Imperial College London

Imperial College London

Department of Physics

Introduction to Shor's quantum factoring algorithm and review of a proof-of-principle implementation using photonic qubits

Author:
Gordon J. Köhn

Supervisor:
Prof. Florian Mintert

A report submitted for the degree of

MSc Physics

January 20, 2021

Abstract

This report provides an introduction to the possible threat to classical cryptography due to quantum attacks by Shor's algorithm. Shor's quantum factoring algorithm allows to factorize an integer efficiently due to quantum effects. This theoretically allows to break most widely used classical asymmetric cryptography schemes.
First, Shor's algorithm will be elucidated in theory and its implementation in quantum circuits. The necessary prerequisites in the domain of quantum information will be introduced on the basis of graduate level knowledge of quantum mechanics.
Second, a recent experimental realization of a proof-of-principle instance of Shor's algorithm using phonic qubits will be explained with focus on the compiling of the algorithm. A brief introduction of optical quantum computing will be provided for context.
Understanding the quantum threat to asymmetric cryptography is of enormous relevance as asymmetric cryptography provides for the security and integrity to all electronic communication in our information age.

Keywords: Quantum Computing, Asymmetric Cryptography, Shor's Algorithm, Factorisation, Optical Quantum Computing

Acknowledgements

The author of this paper would like to thank the following people for their contribution to this project:

Prof. Florian Mintert - for his caring supervision of the entire project.

Prof. Myungshik Kim - for his teaching in Quantum Information.

Contents

Chapter 1

Introduction

This report is aimed at master level physics students with graduate level knowledge in quantum mechanics. A limited introduction to quantum information and optical quantum computing will be provided with the focus to understand Shor's Algorithm and its implementation.

In the 1970s and 1980s a shift in perspective happened in the leading group of pioneers of quantum mechanics. Some pioneers turned their perspective from trying to understand quantum mechanical systems found in nature, to looking at them as systems that could be designed. The new discipline of *Quantum Information* and *Quantum Computation* was founded, as the study of the informational processing of tasks that can be accomplished using quantum mechanical systems. The mysterious nature of quantum mechanics leads to an entirely different way in which algorithms have to be constructed. For instance, where a classical computer is capable to copy or negate an arbitrary piece of binary information, a quantum computer is by the laws of quantum mechanics not able to do either, see *No-Cloning Theorem* and *No Universal Not-Gate*. Therefore, quantum algorithms have to be constructed in an inherently non intuitive way which posed and still poses a big challenge to develop quantum algorithms. Despite these challenges, algorithms have been developed of which some provide efficient solutions to problems to which no efficient classical algorithms are known. One of these quantum algorithms is *Shor's Algorithm*, which allows to solve the problem of finding the prime factors of an integer efficiently. 'Efficiently' is meant here in an algorithmic sense, where it describes an acceptable amount of physical and time resources needed to solve the problem. [1, xvii-7]
It will become clear in Section 1.1 how Shor's algorithm poses a threat to asymmetric cryptography. For this reason the current way nearly all private communication in the internet today is transmitted is at a threat. This is one of the reasons that sparked interest in this class of algorithms. [2]
Since a handful of these quantum algorithms, which are superior to their classical counterparts, have been discovered, the efforts to a build quantum computer of sufficient size to outperform a classical one had a strong motivation. Yet, the current state of a quantum computers does not provide the platform to perform any quantum algorithms to a useful extent, i.e. outperforming a classical computer. Even though no such platform exists yet, current quantum computers have the capability to implement proof-of-principle instances of Shor's Algorithm.
There are various modes to realize a quantum computer physically. One attractive physical system for representing a quantum bit is the optical photon. In Chapter 3) of this report the implementation using such a system will be presented with focus on the algorithmic implementation of Shor's algorithm. [3]
The implementation presented is:

Demonstration of a Compiled Version of Shor's Quantum Factoring Algorithm Using Photonic Qubits - PRL 99, 250504 (*2007*)

Chapter 1) and 2) of this report are intended to provide the foundation to understand this implementation of Shor's algorithm and its relevance.

1.1 Asymmetric Cryptography

The need to secure data in transit or storage by third adversaries has been there since the beginning of man and led to the development of cryptography, the study how to satisfy that need. Modern day electronic communication could only unfold its true potential because of secure cryptography systems. There are two kinds of cryptography systems: *symmetric* and *asymmetric.*

For the purpose of this report the focus here is on asymmetric cryptography, yet it is best introduced and motivated by elucidating symmetric cryptography.

In *symmetric cryptography* the sender and receiver use the same private key, a piece of private information, to encrypt and decrypt a message that is supposed to be kept private. This private key has to be kept private as otherwise the secrecy of the message is lost. This key defines all operations that are needed to transform the message to encrypted form and the encrypted form to the message. This leads the need of exchanging secret keys over public channels. This is where asymmetric cryptography provides the means to solve the problem of *key distribution.* [2]

In *asymmetric cryptography* (also *public key cryptography*) there exist a pair of keys. One is the public key that allows the sending party to encrypt a message and the other is the private key that allows the receiving party, *party A*, to decrypt the message. The decisive difference is that only the private key held by *party A*, has to be kept secret and the public key is be made freely available allowing many parties to send messages securely to *party A*.

Even though in principle, messages may be send via asymmetric cryptography it is computationally expensive and thus mostly used for the key distribution for symmetric cryptography. In addition, asymmetric cryptography is also used for verifying the integrity of documents with digital signatures. [2] Both applications make asymmetric cryptography absolutely essential to modern day electronic communication.

All asymmetric cryptography algorithms have a mathematical problem at their core, that is, heuristically put, *'easy'* to compute in one direction but *'hard'* in the other. These mathematical problems or functions are called *one-way functions.* [4]
The existence of one-way functions is an open mathematical conjecture, yet after extensive research no efficient inverting algorithms have been found for the following examples: [5]

- *Factorization Problem - RSA Cryptosystem:* One of the oldest asymmetric cryptography systems, named after the initials of its inventors, exploits the difficulty of factorizing bi-prime numbers, i.e a large number $N = pq$ into two prime numbers p and q. [6]
- *Discrete Logarithm Problem (DLP):* The difficulty for DLP arises from determining an integer r such that $g^r = x \mod p$. This is then called the discrete logarithm problem of x to the base g as $r = \log_g x \mod p$.

Both of these one-way function are 'hard' to solve in one direction yet 'easy' in the other, for certain sets of parameters. [2] There are other examples of one-way functions used for asymmetric cryptography, yet this report will specifically look at how Shor's algorithm threatens the difficulty of *Factorization* and only comment on how the *Discrete Logarithm Problem* is affected as well.

1.2 Complexity of Factorization - classically

To appreciate the security that asymmetric cryptography provides and the threat Shor's algorithm poses, a more quantitative notion of the prior description of the difficulty than *'easy'* and *'hard'* has to be given. It has been known since before Euclid's time that every integer N can be uniquely decomposed into a product of primes. [7] As an expression this translates as:

$$N = pq \tag{1.1}$$

where p and q are primes of n decimal digits and N has d decimal digits.

Multiplying p and q to get N is the 'easy' way of the problem. More formally, there exist efficient algorithms to accomplish this, like the *long-multiplication algorithm.* The number of operations that have to be performed to multiply two n digit numbers computationally are about n^2. [8]

In computer science the time and resources required of an algorithm are often compared using the *Big-O* notation. It allows to compare the efficiency of algorithms independent of the specific platform and implementation with the same input size. All multiplication algorithms known have, written in *Big-O*, a resource requirement of $O(n^2)$, which means n^2 bit operations are required in the worst case. Thus multiplication scales polynomially in difficulty with the number of digits n of its factors. [5, p.26]

Factorizing the number N on the other hand is 'hard'. The brute force algorithm iterates through all primes p up to $\sqrt{N}$ and checks whether p is a divisor of N. In the worst case, this thus takes $\sqrt{N}$ bit operations, which is exponential in digits d, i.e. $O(e^d)$. There exist more efficient algorithms, like the *quadratic sieve* or the most efficient algorithm called the *general number sieve*, which achieves an asymptotic runtime of $O(exp(d^{1/3}))$. No more efficient classical algorithm is known, thus the difficulty of factorization scales exponentially in the classical world as of our current knowledge. [5, p.26]

This exponential to polynomial difference in difficulty between factorization and multiplication respectively, defines the nature of this one-way function.
Even though in the classical world no more efficient algorithm is known for factorization, Shor's quantum algorithm has been shown to provide a solution in only polynomial time on a quantum computer, destroying the one-way function.

