Noname manuscript No.
(will be inserted by the editor)

Demonstration of a Quantum Calculator on IBM Quantum Experience Platform and its Application for Conversion of a Decimal Number to its Binary Representation

Shreya Singh · Prathamesh P. Ratnaparkhi · Bikash K. Behera · Prasanta K. Panigrahi

Received: date / Accepted: date

Abstract We demonstrate a quantum calculator by experimentally simulating and running the basic arithmetic operations viz. addition, subtraction, multiplication, division, square root, logarithm and fractional powers on real quantum devices using IBM Quantum Experience (IBM QE) platform. We implement a new bit-wise multiplication approach for simulating multiplication of small numbers. The existing division algorithms have limitations in terms of the nature of the two numbers taken. Here, we propose a new generalized division algorithm that can divide any two numbers irrespective of their nature. The proposed algorithm is an analog of the classical Newton-Raphson division algorithm and is found to be more efficient than the existing ones due to its quadratic convergence. We also show alternative two's complement based approach for division of two numbers which does not depend on the nature of the two numbers taken and is always accurate. Then, we propose the first quantum algorithm to convert a decimal number to a binary one. Hence we can realize a practical quantum calculator by performing the operations on a real quantum computer.

Keywords Arithmetic Operations, Calculator, IBM Quantum Experience

Shreya Singh
DGSPG College, Varanasi 221002, UP, India
E-mail: einsteindevotee@gmail.com

Prathamesh P. Ratnaparkhi
Department of Mechanical Engineering, VIT, Pune 411037, India
E-mail: pratham28j@gmail.com

Bikash K. Behera
Department of Physical Sciences, Indian Institute of Science Education and Research Kolkata, Mohanpur 741246, West Bengal, India
E-mail: bkb18rs025@iiserkol.ac.in

Prasanta K. Panigrahi
Department of Physical Sciences, Indian Institute of Science Education and Research Kolkata, Mohanpur 741246, West Bengal, India
E-mail: pprasanta@iiserkol.ac.in

1 Introduction

Recent years have seen an enormous progress in the field of quantum computation. IBM, through its IBM Q Experience platform, has provided access to real quantum computers and simulator which is a great resource in experimental front. A number of experiments have been performed using this platform: testing existing quantum algorithms [1,2], simulation of Ising model [3], study of far-from-equilibrium dynamics [4], observation of Klein-paradox [5], measuring topological phase [6], quantum tunneling simulation [7], quantum artificial intelligence [8], quantum machine learning [9], developing new algorithms for hard problems [10,11], solving quantum games [12,13], designing quantum devices [14,15,16], quantum state and gate teleportation [17,18,19], quantum state discrimination [20,21,22], quantum information [23,24], quantum error correction [25,26,27,28] to name a few.

A calculator primarily involves four basic arithmetic operations which are addition, subtraction, multiplication and division. Quantum algorithms have been developed to realize the above operations [29]. However, implementations of those on a quantum computer or on a quantum simulator have not been done till date. Here, we provide a brief account of existing algorithms and simulate the schemes of addition, subtraction and multiplication on IBM's classical topology simulator. We also include the simulation of the quantum circuits for calculating square root, natural logarithm, and fractional powers by simulating those on the 32-qubit quantum simulator. Although classical computers can efficiently perform arithmetic operations, the need of quantum algorithm arises because they are essential for other algorithms e.g. Shor's algorithm requires modular arithmetic operations [30]. It is well-known that a quantum algorithm is a unitary transformation on the qubits. The most basic requirement of a quantum algorithm is to be reversible i.e., it should be possible to regain the inputs by applying a inverse transformation. Arithmetic operations are many-to-one maps, thus given an output it may not be always possible to arrive at a unique input. The following identities illustrate this fact: a+b=b+a, a.b=b.a, a-b=(a+c)-(b+c), $\frac{a}{b} = \frac{c.a}{c.b} \quad c, b \neq 0$. From the first two identities, we observe that swapping the inputs (which corresponds to two different input configurations) gives the same output. The next two identities show that adding a number to each input (in case of subtraction) and multiplication by a nonzero number (in case of division) results in different input configurations giving the same output. Apart from non-reversibility, gate errors and limited connectivity between the qubits on existing devices make it difficult to realize an arithmetic operation on a real quantum device. The issue of non-reversibility can be addressed by using additional ancillary qubits. Unlike quantum algorithms, classical algorithms can be irreversible hence extra qubits are not required.

The organization of the paper is as follows. In Sec. 2, we simulate a quantum full adder which serves as a basic unit for addition operation. Then in Sec. 3, we provide a method for doing subtraction using two's complement. Multiplication of small numbers is then simulated by bit-wise multiplication approach. Finally, in Sec. 5, we review the existing division algorithms and propose a new generalized division algorithm and also show alternative two's complement based approach for division, which do not have any limitations on the nature of the two numbers taken for division. In Sections 6, 7 and 8, we demonstrate the simulation of quantum circuits for calculating square root, logarithm, and fractional powers respectively. In Section 9, we propose the first quantum algorithm for conversion of a decimal number into its binary form. Finally, we conclude in Section 10 discussing the future implications of the work.

2 Addition

The existing quantum algorithms for addition can be classified into two categories: algorithms with purely quantum approach and algorithms that reflect their classical counterparts. While the classical approach merely translates the ripple carry and carry save schemes, the quantum approach uses quantum Fourier transform (QFT) [31,32]. In quantum approach, conditional rotations are applied on QFT of one of the numbers to be added. The rotations applied commute with each other and depend on the second number. The commutative property of this operation allows parallelization. Since the second number appears in the picture only through the conditional rotations (which can be programmed externally), it is possible to implement this algorithm without storing the second number in a quantum register thus saving qubits. After conditional rotations inverse QFT is applied to get the sum of the two numbers.

A brief account of classical addition approaches viz. ripple-carry and carry-save can be found in paper by Gosset [33]. The number of qubits required for quantum ripple carry algorithm (QRCA) and for quantum carry save algorithm (QCSA) are O(n) and O(n^2) respectively. The respective quantum gate delays are O(n^3) and O($nlogn$) [33]. Although carry save algorithm has better time efficiency, considering the expensiveness of qubits and high processing frequency of quantum devices we prefer QRCA over QCSA. We adopt the schemes given by Vedral *et al.* [34] and Fahdil *et al.* [29] for simulating the addition algorithm.

The quantum circuit for QRCA is essentially a sequence of quantum full adders in which carry at a step is propagated to the next step [33,29]. It is interesting to see that a single CNOT gate has a truth table (Table 2) almost same as the half adder (Table 1) except for a single entry in carry column (shown in square bracket). Because of non bijective nature of addition whatever operation we do to match this entry essentially changes other entries. Thus we have to use an extra ancillary qubit and a Toffoli gate to compute the carry. We use Toffoli gate because its truth table (Table 3) matches the entries in carry column of half adder. We combine two half adders to get a full adder [29] (Table 4). We notice that the carry of the two half adders can not

Table 1 *Half adder truth table.*

Input1	Input2	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	[0]
1	1	0	1

Table 2 *CNOT truth table.*

Control	Target	Target	Control
0	0	0	0
0	1	1	0
1	0	1	1
1	1	0	1

Table 3 *Toffoli gate truth table.*

Control1	Control2	Target	Target
0	0	0	0
0	1	0	0
1	0	0	0
1	1	0	1

Table 4 *Results of full adder implementation. q[0] stores the carry from previous step. It is zero for the first step. q[1] and q[2] store n^{th} significant digits of first and second number respectively. c[0] stores n^{th} digit of answer. c[1] is the carry for the next step.*

q[0]	q[1]	q[2]	c[0]	c[1]
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

be simultaneously 1, hence a single CNOT gate is sufficient to compute the overall carry.

Fig. 1 shows the quantum circuit for full adder as implemented on IBM Q Experience. q[] is the quantum and c[] is the classical register. Table 4 shows the sum (c[0]) and carry (c[1]) for different input configurations of quantum register. The ripple carry circuit uses the n full adders in succession to compute the addition of two n digit numbers [33].

