Nano World Journal

https://doi.org/10.17756/nwj.2023-s4-075

Research Article

Open Access

Design and Implementation of the Deutsch-Jozsa Algorithm Based Quantum Multiplier with a Reduced T-count

A. Dinesh Kumar Reddy*, P. Akash Reddy, E. Siddhartha Goud and J.V.R. Ravindra

Department of Electronics and Communication Engineering, Vardhaman College of Engineering, Hyderabad, Telangana, India

***Correspondence to:**
A. Dinesh Kumar Reddy
Department of Electronics and Communication Engineering,
Vardhaman College of Engineering,
Hyderabad, Telangana, India.
E-mail: aadineshreddy@gmail.com

Received: September 19, 2023
Accepted: November 30, 2023
Published: December 05, 2023

Citation: Reddy ADK, Reddy PA, Goud ES, Ravindra JVR. 2023. Design and Implementation of the Deutsch-Jozsa Algorithm Based Quantum Multiplier with a Reduced T-count. *NanoWorld J* 9(S4): S443-S451.

Published by United Scientific Group

Abstract

To create a more effective and efficient multiplier based on an algorithm called the Deutsch-Jozsa algorithm. Quantum circuits for fundamental operations like multiplication are needed. In this paper, we suggested a design for quantum-efficient integer multiplication. The models, based on the quantum Clifford+T gates, are T-depth and T-count optimized. Gates may be used to create quantum circuits that are leakage, however, the T-gate is exceedingly expensive to put into practice. Therefore, lowering both T-count and T-depth has emerged as crucial optimization targets. The quantity of qubits that can be used in existing quantum hardware is constrained. A unique quantum conditional adder circuit is used in quantum inspire to implement the suggested quantum multiplier architecture, which lowers the T-count. The conditional adder is also changed to a Toffoli gate when one of its operands is zero.

Keywords

Toffoli gate, Control adder, Array multiplier, Quantum multiplier, Deutsch-Jozsa algorithm, Grover's algorithm, Nanotechnology

Introduction

Quantum computing and nanotechnology appears to be one of the most promising new computer paradigms due to its applications in research processing, discovery, encryption, and numerical methods [1-4]. Quantum computing can provide computational power and algorithms to simulate and optimize nanoscale phenomena and systems, such as molecular dynamics, nanoelectronics, and nanomaterials. Together, they can enable new functionalities and applications that are impossible or impractical with conventional technologies. In several of these domains' quantum computations, integer mathematical operations like add, reduction, and multiplication are performed using quantum circuits. For usage with quantum computer languages like Quipper [5] plus LIQU [6] and cQASM, python, Qiskit, and GIT, scientists have developed specific packages of core quantum numerical arithmetic methods, as well as tools for designing quantum computers like those suggested in [6] and [7]. Quantum gates are devices that can be used to do quantum computation. There is a 1-to-1 mapping between the output and input vectors of every quantum circuit. Ancillae have been any stable inputs in a quantum circuit. Every output that would be necessary for the quantum circuit to maintain 1-to-1 mapping but does not constitute a significant input nor a desirable output is referred to as garbage output. The results that resulted from the inputs are not thought of as wasteful results [8]. Circuit overhead should be maintained to a minimum, including ancillae and garbage outputs. Quantum nanoscience is the basic research area at the intersection of nanoscale science and quantum science that creates the understanding that enables development of nanotechnologies. It uses quantum mechanics to explore and use coherent quantum effects in engineered nanostructures.

Researchers are becoming more interested in the construction of quantum circuits with fault tolerance as a result of the noise faults that can occur in physical quantum computers [8, 9]. To get beyond the restrictions imposed, it is possible to use failure quantum error-correcting codes and quantum gates.

The quantum T-gate's increased noise tolerance, however, it is accompanied by an increase in implementation costs [10-12]. Due to the greater cost to implement the T-count, T-gate has evolved as a crucial performance parameter for the error-resistant design of quantum circuitry.

In the papers, the designing of quantum numeric multiplication circuits has received a lot of interest. Garbage-free architectures, similar to those in [13] are built on several examples of highly T-counted circuits for quantum condition adders. The recurrent use of the conditional adder circuit in these designs will increase the multiplication circuit's T-gate cost. Other publications, like the one in [14], show T-gate efficiency architectures but do not factor in the extra ancillae plus T-gate expenses associated with removing trash outputs from the cost estimates.