1.3 Introduction to Quantum Information & Computation

In this section a brief introduction to quantum information and computation is given, assuming a graduate level knowledge of quantum mechanics, emphasizing the properties used later in this report and not aiming for completeness.

1.3.1 The Qubit

Where in classical information theory the *bit* stands, is the *qubit* in quantum information theory. Classical bits have a well-defined state: they are either 0 or 1 at every point during the computation. This restriction is lifted from their quantum counterpart, as qubits may exist in an arbitrary superposition of states of the form

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle = \begin{bmatrix}\alpha\\ \beta\end{bmatrix} \tag{1.2}$$

where $|\alpha|^2$ and $|\beta|^2$ satisfy $|\alpha|^2 + |\beta|^2 = 1$, so that they are the respective probabilities of finding $|0\rangle$ and $|1\rangle$ after measurement in the $\{|0\rangle, |1\rangle\}$ basis. The outcome of a measurement can then be associated with the classical bits. [1, p.17]

1.3.2 Multiple Qubits & Entanglement

Consider a system of n qubits. Such a system has the computational basis state $|x_1x_2...x_n\rangle$ and is defined by 2^n complex amplitudes. A state may also be expressed as a complex vector in Hilbert space of 2^n dimensions.
If the qubits of a system are not correlated, the total system can be written as a tensor product, indicated by the symbol $\otimes$, of the individual states of the n qubits. For instance, the two states $|\psi_1\rangle$ and $|\psi_2\rangle$ have the joint state $|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle$.
Not all possible states can be written as such a collection of independent qubits. When the state can only be written as a whole it is said to be *entangled*, between the qubits for which the states cannot be separated. In other words, the states cannot be described independently. *Entanglement* is a probably the most counterintuitive and important resource in quantum information.
Where in the classical world two bits can take the states 00, 01, 10, 11, any two qubits may be expressed as

$$|\psi\rangle = a_0|00\rangle + a_1|01\rangle + a_2|10\rangle + a_3|11\rangle = \begin{bmatrix}a_0\\ a_1\\ a_2\\ a_3\end{bmatrix} \tag{1.3}$$

and a general n qubit state as:

$$\psi = \sum_{i=0}^{2^2-1} a_i |n\rangle \tag{1.4}$$

where again the amplitudes a_i are complex numbers such that $\sum_{i=0}^{2^2-1} |a_i|^2 = 1$ and also each $|n\rangle$ is a basis vector of the Hilbert space.
Sometimes it is useful to denote larger than two qubit states as logical states indicated by a tilde. Later two following two states will be used:

$$|\tilde{0}\rangle = |0\rangle \otimes ... \otimes |0\rangle \tag{1.5}$$

$$|\tilde{1}\rangle = |0\rangle \otimes ... \otimes |0\rangle \otimes |1\rangle \tag{1.6}$$

1.3.3 Quantum Gates & Circuits

In order to use a quantum mechanical system for computation, one must be able to change the state of the system. The operations that transform states are called *gates*. The evolution of states is only permitted by unitary transformations by the laws of quantum mechanics and is formally facilitated by a unitary operator U acting on a state. In the context of quantum information a system that provides a unitary operator U is called a *gate*.
An algorithm composed out of multiple gates is best visualized analog to classical *logical circuits* in a *quantum circuit*. Most conventions from classical logic circuits have been adapted, see Figure 1.1.

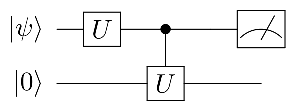


Figure 1.1: Example of an arbitrary quantum circuit for two qubits. The first qubit is initialized in $|\psi\rangle$ and the second in $|0\rangle$. Then a single gate operation, denoted by U, is performed on the first qubit followed by a controlled two qubit operation on the second, controlled by the first qubit, see Section 1.3.2. Finally, a measurement is performed on the first qubit [1, Chap. 1]

Gates can be categorized into *single*, *multiple* and *classically controlled gates.*

Single Qubit Gates

In contrast to classical computation where the only single bit operation is the NOT operation the ability to form superpositions allows for more versatile operations on a single qubit alone. The most commonly used single qubit gates are: *NOT*, *Hadamard*, *Z* and *Phase Shift* gates.

- *Not-Gate:* (also *X-Gate*) is described by the Pauli-X matrix, σ_x. It is the quantum generalization of the classical NOT gate as it flips the logical state it is acting on. A state $|0\rangle$ becomes $|1\rangle$ and vice versa.

- *Hadamard Gate:* brings the state $|0\rangle$ and $|1\rangle$ to respectively $|+\rangle$ and $|-\rangle$ states. The corresponding unitary matrix is:
$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{1.7}$$

- *Z-Gate:* performs a sign shift if the qubit is $|1\rangle$ and nothing otherwise. The transformation is described by the Pauli-Z matrix, σ_z. The Z-Gate belongs to a more general group of gates, the *phase shift gate*, and corresponds to a phase shift gate of phase shift $\Phi = \pi$.

- *Phase Shift Gate, R_Φ:* performs a rotation of a state around the Z-axis by a phase Φ. The state $|0\rangle$ is unchanged, but the state $|1\rangle$ gains a phase $e^{i\Phi}$, i.e. in matrix operator is: [1, p.19] [9]
$$R_\Phi = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\Phi} \end{bmatrix} \tag{1.8}$$

Input	**Gate**	**Output**	**Circuit Representation**
$\alpha\|0\rangle + \beta\|1\rangle$	$\xrightarrow{\text{H}}$	$\alpha\|+\rangle + \beta\|-\rangle$	$-\boxed{H}-$
$\alpha\|0\rangle + \beta\|1\rangle$	$\xrightarrow{\text{X}}$	$\alpha\|1\rangle + \beta\|0\rangle$	$-\boxed{X}-$
$\alpha\|0\rangle + \beta\|1\rangle$	$\xrightarrow{\text{Z}}$	$\alpha\|0\rangle - \beta\|1\rangle$	$-\boxed{Z}-$
$\alpha\|0\rangle + \beta\|1\rangle$	$\xrightarrow{R_\Phi}$	$\alpha\|0\rangle + e^{i\Phi}\beta\|1\rangle$	$-\boxed{R_\Phi}-$

Table 1.1: Operation of selected single-qubit gates and their circuit representation

Multiple Qubit & Controlled Gates

Multiple qubit gates perform operations on more than one qubit. While more than two qubit operations exist, it has been shown that n-qubit operations can always be constructed out of two qubit operations. [10] Multiple qubit gates entangle formerly independent qubits. The most intuitive and most used type of multiple qubit gates are *controlled gates.* This type of gate has two (or more) input qubits, known as the *control* and the *target* qubit, respectively. A controlled gate applies a unitary operation U on the target qubit if the control qubit is in $|1\rangle$, otherwise no action is done. The control qubit is never altered. [1, p.20] [9]
Any single qubit unitary can be used to build a controlled gate, yet for the purpose of this paper only two following examples are relevant:

- *Controlled-Not, CNOT:* (also CX) is the most universally used gate. It is the controlled version of the Not/X gate and performs a selected negation of the target qubit.
- *Controlled Phase Shift, CR_Φ* is the controlled version of the phase shift gate.

