Quantum Factorization: Exploiting Shor's Algorithm for Efficient Prime Factorization

Alisha Karna, *Bipul Bhattarai*, *Madhavi Awasthi*, *Prashant Pant*, *Pramod Acharya*, *Sandeep Chataut*, *Srijana Raut*

***Abstract*— The factorization of large numbers plays a crucial role in various cryptographic schemes and mathematical computations. Traditional algorithms for factorization, although effective for small numbers, become computationally expensive as the size of the number increases. In this paper, we explore the groundbreaking solution provided by Shor's algorithm, leveraging the unique properties of quantum computers. Shor's algorithm exhibits a remarkable speedup in factorization, revolutionizing the field by enabling efficient factorization of large numbers that were previously considered intractable. This paper discusses the fundamental principles of Shor's algorithm, its underlying mathematics, and its potential implications for cryptography and computational complexity. By harnessing the power of quantum computing, Shor's algorithm opens up new avenues for tackling complex factorization problems and lays the foundation for advancements in secure communication and computational efficiency.**

***Index Terms*—Factorization, Quantum Computing, Traditional algorithm, Shor's Algorithm**

I. Introduction

Quantum Computing (QC) originated in the 1980s, harnessing the principles of Quantum mechanics, including techniques like Entanglement and superposition. Quantum computers, which were introduced in 1994, perform computations using these principles. The first problem addressed by QC was solving a sudoku game using 16 qubits [1]. Unlike conventional computers with two states (0 and 1), QC utilizes qubits, where each bit can have two possible states. As of 2018, the largest quantum computer consisted of 72 qubits [2]. In contrast to classical computers, which use up, down, left, and right movements for a spin, quantum computers only use upward and downward movements.

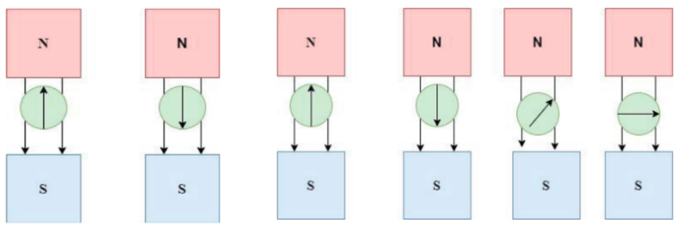

Fig. 1. Spin Movement in case of Quantum Computers (Left). Spin Movement in the case of Classical Computers (Right), is different from classical computing since it is based on the possibilities of amplitudes to give the best results.

This paper is a part of the Quantum Computing Course for replicating the quantum algorithm for learning purposes at the University of South Dakota.

These concepts rely on the principle of superposition, which states that combining two states results in a valid state, as described by the superposition principle. Qubits, representing particles, can be in states such as 00, 01, 10, and 11. In this representation, 0 represents an up spin, while 1 represents a down spin. Let's consider an equation, $\phi = \alpha|0> + \beta|1>$. Here ϕ represents quantum state, and $\alpha|0> + \beta|1>$, represents superposition such that the value of $|0>$ is [1 0] and that of $|1>$ is [0 1] [3].

The main goal of quantum factorization is to efficiently factorize composite numbers into their prime factors using Shor's technique. An important issue in complexity theory and cryptography is prime factorization. As the numbers get bigger, the conventional factorization algorithms get slower. Peter Shor developed Shor's algorithm, which uses quantum computing to greatly accelerate factorization, in 1994 [4]. In comparison to classical techniques, Shor's algorithm offers a solution that is tenfold faster by mixing classical and quantum elements. A crucial part of Shor's algorithm is the quantum order finding algorithm, which effectively establishes the rank of group members and makes it possible to identify prime factors. Every integer n is uniquely decomposable into a product of primes, as was known before Euclid. For almost as long, mathematicians have been wondering how to factor a number into this product of primes. However, it wasn't until the 1970s that academics examined the asymptotic running times of factoring algorithms and applied the paradigms of theoretical computer science to number theory [Adleman 1994]. As a result, factoring algorithms now operate much more efficiently. Shor's factoring algorithm is composed of multiple classical routines that come before and after a quantum order finding algorithm. Although the classical tasks are known to be efficient when executed on a classical computer, the process of order finding is considered difficult to solve using classical methods. However, it is recognized that this specific aspect of the algorithm can be efficiently carried out on a quantum computer [5], [6].

