Near term implementation of Shor's Algorithm using Qiskit

Casimer DeCusatis (Fellow, IEEE) and Emily Mcgettrick
Marist College
Poughkeepsie, NY USA
casimer.decusatis@marist.edu

2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC) | 978-1-6654-1490-6/21/$31.00 ©2021 IEEE | DOI: 10.1109/CCWC51732.2021.9376169

***Abstract*—Although the fundamental principles of quantum computing have been known for decades, it is only within the past few years that practical quantum computers have become available. Since these systems are limited to a small number of qubits, they cannot demonstrate quantum advantage for many practical problems. Accordingly, there has been an effort to develop near term implementations of algorithms purely for research and education. In this paper, we discuss a near term implementation of Shor's Algorithm using the Qiskit language on an IBM Q System One quantum computer. We extend currently available documentation by providing a full circuit diagram for Shor's Algorithm including gates specific to the implementation using Qiskit. We present an implementation capable of factoring small two-digit prime numbers, and discuss the limitations of noise when using real quantum computers vs. simulations.**

Keywords—quantum, Qiskit, Shor, factoring

I. INTRODUCTION

Quantum computing is an emerging field which incorporates fundamentals of both quantum physics and computer science. This approach holds the potential to solve certain exponential execution time problems which are beyond practical limits of current digital computers. While the theoretical basis of this field has been understood for many decades [1-5], only within the past few years have working quantum computers become available.

One of the largest near-term quantum computers is the IBM Q System One [6, 7], which became available in 2019 and is programmed using the Qiskit language. This system is available through a cloud interface called the IBM Quantum Experience [6, 7]. The cloud interface provides access to several Q System machines worldwide, allowing workloads to be distributed. Currently the largest publicly available system is 16 qubits, available through an IBM "open account". Systems with more qubits are currently accessible to selected researchers as part of the IBM Quantum Network, and may become available through "open accounts" in the future. It should be noted that some visualizations using Circuit Composer are limited to smaller numbers of qubits; for example, probability can only be viewed for up to 9 qubits. Accordingly, these systems are being studied for research and educational purposes, such as developing near-term implementations of algorithms which will have a practical impact once larger quantum computers become available. According to published roadmaps, a 1,000-qubit version of the Q System is anticipated by 2023 [8], which will enable a much wider range of applications.

Qiskit includes a graphic user interface (GUI) called Circuit Composer [6], which provides visualization of quantum circuits. Documentation for this system includes an online open source book [9], which has been used as the basis for quantum computing courses [10]. Various online training resources [11] have also been made available. Much like quantum computers themselves, these resources are in their early stages of deployment and continue to be refined and extended as a more complete understanding of quantum computing is achieved.

When developing education materials to introduce quantum computer programming, it becomes apparent that there is a lack of training problems with moderate levels of difficulty. For students familiar with linear algebra and Python, there are classical quantum computer problems whose solutions are fairly straightforward and well documented, such as the Deutsch-Josza Algorithm, Bernstein-Vazirani Algorithm, and Grover's Algorithm [1-5, 6, 9, 11]. At the other extreme of complexity, there is a class of quantum computing problems which requires a much deeper understand and level of expertise, such as simulating chemistry and physics reactions [12]. Researchers are exploring these problems, but given the current limitations of quantum computing hardware it may be years before truly practical solutions become available. There appears to be a gap between straightforward and highly complex quantum computing problems which has not been widely addressed, for example a class of problems which could be implemented from scratch in a few days or weeks by students with a one or two semester background in the field. This gap of intermediate level problems is one factor that distinguishes quantum computer programming from classical computer science education.

One of the most interesting, practical problems is factoring of large (100 digit or more) prime numbers, which is in the NP-class of problems [1-5, 13]. This problem cannot be solved on classical digital computers in less than exponential time; the most efficient known classical algorithms can factor a number N in $2^{O(\text{cube root } N)}$ time [5, 10]. The inability to

978-0-7381-4394-1/21/$31.00 ©2021 IEEE

quickly factor large prime numbers forms the basis for most widely used encryption algorithms, including RSA. For example, factoring a 200-digit prime number would take over 1,000 years using commercially available digital computers; the performance is not substantially improved using dedicated supercomputers [5]. The ability to securely encrypt sensitive financial data using RSA enables trillions of dollars' worth of online transactions each year. It is well known that quantum computers can reduce the prime factorization problem from exponential execution time to quadratic execution time. Thus, a large enough quantum computer can potentially break traditional encryption algorithms, including breaking forward security for the massive volumes of data currently stored with RSA encryption.

