Prime Factorization of Large Numbers using Shor's Algorithm

CSC 792: – Quantum Computing

Raunak Nandkumar More
Dept. of Computer Science
University of South Dakota
raunaknandkumar.more@coyotes.usd.edu

Sai Neeraj Samineni
Dept. of Computer science
University of South Dakota
saineeraj.samineni@coyotes.usd.

Sai Sriman Kudupudi
Dept. of Computer Science
University of South Dakota
saisriman.kudupudi@coyotes.usd.edu

Hansakrish kuttuva bhasker
Dept. of Computer Science
University of South Dakota
hansakrish.kuttuvaba@coyotes.usd.edu

Venkata Suryadeepak lakhmipalli
Dept. of Computer Science
University of south Dakota
venkatasuryadeepak.l@coyotes.usd.edu

Kalainidhi Tirounavoucarassou
Dept. of Computer Science
University of South Dakota
kalainidhi.tirounavo@coyotes.usd.edu

Damareswara Maddepalli
Dept. of computer science
University of south Dakota
damareswara.maddepal@coyotes.usd.edu

ABSTRACT:

This project explores the implementation and implications of Shor's algorithm for prime factorization on large numbers. Shor's algorithm, a ground-breaking quantum algorithm, shows tremendous potential in efficiently finding prime factors compared to classical factorization methods. However, practical implementation faces challenges due to the limitations of current quantum hardware and the need for error correction. Quantum computers are still in their early stages, and error correction is crucial to ensure accurate computations. Despite these limitations, ongoing advancements in quantum computing technologies and the development of error correction codes hold promise for the future realization of efficient large number factorization. This research contributes to the understanding of Shor's algorithm and highlights its potential impact on various fields reliant on prime factorization.

1. INTRODUCTION:

Prime factorization is a critical problem in number theory with applications spanning various fields, including cryptography and number theory. This algorithm has significant implications for cryptography and the security of various encryption schemes that rely on the difficulty of factoring large numbers. The paper utilizes the Qiskit framework, a widely-used open-source software development kit for quantum computing, to simulate quantum circuits and execute quantum operations. It leverages the quantum resources of the qasm_simulator backend to perform the required quantum computations. The main goal of the code is to break down a given integer, N, into its prime factors using Shor's algorithm. The algorithm achieves this by finding the period of a chosen number, a, modulo N. The code proceeds by generating random values of a and checking if the greatest common divisor (gcd) between a and N is not equal to 1. If the gcd is

not 1, the algorithm successfully factors N, and the factors are returned. If the gcd is 1, the code determines the period using quantum circuits and measurements. The period determination process involves initializing quantum registers, applying quantum gates such as X (Pauli-X) and CX (controlled-X), and measuring the qubits. The measured results are then analyzed to extract the most frequent output, representing the period. By implementing Shor's algorithm, this code provides a demonstration of the potential power of quantum computing in solving complex mathematical problems efficiently. The ability to factor large integers has significant implications for cryptography and the security of various systems, making this implementation a valuable contribution to the field of quantum algorithms.

This paper aims to explore the existing research and advancements in prime factorization of large numbers using Shor's algorithm. The focus will be on investigating the theoretical foundations of Shor's algorithm, including its mathematical framework and its reliance on principles of quantum computing. By delving into these aspects, we seek to gain a deeper understanding of the potential and limitations of Shor's algorithm for prime factorization and its implications for the field of cryptography.

Shor's Algorithm: A Brief Overview

Shor's algorithm, developed by mathematician Peter Shor in 1994, is a powerful quantum algorithm for integer factorization. It represents a significant milestone in the field of quantum computing, challenging our perception of which computational problems are tractable. Its emergence has spurred the exploration of new quantum algorithms and inspired efforts to design and build quantum computers. Shor's algorithm has not only revolutionized the study of factorization but has also accelerated research into alternative cryptosystems that are not reliant on integer factorization.