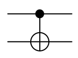
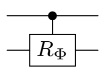
Gate Operation	**Circuit Representation**
$\mathrm{a}_1\|00\rangle + a_2\|01\rangle + a_3\|10\rangle + \|11\rangle \xrightarrow{\text{CNOT}} a_1\|00\rangle + a_2\|01\rangle + a_3\|11\rangle + a_4\|10\rangle$	
$\mathrm{a}_1\|00\rangle + a_2\|01\rangle + a_3\|10\rangle + \|11\rangle \xrightarrow{CR_\Phi} a_1\|00\rangle + a_2\|01\rangle + a_3e^{i\Phi}\|10\rangle + a_4e^{i\Phi}\|11\rangle$	

Table 1.2: Operation of selected controlled gates and their circuit representations

Furthermore, in some applications it may be desirable to combine a classical control signal with the application of a single qubit quantum gate. Such gates are called *classically controlled gates*, yet are non relevant for the further discussion here.

1.4 The Case for Quantum

The introduced toolkit allows to design novel algorithms, yet only some of which hold an inherent advantage over classical algorithms. The advantage of quantum computation is not at all obvious, thus the general idea behind fast quantum algorithms is introduced here heuristically.

One promising way to find solutions to classically intractable problems is to use quantum algorithms to find a *global property,* such as the period or the extremum of some function $f(x)$. Where a classical computer has to compute $f(x)$ for various different inputs x to obtain enough information about the *global property,* a quantum computer can compute many possible inputs simultaneously by taking a superposition of states as an input. The resulting parallel computation is called *Quantum Parallelism.* This does not mean that all the possible outcomes can be accessed by measurement, yet in special cases *quantum interference* effects can be induced, which will reveal the global property. [1][7] The function $f(x)$ is chosen such that the measurable global property contains the desired computational output.
This is the core principle by which fast quantum algorithms, like Shor's, gain an efficiency advantage.

Chapter 2

Shor's Algorithm

Shor's algorithm was invented in 1994 by the American mathematician Peter Shor. The algorithm allows to solve two classically intractable problems, namely the problem of 'prime factors of an integer' and the 'discrete logarithm problem' efficiently on a quantum computer. Here the earlier problem will be explained in detail. [1, p.6] [7]

2.1 Theory

The algorithm is composed of two parts. The first part translates the problem of factorization to a problem of finding the period of a function. This part of the algorithm may be implemented using quantum or classical computation. The second part however is inherently of quantum nature and is responsible for the quantum speedup as it exploits *Quantum Parallelism* to find the period of a function $f(x)$ using the *Quantum Fourier Transform.* [7]

2.1.1 From Factorization to Period Finding

Since the 1970s it has been known to mathematicians that factoring becomes easy if one solves another *'hard'* problem, namely finding the period of the *modular exponential function.*
This problem is to find the smallest positive integer r such that $a^r - 1$ is a multiple of N, where N and a are positive integers, such that $a < N$, which have no common factors. Written using modulo, we equivalently seek to find r, which is the smallest (non-zero) integer satisfying:

$$a^r \bmod N \equiv 1 \tag{2.1}$$

Connecting this to the problem of factorization, N remains the integer desired to be factorized and a is randomly chosen, such that $1 < a < N$. Let it be assumed that N has only two distinct prime factors greater than 2. [7] [11]
Equation 2.1 defines the period r of the function:

$$f(x) = a^x \bmod N \tag{2.2}$$

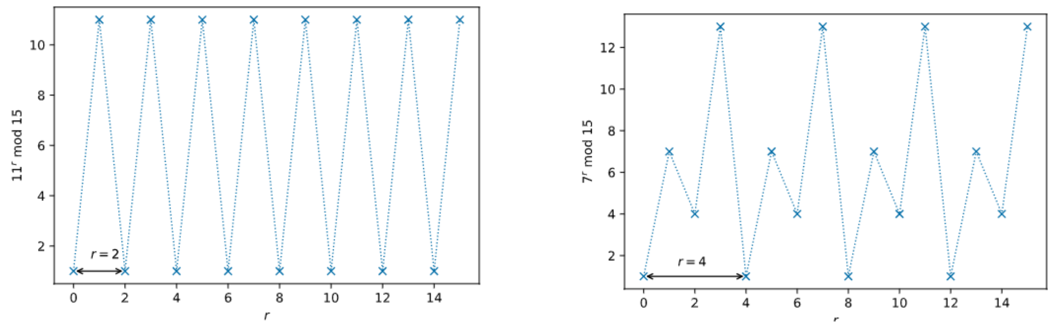
Figure 2.1 illustrates how the periodicity is guaranteed. Rewriting the earlier statement of Equation 2.1 using a mathematical identity, we obtain:

$$a^r - 1 = (a^{r/2} - 1)(a^{r/2} + 1) \tag{2.3}$$

This reveals that $(a^{r/2} - 1)$ is no multiple of N, because then the period would be $r/2$. In the case that $(a^{r/2} + 1)$ is not a multiple of N. Also neither of the integers $a^{r/2} \pm 1$ is a multiple of N, yet their product is, see Equation 2.3. This in turn is only possible if the co-primes p and q, defining $N = pq$, are prime factors of $(a^{r/2} - 1)$ and $(a^{r/2} + 1)$ themselves. Thereby, we may compute the primes p and q by the *greatest common denominator*, *gcd* to find the shared prime numbers as

$$p, q = \gcd(N, a^{r/2} \pm 1) \tag{2.4}$$

The greatest common denominator can be efficiently calculated using Euclid's Algorithm. [12] [13] [14]
In summary, if the period r of the function can be found and the number a had been chosen such



(a) for $a = 11$ (b) for $a = 7$

Figure 2.1: Examples of the periodic function in Shor's algorithm. The function $f(x)$, defined as in Equation 2.2, is plotted for the positive integers up to N clearly showing their periodicity of period r. Note the lines between points are just meant to help see the periodicity and do not represent intermediate points. Here it can be seen that the smallest integer satisfying Equation 2.1 naturally gives the period.

a	**Period r**	$\mathbf{gcd(15, a^{r/2} - 1)}$	d $\mathbf{gcd(15, a^{r/2} + 1)}$
2	4	3	5
4	2	3	5
7	4	3	5
8	4	3	5
11	2	5	3
13	4	3	5
14	2	1	15

Table 2.1: Example instance of period to prime factor conversion for $N = 15$. Note that $a = 14$ is an example of the case when $(a^{r/2} + 1)$ is a multiple of N.

that $(a^{r/2} + 1)$ is not a multiple of N then the prime factors of the original factorization problem can be recovered.
In the unfortunate case of $(a^{r/2}+1)$ being a multiple of N however, the primes cannot be obtained. For $N = 15$ an example of this step is shown in Table 2.1. [7] [11]

Procedure of the Classical Part

The above arguments allow to construct an entirely classical procedure for the reduction of the factoring problem to period-finding. This procedure is as follows: [7] [11]

1. Choose a random number a, such that $1 < a < N$.
2. Check if by chance the number a is already a prime factor of N, by $\gcd(a, N)$.
3. In the unlikely case that $\gcd(a, N) \neq 1$, then this number a is already a nontrivial factor of N and the second can easily be found by division. The problem is solved.
4. Otherwise, the period of the function $f(x)$, see Equation 2.2, is found using *quantum subroutine* for period finding, explained in the next Section 2.1.2.
5. If the period r is found to be odd or $(a^{r/2} + 1)$ is a multiple of N, return to step 1.
6. Otherwise, use Equation 2.4 to obtain the nontrivial factors of N.

2.1.2 Finding the Period

The quantum computational part of Shor's algorithm is best understood in three parts. First, the *quantum Fourier transform*, which is the core of Shor's algorithm, is explained. Second, the subroutine of *Phase Estimation* shows how hence a phase can be found and thirdly how this phase can be modulated to contain the period of the function $f(x)$ using *Modular Exponentiation.*

Quantum Fourier Transform

Mathematical symbols do NOT correspond to symbols used earlier to follow conventional notation.

