SUMMER PROJECT REPORT

Analysis of Quantum Algorithms

Author
Arnav METRANI

Supervisor
Dr. Kuntal ROY

July 25,2023

QUANTUM ALGORITHMS

ARNAV METRANI

Abstract. This report presents quantum algorithms that have a lower time complexity than their classical counterparts, such as the Bernstein-Vazirani algorithm and Shor's algorithm. This also covers other significant algorithms (such as Grover's search), as well as a general overview of Quantum Error Corrections. The report is largely self-contained and requires basic knowledge about quantum computing and linear algebra.

1. Historical survey

It is impossible to pinpoint a singular event that prompted the birth of classical computation, as there is no strict definition for classical computing. As a result, the possible pioneers range from the ancient Greeks[1], to Charles Babbage and Alan Turing.
Nevertheless, the works of Turing have played a vital part in the development of classical computing, which in turn has brought about once-unimaginable changes that we now take for granted (ranging from telecommunication to space travel).

Despite the depth of analysis in classical computing and the vast complexity of this field, there are still a large set of problems for whom no "efficient" classical algorithm has been developed. The Traveling Salesman problem and integer factorization are a few examples.

To overcome this hurdle, certain researchers have attempted to utilise the properties of quantum mechanics to develop more efficient algorithms, giving birth to the field of quantum computing and quantum algorithms.
Compared to classical computing, the history of Quantum Computing is easier to trace out. In the early 1980′s, Paul Beinoff published a set of articles[2, 3] outlining the construction of a Quantum Turing Machine, while Feynman[4] and Yuri Manin speculated on the development of hardware that utilised quantum-mechanical phenomena for speedups. David Deutsch provided further refinements through the extended Church-Turing thesis and the development of a Universal Quantum computer. [5] This was followed by the development of quantum algorithms to solve oracle problems, such as the Deutsch-Jozsa algorithm[6], Bernstein-Vazirani algorithm[7] and Simon's algorithm[8].
David Coppersmith then developed an algorithm[9] that can efficiently perform Discrete Fourier Transform on quantum states. Peter Shor utilised Simon's results and Coppersmith's QFT to devise quantum algorithms[10] for the Discrete Log Problem and for integer factorization, both of which provided exponential speedups to their classical counterparts. Shor's result was a massive breakthrough as it provided a method to crack RSA and the Diffie–Hellman cryptosystems; cryptosystems that are still widely used today.
Lov Grover shortly after devised an algorithm[11] to solve the unstructured search problem, providing a quadratic speedup to the best possible classical method. These algorithms (and the techniques used by them) are now widely used as subroutines in other quantum algorithms.
Recent developments include the HHL algorithm [12, 13], Quantum walks[14],Variational quantum eigensolver(VQE)[15], and post-quantum cryptography[16].

2. Resources and methods

Quirk and IBM Q-Composer have been used to build the quantum circuits in this report, and all jobs have been run on IBMQ's public quantum computers. All results can be found in the .ipynb file at https://github.com/ArnavMetrani/Quantum-Algorithms-report/blob/main/QCReport.ipynb. All circuits have been run for 1024 shots. The circuits of Quirk and Q-Composer my differ slightly as they follow different convention. For example, Quirk follows the big endian convention and uses a different orientation of axes for its qubits. For more information, contact ms21254@iisermohali.ac.in

3. Algorithms

3.1. **Quantum Teleportation.**

The conditions are as follows:

- A wants to convey the information encoded in qubit $|\psi\rangle$ to B.
- A does not know what the state of the qubit is, and they can only share classical information.

At first this seems like an impossible task, as conveying $|\psi\rangle$ may require an infinite amount of classical bits for encoding.

The problem is mitigated by the following algorithm-

Setup:

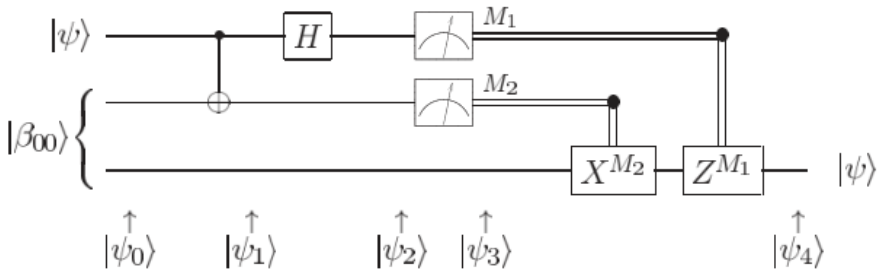


Figure 1. Circuit for quantum teleportation.[17]

- A and B generate a Bell pair $|\beta_{00}\rangle$ and each takes one qubit.
- Since state to be teleported is $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, input state $|\psi_0\rangle = |\psi\rangle|\beta_{00}\rangle$.
- Applying CNOT on the qubits and Hadamard gate on the first qubit:

$$|\psi_2\rangle = \frac{\alpha(|0\rangle + |1\rangle)(|00\rangle + |11\rangle) + \beta(|0\rangle - |1\rangle)(|10\rangle + |01\rangle)}{2}$$

$$|\psi_2\rangle = \frac{1}{2}[\alpha|000\rangle + \alpha|100\rangle + \alpha|011\rangle + \alpha|111\rangle + \beta|010\rangle - \beta|110\rangle + \beta|001\rangle - \beta|101\rangle]$$

Assume that the first two qubits are measured and the outcome is 00. Then state will collapse to $\alpha|000\rangle + \beta|001\rangle$. Due to this, the state of the third qubit after measurement is $\alpha|0\rangle+\beta|1\rangle$. This can be interpreted using the distributive law, $[|00\rangle|1\rangle = |0\rangle|01\rangle]$.

As a consequence of the above result, the terms can be regrouped as:

$$|\psi_2\rangle = \left[\frac{|00\rangle(\alpha|0\rangle + \beta|1\rangle) + |01\rangle(\alpha|1\rangle + \beta|0\rangle) + |10\rangle(\alpha|0\rangle - \beta|1\rangle) + |11\rangle(\alpha|1\rangle - \beta|0\rangle)}{2}\right]$$

Next, A measures its qubits and sends the measurements to B.

$$\begin{aligned}
00 &\longmapsto |\psi_3(00)\rangle \equiv \left[\alpha|0\rangle + \beta|1\rangle\right] \\
01 &\longmapsto |\psi_3(01)\rangle \equiv \left[\alpha|1\rangle + \beta|0\rangle\right] \\
10 &\longmapsto |\psi_3(10)\rangle \equiv \left[\alpha|0\rangle - \beta|1\rangle\right] \\
11 &\longmapsto |\psi_3(11)\rangle \equiv \left[\alpha|1\rangle - \beta|0\rangle\right].
\end{aligned}$$

FIGURE 2. Measurement interpretation for quantum teleportation.[17]

Depending on measurements received, B applies the necessary gates to recover $|\psi\rangle$. However, this method does not allow for faster than light communication as the measurements taken by A need to be transferred to B through a different method.

3.2. **Superdense Coding.**

This algorithm allows for the transfer of two classical bits of information, through the transfer of a single qubit. The procedure is as follows:

A Bell pair is generated (here, we take $|\beta_{00}\rangle$), and A and B are given control of one qubit each (of the pair). If A wants to communicate:

- 00: Apply no gate and send to B. B now has $\frac{|00\rangle+|11\rangle}{\sqrt{2}}$.
- 01: Apply Z gate and send to B. B now has $\frac{|00\rangle-|11\rangle}{\sqrt{2}}$.
- 10: Apply X gate and send to B. B now has $\frac{|10\rangle+|01\rangle}{\sqrt{2}}$.
- 11: Apply X and Z gate and send to B. B now has $\frac{|01\rangle-|10\rangle}{\sqrt{2}}$.

B applies the CNOT gate (with B's original qubit as target) and then applying H gate on the received qubit. On measurement, B will measure $|00\rangle$, $|01\rangle$, $|10\rangle$ or $|11\rangle$ corresponding to their classical values as intended by A.

Superdense coding is a secure method of quantum communication.
If C intercepts this qubit, it will not be able to determine the information being transmitted without access to B's qubit. However the attacker can still scramble the information by applying an operation on the intercepted qubit. *Here we assume that the attacker C does not have control of the Bell state qubit with B (which would allow C to completely replace B in the procedure) and thus can only intercept the qubit that A sends to B.*
Ultradense coding ($2+\varepsilon$: 1 encoding ratio) does not seem to be a practical possibility.[18]

3.3. **Deutsch Algorithm.**

Problem: Given a function f: {0,1} → {0,1}, how many queries are required to determine if the function is constant or balanced? Here, we interpret the function to be a blackbox, unlike $f(x) = x^2$.

In classical computing, we will require 2 queries to determine. This quantum algorithm requires only 1 query to the function.

For this algorithm, we define a gate U_f : |xy⟩ → |x, y ⨁ f(x)⟩.
($|x, 0\rangle \rightarrow |x, f(x)\rangle$ *cannot be defined as a gate since it is not a unitary operation.*)
The algorithm works by effectively testing all possible inputs simultaneously. The procedure is as follows:

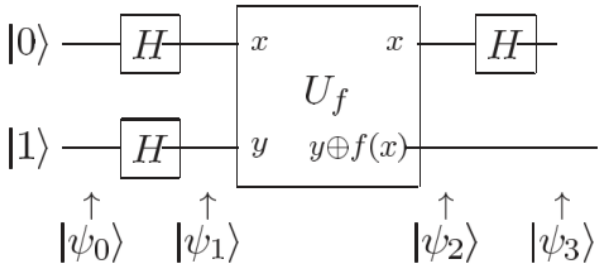


FIGURE 3. Circuit for Deutsch algorithm.[17]

- Input state = $|\psi_1\rangle = \left[\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right]\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right]$
- Since applying X gate on $|-\rangle$ gives $-|-\rangle$, we can define U_f as:

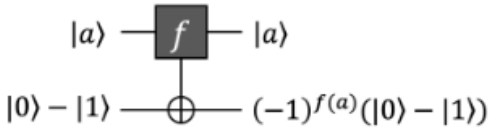


FIGURE 4. Construction of oracle.[19]

Due to phase kickback, the output can be interpreted as:

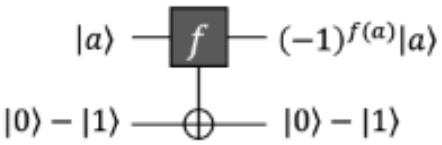


FIGURE 5. [19]

Thus, If $f(0) = f(1), \left|\psi_2\right\rangle = (-1)^{f(x)}\left|+\right\rangle \left|-\right\rangle$. If $f(0) \neq f(1)$, $\pm|-\rangle|-\rangle$.

On applying the Hadamard gate on the first qubit and measuring it, measuring 0 would imply a constant function and measuring 1 would imply a balanced function.

Phase kickback exploits the property that applying a phase on a pair of entangled qubits allows the phase to be shifted to either of the entangled qubits.[20]

3.3.1. *Experimental verification:.*

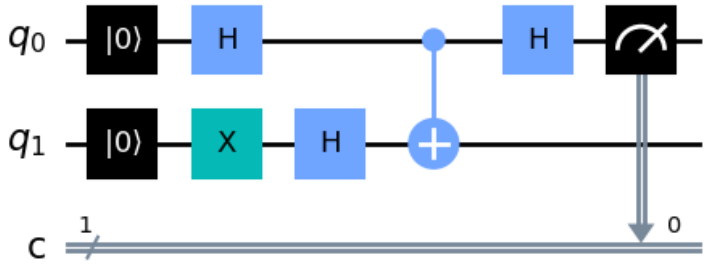


FIGURE 6. Here, f(0)=0, f(1)=1.

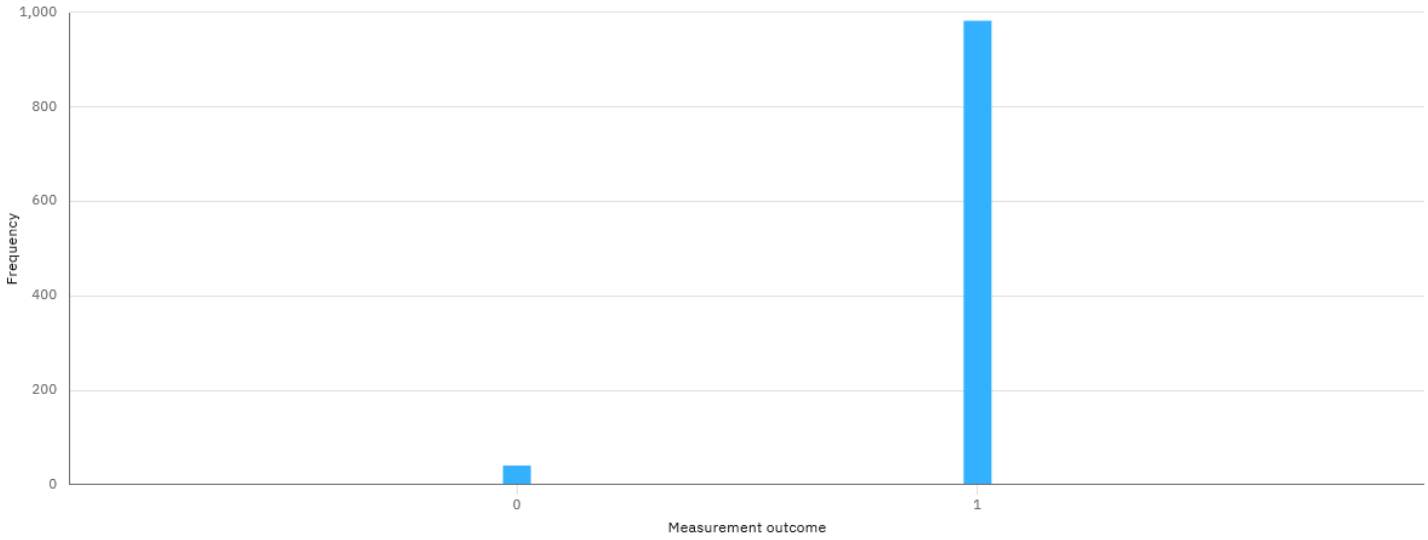


FIGURE 7. “0”:41,"1″:983.

Run on IBMQ-Lima, an accuracy of 96% was observed.

3.4. **Deutsch-Jozsa Algorithm.**

This is a generalization of the Deutsch problem.

Problem: Given a function f: $\{0,2^n-1\} \rightarrow \{0,1\}$, how many queries are required to determine if the function is constant or balanced?
In classical computing, in the worst case $2^{n-1}+1$ queries are required. This algorithm needs to make only one query.

The procedure is as follows:

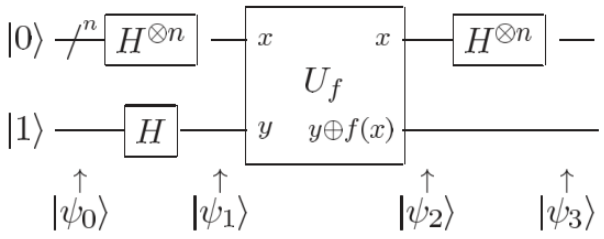


FIGURE 8. Circuit for Deutsch-Jozsa algorithm.[17]

$$|\psi_1\rangle = \sum_{x\in\{0,1\}^n} \frac{|\ x\rangle}{\sqrt{2^n}}\ |-\rangle, |\ \psi_2\rangle = \sum_x \frac{(-1)^{f(x)}\ |\ x\rangle}{\sqrt{2^n}}\ |-\rangle$$

.

- Applying the Hadamard transform on the first set of qubits and measuring it tells us whether the function is balanced or constant.
- The phase kickback method is used here. If the function is constant, then the amplitude of $|0...0\rangle$ is either -1 or 1, and as a result we should obtain 0...0 on measurement. If it were balanced, then $|0...0\rangle$ will have 0 probability amplitude since the positive and negative contributions cancel out, and thus we would obtain any other state on measurement.

3.4.1. *Worked example and experimental verification.*

Let the function be $f(0)=1, f(01)=0, f(10)=0, f(11)=1$.

$$|\psi_0\rangle = |00\rangle \otimes |\ 1\rangle$$

.

$$|\ \psi_1\rangle = \left[\frac{|00\rangle + |01\rangle + |10\rangle + |11\rangle}{2}\right]\left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right]$$

$$|\ \psi_2\rangle = \left[\frac{|001\rangle + |010\rangle + |100\rangle + |111\rangle - |000\rangle - |011\rangle - |101\rangle - |110\rangle}{2\sqrt{2}}\right]$$

$$|\ \psi_2\rangle = \left[\frac{|00\rangle \otimes -(|0\rangle - |1\rangle) - |01\rangle \otimes -(|0\rangle - |1\rangle) - |10\rangle \otimes -(|0\rangle - |1\rangle) + |11\rangle \otimes -(|0\rangle - |1\rangle)}{2\sqrt{2}}\right]$$

$$|\psi_2\rangle = [|-\rangle|-\rangle] \otimes - \ |-\rangle$$

Applying Hadamard transform on the first two qubits gives $|11\rangle$, thus on measurement we obtain a balanced function as result.