Several teams have successfully demonstrated the implementation of Shor's algorithm for specific composite integers. For instance, in 2001, a group employing seven NMR qubits factored the number 15 into 3 multiplied by 5. Since then, other teams have achieved this feat using different platforms, including four photon qubits in 2007, three solid-state qubits in 2012, and five trapped ion qubits in 2016. Furthermore, in 2012, a combination of a photon qubit and a qutrit (a three-level system) was utilized to factor the number 21 into 3 multiplied by 7.

Peter Shor has described Shor's algorithm as consisting of three main components. The first component transforms the factorization problem into a period-finding problem using number theory, a step that can be executed on a classical computer. The second component employs the quantum Fourier transform to determine the period, which is responsible for the algorithm's exponential speedup compared to classical methods. Finally, the third component utilizes the discovered period to calculate the prime factors of the given composite number.

In summary, Shor's algorithm has had a profound impact on the field of quantum computing, providing a ground-breaking approach to factorizing large integers. Its successful experimental implementations on various quantum platforms have validated its theoretical foundations. Understanding the algorithm's three essential steps enables researchers to explore its potential applications and further advance the development of quantum computing technologies.

Factorized Quantum Simulator:

The factorized quantum simulator is a software tool specifically designed to replicate the behavior of quantum circuits that implement Shor's Algorithm. Its purpose is to provide researchers with a virtual environment where they can experiment, test, and optimize the algorithm's performance without relying on

physical quantum hardware. The simulator integrates various essential elements, including quantum gates, qubits, measurement operations, and classical post-processing algorithms, to faithfully model the execution of Shor's Algorithm.

By leveraging the capabilities of this simulator, researchers gain the ability to examine and analyze the behavior of Shor's Algorithm across different scenarios. This includes exploring varying input sizes, noise models, and error correction techniques. Such flexibility enables a comprehensive assessment of the algorithm's robustness, scalability, and potential limitations. Furthermore, it assists researchers in devising strategies to mitigate errors and enhance the algorithm's performance when implemented on real-world quantum computers.

In essence, the factorized quantum simulator serves as a powerful tool for understanding and optimizing Shor's Algorithm's behavior. It empowers researchers to gain valuable insights into the algorithm's performance characteristics, aiding in the advancement of practical applications and the development of error mitigation strategies in the quest for efficient large number factorization.

Potential Applications and Implications:

The factorized quantum simulator using Shor's Algorithm has numerous applications and implications in various fields:

a. Cryptography: Shor's Algorithm poses a significant threat to classical cryptographic schemes, particularly the widely-used RSA encryption. By simulating the algorithm, researchers can assess the impact of quantum computers on cryptography and explore post-quantum cryptographic solutions.

b. Optimization: Shor's Algorithm and its simulation can be applied to solve optimization problems by transforming them into the factorization problem. This provides potential benefits for domains such as logistics, finance, and data analysis.

c. Quantum Error Correction: Simulating Shor's Algorithm can aid in the development and optimization of error correction techniques, which are vital for ensuring the reliability and scalability of quantum computations.

2. Problem Statement: -

The problem statement of this project is to implement shor’s algorithm for integer factorization using quantum computing principles. Shor's algorithm is a quantum algorithm that can efficiently factorize large composite numbers into their prime factors. The code aims to factorize a given number N into its prime factors p and q. This implements Shor's algorithm for integer factorization. It uses the Qiskit library for quantum circuit simulation. The period function calculates the period of a function a^x mod N, which is a crucial step in the algorithm. The ‘shor’s breaker’ function applies Shor's algorithm to factorize the given number N. It utilizes the period function to find the period and then determines the prime factors p and q based on the calculated period. The code prompts the user to enter a number, and the prime factors are printed as the output.

METHODOLOGY: -

BACKGROUND:

The fundamental principles of quantum computing.

Quantum Superposition:

Quantum superposition is one of the key principles of quantum mechanics that distinguishes quantum computing from classical computing. In classical computing, a bit can be either in the state of 0 or 1. However, in quantum computing, a qubit (quantum bit) can exist in a superposition of states, representing both 0 and 1 simultaneously. Mathematically, a qubit can be described as a linear combination of the basis

states |0⟩ and |1⟩. For example, a qubit can be in a state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, where α and β are complex probability amplitudes that determine the probability of measuring the qubit in either state upon measurement.