3 Subtraction

Subtraction can be thought of as addition of negative numbers so that there is no need for a separate algorithm for subtraction. We just have to find a

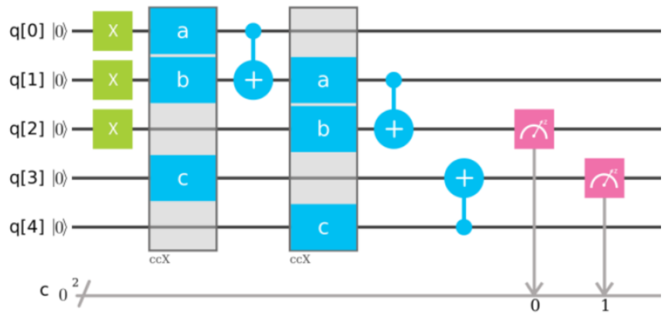

Fig. 1 *Full adder implementation on IBM Q. For Toffoli gate in blue color A and B are the controls and C is the target. The first Toffoli gate followed by CNOT gate forms a half adder. Similarly, the second Toffoli gate and CNOT gate form another half adder. These two half adders are connected using a CNOT gate to form a full adder. This particular diagram is for input 111.*

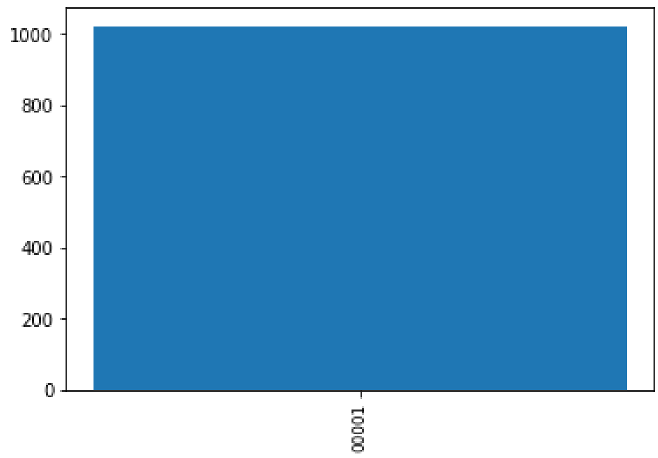

Fig. 2 *Result of full adder implementation on IBMQ simulator.*

way to represent the negative numbers. This representation can be achieved by taking two's complement. Thus subtraction of b from a is the same as the addition of two's complement of b to a. Since we have already implemented the addition algorithm, the problem of subtraction reduces to finding algorithm for two's complement. Taking two's complement of a binary number amounts to flipping all the bits and adding 1 to it. The quantum circuit for computing two's complement for 3 bit signed binary number is shown in Fig. 6.

In the quantum circuit, strings q[3]q[2]q[1] and c[2]c[1]c[0] represent input binary number and two's complement of input respectively. We apply NOT gates on the input qubits and then add 1 (stored in q[0]) to obtain two's complement. Since negative zero (100) is same as the positive zero (000) they

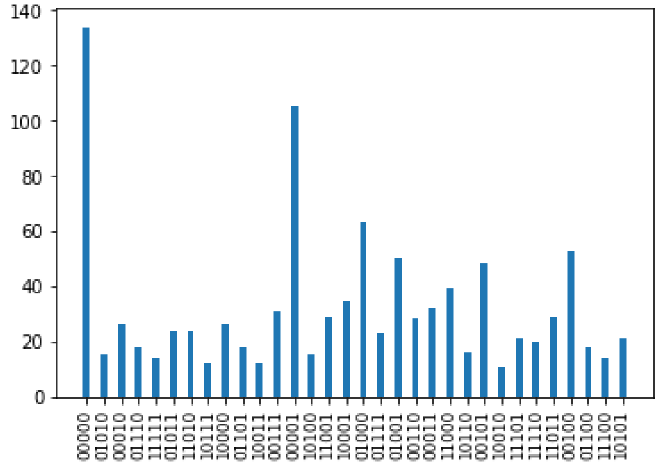

Fig. 3 *Result of full adder implementation on IBMQx2.*

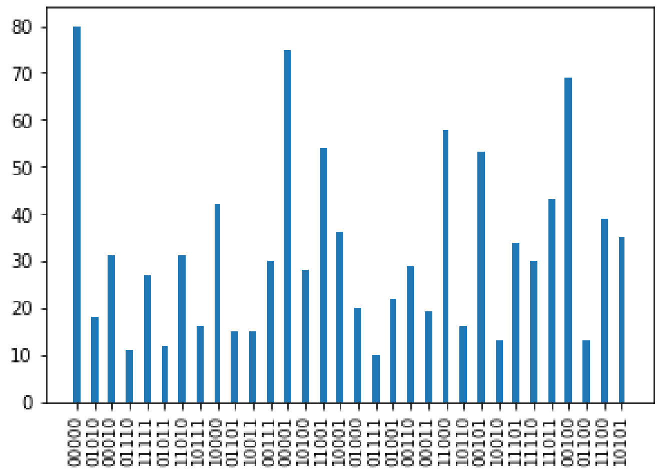

Fig. 4 *Result of full adder implementation on IBMQx4.*

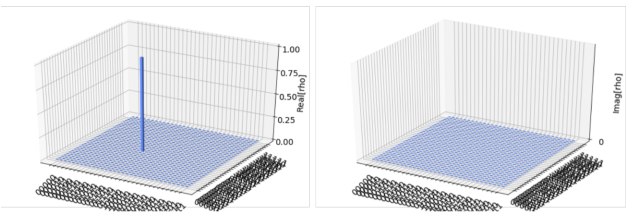

Fig. 5 *State tomography plots for full adder circuit given in Fig. 1.* Time taken: 18.92717432975769, Least-Sq Reconstruction, Time taken: 4.527880907058716, Fit Fidelity: 0.9937225404847748, CVX Reconstruction, Time taken: 68.47750735282898, Fidelity: 0.9999285813161057.

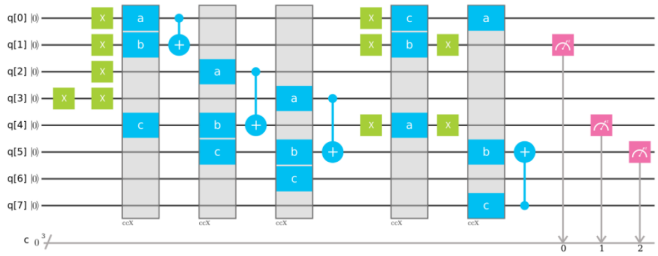

Fig. 6 *Quantum circuit for two's complement. For Toffoli gate in blue color A and B are the controls and C is the target. In this particular circuit the input is 100.*

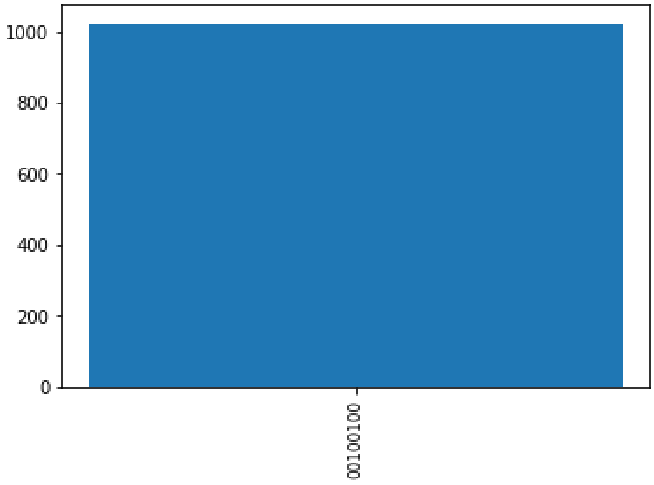

Fig. 7 *Result of two's complement for input 100 on IBMQ simulator.*

should both give output 000. If we go by the definition of two's complement then we get 100 as the two's complement of 000. The entire circuit after third Toffoli gate is designed to make a conditional bit-flip on the most significant digit so that both inputs 100 and 000 map to 000. Two extra qubits are required to do this conditional flip, which are obtained by reusing q[0] and adding an extra qubit q[7]. q[4], q[5] and q[6] are required as the extra qubits for addition. Table 5 shows the result of two's complement circuit.

4 Multiplication

Multiplication is computationally more expensive than addition. Multiplication of a with b can be thought of as adding a to itself b times. This process can be used to design a recursive algorithm to compute multiplication. Such an algorithm can be found in the paper by Florio and Picca [35]. In this method, the numbers a and b are stored in two different quantum registers. A separate

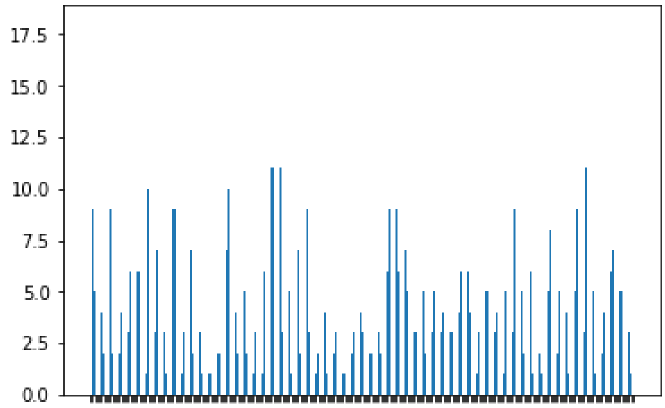

Fig. 8 *Result of two's complement for input 100 on IBMQ16 Melbourne (due to large circuit, there was a large data discrepancy (as larger the circuit, more the variation due to noise in real quantum device). The largest-valued data element was 11111111 with probability = 1 (out of 1024 shots). The lowest-valued data element was 00000000 with probability = 9 (out of 1024 shots). The data element with the highest probability was 00000010 with probability = 13 (out of 1024 shots). The correct value for the circuit (as obtained from IBMQ simulator) '00100100' had probability = 12 (out of 1024 shots)).*

Table 5 *Result of two's complement circuit.*

q[3]q[2]q[1]	c[2]c[1]c[0]
000	000
001	111
010	110
011	101
100	000
101	011
110	010
111	001

register is used for storing the answer, which initially stores zero. Now a is added to answer register and b is decremented by 1 each time. The control exits the loop when b is equal to zero. Thus this algorithm is composed of comparison, addition and subtraction. A difficulty in implementing such an algorithm on existing devices is designing the escape from loop which requires a controlled measurement. The quantum version of Booth algorithm [36] also has the same limitation when it comes to simulation.