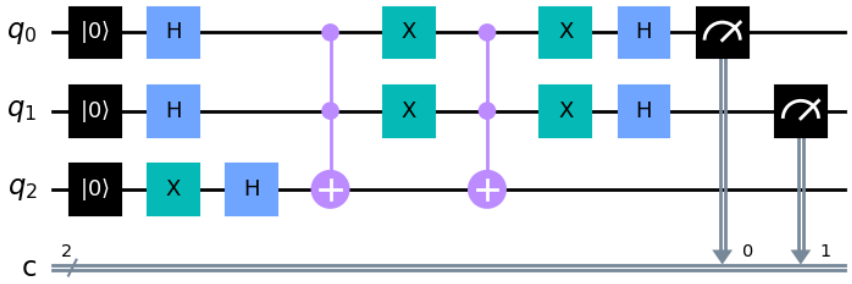


FIGURE 9. Here, $f(0) = 1, f(1) = 0, f(2) = 0, f(3) = 1$

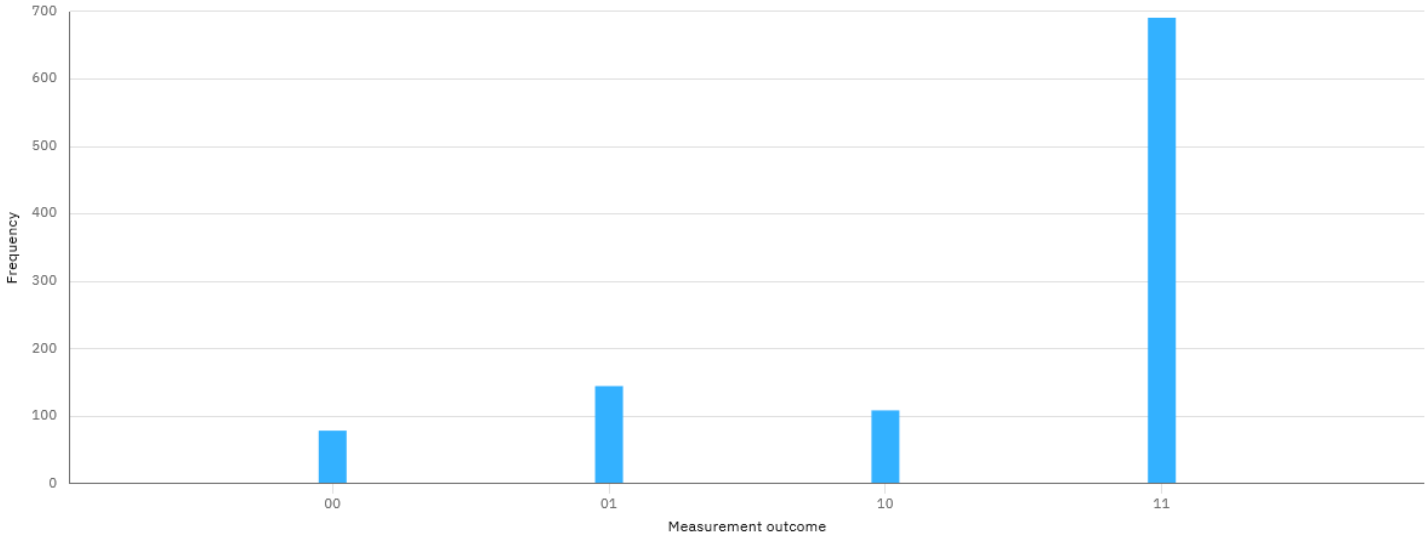


FIGURE 10. "00": 79, "01": 145, "10": 109, "11": 691

Run on IBMQ-Quito, an accuracy of 67.4% was observed.

3.5. **Bernstein-Vazirani Algorithm.**

Problem: Given a function $f : \{0,1\}^n \to \{0,1\}$ where $f = s.x$ and $s \in \{0,1\}^n$, what is the minimum number of queries required to determine s?

(*Here we refer to bitwise multiplication, where 101.111=1+0+1=0*)

In classical computing, we require n queries of the form $(100...0), (010...000),.., (...001)$ to extract each digit of s separately. This algorithm requires one query.

Although there are no restrictions on the construction of the oracle, for the sake of explanation the oracle has been interpreted as a composition of CNOT gates[21].:

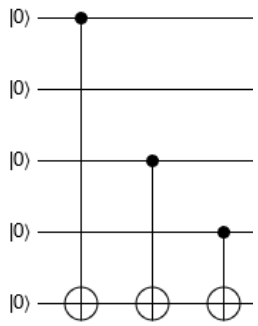


FIGURE 11. Oracle for s=1101.

Any results obtained from this construction will be indistinguishable from any other construction of the oracle. The objective is to flip the phase of every qubit $|x_i\rangle$ for which $x_i * s_i = 1$. This is achieved through the following circuit:

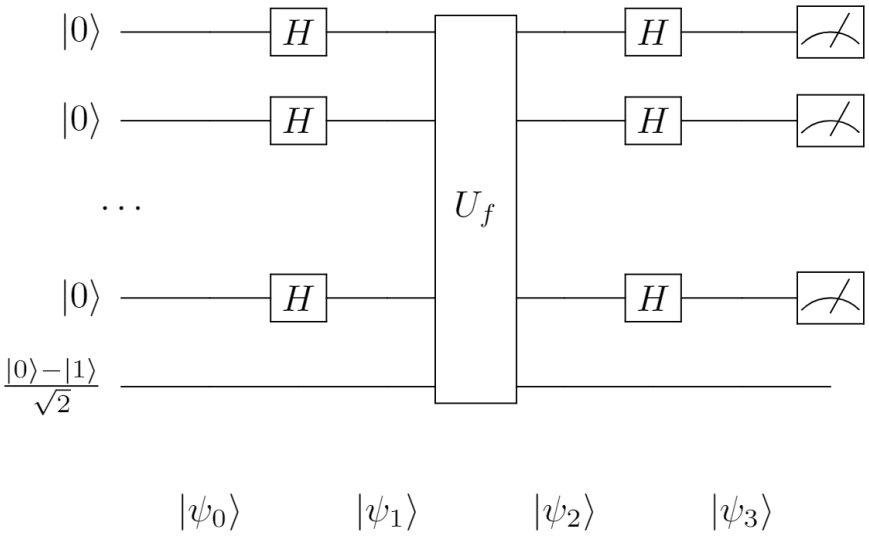


FIGURE 12. Construction for Bernstein-Vazirani algorithm[17]

We observe that the following gate compositions are equvialent:

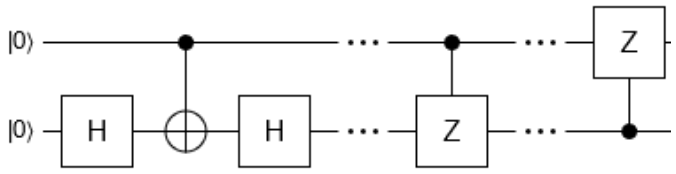


FIGURE 13. Equivalent circuits (Spaced)

Thus, we can interpret the algorithm as follows[21]:

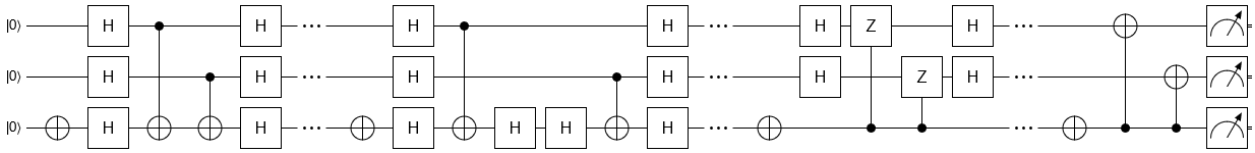


FIGURE 14. Transformations of circuit for s=11 (spaced).

3.5.1. *Experimental verification.*

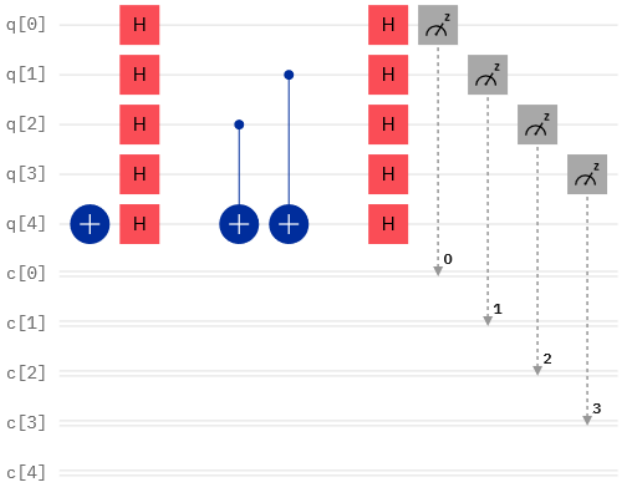


FIGURE 15. Circuit for s=0110.

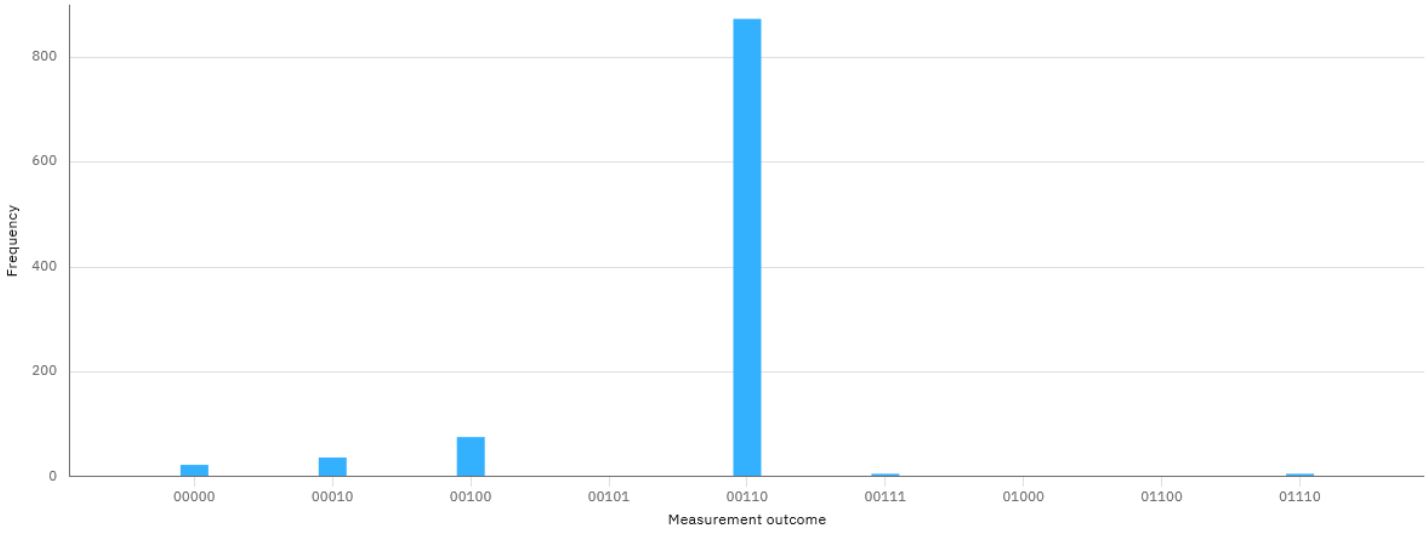


FIGURE 16. "0110":873

Run on IBMQ-Nairobi, an accuracy of 85.2% is observed.

3.6. **Grover's Search Algorithm.**

Problem: Given a function f: {0,1,..N-1} -> {0,1}, it is known that f(x) = 1 only for a single value ω. How many queries are required to determine ω?

Although this problem was referred by Lov Grover himself to be an unstructured database search algorithm [11], the interpretation given above is adhered to in most cases.

In classical computing, since there is no pattern to follow in order to determine the correct element, the best possible algorithm requires N/2 queries on average, and requires N queries in the worst case.
On average, this algorithm requires roughly $\sqrt{N}$ queries.
This algorithm works by flipping the phase of the target by using an oracle and "boosting" its probability amplitude iteratively. This oracle can either be hard coded or be constructed such that it "computes" the output, as seen in 3-SAT problems.[22, 23] Here, we work with the hidden bit string construction.

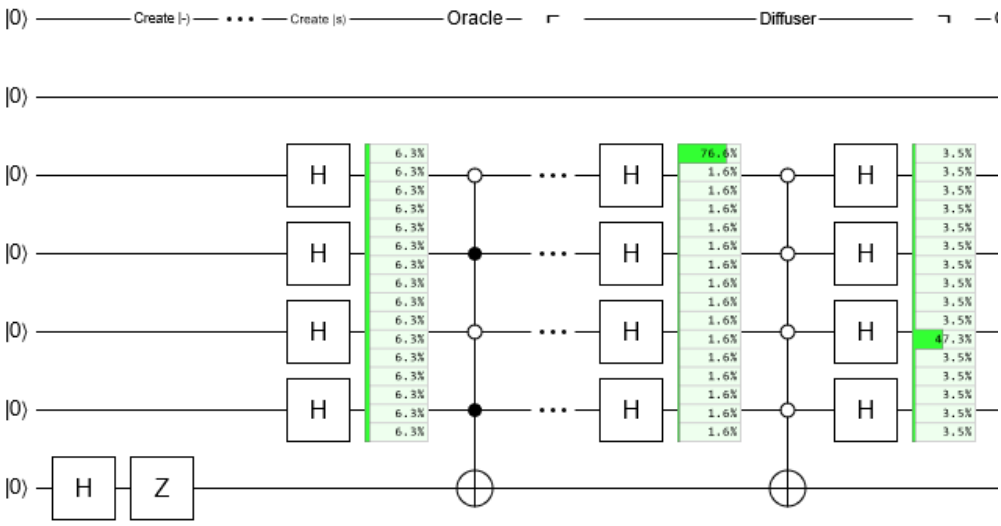


FIGURE 17. Construction for f(1010)=1.

The procedure is as follows:

- Via Hadamard transform, convert the input $|00...0\rangle$ to a superposition of all possible basis states $[|s\rangle]$.
- Apply the Oracle gate to flip the phase of the target $|\omega\rangle$ (a substate of $|s\rangle$) only. This reduces the mean amplitude to below $\frac{1}{\sqrt{n}}$.
- Apply the diffusion operator $2|s\rangle\langle s| - I$ to invert the amplitude about the mean amplitude.
- Perform steps 2 to 3 roughly $\sqrt{N}$ steps to increase probability of measuring the target state to near 100%.

The diffusion operator is responsible for "boosting" the probability amplitude of the target state. It can be interpreted using a Hadamatrix[24] or through geometry.

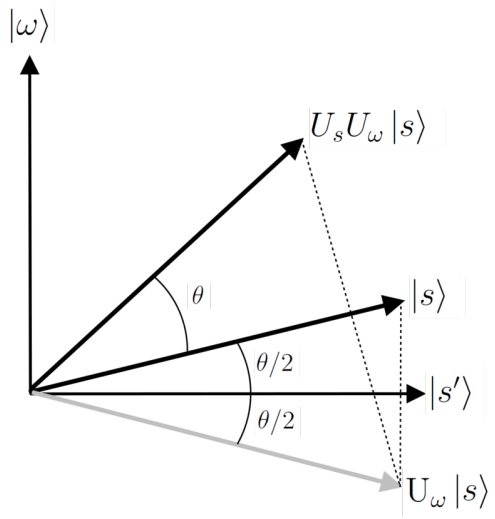


FIGURE 18. Geometric interpretation of Grover's algorithm

We can define our initial state $|s\rangle$ as a vector in the vector space spanned by $|\omega\rangle$ and $|s'\rangle$, where $|s'\rangle$ is a state without ω component. The application of the oracle gate results in a reflection about $|s'\rangle$, and the diffusion operator can be interpreted as a reflection about $|s\rangle$. The combination of these reflections is equivalent to a rotation towards $|\omega\rangle$. The oracle operation and diffusion operation need to be performed $\frac{\pi}{4}\sqrt{n}$ to ensure the maximum probability of measuring the desired state. Even though this algorithm is not deterministic, this can be rectified with modifications [25, 26]. Zalka shows that the Grover search is optimal.[27]

3.6.1. *Experimental Verification.*

The Quirk circuit for `hidden string=1010` indicates 90.8% success rate for 2 iterations, and similarly the Q-Composer circuit for `hidden string=0101` indicates 91.4% success rate for 2 iterations.

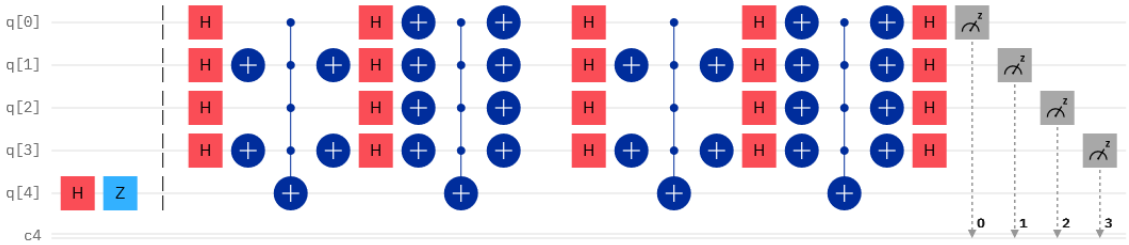


FIGURE 19. Construction for f(0101)=1.

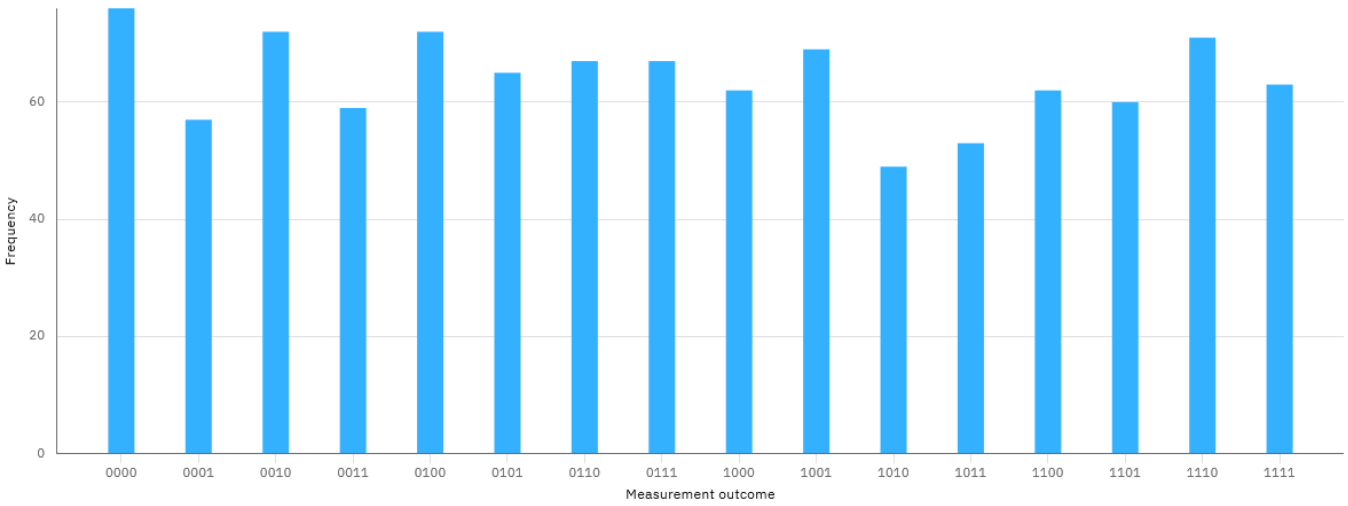


FIGURE 20. "0101": 65

Run on IBMQ-Perth, an accuracy of 6.3% was observed. This result replicates the results of other papers[28], showing that publicly available quantum computers are only able to handle simple calculations and gate operations without a high frequency of errors.