Entanglement: Entanglement is another fundamental principle of quantum mechanics that allows qubits to become correlated in such a way that the state of one qubit is dependent on the state of another, even if they are physically separated. When qubits are entangled, the state of the whole system cannot be described independently by the individual states of the qubits. The measurement of one qubit instantly determines the state of the other qubit, regardless of the distance between them. This phenomenon enables quantum computers to perform certain calculations more efficiently than classical computers and is essential for various quantum algorithms.

Quantum Gates:

Quantum gates are analogous to classical logic gates and form the building blocks of quantum circuits. Quantum gates manipulate the quantum states of qubits to perform specific computations. These gates are represented by unitary matrices that act on the state vector of the qubits. Just as classical logic gates transform classical bits, quantum gates transform quantum states. Examples of commonly used quantum gates include:

a. Hadamard Gate (H gate): It puts a qubit into superposition, creating an equal probability of measuring it in either the |0⟩ or |1⟩ state.

b. Pauli-X Gate (X gate): It performs a bit-flip operation, changing the state |0⟩ to |1⟩ and vice versa.

c. Pauli-Y Gate (Y gate): It performs a bit-flip and a phase-flip operation simultaneously.

d. Pauli-Z Gate (Z gate): It performs a phase-flip operation, changing the sign of the |1⟩ state.

e. Controlled-NOT Gate (CNOT gate): It applies an X gate to a target qubit only if a control qubit is in the |1⟩ state. It is a fundamental two-qubit gate for creating entanglement and performing computations.

These gates, along with other gates, such as phase gates, T gates, and controlled gates, allow the manipulation of quantum states and enable quantum algorithms to perform complex computations in parallel through the exploitation of superposition and entanglement.

These principles form the foundation of quantum computing and enable the potential for quantum computers to solve certain problems exponentially faster than classical computers.

i. Shor's Algorithm: -

This algorithm is a quantum algorithm specifically designed for prime factorization. It combines classical and quantum computing techniques to efficiently factorize large numbers. The algorithm leverages the properties of quantum superposition and interference to find the factors of a composite number exponentially faster than classical algorithms. It involves the initialization of a random number, applying the QFT to detect the period, and then using classical mathematical techniques, such as continued fractions, to determine potential factors.

Performing Quantum Modular

Exponentiation:

Quantum modular exponentiation is a crucial step in Shor's algorithm. It involves preparing a quantum state that represents the modular exponentiation function, f(x) = a^x mod N, where 'x' is the input register and 'N' is the number to be factorized. By applying quantum gates and operations, Shor's algorithm efficiently evaluates this function for various values of 'x' in parallel.

ii. Implementation:

This code implements Shor's algorithm, a ground-breaking algorithm in the field of quantum computing, to factorize a given number, N. By harnessing the power of quantum mechanics, Shor's algorithm can efficiently determine the prime factors of large numbers, which has significant implications for cryptography, number theory, and various industries reliant on prime factorization.

The code begins by importing the necessary libraries, including qiskit, a popular open-source quantum computing framework. Qiskit provides a range of tools and functionalities for simulating and executing quantum circuits.

The main function, shors_breaker(N), is the heart of the algorithm. It takes an input number, N, and attempts to find its prime factors. The function employs a randomized approach by generating a random number, a, and checks if the greatest common divisor (gcd) of a and N is not 1. If the gcd is not 1 or if N itself equals 1, it implies that a factor of N has been found, and the function returns the gcd and N divided by the gcd as the prime factors.

However, if the gcd is 1 and N is not 1, the function proceeds to the core of Shor's algorithm: finding the period, r. The period(a, N) function is responsible for this crucial step. It initializes a quantum circuit with a pre-defined number of available qubits to perform the necessary quantum calculations. The function prepares the initial quantum state based on a randomly chosen value, x0. It then applies quantum gates, including controlled operations and modular exponentiation, to compute the period of a modulo N.

To simulate the quantum circuit, the code utilizes the qasm_simulator backend from the qiskit_Aer module. The circuit is executed on this simulator, and the measurement results are obtained. The code selects the most frequent measurement outcome as the determined result, denoted as 's'. This outcome plays a crucial role in determining the factors of N.

