An Experimental Study of Shor’s Factoring Algorithm on IBM Q

Mirko Amico,1 Zain H. Saleem,2 and Muir Kumph3
1The Graduate School and University Center, The City University of New York, New York, NY 10016, USA
2Theoretical Research Institute of Pakistan Academy of Sciences, Islamabad 44000, Pakistan
3IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, USA

We study the results of a compiled version of Shor’s factoring algorithm on the ibmqx5 superconducting chip, for the particular case of $N = 15, 21$ and $35$. The semi-classical quantum Fourier transform is used to implement the algorithm with only a small number of physical qubits and the circuits are designed to reduce the number of gates to the minimum. We give a quantitative measure of the similarity between the experimentally obtained distribution and the predicted theoretical distribution of phases corresponding to the periods, through the square of the statistical overlap (SSO). This allows us to assign a period to the experimental data without the use of the continued fraction algorithm. Also, a quantitative estimate of the error in our assignment of the period is given by using the overlapping coefficient (OLV).

I. INTRODUCTION

Shor’s factoring algorithm [1] is a well known example of a quantum algorithm outperforming the best known classical algorithm. Experimental implementation of the algorithm with physical qubits however remains a challenge because of the errors introduced by the large number of qubits and gates required to execute the algorithm. In this article we provide a proof-of-principle demonstration of a compiled version of Shor’s factoring algorithm to factor the numbers $N = 15, 21$ and $35$ using five, six and seven superconducting qubits, respectively. Similar experiments have been done on setups like NMR [2], trapped ions [3], photons [4–6], photonic chips [7] and superconducting qubits [8] [9]. However, with the exception of Refs. [4] [5], all these realizations involve an oversimplified version of the algorithm which is equivalent to coin flipping [10] and no quantum hardware is needed to obtain the same results.

The algorithm turns the problem of factoring into the problem of order finding, where a quantum speed-up exists. In fact, finding the prime factors of a number $N$ is equivalent to finding the exponent $x$ for which the function $a^x \mod N = 1$, where $a$ is an integer smaller than $N$ picked at random. Such exponent is called the order, or period, of $a$. Let us briefly review the quantum part of the algorithm before diving into the details of the experiment. Two quantum registers are needed for the computation. One register is used to store the period, called period register, and the other stores the results of the computation, called computational register. The size of both registers depends on the number $N$ to be factored. In particular, the period register should have a number of qubits $n_p$ in the interval $\log_2(N^2) \leq n_p \leq \log_2(2N^2)$ and the computational register should be large enough to be able to represent the number $N - 1$, resulting from the modular exponentiation function (MEF) $a^x \mod N$, thus requiring $n_q = \log_2 N$ qubits.

At the beginning of the quantum algorithm, the two registers are initialized to the state $|00...0\rangle_p |00...0\rangle_q$, where the subscripts $p$ and $q$ denote the period register and the computational register, respectively. The period register stores all the possible values of the exponent $x$, which will give an estimate of the period, by creating a uniform superposition of all possible bit-strings through Hadamard gates on all qubits $\frac{1}{\sqrt{Q}} \sum_{x=0}^{Q-1} |x\rangle_p$, where $Q = 2^n$. While the computational register stores the results of the MEF, $a^x \mod N$. After the first step, the two registers are in the state $\frac{1}{\sqrt{Q}} \sum_{x=0}^{Q-1} |x\rangle_p |a^x \mod N\rangle_q$. Then, the quantum Fourier transform (QFT) is applied to the period register so that $|x\rangle_p \rightarrow \frac{1}{\sqrt{Q}} \sum_{s=0}^{Q-1} e^{2\pi Q i x s} |s\rangle_p$. As a result of the QFT, interference between all the possible states occurs. If the period register is then measured, a value of $s$ such that $\frac{x}{Q} = \frac{s}{d}$, where $d$ is an integer and $r$ is the period, will be observed. More specifically, the measurement of the period register allows us to find an approximation to the fraction $\frac{x}{Q}$ proportional to $\frac{d}{Q}$ with high probability. The final part of the algorithm involves classical processing of the measurement obtained in the quantum part. The value of the period $r$ can be found from the fraction $\frac{x}{Q}$ by using the continued fraction algorithm. If the period $r$ calculated in this way is odd or $r = 0$, the algorithm fails and one restarts by picking a different base $a$. If $r$ is even, $(a^r - 1) \mod N$ can be factored into $(a^{r/2} - 1)(a^{r/2} + 1) \mod N$. The final step is to check if $(a^{r/2} + 1) \mod N$ has a common divisor with $N$ by checking that $\gcd(a^{r/2} + 1, N) \neq 1$. If that’s true, then the two factors of $N$ are $\gcd(a^{r/2} + 1, N)$ and $\gcd(a^{r/2} - 1, N)$.