A. Entanglement

When two systems are in a quantum entangled state, no matter how far apart they are, learning something about one system immediately reveals something about the other. Einstein was perplexed by this phenomenon since it defies the axiom that no information can be conveyed faster than the speed of light and referred to it as "a spooky action at a distance." However, more investigation into the use of photons and electrons to validate entanglement. In quantum computing, entanglement is crucial and is thought to be the main reason why some algorithms—most notably Shor's Algorithm—perform better than their classical equivalents (Kendon & Munro, 2005). In contrast to classical computers, which carry out operations sequentially, bit by bit, quantum computers may carry out arithmetic and logical operations on many qubits simultaneously because to the phenomena of entanglement. In quantum computers, entanglement enables quick state transitions in paired qubits, increasing computing speed [7]. The processes cannot be directly scaled by increasing the qubit count because each qubit discloses information about many entangled qubits. According to research, quantum entanglement is essential for quantum algorithms to achieve exponential speedups over classical computations. Pairs or groups of particles can show behavior known as quantum entanglement, which prevents the states of the particles from being described independently. No matter how far apart the particles are, changing the state of one will change the state of the others. Even if entangled particles are separated by great distances, such as light years, changing the state of one particle will still cause the other particles' states to change [8].

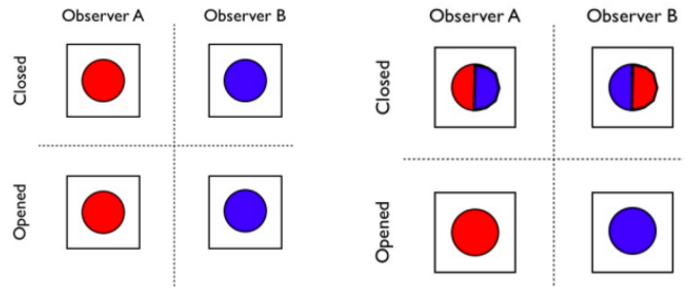

Fig. 2. Left(Non-Entangled Balls) The ball colors remain consistent for both observers, regardless of whether the boxes are open or closed., Right(Entangled Balls), In a parallel universe, observer A finding a red ball implies that observer B has a blue ball, even without opening his box. This collapse of states is due to entanglement between their balls.

B. *Prime Factorization & Shor's Algorithm*

Prime factorization is the process of expressing an integer as a product of prime numbers. It was initially seen as a theoretical concept until researchers discovered its practical application in cryptography in 1976. This led to the development of Public Key Cryptography (PKC), where encryption and decryption are done using separate keys. PKC enables secure communication, especially in scenarios where multiple users need to send information to a central recipient, such as businesses receiving sensitive data from customers. Quantum computers can solve complex tasks much faster than classical computers [9]. They excel at trial-and-error tasks by trying multiple solutions simultaneously. Shor's Algorithm is a hypothetical quantum algorithm that can quickly factorize prime numbers, posing a threat to current encryption systems. However, quantum computers are not yet powerful enough to run Shor's Algorithm effectively. Nevertheless, a newly developed algorithm allows them to factorize primes even faster [10].