Although the circuits for integer multiplication reported in earlier publications like [7] and [13] are attractive concepts, their fault-tolerant implementations are unworkable due to a high T-count expense.

This study includes proposing the design of a garbage-free multiplier with 4.n+1 Qubits, where n = number of bits for multiplying. the suggested multiplier can be designed using the Toffoli gate which contains 4-T gates and a control adder that has no input carry and is garbage-less T-count optimization is the foundation of a quantized numeric multiplication circuit and the design is implemented by an algorithm using cQASM programming language in Quantum Inspire Tool. Comparing the proposed design with other ones, it can be shown that it performs better in terms of T-count and algorithm level.

Performance evaluation of quantum circuits

Since the T-gate's error-tolerant implementation costs significantly more than those of the various Clifford+T gates' failure implementations, it is interesting to assess quantum network effectiveness in terms of T-count [10-12]. The overall number of T-gates or symmetric transposes of a T-gate within a quantum circuit is known as the T-count. The count of the T-gate for the Clifford+T design is 4.

Researchers are looking at techniques for integer multiplication. Numerous multiplication methods, including Booth's method, shifting and add, as well as Karatsuba's algorithm, have been devised by researchers. In this research, we are showing a quantum implementation of the T-count-optimized shifting and introduce the quantum inspire multiplication approach.

Let's consider multiplying 2n-bit values, X and Y. The shifting and adding multiplication method gives the product R of multiplying the two integers X and Y at the conclusion of processing. Regarding the X-digit number multiplied by the number Y, the stages of the shifting and add algorithm for multiplying are shown.

Procedure

```
R = 0
N = number of bits
for P in range(0, N):
R = R + (X^Y)*(2**i)
print(R)
```

Literature review

Inside the literature, the architecture of quantum circuits for multiplying integers has drawn a lot of interest. The majority of efforts, however, focus on versatile computing and have high production costs for garbage. Clifford+T gate prices in works like [10] and [15] that provide in the study of the quantum Fourier series, multiplication circuits are unreasonably expensive.

For example, the T-gate value of the quantum multiplier design in [13] is on the order O.(n3). The characteristics of the quantum multipliers in [12] and [13] are shown in table 1. Quantum multiplier circuits are shown in designs like those in [6, 9, 10] that can be realized with far fewer quantum gates. The characteristics of the quantum multipliers in [9] and are shown in table 1. The Square Root of Not, Pauli gates, and Toffoli gates may be attained using this technique.

These known quantum arithmetic multiplication circuits employ the Fredkin, Toffoli. The current quantum conditional addition circuits provide the foundation for the quantum integer multiplication circuits described in works of [12] and [15]. A conditional adder circuit based on the architecture described in [19] is applied to create the multiplication circuit presented in [8]. The quantum multiplier circuit architecture is described in [20] and consists of a unique quantum conditional adder circuit.

Based on a two-step method, the multiplication circuits in [10] and [15] multiply the numbers: Create all partial products, and then use adders to merge them all. A novel quantum plus circuit and a complete quantum adder circuit makes up the architecture shown in [8]. CNOT gate and the square of not gates are used to build the quantum AND circuit and quantum full adder. To reduce the depth of a circuit, the intermediate products are produced simultaneously. Toffoli gates and other devices are used in the design shown in [13]. To decrease circuit depth, the intermediate products are built concurrently with a series of Toffoli gates. Examples of quantum ripple carryover adders are shown in [15] and are employed to complete the product addition partially depth of the intermediate product addition is lowered using a design strategy formed on a bit's addition tree, which is similar to the idea displayed in [13]. Despite being depth-optimized, the systems have large ancillae and garbage output expenses. The multiplication circuits which are only for integers described in articles [11] and [13] have interesting designs, but they have a high T-count cost (Table 1).

Theory about Deutsch-Jozsa algorithm

In the Deutsch-Jozsa method, we use an oracle code to determine whether a binary function is correct. A basic quan-

Table 1: Overview of the quantum multipliers currently in use.

