KLE Society's

KLE Technological University, Hubballi.

A Minor Project Report

on

Integer Factorization using Shor's Algorithm

submitted in partial fulfillment of the requirement for the degree of

Bachelor of Engineering

in

Computer Science and Engineering

Submitted by

Aayush Rajwade	01FE18BCS005
Raj Jain	01FE18BCS002
Ayush Utsav	01FE18BCS059
Aman Kumar	01FE18BCS029

Under the guidance of

Mr. KMM Rajashekharaiah

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

Hubballi – 580 031

2020-21

KLE Society's

KLE Technological University, Hubballli.

2020 - 2021

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that Minor Project titled “Integer Factorization using Shor’s Algorithm” is a bonafied work carried out by the student team comprising of Raj Jain USN:01FE18BCS002, Aayush Rajwade USN:01FE18BCS005, Ayush Utsav USN:01FE18BCS059, Aman Kumar USN:01FE18BCS029 for partial fulfillment of completion of sixth semester B. E. in Computer Science and Engineering during the academic year 2020-2021.

Guide	Head, SoCSE
Mr. KMM Rajashekharaiah	Dr. Meena S. M

Viva -Voce:

	Name of the Examiners	Signature with date
1.		
2.		

Acknowledgement

We would like to thank our faculty and management for their professional guidance towards the completion of the project work. We take this opportunity to thank Dr. Ashok Shettar, Vice-Chancellor, Dr. N.H Ayachit, Registrar, and Dr. P.G Tewari, Dean Academics, KLE Technological University, Hubballi, for their vision and support.

We also take this opportunity to thank Dr. Meena S. M, Professor and Head, SoCSE for having provided us direction and facilitated for enhancement of skills and academic growth.

We thank our guide Mr. KMM Rajashekharaiah, Professor, SoCSE for the constant guidance during interaction and reviews.

We extend our acknowledgement to the reviewers for critical suggestions and inputs. We also thank Project Co-ordinator Dr. Sujatha C. and faculty in-charges for their support during the course of completion.

We express gratitude to our beloved parents for constant encouragement and support.

Raj Jain - 01FE18BCS002
Aayush Rajwade - 01FE18BCS005
Aman Kumar - 01FE18BCS029
Ayush Utsav - 01FE18BCS059

ABSTRACT

In the field of number theory, integer factorization is the decomposition of a composite number into a product of smaller integers. If the above process is only constrained to prime numbers, the process is called prime factorization. When the number is large enough, no efficient classical integer factorization algorithm is known. Different areas of mathematics and computer science have been brought to bear on the problem, which include algebraic number theory and quantum computing. The hardest instances of these problems are semi primes, the product of two prime numbers. When they both are large, even the fastest prime factorization algorithms on the fastest hardware can take enough time and resources to make the search impractical.

No algorithm can factor the integers in polynomial time. However, Peter Shor discovered an algorithm that solves it in polynomial time by harnessing the power of quantum computing.This algorithm mainly uses QFT(Quantum Fourier Transform) to find the factors.But it is a slow method because of repeated measurement of phase shift gates which increases the quantum noise .In this project, an alternate method is used to implement the algorithm using a different approach called SQFT(Sequential Quantum Fourier) which uses the combined concept of modular arithmetic,period finding and fourier transform.This approach uses less qubits as compared to previous method and detailed implementation is discussed later in this report.

The project implementation is done on the IBM Q platform using qiskit framework.The qiskit framework is limited to using only 12 qubits.The maximum probability success rate achieved is 59% using SQFT with iteration count of 22.

CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

Prime number factorization is a problem in computer science where the solution to that problem takes super-polynomial time classically. Shor's quantum factoring algorithm can solve the problem in polynomial time by harnessing the power of quantum computing. Shor's algorithm is quantum algorithm used to find the period of cyclic or periodic functions. By representing a product of two prime numbers, called the co-prime, as a periodic function using the modulo operator, and converting this equation into a form that a quantum computer can process, Shor's algorithm can determine the period of that function, which in return helps to find the factors of the number.

1.1 Motivation

Though we are decades away from having error-corrected quantum device with sufficient number of qubits. Still we can at least explore small co-prime numbers like '15'. The idea here is to explore the possibility of implementation of Shor's quantum factoring algorithm with limited resources and optimized quantum circuit. With the increasing hardware capability of quantum computers in terms of qubits and reduced quantum noise, the work done can later be tested on sufficiently powerful quantum computers and the tedious task of factoring large co-prime numbers in future might become slight easier.

1.2 Literature Review / Survey

Quantum mechanics is a fundamental theory in physics which describes the inner workings of the universe on the smallest energy scale [5]. This fundamental theory establishes new understandings and possibilities previously untapped by classical physics. Today the quantum computer is one of the biggest scientific breakthroughs in the 20^{th} century. A quantum computer is made by infusing the theory of quantum mechanics into computer science. It harnesses the power of quantum mechanics to perform computation [7]. Factoring integers is generally considered to be a tedious task on a classical computer specially when the number to be factored is large. But with the advancement in quantum technology this might be possible in near future that to in polynomial time. The algorithm that is used in quantum computers is Shor's quantum factoring algorithm developed by Peter W. Shor [3]. For any given number n, he gave a technique of finding the order r of an element x (mod n) instead of giving a quantum algorithm directly for the very same purpose. This indirect algorithm is feasible due to factorization being reduced to find the order of an element [6]. To find a factor of an odd number n, given a method for computing the order r of x, choose a random x (mod n), find its order r, and compute gcd($x^{r/2}$ - 1,n). The Euclidean algorithm can be used to compute gcd($x^{r/2}$-1,n) in polynomial time. Since ($x^{r/2}$ - 1)($x^{r/2}$ + 1) = x^r - 1 0(mod n), the numbers gcd($x^{r/2}$ − 1, n) and gcd($x^{r/2}$ + 1, n) will be two factor of n. This procedure fails only if r is odd, in which case r/2 is not integral, or if $x^{r/2}$ -1(mod n), in which case the procedure yields the trivial factors 1 and n. Using this criterion, it can be 1shown that this procedure, when applied to a random x(mod n), yields a nontrivial factor of n with probability at least 1 − 1/2k-1, where k is the number of distinct odd prime factors of n [4].

1.3 Problem Statement

To perform integer factorization using Shor's Quantum Algorithm.