As mentioned earlier, the execution of this version of the algorithm requires $n_q = \log_2(N) \mod N$ qubits in the computational register to perform the modular exponentiation and at least another $n_p = 2\log_2(N) \mod N$ qubits in the period register to perform the QFT. Thus the complete algorithm requires a total number of $3\log_2(N) \mod N$ qubits.

Even the factoring of a number as small as $N = 15$ needs 12 qubits in the input register to execute this algorithm, which is still a challenge for today’s physical realizations of quantum computers. However, Kitaev [11] observed that for the purpose of algorithms like Shor’s, where one doesn’t need the information on the relative phase of the
output states but only their measured probability amplitudes, one can replace the fully coherent quantum Fourier transform with the semi-classical quantum Fourier transform (sc-QFT). In the sc-QFT, one of the qubits of the period register is measured each time. The result of the measurement of the qubit is then used to determine the type of measurement on the next one. This enables the replacement of the $2 \log_2 (N)$ qubits of the period register with a single control qubit measured multiple times. For the case of factoring $N = 15$, Kitaev’s approach reduces the total number of qubits required to $n = 5$ and for the case of $N = 21$ and $N = 35$ to $n = 6$ and $n = 7$, respectively, which are small enough numbers for the presently available hardware to handle. This decrease in the system size, however, comes with the drawback of requiring in-sequence single-qubit readout and state re-initialization together with feed-forward of gate settings based on previous measurement results. The implementation of the sc-QFT has been described in [12] [13] and realized in [14]. At present the IBM quantum computer doesn’t perform in-sequence single qubit read out and qubit re-initialization. In this paper we provide a procedure for going around this hurdle to implement the sc-QFT.

The article is organized in the following way. Sec. II describes the hardware used for the experiment. In Sec. III the implementation of the factoring experiment for $N = 15, 21$ and $35$, respectively, is described. The results obtained from running the algorithm on the ibmqx5 quantum processor are analyzed and discussed in Sec. IV. Conclusions follow in Sec. V.

II. HARDWARE

We use IBM ibmqx5 chip with sixteen superconducting qubits to implement our experiments for factoring the numbers $N = 15, 21$ and $35$. The qubits are distributed on the plane, as two adjacent arrays of eight qubits each. The coupling map of the qubits is shown in Fig. 1.

![Fig. 1: Coupling map between the qubits of the ibmqx5 device.](image1)

The qubits’ relaxation time $T_1$ ranges from $25 \sim 60$ $\mu$s and their dephasing time $T_2$ from $20 \sim 100$ $\mu$s. The single-qubit gates have a high fidelity, measured at $\sim 99.8\%$ at the time of the experiment. The multi-qubit gate fidelity was measured around $95\% - 98\%$ depending on the pairs of qubits considered. Another source of error comes from the read-out of the states of the qubits, which amounts to roughly an error of $5\%$. Using these parameters, the noise can be incorporated in the simulation, obtaining a more accurate prediction for the output of the device.

III. EXPERIMENT

Following the example given in [3], we implement the quantum part of Shor’s factoring algorithm using the circuit depicted in Fig. 2.