Instead of a recursive algorithm a quantum algorithm for multiplication can be based on digit by digit multiplication. From Table 3 we can see that a Toffoli gate can be used to do multiplication of two single digit binary numbers. For calculating product of two multi-digit numbers we use multiple Toffoli gates. The circuit in Fig. 9 demonstrates quantum algorithm for digit by digit multiplication of two 2-digit binary numbers. The numbers are represented by strings a_1a_0 and b_1b_0. The first four Toffoli gates taken in order to calculate $a_0 \times b_0$, $a_0 \times b_1$, $a_1 \times b_0$ and $a_1 \times b_1$. $a_0 \times b_0$ directly forms the least significant qubit of answer say c_0. Rest of the circuit implements two half adders to get the remaining digits of answer. Table 6 shows the output of this circuit.

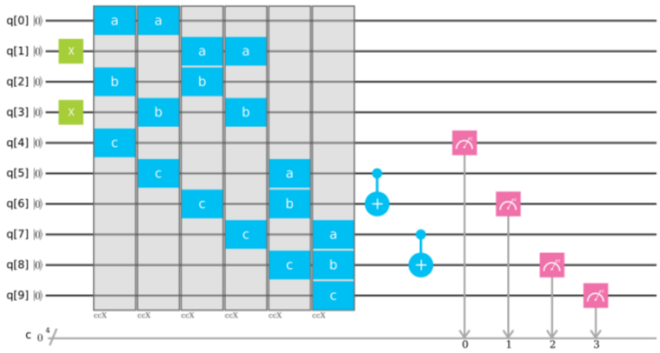

Fig. 9 *Quantum circuit for multiplication. q[1]q[0] stores the first number a_1a_0 and q[3]q[2] stores the second number b_1b_0. The blue colored gates are Toffoli gates with A, B as the controls and C as the target. The circuit is configured for both the inputs equal to 10.*

5 Division

Out of the four basic arithmetic operations, division is computationally most demanding operation. Division N/D = Q can be alternatively posed as a multiplication $N = D \times Q$ i.e. which number when multiplied by D gives N. If

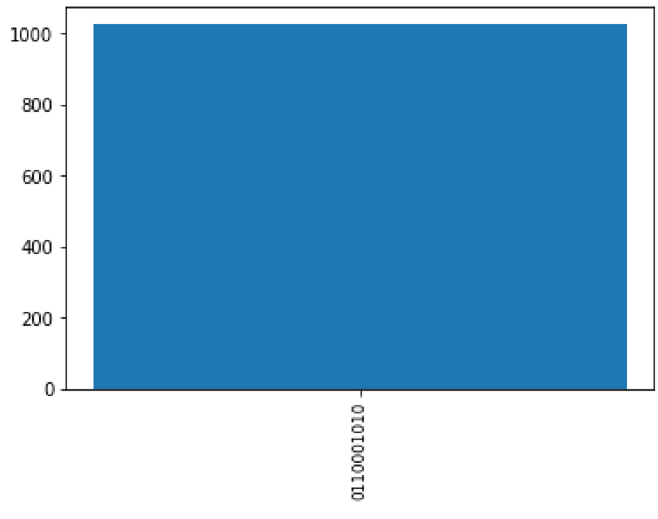

Fig. 10 *Result of multiplication quantum circuit for both the inputs equal to 10 on IBMQ simulator.*

Table 6 *Results of multiplication of two 2-digit binary numbers.*

a_1a_0	b_1b_0	$c_3c_2c_1c_0$	Multiplication in decimal form
00	00	0000	$0 \times 0 = 0$
00	01	0000	$0 \times 1 = 0$
00	10	0000	$0 \times 2 = 0$
00	11	0000	$0 \times 3 = 0$
01	01	0001	$1 \times 1 = 1$
01	10	0010	$1 \times 2 = 2$
01	11	0011	$1 \times 3 = 3$
10	10	0100	$2 \times 2 = 4$
10	11	0110	$2 \times 3 = 6$
11	11	1001	$3 \times 3 = 9$

an algorithm for multiplication by a fixed number D is known then such algorithm can serve as an oracle in Grover search algorithm to find N/D where D is fixed and N can be any number. Note that the oracle changes when we change D which means that this is not a prescription for a general division algorithm. Implementation of Grover search algorithm for division by numbers of form $2^k + 1$ is proposed by [37]. The oracle is based on algorithm for multiplication by numbers of type $2^k + 1$ proposed by [38]. The problem with such approach is probabilistic nature of Grover search algorithms. Also due to irreversible nature of multiplication, construction of a general oracle which can be used to divide by any arbitrary number is not possible. Other approaches use repetitive subtraction of denominator from numerator until a numerator becomes less than denominator [29,39]. The current devices do not support conditional loops thus such methods have little practical viability. However, a better way to do division would be to create a quantum analog of classical Newton-Raphson division algorithm [40]. Here we provide a pseudo code for such algorithm.

- N, D : Take N and D as numerator and denominator respectively in floating point format $M \times 2^e$, where M is mantissa and e is exponent with $1 \leq M < 2$.
- $D = D/2^{e+1}$: Shift qubits of D to scale down such that $0.5 \leq M < 1$.
- $N = N/2^{e+1}$: Shift qubits of N such that the fraction N/D does not change.
- R : Take R as reciprocal of D.
- $R = R_0$: Compute initial guess for reciprocal.
- $REPEAT(S)$: S is the number of steps.
- $R = R+R\times(1-D\times R)$: Calculate successive approximation for reciprocal.
- END.
- RETURN $N \times R$.

The process of division can be interpreted as multiplication by reciprocal of denominator. Finding reciprocal of a number D can be framed as finding root of the function $f(x) = 1/x - D$. Clearly the solution to equation $f(x) = 0$ is $1/D$. By applying Newton-Raphson method to this function, we get $x_{i+1} = x_1 + x_i(1 - D \times x_i)$ as the recursion formula. The optimum initial guess is

```
Step 1: Subtract divisor from dividend
Step 2: Subtract divisor from difference
Step 3: Repeat Step 2 till difference is less than divisor
Step 4: The number of steps required to reach Step 3 is the quotient
Step 5: The difference in Step 3 is the remainder
```

Fig. 11 *Method to compute quotient and remainder using quantum circuit for two's Complement.*

calculated for a given range of D. Classical ways to find the initial guess [41, 42] can be used in quantum case as well. The number of steps (S) depends on precision required. As this method has quadratic convergence (number of correct digits doubles with each iteration), we get a good approximation of reciprocal with fewer steps. Multiplication of reciprocal of denominator with numerator gives the required division.

Alternatively, we can find quotient and remainder via iterations of quantum circuit for two's complement by following the steps given in Fig. 11. The quantum circuit for two's complement is given in the Section 3.

6 Square root

To find square root, we use Algorithm 1 SQRT as proposed by Bhaskar *et al.* [43]. All the steps involved in this algorithm can be carried out by multiplication and addition in fixed precision binary arithmetic. The following steps are taken to calculate the square root of a number. Let us consider a number $w = 6.25$ whose square root needs to be calculated. So we convert the number to its binary form with b bits after decimal point. Greater the number b, greater is the accuracy of the result. For input to the first circuit, we keep $b \geq n - m$, where n is the number of digits in binary representation of n and m is the number of integers in w. Binary representation of 6.25 is 110.01. Adding to it $b = 3 > 5 - 2$ bits after decimal point, our input will be 110.01000. We use this as input in the circuit as shown in Fig. 12 and simulate it using IBMQ simulator. After truncating it to 4 digits after decimal point, the output is obtained to be 0010001. Let the output be denoted by x_0. Now, let us take $s = log_2 b$ $= 2$. Then, $x_1 = -wx_0^2 + 2x_0$ $x_2 = x_s = -wx_1^2 + 2x_1 = 0.1011$. This will be the input for the circuit shown in Fig. 14. Now we take $b = 7 \geq max\{2m, 4\}$. The output of this circuit, 00.0011111 is truncated to 7 digits after decimal points and let it be denoted by y_0. To this we perform last 2 steps of Algorithm 1 SQRT [43]. We know $s = 2$ (from above) $y_1 = 1/2(3y_0 - x_s y_0^3)$, $y_2 = y_s = 1/2(3y_1 - x_s y_1^3) = 10.1000001011$ which we can approximate to 10.1. Our final result is 10.1 which is equal to 2.5, the square root of 6.25. The cost of each iterative step is O$log_2 b$ qubits and the number of quantum operations required to implement addition and multiplication is a low degree polynomial in $n + b$.