1.4 Applications

- The algorithm proposed can be used to factor large numbers which can be used break RSA cryptography with sufficient number qubits in future.
- In near future this proposed algorithm can have many application in the domain of quantum cryptography(called post-quantum cryptography).

1.5 Objectives and Scope of the project

Following are the objectives and scope of Shor's quantum factoring algorithm using SQFT (Sequential Quantum Fourier Transform). The objectives are elementary since there is scalability issue persisting in the quantum domain.

1.5.1 Objectives

- To do integer factorization using Shor's Quantum Algorithm.
- To do factoring of integers with maximum possible probability success rate.
- To reduce the number of iterations.

1.5.2 Scope of the project

Shor's Algorithm is a Polynomial time factoring algorithm which works on quantum computing. GNFS (General Number Field Sieve) is the most efficient classical factoring algorithm but cannot factor integers in polynomial time. Shor's Algorithm emerged to be efficient solution for this problem. The method proposed here however is only limited to an initial stage like factoring of co-prime numbers as small as 15 because of constraints in quantum hardware resource. The scope of this project is limited to exploratory level.

So, in the next chapter the different requirement analysis is discussed that includes functional, non-functional, hardware and, software requirements that will shed further shed light on the resources required for the implementation phase.

Chapter 2

REQUIREMENT ANALYSIS

Quantum Computation is the field that investigates the computational power and possibilities of computers based on quantum mechanics, while classical computers are based on classical mechanics. The interest in quantum computers is due to the fact that it is possible to find quantum algorithms that are significantly faster than all known classical algorithms solving the same problem. Shor's quantum algorithms take a polynomial amount of steps, namely the factoring algorithm takes asymptotically O((log n)2(log log n)(log log log n)) steps along with a polynomial amount of time on a classical computer to convert the output of the quantum computer to factors of n.

2.1 Functional Requirements

Since this project is limited to an exploratory level of playing with the qubits, quantum circuits and the algorithm. So the functional requirement is limited to a basic level because of complex domain understanding as well as constraint in resources availability.

Following are the functional requirements-

- To do integer factorisation using shor's quantum algorithm using SQFT.
- To do factoring of integers with maximum possible probability success rate.

2.2 Non Functional Requirements

Even with so much boom in quantum domain in recent times there is still a lot to be done in terms of hardware as well as software capabilities.Currently IBM has made available only 5 qubit processor which limits the testing and application capability of our proposed methodology.

Following are the Non-Functional requirements-

- To reduce the number of iteration count under 25.
- To reduce the usage of qubits which should be under 20.

2.3 Hardware Requirements

Quantum computers are very sensitive devices which are maintained under zero temperature for it's working because qubits are very hard to store and processing information is even harder. So these devices are very rare and require a lot of investment and maintenance and can only be done by some big organization in terms of money and infrastructure and that's why it is very hard to access them directly and is available to only selected organization like IBM, Google and some heavily funded startup etc.

2.4 Software Requirements

Since quantum computers are very rare and high end machines which are available to selected large companies and institutions like IBM, GOOGLE. The implementation of this project is done with the help of IBMQ which is a cloud service provided by IBM to run quantum codes and circuits. It helps the researchers and students across the globe in testing their ideas and hypothesis. Though it is limited to only using 15 qubits. It takes requests and queue them as per fcfs basis and also on the number of jobs on different quantum computers across the world. So , this project's implementation is totally on this cloud platform available to us by IBM.

Through this chapter it is realised that hardware implementation is very far sighted. However, the software implementation in the form of cloud services is made available for testing by different tech giants. In the next chapter the proposed system and the component level design of the algorithm is discussed taking into account the component level designs.

Chapter 3

SYSTEM DESIGN

In this chapter the proposed system and the component level design of the circuit used in the implementation of Shor's quantum algorithm using SQFT is discussed. The component level design explains the basic implementation part in an abstract manner which helps in getting the bigger picture of what's happening in implementation part.

3.1 Component Level Design

Shor's algorithm is divided into five stages. In the first stage the quantum registers are initialized. In the second stage the superposition state of all the integers in the quantum registers is generated by applying Hadamard gate (H) on each of the qubits separately. In the third stage the algorithm is executed the unitary operator Uf entangles each input value with it's corresponding function value. In the fourth stage the quantum fourier transformation is applied on the quantum registers which squeezes the probability amplitude into peak due to it's periodicity. In the last stage the measurement of quantum register finally yields the period 'r '.Figure 3.1 gives the clear picture of the above discussed 5 stages of implementation.