3.7. **Simon's Algorithm.**

Problem: A function $f : \{0,1\}^n \to \{0,1\}^n$ is defined such that for any two domain elements a and b, if $a \bigoplus b = r$ then $f(a) = f(b)$. How many queries are required to determine r? ($\bigoplus$ is the bitwise XOR, thus no carry over.)

In the case where r= 0^n, $a = b$, and our function will be 1-to-1. In any other case, our function will be 2-to-1 and if a and b are linked to an output, then $b = a \bigoplus r$.

a	f(a)
000	011
001	101
010	000
011	010
100	101
101	011
110	010
111	000

This is a function that satisfies all the criteria, and here $r = 101$.

A classical computer will require at most $2^{N-1} + 1$ queries, while a classical probabilistic computer must still make queries of the order $O\left(\sqrt{2^N}\right)$ to be correct 75% of the time.[29] This algorithm has a time complexity of O(N). Similar to the original solution to Grover's search, this is not a deterministic algorithm. The procedure is as follows:

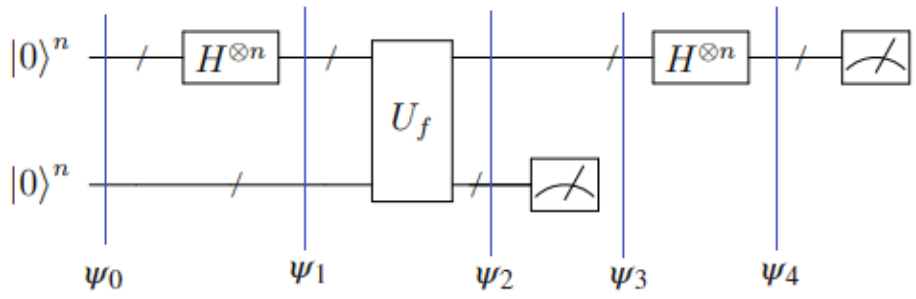


FIGURE 21. Construction of circuit for Simon's algorithm.

Where the oracle U_f can be defined as:

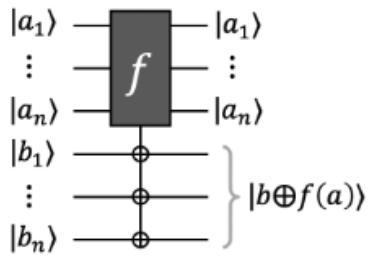


FIGURE 22. *a and b is not used in the same manner for explanation as it is in the oracle diagram. The above is only for illustration purposes.*

The procedure is as follows (as stated by Richard Cleve[30])

- Let the state of the first n-qubits after the Hadamard transform be $|x\rangle$.
- $|\psi_1\rangle = \frac{1}{\sqrt{2^n}}|x\rangle|0^n\rangle$
- $|\psi_2\rangle = \frac{1}{\sqrt{2^n}}\sum_{a\in\{0,1\}^n}(|a\rangle\ |f(a)\rangle)$

A basis state of the first set of qubits is entangled with a basis state of the second set of qubit, analogous to the manner the domain of the function is linked to its range.

On measuring the second set of qubits, we are collapsing our domain to the set of basis states that was entangled with the measured basis state. Let this measured basis state be b.

Measuring the second set of qubits gives us $|f(b)\rangle$, but at the same time it also gives us $|f(b \bigoplus r)\rangle$ due to the property of the function.

As a result, the measurement of the second set of qubits eliminates those set of values that do not map to the function's output, giving us $|\psi_3\rangle = \frac{1}{\sqrt{2}}|b\rangle + \frac{1}{\sqrt{2}}|b \oplus r\rangle$ as the final state of the first set of qubits.

Since $H^{\otimes n}|b\rangle = \frac{1}{\sqrt{2^n}}\sum_{a\in\{0,1\}^n}(-1)^{a\cdot b}\ |a\rangle$, applying the Hadamard transform on $|\psi_3\rangle$ gives:

$$H^{\otimes n}\left(\frac{1}{\sqrt{2}}\mid b\rangle + \frac{1}{\sqrt{2}}\mid b\oplus r\rangle\right) \Rightarrow \frac{1}{\sqrt{2^{n+1}}}\sum_{a\in\{0,1\}^n}\left((-1)^{a\cdot b}(1+(-1)^{a.r})\mid a\rangle\right)$$

If $a.r = 1$, then the probability amplitude of $\mid a\rangle$ becomes 0. As a result, for our state measured, $a.r = 0$ must be true, thus $(a_1r_1 + a_2r_2....)mod\ 2 = 0$.

By resetting the qubits and repeating the process, we can obtain more unique equations. For N queries, we obtain N distinct equations, and via Gaussian elimination the system of equations can be solved in $O(N^3)$. Even if on certain iterations of this algorithm a unique equation is not obtained, it can be shown through analysis that the expected number of iterations required to obtain all the unique equations still lie in O(N).

Simon's algorithm was the first to demonstrate an exponential speedup.

3.7.1. *Example and Experimental verification.*

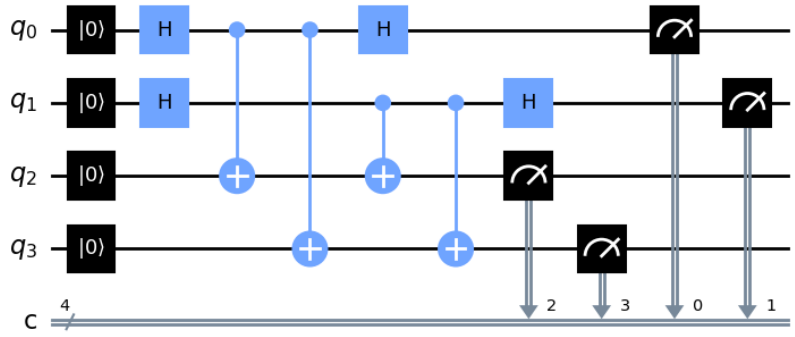


FIGURE 23. The function is: $f(00) = f(11) = 00; f(01) = f(10) = 01$, and $r = 11$.

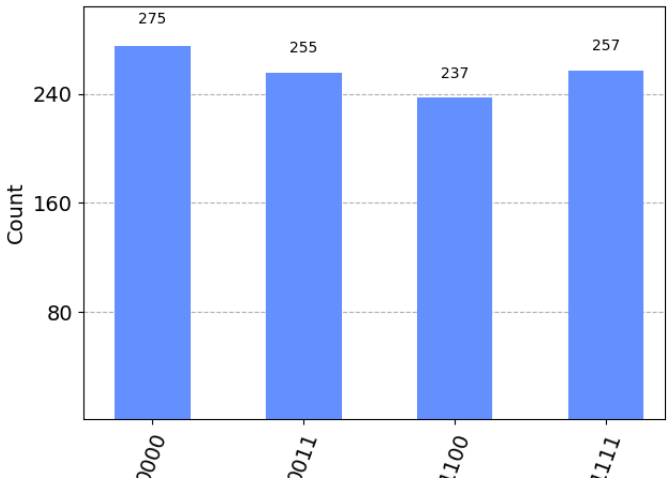


FIGURE 24. Measurement on a Quantum Computer simulation.

After measuring the second set of qubits, we get either 00 or 11 with equal probability. Measurement of the first set of qubit over various iterations gives us $r.00 = 0$ and $r.11 = 0$ from which we can say that $r = 11$.

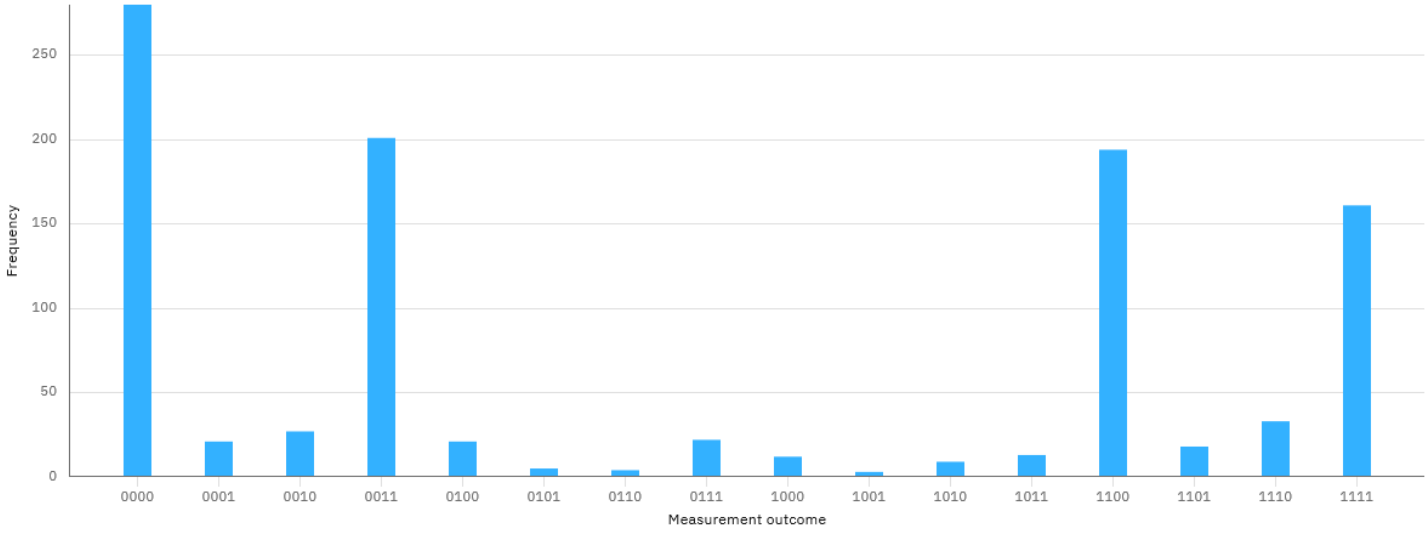


FIGURE 25. Favourable outcomes: 856

Run on IBMQ-Nairobi, we observe an 85.6% accuracy. This implies that there is an 85.6% chance of obtaining a non-erroneous system of equation. Further probabilistic analysis is necessary to determine the feasibility of this algorithm with the presence of errors.

3.8. **Quantum Fourier Transform.**

This algorithm allows for the "efficient" application of Direct Fourier Transform in Quantum Computing.
Just as how a basis vector in a discrete space can be mapped to a Fourier basis, the QFT maps a computational basis state to a Fourier basis state:

$$F\left|a\right\rangle = \tfrac{1}{\sqrt{m}} \sum_{j=0}^{m-1} \omega^{ja} \left|j\right\rangle,$$

Where $a \in \{0, m-1\}$ and $\omega^m = 1$, $\omega = e^{2\pi i/m}$, $m = 2^n$. Its matrix representation is given by :

$$F_m = \frac{1}{\sqrt{m}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{m-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(m-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{m-1} & \omega^{2(m-1)} & \cdots & \omega^{(m-1)^2} \end{bmatrix}.$$

From this, we can calculate the inverse QFT as well. Both are unitary.
Applying QFT on the 0 basis vector gives is equvialent to applying the Hadamard transform on that statevector. Lastly, if we were to apply it on an n-digit input in $\bmod m$ we get:

$$(F_m)^{\otimes n} \left|a_1, a_2, \ldots, a_n\right\rangle = \frac{1}{\sqrt{m^n}} \sum_{b \in \mathbb{Z}_m^n} \omega^{a \cdot b} \left|b_1, b_2, \ldots, b_n\right\rangle$$

$$(F_m^*)^{\otimes n} \left|a_1, a_2, \ldots, a_n\right\rangle = \frac{1}{\sqrt{m^n}} \sum_{b \in \mathbb{Z}_m^n} \omega^{-a \cdot b} \left|b_1, b_2, \ldots, b_n\right\rangle$$

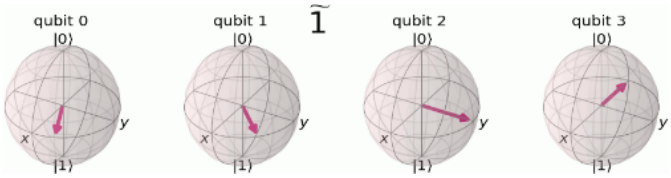


FIGURE 26. "1" encoded in Fourier basis.[31]

(In the given diagram, the ascending order of significance of the qubit runs from right to left.)

The transformation can be interpreted as follows:

We can write a as $a = a_n \cdot 2^{n-1} + a_{n-1} \cdot 2^{n-2} + \ldots + a_1 \cdot 2^0$ for $a_k = 0, 1$.

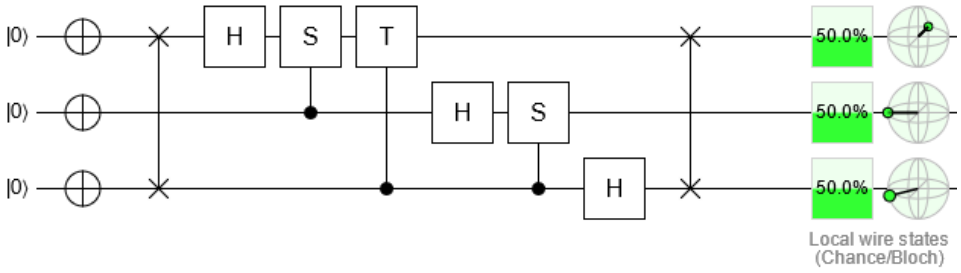
$$F_N \mid a\rangle = \frac{1}{\sqrt{N}} \left(\left|0\right\rangle + e^{\frac{2\pi i}{2}a}\left|1\right\rangle\right) \otimes \left(\left|0\right\rangle + e^{\frac{2\pi i}{2^2}a}\left|1\right\rangle\right) \otimes \ldots \otimes \left(\left|0\right\rangle + e^{\frac{2\pi i}{2^{n-1}}a}\left|1\right\rangle\right) \otimes \left(\left|0\right\rangle + e^{\frac{2\pi i}{2^n}a}\left|1\right\rangle\right)$$

First, applying the Hadamard gate on the qubit associated with the kth digit of a number (henceforth referred to as $\mid a_k\rangle$) will yield either $\mid +\rangle$ or $\mid -\rangle$. We choose the vector associated with $\mid +\rangle$ to be the starting vector as it points along the x-axis the Bloch sphere representation.
The Fourier basis is set up in such a manner that the frequency of rotation of the kth qubit's state vector (after Hadamard) decreases with increase in k. (Here, frequency is interpreted as the difference in phase angle of a_k b/w $F_N \mid a\rangle$ and $F_N \mid a+1\rangle$.) Here, $2\pi i \,/\, 2^n$ is treated as one unit of rotation. QFT performs the following transformation:

$$\mid a_k\rangle \rightarrow \frac{1}{\sqrt{2}}\Big(\big|0\rangle + e^{\frac{2\pi i a}{2^k}}\big|1\rangle\Big) \Rightarrow \frac{1}{\sqrt{2}}\Big(\big|0\rangle + \omega^{a2^{n-k}}\big|1\rangle\Big)$$

. Since $a = a_1 \cdot 2^0 + a_2 \cdot 2^1 + ... + a_n \cdot 2^{n-1}$ for $a_k = 0, 1$, we can control the phase of a qubit's state by the conditional operation of rotation gates on it.

For example, let $a = 7$. its bit representation is 111. To transform $|a_k\rangle$ we apply the following algorithm below:



The qubit values are swapped at the start of the algorithm in order to maintain the little endian convention. Here, the $\mid a_3\rangle$'s state has been swapped with that of $\mid a_1\rangle$. (To avoid confusion, the qubit states will only refer to their current designated qubit, instead of the original or intended qubit.) $|a_3a_2a_1\rangle = |111\rangle$ To transform $|a_3\rangle$:

$$|a_3\rangle = |1\rangle \Rightarrow \frac{1}{\sqrt{2}}\Big(\big|0\rangle + e^{2\pi i\frac{7}{8}}\big|1\rangle\Big) = \frac{1}{\sqrt{2}}\big(|0\rangle + \omega^7|1\rangle\big)$$

.

$$\omega^7 = \omega^{a_1+a_2+4a_3}$$

- Since the value of $\mid a_3\rangle$ has been shifted to $\mid a_1\rangle$, applying the Hadamard on $\mid a_1\rangle$ causes rotation by π from $\mid +\rangle$. This can also be imagined as a control-Z gate being applied with the control and target qubit being the same. Thus, this takes care of the $4a_3$ factor.
- In the same manner, the conditional S gate takes care of the $2a_2$ factor, and the controlled T gate takes care of the a_1 factor.

Since T causes rotation by $\pi \,/\, 8$ which corresponds with our one unit of rotation, the transformation of $\mid a_3\rangle$ is complete, albeit at as the value of $\mid a_1\rangle$.
Similarly,

$$|a_2\rangle = |0\rangle \Rightarrow \frac{1}{\sqrt{2}}\Big(\big|0\rangle + e^{2\pi i\frac{14}{8}}\big|1\rangle\Big) = \frac{1}{\sqrt{2}}\big(|0\rangle + \omega^{14}|1\rangle\big)$$

$$\omega^{14} = \omega^{2(a_1+2a_2+4a_3)} = \omega^{2a_1+4a_2+8a_3}$$

We can see that the net result of the controlled rotation due to $|\ a_3\rangle$ is equivalent to rotation by 2π, thus we can completely neglect the rotation caused by it. The Hadamard behaves like a controlled -Z rotation, and the controlled-S gate acts on it, taking care of the factor of $4a_2$ and $2a_1$ respectively. By doing the same process for a_1, $\omega^{28} = \omega^{4a_1+8a_2+16a_3}$ and thus, only the Hadamard needs to be applied.
The last operation that needs to be applied is SWAP since the transformed value of $|\ a_1\rangle$ is stored as the value of $|\ a_3\rangle$ and vice versa.[32]

Lastly, for the Inverse QFT, only the order of operations needs to be reversed and the angle of rotation be negated (eg. $-\frac{\pi}{2}$ instead of $\frac{\pi}{2}$).