Work	Lin et al. [11]	Jayashree et al. [16]	Kotiyal et al. [17]	Babu [18]
Algorithm	Add and rotate	Shift and add	2 step algorithm	2 step algorithm
Modules	Conditional adder, swap gates	Conditional adder, swap gates	Partial product circuit, adder tree	Partial product circuit, adder tree
Quantum gates	CNOT, Toffolo gate, Fredkin gate	CNOT, Toffolo gate	CNOT, Toffolo gate, Peres gate	CNOT, square root of not gate
Qubit gates	O(n)	O(n)	O(n)	NA
T-count	O(n^2)	O(n^2)	O(n^2)	O(n^2)

tum method that may be used to accelerate a search is the Deutsch-Josza algorithm. It can determine whether a function possesses a particular attribute, as will be detailed below (being balanced). In order to do this, with such a quantum computer, the function must only be called once, whereas a conventional approach would require two calls. When an operation requires a lot of processing resources, for example, it might be quite advantageous to compute it only once rather than twice.

A polynomial or even exponential speedup over conventional algorithms may be achieved by other quantum algorithms that use the same quantum mechanical phenomena, despite the fact that the speedup of this particular technique is just a factor of two. Figure 1 presents circuit diagram of Deutsch-Jozsa algorithm.

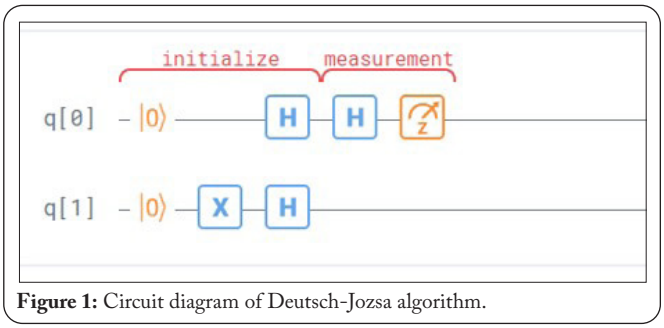

Figure 1: Circuit diagram of Deutsch-Jozsa algorithm.

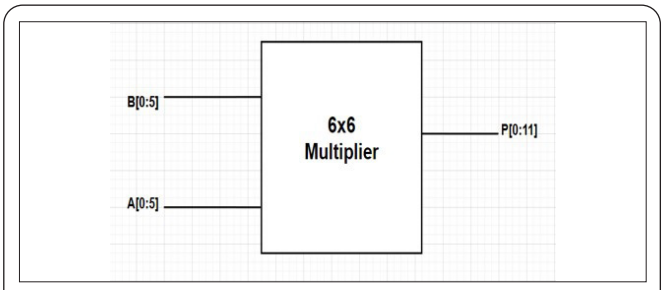

Figure 2: Block diagram of a multiplier.

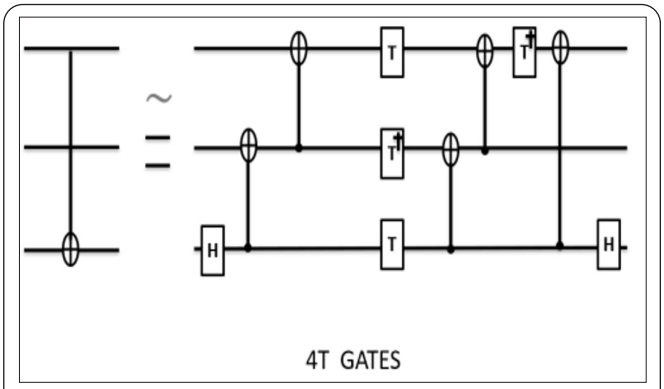

Figure 3: Implementation of Toffoli gate with Clifford+T gates.

Experimentation

Design of a proposed method

P = A * B's multiplication matrix. In this diagram, the 2n-bit product is created by multiplying the n-bit multiplicand, A, by the n-bit multiplier, B [6, 9].

A = A0 to An-1

B = B0 to Bn-1

P = P0 to P2n-1

In figure 2, it's clearly, we deduced a generic formula to calculate the value of its products when A and B's input values are known. We next examined other examples of the process of multiplying different values of n. that are shown in table 2.

Algorithm for a Toffoli gate

To write oracle code for Toffoli gate (Figure 3) by using a Deutsch-Jozsa algorithm.