Shor's Algorithm for prime number factorization was discovered in 1994 [13], and was considered to be of academic rather than practical interest until the advent of practical quantum computers. Recently, this algorithm has been widely studied, and implemented on several quantum computing platforms. Crucially, a complete Qiskit implementation of the full algorithm, including a Circuit Composer diagram and results from near-term quantum computer tests, does not seem to be available. The seminal reference in this area is the Qiskit open source book [9] which includes a discussion of the theory behind this algorithm and sample circuits for phase estimation but does not present a full implementation. Rather, correct operation of the algorithm is inferred by creating a quantum state which is a superposition of a single digit prime number, then showing how part of the algorithm collapses this state into the expected result. Other open source resources discuss recursive implementations, but without detailed descriptions of Qiskit circuits, or discuss hard coded versions of the algorithm using simulators, not real quantum computers [11]. Thus, there is a lack of near-term implementations demonstrating Shor's Algorithm in Qiskit running on an actual quantum computer. Such an approach would support undergraduate education and research efforts by providing a much-needed problem of intermediate difficulty in quantum computer programming.

In this paper, we present (for the first time to our knowledge) a circuit diagram implementation of Shor's Algorithm using Qiskit custom gates, along with both simulated and real quantum computing results. Due to the limitations of the Q System One, this algorithm can currently only factor small two-digit prime numbers. The implementation is scalable to larger prime numbers as the number of qubits increases. We present experimental results from this algorithm using both a simulation and a real quantum computer, and discuss the impact of noise on the quantum computer implementation.

The remainder of this paper is organized as follows. Following the introduction, section II provides a brief background and review of the mathematical basis for Shor's Algorithm. Section III includes our implementation, Circuit Composer diagram with instructions on developing custom gates for Qiskit, and results from both simulated and actual quantum computers. Finally, Section IV summarizes our conclusions and suggests directions for future work.

II. Background on Shor's Algorithm

Consider a number N which is the product of two primes. To determine the prime factors, we first take an initial guess at some number, g, which is either a factor of N or which shares a factor with N. The fact that we can use a guess that shares factors with N derives from Euclid's Theorem [4, 14], a 2,000-year-old method from discrete mathematics which allows us to find the greatest common divisor (GCD), and therefore the factors of interest. This makes the problem significantly easier to solve. However, for a reasonably large N it's highly unlikely that our initial guess g will turn out to either be a factor of N or share a factor with N.

It can be shown [13] that for any pair of whole numbers A and B which do not share a factor, multiplying A by itself enough times eventually results in an integer multiple of B plus 1, i.e.

$A^p = mB + 1$ (equation 1)

For some integers p and m. This means that

$g^p = mN + 1$ (equation 2)

Rearranging terms and factoring yields the following:

$g^p - 1 = mN$ (equation 3)

$(g^{p/2} - 1)(g^{p/2} + 1) = mN$ (equation 4)

The two terms on the left side of equation (4) are factors of mN. Of course, we are interested only in factors on N, not m. Further, we're interested only in integer factors of N, so this will only work if p is an even number (if p is odd, then p/2 is a fraction, not a whole number). If we encounter either of these conditions when attempting to solve equation (4), we simply start over with a new value of g. It can be shown [13] that we're 99% likely to find a useful guess within 10 attempts. At this point, solving for p would take exponential time on a conventional computer, but can be done in quadratic time on a quantum computer. To see this, we will state without proof the following theorem for integer values m, m_2 and r [13]:

If $g^x = mN + r$, then $g^{x+p} = m_2 N + r$ (equation 5)

From this, we can see that p repeats with some period r, or in other words g^x, g^{x+p}, g^{x+2p}, g^{x-p} and so on are all separated by some constant value r. We can therefore reduce the problem of factoring N to the problem of finding the period r. This can be achieved using a Quantum Fourier Transform (QFT) [9]. When using the QFT, we input a sequence of values and the output is a superposition of all other numbers

weighted in a particular manner (the weights correspond to the frequency of the input value). If we input a superposition, then the output is also a superposition of all possible states, which add or subtract either constructively or destructively. Quantum physics tells us that if we input a superposition and the resulting output could have come from more than one element in the superposition, then we'll be left with a superposition of just those elements. In our algorithm, if we take an input superposition of all possible exponents (i.e. x, x+p, x-p, x+2p, etc.) then the output contains just those possibilities that would result in the same value of r, spaced apart with a constant period, p (or equivalently with a frequency which is the reciprocal of the period). We now have a quantum superposition of values that repeats with a period p; if we can find the frequency of these repeating values, we can determine p. Expressing this in standard bra-ket notation yields the relationship shown in figure 1.