3.8.1. *Examples and Experimental verification.* The results obtained on a simulated QC match our calculations, as seen in the `8 qubit QFT circuit in Quirk` and the `4 qubit QFT circuit in Q-Composer`. This algorithm can only be tested on a real QC by applying QFT on an input and then applying the inverse, to check if the final result matches with the input.

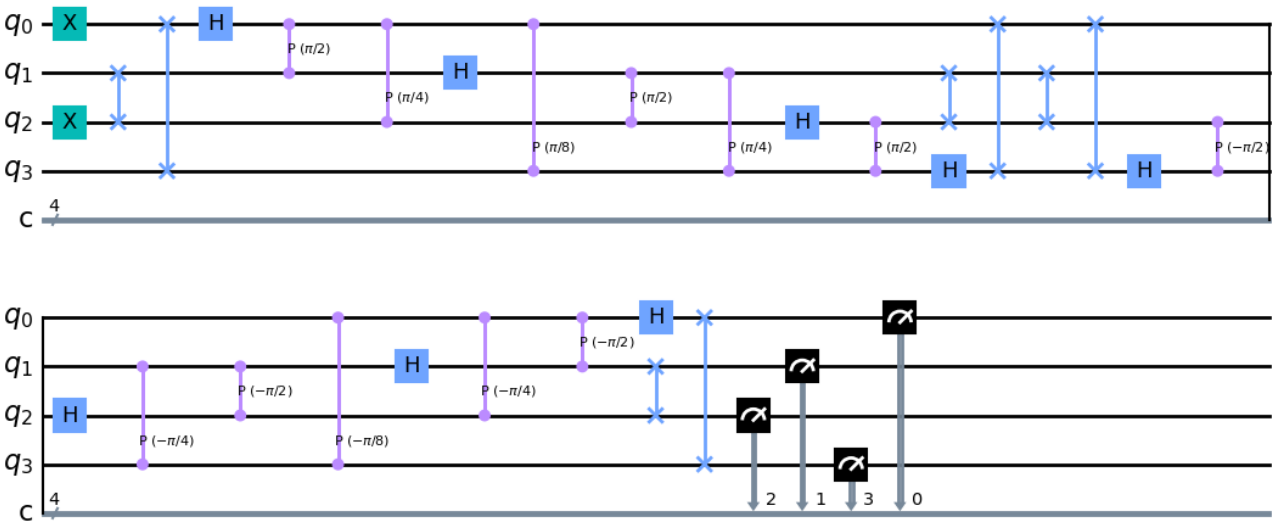


FIGURE 27. 4 digit QFT-IQFT circuit, x=0101.

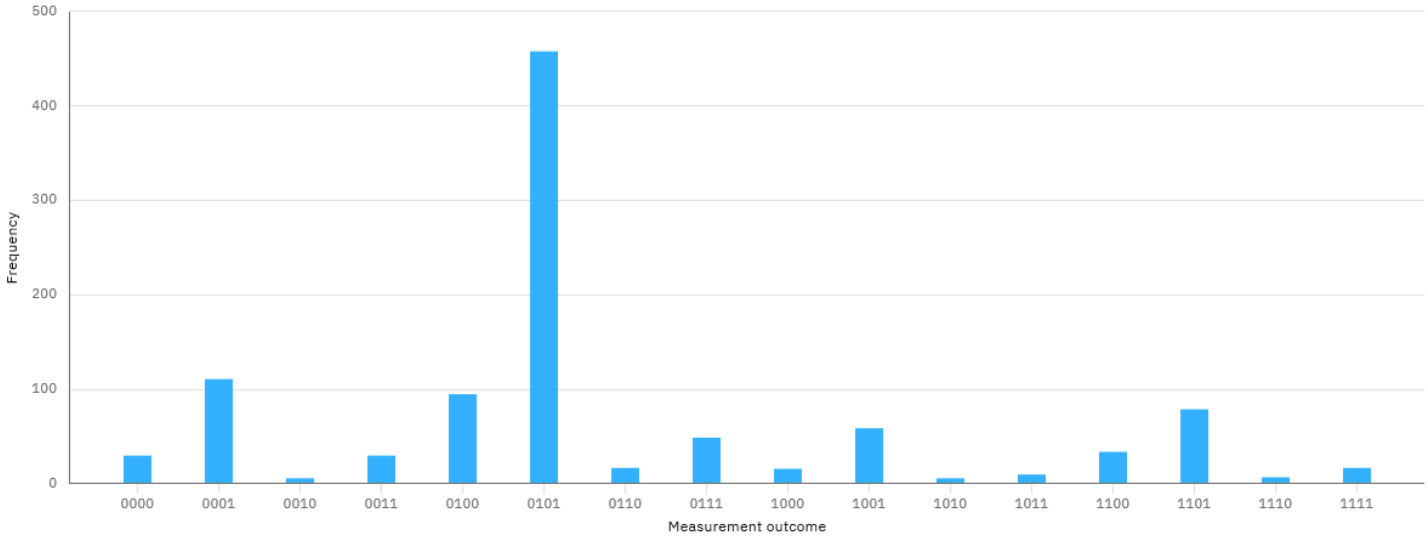


FIGURE 28. "0101":458

The 4-qubit QFT circuit was run on IBMQ-Quito for x=0101, and the inverse QFT was applied to recover the input. 44.7% accuracy was observed.

3.9. **Shor's Algorithm for Discrete Log Problem.**

This algorithm will be approached in the manner stated by Richard Cleve[33]:

3.9.1. *Importance.* A and B want to exchange some secret information through an insecure channel. They use the **Diffie–Hellman–Merkle key exchange** to mitigate the problem.

- A and B agree on a public key (here, value x).
- A and B choose a random value to be their private keys (a and b respectively)
- A and B send each other $x^a(\alpha)$ and $x^b(\beta)$ respectively.
- A and B calculate β^a and α^b respectively.
- Both now have the value x^{ab}. This will serve as their secret key for communication.

If an attacker C intercepts the exchange, it will have access to x and either x^a and x^b or both. It will then have to determine a and b by taking the logarithm of the intercepted values. Unfortunately (or fortunately) there is no classical algorithm as of today that can solve this problem in polynomial time. Due to this reason, it is still a widely used protocol for internet encryption (even if it may not be bulletproof any longer[34]).

Shor's algorithm can solve this problem in polynomial time.[10]

3.9.2. *Definitions.*

(Ring) $\mathbb{Z}_{\boldsymbol{m}}$: *The set* $\{0, 1,..., m-1\}$ *with the operations of addition and multiplication mod* m.

(Group) $\mathbb{Z}^*{}_{\boldsymbol{m}}$: *Set of* $\mathbb{Z}_m$ *elements that have multiplicative inverse in* $\mathbb{Z}_m$. *This set has the operation of multiplication mod* m.
(Valid only for $m \geq 2$.)

As seen, 0 and all other numbers that are factors of m have no multiplicative inverse in $\mathbb{Z}_m$. Thus, for any prime number $p, \mathbb{Z}^*{}_p = \{1,..., p-1\}$ with operation of multiplication mod p.

An element g of the group $\mathbb{Z}^*{}_{\boldsymbol{p}}$ is called a generator of the group if $\mathbb{Z}^*{}_{\boldsymbol{p}}$ can be constructed from the powers of g.
If a group has such an element, it is called cyclic.

For example, 2 and 3 are the generators for $\mathbb{Z}^*{}_5$, since $2^4 mod\, 5 = 3^4 mod\, 5 = 1$, $2^1 mod\, 5 = 3^3 mod\, 5 = 2$, $2^1 mod\, 5 = 3^3 mod\, 5 = 2$,...

Theorem 3.9.2.1. $\mathbb{Z}^*{}_p$ *has at least one generator* g, *such that* $\mathbb{Z}^*{}_p = \{g^k : k \in \{0, 1,..., p-2\}\}$.

It only needs to go till $p-2$ since $\mathbb{Z}_p$ only has $p-1$ elements. For example, 3 is the generator of $\mathbb{Z}^*{}_7$.

Through the above example we can see that multiplying any two elements in $\mathbb{Z}^*{}_p$ is equivalent to addition mod $p-1$ of the exponents of g corresponding to the elements. Thus, this holds when the exponents corresponding to the elements $\in \mathbb{Z}_{p-1}$.

Discrete exp function: For a chosen prime p and a chosen generator of $\mathbb{Z}^*{}_p$, we define $exp_g : \mathbb{Z}_{p-1} \to \mathbb{Z}^*{}_p \Rightarrow exp_g(r) = g^r$.
Discrete log function: For a chosen prime p and a chosen generator of $\mathbb{Z}^*{}_p$, we define $log_g : \mathbb{Z}^*{}_p \to \mathbb{Z}_{p-1}$ such that it is the inverse of the discrete exp function. When given an element from $\mathbb{Z}^*{}_p$, it gives the corresponding r.

3.9.3. *Formal statement of problem.*

Discrete exp problem: Given p, g, r, find $exp_g(r)$. This problem can be solved quite easily via exponentiation by squaring, and is of $O(n^2 log\, n)$ order.

Discrete log problem: Given p, g,and $s \in \mathbb{Z}^*{}_p$, find r such that $g^r = s$.

Simply put, we are given a prime p and two numbers. If we are promised that the first number raised to some number x when modded with p is equal to the second number, what is x?

No polynomial-time classical algorithms are known as of today.
Shor's algorithm solves it in at least polynomial time.

3.9.4. *Generalization of Simon's problem.*

We define Shor's function as $f : \mathbb{Z}_{p-1} \times \mathbb{Z}_{p-1} \to \mathbb{Z}^*{}_p$ such that $f(a_1, a_2) = g^{a_1} s^{a_2}$. If we see the function we encountered in Simon's problem, it had the property that if $f(a) = f(b)$, then $a \bigoplus b = r$ for some r.

We can see that the XOR operator $\bigoplus$ is equivalent to the subtraction of two numbers in mod 2 arithmetic. As a result, we can say that Simon's function has the following property: if $f(a) = f(b)$, then $a - b =$ multiple of r in mod 2 arithmetic. (Although it is a moot point here since multiple=even will give 0 and multiple=1 will give r)

We can see that Shor's function has a more generalized version of the above property, namely: $f(a_1, a_2) = f(b_1, b_2)$ when $(a_1, a_2) - (b_1, b_2)$ is a multiple of $(r, 1)$

in mod p-1 arithmetic.

Proof. If $f(a_1, a_2) = f(b_1, b_2)$, then $g^{a_1} s^{a_2} = g^{b_1} s^{b_2}$.
Applying log_g on both sides and using $s = g^r$, we get $a_1 + ra_2 = b_1 + rb_2 \Rightarrow a_1 - b_1 = r(b_2 - a_2)$
Thus, $a_1 - b_1$ is a multiple of r. Similarly, we can see that $a_2 - b_2$ is trivially a multiple of 1. Hence proved. □

Having showed this, we first solve a more generalized version of Simon's problem and show that it is linked to DLP.
The motivation for doing this is that there exists a a special Shor function that satisfies the generalized version of Simon's problem, and one of its outputs matches the output of the other. Solving the first gives us the second.

3.9.5. *Simon mod m problem.*
The generalized problem is as follows:

Given a m-to-1 function $f : (\mathbb{Z}_m)^d \to T$ has the property that there exists an element $r \in (\mathbb{Z}_m)^d$ such that if $f(a) = f(b)$, then $a - b$ is a multiple of r. Determine r.

Here the input has d digits and each digit is in base m. Here we can define $T = \{0, 1\}^k$ for some k. This gives us the freeway to set the size of the range's set accordingly. Note that T does not have to be in the same base, but it must have at least m^{d-1} elements. However, for our current purposes we take it to be $T = \mathbb{Z}_m^d$. Since the number of terms in the domain is m^d, it is always possible to make such a function.

For example, setting $d = 2, m = 3, r = 12$, here is a function that satisfies the criteria:

x	f(x)
00	00
01	01
02	10
10	01
11	10
12	00
20	10
21	00
22	01

Here, the multiples of $r = 12, 21, 00$. As seen, not all elements of T are included in the function's range. We can also see that the sets of points that have the same output are of the form $\{a, a + r,..., a + (m - 1)r\}$

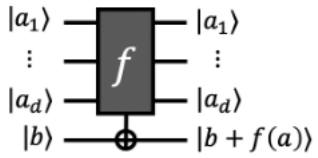


FIGURE 30. Oracle for Simon's mod m problem.[30]

As we can see, the output might be an element that does not belong to the range. (In the function table above, with input 11 and b=01, we obtain 11, a point not in the range.)
The digits are not in mod 2 either, instead they are in mod m. Thus, here the digits are not encoded into qubits, rather they are stored in "qumits" which consists of "m" mutually orthogonal quantum states. In the function table above, those states are $|0\rangle, |1\rangle, |2\rangle$.
The algorithm is implemented in the following manner:

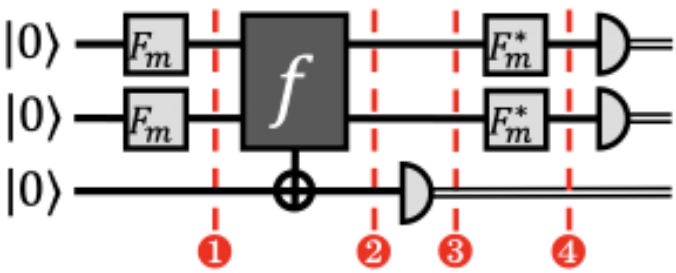


FIGURE 31. Circuit for Simon's mod m problem.[30]

Here, Fm and F*m are QFT and inverse QFT respectively (applied over the set of input qubits, not individually over each). The ancillary/ "target" of the Oracle (denoted in the oracle figure as $|b\rangle$) consists of 2 qumits, since d=2. Similarly, the first two qumits (the ones undergoing QFT) serve as the "input" to the oracle. One two digit input, one two digit output.

The input into this circuit is $|00\rangle$. After applying QFT,

$$\frac{1}{\sqrt{m^2}} \sum_{a \in \mathbb{Z}_m \times \mathbb{Z}_m} |a_1, a_2\rangle \, |0\rangle$$

After the oracle's action:

$$\frac{1}{\sqrt{m^2}} \sum_{a \in \mathbb{Z}_m \times \mathbb{Z}_m} |a_1, a_2\rangle \, |f(a_1, a_2)\rangle$$

On measurement of the third qubit, we collapse it to a particular output, and as a result collapse the number of observable states to those that would yield the output.

$$\frac{1}{\sqrt{m}} \sum_{k \in \mathbb{Z}_m} |(a_1, a_2) + k(r_1, r_2)\rangle$$

(The normalization factor here is the square root of m because the function is m-to-1, and as a result we have m states mapping to 1 output, whereas earlier we we dealing with the combination of two m-digit states.)
Applying the inverse QFT:

$$\frac{1}{\sqrt{m}} \sum_{b \in \mathbb{Z}_m \times \mathbb{Z}_m} \omega^{-a \cdot b} \left(\frac{1}{m} \sum_{k \in \mathbb{Z}_m} \omega^{-k(r \cdot b)} \right) |b_1, b_2\rangle$$

As seen, $\omega^{-a.b}$ plays no role, as its absolute value is 1. Thus, on measurement we get:

$$\begin{array}{ll} \frac{1}{m} & \text{if } (r_1, r_2) \cdot (b_1, b_2) = 0 \\ 0 & \text{if } (r_1, r_2) \cdot (b_1, b_2) \neq 0. \end{array}$$

From this, we get our set of equations. By repeating this process, we get r. Thus, we have shown how the generalized problem can be solved as well.

3.9.6. *Link with Discrete Log Problem.* We see that the solution discrete log problem is *exactly analogous* to the generalized Simon's problem.

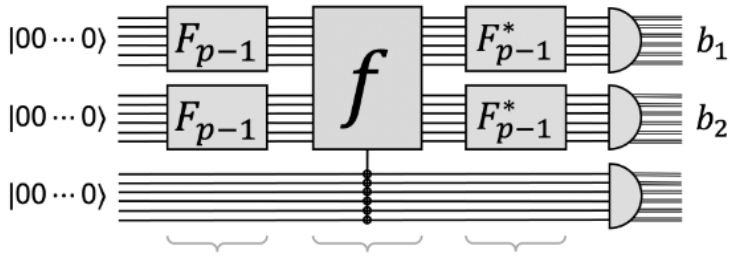


FIGURE 32. Circuit for DLP.

The only points to consider are:

- The function/oracle still receives only one input, but the input

consists of two independent parameters that we control. ($|\ ab\rangle$)

- The function must calculate $g^a s^b$, thus

the oracle must be hard coded with the value of s beforehand.
As stated in the discrete exponential problem, this multiplication and exponentiation can be done "efficiently".

3.10. **Shor's algorithm for factorization.**

3.10.1. *Importance.*
RSA is the most widely used cryptosystems and is used for bulk encryption-decryption. The details required for RSA without `padding` is described below:

- If $a = b\ mod\ n$, then $\exists\ k$ such that $a - b = kn \Rightarrow a = kn + b$.
- Bezout's identity states that $\exists\ r, s \in \mathbb{Z}$ such that $ar + bs = gcd(a, b)$
- Since $gcd(a, b) = gcd(b, a \bmod b)$ and $gcd(x, 0) = x$, we can find the gcd and then "build back up" r and s.[35] For example:
- $gcd(78, 66) = gcd(66, 12) = gcd(12, 6) = gcd(6, 0) = 6$

$$\begin{aligned}
&\Rightarrow 6 = 12 - 1(6)\\
&= 12 - 1(66 - 5(12))\\
&= 12 - 1(66) + 5(12)\\
&= -1(66) + 6(12)\\
&\cdots\\
&= 6(78) - 7(66)\\
&\Rightarrow r = 6, s = -7
\end{aligned}$$

For two coprimes, their gcd = 1.
For the rest of the section, 616 and 487 will be used. For this pair, $r = -151, s = 191$.