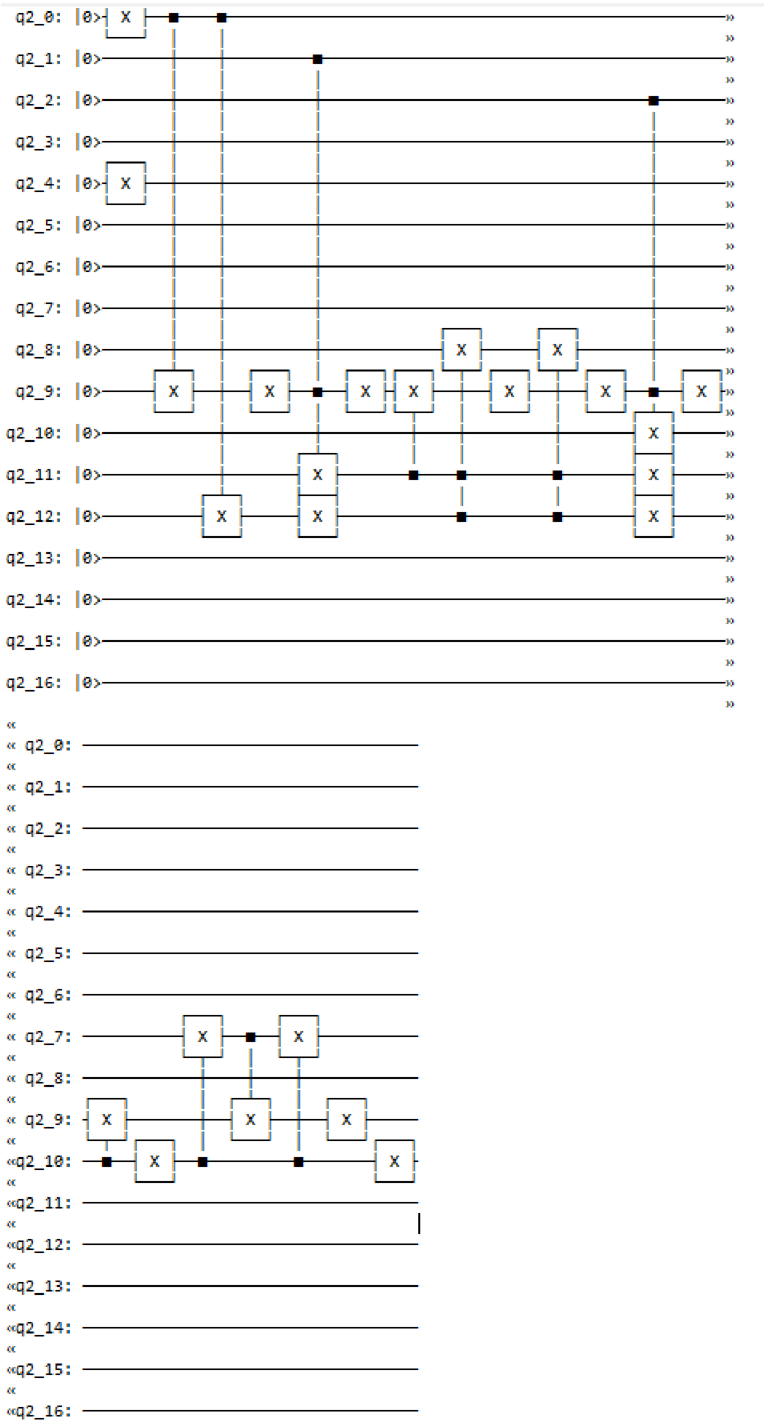

Fig. 12 *Quantum circuit for square root. This circuit computes* x_0.

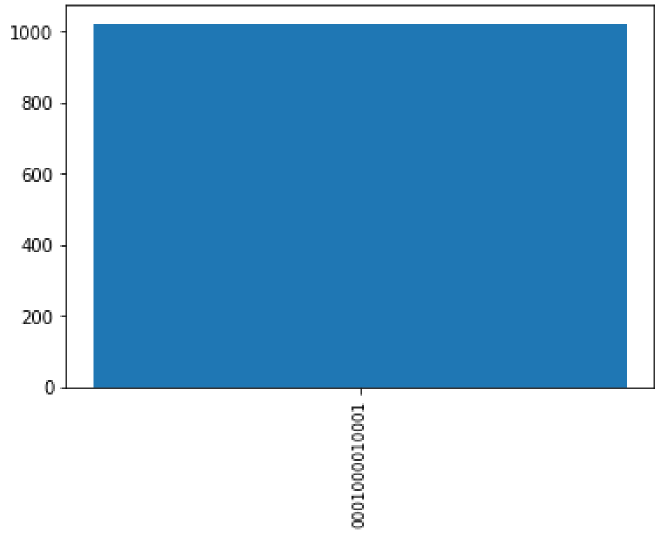

Fig. 13 *Result of first quantum circuit for square root on IBMQ simulator that computes* x_0

7 Logarithm

To compute natural log of a number, we use Algorithm 3 LN [43]. Let us consider the number $w = 2$. Then, we input the binary form of 2 which is equal to 10 and compute x in QISKit using IBMQ simulator as shown in Fig. 16 and get $x = 01$. Then we compute $p-1$ as shown in Fig. 17 and get $p-1 = 00$ when simulated using IBMQ simulator. Now, $w_p = w2^(1-p)$ which is found to be 2. Fractional part of w_p is $x_p = w_p - 1 = 1$. Since $x_p \neq 0$, therefore $z_p \neq 0$. We define another parameter $l \geq log_2 8n$, where n is the number of digits in binary representation of w, so we take $l = 4$. And b determines the accuracy as greater the b, greater is the accuracy. $b \geq max\{5l, 25\}$, so we take $b = 25$. $r = ln2$. $t_p = 1/2^l th$ root. We compute t_p using Algorithm 2 PowerOf2Roots [43]. We get $t_p = 1.11000111101011100001$ which is approximately equal to 1.77. $y_p = (t_p - 1/2(t_p - 1)^2$ which is approximately equal to 0.08657. $z_p = 2^l y_p = 1.38512$. $ln\ w = ln\ 2 = z_p + (p-1)r = 0.08657 + (-1)ln2 = 0.9212$ which is approximately equal to 0.693, the natural log of 2. Generalization of circuit used to compute $p - 1$ in Fig. 17 with input x- Each qubit of the form 2^{-n}, say q, is connected to the ancilla qubit and to qubits whose sum make up the exponent of q. The overall cost of this algorithm is $\mathrm{O}l(n + blogb)$ qubits and for implementation, the number of quantum operations required is proportional to $llog\ bp(n + b)$.

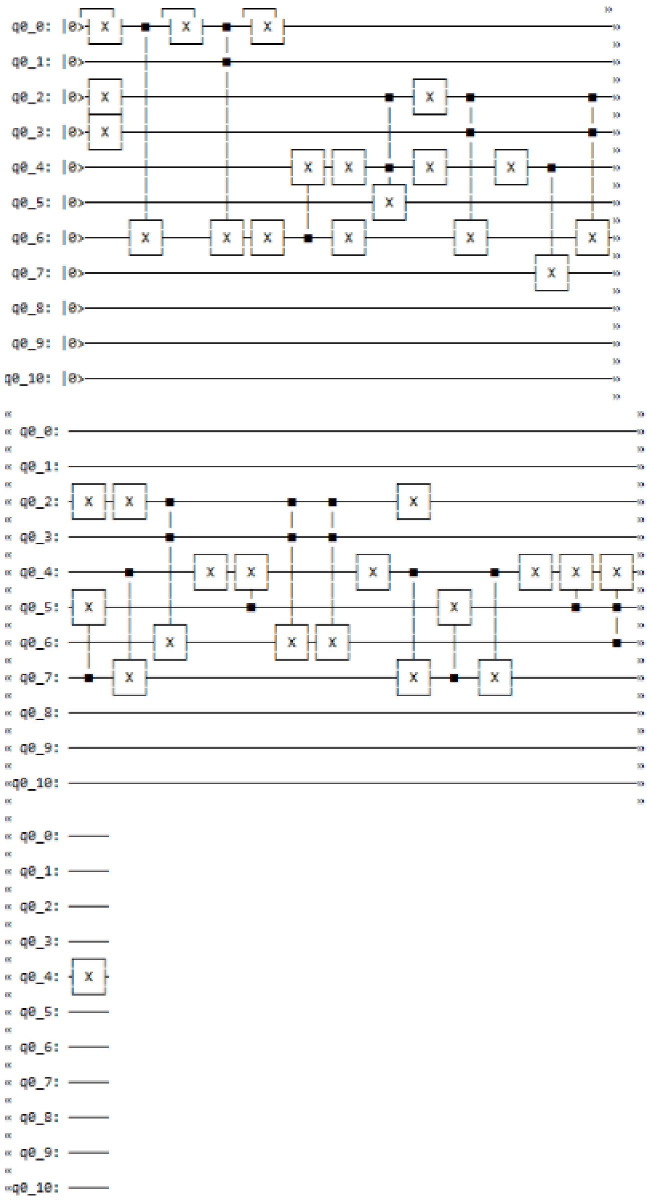

Fig. 14 *Quantum circuit for square root. This circuit computes y_0.*

8 Fractional Powers

The problem of finding fractional powers of a number can be approached in two ways, based on the fact that if the number is a positive integer or a

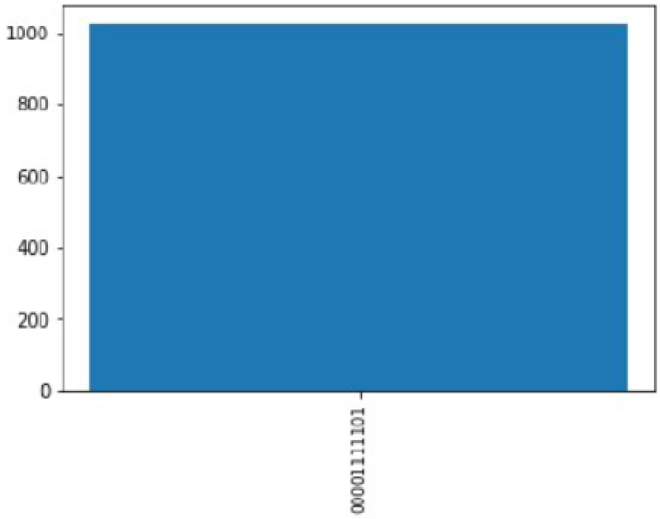

Fig. 15 *Result of second quantum circuit for square root on IBMQ simulator that computes* y_0

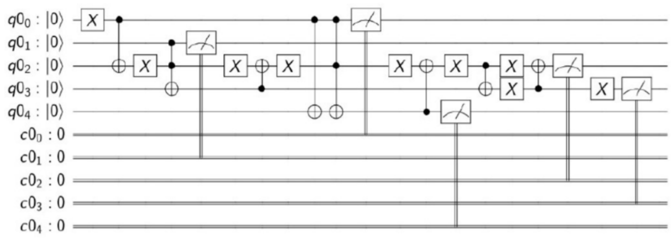

Fig. 16 *Quantum circuit for natural logarithm. This circuit computes* x.

positive fractional number and the nature of exponent, as defined in Algorithm 4 FractionalPower [43] and Algorithm 5 FractionalPower2 [43].