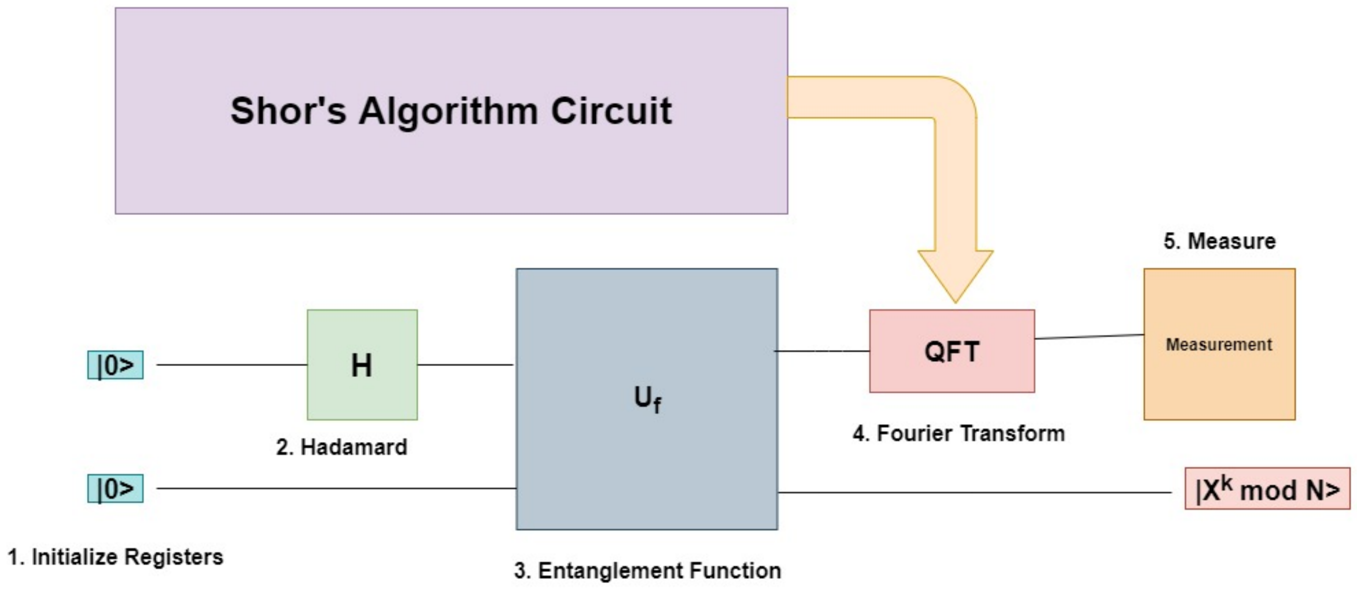


Figure 3.1: Component level design

Chapter 4

IMPLEMENTATION

Before diving deep into the implementation part there are some basic terminologies and concepts that is important to understand. In the following section there is a brief discussion about qubits(short of Quantum bit),Multi-qubit systems, entangled states, quantum gates like NOT gate, entanglement ,One-Qubit Hadamard Gate, Control-NOT gate .

4.1 Qubit

Like classical computes work on bits the same way the quantum computers has qubits (quantum bits). Qubit is the basic unit of information he quantum version of the classic binary bit physically realized with a two-state device. Following is the representation of qubits mathematically.

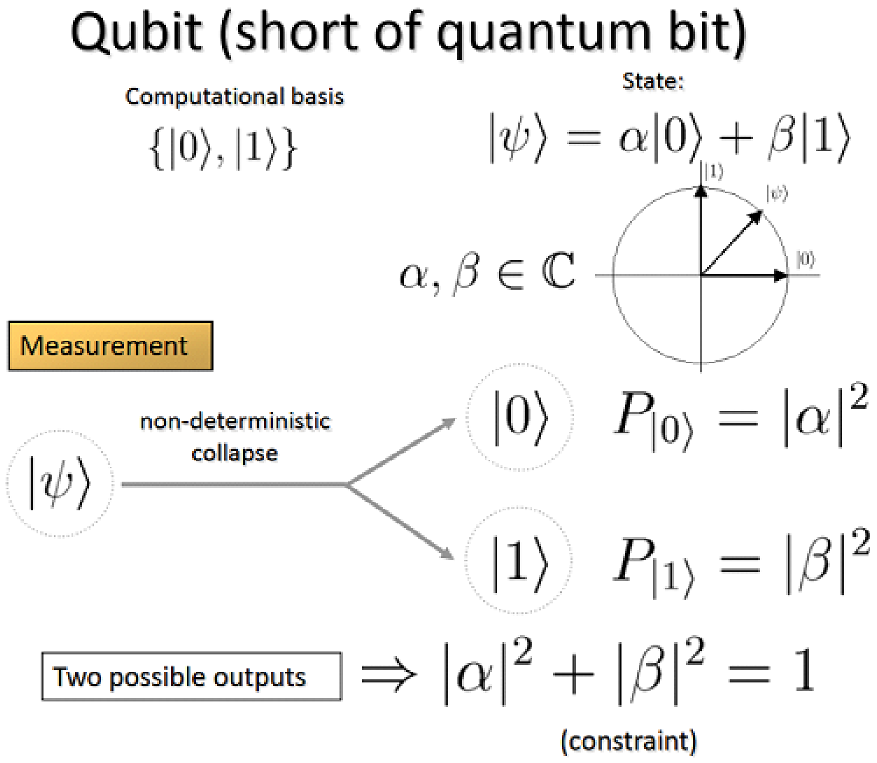
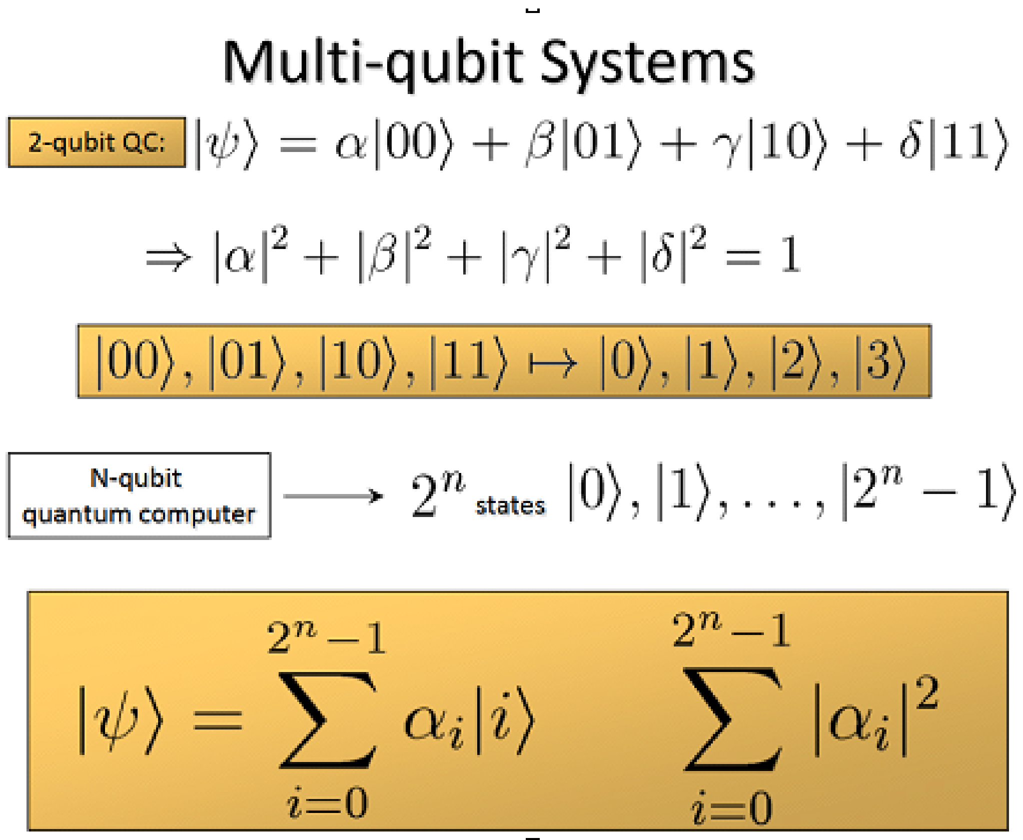

Qubit (short of quantum bit)
Computational basis
$\{|0\rangle, |1\rangle\}$
State:
$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$
$|1\rangle$
$|\psi\rangle$
$|0\rangle$
$\alpha, \beta \in \mathbb{C}$
Measurement
non-deterministic collapse
$|\psi\rangle$
$|0\rangle$
$P_{|0\rangle} = |\alpha|^2$
$|1\rangle$
$P_{|1\rangle} = |\beta|^2$
Two possible outputs
$\Rightarrow |\alpha|^2 + |\beta|^2 = 1$
(constraint)

Figure 4.1: Working of qubit

4.2 Multi-qubit system

Multi-qubit system contains more than 2 qubits. These systems are used in the implementation of different quantum ideas , to design and, quantum circuit and algorithms.