![Fig. 2: Circuit for factoring $N = 15, 21$ and $35$ as implemented in [3].](image2)

As can be seen in the circuit diagram in Fig. 2, rotations of the control qubit depend on the outcome of each of its measurement in the previous steps. Since the ibmqx5 chip does not allow for qubit reset and conditional operation based on measurements, which are required to implement the sc-QFT suggested by Kitaev, we implement the algorithm in three separate parts. The different parts can be joined by feeding into the next section the output quantum states of the computational register of the previous section (for details see Appendix A) and adding the right rotation gates on the period qubit based on the results of previous measurements.

The case $N = 15$ is the simplest possible case and it does not provide an example where quantum interference between the states of the computational register brings an advantage to the computation. For this reason, we attempt to factor the second smallest number which is a product of two primes, $N = 21$. In this case, there are bases $a$ for which the period is not a multiple of two, thus constructive quantum interference between states.
in the computational register is needed to increase the likelihood of finding the correct result. An example of such case was first demonstrated in Ref. 5.

We implement an algorithm for factoring \( N = 21 \) with base \( a = 2 \) using three bits of precision for the estimation of the phase which encodes the period. In this case the quantum register is composed of five qubits in the computational register and one qubit in the period register. We adopt the same methodology used previously, breaking each stage of the modular exponentiation and manually feeding the output of each section as input to the next (details in Appendix A). This means that the circuit will have three stages of modular exponentiation, where a single bit of the phase which encodes the period is estimated at each stage. Therefore, the circuit looks like the one in Fig. 2. The modular exponentiation circuit are specifically designed to calculate \( a^x \mod 21 \), where we choose the base \( a = 2 \). This basis has periods \( r = 6 \), thus \( 1/r \) cannot easily be represented in binary. Therefore, the accuracy of the estimation of the period depends on the number of bits used for the phase estimation.

The same method is applied to factor \( N = 35 \) with base \( a = 4 \). In this case we need six qubit in the computational register and one qubit in the period register. As in the case of \( N = 21 \), the period of \( 4^x \mod 35 \) is \( r = 6 \), therefore \( 1/r \) cannot be easily represented in binary. As a result of running the quantum algorithm we obtain a probability distribution for the estimated phase \( s \) which is peaked around the multiples of \( 1/r \). We use a three bits register for the estimation of the phase which encodes the period. Again, the circuit for running the algorithm is realized as shown in Fig. 2, each stage estimating one bit of the phase is implemented separately and then joined through a classical algorithm. The individual circuits which compute the MEF at the different stages can be found in Appendix A.

IV. RESULTS AND DATA ANALYSIS

Figs. 3a, 4a, 5a and 6a show the results obtained running the factoring algorithm on the ibmqx5 superconducting device. Depicted are the experimental relative probabilities found (in blue) side by side with the expectation values which can be computed theoretically (in green) for each value of the estimated phase \( s \) for the bases \( a \) used. The algorithm was run 1000 times for each base.

The success of the experiment is evaluated in two different ways. We use the probability plots to give a qualitative estimation of the correctness of the results, while the square of the statistical overlap (SSO) is used as a quantitative measure. Probability plots \cite{14} are a useful tool to visually compare two distributions. In a probability plot, one distribution is plotted against the other. If the two distributions are the same, the plot will show a straight line \((y = x)\). The amount of deviation from the straight \( y = x \) line in a probability plot is an indication of the difference between the two probability distributions plotted. The probability plots between the experimental distribution and the expected theoretical one for each case are shown in Figs. 3a, 4a, 5a and 6a. In the case of \( N = 15 \), the data in Figs. 3a and 4a are on a straight line very close to the \( y = x \) line (tagged as "Ideal" on the

---

**FIG. 3:** (a) Probability of finding a given phase for \( N = 15 \) with base \( a = 2 \), and (b) probability plot of the theoretical distribution and the experimental distribution for \( r = 4 \). The experimental distribution is depicted through the collection of data and the fit of the data. (c) SSO of the experimental data with the theoretical probability distribution corresponding to all possible values of the period \( r \).
FIG. 4: (a) Probability of finding a given phase for $N = 15$ with base $a = 11$, and (b) probability plot of the theoretical distribution and the experimental distribution. The experimental distribution is depicted through the collection of data and the fit of the data. (c) SSO of the experimental data with the theoretical probability distributions corresponding to all possible periods.