Let the number to be computed is w^f where $w >= 1$ and $f[0,1]$, n is the number of qubits used to express w, n_f is the number of qubits used to express f and l be a natural number chosen by user to increase the accuracy of the result. Result of each step is truncated to b qubits after the decimal point where $b \geq max\{n, n_f, [5(l+2m+ln\ n_f)], 40\}$. If the exponent is in the form of 2^k, then we can simply use the process of finding square root in a recursion as defined in Algorithm 4 Fractional Power [43] which used Algorithm 2 Power Of 2 Roots [43]. The cost of this approach is $0(n_f(n+b)logb)$ qubits and quantum operations of order $n_f logbp(n+b)$.

If exponent is a rational number of the form p/q such that it cannot be expressed in power of 2 or if the number w is a positive fractional number, then we can approach it using the Algorithm 5 Fractional Power 2 [43]. The

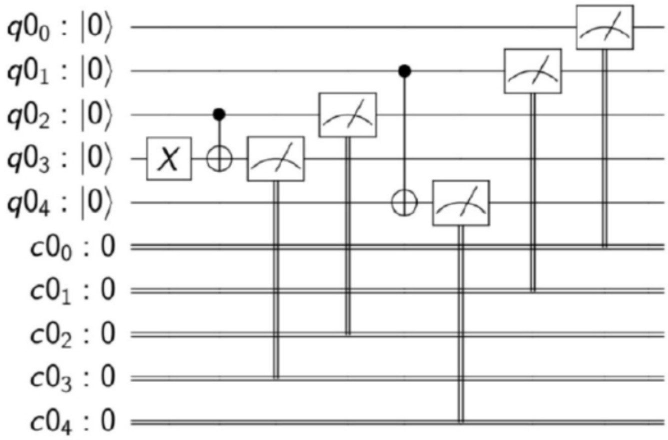

Fig. 17 *Quantum circuit for natural logarithm. This circuit computes* $p-1$.

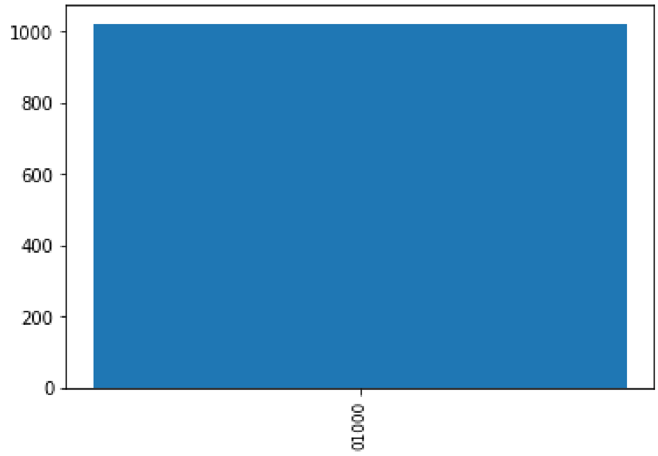

Fig. 18 *Result of the first quantum circuit for natural logarithm on IBMQ simulator that computes* x.

cost of this algorithm in terms of both the number of quantum operations and the number of qubits is a low-degree polynomial in n, n_f, and l.

Generalization of the circuit used to compute fractional power if the number is a positive fractional number in Fig. 27 with input w- Each qubit of the form 2^{-n}, say q, is connected to the ancilla qubit and to qubits whose sum make up the exponent of q.

For example, let the number w be 16 and f be 1/4 such that i in $(1/2)^2$ will be 2. Then, we'll be using Algorithm 4 Fractional Power [43] to find fourth root of 16.

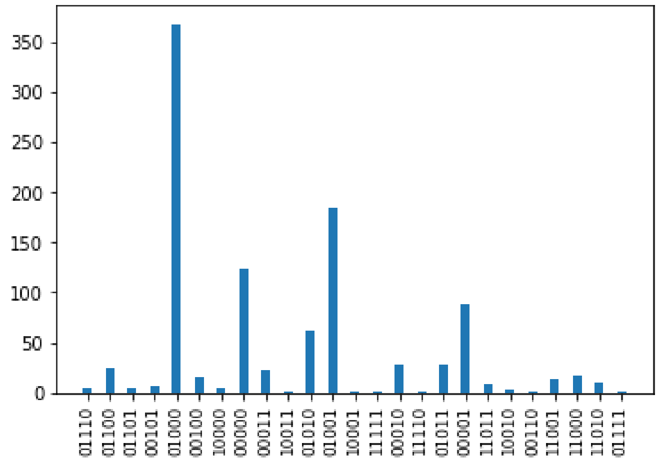

Fig. 19 *Result of the first quantum circuit for natural logarithm on IBMQx2 that computes x.*

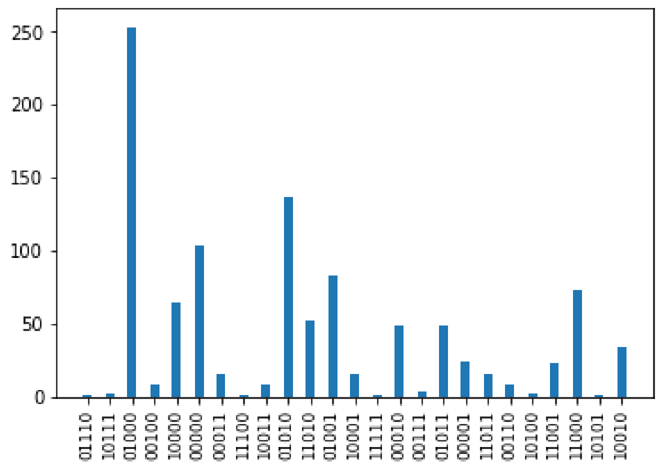

Fig. 20 *Result of the first quantum circuit for natural logarithm on IBMQx4 that computes x.*

Taking another example, let the number w be 0.5. Then, the quantum circuit which will be used to find k that is required to compute any required fractional power of w, say 1/2, according to Algorithm 5 Fractional Power 2 [43] has been given in the Fig. 28. In this circuit, the first two qubits represent w, the third qubit is the ancilla qubit and the last qubit represents k (such that $(2^k)w \geq 1 > (2^{k-1})w$). The value of k when run on quantum simulator and real quantum devices have been given in the Fig. 29, Fig. 30, Fig. 31 and Fig. 32.

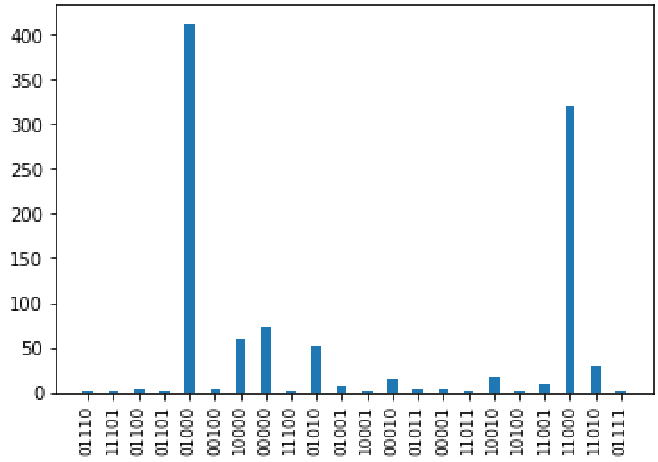

Fig. 21 *Result of the first quantum circuit for natural logarithm on IBMQ16 Melbourne that computes x.*