Multi-qubit Systems

2-qubit QC: $|\psi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle$

$$\Rightarrow |\alpha|^2 + |\beta|^2 + |\gamma|^2 + |\delta|^2 = 1$$

$$|00\rangle, |01\rangle, |10\rangle, |11\rangle \mapsto |0\rangle, |1\rangle, |2\rangle, |3\rangle$$

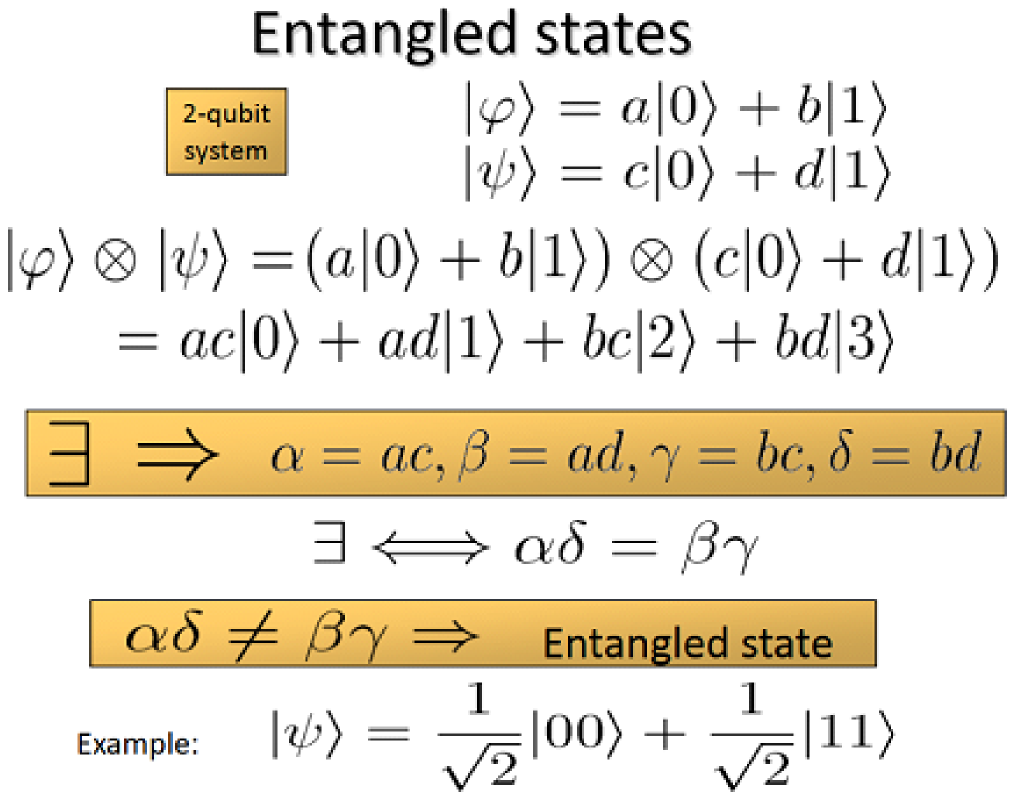
N-qubit quantum computer $\longrightarrow$ 2^n states $|0\rangle, |1\rangle, \ldots, |2^n - 1\rangle$

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

Figure 4.2: Multiqubit system

4.3 Entangled States

When the particles are entangled the quantum state cannot be factored as a product of its state which on simpler terms means that they are not individual particles anymore but are inseparable as a whole. So one constituent cannot be fully described without considering the other.



Entangled states

2-qubit system

$$|\varphi\rangle = a|0\rangle + b|1\rangle$$
$$|\psi\rangle = c|0\rangle + d|1\rangle$$

$$|\varphi\rangle \otimes |\psi\rangle = (a|0\rangle + b|1\rangle) \otimes (c|0\rangle + d|1\rangle)$$
$$= ac|0\rangle + ad|1\rangle + bc|2\rangle + bd|3\rangle$$

$$\boxed{\exists \implies \alpha = ac, \beta = ad, \gamma = bc, \delta = bd}$$

$$\exists \iff \alpha\delta = \beta\gamma$$

$$\boxed{\alpha\delta \neq \beta\gamma \implies \text{Entangled state}}$$

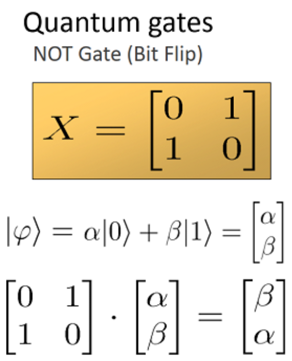
Example: $|\psi\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$

Figure 4.3: Working of entangled states

4.4 Quantum Gates

- The NOT Gate

  This is quantum counterpart of classical NOT gate . It basically is Pauli X gate which performs negation. Hence it corresponds to a classical gate .



Quantum gates

NOT Gate (Bit Flip)

$$\boxed{X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}}$$

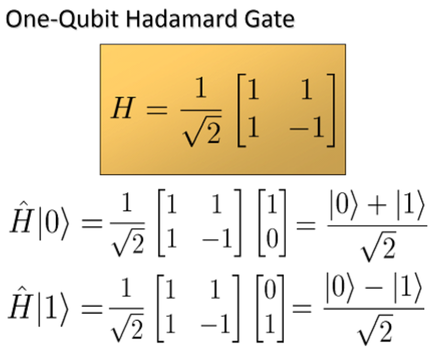
$$|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$$

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \beta \\ \alpha \end{bmatrix}$$

Figure 4.4: NOT Gate

- One-qubit Hadamard Gate

  Also known as the H gate . This gate is used to convert the qubit from clustering state to uniform superimposed state .