The *quantum Fourier transform* (also *QFT*) is the quantum equivalent of the *discrete Fourier transform.* The discrete Fourier transform takes an input vector of complex numbers $x_0, ...x_{N-1}$ where N the length of the vector and outputs a vector of complex numbers $y_0, ...y_{N-1}$ defined by

$$y_k \equiv \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{2\pi ijk/N}. \tag{2.5}$$

The quantum Fourier transform performs the same transformation, yet on an orthonormal basis $|0\rangle, ... |N-1\rangle$ as a linear operator with the action on the basis states,

$$|j\rangle \longrightarrow \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi ijk/N} |k\rangle. \tag{2.6}$$

For an arbitrary state this action is then

$$\sum_{j=0}^{N-1} x_j |j\rangle \longrightarrow \sum_{k=0}^{N-1} y_k |k\rangle \tag{2.7}$$

where the y_k are the amplitudes defined by the discrete Fourier transform in Equation 2.5.
In order to construct a quantum circuit consider a n qubit quantum computer with the computational basis $|0\rangle, ... |2^n-1\rangle$, i.e $N = 2^n$ for some integer n. Furthermore, it will become apparent later that is is useful to express $|j\rangle$ in the binary representation,

$$|j\rangle = |j_1 j_2 ... j_n\rangle = \left|j_1 2^{n-1} + j_2 2^{n-2} + ... + j_n 2^0\right\rangle \tag{2.8}$$

and also to adopt the notation to represent the binary fraction $0.j_l j_{l+1}...j_m = j_l/2 + j_{l+4} + ... + j_m/2^{m-l+1}$.
Having introduced this notation it can be shown that Equation 2.6 can be written in *product representation*:

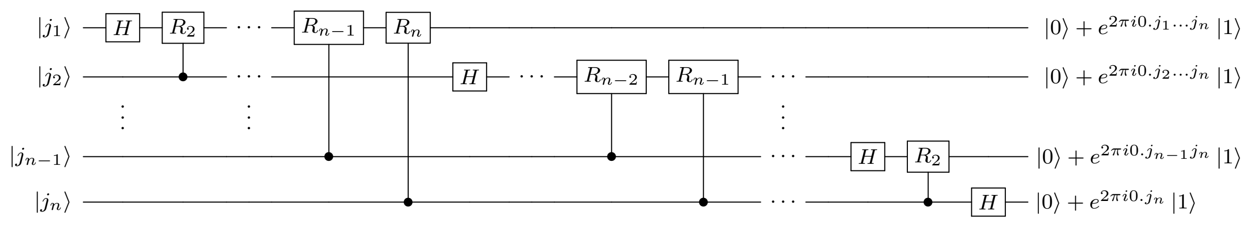
$$|j_1, ..., j_n\rangle \rightarrow \frac{\big(|0\rangle + e^{2\pi i 0.j_n}|1\rangle\big)\big(|0\rangle + e^{2\pi i 0.j_{n-1}j_n}|1\rangle\big)...\big(|0\rangle + e^{2\pi i 0.j_1 j_2 ... j_n}|1\rangle\big)}{2^{n/2}}, \tag{2.9}$$

which is a very useful form to derive the corresponding quantum circuit. The explicit algebra can be found at [1, p.218].
A quantum circuit performing this action can be derived using only two types of gates, namely the Hadamard and rotational gate introduced in Section 1.3.3. The efficient circuit is shown in Figure 2.2, where for convenience we denote the rotational gate as R_k with a phase $\phi = 2\pi/2^k$. To see how the depicted circuit indeed performs the action defined by Equation2.9, consider that action the the state $|j_1, ...j_n\rangle$ as the input. The first operator action is the Hadamard on the first qubit leaving the state in

$$\frac{1}{2^{1/2}}\big(|0\rangle + e^{2\pi i 0.j_1}|1\rangle\big)|j_2...j_n\rangle. \tag{2.10}$$

Next, the controlled rotational gate R_2 is applied to the first qubit. As $e^{2\pi i 0.j_1} = 1$ if j_1 and $+1$ otherwise, the state becomes

$$\frac{1}{2^{1/2}}\big(|0\rangle + e^{2\pi i 0.j_1 j_2}|1\rangle\big)|j_2...j_n\rangle. \tag{2.11}$$



$|j_1\rangle$
$|j_2\rangle$
$|j_{n-1}\rangle$
$|j_n\rangle$
H
R_2
R_{n-1}
R_n
R_{n-2}
$|0\rangle + e^{2\pi i 0.j_1 \cdots j_n}|1\rangle$
$|0\rangle + e^{2\pi i 0.j_2 \cdots j_n}|1\rangle$
$|0\rangle + e^{2\pi i 0.j_{n-1} j_n}|1\rangle$
$|0\rangle + e^{2\pi i 0.j_n}|1\rangle$

Figure 2.2: Quantum circuit for the quantum Fourier transform. For readability, SWAP-gates, which reverse the order of the qubits, before the output of the circuit and normalization factors of of the output labels of $\frac{1}{\sqrt{2}}$ have been omitted. Figure based on [1, fig. 5.1]

Following the controlled rotational gates R_3 to R_n are performed yielding the state

$$\frac{1}{2^{1/2}}\big(\,|0\rangle + e^{2\pi i 0.j_1 j_2 \ldots j_n}\,|1\rangle\,\big)\,|j_2 \ldots j_n\rangle\,, \tag{2.12}$$

which is notably the last product term in the desired action, see Equation 2.9. The same sequence of gates is performed on each of the other qubits. For the second qubit the Hadamard is followed by the gates R_2 to R_{n-1}. The final state obtained after this procedure on all of the qubits yields

$$\frac{1}{2^{n/2}}\big(\,|0\rangle + e^{2\pi i 0.j_1 j_2 \ldots j_n}\,|1\rangle\,\big)\big(\,|0\rangle + e^{2\pi i 0.j_2 \ldots j_n}\,|1\rangle\,\big)\ldots\big(\,|0\rangle + e^{2\pi i 0.j_n}\,|1\rangle\,\big)\,|j_2 \ldots j_n\rangle \tag{2.13}$$

To obtain the desired action of the quantum Fourier transform described in Equation 2.9, the order of the qubits had to be reversed. The order of the qubits can be reversed using so called SWAP-gates, which can be construed of CNOT-gates, yet are not explained here.

How many gates does the algorithm require? Following through the circuit the number of operations can counted for each qubit as follows:

1st*qubit*	H followed by $n-1$ controlled R gates	$= n$ gates
2nd*qubit*	H followed by $n-2$ controlled R gates	$= n-1$ gates
$\vdots$	$\vdots$	$\vdots$
nth*qubit*	H	$= 1$ gate

Summing all operations up yields the expression $n + (n-1) + \ldots + 1 = n(n+1)/2$. To complete the quantum Fourier transform algorithm the in the figure omitted SWAP operation have to be performed. A SWAP operation is always preformed on two qubits simultaneously, thus at most $n/2$ SWAP gates are needed. Each SWAP gates is constructed out of three controlled-NOT gates. The number of gates, hence operations needed to perform the quantum Fourier transform is defining the efficiency of the algorithm. Facilitating the Big-O notation introduced earlier, it can be seen that the number of operations scales with n^2, thus the algorithm is of the kind $O(n^2)$, where n is the number of qubits.
This is a magnificent result as the best classical algorithms, such as the *Fast Fourier Transform* on the same number of states/elements, 2^n, requires $O(n2^n)$ gates, i.e. scales exponentially. [1, chap.5] Despite this advantage this efficiency can only be exploited under special conditions because of multiple problems, as for instance that the amplitudes cannot be directly measured. Nevertheless, more subtle applications of the QFT as for instance in the *Quantum Phase Estimation* allow to exploit the efficiency of the algorithm. [1, chap.5]

Quantum Phase Estimation

The aim of *quantum phase estimation* is to estimate the phase ϕ of the eigenvalue of an unitary operator U satisfying the eigenequation

$$U\,|u\rangle = e^{2\pi i\phi}\,|u\rangle\,. \tag{2.14}$$

The quantum phase estimation is not a self contained algorithm, it is rather thought of as a 'subroutine' as it requires the state $|u\rangle$ to be prepared as well as the unspecified operator U. [1,