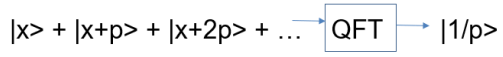

Figure 1 – bra-ket notation for period finding

Finding p means we can compute ($g^{p/2}$ +/- 1), which is an improved guess that shares factors with N. The period finding algorithm is shown schematically in figure 2, for various size input registers of qubits initialized to zero. Note that this is a version of Simon's Algorithm [9] using the Simon oracle Q_f.

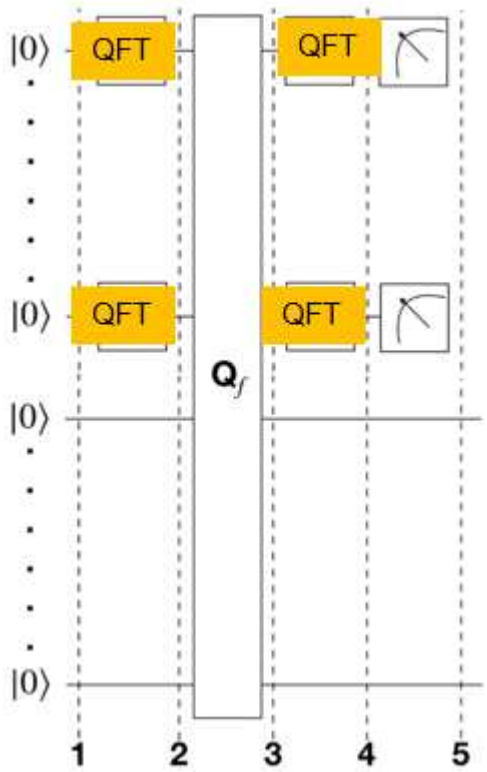

Figure 2 – high level circuit for period finding

III. Experimental Results

The implementation of Shor's Algorithm involves a classical computation to find a reasonable initial guess, g, followed by a quantum computation to find p. The classical computation is a straightforward application of Euclid's Theorem for GCD, and has been discussed elsewhere [9, 13]. We will focus on the quantum computation, which is most easily implemented with a predetermined number of input qubits and a reasonable guess at the output. In Qiskit, the Circuit Composer representation of Shor's Algorithm is shown in figure 3 for the example N = 15, g = 13. To establish that this solution represents an intermediate difficulty problem in near term quantum computer programming, the solution was implemented and tested over the course of several weeks by an undergraduate researcher with one semester's background in the field. In order to represent the value 15, we need a minimum of four qubits, which when measured can collapse into any binary state between zero (0 0 0 0) and fifteen (1 1 1 1). Note that this implies an IBM Q System with 8 qubits could factor prime numbers less than 256, and so on (neglecting qubits used for error correction, which will be discussed shortly). Initially Qiskit forces all qubits to a value of zero, then we apply Hadamard and CNOT gates to put all of the input qubits into a superposition state. Note the recursive nature of the n-qubit QFT implementation; only the last qubit depends on the input value of all the other input qubits, and each further output qubit depends less and less on the input qubits. This is useful in physical implementations of the QFT, where nearest neighbor couplings are easier to achieve on near-term quantum computer hardware.

The QFT is implemented using a single qubit-controlled rotation (CROT) gate for each input, which produces a rotation in state space proportional to the number of qubits in the input register. On an IBM Q System, all single qubit operations are compiled to one of three physical gates designated U1, U2, and U3 before running on the quantum computing hardware [9]. The U1 gate shown in figure 3 is functionally equivalent to an ROT gate, and is given by

$$U1 = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\lambda} \end{pmatrix} \quad \text{(equation 6)}$$

IBM hardware currently realizes this gate as a frame change, which takes zero execution time [9]. Qiskit requires a controlled U1 gate to implement Shor's Algorithm. This gate is not presently accessible directly from the Circuit Composer GUI, and must be generated manually. For example, the gate shown in figure 4 can be realized by the code statement

cu1(pi/2) q[0], q[1]; (equation 7)