One-Qubit Hadamard Gate

$$\boxed{H = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}}$$

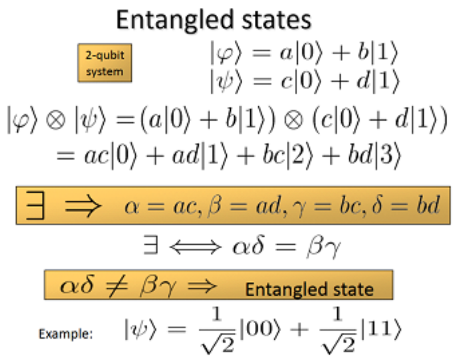
$$\hat{H}|0\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\begin{bmatrix}1\\ 0\end{bmatrix} = \frac{|0\rangle + |1\rangle}{\sqrt{2}}$$

$$\hat{H}|1\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\begin{bmatrix}0\\ 1\end{bmatrix} = \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

Figure 4.5: Hadamard gate

- Control-NOT gate

  This gate is essential as it is used for the construction of a gate-based quantum computer. It is often used to entangle and disentangle the states.



Entangled states

2-qubit system

$$|\varphi\rangle = a|0\rangle + b|1\rangle$$
$$|\psi\rangle = c|0\rangle + d|1\rangle$$

$$|\varphi\rangle \otimes |\psi\rangle = (a|0\rangle + b|1\rangle) \otimes (c|0\rangle + d|1\rangle)$$
$$= ac|0\rangle + ad|1\rangle + bc|2\rangle + bd|3\rangle$$

$$\boxed{\exists \;\Rightarrow\; \alpha = ac, \beta = ad, \gamma = bc, \delta = bd}$$

$$\exists \iff \alpha\delta = \beta\gamma$$

$$\boxed{\alpha\delta \neq \beta\gamma \Rightarrow \text{Entangled state}}$$

Example: $|\psi\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$

Figure 4.6: CNOT Gate

The algorithm is implemented into 2 main parts :

- Conversion of the problem of the factoring to the problem of finding the period(implemented classicaly).
- Finding the period(Quantum period finding) using the sequential quantum Fourier transform, and is responsible for quantum speedup.

4.5 Module 1

Classical Part: Converting the problem of factoring to the problem of finding the period is discussed in module 1. Following are the sequential steps that needs to followed to get the period.

1. A random number a<N picked.
2. Compute gcd(a,N) .This may be done using the Euclidean algorithm.
3. If gcd(a,N) 1, then there is a nontrivial factor of N.
4. f(x+r) = a^{x+r} mod N = a^x mod N = f(x).
5. If r is odd, we go to step 1.
6. If$a^{r/2}$ -1 (mod N), go back to step 1.
7. gcd($a^{r/2}$ $\pm$ 1,N) is a non-trivial factor of N.

4.6 Module 2

Quantum Part: In this module the use of SQFT is discussed. Following are the steps that need to be followed to get the period and results.

1. Choose an integer q such that N^2 < q < 2 N^2 . Since the biggest number to be chosen that we can go for is 15 and 15^2 = 225 .We have to choose a number q such that it lies b/w 225 and 450 .
2. Choose a random integer x such that GCD(x,N) = 1. Let's choose the number 7 as the gcd(7,15) = 1.
3. Create the quantum registers (these registers must also be entangled so that the collapse of the input register corresponds to the collapse of the output register). The number of qubit requirement is chosen on the basis of following points.
   - Input register : must contain enough qubits to represent a number that should be as large as q-1 i.e. up to 255 . So when we equate
     2^k= 255
     k = 8

- Output register : It must have a definite number of qubits such that it is enough to represent a number large as N-1 i.e. up to 14 .So when we equate

  $2^k = 14$

  $k = 4$

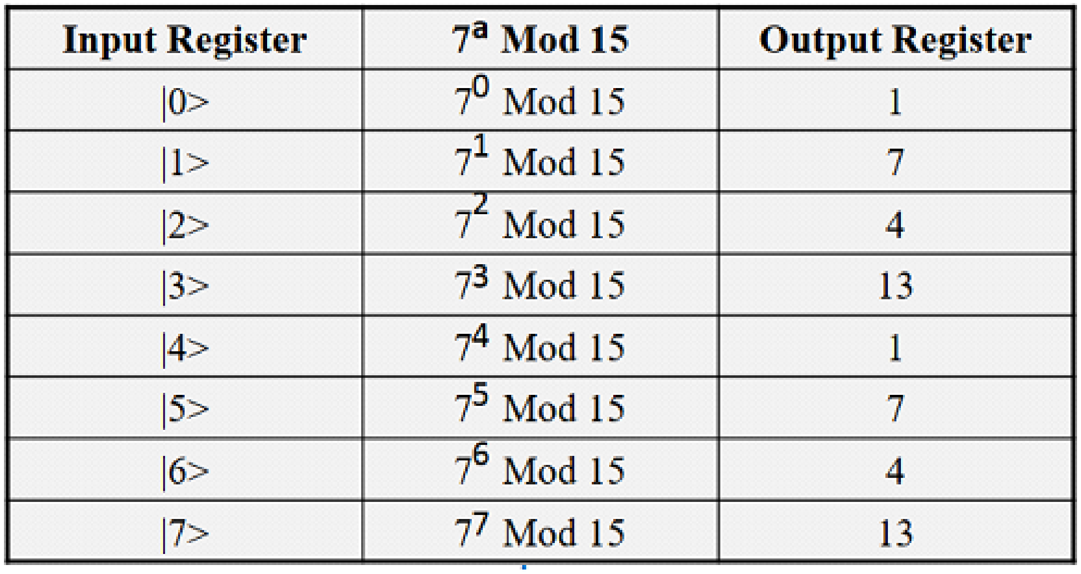
4. The input register is loaded with an equally weighted superposition of all integers from 0 to q-1 i.e . from 0 to 255 .

5. We load the output register with all zeroes. The total state of the system at this point will be :

   $\frac{1}{\sqrt{256}}\sum_{a=0}^{255} |a, 000>$

6. The initial term from sqrt(256) to 'a' is representing the input register and |000> is representing the output register and comma denotes that registers are being entangled.

7. Then there is application of transformation x^a mod N to each number in the input register, storing the result of each computation in the output register.



Input Register	7^a Mod 15	Output Register
|0>	7^0 Mod 15	1
|1>	7^1 Mod 15	7
|2>	7^2 Mod 15	4
|3>	7^3 Mod 15	13
|4>	7^4 Mod 15	1
|5>	7^5 Mod 15	7
|6>	7^6 Mod 15	4
|7>	7^7 Mod 15	13

Figure 4.7: Modular operation of $x^a modN$

8. Superposition collapse using inverse sqft. Here we take the measurement on the output register. Due to this there is a collapse from the superposition state to just one of the results of the transformation, let it be called as d.