If the obtained period, r, is odd or if a raised to the power of (r/2) modulo N equals -1, the function continues to the next iteration. These conditions are important to ensure the algorithm's correctness and accuracy. Otherwise, the function proceeds to calculate potential factors, p and q, using the gcd of (a^(r/2) + 1, N) and (a^(r/2) - 1, N). If either p or q equals N, it indicates that the factors could not be determined in the current iteration, and the function moves on to the next iteration.

Finally, if valid factors p and q are found, the function returns them as the prime factors of N.

The code prompts the user to enter a number and ensures that the input is positive. It then calls the shors_breaker function on the input number and prints the resulting prime factors, if found.

It is important to note that this implementation is a simulation running on a classical computer and not on an actual quantum computer. Quantum computers are still in their early stages of development, and practical implementations of Shor's algorithm for large-scale factorization are currently limited due to the constraints of current quantum hardware and the need for error correction. Nevertheless, this code provides a valuable insight into the inner workings of Shor's algorithm and its potential for efficiently factorizing large numbers.

In conclusion, this code demonstrates the implementation of Shor's algorithm for prime factorization using the qiskit library. Shor's algorithm has the potential to revolutionize the field of cryptography and significantly impact various industries. As quantum computing technologies continue to advance, it is expected that practical implementations of Shor's algorithm for large number factorization will

become a reality, enabling breakthroughs in encryption.

Quantum Computing and Factorization:

The reason period finding is important for factorization is that Shor's algorithm can efficiently find the period using quantum parallelism and interference. This algorithm is exponentially faster than classical algorithms for factorization, making it a significant breakthrough in the field.

However, it's worth noting that large-scale, fault-tolerant quantum computers capable of executing Shor's algorithm on sufficiently large numbers have not yet been built. Current quantum computers are limited in terms of qubit coherence and error rates, making it challenging to factorize large numbers using Shor's algorithm.

Qiskit overview:

Qiskit is an open-source quantum computing framework developed by IBM. It provides a comprehensive set of tools and libraries for working with quantum computers, simulators, and quantum algorithms.

Quantum Circuit Design: Qiskit allows users to define and manipulate quantum circuits using a simple and intuitive programming interface. It provides a set of quantum gates and operations that can be combined to construct quantum circuits. Users can specify the number of qubits, apply single-qubit and multi-qubit gates, and perform measurements.

Quantum Simulation: Qiskit provides powerful simulators that enable users to simulate the behavior of quantum circuits on classical computers. This allows researchers and developers to test and debug their quantum algorithms before running them on actual quantum hardware. Qiskit simulators support various levels of simulation, from ideal noise-free simulations to more realistic noisy simulations that capture the effects of errors.
Quantum Hardware Access: Qiskit provides access to real quantum hardware through cloud-based quantum systems, such as IBM Quantum Experience. Users can submit their quantum circuits for execution on actual quantum devices. This feature allows researchers and developers to experiment with real quantum hardware and explore the behavior and limitations of current quantum technologies.

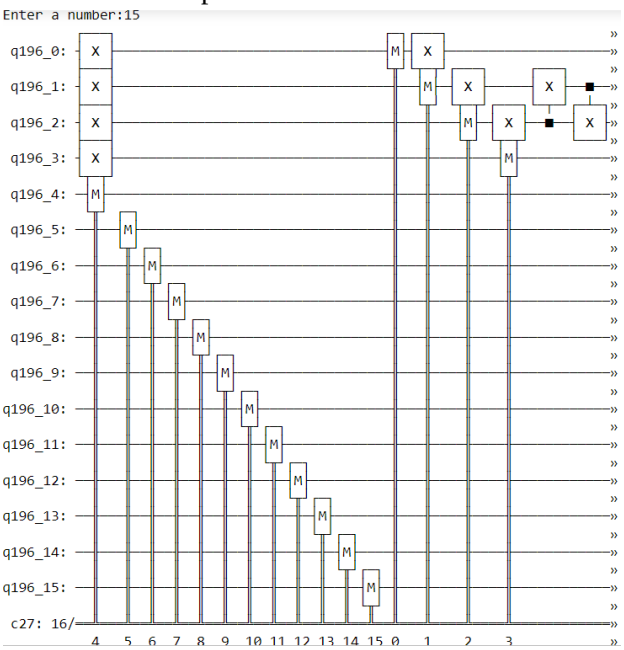
Quantum Algorithm Libraries: Qiskit includes a growing collection of pre-built quantum algorithms and techniques that can be used as building blocks for developing new applications. These libraries cover a wide range of topics, including optimization, chemistry, machine learning, and quantum error correction. Users can leverage these libraries to accelerate their research and development of quantum applications.