- VERSION 1.0
- #INITIALIZE THE NUMBER OF QUBITS REQUIRED

Table 2: Procedure for multiplication (n = 6).

						B_5	B_4	B_3	B_2	B_1	B_0
					x	A_5	A_4	A_3	A_2	A_1	A_0
	B_5A_5	B_4A_5 B_5A_4	B_3A_5 B_4A_4 B_5A_3	B_2A_5 B_3A_4 B_4A_3 B_5A_2	B_1A_5 B_2A_4 B_3A_3 B_4A_2 B_5A_1	B_0A_5 B_1A_4 B_2A_3 B_3A_2 B_4A_1 B_5A_0	B_0A_4 B_1A_3 B_2A_2 B_3A_1 B_4A_0	B_0A_3 B_1A_2 B_2A_1 B_3A_0	B_0A_2 B_1A_1 B_2A_0	B_0A_1 B_1A_0	B_0A_0
P_{11}	P_{10}	P_9	P_8	P_7	P_6	P_5	P_4	P_3	P_2	P_1	P_0

- QUBITS 3
- #SET ALL QUBITS TO 0
- PREP_Z Q[0:2]
- {X Q[0] | |X Q[1]}
- {H Q[2]}
- .ADD
- CNOT Q[2], Q[1]
- CNOT Q[1], Q[0]
- T Q[0]
- TDAG Q_[1]
- T Q[2]
- CNOT Q[2], Q[1]
- CNOT Q[1], Q[0]
- {TDAG Q[0]}
- CNOT Q[2],Q[0]
- H Q[2]
- .MEASUREMENT
- MEASURE Q[0,1,2]

Circuit design for quantum conditional addition without input carry

Considering the outline, the layout of the suggested input-free quantum conditional adder circuit (Figure 4). The design doesn't produce any trash. Comparing the suggested design to existing design strategies that don't produce garbage, the suggested design has a less T-count. The paper [21] provided a detailed explanation of the design stages.

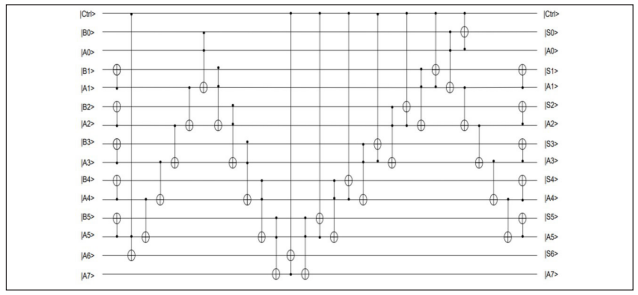

Figure 4: Control adder circuit.

Algorithm for a control adder

- VERSIONi 1.0
- QUBITS 15
- PREP_Z Q[0:14]
- .INIT
- {X Q[0] | X Q[4] | X Q[5] | X Q[8] | X Q[9]}
- .ADD
- CNOT1 Q[4],Q[3]
- CNOT2 Q[6],Q[5]
- CNOT3 Q[8],Q[7]
- CNOT4 Q[10],Q[9]
- CNOT5 Q[12],Q[11]
- TOFFOLI1 Q[0],Q[12],Q[13]
- CNOT6 Q[10],Q[12]
- CNOT7 Q[8],Q[10]
- CNOT8 Q[6],Q[8]
- CNOT9 Q[4],Q[6]
- TOFFOLI2 Q[1],Q[2],Q[4]
- TOFFOLI3 Q[3],Q[4],Q[6]
- TOFFOLI4 Q[5],Q[6],Q[8]
- TOFFOLI5 Q[7],Q[8],Q[9]
- TOFFOLI6 Q[10],Q[11],Q[13]
- TOFFOLI7 Q[11],Q[12],Q[14]
- TOFFOLI8 Q[0],Q[14],Q[13]
- TOFFOLI9 Q[11], Q[12], Q[14]
- TOFFOLI10 Q[0] ,Q[12], Q[11]
- TOFFOLI11 Q[9], Q[10], Q[12]
- TOFFOLI12 Q[0], Q[10], Q[9]
- TOFFOLI13 Q[7], Q[8], Q[10]
- TOFFOLI14 Q[0], Q[8], Q[7]
- TOFFOLI15 Q[5], Q[6], Q[7]
- TOFFOLI16 Q[0], Q[6], Q[5]
- TOFFOLI17 Q[3], Q[4], Q[6]
- TOFFOLI18 Q[0], Q[4], Q[3]
- TOFFOLI19 Q[1], Q[2], Q[4]
- TOFFOLI20 Q[0], Q[2], Q[1]
- CNOT10 Q[4],Q[6]
- CNOT11 Q[6],Q[8]
- CNOT12 Q[8],Q[10]
- CNOT13 Q[10],Q[12]
- CNOT14Q[4],Q[3]
- CNOT15 Q[6],Q[5]
- CNOT16 Q[8],Q[7]
- CNOT17 Q[10],Q[9]
- CNOT18 Q[12],Q[11]
- .MEASUREMENT
- MEASURE Q[1,3,5,7,9,11,13]

