., Providence, RI, 2002.
- [Sho03] Peter W. Shor. Why haven't more quantum algorithms been found? *J. ACM*, 50(1):87–90 (electronic), 2003.
- [Sim97] Daniel R. Simon. On the power of quantum computation. *SIAM J. Comput.*, 26(5):1474–1483, 1997.
- [Tur] A. M. Turing. On Computable Numbers, with an Application to the Entscheidungsproblem. *Proc. London Math. Soc.*, S2-42(1):230.