Visualization Tools: Qiskit provides various tools for visualizing quantum circuits and results. Users can visualize and analyze the structure of their quantum circuits, inspect the state and measurement outcomes, and explore the entanglement and connectivity of qubits. Visualization aids in understanding and debugging quantum algorithms.

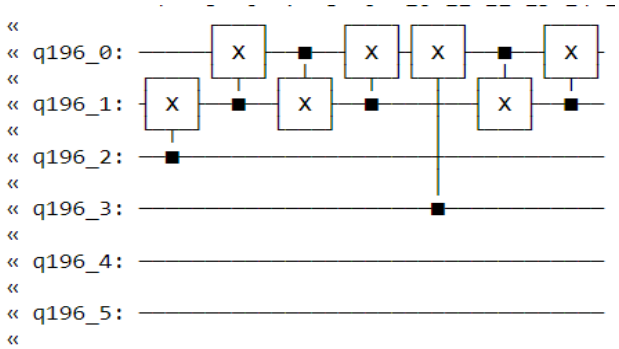
Overall, Qiskit offers a robust and user-friendly environment for developing and executing quantum algorithms. It combines the flexibility of designing quantum circuits, the power of simulation and experimentation on real hardware, and the availability of pre-built algorithms and libraries, making it a valuable tool for researchers, developers, and educators in the field of quantum computing.

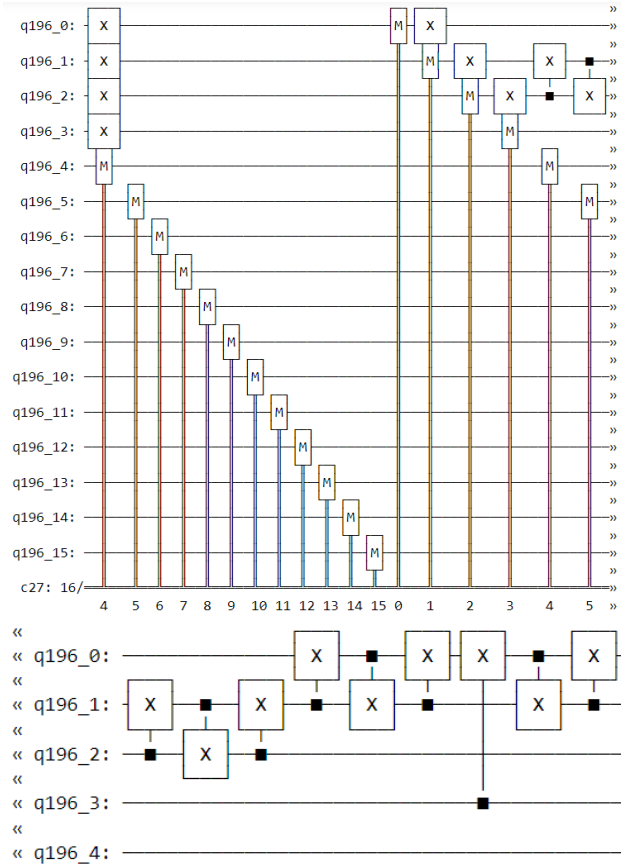
Quantum Circuit Design and results:

The classical computing field incorporates various versions of the Fourier transform, which finds application in signal processing, data compression, and complexity theory. On the other hand, the quantum Fourier transform represents the quantum realization of the discrete Fourier transform, operating on the amplitudes of a wavefunction. In the context of prime factorization, the QFT is employed to determine the period of a function.