   Our output register will collapse to represent one of the followings:

   $| 1 >, | 4 >, | 7 >,$ or $| 13 >$

9. We know that the registers that are initialized are entangled, so when we measure the output register will have the effect of the input register which are partially collapsing into equal superposition of each state b/w 0 to q-1 that yielded d (the value of the collapsed output register.)

   The output register collapsed to $| 1 >$, the input register will partially collapse to following

   $\frac{1}{\sqrt{64}}|0> + \frac{1}{\sqrt{64}}|4> + \frac{1}{\sqrt{64}}|8> + \frac{1}{\sqrt{64}}|12....$

   The probabilities in the above case are 1/64 since our register is now an equal superposition as 64 value(0,4,8.,. . . ..252) .

10. After the above steps there is the application of Sequential Quantum Fourier transform on the input register that are partially collapsed . The transform will take the effect of taking a state $| a >$ and transforming it into a state given by :

    $\frac{1}{\sqrt{q}}\sum_{d=0}^{q-1}|d> * e^{2\pi iad/256}$

    $\frac{1}{\sqrt{64}}\sum_{a\epsilon A}^{n}|a>, |1>$,here on replacing a to $\frac{1}{\sqrt{256}}\sum_{d=0}^{255}|c> * e^{2\pi iad/256}$

    Here A is the set of all the values that 7a mod 15 yielded 1. In our case

    A = 0,4,8,. . . . . ,252

    So we obtain the final state of the input register after the QFT is :

    $\frac{1}{\sqrt{64}}\sum\frac{1}{\sqrt{256}}\sum_{d=0}^{255}|c> * e^{\frac{2\pi iac}{256}}, |1>$

    After this step the SQFT will peak the probability amplitude at integer multiples of q/4 which in our case is 256/4, or 64.

    $| 0 >, | 64 > , | 128 > , | 192 >,. . . . . .$

    So now we no longer have an equal superposition of states , the probability amplitudes of the above states are now higher than the other states in our register. The next step is the measurement of the register, and it collapses with high probability to one of these multiples of 64, we can call this as p. Since 'q' and 'p' are known we calculate the period by using continuous fraction expansion of the ratio b/w q and p .

11. Once we have the period, the factors of N can be determined by taking the greatest common divisor of N with respect to $x^{(P/2)} + 1$ and $x^{(P/2)} - 1$. So this computation is done on a classical computer.

    Computation :

    Gcd ($7^{4/2}$ + 1, 15) = 5

    Gcd ($7^{4/2}$ - 1, 15) = 3

    So we successfully factored 15.

4.6.1 Circuit design

Quantum computers do not operate on high level programming languages like python, java, solidity etc. They operate using complex circuits as an input. These complex circuits are constructed via different quantum gates as discussed in brief in section of quantum gates. Below are the different circuits that were used to achieve the proposed implementation. Firstly there will be discussion on normal quantum Fourier transform circuit and then the sequential Quantum Fourier transform circuit. Later comparisons are made on accuracy of factoring integers.

4.6.2 QFT Circuit

In this method after applying Hadamard gate to each qubit we are also applying rotation gate to measure phase shift. Since we know that on measurement of quantum system there is slight noise called the quantum noise because of Heisenberg uncertainty principle which says that on measurement of quantum system leads to uncertainty which in turn create quantum noise. So greater the uncertainty greater is the noise. Here in the circuit it can be seen that every qubit has a rotation gate to measure the phase shift which is increasing the uncertainty as well as the noise.

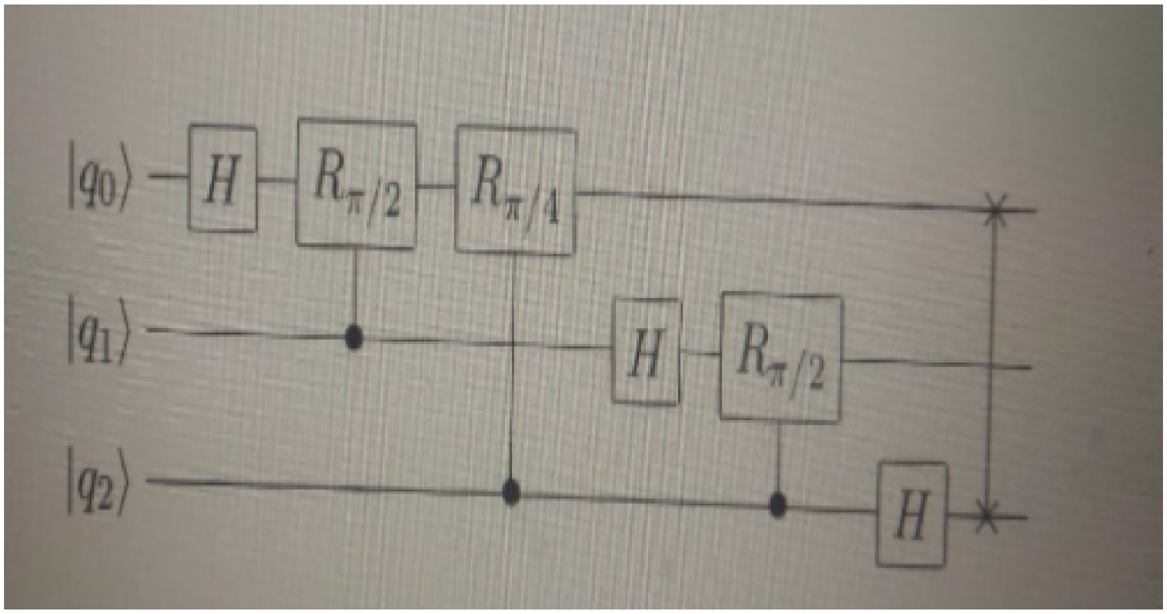


Figure 4.8: QFT Circuit diagram

4.6.3 SQFT Circuit

It stands for Sequential Quantum Fourier Transform. In this method the first eight qubits act as input qubit stored in input registers are superimposed. After superimposition on register two SQFT is applied xk mod N. The measurements array preparation could be made by equal weighted superposition of the value q. This process will produce the value of q/r where in the later stages it can be used to find the desired period.