The prime factorization problem introduced through Shor's algorithm is. Given an integer N, find exactly two primes p and q such that N = pq. Here, the goal of the prime factorization issue is to factor huge integers. The quantum part of Shor's algorithm has a run time of:

$$\mathcal{O}(logN^2).log((log(N))) \tag{1}$$

, This is significantly more efficient than the GNFS algorithm which has a run time of

$$\mathcal{O}(exp((log(N))^{1/3}.(log(logN))^{2/3})) \tag{2}$$

In other words, Shor's technique finds the period, r, of some superposed periodic function $f(x) = a^x mod(N)$ to solve the prime factorization problem for big numbers in polynomial time and then applies classical algorithms to find a factor corresponding to the period. If r is even, then for some integer x, we can solve, $gcd(x^{r/2} - 1, N$ and $gcd(x^{r/2} + 1, N$, where gcd(a, b) is the greatest common divisor of a and b, i.e., the largest integer that divides both a and b. The Euclidean algorithm [Knuth 1981] can be used to compute gcd(a, b) in polynomial time. Which will ideally give a factor of N. If r is not even then $(x^{r/2} \pm 1, N)$, will not be an integer which means we do not have a valid prime factor of N. In such situations, we can iterate the process and continue attempting different values of "r" until we find the correct factors. Finding the period is the trickiest element of this prime factorization issue, which Shor did brilliantly [11].

Shor's Algorithm consists of two main components. The initial part converts the prime factorization problem into the period finding problem, which can be executed on a classical computer. The second part of the algorithm employs the Quantum Fourier Transform to determine the period. While it may appear that both parts could be implemented on a classical computer, the goal is to factor in large positive composite integers. Implementing a Discrete Fourier Transform on a large set would be impractical and inefficient. Shor's Algorithm takes advantage of the quantum computer's capability to perform simultaneous operations on quantum states in superposition. This parallel computing is achieved through entanglement and superposition. Instead of repeatedly applying the DFT on a classical computer to find the period, Shor's Algorithm performs the Quantum Fourier Transform once on all states, providing a high probability of obtaining the period [12].

The Euclidean algorithm aims to determine the greatest common divisor of two integers, a and b. The initial step involves fitting a and b into an equation, where q0 represents the number of times b fits into a, and r0 is the remainder. $a = q_0 b + r_0$, This step then continues in this fashion by using b and the remainder r0, and then with r0 and the next remainder r1, until there is a remainder of 0. $b = q_1 r_0 + r_1$, $r_0 = q_2 r_1 + r_2$,...,.$r_n = q_{n+2} r_{n+1} + r_1 + 0$ Once the remainder is zero, the answer is given as r_{n+1}

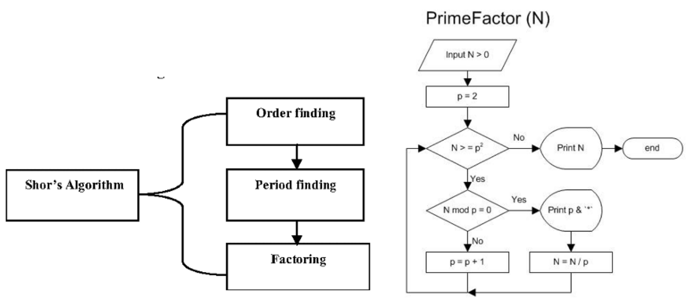

Fig. 3. Left(Shows the Concatenation of three algorithms for shor's algorithm). Right(Flow Chart of Algorithm.)

C. *QUANTUM FOURIER TRANSFORM*

The DFT takes some vector of input data, applies a transformation on the vector, then outputs the transformed data. Commonly in signal processing, this would be like taking a signal which resides in some time domain and applying a transformation to that signal and the result outputs all of the frequencies within that signal i.e. the output of the signal is now in the frequency domain [13]. Suppose we have a vector of length N with input values $x_0, x_1, ...x_{N-1} \in C$. The DFT takes these input values and outputs a vector of transformed values calling them $y_0, y_1, ...y_{N-1} \in C$ by performing the following transformation

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=1}^{N-1} x_j e^{2\pi ijk/N} \tag{3}$$