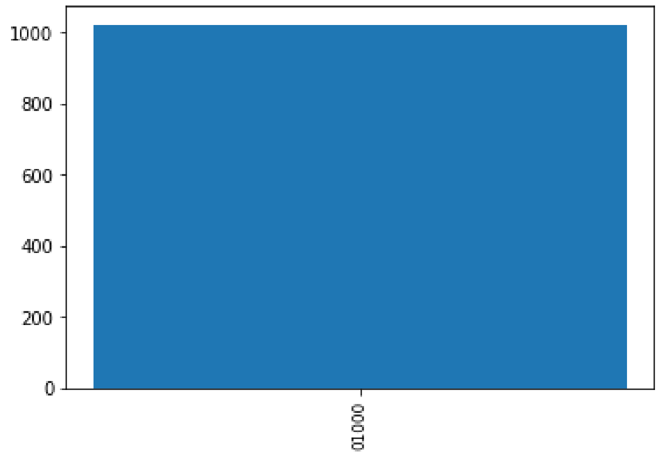

Fig. 22 *Result of the second quantum circuit for natural logarithm on IBMQ simulator that computes $p-1$.*

9 Quantum Algorithm for Conversion of Decimal number to its Binary representation

To the best of our knowledge, we haven't yet come across any other quantum computation based approach for converting a decimal number to its binary form. Classical algorithms are some variation of dividing decimal number by 2 and storing the remainder, but here we devise a new quantum approach and realize it on a real quantum computer. This algorithm can be used for both the prime and composite numbers. It is based on quantum adder. A quantum adder

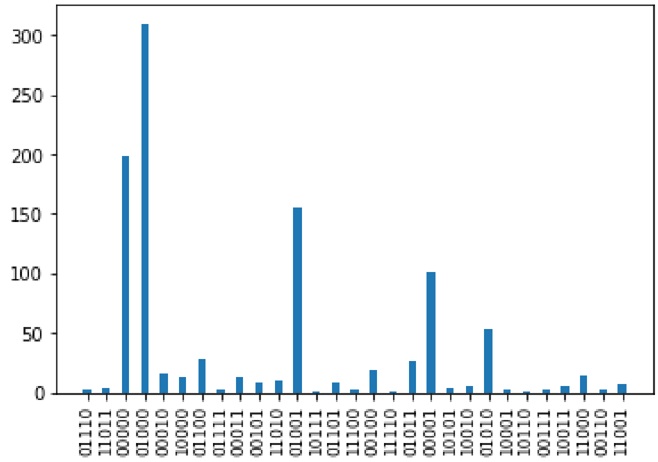

Fig. 23 *Result of the second quantum circuit for natural logarithm on IBMQx2 that computes $p-1$.*

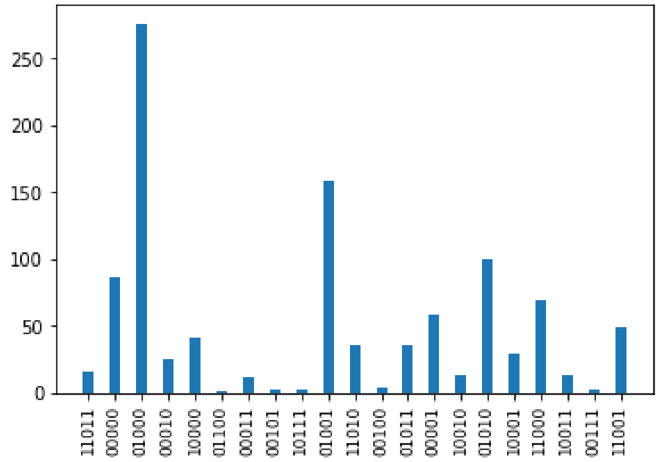

Fig. 24 *Result of the second quantum circuit for natural logarithm on IBMQx4 that computes $p-1$.*

outputs sum of decimal numbers input in binary form. So let's say number is n. Then we can input 1 n times by flipping qubits on a quantum computer and apply quantum adder circuit on the input, we get output which is binary representation of n. For this, we can make either make a circuit that adds 1 n times or we can make a circuit to first add 2 qubits, then we get output which is binary representation of 2 and we can input that again and add 1. But the cost for operation is obviously in latter approach. Let decimal number n be 2. Then, we applied two X gates to flip qubits and to that, we applied quantum adder circuit as shown in Fig. 34 to add two 1s. Then our output is

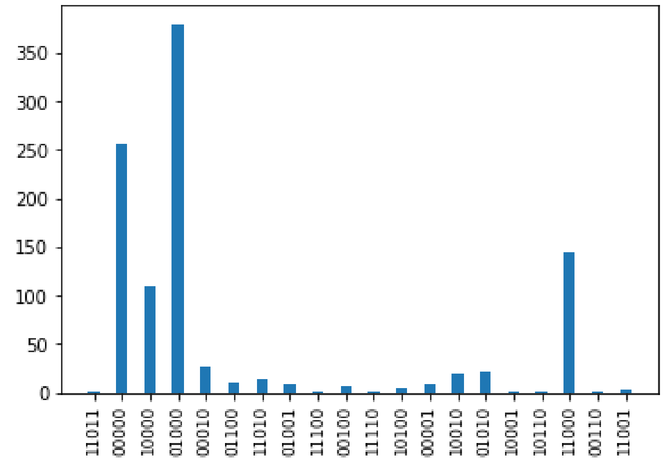

Fig. 25 *Result of the second quantum circuit for natural logarithm on IBMQ16 Melbourne that computes $p-1$.*

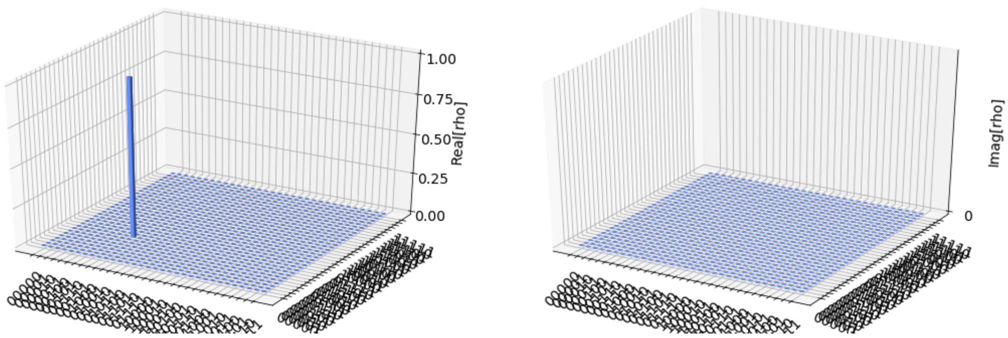

Fig. 26 *State tomography plots for natural logarithm circuit given in Fig.16.* Time taken: 9.441224098205566, Least-Sq Reconstruction, Time taken: 3.356473922729492, Fit Fidelity: 0.9938264377148213, CVX Reconstruction, Time taken: 49.066205739974976, Fidelity: 0.9999099312896844.

10 which is the binary representation of 2. If the number is composite, then we can break it into its factors and apply quantum multiplication in coordination with quantum adder. Let decimal number again be $n = 4$. To return binary representation of 4, we applied two X gates to flip qubits and to that, we applied quantum adder circuit as shown in Fig. 34 to add two 1s. Then, we multiplied two 2s in binary form (which we got from the previous circuit to be 10) as shown in the circuit given in Fig. 35 and our output is 100 which represents 4.

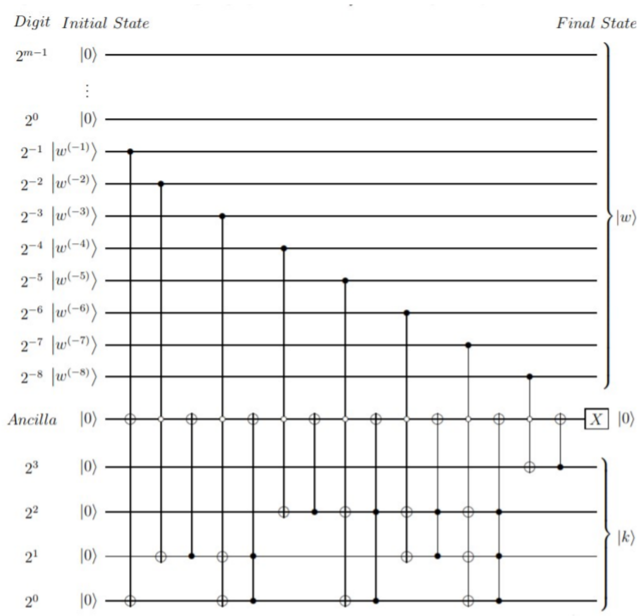

Fig. 27 *Quantum circuit for fractional powers. This circuit computes positive integer k having $n - m = 8$ significant digits after decimal point.*