The value of N is the number of bits required to represent the integer value of q-1 while k represents register one that is loaded with equal superposition of integers up to q-1. Modular exponentiation is done by performing modular multiplication qubit by qubit. All the values inside the array measurements are added to find the value of q/r which in turn helps to find the period and hence the factors.

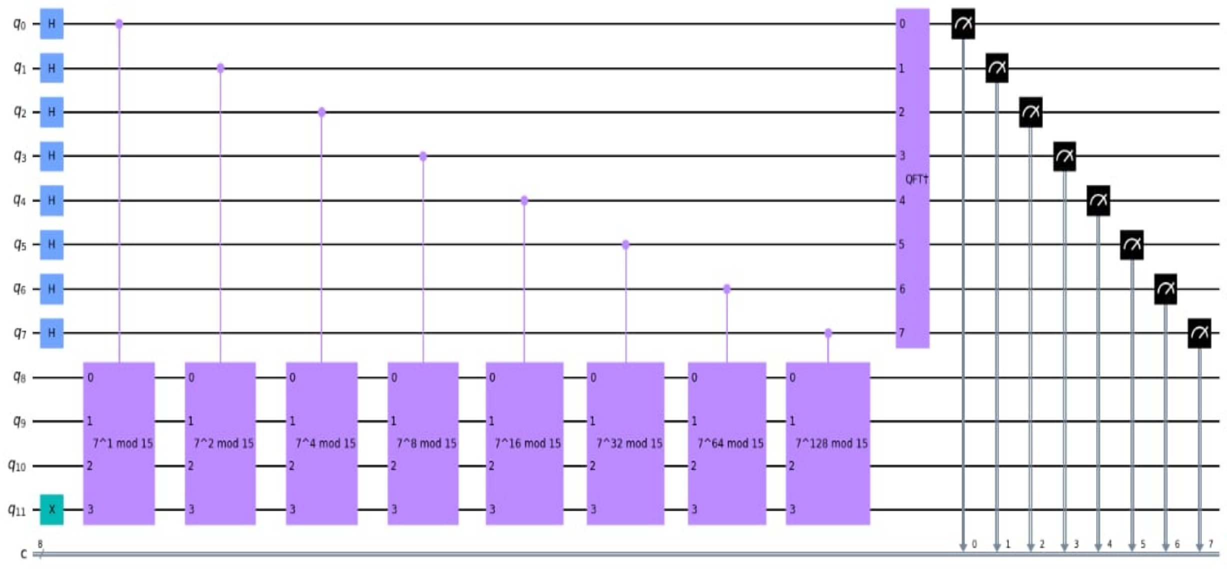


Figure 4.9: SQFT(Sequential Quantum Fourier Transform) Circuit diagram

4.6.4 Comparison between QFT and SQFT

QFT does not use modular exponentiation. Instead it relies on phase estimation which slows down the computation process because each qubit used in QFT act as speed breaker in computation. However in SQFT there is use of periodicity, modular arithmetic which accelerates the process of factoring because there is no need now to calculate phase estimation on each qubit. This work is done in SQFT part by applying inverse QFT and then measurement are done to get the result.

Chapter 5

RESULTS AND DISCUSSIONS

On applying SQFT and running the program we got varying probability success rate for different value of x and the number of iterations. On trying successive test runs different accuracies were generated. The final accuracy attained was 59.9909 % with an iteration count of 22. While performing different test runs it is observed that the accuracy and number of iteration are related to each other in a probabilistic fashion.

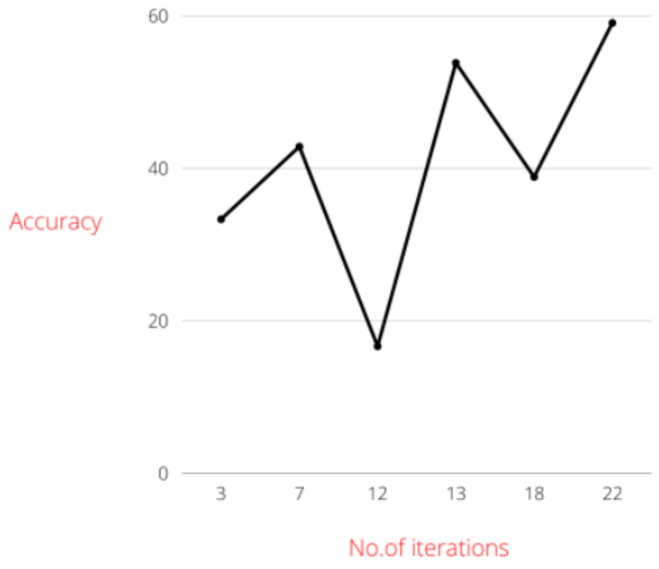


Figure 5.1: Accuracy vs Interaction count graph

On plotting the accuracy vs no. of iteration graph it can be analysed weather the increase in the number of iteration causes the accuracy to increase or not. From the fig 5.1 it can be seen that when number of iteration is 7 the obtained accuracy was 42.8571 %, then for the 12 iteration value the accuracy is decreased to 16.6667%, then on the 16 iteration value it can be seen that the accuracy increases to 53.8462%.

This pattern gives the proof the probabilistic nature of quantum computation in the Shor's quantum algorithm for integer factorization and also proves that there is no correlation between number of iteration and accuracy.

Chapter 6

CONCLUSION AND FUTURE SCOPE OF THE WORK

In this project the proposed implementation of Shor's quantum factoring algorithm has been implemented successfully using IBMQ qiskit framework. It could also be concluded that probabilistic nature of quantum algorithm can be further improved with sufficient number of qubits and more advanced algorithms. The implementation developed from this project achieved probability success rate for factoring is 59.9909% with an iteration count of 22.