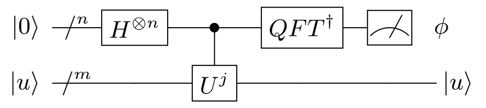


Figure 2.3: Quantum phase estimation schematic circuit. The top n qubits are denotes as a sets by '/' and form the first register. Similarly for second sets of qubits or second register $|u\rangle$. The gate $QFT^\dagger$ denotes an inverse QFT upon the first register. [1, fig. 5.3]

chap.5]

The subroutine requires two sets of qubits. Such sets are also called *registers.* The first register of n qubits is initially prepared in state $|0\rangle$. The number of qubits required, n, depends on the number of binary digits needed to encode the phase to be found ϕ. It also determines the probability of success of the routine, yet not explained here.
The second register is initially in state $|u\rangle$ and has as many qubits as are needed to encode the state, defining m. [1, chap.5]

Preparation of States - The subroutine starts with applying Hadamard gates on all qubits of the first register. This is followed by controlled-U gates on the second register with U raised to successive powers of 2.
This leaves the first register in the state:

$$\frac{1}{2^{n/2}}\big(|0\rangle + e^{2\pi i 2^{n-1}\phi}|1\rangle\big)\frac{1}{2^{n/2}}\big(|0\rangle + e^{2\pi i 2^{n-2}\phi}|1\rangle\big)...\frac{1}{2^{n/2}}\big(|0\rangle + e^{2\pi i 2^{0}\phi}|1\rangle\big), \tag{2.15}$$

omitting $|u\rangle$ as it remains unchanged throughout. Consider the simplest case for which the first register has exactly as many bits as are needed to encode ϕ, then writing $\phi = 0.\phi_1...\phi_n$, the state becomes [1, chap.5]

$$\frac{1}{2^{n/2}}\big(|0\rangle + e^{2\pi i 0.\phi_n}|1\rangle\big)\frac{1}{2^{n/2}}\big(|0\rangle + e^{2\pi i 0.\phi_{n-1}\phi_n}|1\rangle\big)...\frac{1}{2^{n/2}}\big(|0\rangle + e^{2\pi i 0.\phi_1\phi_2...\phi_n}|1\rangle\big). \tag{2.16}$$

Inverse Fourier Transform - Following, the reverse to the earlier introduced QFT is applied to the first register. The inverse circuit is simply the earlier introduced circuit in reverse order, see Figure 2.2. The action is of the $QFT^\dagger$ is

$$\frac{1}{2^{n/2}}\sum_{j=0}^{2^n-1} e^{2\pi i\phi j}|j\rangle|u\rangle \longrightarrow |\tilde{\phi}\rangle|u\rangle \tag{2.17}$$

where $\left|\tilde{\phi}\right\rangle$ denotes a state providing a good estimate of ϕ or exactly ϕ if the first register has the right number of qubits.
Measurement in computational basis is the last step of the subroutine. When a measurement is performed on the first register in the $|\phi_1...\phi_n\rangle$ basis the phase ϕ or a an estimate of it is obtained.

The number of operations required in total mainly depend on the quantum Fourier transform, if the implementation of the unspecified U gate is done efficiently, such that the difficulty scales as $O(n^2)$, as before.

In brief, quantum phase estimation allows to determine phase of the eigenvalue of an operator U given the eigenvector $|u\rangle$, at its heart using the inverse QFT. This operator U is the last part needed to complete Shor's algorithm and is provided by the *functional modular exponentiation.* [1, chap.5]

Modular Exponentiation

The *functional modular exponentiation* is the part of the algorithm, which implements the function $f(x) = a^x \bmod N$ and connects the classical part of the algorithm logically with the quantum part. Hence it will become apparent that in the following a and N are defined as in Section 2.1.1.
The modular exponentiation provides the unitary operator needed to perform the quantum phase estimation such that

$$U|y\rangle \equiv |ay \bmod N\rangle\,. \tag{2.18}$$

a = 11	a = 7
$U\|1\rangle = \|11\rangle$	$U\|1\rangle = \|7\rangle$
$U^2\|1\rangle$	$U^2\|1\rangle = \|4\rangle$
	$U^3\|1\rangle = \|13\rangle$
	$U^4\|1\rangle = \|1\rangle$

Table 2.2: Table of successive operations of operator U, see Equation 2.18, for random number $a = 11$ and $a = 7$. After r applications the state $|1\rangle$ is obtained again. Figure based on [11].

To realize how this is useful, consider the successive application of U on logical state $|1\rangle$, for a random number a and the number N that is to be factorized, see Table 2.2,
After r applications of U on the state the state $|1\rangle$ is found again, where r is the period as of $f(x)$. In this process, the outcome state cycles through different states for $a = 11$, $|u_0\rangle = \{|11\rangle, |1\rangle\}$ and $|u_0\rangle = \{|7\rangle, |4\rangle, |13\rangle, |1\rangle\}$ in the case of $a = 7$. Now consider a superposition the states these states of a such cycle,

$$|u_0\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} |a^k \bmod N\rangle \tag{2.19}$$

To see that such a superposition is an eigenstate of the operator U consider the $|u_0\rangle$ for the simplest case of $a = 11$,

$$|u_0\rangle = \frac{1}{\sqrt{2}}\big(|1\rangle + |11\rangle\big) \tag{2.20}$$

$$U|u_0\rangle = \frac{1}{\sqrt{2}}\big(U|1\rangle + U|11\rangle\big) = |u_0\rangle \tag{2.21}$$

Next assume that the superposition $|u_0\rangle$ for which the phase is different for each of the states. Let the phase of the k^{th} state be proportional to k and s an arbitrary integer,

$$|u_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-\frac{2\pi i s k}{r}} |a^k \bmod N\rangle \tag{2.22}$$

and the corresponding eigenequation

$$U|u_s\rangle = e^{\frac{2\pi i s}{r}} |u_s\rangle . \tag{2.23}$$

Such a superposition is very desirable as the eigenvalue does contain the period of r of $f(x)$ along with the random number s. Summing up all the states leads to canceling of states of different phases, such that all basis states except $|1\rangle$ vanish, so

$$\frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |u_s\rangle = |1\rangle . \tag{2.24}$$

This is exactly where the *Quantum Interference* effect happens which leads to the canceling, mentioned in the introduction.[11]
Having obtained a single state of an eigenvalue containing the period we shall recall the action of the quantum phase estimation subroutine, see Equation 2.14 and comparing with Equation 2.23. So then if the quantum phase estimation is applied on U using the state $|1\rangle$ the phase will be:

$$\phi = \frac{s}{r} \tag{2.25}$$

with s the random integer between 0 and $r - 1$. This is a magnificent result as having measured ϕ, the r can now be computed using the classical algorithm of *continued fractions.* [15] [11]

In the above section, it was shown how an operator U performing the modular exponentiation allows to prepare a state such that the quantum phase estimation algorithm can extract the period of the function $f(x)$. It is not explained here how such an operator U is implemented in quantum gates, as it not responsible for the efficiency advantage of Shor's algorithm.
Such implementations of modular exponentiation between two registers of the number of qubits

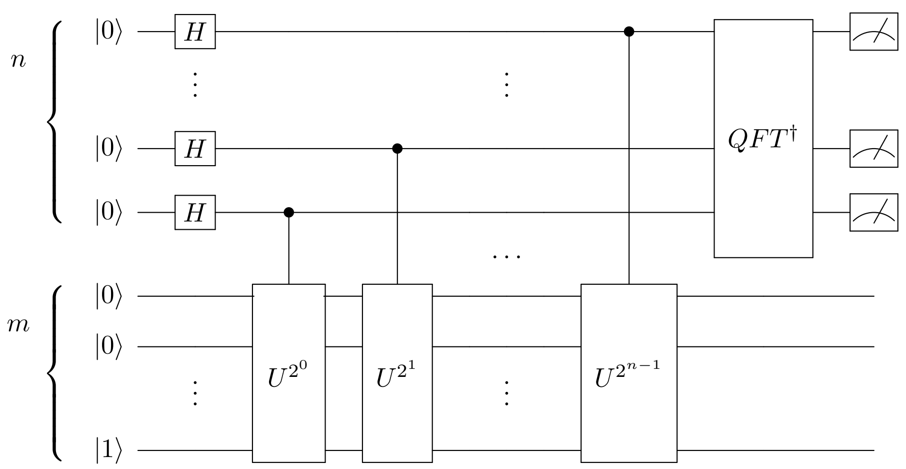


Figure 2.4: Quantum circuit of Shor's algorithm. The first n qubits make up the first register and are initialized in $|\tilde{0}\rangle$. The second register is formed by the m lower qubits initialized in $|\tilde{1}\rangle$ for the purpose of modular exponentiation. The SWAP gates following the QFT as part of the phase estimation subroutine on the first register are omitted here for readability. Figure based on [17]

n and m, as in the phase estimation, scale in difficulty in the general worst case as $O(m^2)$. [1, p.228] [16] There is active research how to make implement modular exponentiation more efficient, yet this peripheral to understanding the advantage of Shor's algorithm. [16]