Note: Here we had given the numbering for CNOT and Toffoli gates for readability purposes.

T-count

The suggested quantum Ctrl_Add circuit for an n-bit size has the following total T-count: 12.n8.

Ancillae cost

For each phase of the suggested design process, a brief

Table 3: Comparison of quantum ctrl_add circuit.

Parameters	1	2	3	4	Proposed
T_Count	56.N	28.N+7	21.N+7	28.N+7	12.N+8
Qubits	4.N+2	2.N+3	2.N+3	2.N+3	2.N+3
Ancillae	2	2	2	1	2

Note: Lin et al. creation is number 1 [11]. Jayashree et al. design number 2 [16], Thapliyal et al. design number 3 [21], Markov and Shi design is number 4 [22].

illustration of the necessary ancillae is offered for the proposed quantum control adder circuit. By adding the ancillae needed for each stage of the suggested design technique, we determine the total expenses associated with the projected control adder circuit. An n-bit suggested quantum Ctrl_Add circuit has a total of two ancillae (Table 3).

Design of a suggested quantum arithmetic multiplication circuit

Compared to the design methodologies currently used, which do not contain garbage outputs, the suggested quantum multiplication circuit for integers is constructed lacking garbage outputs plus a reduced T-count. The suggested quantum control adder. The quantum arithmetic multiplication circuit is according to the Toffoli gate array and the circuit and without any input carry. the total circuit operation explained in [21].

- Step - 1: In this step, we will get each Toffoli gate output which is input to the control adder as shown in step 2.
- Step - 2: In this step, the outputs of Toffoli gates are given to the control adder circuit. The operation is clearly explained in [21].
- Step - 3: Similarly, by keeping side-by-side control adders we will get the output of the multiplier.

The following steps are shown in the form of a circuit (Figure 5). The code will be split into five blocks and initialized using the Toffoli gate and control adder technique for this multiplier circuit. The multiplier code operates similarly and is based on the following process initialized.

Algorithm for a multiplier circuit

- Set up the necessary qubits.
- Next, use Toffoli gates in accordance with the logic.
- Use the control adder algorithm next.
- Carry out this step five more times.
- Evaluate the output of the multiplier.

Cost analysis

T-gate cost

For each phase of the suggested design process, a brief illustration of the T-count of the suggested quantum arithmetic multiplication circuit is provided (Table 4). By adding the T-counts for every stage of the suggested design process, we determine the overall account for the suggested quantum arithmetic multiplication circuit (Table 5). The suggested quantum arithmetic multiplication circuit's overall T-count is as follows.

12.n^2-24.n+8

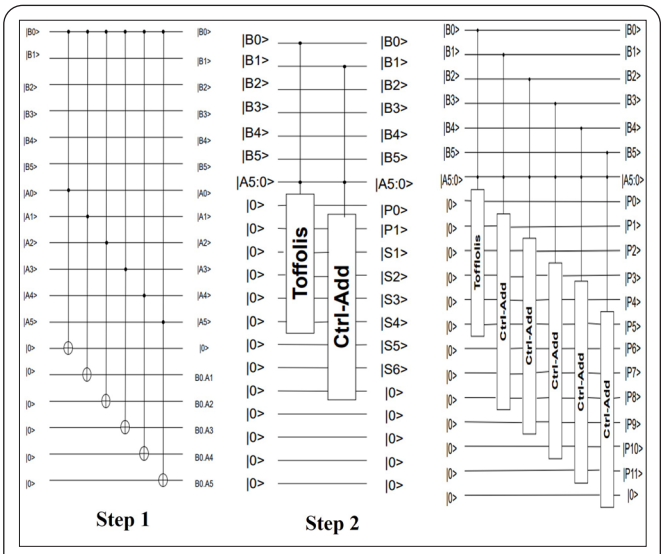

Figure 5: Creation of a six-qubit quantum computer, Ctrl-Add circuit without input carry: steps 1 and 2.