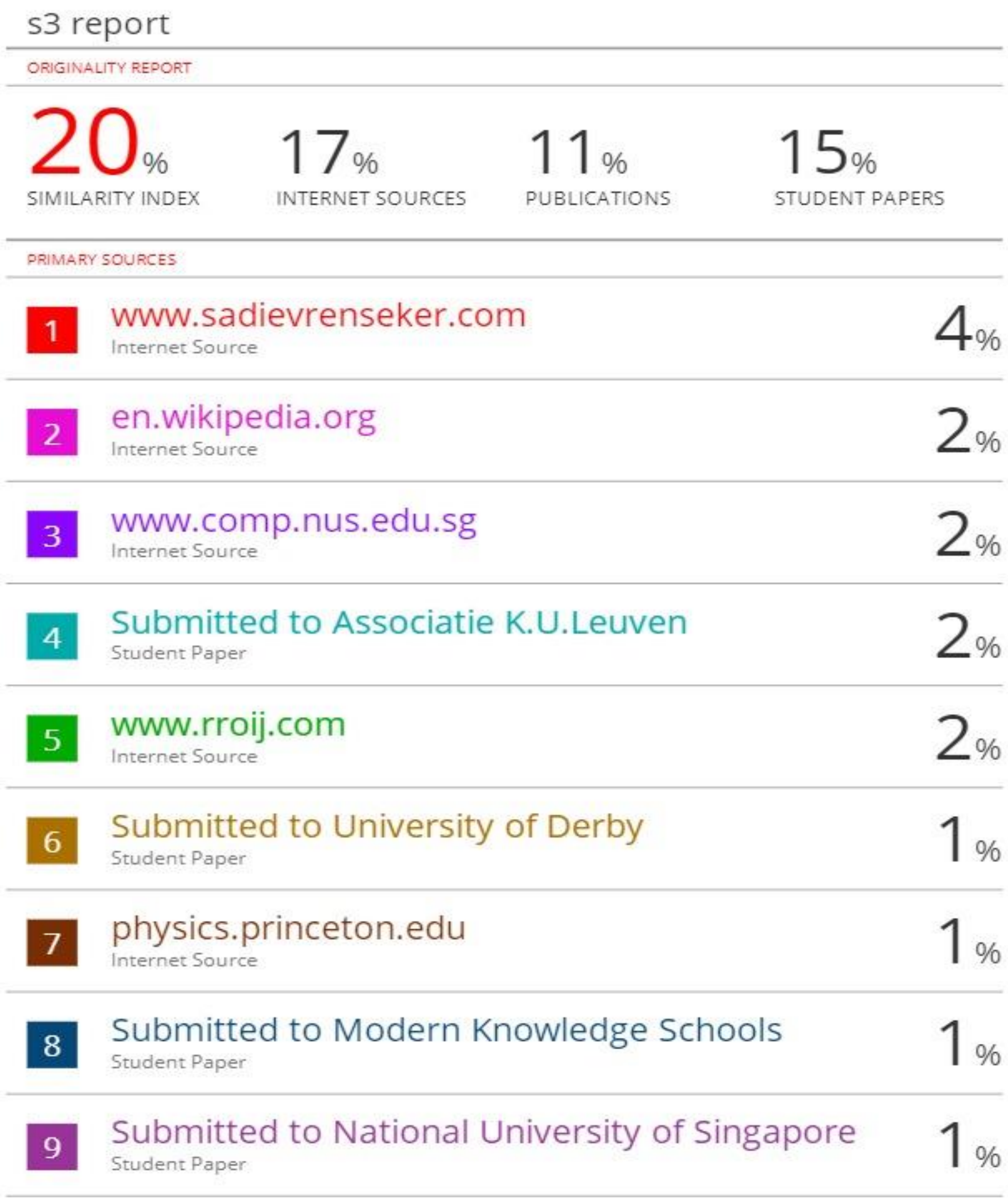
If sufficient powerful quantum computer is developed in future, the proposed methodology can be tested to evaluate it's efficiency of calculating correct factors of large integers.Till than different approaches for implementing the Shor's algorithm can be tried and tested on the current resources.

Plagiarism Report

Project Title: Integer Factorization using Shor's Algorithm

Team No: S3

Project Domain: Quantum Computing and Number Theory



s3 report

ORIGINALITY REPORT

20%	17%	11%	15%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.sadievrenseker.com Internet Source	4%
2	en.wikipedia.org Internet Source	2%
3	www.comp.nus.edu.sg Internet Source	2%
4	Submitted to Associatie K.U.Leuven Student Paper	2%
5	www.rroij.com Internet Source	2%
6	Submitted to University of Derby Student Paper	1%
7	physics.princeton.edu Internet Source	1%
8	Submitted to Modern Knowledge Schools Student Paper	1%
9	Submitted to National University of Singapore Student Paper	1%

10	James F. Peters, Arturo Tozzi. "Quantum Entanglement on a Hypersphere", International Journal of Theoretical Physics, 2016 Publication	1 %
11	Xiangping Meng, Zhaoyu Pian. "Vulnerability Assessment of the Distribution Network Based on Quantum Multiagent", Elsevier BV, 2016 Publication	1 %
12	Submitted to University of Michigan-Shanghai Jiao Tong University Joint Institute Student Paper	1 %
13	Submitted to SVKM International School Student Paper	<1 %
14	www.phys.nthu.edu.tw Internet Source	<1 %
15	www.semanticscholar.org Internet Source	<1 %
16	Submitted to Cardiff University Student Paper	<1 %
17	opensiuc.lib.siu.edu Internet Source	<1 %
18	www.authorstream.com Internet Source	<1 %

19	rd.springer.com Internet Source	<1 %
20	uwspace.uwaterloo.ca Internet Source	<1 %
21	journal.uad.ac.id Internet Source	<1 %
22	www.unep.org Internet Source	<1 %

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		

REFERENCES

[1] Wicaksana, Arya, and Adjie Wahyu Wicaksono. "Web-app realization of Shor's quantum factoring algorithm and Grover's quantum search algorithm."Telkomnika18, no. 3 (2020): 1319-1330.

[2] Monz, Thomas, Daniel Nigg, Esteban A. Martinez, Matthias F. Brandl, Philipp Schindler, Richard Rines, Shannon X. Wang, Isaac L. Chuang, and Rainer Blatt. "Realization of a scalable Shor algorithm."Science351, no. 6277 (2016): 1068-1070.

[3] Shor, Peter W. "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer."SIAM review41, no. 2 (1999): 303-332.

[4] Cao, Zhengjun. "A Note on Shor's Quantum Algorithm for Prime Factorization."IACR Cryptol. ePrint Arch.2005 (2005): 51.

[5] Abhijith, J., Adetokunbo Adedoyin, John Ambrosiano, Petr Anisimov, Andreas Bärtschi, William Casper, Gopinath Chennupati et al. "Quantum algorithm implementations for beginners."arXiv e-prints(2018): arXiv-1804.

[6] Miller, Gary L. "Riemann's hypothesis and tests for primality."Journal of computer and system sciences13, no. 3 (1976): 300-317.

[7] Speiser, Jacqueline. "Implementing and Comparing Integer Factorization Algorithms."