The Quantum Fourier Transform (QFT) is the same linear transformation as the DFT applied to quantum bits. More specifically, the QFT applies a DFT to the amplitudes of some quantum state. As we will later see, Shor's Algorithm exploits this transformation in order to solve the factorization problem. The QFT acting on some orthonormal basis $|0>, |1> ....|N-1>$ is a linear operator written as

$$QFT(|x>= \frac{1}{\sqrt{N}} \sum_{x=1}^{N-1} x_j e^{2\pi ixk/N} |K> \tag{4}$$

Let F denote the QFT, then F is a unitary operator, which is $F^+F = FF^+ = I$. Let $|\tilde{\psi}>= F|\psi>$ for some quantum state ψ. Here $\tilde{\psi}$ represents the quantum state after the transformation has been applied. We can write the unitary operator, F, as

$$F = \frac{1}{\sqrt{N}} \sum_{j'=0}^{N-1} \sum_{k'=0}^{N-1} e^{2\pi ij'k'/N} |k'><j'| \tag{5}$$

II. METHODOLOGY

A. *Complexity*

Complexity refers to the level of intricacy, difficulty, or sophistication in a system, process, or problem. It is a measure of how complexity and difficulty. The difference between the complexity of classical and quantum computers are [14];

The complexity of quantum computer:

$$\mathcal{O}\left((\log N)^2 (\log\log N)(\log\log\log N)\right)$$

Complexity on classical computer:

$$\mathcal{O}\left(e^{1.9(\log N)^{\frac{1}{3}}(\log\log N)^{\frac{2}{3}}}\right)$$

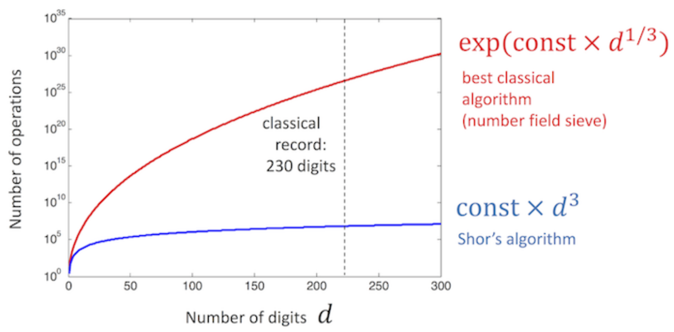

Fig. 4. Complexity of Classical vs quantum Computer: Copied from IBM Quantum

B. *Shor's Algorithm*

1) Classical Part: - The factoring problem can be reduced to the problem of order-finding using a classical computer.

2) Quantum Part: - There exists a quantum algorithm capable of solving the order-finding problem.

Shor's algorithm relies on a fundamental finding from number theory. The result is;

The usefulness of the function $\mathcal{F}$(a) in factoring large numbers lies in its periodic nature [15]. This means that there is a specific period, denoted as r, for which $\mathcal{F}$(a) repeats its values. Starting from $x^0 \ mod \ n = 1$, as we increase the exponent by multiples of r (such as $x^r \ mod \ n = 1$ and $x^{2r} \ mod \ n = 1$), the function continues to yield the same result due to its periodic behavior.

$$x^r \equiv 1 \mod n$$

$$(x^{r/2})^2 \equiv x^r \equiv 1 \mod n$$

$$(x^{r/2})^2 - 1 \equiv 0 \mod n$$

And if r is an even number:

$$(x^{r/2} - 1)(x^{r/2} + 1) \equiv 0 \mod n$$

If we have a nontrivial value for $x^{r/2}$ (not equal to ± 1), then the product $(x^{r/2}-1)(x^{r/2}+1)$ is a multiple of the number we want to factor, denoted as n. This implies that either $(x^{r/2}-1)$ or $(x^{r/2}+1)$ shares a nontrivial factor with n. By calculating the greatest common divisor (gcd) of $(x^{r/2}-1)$ and n, as well as the gcd of $(x^{r/2}+1)$ and n, we can obtain a factor of n.

To factor an odd integer N, such as N = 15, using the example given, we can follow these steps:

1. Determine the Nature of N: Check if N is a prime number, an even number, or an integer power of a prime number. In the case of N = 15, it is neither prime nor even, and not an integer power of a prime number.

2. Choose an Integer q: Select an integer q that satisfies the condition $N^2 < q < 2N^2$. For our example, let's pick q = 256.

3. Choose a Random Integer x: Randomly choose an integer x such that the greatest common divisor (GCD) of x and N is 1. In this case, let's select x = 7.