- Euler's Totient Function gives us the number of integers co-prime to the input.

$$\varphi(n) = n \prod_{p|n} \left(1 - \frac{1}{p}\right)$$

FIGURE 33. Euler's Totient Function.

It is clear that $\phi(prime) = (prime - 1)$, thus for primes p, q we see that $\phi(pq) = (p-1)(q-1)$.

The basic procedure behind RSA is described below:

- A and B want to communicated securely.
- A chooses at random two primes p and q. Here, p=23, q=29.
- A determines $a = \phi(pq) = \phi(667) = 616$.
- Choose a co-prime to a $\Rightarrow b$ =487 (verified via Euclid's algorithm).
- Find their Bezout coefficients r and s (-151,191).
- Since $a \cdot r + b \cdot s = 1, b \cdot s - 1 = r \cdot a, b \cdot s = 1 \bmod a$.
- A's public key is $(pq, b) \Rightarrow$ (667,487). A's private key is $(pq, s) \Rightarrow$(667,191).
- Similarly, B repeats this process and creates its own keys, B's public key is $(p'q', b')$. B's private key is $(p'q', s')$.
- If A wants to send a message to B, A encodes its message using B's public key, and B decrypts it using its own private key.
- Let the message to be encrypted be encoded as the number x.
- A sends the cipher number c to B, where $c = x^{b'} \bmod p'q'$.
- B uses his private key to calculate $c^{s'} \bmod p'q' = x$.

The last two steps of encryption and decryption can be proven through calculations.[36]

Bezout's coefficients can be calculated "efficiently" using classical algorithms. The biggest problem an attacker faces is the calculation of the Totient function's value itself, as its time complexity is equivalent to that of factorizing the number. Although textbook RSA/RSA without padding can easily be exploited[37], neither Totient function calculation for non-primes, nor integer factorization have any polynomial time classical algorithms as of today.

3.10.2. *Shor's algorithm for order finding.* Shor devises an algorithm for order finding, then proceeds to show that integer factorization can be reduced to a problem of order finding.

If we can determine the period of $a^x \bmod n$, we can factorize a number efficiently.

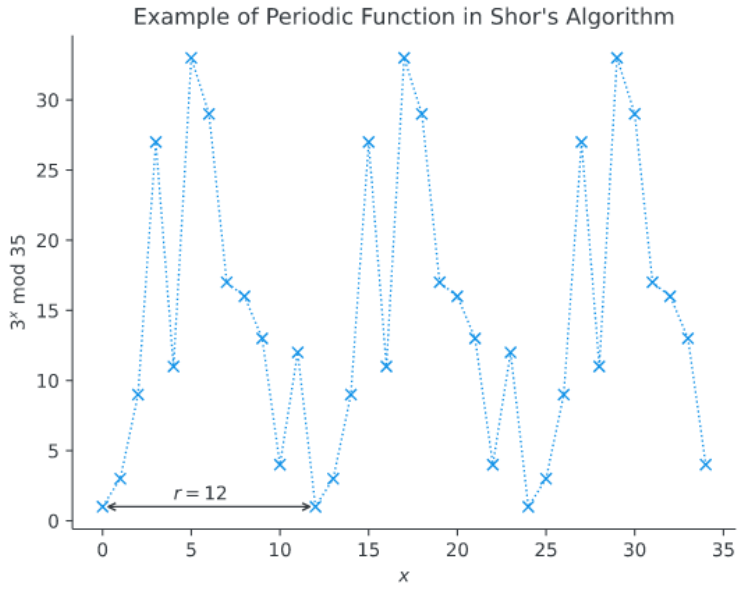


FIGURE 34. Periodicity of $3^x \bmod 35$.[38]

Given a function $f(x) = a^x \bmod n$, the period of the function is the smallest positive non-value of x such that $f(x) = 1$.
Let us define the unitary operator $U|y\rangle = |ay \bmod N\rangle$. We see that this is equivalent to the same operation being performed above but iteratively, since:

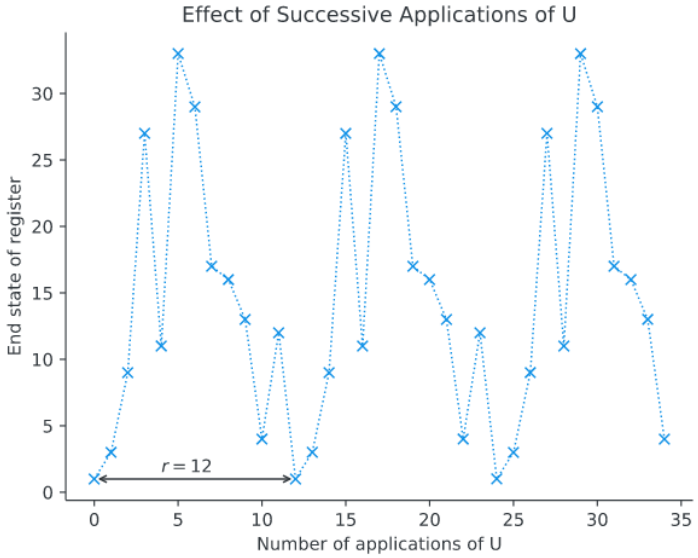


FIGURE 35. Successive applications of U.[38]

As a result, if we take the superposition of all the distinct values of the function $|1\rangle, |3\rangle, ..., |12\rangle$ and applied this operation, we would get back the same state as $|1\rangle \rightarrow |3\rangle, ..., |12\rangle \rightarrow |1\rangle$.

$$\mid u_0\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} U^k |1\rangle$$

Thus, this superposition state is an eigenstate of this gate, with eigenvalue =1.

Now, let us set up a superposition state such that the phase of the kth state is proportional to k:

$$\mid u_1\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{\frac{-2\pi i k}{r}} U^k |1\rangle$$

We see that the eignevalue here is $e^{\frac{-2\pi i}{r}}$. This can be generalised to $U\Big|u_s\rangle = e^{\frac{-2\pi i s}{r}}\Big|u_s\rangle$. Thus, for each integer s in $\{0, r-1\}$, there exists a unique corresponding eigenstate.

We can draw a parallel to this with the roots of unity. The r^{th} roots of unity are equally spaced points on the unit circle. In $\mid u_1\rangle$, the phase of $\mid k\rangle$ is equal to ω^k. For $\mid u_s\rangle$, the eigenvalue is equal to ω^s. We can also see that $1 + \omega^k + \omega^{2k} ... + \omega^{(r-1)k} = 0$, thus:

$$\frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |u_s\rangle = |1\rangle$$

,

($|1\rangle$ is not cancelled out as it has constant phase in all the states above.) As a result, every time we perform Quantum Phase Estimation[39, 30, 17] on $\mid 1\rangle$, we will measure a random phase of the form $s\,/\,r$, where s lies b/w 0 and r-1. To perform QPE and approximate (and conclusively determine) the value of s/r, l=2N+1 bits are required.[30].

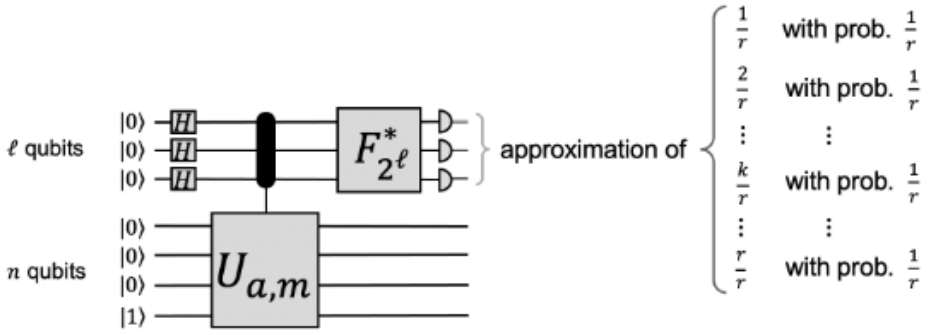


FIGURE 36. Circuit for Shor's order determination algorithm.

Here, we are utilising a multiplicity-controlled-U[19] gate rather than a controlled U gate. In the computational basis, the state of the control qubits specifies how many times U should be applied. Through classical processing,we can determine the denominator r.

3.10.3. *Algorithm of factorization.* Factorization is reduced to an extended version of the order finding algorithm in the following manner:

1. Initialize the number to be factorized (n).
2. Choose a random $a \in \{1, 2, ..., m-1\}$.

3. Verify if a and n are coprimes through Euclid's algorithm. If gcd(a,n)>1, the factors are gcd(a,n), n/gcd(a,n).
4. Determine order through order finding algorithm.
5. If r=odd, discard and restart from step 2.
6. If r=even, $a^r - 1 = \left(a^{r/2} + 1\right)\left(a^{r/2} - 1\right) = xy = Nk, N = pq$. We get the following possibilities:
    - p divides x and q divides y.
    - q divides x and p divides y.
    - p and q both divide x. (N cannot be divided by y as that would imply an order of r / 2).
7. Calculate gcd(n,$a^{r/2} + 1$) and gcd(n,$a^{r/2} - 1$). If gcd > 1, then the factors have been found. Else, repeat from step 2.

3.10.4. *Additional points*

.

Several resources exist that give a detailed explanation of Shor's algorithms and the mathematics behind them[40, 33, 41], however there still has been close to no progress towards factorization of bigger numbers, leading to the "15,21" deluge of papers. These have been criticized by some researchers[42]. Thus, the largest number to be factored is disputed[43]. Nevertheless, rough estimates have been drawn to determine the resources required to crack RSA in the NISQ era.[44]

4. Quantum Error Correction

Objective: This section explains and provides experimental verification of QEC methods up to Shor's code, and will not go into Steane's codes[45]. In addition to the above points, for the bitflip and phaseflip code in the report, only their error detection capabilities have been dealt with, as they can be easily modified to be self-correcting circuits.

The objective here is to reduce interference by noise as much as possible and maintain high fidelity. In this section we will only deal with the errors that occur due to noisy channels, and not reading errors or gate application errors.

In classical systems, errors occur due to interference, signal distortion and attenuation. In quantum circuits, decoherence is a major problem due to the sensitive nature of the setup. The nature of the system also gives rise to more "variety" in the errors that can occur compared to classical computing.

There are a few differences between classical error correction and QEC:

- No cloning theorem prevents the creation of independent qubits from some operation applied on a qubit.
- Errors can include either the shift in phase or probability amplitude of the components of a mixed state.
- Measurement of the qubit destroys the quantum nature of the information.

4.1. **Three bit repeititon code.**

Let us say that a classical bit must be transferred through a noisy channel where the probability of a bitflip is p. To mitigate this, instead of passing the bit as it is (as 0 or 1), 000 or 111 is passed instead. The erroneous bit is identified through majority voting. This method fails if two or more bits are flipped by the channel. Since p(2 or more flipped)=$3p^2 - 2p^3$, this method is beneficial only when p<1/2.

4.2. **Quantum bitflip channel correction.**

In this channel, with a probability p the bit gets flipped, thus:

$$|\psi\rangle = \alpha|0\rangle + \beta \mid 1\rangle \rightarrow (1-p)(\alpha|0\rangle + \beta|1\rangle) + p(\beta|0\rangle + \alpha|1\rangle)$$

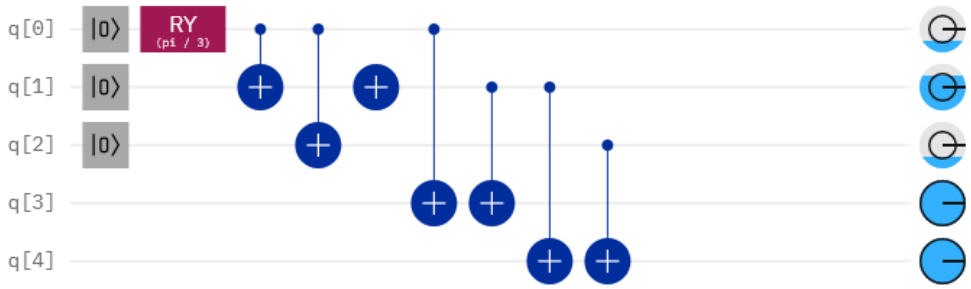


FIGURE 37. Quantum bitflip circuit, with $|\psi\rangle = \sqrt{3}\,/\,2|0\rangle + 1\,/\,2\,|1\rangle$ and flip in 2nd qubit.

Here, q[0] is the initial information carrying qubit, which for the purposes of explanation I have encoded as $\sqrt{3}\,/\,2|0\rangle + 1\,/\,2|1\rangle$ through CNOT gates the qubits are encoded as $\sqrt{3}\,/\,2|000\rangle + 1\,/\,2|111\rangle$)
Let us assume that the middle qubit is flipped by the channel.

This flipping changes the state of the syndrome qubits q[3] and q[4] which are controlled CNOT gates. Since the syndrome qubits read 11, we determine that the middle qubit was affected and we can apply the X gate to get our original data. (Alternatively, we can choose to ignore that qubit's data and not apply the correction, or use the syndrome bits to correct the error.)

q[3]	q[4]	Result
0	0	No flip
0	1	Third qubit flipped
1	0	First qubit flipped
1	1	Second qubit flipped

The measurement of the syndrome qubits does not give us any information about the states of the qubits that are carrying the data, it only indicates where the error has occurred. This circuit can only handle one bitflip, as it works on the principle of majority voting.

4.2.1. *Examples and experimental verification.*

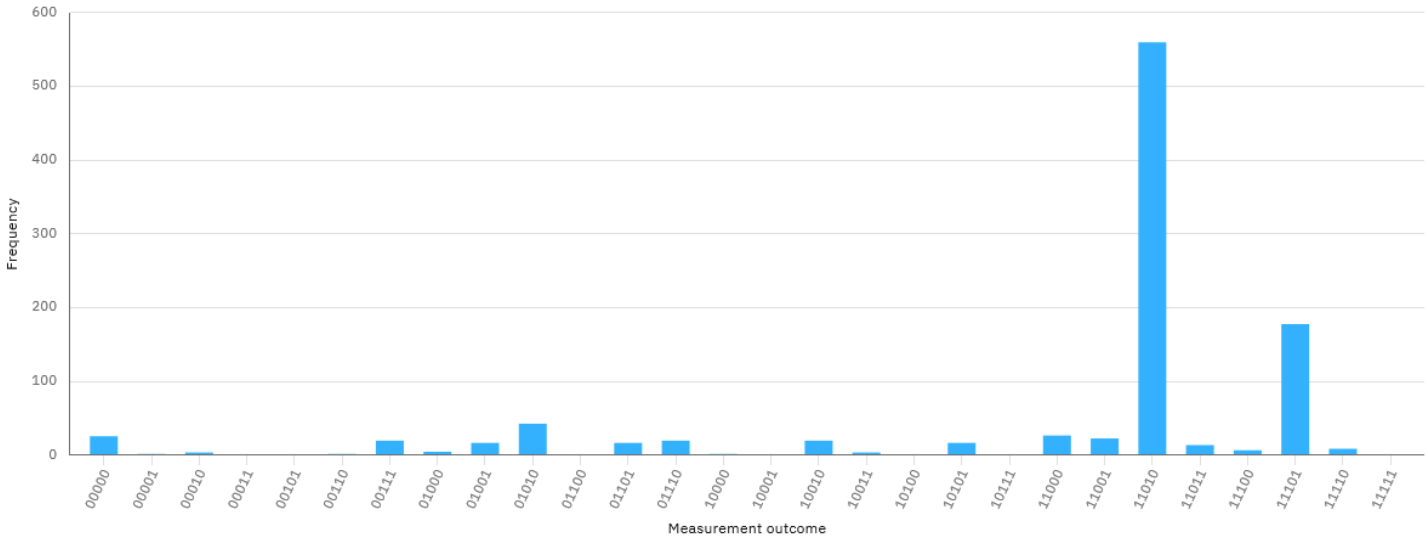


FIGURE 38. Measurement results on IBMQ-Lima.

`Q-Composer circuit` run on IBMQ-Lima; 80% accuracy in detecting the error, 90.11% accuracy in "favourable results" (Measurements "11010" and "11101") after error detection.

4.3. **Quantum phaseflip channel correction.**

In this channel, with a probability p the phase of the qubit gets flipped, thus $|\psi\rangle = \alpha|0\rangle + \beta \mid 1\rangle \rightarrow (1-p)(\alpha|0\rangle + \beta|1\rangle) + p(\alpha|0\rangle - \beta|1\rangle)$.
Just as the bitflip channel flips a 0 to a 1, the phaseflip channel flips $|+\rangle$ to $|-\rangle$. The only modifications to the bitflip code to correct phaseflips is to encode information into the above Hadamard basis states, which can be done by a Hadamard gate. Once the error is detected, the inverse gate (which is the Hadamard itself) can be applied to get back error-affected information in our computational basis states and apply the required correction as was for the bitflip error.

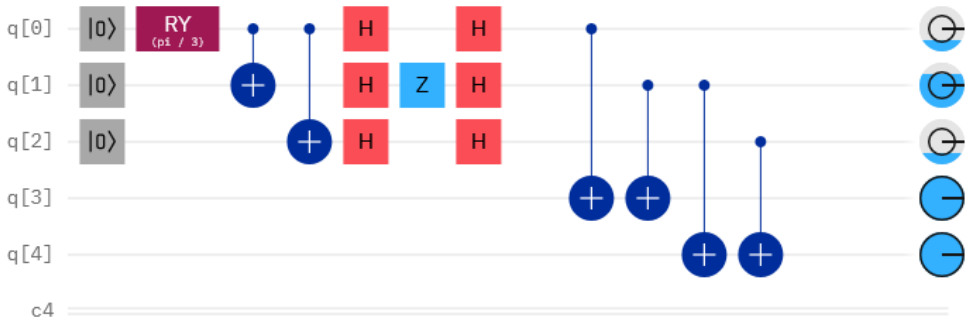


FIGURE 39. Quantum phaseflip correction circuit, with $|\psi\rangle = \sqrt{3}\,/\,2|0\rangle + 1\,/\,2\,|1\rangle$ and flip in 2nd qubit.