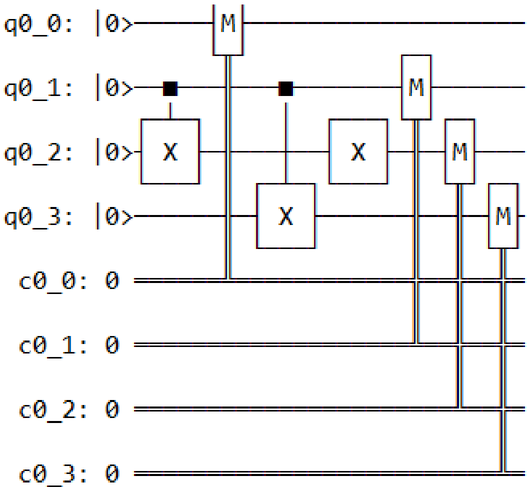

Fig. 28 *Quantum circuit for fractional powers when w is* 0.5

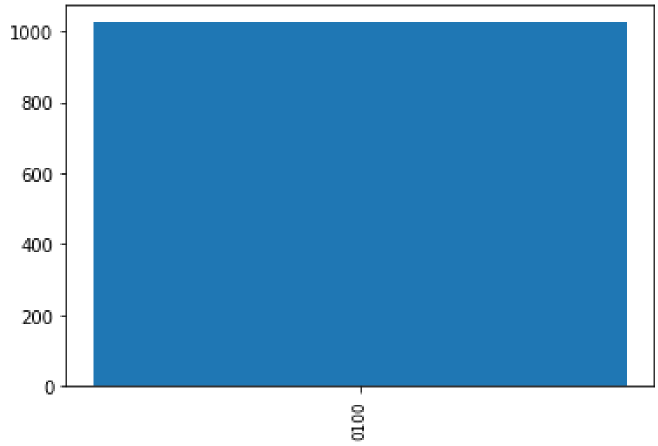

Fig. 29 *Result of the quantum circuit given in Fig. 28 on IBMQ simulator.*

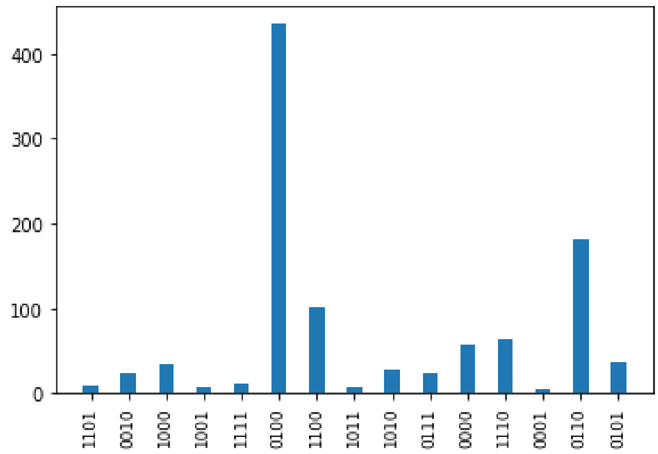

Fig. 30 *Result of the quantum circuit given in Fig. 28 on IBMQx2.*

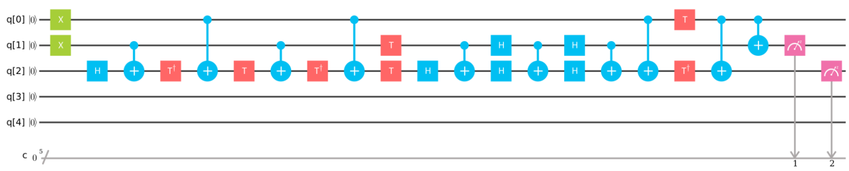

Fig. 34 *Quantum half-adder circuit, in which the input is given by applying two x gates, to add two 1's and simulating it, returns the result 10 which is binary representation of 2.*

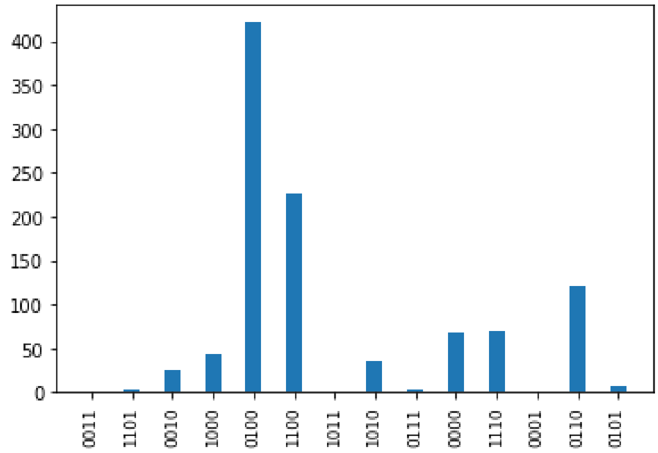

Fig. 31 *Result of the quantum circuit given in Fig. 28 on IBMQx4.*

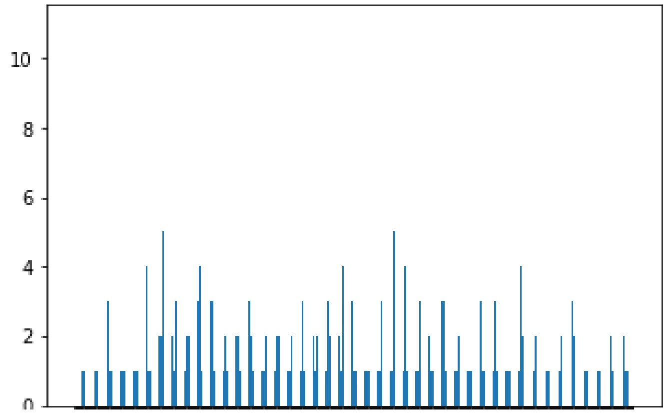

Fig. 32 *Result of the quantum circuit given in Fig. 28 on IBMQ16 Melbourne.*

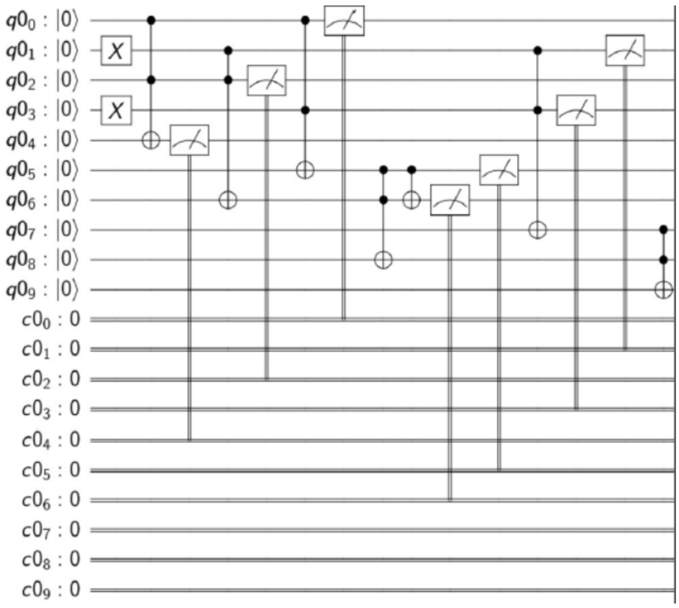

Fig. 35 *Quantum multiplication circuit, in which the input is given by applying two x gates at qubits 1 and 3, to multiply two 2's and the circuit returns 100 which is binary representation of 4.*

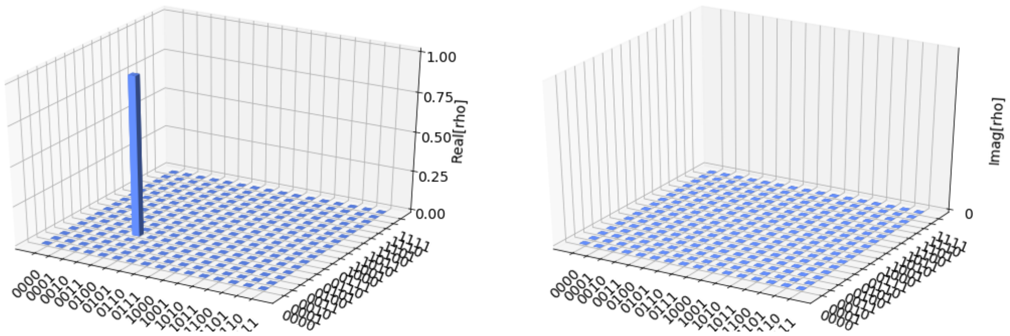

Fig. 33 *State tomography plots for circuit given in Fig.28.* Time taken: 0.923128604888916, Least-Sq Reconstruction, Time taken: 0.3053920269012451, Fit Fidelity: 0.9464690310361814, CVX Reconstruction, Time taken: 1.9022772312164307, Fidelity: 0.9900836435499704

10 Conclusion

In conclusion, we have demonstrated the basic arithmetic operations, three more advanced operations which form the basis of a calculator and quantum algorithm for conversion of decimal number to its binary form. Most of the previous works on quantum arithmetic operations are theoretical and a few attempts have been made towards simulation or implementation on a quantum computer. The focus of this work has been to simulate and run the algorithms on real quantum devices with currently available resources. Thus the algorithms simulated here are not the most efficient but the simplest to simulate with current technology. In case of division, as mentioned earlier even the simplest approaches have limitations when it comes to simulation, we have proposed more general and efficient algorithms. The quantum algorithms for calculating square root of a number, natural logarithm and fractional powers have been simulated on the IBMQ simulator provided by IBM Q platform. We proposed quantum algorithm for conversion of decimal number into its binary form. We hope that with greater connectivity between the qubits, and more versatile operations, it will be possible to implement efficient algorithms for arithmetic operations along with other useful operations on real quantum computer devices, so that it could be used as a quantum calculator.

Acknowledgements

S.S. would like to thank IISER Kolkata for the support and hospitality provided during this work which made it easy and fun. She would also like to thank DGSPG College for providing the opportunity. P.P.R. would like to thank IISER Kolkata for the support and hospitality provided during this work. He would also like to thank VIT Pune for providing the opportunity. B.K.B. acknowledges the financial support provided by IISER Kolkata. We are extremely grateful to IBM team and IBM QE project. The discussions and opinions developed in this paper are only those of the authors and do not reflect the opinions of IBM or IBM QE team.