4. Create Two Quantum Registers: Prepare two quantum registers: an input register and an output register.

a. Input Register: Load the input register with an equally weighted superposition of all integers from 0 to q-1 (from 0 to 255 in this case). This requires 8 qubits to represent numbers as large as q-1.

b. Output Register: Initialize the output register with all zeros. It needs to contain enough qubits to represent numbers as large as N-1 (up to 14 in this case), so we need 4 qubits.

5. Apply the Transformation x mod N: Apply the transformation x mod N to each number in the input register, storing the result of each computation in the output register. This operation calculates the remainder of dividing each input number by N.

6. Measurement on the Output Register: Perform measurement on the output register. This will collapse the quantum state and yield one of the results of the transformation, denoted as c.

These steps are part of Shor's algorithm, a quantum algorithm used for factoring large numbers efficiently. By applying these steps, we can leverage the power of quantum computation to determine factors of the given odd integer N.

C. *Implementation details*

We further discuss Qiskit used in this study:

1) Qiskit: The Qiskit is an open-source software development kit (SDK) provided by IBM for working with quantum computers. It is a comprehensive framework that allows researchers, developers, and enthusiasts to design, simulate, and run quantum algorithms, as well as work with quantum hardware.

For reproducibility, the code is available at: https://github.com/rautsrijana/Quantum$_Computing$.

D. *Performace metrics*

III. Result and Analysis

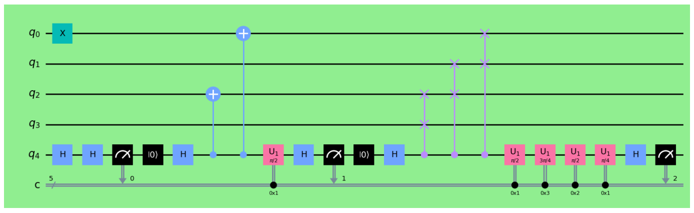

Fig. 5. Circuit Diagram implemented in our Shor's algorithm for prime factorization.

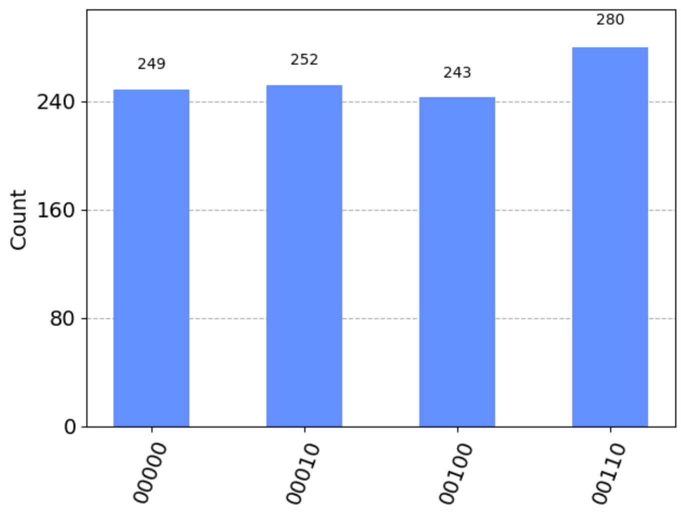

Fig. 6. Bar graph obtained from simulation, which shows the counts for different states.

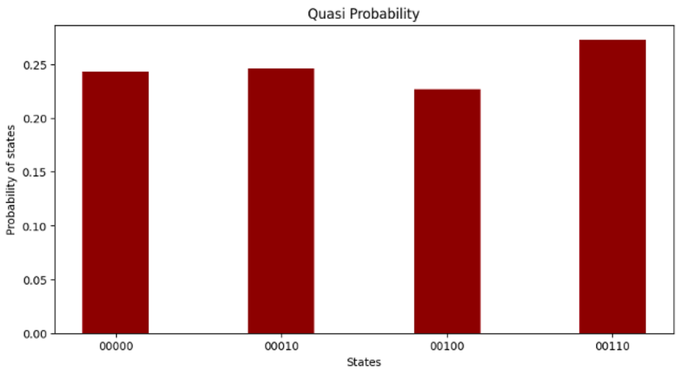

Fig. 7. Shown are the probabilities of different possible quantum states while simulating the shor's algorithms with 1024 shots. The probability of these states signifies that the occurrence of these states is equally probable between 0.24 to 0.28.