Here, q[0] is the intended qubit (initialized as $\sqrt{3}\,/\,2|0\rangle + 1\,/\,2|1\rangle$) Which is converted to $\sqrt{3}\,/\,2|000\rangle + 1\,/\,2|111\rangle$).
After the Hadamard transform, we get $\sqrt{3}\,/\,2|{+}{+}{+}\rangle + 1\,/\,2|{-}{-}{-}\rangle$) Let us assume that the middle qubit is phase flipped by the channel, due to which we get $\sqrt{3}\,/\,2|{+}{-}{+}\rangle + 1\,/\,2|{-}{+}{-}\rangle$) The Hadamard transform again gives us $\sqrt{3}\,/\,2|010\rangle + 1\,/\,2|101\rangle$)
This flipping changes the state of the syndrome qubits q[3] and q[4] which are controlled by CNOT gates. Since our syndrome qubits read 11, we determine that the middle qubit was affected and the X gate can be applied to get the intended data (through action of the syndrome bits itself.)

q[3]	q[4]	Result
0	0	No flip
0	1	Third qubit phase flipped
1	0	First qubit phase flipped
1	1	Second qubit phase flipped

This circuit can only handle one phaseflip, as it works on the principle of majority voting.

4.3.1. *Examples and experimental verification.*

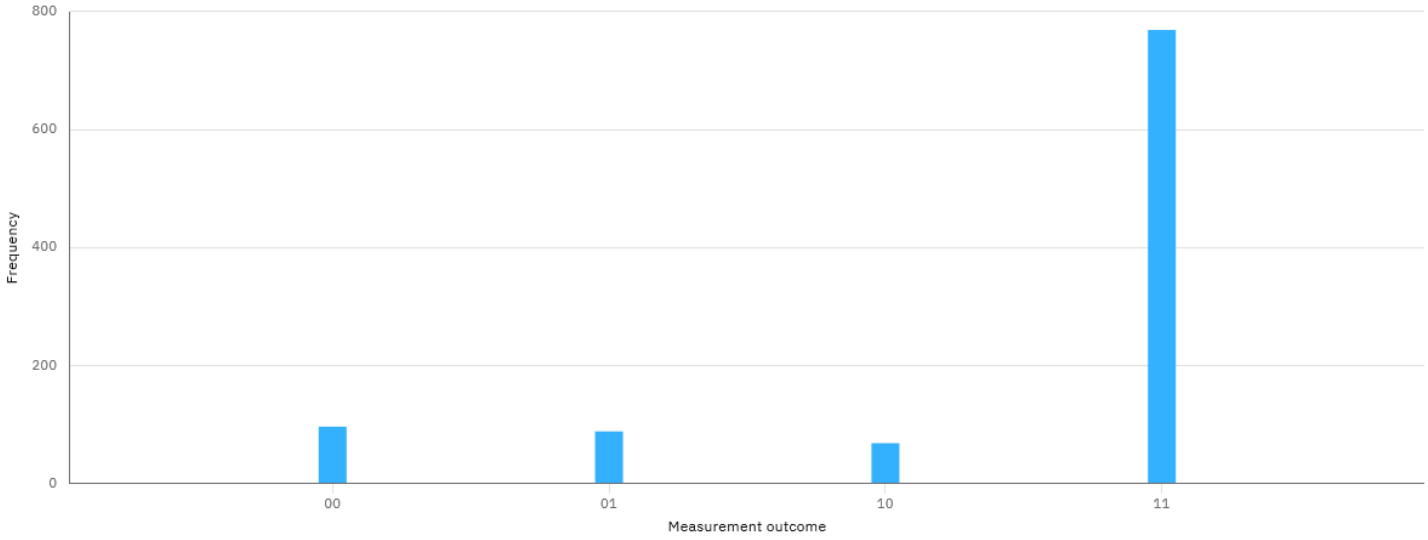


FIGURE 40. "11":769

Running the `Q-Composer` circuit on IBMQ-Lima and checking only for accuracy in error detection gives 75% accuracy.

4.4. **Shor's Code.**

In the repetition code setup, if we are only concerned with the bitflip error in a particular qubit, we can correct the error with 3 qubits.

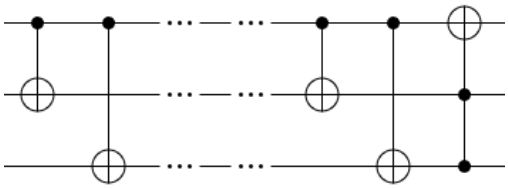


FIGURE 41. Bitflip correction for 1st qubit.

The input into this section is $|00\psi\rangle$.If any bitflips occur in the area demarcated through the spacers, the 1st qubit is unaffected. This sub-routine is used for error correction in Shor's code. Shor's Code correct bit flip and phase flip errors (and can be shown to handle an arbitrary error as well) by creating a hierarchy inside the circuit.

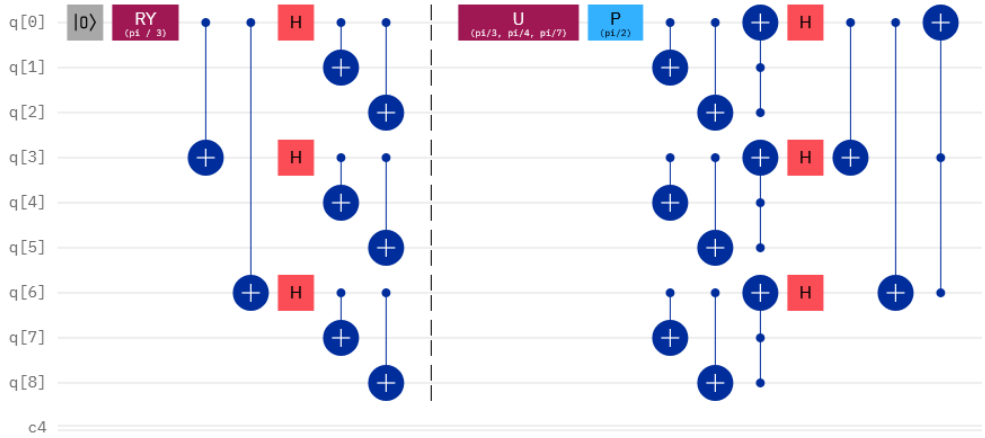


FIGURE 42. Circuit for Shor's code with error in first qubit denoted by U,P.

Circuit on Quirk and circuit on Q-Composer.

The inner layer corrects bit flips, while the outer layer corrects phase flips. The inner layer has three groups, with each group having a primary qubit and two syndrome qubits. In the circuit above, the inner groups are:

$$(q[0], q[1], q[2]),$$

$$(q[3], q[4], q[5]),$$

$$(q[6], q[7], q[8])$$

The outer layer consists of the primary qubits of the groups:

$$q[0], q[3], q[6]$$

The outer layer corrects phase flips. (These are the groups qubits on which the Hadamard has been applied.) For our purposes here, we have set the original information carrying qubit to be $\alpha|0\rangle + \beta|1\rangle$.

After the application of the CNOT gates and the Hadamard (and the second set of CNOT gates), the state we have obtained (marked at the divider) is:

$$\alpha(|000\rangle + |111\rangle)(|000\rangle + |111\rangle)(|000\rangle + |111\rangle) \,/\, 2\sqrt{2}$$

$$+\beta(|000\rangle - |111\rangle)(|000\rangle - |111\rangle)(|000\rangle - |111\rangle) \,/\, 2\sqrt{2}$$

4.4.1. *Bitflips:*

. **Bit flip of 1st qubit of inner layer's 1st group:**

$$\alpha(|100\rangle + |011\rangle)(|000\rangle + |111\rangle)(|000\rangle + |111\rangle) \,/\, 2\sqrt{2}$$

$$+\beta(|100\rangle - |011\rangle)(|000\rangle - |111\rangle)(|000\rangle - |111\rangle) \,/\, 2\sqrt{2}$$

When the CNOT gates are applied concurrently in the three groups, it flips the two syndrome bits associated to the first group of the inner layer, due to which:

$$\alpha(|111\rangle + |011\rangle)(|000\rangle + |100\rangle)(|000\rangle + |100\rangle) \,/\, 2\sqrt{2}$$

$$+\beta(|111\rangle - |011\rangle)(|000\rangle - |100\rangle)(|000\rangle - |100\rangle) \,/\, 2\sqrt{2}$$

As seen, in the groups where there is no bitflip the CNOT gates do the job of "turning off" the syndrome qubits associated to that group.
The code rectifies the error with the Toffoli gate:

$$\alpha(|011\rangle + |111\rangle)(|000\rangle + |100\rangle)(|000\rangle + |100\rangle) \,/\, 2\sqrt{2}$$

$$+\beta(|011\rangle - |111\rangle)(|000\rangle - |100\rangle)(|000\rangle - |100\rangle) \,/\, 2\sqrt{2}$$

Since Hadamard gate only acts on the primary qubits which we can see are not corrupted, any further calculations are not affected by the bitflip.

2nd qubit of inner layer's first group:

$$\alpha(|010\rangle + |101\rangle)(|000\rangle + |111\rangle)(|000\rangle + |111\rangle) \,/\, 2\sqrt{2}$$

$$+\beta(|010\rangle - |101\rangle)(|000\rangle - |111\rangle)(|000\rangle - |111\rangle) \,/\, 2\sqrt{2}$$

After CNOT and Toffoli gate:

$$\alpha(|010\rangle + |110\rangle)(|000\rangle + |100\rangle)(|000\rangle + |100\rangle) \,/\, 2\sqrt{2}$$

$$+\beta(|010\rangle - |110\rangle)(|000\rangle - |100\rangle)(|000\rangle - |100\rangle) \,/\, 2\sqrt{2}$$

As seen, the bitflip of the second qubit does not affect the primary qubit of their inner group.
Thus, a single bitflip in an inner group will not have any effect. However, more than one bitflip within an inner group will cause errors in the final reading.
For a single flip within a inner group,

S1	S2	Result
0	0	No flip
0	1	S2 was flipped
1	0	S1 was flipped
1	1	Primary qubit flipped

Shor's Code can deal with one bit flip in one inner group, and can deal with a bitflip in more than one inner group.

4.4.2. *Phaseflips*
. The phase flips are handled by the outer layer.
Phase flip of first qubit in 1st inner group:

$$\alpha(|000\rangle - |111\rangle)(|000\rangle + |111\rangle)(|000\rangle + |111\rangle) \,/\, 2\sqrt{2}$$

$$+\beta(|000\rangle + |111\rangle)(|000\rangle - |111\rangle)(|000\rangle - |111\rangle) \,/\, 2\sqrt{2}$$

(We obtain this state since $|\psi\rangle| - \psi\rangle|\psi\rangle \rightarrow -|\psi\psi\psi\rangle$.)

After CNOT and Toffoli gates:

$$\alpha(|000\rangle - |100\rangle)(|000\rangle + |100\rangle)(|000\rangle + |100\rangle) \,/\, 2\sqrt{2}$$

$$+\beta(|000\rangle + |100\rangle)(|000\rangle - |100\rangle)(|000\rangle - |100\rangle) \,/\, 2\sqrt{2}$$

On applying the Hadamard gate on the primary qubits (1st, 4th and 7th qubit):

$$\alpha(|{+}00\rangle - |-00\rangle)(|{+}00\rangle + |-00\rangle)(|{+}00\rangle + |-00\rangle) \,/\, 2\sqrt{2}$$
$$+\beta(|{+}00\rangle + |-00\rangle)(|{+}00\rangle - |-00\rangle)(|{+}00\rangle - |-00\rangle) \,/\, 2\sqrt{2}$$

It is clear to see that even if a bitflip had occurred in any of the inner groups (or even on the same qubit), it would have been corrected prior to the application of the Hadamard gate and not affected the state of the primary qubits.
For convenience, we neglect the syndrome bits of each inner group and simplify. (We are able to ignore the syndrome qubits as their measurement does not affect the primary qubits in this case.)

$$\alpha(|+\rangle - |-\rangle)(|+\rangle + |-\rangle)(|+\rangle + |-\rangle) \,/\, 2\sqrt{2}$$
$$+\beta(|+\rangle + |-\rangle)(|+\rangle - |-\rangle)(|+\rangle - |-\rangle) \,/\, 2\sqrt{2}$$
$$\Rightarrow \frac{\alpha(|\ 1\rangle)(0\rangle)(|\ 0\rangle) + \beta(|\ 0\rangle)(|\ 1\rangle)(|\ 1\rangle)}{2\sqrt{2}}$$

As seen above, the phase flip in the first inner group results in the bitflip of the first primary qubit. This phase flip can be corrected by applying the gate combination that defined in the starting of this section on Shor's code; treating the second and third primary qubits as the syndrome qubits to the first primary qubit.

S1 (q[3])	S2 (q[6])	Result
0	0	No flip
0	1	Phase flip in 3rd inner group
1	0	Phase flip in 2nd inner group
1	1	Phase flip in 1st inner group

Unlike the bitflips, Shor's code cannot deal with phaseflip in more than one inner group. For example, if there were phase flips in the first and third inner groups, then:

$$(\alpha(|\ 1\rangle)(|1\rangle)(|\ 0\rangle) + \beta(|\ 0\rangle)(|\ 1\rangle)(|\ 0\rangle)) \,/\, 2\sqrt{2}$$

Not only has our resultant qubit been bitflipped, but also the circuit indicates that the phaseflip has occurred in the 2nd inner group.

If a phase flip and bit flip occurred on the same qubit (and only one qubit was affected), the syndrome qubits would indicate where the bitflip and phase flip occurred.

4.4.3. *Arbitrary error*

.

It can be proven that Shor's code can correct arbitrary error in a qubit, such as the rotation of the qubit's statevector about an arbitrary axis by an arbitrary degree. Let us assume that the error does not cause the collapse of the qubit's state itself. Following this assumption, we can say that the error is equivalent to the

action of an arbitrary gate U. For a four dimensional complex vector, we can make the following adjustment in notation:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \Rightarrow \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix}; a, b, c, d \in \mathbb{C}$$

This allows us to see that every unitary matrix can be defined as a unit vector in this space. We can show that below set of unitary matrices span the 4-dimensional complex vector space.

$$I, X, Z \; and \; XZ \;\; \text{span } \mathbb{C}^4$$

Thus, the arbitrary gate is equivalent to the linear combination of the elementary gates with some probability amplitude. (Here, sum of squares of coefficients=1.)

The result above shows that the effect of an arbitrary error/gate acting on the qubit is equivalent to a superposition of the qubit's state after the action of elementary gates I,X,Z and/or XZ on it.

Measurement of the other 8 qubits will collapse the error that has occurred on the first qubit to that of an elementary gate's action and it is clear that measuring the 8 qubits isn't necessary for the error to be corrected in the first place. Since the code can deal with the action of elementary gates, due to the reasoning above it can deal with the action of arbitrary gates as well.
For example:
Let the error be the equivalent of the action of the Hadamard gate.

$$H = \frac{1}{\sqrt{2}}(X + Z)$$

This implies that on the measurement of the other 8 qubits, we will see a 50% chance of a bitflip having acted, or 50% chance of phaseflip having acted.
Let the first qubit's state be $\alpha|0\rangle + \beta|1\rangle$. As before, the concatenated state is obtained:

$$\alpha(|000\rangle + |111\rangle)(|000\rangle + |111\rangle)(|000\rangle + |111\rangle) \,/\, 2\sqrt{2}$$
$$+\beta(|000\rangle - |111\rangle)(|000\rangle - |111\rangle)(|000\rangle - |111\rangle) \,/\, 2\sqrt{2}$$

Ignoring the 2nd and the 3rd group, and solely focusing on the first group's expression, on applying the Hadamard (which serves as the error) on the first group's first qubit gives:

$$\alpha\left(\frac{|000\rangle + |100\rangle + |011\rangle - |111\rangle}{\sqrt{2}}\right) + \beta\left(\frac{|000\rangle + |100\rangle - |011\rangle + |111\rangle}{\sqrt{2}}\right)$$

After the CNOT gates:

$$\alpha\left(\frac{|000\rangle + |111\rangle + |011\rangle - |100\rangle}{\sqrt{2}}\right) + \beta\left(\frac{|000\rangle + |111\rangle - |011\rangle + |100\rangle}{\sqrt{2}}\right)$$

After the Toffoli gate (that corrects the bitflip if triggered by syndrome qubits):

$$\alpha\left(\frac{|000\rangle + |011\rangle + |111\rangle - |100\rangle}{\sqrt{2}}\right) + \beta\left(\frac{|000\rangle + |011\rangle - |111\rangle + |100\rangle}{\sqrt{2}}\right)$$
$$\Rightarrow \alpha(|-00\rangle + |+11\rangle) + \beta(|+00\rangle + |-11\rangle)$$

As seen, if the error syndrome on measurement gives 11, then the net result of the Hadamard gate would be equivalent to a bitflip. If 00 is measured, then the net

effect of the Hadamard gate would be indistinguishable from a phase flip.
The Hadamard gate's action presents itself as a superposition of the action of the X and Z with equal probability, and the measurement collapses it to one of the above.
Thus, even without measurement it can deal decompose the Hadamard gate error as a superposition of a bitflip error and a phaseflip error.