Table 4: Circuits for quantum arithmetic multiplication are compared.

Parameters	1	2	3	4	Proposed
T_Count	56.N^2	28.N^2+7.n	42.N^2	21.N^2-14	12.N^2-24.N+8
Qubits	5.N+1	4.N+1	NA	4.N+1	4.N+1
Ancillae	3.N+1	2.N+1	NA	2.N+1	2.N+1

Note: The design by Lin et al. is number 1 [11], The design by Jayashree et al. is number 2 [16], 3 is a modified version of Babu's [18] design that doesn't produce waste, 4 is a modified version of Thapliyal et al. [21] design that doesn't produce garbage.

Ancillae cost

The suggested quantum arithmetic multiplication circuit's necessary ancillae are briefly presented for each stage of the suggested design process. We add the ancillae needed for each stage of the suggested design process to determine the sum of ancillae for the suggested quantum arithmetic multiplication circuit. The suggested quantum arithmetic multiplication circuit's whole ancillae are listed as follows.

2.n+1

Results and Discussion

Here, presenting a single result circuit diagram, in which we have included numerous metrics concerning the efficiency of the quantum circuit following the use of the provided approach. In this article, we offer one design approach for a high-speed multiplier that is fault tolerant. The results were verified in quantum inspire, Cadence tool, and Verilog simulations.

Toffoli gate output

The inputs are Q[0]=1,Q[1]=0,Q[2]=1 (Figure 6).

Then Toffoli gate output is Q[0]=1,Q[2]=0,Q[2]=0

(Figure 7).

Here we tested all eight combinations. The outputs were verified.

Table 5: Qubit multiplication circuits' comparability in T-counts.

N	1	2	3	4	Proposed	%Impr w.r.t. 1	%Impr w.r.t. 2	%Impr w.r.t. 3	%Impr w.r.t. 4
4	896	476	528	476	322	64	32	39	32
8	3584	1848	2352	2240	1330	62	28	43	40
16	14336	7280	10032	9716	5362	62	26	46	44
32	57344	28896	41520	40600	21490	62	25	48	47
64	229376	115136	169008	166460	86002	62	25	49	48
128	917504	459648	682032	675360	344050	62	25	49	49
256	3670016	1836800	2740272	2723588	1376242	62	25	49	49
512	14680064	7343618	10985520	10945256	5505010	62	25	49	49
1024	58720256	29367296	43991088	43896524	22020082	62	25	49	49
2048	234881024	117454548	176062512	175845040	88080370 **Average:**	62 **62**	26 **26**	49 **47**	49 **46**

Note: 1 is Lin et al. [11], 2 is Jayashree et al. [16], 3 is Babu [18], 4 is Thapliyal et al. [21].

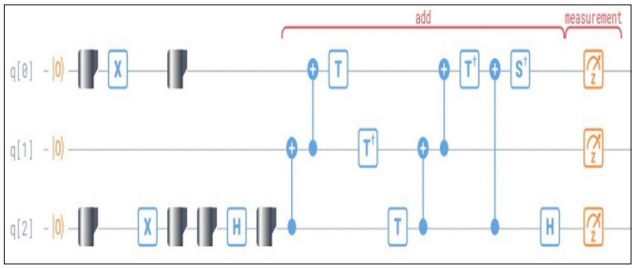

Figure 6: Implementation of Toffoli gate in quantum inspire.