plots). For the $N = 21$ case, the data lie on a straight line parallel to the $y = x$ ideal line as can be seen from Fig. 5a. This means that there is an offset in our experimental distribution but the overall shape coincides with the theoretical one. Finally, for $N = 35$, the data in Fig. 6c lie on a straight line which is very far from the $y = x$ line, indicating an important deviation of the experimental results from the theoretically expected ones. In the probability plots we have added a fit of the data with error bars. It can be seen that for $N = 15$ and $N = 21$ the fit is close to the ideal line (within the error bars) but for $N = 35$ it is not. We believe that the probability plots provide a good qualitative measure of
is the likelihood that this data comes from a given probability distribution? The answer to this question will reveal two aspects of our experiment. First, it will allow us to assign a period to our result without the need for the continued fraction algorithm. Second, it will give us a measure of the error we make in the assignment.

To assign a period to the data, we compare the probability distribution obtained experimentally with the expected probability distribution corresponding to the possible values of the period compatible with a 3-bit precision estimate (thus \( r \) ranges from 2 to 7). This can be done because there is a unique theoretical probability distribution corresponding to each period, independently of the basis \( a \) used for in the experiment. The period of the theoretical distribution which is most similar to the experimental data is then assigned to the experiment. Following [3], we use the square of the statistical overlap (SSO) introduced in [13] as a measure of similarity between probability distributions. The SSO is defined as

\[
SSO = \left( \sum_j m_j^{1/2} e_j^{1/2} \right)^2, \tag{1}
\]

where \( m_j \) and \( e_j \) are the measured and expected output-state probabilities of state \( |j\rangle \), respectively.

One can calculate the error on the SSO from the Poissonian counting error of the data, assuming Gaussian propagation of errors

\[
\Delta SSO = \sqrt{\sum_j \frac{\partial}{\partial m_j} (m_j^{1/2} e_j^{1/2})^2 \Delta m_j^2}. \tag{2}
\]

This is used to determine the error in the assignment of the period in the following way. First we plot unit area normalized distributions with SSO as the mean and \( \Delta SSO \) as the variance. Then, the error made when assigning a certain period to the experimental data is found by calculating the area of overlap between the Gaussian distribution with the highest SSO and the second closest one. This is done through the overlapping coefficient (OVL) [10] between the normal distributions. The OVL is defined as

\[
\text{OVL} = \sum_x \min(f(x_1), f(x_2)), \tag{3}
\]

where \( f(x_1) \) is the normal distribution with the highest SSO and \( f(x_2) \) is the normal distribution with second highest SSO. The OVL tells us what is the probability that the assignment is done incorrectly i.e. the highest SSO for our experimental data comes from a different theoretical probability distribution than the assigned one. Thus, we quantify the error on our assignment as \( \epsilon_{ij} = \text{OVL}_{ij} \) where \( i \) is for the assigned period and \( j \) for the period of the distribution with the second highest
SSO.