Putting it all together Shor's algorithm uses the quantum phase estimation powered by the quantum Fourier transform to obtain the period of the modular exponentiation function. The full quantum circuit then looks like, see 2.4. First, a Hadamard gate is applied on all qubits in the first register. This is followed by the modular exponentiation, here represented by the successive powers of U between the two registers. Last, the inverse quantum Fourier transform is applied on the first register, upon which these states are measured in the ϕ_1, ϕ_n basis yielding the binary representation of ϕ, in the order of the basis from top to bottom.
The exact form of the circuit depends on the parameters a and N chosen. The number of qubits n of the first register depends on the number of binary digits needed to represent ϕ, as described in the phase estimation section. The number of qubits in the second register depends on the number of states needed to form a cycle forming a full modular exponentiation. It can be shown that these are respectively:

$$n = 2\lceil \log_2 N \rceil \quad (2.26) \qquad m = \lceil log_2 N \rceil \quad (2.27)$$

Furthermore, note that the exact form of the modular exponentiation, which is implemented in the circuit by the U gates, encodes a and N. The form of the circuit is unique for each instance of a and N.
The total number of gates of the full algorithm may be said to be $O(n^2)$ with the highest number of gates arising from the QFT. Optimised circuits of modular exponentiation are sufficiently efficient so that they do not contribute to the complexity.

This completes the second part of Shor's algorithm and proved that the period and hence the factors of the number N can be found in only a polynomial scaling number of operations depending on the size of N! This destroys the asymmetry of the factorization problem, allowing theoretically to break most used asymmetric cryptography schemes.

Shor's algorithm may similarly be applied to the discrete logarithm problem introduced earlier. In this case the operator U has to be defined differently, yet the structure and logic of the algorithm stays similar, allowing for the problem to be solved efficiently. [1, p.235]

For correctness, let it be stated that this section followed is based on the works [1, chap.5], [3] and [11]. Furthermore, please acknowledge that the this section does not aim for completeness and in particular the following aspects have been deliberately left out: circuit of modular exponentiation, accuracy of phase and success probability of algorithm.

Chapter 3

Review of a recent Proof-of-Principle Implementation

Based on the foundations formed in the last two chapters a closer look at a particular instance of Shor's algorithm that was recently implemented experimentally will be made here. In 2007, a group mainly based at the University of Science and Technology of China reported to have demonstrated a complied version of Shor's algorithm. Their experimental implementation successfully demonstrated the factorization of the number $N = 15$ into its factors. The quantum computing device they used was an *optical quantum computer.* [3]
This chapter will briefly introduce the idea of an optical quantum computer for context at the beginning. Following on this is a description of the *Hong-Ou-Mandel effect*, which is key to understanding the quantum nature of an optical quantum computer. After this experimental context is established, the particular instance of the algorithm will be explained and calculated though in detail, for understanding of the algorithm.

3.1 Introduction to Optical Quantum Computing

In order to realize the elementary unit of the quantum information theory one needs a *two-level system*, to represent a *quantum bit.* The infamous *Stern-Gerlach experiment* demonstrated how two-level systems can be realized in reality.[1, Chap.7] Assuming the states of the system are well defined further perquisites exist to build a quantum computer on physical system. These criteria require that the preparation of initial states and measurement of the individual qubits on output has the be possible. Furthermore, to make any calculations a universal set of operators must be possible to implement. These are a few of the criteria a physical system must pass in order to be a viable candidate for a quantum computing system. A complete and formal description of these criteria was made by Di-Vincenzo. [10], called *Di-Vincenzo criteria.*
A major experimental challenge arising from those requirements is to keep a quantum system isolated enough to prevent states from *decoherence.* Decoherence describes the loss of a defined phase relation between differnt states and leads to the loss of the information of the quanutm system.
No consensus has yet been established in the scientific community regarding what kind of physical system is most favorable under Di-Vincenzo criteria. A promising system is the *optical photon.* Photons are appealing because of their low susceptibility to noise and the presumed ease of manipulation with conventional optical components. [18]
Optical quantum computer represent qubits in one of two ways. A two level system is formed by either the location of a single photon between two modes, $|01\rangle$ or $|10\rangle$, or by realizing two optical modes by horizontal and vertical polarization, $|H\rangle$ and $|V\rangle$. The initial states are prepared by obtaining single photons by attenuating a laser light. The readout is done by detecting single photons by a photo-multiplier tube. Gates are constructed from optical elements such as phase sifters, beamsplitters and non-linear Kerr media.[1] Unfortunately, photons interact only very weakly with each other, making it hard to construct gates mediating interactions between two qubits. [1, p.296] [18] Nevertheless, small scale solutions to construct a universal set of gates have been found

[1] *Non-Linear Kerr* media provide cross phase modulation between two polarization modes of light and are hence important for the interaction of photons.

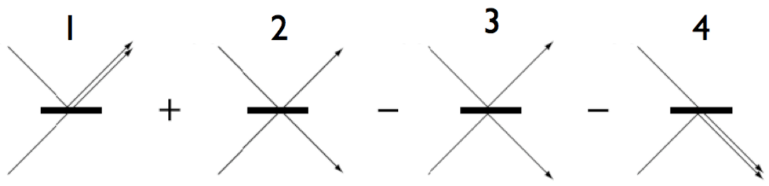


Figure 3.1: Schematic showing the four different ways for two photons to interact on a beam splitter. The signs correspond to the sign introduced by a phase shift in the associated term in the superposition of states. Figure based on [21]

and research is done about the scalability. [18] [19] The construction of those gates is part of the practical implementation and not discussed here, yet to see how in principle the interaction between states is established the Hong-Ou-Mandel effect shall be explained qualitatively.

3.1.1 Hong-Ou-Mandel Effect

It has been proven that a necessary condition for a quantum computational speedup over classical computation requires entanglement of states in the calculation. [20] The Hong-Ou-Mandel effect is the source of entanglement in an optical quantum computer.

It is a two-photon interference effect theorized and demonstrated in 1987 by its name givers. It occurs when two identical single photons enter a beam splitter from each side. When a photon enters a beam splitter it is either reflected or transmitted with a probability each depending on the type of beam splitter. Assume a 50/50 beam splitter for equal probabilities. When two indistinguishable photons enter from either side of a beam splitter simultaneously there are four possible behaviours to be expected:

1. photon 1 is reflected and photon 2 is transmitted
2. both photons are transmitted
3. both photons are reflected
4. photon 2 in transmitted and photon 1 is reflected

As no measurement is done in the process, the two photon system exist in a superposition of these four states after leaving the beam splitter. Further note that reflection from the lower side of the beam splitter adds a phase shift of $\pi/2$ to the amplitude of the associated term in the superposition, resulting in a minus sign. Thereby, as the two photons are indistinguishable, the states 2 and 3 cancel each other out. This leads to the stunning result that two photons always leave the beam splitter on the same side, see Figure 3.1. This is a destructive interference effect, which entangles the two photons.

More subtleties of the effect exist, yet for the purpose of demonstration how two photon state provide the resource of entanglement needed for quantum computing this description shall suffice.

The experimental implementation discussed here used polarized states of single photos as qubits. Some quantum gates involved beam splitter at which the described Hong-Ou-Mandel effect was observed for equally polarized photons, providing entanglement between states.