Explicit calculations for arbitrary error correction in Shor's code:

An arbitrary gate U's action on a qubit can be described as:

$$U|0\rangle \to a_0|0\rangle + b_0|1\rangle; U|1\rangle \to a_1|0\rangle + b_1|1\rangle$$

Let this error act on the first qubit (that is "protected" by the algorithm):

$$\alpha(U|0\rangle|00\rangle + U|1\rangle|11\rangle)(|000\rangle + |111\rangle)(|000\rangle + |111\rangle) \,/\, 2\sqrt{2}$$
$$+\beta(U|0\rangle|00\rangle - U|1\rangle|11\rangle)(|000\rangle - |111\rangle)(|000\rangle - |111\rangle) \,/\, 2\sqrt{2}$$
$$\Rightarrow$$
$$\alpha(a_0|000\rangle + b_0|100\rangle + a_1|011\rangle + b_1|111\rangle)(|000\rangle + |111\rangle)(|000\rangle + |111\rangle) \,/\, 2\sqrt{2}$$
$$+\beta(a_0|000\rangle + b_0|100\rangle - a_1|011\rangle - b_1|111\rangle)(|000\rangle - |111\rangle)(|000\rangle - |111\rangle) \,/\, 2\sqrt{2}$$

After CNOT and Toffoli gate,

$$\alpha(a_0|000\rangle + b_0|011\rangle + a_1|111\rangle + b_1|100\rangle)(|000\rangle + |100\rangle)(|000\rangle + |100\rangle) \,/\, 2\sqrt{2}$$
$$+\beta(a_0|000\rangle + b_0|011\rangle - a_1|111\rangle - b_1|100\rangle)(|000\rangle - |100\rangle)(|000\rangle - |100\rangle) \,/\, 2\sqrt{2}$$

After Hadamard on primary qubits (and neglecting syndrome bits of inner groups):

$$\alpha(a_0|+\rangle + b_0|+\rangle + a_1|-\rangle + b_1|-\rangle)(|+\rangle + |-\rangle)(|+\rangle + |-\rangle) \,/\, 2\sqrt{2}$$
$$+\beta(a_0|+\rangle + b_0|+\rangle - a_1|-\rangle - b_1|-\rangle)(|+\rangle - |-\rangle)(|+\rangle - |-\rangle) \,/\, 2\sqrt{2}$$
$$\Rightarrow$$
$$\alpha(a_0|+\rangle + b_0|+\rangle + a_1|-\rangle + b_1|-\rangle)(|\,0\rangle)(|\,0\rangle) \,/\, 2\sqrt{2}$$
$$+\beta(a_0|+\rangle + b_0|+\rangle - a_1|-\rangle - b_1|-\rangle)(|\,1\rangle)(|\,1\rangle) \,/\, 2\sqrt{2}$$
$$\Rightarrow$$
$$\alpha\left[\left(\frac{[a_0+b_0+a_1+b_1]|0\rangle + [a_0+b_0-a_1-b_1]|1\rangle}{\sqrt{2}}\right)(|\,0\rangle)(|\,0\rangle)\right] \,/\, 2\sqrt{2}$$
$$+\beta\left[\left(\frac{[a_0+b_0-a_1-b_1]|0\rangle + [a_0+b_0+a_1+b_1]|1\rangle}{\sqrt{2}}\right)(|\,1\rangle)(|\,1\rangle)\right] \,/\, 2\sqrt{2}$$

After the final set of CNOT and Toffoli operations:

$$\alpha\left[\left(\frac{[a_0+b_0+a_1+b_1] \mid 0\rangle}{\sqrt{2}}\right)(|\,0\rangle)(|\,0\rangle) + \left(\frac{[a_0+b_0-a_1-b_1] \mid 0\rangle}{\sqrt{2}}\right)(|\,1\rangle)(|\,1\rangle)\right] \,/\, 2\sqrt{2}$$
$$+\beta\left[\left(\tfrac{[a_0+b_0-a_1-b_1] \mid 1\rangle}{\sqrt{2}}\right)(|\,1\rangle)(|\,1\rangle) + \left(\tfrac{[a_0+b_0+a_1+b_1] \mid 1\rangle}{\sqrt{2}}\right)(|\,0\rangle)(|\,0\rangle)\right] \,/\, 2\sqrt{2}$$

This can be regrouped as:

$$\frac{a_0+b_0+a_1+b_1}{\sqrt{2}}(\alpha|0\rangle + \beta|1\rangle) \mid 00\rangle + \frac{a_0+b_0-a_1-b_1}{\sqrt{2}}(\alpha|0\rangle + \beta|1\rangle)|11\rangle$$

Which results in:

$$(\alpha|0\rangle + \beta|1\rangle)\left(\frac{a_0 + b_0 + a_1 + b_1}{\sqrt{2}} \mid 00\rangle + \frac{a_0 + b_0 - a_1 - b_1}{\sqrt{2}} \mid 11\rangle\right)$$

In the last two steps, only matrix manipulations have taken place, and its "physical" interpretation is that the measurement of the other two primary qubits and all the remaining syndrome qubits tells us what error has been measured and where.

With this, the proof is complete.

5. Quantum Arithmetic Algorithms

5.1. **Addition (without QFT)**

. The most common algorithm for classical addition is the straightforward add-and-carry method; add the digits column wise, and shift any carry over bits to the left column.
In a full adder, we keep track of the sum as well as the carry over. Inputs are a, b, carry over is c. This can be implemented directly into quantum computers[46]:

- Take in two n-bit inputs as we store them in $2n$ qubits.
- Define a n-qubit register to serve as the carry over.
- Define a $n + 1$ bit classical register to store the output.
- Starting from the least significant digit of the numbers, calculate the carry value and transfer it to the carry qubit to the left of the current carry qubit.
- Add the value of the qubits associated to that digit place for a and c to b.
- Repeat the process for the digit to the left.

In this algorithm, we handle the carry over bit as well as the sum bit in the same manner as we do classically.

5.1.1. *Carry digit*

. The carry digit's value can be defined as follows: (Here, **next carry digit** is the carry digit associated with the digits that are the next significant/to the left of the current digit being analysed.)

c	a	b	Next carry digit
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

The above table is implemented as follows:

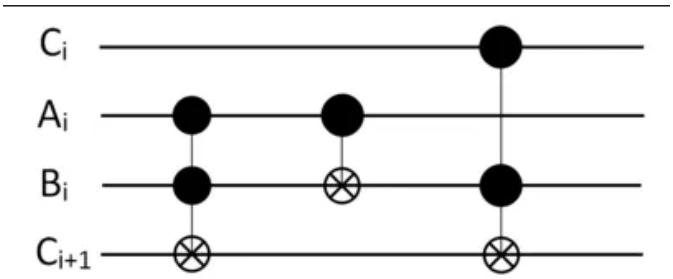


FIGURE 43. Circuit for carry over digit.

Flipping b_i is necessary in order to prevent triggering a carry over flip once it has already been done. Since the set of qubits that store b are used to store the output as well, the flipping of b_i serves as the addition of a_i mod 2 to it. Thus, all that is left to be done is to perform the modular addition of the corresponding carry over bit with b_i.

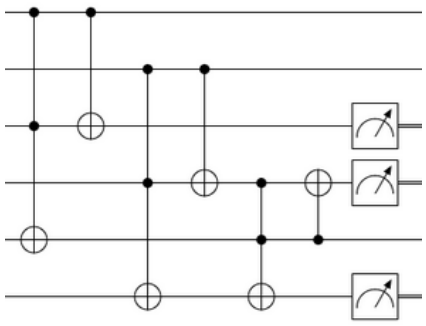


FIGURE 44. Circuit for two qubit full adder.

This is for a two qubit full adder. As seen, we use the carry qubit of the 0th qubit in the calculations for the 1st digit. In the end, we measure b and we measure the most significant carry qubit as it serves as the most significant digit of the output. We can also use two carry qubits instead of n, by constantly resetting them and reusing it throughout the operation. To prevent modular addition, the final carry over qubit is measured as it would serve as the most significant digit (in base 2).

5.1.2. *Examples and experimental verification*
. `Three qubit full adder on Quirk` and `Three qubit full adder on Q-Composer`.

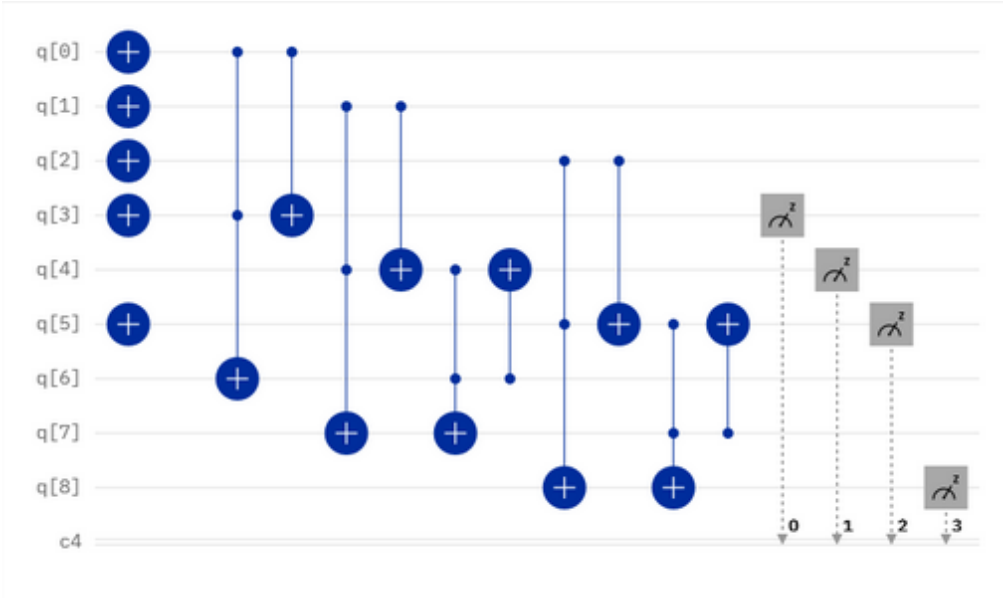


FIGURE 45. 3 Qubit full adder with $a = 11, b = 11$.

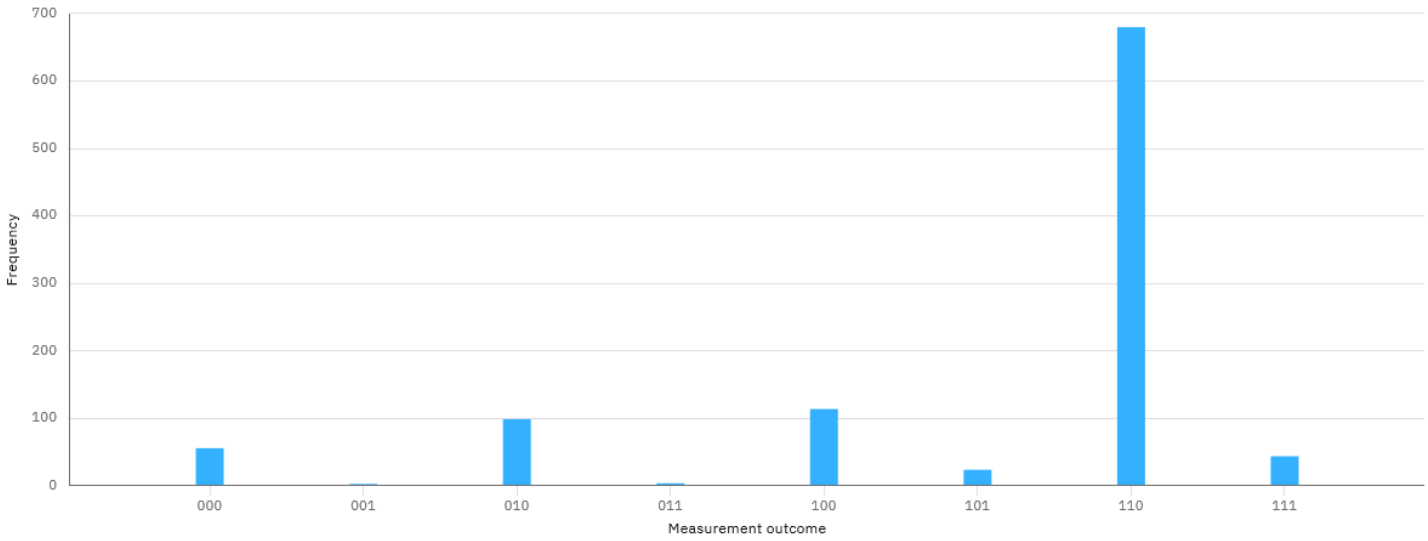


FIGURE 46. "110":680

Run on IBMQ-Jakarta, an accuracy of 66.4% is measured.

5.2. **QFT arithmetic algorithms.**

5.2.1. *Addition and subtraction*

. The QFT-based methods stated above are pure emulations of the classical methods, and provides no advantages. Similarly, other classical algorithms of addition[47] do not provide any significant advantages when applied on quantum computers.

Thomas Draper's algorithm[48, 49] uses QFT for addition by shifting one number into the Fourier basis and performing controlled rotations on it through the second number's digits. It saves the use of carry bits but does not provide any significant boost in runtime, and can only perform mod 2^n addition for two n-bit inputs.

However, the mod 2^n addition problem can be mitigated easily. If the inputs are a and b, we pass $|0a\rangle$ and $|b\rangle$ as input, and perform our addition (and QFT) on $|0a\rangle$. Since the most significant qubit's state is $|0\rangle$, it acts as a failsafe if the output cannot be contained in n bits (and thus requires n+1 bits).

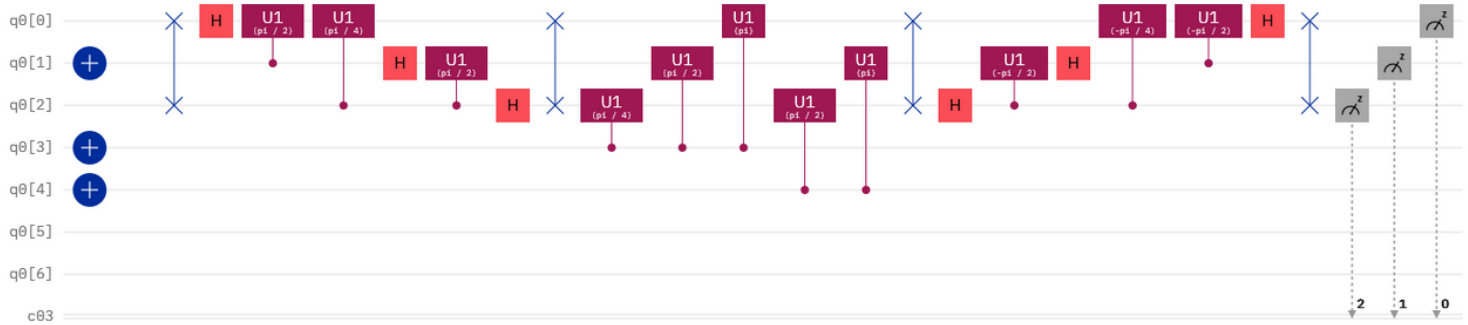


FIGURE 47. Two qubit QFT full adder with $a = 10, b = 11$

This method provides versatility as subtraction can be performed by the same circuit by simply negating the angle of rotation.

Lastly, as the numbers of qubits required for the operation increase, the rotation angle reduces, making it harder to prevent errors. This is mitigated by converting the QFT adder to a Toffoli-based adder.[50]

5.2.1.1. *Examples and experimental verification*

.

Two qubit QFT adder in Quirk and two qubit QFT full adder in Q-Composer

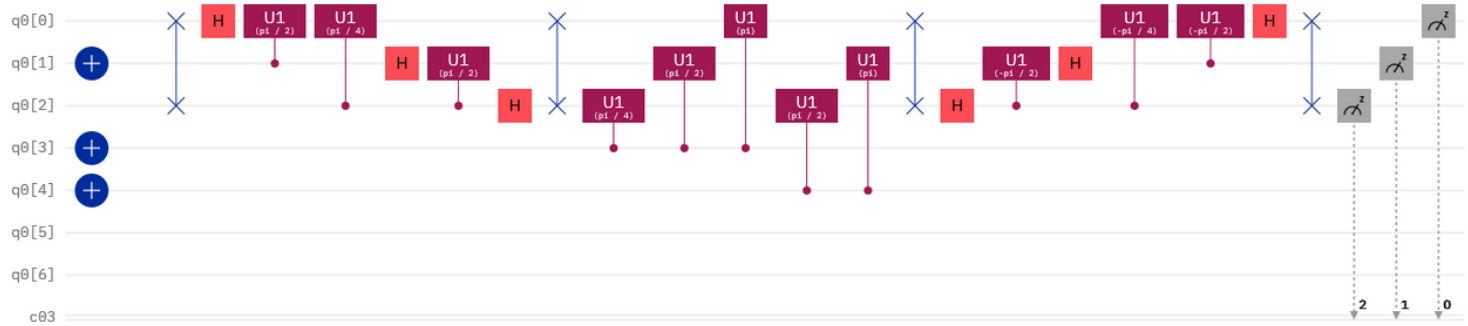


FIGURE 48. Two qubit QFT full adder with $a = 10, b = 11$

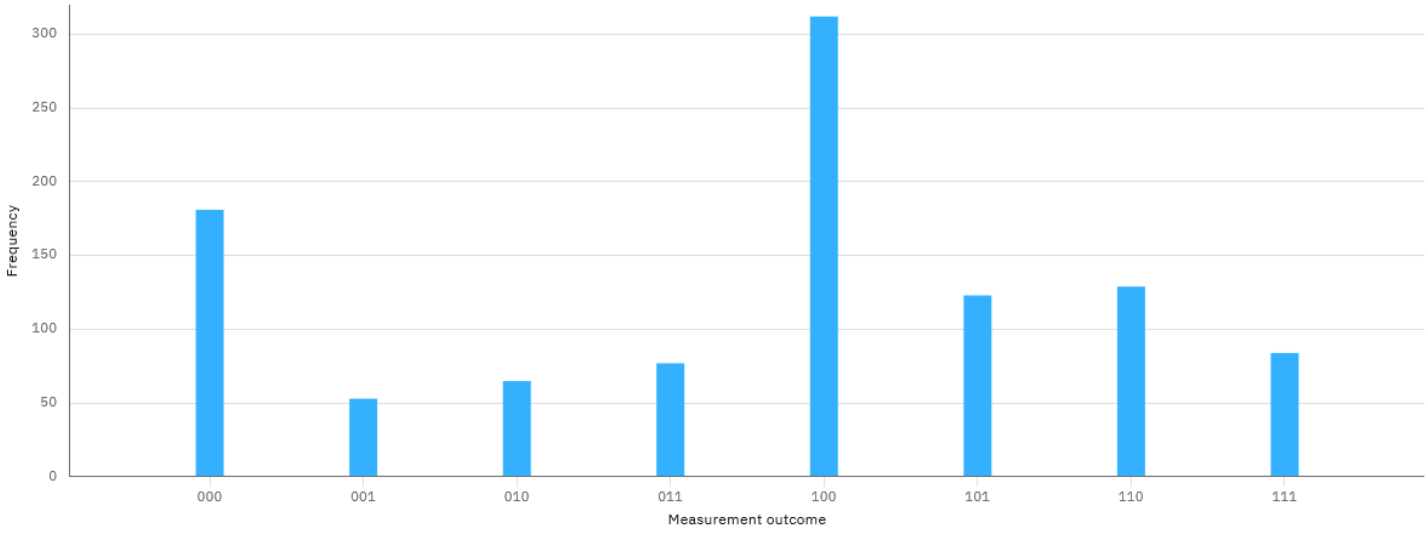


FIGURE 49. "100":312

Run on IBMQ-Lima, an accuracy of 30.4% is observed.