IV. Discussion and Conclusion

After the measurements are performed in the circuit, we observe that the outcomes x = 0, 2, 4, and 6 occur with equal probabilities, accounting for some noise. In typical cases, at the continued fraction expansion for $x/2^k$ would be used. However, in this specific scenario where r divides q (the desired period), there is no need to employ the continued fraction expansion. We can simply compute $p = GCD(a^{r/2} + 1, 15) = 5$ and $q = GCD(a^{r/2} - 1, 15) = 3$. Our simulated results occurred with the equally probable for 4 different quantum states. This attribute of Shor's algorithm factorizes large numbers much faster than classical algorithms, making significant advancements in number theory and cryptography. With higher probabilities, its reliability and practical feasibility are highlighted. Its remarkable success rate has the potential to revolutionize cryptography and encryption, instilling confidence in its practical applications and its potential impact on multiple industries.

Shor's algorithm is exponentially faster than conventional algorithms because it makes use of quantum computing principles. It has major consequences for subjects like cryptography, data security, and computational mathematics since it has the ability to undermine encryption techniques that depend on the difficulty of prime factorization.

ACKNOWLEDGMENT

We would like to express our sincere gratitude to Dr. Rodrigue Rizk for his invaluable guidance and support throughout the entire Quantum Computing Course. His expertise and mentorship have been instrumental in shaping the direction and quality of this work. We are particularly grateful for Professor Rizk's insightful feedback on our research methodology and his acceptance of this topic. Thank you for your unwavering support.

REFERENCES

[1] John Kubiatowicz. The future is quantum computing? In *2007 IEEE Hot Chips 19 Symposium (HCS)*, pages 1–5. IEEE, 2007.

[2] Chris Bernhardt. *Quantum computing for everyone*. Mit Press, 2019.

[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):250504, 2007.

[4] Stephane Beauregard. Circuit for shor's algorithm using 2n+ 3 qubits. *arXiv preprint quant-ph/0205095*, 2002.

[5] Zhengjun Cao and Lihua Liu. A note on one realization of a scalable shor algorithm. *arXiv preprint arXiv:1611.00028*, 2015.

[6] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.

[7] Richard Jozsa. Quantum factoring, discrete logarithms, and the hidden subgroup problem. *Computing in science & engineering*, 3(2):34–43, 2001.

[8] Yahui Wang, Huanguo Zhang, and Houzhen Wang. Quantum polynomial-time fixed-point attack for rsa. *China Communications*, 15(2):25–32, 2018.

[9] Artur Ekert and Richard Jozsa. Quantum computation and shor's factoring algorithm. *Reviews of Modern Physics*, 68(3):733, 1996.

[10] Ronald De Wolf. Quantum computation and shor's factoring algorithm. *CWI and University of Amsterdam*, 1999.

[11] Lieven MK 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, 2001.

[12] 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):012305, 2019.

[13] Shuangbao Paul Wang and Eric Sakk. Quantum algorithms: overviews, foundations, and speedups. In *2021 IEEE 5th international conference on cryptography, security and privacy (CSP)*, pages 17–21. IEEE, 2021.

[14] Baonan Wang, Feng Hu, Haonan Yao, and Chao Wang. Prime factorization algorithm based on parameter optimization of ising model. *Scientific reports*, 10(1):1–10, 2020.

[15] Aminah Albuainain, Jana Alansari, Samiyah Alrashidi, Wasmiyah Alqahtani, Jana Alshaya, and Naya Nagy. Experimental implementation of shor's quantum algorithm to break rsa. In *2022 14th International Conference on Computational Intelligence and Communication Networks (CICN)*, pages 748–752. IEEE, 2022.