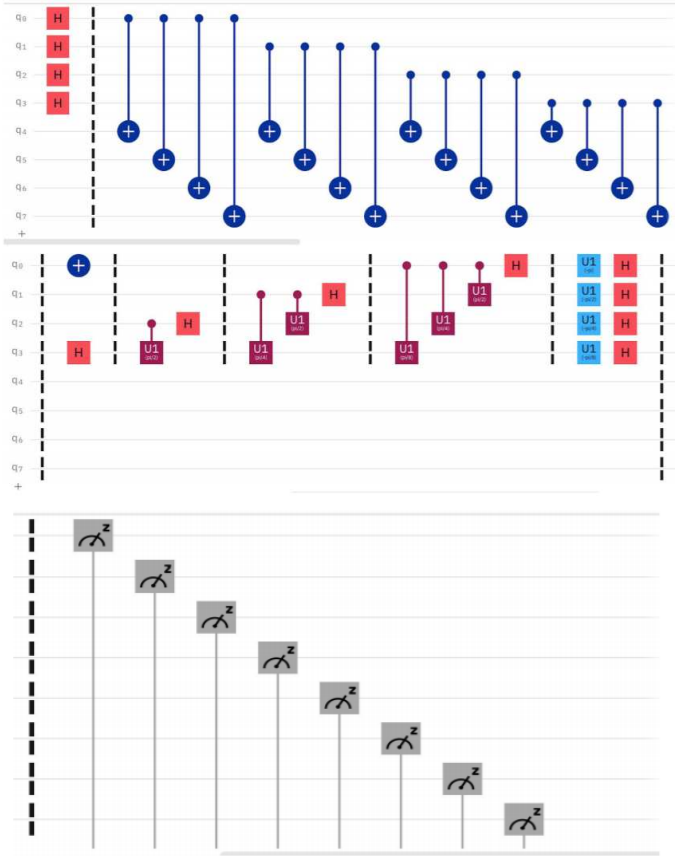

Figure 3 – Circuit Composer realization of Shor's Algorithm

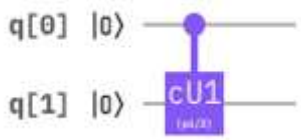

Figure 4 – Custom U-gate realized by equation (7) in Qiskit

The histogram of probability amplitudes resulting from this algorithm running on a simulator in the IBM Quantum Experience with 1028 attempts is shown in figure 5. We can identify a periodicity of 4 in this output.

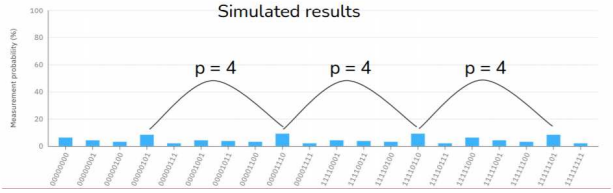

Figure 5 – Simulated period finding algorithm results

Returning to our previous equations, this result yields

$g^{p/2} = 13\ 4/2 \pmod{15} = 4$ (equation 8)

$g^{p/2} +/- 1 = (4+1)\ (4-1) = (5)(3) = 15$ (equation 9)

Thus, we can determine that 5 and 3 are the two prime factors of 15. If we had used a different initial guess, we might have ended up with two values ($g^{p/2}$ +/- 1) which were not themselves factors of 15, but which shared a common factor with 15. In this case, the common factors can be determined with a trivial calculation, yielding the desired factors of 15. For example, if we used the same value of N = 15 but instead an initial guess g = 7, we would still have found p = 4. In this case, $g^{p/2} = 49$, and we would compute ($g^{p/2}$ +1) = 50 as well as ($g^{p/2}$ -1) = 48. It is straightforward to determine that 15 and 50 share a common factor (5), and that 15 and 48 share a common factor (3), thus yielding the correct prime factors 5 and 3 as before.

Note that the simulator probability amplitudes are all on the order of 10% probability or less, meaning that they are highly susceptible to noise. The histogram resulting from running this algorithm on a real quantum computer with 1028 attempts is shown in figure 6. These results were obtained using the IBM Q-16-Melbourne system. We expect a significant increase in noise on the real quantum computer, and in this case, it becomes more difficult to determine the correct value of p. We may be able to mitigate the effects of noise by using additional qubits for error correction, however this isn't feasible on near term quantum computers due to the limited number of available qubits.