Figs. 3c and 3d show the SSO of the experimental distributions and their deviations obtained for \( N = 15 \), \( a = 2 \) and \( a = 11 \), respectively. For \( a = 2 \), the highest SSO is 0.97 for the theoretical distribution corresponding to the period \( r = 4 \). Therefore we assign the period \( r = 4 \) to the experimental distribution obtained. The error we make in assigning the period \( r = 4 \) instead of period \( r = 7 \), which is the closest match, is \( \epsilon_{47} = 3.8 \times 10^{-134} \). For \( a = 11 \), the highest SSO is 0.92 which correspond to \( r = 2 \). The error in the assignment of \( r = 2 \) with respect to \( r = 4 \), which has the second highest SSO, is \( \epsilon_{24} = 4.1 \times 10^{-31} \). The results obtained for \( N = 21 \) and \( a = 2 \) are shown in Fig. 5c. Here, it is more difficult to determine the period with certainty. The highest SSO is 0.78, which corresponds to the theoretical distribution with \( r = 6 \). The error in assigning \( r = 6 \) to the experimental data is \( \epsilon_{67} = 1.2 \times 10^{-3} \). Therefore, there is a ~0.1% chance that we assigned the period incorrectly and the true period was \( r = 7 \) instead. For the case of \( N = 35 \) and \( a = 4 \), the results presented in Fig. 6c show that the highest SSO between the experimental data and the theoretical distribution corresponding to all possible periods is 0.99 for \( r = 7 \), although this is not the expected period. There is another close match with an SSO of 0.98 for \( r = 6 \), which is the correct one. The error in assigning period \( r = 7 \) to the experimental data instead of \( r = 6 \) is \( \epsilon_{76} = 0.14 \). Thus, in this case it is quite difficult to discern the correct period.

V. CONCLUSIONS

Although the results are obtained with a compiled and simplified version of Shor’s factoring algorithm, our purpose is to show a way to proceed with the implementation of generic algorithms on the approximate quantum computers available now. In practice, the non-negligible noise and the lack of key functions of the device force us to rethink how to design algorithms that can work on these machines. As it is evident from this work, one needs to supplement the deficiencies of the hardware with a more detailed theoretical analysis and classical processing. By doing so, one can reduce the length of the circuit needed to implement the algorithm, mitigating the effects of noise, and overcoming the lack of particular functions assumed for a general-purpose quantum computer. We emphasize that the simplification by inspection done here was possible only due to the small size of the circuit. Larger circuits would require a more sophisticated optimization. We introduce different ways to evaluate the success of the experiment. Probability plots give a qualitative measure of this similarity between the distribution of the experimental data and the expected theoretical one. We adopt a new way to assign a certain period to the probability distribution obtained from the experimental data by means of the SSO. To correctly quantify the error which can be made in this assignment, we calculate the overlapping coefficient between different candidates for the period. Overall, the experimental results obtained from running the algorithm on the ibmqx5 device are in agreement with the theoretical expectation values. Excellent agreement is found for \( N = 15 \), while deviations from the theoretical results become more noticeable for larger \( N \). This is due to the cumulative errors coming from the increasing number of two-qubits gates necessary to implement the more complex MEF.

Acknowledgments

We acknowledge use of the IBM Q for this work. The views expressed are those of the authors and do not reflect the official policy or position of IBM or the IBM Q team. The authors are grateful to N. T. Bronn and R. Ya. Kezerashvili for the valuable and stimulating discussions.

VI. APPENDIX A: CIRCUITS FOR THE MEF

Here we present the procedure used to implement the MEF in the experiments for factoring \( N = 15, 21 \) and 35. The circuits used for the MEF in the experiment for factoring \( N = 15 \) are shown in Fig. 7.

![Circuits for the MEF](image-url)
one of the elements of the set \(\{4, 11, 14\}\), the modular exponentiation function is again the identity in Fig. 7a and this step turns again into a deterministic step. If the base is one of the elements of the set \(\{2, 7, 8, 13\}\), the MEF has the same simple circuit for any of these \(a\), which can be seen in Fig. 7b. For the MEF in Part 3, examples of the circuits for two specific bases are given in Figs. 7c, 7d. Optimized circuits for each base have been designed to reduce the number of gates to the minimum.

![Diagram](image)

**FIG. 8: Modular exponentiation circuits for \(N = 21\) and base \(a = 2\).** (a) \(2^4 \mod 21\), (b) \(2^2 \mod 21\), (c) \(2^5 \mod 21\) for the following results of previous measurements of the period register: \(\text{bit}^{(0)} = 0\) and \(\text{bit}^{(1)} = 0\); (d) \(2^2 \mod 21\) for \(\text{bit}^{(0)} = 1\) and \(\text{bit}^{(1)} = 0\); (e) \(2^4 \mod 21\) for \(\text{bit}^{(0)} = 0\) and \(\text{bit}^{(1)} = 1\); and (f) \(2^5 \mod 21\) for \(\text{bit}^{(0)} = 1\) and \(\text{bit}^{(1)} = 1\).