5.2.2. *Multiplication and division.*

Division:
Quantum algorithms for integer division is under active research(See[51] for methods and further references). An algorithm devised by F.Orts et al.[52] provides a unique method of approaching this problem by using Grover's search algorithm.
Multiplication:
There exist several quantum algorithms for multiplication[53, 54], but for this report I will be focusing on the method that shows a direct application of QFT.[55]

In the paper by Lidia Ruiz-Perez and Juan Carlos Garcia-Escartin, they propose the concatenation of controlled weighted sum blocks:

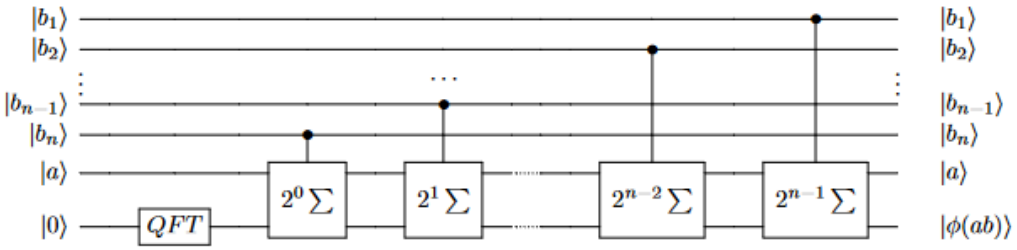


FIGURE 50. Proposed QFT multiplier circuit.

The result will be a 2n-qubit register encoding the number a · b.

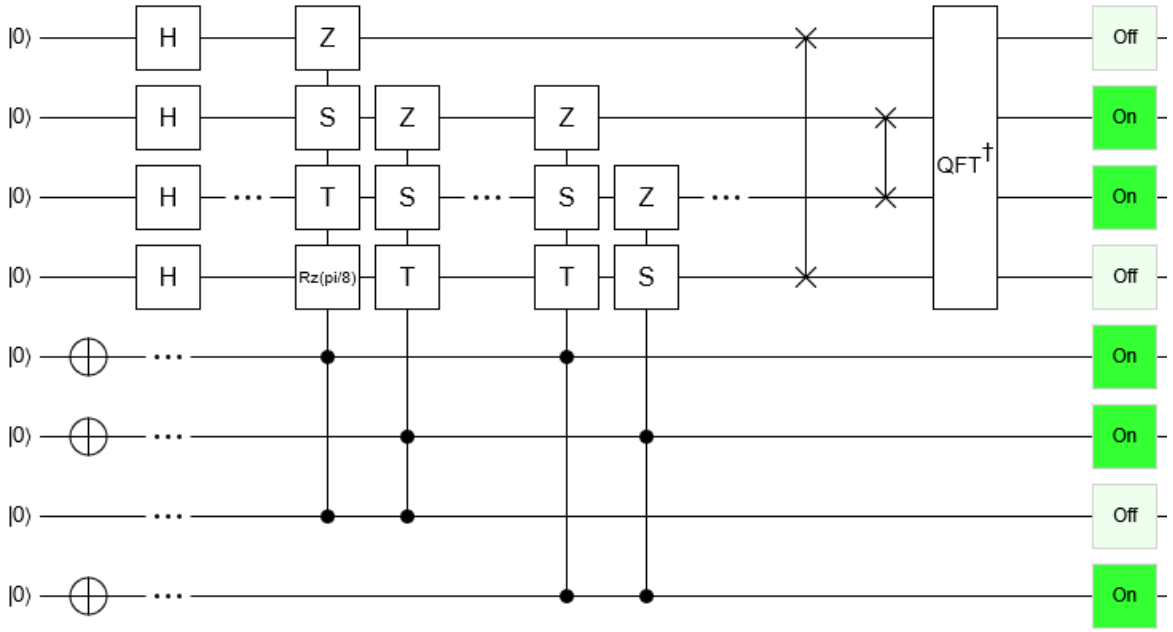


FIGURE 51. Two digit quantum multiplier, $a = 11, b = 10$.

(SWAP gates are for maintaining little endian notation.) In the circuit, the qubits that will store the product are first "initialized" through a Hadamard transform. Then (taking the first set of qubits "below" the product qubit set to be a) controlled additions are performed and are scaled according to the significance of the digits. For example, $a = 11, b = 10$. Product=$11.10 = 0 * (1^1 + 2^1) + 1 * (1^1 + 2^1) = 3$ We obtain the result by performing Inverse QFT.

6. Further discussions and observations

The analysis of an arbitrary error's correction in Shor's code (which is usually covered in a casual manner) provided insight into how the error can be interpreted as a superposition of "easily treated" errors. Such analysis may prove helpful for other problems as well.

Despite Shor's algorithm being "efficient", there has been extreme difficulty in implementing it, thus the factorization problem is open to attack from other methods. Following this theme, string matching quantum algorithms may prove fruitful in solving unsolved cases (for specific sets of tiles) of Post's Correspondence Problem. There seems to be very less analysis on the probability of conclusively determining the hidden value, when accounting for noisy systems.

Lastly, despite rapid developments in the field of quantum algorithms, the quantum computers currently are not stable enough to provide high accuracy answers for several algorithms.

7. Conclusion

In this report, I have analysed quantum algorithms, quantum error correction procedures, and covered quantum algorithms for arithmetic. Along with this, I have covered the feasibility of running these algorithms on the currently available public quantum computers.

References

[1] K. Efstathiou, and M. Efstathiou, "Celestial Gearbox," *Mech. Eng.*, vol. 140, no. 9, pp. 31–35, Sep. 2018, doi: 10.1115/1.2018-SEP1. Accessed: Jul. 21, 2023. [Online]. Available: https://asmedigitalcollection.asme.org/memagazineselect/article/140/09/31/366620/Celestial-GearboxThe-Oldest-Known-Computer-is-a

[2] P. Benioff, "The computer as a physical system: a microscopic quantum mechanical hamiltonian model of computers as represented by turing machines," *J. Statistical Phys.*, vol. 22, no. 5, p. 563, May 1980. [Online]. Available: http://link.springer.com/10.1007/BF01011339

[3] P. Benioff, "Quantum mechanical hamiltonian models of turing machines," *J. Statistical Phys.*, vol. 29, no. 3, p. 515, Nov. 1982. [Online]. Available: http://link.springer.com/10.1007/BF01342185

[4] R. P. Feynman, "Simulating physics with computers," *Int. J. Theor. Phys.*, vol. 21, no. 6–7, p. 467, Jun. 1982. [Online]. Available: http://link.springer.com/10.1007/BF02650179

[5] *Proc. Roy. Soc. London. A. Math. Physical Sciences*, vol. 400, no. 1818, p. 97, Jul. 1985. [Online]. Available: https://royalsocietypublishing.org/doi/10.1098/rspa.1985.0070

[6] "Rapid solution of problems by quantum computation," *Proc. Roy. Soc. London. Ser. A: Math. Physical Sciences*, vol. 439, no. 1907, p. 553, Dec. 1992. [Online]. Available: https://royalsocietypublishing.org/doi/10.1098/rspa.1992.0167

[7] E. Bernstein, and U. Vazirani, "Quantum complexity theory," in *Proc. Twenty-Fifth Annu. ACM Symp. Theory Comput. - STOC '93*, San Diego, California, United States, 1993, p. 11. [Online]. Available: http://portal.acm.org/citation.cfm?doid=167088.167097

[8] D. R. Simon, "On the power of quantum computation," *SIAM J. Comput.*, vol. 26, no. 5, p. 1474, Oct. 1997. [Online]. Available: http://epubs.siam.org/doi/10.1137/S0097539796298637

[9] D. Coppersmith, "An approximate fourier transform useful in quantum factoring," Jan. 2002. [Online]. Available: http://arxiv.org/abs/quant-ph/0201067

[10] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM J. Comput.*, vol. 26, no. 5, p. 1484, Oct. 1997. [Online]. Available: http://arxiv.org/abs/quant-ph/9508027

[11] L. K. Grover, "A fast quantum mechanical algorithm for database search," Nov. 1996. [Online]. Available: http://arxiv.org/abs/quant-ph/9605043

[12] A. W. Harrow, A. Hassidim, and S. Lloyd, "Quantum algorithm for solving linear systems of equations," *Physical Rev. Lett.*, vol. 103, no. 15, p. 150502, Oct. 2009. [Online]. Available: http://arxiv.org/abs/0811.3171

[13] H. J. Morrell Jr, A. Zaman, and H. Y. Wong, "Step-by-step hhl algorithm walkthrough to enhance the understanding of critical quantum computing concepts," Mar. 2023. [Online]. Available: http://arxiv.org/abs/2108.09004

[14] K. Kadian, S. Garhwal, and A. Kumar, "Quantum walk and its application domains: a systematic review," *Comput. Sci. Rev.*, vol. 41, p. 100419, Aug. 2021. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1574013721000599

[15] A. Peruzzo, J. McClean, et al., "A variational eigenvalue solver on a quantum processor," *Nature Commun.*, vol. 5, no. 1, p. 4213, Jul. 2014. [Online]. Available: http://arxiv.org/abs/1304.3061

[16] R. Bavdekar, E. J. Chopde, et al., "Post quantum cryptography: techniques, challenges, standardization, and directions for future research," Feb. 2022. [Online]. Available: http://arxiv.org/abs/2202.02826

[17] M. A. Nielsen, and I. L. Chuang, *Quantum Computation and Quantum Information*, 10th anniversary ed, Cambridge ; New York: Cambridge University Press, 2010.

[18] G. Craig, "Ultradense coding would allow ftl signalling," May 2016. [Online]. Available: https://algassert.com/2016/05/29/ultra-dense-coding-allows-ftl.html

[19] C. Richard, "Quantum information processing — quantum algorithms i," 2021. [Online]. Available: http://cleve.iqc.uwaterloo.ca/resources/QIC-710-F21/Qic710QuantumAlgorithmsPart1.pdf

[20] S. Dutta, "Why does the "phase kickback" mechanism work in the quantum phase estimation algorithm?," 2019. [Online]. Available: https://quantumcomputing.stackexchange.com/q/2565

[21] "Bernstein-vazirani algorithm - intro to quantum software development." [Online]. Available: https://stem.mitre.org/quantum/quantum-algorithms/bernstein-vazirani-algorithm.html

[22] C. Gidney, "Answer to "oracle construction for grover's algorithm"," 2017. [Online]. Available: https://cstheory.stackexchange.com/a/38551

[23] "Grover's algorithm, an intuitive look — quantum computing news and features_2021, " 2021. [Online]. Available: https://quantumzeitgeist.com/grovers-algorithm-an-intuitive-look/

[24] "Grover's algorithm - intro to quantum software development." [Online]. Available: https://stem.mitre.org/quantum/quantum-algorithms/grovers-algorithm.html

[25] G. L. Long, "Grover algorithm with zero theoretical failure rate," *Physical Rev.*, vol. 64, no. 2, p. 22307, Jul. 2001. [Online]. Available: http://arxiv.org/abs/quant-ph/0106071

[26] T. Roy, L. Jiang, and D. I. Schuster, "Deterministic grover search with a restricted oracle," *Physical Rev. Res.*, vol. 4, no. 2, Apr. 2022. [Online]. Available: http://arxiv.org/abs/2201.00091

[27] C. Zalka, "Grover's quantum searching algorithm is optimal," *Physical Rev.*, vol. 60, no. 4, p. 2746, Oct. 1999. [Online]. Available: http://arxiv.org/abs/quant-ph/9711070

[28] A. Mandviwalla, K. Ohshiro, and B. Ji, "Implementing grover's algorithm on the ibm quantum computers," in *2018 IEEE Int. Conf. Big Data (Big Data)*, Seattle, WA, USA, Dec. 2018, p. 2531. [Online]. Available: https://ieeexplore.ieee.org/document/8622457/

[29] C. Richard, "Classical lower bound for simon's problem," 2011. [Online]. Available: https://cs.uwaterloo.ca/~cleve/courses/F11CS667/SimonClassicalLB.pdf

[30] C. Richard, "Quantum information processing quantum algorithms ii," 2021. [Online]. Available: http://cleve.iqc.uwaterloo.ca/resources/QIC-710-F21/Qic710QuantumAlgorithmsPart2.pdf

[31] "Quantum fourier transform." [Online]. Available: https://learn.qiskit.org/course/ch-algorithms/quantum-fourier-transform

[32] T. Mullor, "Answer to "why do we need to reverse the order of qubits in quantum fourier transform?"," 2022. [Online]. Available: https://quantumcomputing.stackexchange.com/a/26339

[33] C. Richard, "Quantum information processing quantum algorithms iii," 2021. [Online]. Available: http://cleve.iqc.uwaterloo.ca/resources/QIC-710-F21/Qic710QuantumAlgorithmsPart3.pdf

[34] D. Adrian, K. Bhargavan, et al., "Imperfect forward secrecy: how diffie-hellman fails in practice," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.* in Ccs '15, New York, NY, USA, Oct. 2015, p. 5. [Online]. Available: https://dl.acm.org/doi/10.1145/2810103.2813707

[35] R. Canty, "Understanding cryptography with rsa," 2020. [Online]. Available: https://jryancanty.medium.com/understanding-cryptography-with-rsa-74721350331f

[36] B.Swan, "Answer to "how to understand rsa encryption/decryption equation?"," 2019. [Online]. Available: https://math.stackexchange.com/a/3443510

[37] T. Pornin, "Answer to "cracking plain rsa without private key"," 2011. [Online]. Available: https://crypto.stackexchange.com/a/1409

[38] [Online]. Available: https://learn.qiskit.org/

[39] "Quantum phase estimation." [Online]. Available: https://learn.qiskit.org/

[40] "Shor's algorithm - intro to quantum software development." [Online]. Available: https://stem.mitre.org/quantum/quantum-algorithms/shors-algorithm.html

[41] J. Barzen, and F. Leymann, "Continued fractions and probability estimations in shor's algorithm: a detailed and self-contained treatise," *Appliedmath*, vol. 2, no. 3, p. 393, Jul. 2022. [Online]. Available: https://www.mdpi.com/2673-9909/2/3/23

[42] J. A. Smolin, G. Smith, and A. Vargo, "Pretending to factor large numbers on a quantum computer," *Nature*, vol. 499, no. 7457, p. 163, Jul. 2013. [Online]. Available: http://arxiv.org/abs/1301.7007

[43] "Answer to "largest integer factored by shor's algorithm?_2018," 2018. [Online]. Available: https://crypto.stackexchange.com/a/59796

[44] C. Gidney, and M. Ekerå, "How to factor 2048 bit rsa integers in 8 hours using 20 million noisy qubits," *Quantum*, vol. 5, p. 433, Apr. 2021. [Online]. Available: http://arxiv.org/abs/1905.09749

[45] "Steane's error correction code - intro to quantum software development." [Online]. Available: https://stem.mitre.org/quantum/error-correction-codes/steane-ecc.html

[46] S. Anagolum, "Arithmetic on quantum computers: addition," 2019. [Online]. Available: https://medium.com/@sashwat.anagolum/arithmetic-on-quantum-computers-addition-7e0d700f53ae

[47] G. Schay, "How to add fast—on average," *Amer. Math. Monthly*, vol. 102, no. 8, p. 725, Oct. 1995. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/00029890.1995.12004648

[48] T. G. Draper, "Addition on a quantum computer," Aug. 2000. [Online]. Available: http://arxiv.org/abs/quant-ph/0008033

[49] S. Anagolum, "Arithmetic on quantum computers: addition, faster," 2018. [Online]. Available: https://medium.com/@sashwat.anagolum/qftaddition-ce0a0b2bc4f4

[50] A. Paler, "Quantum fourier addition, simplified to toffoli addition," *Physical Rev.*, vol. 106, no. 4, p. 42444, Oct. 2022. [Online]. Available: http://arxiv.org/abs/2209.15193

[51] H. Thapliyal, T. S. S. Varun, E. Munoz-Coreas, K. A. Britt, and T. S. Humble, "Quantum circuit designs of integer division optimizing t-count and t-depth," in *2017 IEEE Int. Symp. Nanoelectronic Inf. Syst. (Inis)*, vol. 0, 2017, pp. 123–128, doi: 10.1109/iNIS.2017.34.

[52] F. J. Orts Gómez, G. Ortega Lopez, and E. M. Garzon, "A quantum circuit for solving divisions using grover's search algorithm," 2018.

[53] A. Parent, M. Roetteler, and M. Mosca, "Improved reversible and quantum circuits for karatsuba-based integer multiplication," Jun. 2017. [Online]. Available: http://arxiv.org/abs/1706.03419

[54] S. Anagolum, "Arithmetic on quantum computers: multiplication," 2020. [Online]. Available: https://medium.com/@sashwat.anagolum/arithmetic-on-quantum-computers-multiplication-4482cdc2d83b

[55] L. Ruiz-Perez, and J. C. Garcia-Escartin, "Quantum arithmetic with the quantum fourier transform," *Quantum Inf. Process.*, vol. 16, no. 6, p. 152, Jun. 2017. [Online]. Available: http://arxiv.org/abs/1411.5949

Indian Institute of Science Education and Research, Mohali, Mohali, Punjab
Email address: ms21254@iisermohali.ac.in
URL: www.arnavmetrani.github.io