3.2 Shor's Algorithm applied – $N = 15$

The aim of the demonstration, reviewed here [3], was to proof that the first meaningful instance of Shor's algorithm, namely factoring $N = 15$, could be done using photons as qubits. Based on the leanings from Chapter 1 and 2 how would the algorithm be implemented?

3.2.1 Theory & Circuit

Referring to the steps of the classical part of the algorithm, see Section 2.1.1, suitable candidates for the random number a have to be identified. For $N = 15$, a may be 2, 4, 7, 8, 11, 13 or 14.

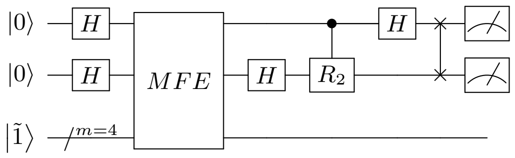


Figure 3.2: Shor's algorithm using the simplification to $n = 2$ qubits in the first register. MFE denotes the modular functional exponentiation. The operations following the MFE implement the inverse quantum Fourier transform, including the SWAP operation denoted by the connected crosses.

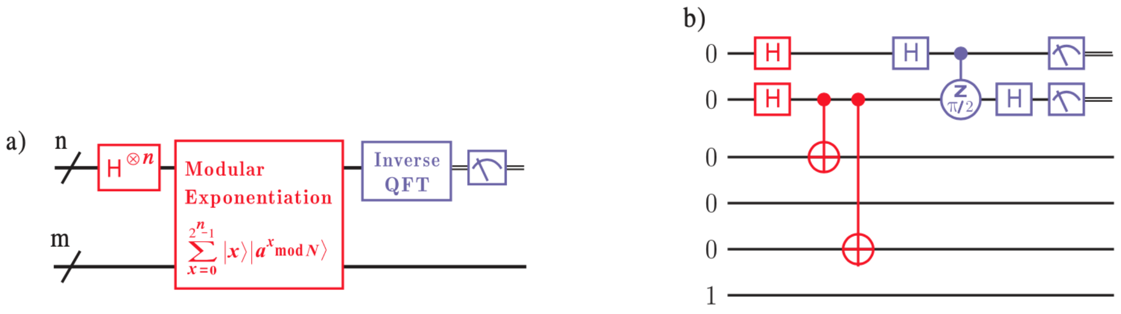


Figure 3.3: Quantum circuits of proof-of-principle implementation taken from paper [3]. a) Schematic of the quantum circuit highlighting the main part of the algorithm by colors. b) Quantum circuit of implementation. Note that the two CNOT gates implement the modular exponentiation, not explained. Further, note that the $Z_{\pi/2}$ gate corresponds to the R_2 gate defined in this report.

This list may be categorised in two sets depending on the smallest r for which Equation 2.1 is satisfied. Thus we find $a = \{2, 7, 8, 13\}$ satisfying $a^4 \bmod 15 = 1$ so $r = 4$ and a second set as $a = \{4, 11, 14\}$ satisfying $a^2 \bmod 15 = 1$ i.e. with period $r = 2$. These two cases shall be called the "difficult" ($r = 4$) and "easy" ($r = 2$) case. This is because for the case of $r = 2$ the modular exponentiation is simpler. Even though modular exponentiation algorithms and circuits have not been shown here, consider the binary representation of $a^x = a^{2^{n-1}x_{n-1}}...a^{2x_1}a^{x_0}$, where x_l are the binary digits of x. For the "difficult" case this means that $a^{2^l} \bmod 15 = 1$ for all $l \geq 2$, so that $f(x)$ can be simplified to multiplications controlled by just two bits, x_0 and x_1. For the "easy" case $a^2 \bmod 15 = 1$, and therefore only x_0 is relevant. How exactly these simplification are made is not shown here explicitly, yet can be found at Reference [22]. Therefore the first register of the quantum circuit can be as small as $n = 2$ in either case. This is already a major simplification of the quantum circuit. In Section 2.1.2, it was discussed that in general the number of qubits required in the first register is $n = 2\lceil \log_2 N \rceil = 2\lceil \log_2 15 \rceil = 8$, see Equation 2.26. The second register has to have $m = 4$ qubits, see Equation 2.27.
This leads one to expect a quantum circuit as, shown in Figure 3.2. A comparison of this circuit with the actual compiled quantum circuit, see Figure 3.3 b), that can be found in the paper [3] confirms the working. Note that the gates corresponding the QFT are mirrored between qubit 1 and qubit 2 in the first register. This is because the SWAP operation between just two qubits may just be omitted leading to the same result, which is another practical simplification of the quantum circuit.

3.2.2 Evolution of System

To fully appreciate the evolution of the qubits evolving in the circuit of the implementation, Figure 3.3 b), it is worth to calculate the intermediate states. Let the qubits be numbered from the top to bottom as $|\ \rangle_1 ... |\ \rangle_6$. Further, as states $|\ \rangle_4$ and $|\ \rangle_6$ remain unchanged throughout the calculation, they will be ignored. Tensor products are implicit.
Then the initial state is:

$$|0\rangle_5 |0\rangle_3 |0\rangle_2 |0\rangle_1 , \tag{3.1}$$

which evolves to

$$\frac{1}{2}\Big\{\left(\,|0\rangle_5\,|0\rangle_3\,|0\rangle_2\,|0\rangle_1\,\right)+\left(\,|0\rangle_5\,|0\rangle_3\,|0\rangle_2\,|1\rangle_1\,\right)+\left(\,|0\rangle_5\,|0\rangle_3\,|1\rangle_2\,|0\rangle_1\,\right)+\left(\,|0\rangle_5\,|0\rangle_3\,|1\rangle_2\,|1\rangle_1\,\right)\Big\}, \quad (3.2)$$

by the application of the Hadamard gates on $|\ \rangle_1$ and $|\ \rangle_2$. After the application of the two CNOT gates, representing the MFE, the state is,

$$\frac{1}{2}\Big\{\left(\,|0\rangle_5\,|0\rangle_3\,|0\rangle_2\,|0\rangle_1\,\right)+\left(\,|0\rangle_5\,|0\rangle_3\,|0\rangle_2\,|1\rangle_1\,\right)+\left(\,|1\rangle_5\,|1\rangle_3\,|1\rangle_2\,|0\rangle_1\,\right)+\left(\,|1\rangle_5\,|1\rangle_3\,|1\rangle_2\,|1\rangle_1\,\right)\Big\}. \quad (3.3)$$

Next the inverse QFT is applied. The Hadamard gate leads to canceling of terms and the R_2 gate controlled by the $|\ \rangle_1$ has no effect, yet it was implemented in the circuit for the sake of having a full proof of concept circuit. The application on the second Hadamard gate as part of the inverse QFT leads to the final state,

$$\frac{1}{2}\Big\{\left(\,|0\rangle_5\,|0\rangle_3\,|0\rangle_2\,|0\rangle_1\,\right)+\left(\,|0\rangle_5\,|0\rangle_3\,|1\rangle_2\,|0\rangle_1\,\right)+\left(\,|1\rangle_5\,|1\rangle_3\,|0\rangle_2\,|0\rangle_1\,\right)+\left(\,|1\rangle_5\,|1\rangle_3\,|1\rangle_2\,|0\rangle_1\,\right)\Big\}. \quad (3.4)$$

Upon measurement of the first and second qubit in the Z basis the two state,

$$|0\rangle_2\,|0\rangle_1 \text{ and } |1\rangle_2\,|0\rangle_1 \quad (3.5)$$

are found. This is the measurement of as estimate of $\phi = s/r$, where s is an integer between 0 and $r-1$. Therefore in this simple instance of Shor's algorithm we must have $s = 1$ as $r = 2$, no continued fraction algorithm needed, assuming the knowledge of the period r. A binary number representation of ϕ has been obtained as either $0.0_2 = 0_{10}$ or $0.1_2 = 1/2_{10}$, where the indices denote the number system. Therefore the algorithm has failed with a 50% chance returning 0 and succeeded with a 50% returning $r = 2$. The period has been obtained. Using Equations 2.4, we may now obtain the prime factors $gcd(15, 11^{2/2} \pm 1) = 3, 5$!