The circuits of the MEF used in the experiment of factoring \(N = 21\) are presented in Fig. 8. The experiment was conducted only with the base \(a = 2\), therefore all circuits have been developed only for this base. The MEF for Part 1 is shown in Fig. 8a. For Part 2, the circuit in Fig. 8b was used. In Part 3, depending on the bits measured in the period register in the previous stages, different states are prepared as input. For this reason, different modular exponentiation circuits were designed according to the results of the measurements of the period register. The various possibilities are shown in Figs. 8c, 8d, 8e and 8f corresponding to the four possible outcomes 00, 01, 10 and 11, respectively.

![Diagram](image)

**FIG. 9: Modular exponentiation circuits for \(N = 35\).** (a) \(4^4 \mod 35\), (b) \(4^2 \mod 35\), (c) \(4^4 \mod 35\).

The MEFs implemented in the experiment of factoring \(N = 35\) are depicted in Fig. 9. The circuits are designed for the algorithm where the base \(a = 4\). The MEF for Part 1, 2 and 3 are shown in Figs. 9a, 9b and 9c respectively. In this case, one circuit which works for any input was designed for the MEF at each stage.

---

1. P. Shor, in Proc. 35th Annu. Symp. on the Foundations of Computer Science, edited by S. Goldwasser (IEEE Computer Society Press, Los Alamitos, California, 1994), p. 124-134.
2. L. M. K. Vandersypen, M. Steffen, G. Breyla, C. S. Yannoni, M. H. Sherwood, and I. L. Chuang, Nature 414, 883 (2001).
3. T. Monz, D. Nigg, E. A. Martinez, M F. Brandl, P Schindler, R Rines, S X. Wang, I L. Chuang, R Blatt, Science Vol 351, 1068-1070 (2016).
4. C. Y. Lu, D. E. Browne, T. Yang, and J. W. Pan, Physical Review Letters 99, 250504 (2007).
5. E. Martin-Lopez, A. Laing, T. Lawson, R. Alvarez, X.-Q. Zhou, and J. L. O’Brien, Nature photonics 6, 11 (2012).
6. B.P. Lanyon, T.J. Weinhold, N.K. Langford, M. Barbieri, D.F.V. James, A. Gilchrist, A.G. White, Physical Review Letters 99, 250505 (2007).
7. A. Politi, J. C. F. Matthews, and J. L. O’ Brien, Science 325, 1221 (2009).
8. E. Lucero, R. Barends, Y. Chen, J. Kelly, M. Mariantoni, A. Megrant, P. O’Malley, D. Sank, A. Vainsencher, J. Wenner, T. White, Y. Yin, A. N. Cleland and John M.
Martinis, Nature Physics volume 8, 719-723 (2012).
[9] P. J. Coles et al., arXiv:1804.03719 2018 (to be published).
[10] J. A. Smolin, G. Smith, and A. Vargo, Nature (London) 499, 163-165 (2013).
[11] A. Y. Kitaev, quant-ph/9511026 1995 (to be published).
[12] R. B. Griffiths and C. S. Niu, Physical Review Letters 76, 3228 (1996).
[13] J. Chiaverini, J. Britton, D. Leibfried, E. Knill, M. D. Barrett, R. B. Blakestad, W. M. Itano, J. D. Jost, C. Langer, R. Ozeri, et al., Science 308, 997 (2005).
[14] B. Kleiner, J. M. Chambers, W. S. Cleveland, and P. A. Tukey, Wadsworth International Group, CA and Duxbury Press (1983).
[15] K. Pearson, Proceedings of the Royal Society of London 58, 240-242 (1895).
[16] H. F. Inman and E. L. Bradley Jr , Communications in Statistics - Theory and Methods, 18 (10), 3851-3874 (1989).