Author Contributions

S.S. worked on square root, logarithm and fractional powers, proposed alternative two's complement based approach for division and quantum algorithm to convert decimal number to binary number, performed simulation, running on real devices and state tomography of every operation possible with currently available technology. P.R. simulated addition, subtraction, and proposed the bit-wise multiplication approach for simulating multiplication of small numbers and quantum analog of the classical Newton-Raphson division algorithm. B.K.B. gave the idea of realizing a quantum calculator on a quantum computer. S.S., P.R. and B.K.B. contributed to the composition of the manuscript. B.K.B. supervised the project. S.S., P.R. and B.K.B. have completed the project under the guidance of P.K.P.

References

1. Gangopadhyay, S., Manabputra, Behera, B. K., Panigrahi, P. K.: Generalization and Demonstration of an Entanglement Based Deutsch-Jozsa Like Algorithm Using a 5-Qubit Quantum Computer, Quantum Inf. Process. **17**, 160 (2018)
2. Coles, P. J. *et al.*: Quantum Algorithm Implementations for Beginners, arXiv:1804.03719 (2018)
3. Cervera-Lierta, A.: Exact Ising model simulation on a Quantum Computer, Quantum **2**, 114 (2018).
4. Zhukov, A. A., Remizov, S. V., Pogosov, W. V., Lozovik, Y. E.: Algorithmic simulation of far-from-equilibrium dynamics using quantum computer, Quantum Inf. Process. **17**, 223 (2018)
5. Kapil, M., Behera, B. K., Panigrahi, P. K.: Quantum Simulation of Klein Gordon Equation and Observation of Klein Paradox in IBM Quantum Computer, arXiv:1807.00521 (2018)
6. Viyuela, O. *et al.*: Observation of topological Uhlmann phases with superconducting qubits, npj Quantum Inf. **4**, 10 (2018)
7. Hegade, N. N., Behera, B. K., Panigrahi, P. K.: Experimental Demonstration of Quantum Tunneling in IBM Quantum Computer, arXiv:1712.07326 (2017)
8. Alvarez-Rodriguez, U., Sanz, M., Lamata, L., Solano, E.: Quantum Artificial Life in an IBM Quantum Computer, Sci. Rep. **8**, 14793 (2018).
9. Zhao, Z., Pozas-Kerstjens, A., Rebentrost, P., Wittek, P.: Bayesian Deep Learning on a Quantum Computer, Quantum Mach. Intel. **1**, 4 (2019).
10. Srinivasan, K., Satyajit, S., Behera, B. K., Panigrahi, P. K.: Efficient quantum algorithm for solving travelling salesman problem: An IBM quantum experience, arXiv:1805.10928 (2018)
11. Dash, A., Sarmah, D., Behera, B. K., Panigrahi, P. K.: Exact search algorithm to factorize large biprimes and a triprime on IBM quantum computer, arXiv:1805.10478 (2018)
12. Pal, A., Chandra, S., Mongia, V., Behera, B. K., Panigrahi, P. K.: Solving Sudoku Game Using Quantum Computation, DOI: 10.13140/RG.2.2.19777.86885 (2018)
13. Mahanti, S., Das, S., Behera, B. K., Panigrahi, P. K.: Quantum Robots Can Fly; Play Games: An IBM Quantum Experience, Quantum Inf. Process. **18**, 219 (2019).

14. Behera, B. K., Seth, S., Das, A., Panigrahi, P. K.: Demonstration of Entanglement Purification and Swapping Protocol to Design Quantum Repeater in IBM Quantum Computer, Quantum Inf. Process. **18**, 108 (2019).
15. Behera, B. K., Reza, T., Gupta, A., Panigrahi, P. K.: Designing Quantum Router in IBM Quantum Computer, arXiv:1803.06530 (2018)
16. Behera, B. K., Banerjee, A., Panigrahi, P. K.: Experimental realization of quantum cheque using a five-qubit quantum computer, Quantum Inf. Process. **16**, 312 (2017)
17. Fedortchenko, S.: A quantum teleportation experiment for undergraduate students, arXiv:1607.02398 (2016)
18. Sisodia, M., Shukla, A., Thapliyal, K., Pathak, A.: Design and experimental realization of an optimal scheme for teleportation of an n-qubit quantum state, Quantum Inf. Process. **16**, 292 (2017)
19. Vishnu, P. K., Joy, D., Behera, B. K., Panigrahi, P. K.: Experimental Demonstration of Non-local Controlled-Unitary Quantum Gates Using a Five-qubit Quantum Computer, Quantum Inf. Process. **17**, 274 (2018)
20. Majumder, A., Kumar, A.: Experimental Demonstration of Non-Destructive Discrimination of Arbitrary Set of Orthogonal Quantum States Using 5-qubit IBM Quantum Computer on Cloud, arXiv:1803.06311 (2018)
21. Satyajit, S., Srinivasan, K., Behera, B. K., Panigrahi, P. K.: Nondestructive discrimination of a new family of highly entangled states in IBM quantum computer, Quantum Inf. Process. **17**, 212 (2018)
22. Sisodia, M., Shukla, A., Pathak, A.: Experimental realization of nondestructive discrimination of Bell states using a five-qubit quantum computer, Phys. Lett. A **381**, 3860 (2017)
23. Kalra, A. K., Prakash, S., Behera, B. K., Panigrahi, P. K.: Demonstration of the No-Hiding Theorem on the 5 Qubit IBM Quantum Computer in a Category Theoretic Framework, Quantum Inf. Process. **18**, 170 (2019)
24. Roy, S., Behera, B. K., Panigrahi, P. K.: Experimental Realization of Quantum Violation of Entropic Noncontextual Inequality in Four Dimension Using IBM Quantum Computer, arXiv:1710.10717 (2017)
25. Singh, R. K., Panda, B., Behera, B. K., Panigrahi, P. K.: Demonstration of a general fault-tolerant quantum error detection code for (2n + 1)-qubit entangled state on IBM 16-qubit quantum computer, arXiv:1807.02883 (2018)
26. Harper, R., Flammia, S.: Fault-Tolerant Logical Gates in the IBM Quantum Experience, Phys. Rev. Lett. **122**, 080504 (2019)
27. Willsch, D., Nocon, M., Jin, F., Raedt, H. D., Michielsen, K.: Testing quantum fault tolerance on small systems, Phys. Rev. A **98**, 052348 (2018)
28. Ghosh, D., Agarwal, P., Pandey, P., Behera, B. K., Panigrahi, P. K.: Automated error correction in IBM quantum computer and explicit generalization, Quantum Inf. Process. **17**, 153 (2018)
29. Fahdil, M. A., Azawi, A. F. A., Said, S.: Operations algorithms on quantum computer, Int. J. Comput. Sci. Netw. Secur. **10**, 1 (2010)
30. Zalka, C.: Fast versions of Shor's quantum factoring algorithm, arXiv:quant-ph/9806084v1 (1998)
31. Draper, T. G.: Addition on a quantum computer, arXiv:quant-ph/0008033 (2000)
32. Cherkas, A. V., Chivilikhin, S. A.: Quantum adder of classical numbers, J. Phys.: Conf. Ser. **735**, 012083 (2016)
33. Gossett, P.: Quantum carry-save arithmetic, arXiv:quant-ph/9808061v2 (1998)
34. Verdal, V., Barenco, A., Ekert, A.: Quantum networks for elementary arithmetic operations, Phys. Rev. A **54**, 147 (1996)
35. Florio, G., Picca, D.: Quantum implementation of elementary arithmetic operations, arXiv:quant-ph/0403048v1 (2004)
36. Álvarez-Sánchez J. J., Álvarez-Bravo J. V., Nieto L. M.: A quantum architecture for multiplying signed integers, J. Phys.: Conf. Ser. **128**, 012013 (2008)
37. Orts, F., Ortega, G., Garzon, E. M.: Quantum circuit for solving divisions using Grover's search algorithm, Proc. 18th Int. Conf. Comput. Math. Method. Sci. Eng. (2018)
38. Wang, F., Luo, M., Li, H., Qu, Z., Wang, X.: Improved quantum ripple-carry addition circuit, Sci. Chin. Inf. Sci. **59**, 042406 (2016)

39. Khosropour, A., Aghababa, H., Forouzandeh, B.: Quantum Division Circuit Based on Restoring Division Algorithm, 2011 Eighth Int. Conf. Inf. Technol.: New Gen., 1037 (2011)
40. Flynn, M.: On division by functional iteration, IEEE Trans. Comput. **C-19**, 8 (1970)
41. Schulte, M. J., Omar, J., Swartzlander Jr,, E. E.: Optimal initial approximations for the Newton-Raphson division algorithm, Computing **53**, 233 (1994)
42. Kornerup, P., Muller, J. M.: Choosing starting values for certain Newton Raphson iterations, Theor. Comput. Sci. **351**, 101 (2006)
43. Bhaskar, Mihir, K., Hadfield, Stuart, Papageorgiou, Anargyros, Petras, Iasonas: Quantum Algorithms and Circuits for Scientific Computing, arXiv:quant-ph/151108253