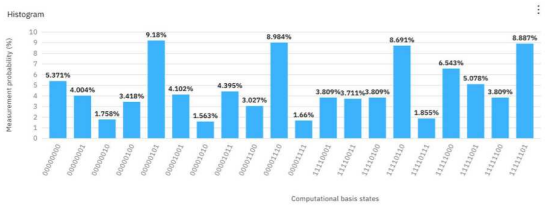

Figure 6 – Real quantum computer result of period finding using Shor's Algorithm

IV. Summary and Conclusions

We have discussed the implementation of Shor's Algorithm using Qiskit on the IBM Q System, and provided a Circuit Composer diagram with a 4-qubit input register for factoring the number 15. This implementation can be adopted by education and research programs as an intermediate difficulty problem in quantum computing, allowing students to investigate prime factorization with minimal background in quantum computer programming. This approach can be scaled to larger prime numbers as the capacity of the quantum computer increases. For example, an 8-qubit system is capable of factoring prime numbers less than 256, and so on. We have noted the limitations of noise in a real quantum computer implementation, which is difficult to interpret correctly even with a 4-qubit input register. Future research will endeavor to scale this algorithm to larger values of N, employing additional qubits for error correction. We have also begun investigating quantum resistant encryption methods, such as the online quantum safe drop box LatLock [15].

ACKNOWLEDGMENT

We gratefully acknowledge the support of Marist College and the New York State Cloud Computing and Analytic Center (CCAC). We also acknowledge the support of Greg Lacey and Fae Gholami of IBM. The terms Q System One and Circuit Composer are registered trademarks of IBM Corporation.

REFERENCES

[1] R.S. Sutor, Dancing with Qubits, Packt Publishing, Birmingham, UK (November 2019)

[2] E.R. Johnson, N. Harrigan, and M. Gimeno-Segovia, Programming Quantum Computers, O'Reilly Media, Boston, MA (2019)

[3] C. Bernhardt, Quantum computing for everyone, MIT Press, Cambridge, MA (2019)

[4] S. Aaronson, Quantum computing since Democritus, Cambridge University Press, Cambridge, UK (2014)

[5] M. Cozzens and S.J. Miller, The mathematics of encryption, American Mathematical Society, New York ,NY (2013)

[6] Press Release, "IBM unveils world's first integrated quantum computing system for commercial use", Jan. 8, 2019, https://newsroom.ibm.com/2019-01-08-IBM-Unveils-Worlds-First-Integrated-Quantum-Computing-System-for-Commercial-Use (last accessed December 6, 2020)

[7] "IBM Quantum Update: Q System One Launch, New Collaborators, and QC Center Plans". *HPCwire*. 10 January 2019

[8] J. Hruska, "IBM unveils quantum roadmap, plans 1,000 qubit chip by 2023", Extreme Tech, Sept. 15, 2020 https://www.extremetech.com/computing/315020-ibm-unveils-quantum-roadmap-plans-1000-qubit-chip-by-2023 (last accessed December 5, 2020)

[9] A. Asfaw et.al., Learning quantum computing using Qiskit, online open source book, 2020, http://community.qiskit.org/textbook (last accessed December 6, 2020)

[10] C. DeCusatis, "Quantum Computer Algorithms and Programming", Proc. NSF Enterprise Computing Conference (ECC), Marist College, Poughkeepsie, NY (June 7-9, 2020) (virtual conference)

[11] IBM online course, Introduction to Quantum Computing and Quantum Hardware, https://qiskit.org/learn/intro-qc-qh/ (last accessed December 6, 2020); see also https://www.youtube.com/watch?v=EdJ7RoWcU48 (last accessed December 6, 2020)

[12] A.D. Corcoles et.al., "Challenges and opportunites of near term quantum computing systems", Oct. 2019, https://arxiv.org/abs/1910.02894 (last accessed December 5, 2020)

[13] IBM Quantum Experience, Shor's Algorithm, https://quantum-computing.ibm.com/docs/iqx/guide/shors-algorithm (last accessed 12/3/2020); see also https://pennylane.ai/ (last accessed 12/3/2020)

[14] P. Shor, "Polynomial time algorithms for prime factorization and discrete logarithms on a quantum computer", SIAM J. Sci. Statist. Comput. Vol 26, p. 1484 (1995), https://arxiv.org/abs/quant-ph/9508027 (last accessed December 5, 2020)

[15] D. Yost and C. DeCusatis, "Cybersecurity in the quantum era", Proc. NSF Enterprise Computing Conference (ECC), Marist College, Poughkeepsie, NY (June 7-9, 2020) (virtual conference)