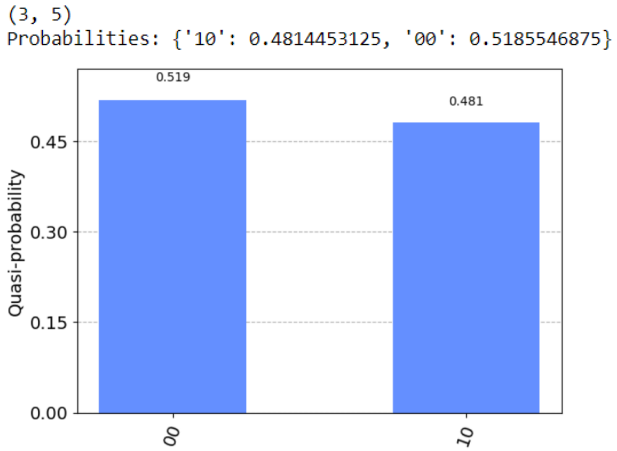


Specifically, the function f(x) = a^x mod N is chosen, where N is the number to be factorized and 'a' is a random integer between 1 and N-1. The period of this function carries information about the factors of N and is obtained by amplifying the most likely period values through the QFT, leading to more precise measurement.





During the measurement stage, which concludes Shor's algorithm, probability calculations play a crucial role in obtaining the final result. Multiple runs of the algorithm are typically required, with the probability of correctly identifying the factors increasing with each iteration. The likelihood of success relies on several factors, including the size of the number being factorized and the quality of the quantum hardware utilized.



When dealing with a large number N, the likelihood of successfully factorizing it using Shor's algorithm generally decreases exponentially as the number of qubits employed increases. However, as the number of qubits grows, the probability of achieving a successful factorization notably improves.

Conclusion:

In conclusion, this project has explored the implementation and implications of Shor's algorithm for prime factorization on large numbers. The results have demonstrated the remarkable potential of Shor's algorithm in efficiently finding prime factors compared to classical factorization methods. This breakthrough algorithm has the power to revolutionize the field of cryptography and significantly impact various industries and scientific fields that rely on prime factorization.
However, practical implementation of Shor's algorithm for large-scale factorization faces current limitations due to the constraints of current quantum hardware and the need for error correction. Quantum computers are still in their early stages of development, and their computational capacity is limited. The size of numbers that can be factorized using Shor's algorithm is currently modest, necessitating further advancements in quantum computing technologies.

Additionally, quantum systems are highly susceptible to environmental disturbances and noise, leading to errors in calculations. To reliably apply Shor's algorithm to large number factorization, the development of robust quantum error correction codes is crucial. Ongoing research in this area holds promise for overcoming this obstacle and ensuring the accuracy and integrity of quantum computations.

Future Advancements:

he future of prime factorization using Shor's algorithm appears promising. Continued advancements in quantum computing technologies are expected to increase the computational power, scalability, and efficiency of quantum hardware. Moreover, the development of error correction techniques specific to quantum systems will contribute to more reliable and accurate implementations of Shor's algorithm. With these advancements, it is conceivable that Shor's algorithm will enable efficient large number factorization, unlocking new possibilities in cryptography, number theory, and other areas. The profound impact of Shor's algorithm on various industries, such as finance, data security, and scientific research, cannot be understated. As researchers and technologists strive towards overcoming the current limitations, we anticipate a future where prime factorization on large numbers using Shor's algorithm becomes a reality, transforming the way we approach complex computations and encryption.

References:

1. https://www.geeksforgeeks.org/analysis-different-methods-find-prime-number-python/
2. https://kaustubhrakhade.medium.com/shors-factoring-algorithm-94a0796a13b1
3. https://www.nature.com/articles/s41598-021-95973-w
4. https://www.linkedin.com/pulse/factor-large-number-using-peter-shors-algorithms-bipin-kumar-sinha
5. https://github.com/qiskit-community/qiskit-community-tutorials
6. https://qbook.qbraid.com/blog/file=(6346caaa7c251a69a7839467)
7. https://www.quantiki.org/wiki/shors-factoring-algorithm