For a practically useful application of the algorithm a more careful treatment of the random number s has to be applied with respect to the continued fraction algorithm. Furthermore, the accuracy estimate of ϕ has to be discussed more carefully.

3.3 Conclusions of Implementation

The experiment successfully implemented a proof-of-principle demonstration of a compiled version of Shor's algorithm. The therefore required quantum nature, i.e. genuine entanglement between states, also has been proven experimentally. The experiment nevertheless does not imply the feasibility of larger implementations, further research of scalability have to be made. [3]
Notably, this implementation and others [22] [23] [24] are under critique. The paper [25] suggest that all these implementations violated a criterion stated in Shor's algorithm that the number of qubits in the first register, n, must satisfy $N^2 \leq 2^n < 2N^2$.

Chapter 4

Conclusion

This work aimed to provide an introduction to the structure and key notions of Shor's algorithm and show a simple application by explaining the quantum circuit of a recent proof-of-principle implementation. No new conclusions were drawn. Here merely an introduction to the topic was made for the target audience of master level university physics students.

Despite the age of Shor's algorithm, of now more than two decades, it appears not clear how exactly the algorithm shall be implemented in quantum circuits, when simplifications to the circuit are made. [25] The first experimental implementation of Shor's algorithm was made in 2001 by IBM [26], followed by now already half a dozen other implementations including the one explained here. [23] [27] [24] [28] The most recent implementation factored the number $N = 35$. The largest challenge remains to build a quantum computer of sufficient size to factorize larger numbers. [29] In 2018, a new method was proposed to accomplish the task of factorization with lower resource requirements, called *quantum annealing*. [30]

The experimental implementation of Shor's algorithm remains more than 25 years after its discovery an important step to practically show the superiority of quantum computing. The consequences of a scalable implementation would change the way in which we use asymmetric cryptography forever.

Bibliography

[1] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information (10th Anniversary Edition).* Cambridge University Press, 2010.

[2] Vasileios Mavroeidis, Kamer Vishi, Mateusz D., and Audun Jøsang. The impact of quantum computing on present cryptography. *International Journal of Advanced Computer Science and Applications*, 9(3), 2018.

[3] Chao-Yang Lu, Daniel E. Browne, Tao Yang, and Jian-Wei Pan. Demonstration of a compiled version of shor's quantum factoring algorithm using photonic qubits. *Physical Review Letters*, 99(25), Dec 2007.

[4] Miloslav Dušek, Norbert Lütkenhaus, and Martin Hendrych. Quantum cryptography. *Progress in Optics*, page 381–454, 2006.

[5] Shafi Goldwasser and Mihir Bellare. *Lecture Notes on Cryptography.* 8 2008. Summer course on cryptography, MIT, 1996–2001.

[6] Adi Shamir Ronald L. Rivest and Leonard M. Adleman. Cryptographic communications system and method. *Original RSA Patent as filed with the U.S. Patent Office by Rivest*, 9 1983.

[7] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, Oct 1997.

[8] Anatolii Karatsuba and Yu Ofman. Multiplication of multidigit numbers on automata. *Soviet Physics Doklady*, 7:595, 12 1962.

[9] Juan Carlos Garcia-Escartin and Pedro Chamorro-Posada. Equivalent quantum circuits, 2011.

[10] David P. DiVincenzo. Two-bit gates are universal for quantum computation. *Phys. Rev. A*, 51:1015–1022, Feb 1995.

[11] Abraham Asfaw, Luciano Bello, Yael Ben-Haim, Sergey Bravyi, Nicholas Bronn, Lauren Capelluto, Almudena Carrera Vazquez, Jack Ceroni, Richard Chen, Albert Frisch, Jay Gambetta, Shelly Garion, Leron Gil, Salvador De La Puente Gonzalez, Francis Harkins, Takashi Imamichi, David McKay, Antonio Mezzacapo, Zlatko Minev, Ramis Movassagh, Giacomo Nannicni, Paul Nation, Anna Phan, Marco Pistoia, Arthur Rattew, Joachim Schaefer, Javad Shabani, John Smolin, John Stenger, Kristan Temme, Madeleine Tod, Stephen Wood, and James Wootton. Learn quantum computation using qiskit, 2020.

[12] Hing Leung. A note on extended euclid's algorithm. *CoRR*, abs/1607.00106, 2016.

[13] Shaohua Zhang. The concept of primes and the algorithm for counting the greatest common divisor in ancient china, 2009.

[14] J Stein. Computational problems associated with racah algebra. *Journal of Computational Physics*, 1(3):397 – 405, 1967.

[15] Dr. H. Siebeck zu Breslau. Ueber periodische kettenbrueche, 1826.

[16] Igor L. Markov and Mehdi Saeedi. Constant-optimized quantum circuits for modular multiplication and exponentiation, 2015.

[17] S. Beauregard. Circuit for shor's algorithm using 2n+3 qubits. *Quantum Inf. Comput.*, 3:175–185, 2003.

[18] R. Okamoto, J. L. O'Brien, H. F. Hofmann, and S. Takeuchi. Realization of a knill-laflamme-milburn controlled-not photonic quantum circuit combining effective optical nonlinearities. *Proceedings of the National Academy of Sciences*, 108(25):10067–10071, Jun 2011.

[19] E. Knill, R. Laflamme, and G. J. Milburn. A scheme for efficient quantum computation with linear optics. *Nature*, 409(6816):46–52, 2001.

[20] S. L. Braunstein, C. M. Caves, R. Jozsa, N. Linden, S. Popescu, and R. Schack. Separability of very noisy mixed states and implications for nmr quantum computing. *Physical Review Letters*, 83(5):1054–1057, Aug 1999.

[21] Agata M. Brańczyk. Hong-ou-mandel interference, 2017.

[22] Lieven Vandersypen, Matthias Steffen, G. Breyta, Costantino Yannoni, Mark Sherwood, and Isaak Chuang. Experimental realization of shor's quantum factoring algorithm using nuclear magnetic resonance. nature, 414:883. *Nature*, 414:883–7, 12 2001.

[23] B. P. Lanyon, T. J. Weinhold, N. K. Langford, M. Barbieri, D. F. V. James, A. Gilchrist, and A. G. White. Experimental demonstration of a compiled version of shor's algorithm with quantum entanglement. *Phys. Rev. Lett.*, 99:250505, Dec 2007.

[24] Erik Lucero, R. Barends, Y. Chen, J. Kelly, M. Mariantoni, A. Megrant, P. O'Malley, D. Sank, A. Vainsencher, J. Wenner, and et al. Computing prime factors with a josephson phase qubit quantum processor. *Nature Physics*, 8(10):719–723, Aug 2012.

[25] Zhengjun Cao, Zhenfu Cao, and Lihua Liu. Remarks on quantum modular exponentiation and some experimental demonstrations of shor's algorithm, 2014.

[26] Lieven M. K. Vandersypen, Matthias Steffen, Gregory Breyta, Costantino S. Yannoni, Mark H. Sherwood, and Isaac L. Chuang. Experimental realization of shor's quantum factoring algorithm using nuclear magnetic resonance. *Nature*, 414(6866):883–887, Dec 2001.

[27] Shor's quantum factoring algorithm on a photonic chip. *Science AAAS*, 325, Sep 2009.

[28] Enrique Martín-López, Anthony Laing, Thomas Lawson, Roberto Alvarez, Xiao-Qi Zhou, and Jeremy L. O'Brien. Experimental realization of shor's quantum factoring algorithm using qubit recycling. *Nature Photonics*, 6(11):773–776, 2012.

[29] Mirko Amico, Zain H. Saleem, and Muir Kumph. Experimental study of shor's factoring algorithm using the ibm q experience. *Physical Review A*, 100(1), Jul 2019.

[30] Shuxian Jiang, Keith A. Britt, Alexander J. McCaskey, Travis S. Humble, and Sabre Kais. Quantum annealing for prime factorization, 2018.