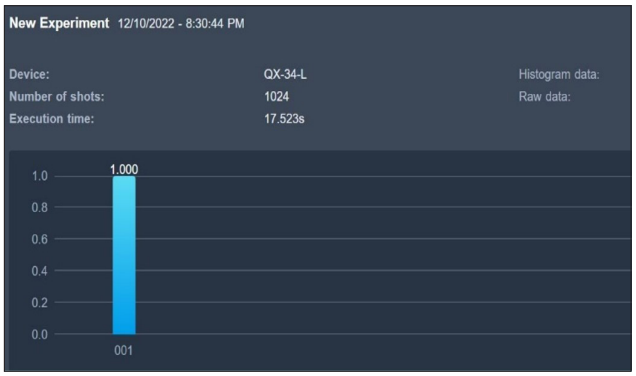

Figure 7: Result of the Toffoli gate.

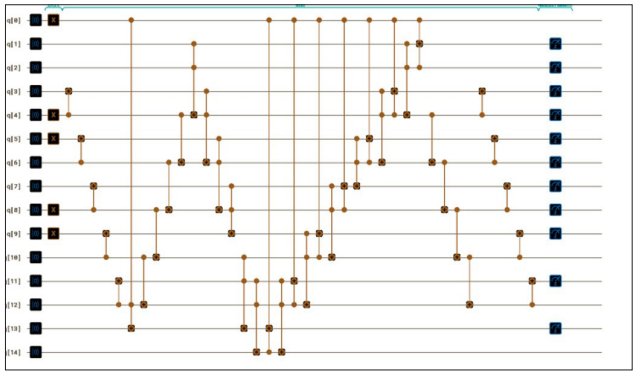

Figure 8: Implementation of Control Adder circuit in quantum inspire.

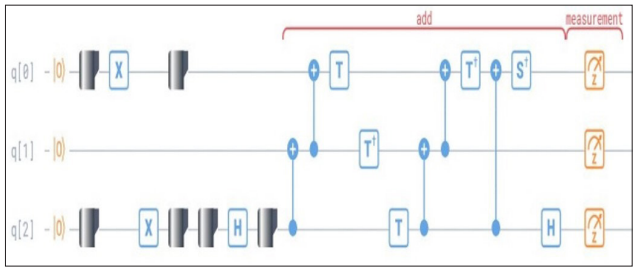

Figure 9: The output of the control Adder.

Control adder

Let's have a look at a 6 x 6 multiplication matrix (n = 6) where the input comes from two different references A and B.

We should give an illustration of how the outcomes of the generalized formula are obtained. the expression that would be consistent with results produced by computers or other quantitative approaches like [7] would then be acquired after crossing the carry, we would proceed as before.

- Keep the ctrl bit set to one.
- As stated, A and B have the following values:
- A = 001010 = 10
- B = 010100 = 20
- The output given by the control adder is 30.
- i.e., 0 x 0 x 1 x 1 x 1 x 1 x 0

After writing the code by using Deutsch-Jozsa algorithm, quantum inspire generated the below circuit and the outputs are verified (Figure 8 and figure 9).

Multiplier

Let's have a look at a 6 x 6 multiplication matrix (n = 6) where the input comes from two different sources, A and B.

We would provide an example to demonstrate how the generalized formula was obtained and how to calculate the product values. After that, we would proceed with passing the carry until we had the product term that matched the numbers produced by arithmetic or software algorithms like [8]. As stated, A and B have the following values:

A = 010001 = 17

B = 011100 = 28

P = 00011101110

The output given by the multiplier is 476, which will be given in the form of hexadecimal i.e., 0x1DC. The circuit was given in figure 10 and figure 11.

The outputs were verified in Verilog simulations also for writing a Verilog code that will be dumped in the cadence tool and power, delay, and area calculations were calculated. The performance of the 6 x 6 multiplier for signed bits was determined in this work and implemented. With Verilog serving as the hardware description language, we synthesized these multipliers utilizing cadence software. The analysis is done on the radix6 schematic circuits and waveforms [6]. The transistor-level circuit is created by cadence software in the accurate multiplier of radix-6, which has 12 input bits and 12 output bits (Figure 12).

The window for the 6-bit quantum multiplier. The cadence software generated the power (Figure 13), delay (Figure 14), and area (Figure 15) reports.

Here we are not comparing with any other multipliers because there are no previous multipliers that are done in quantum IBM or quantum inspire.

We also noted that the kind of multiplier being employed will have an impact on multiplier performance. Although rapid adders will operate well, they may be used to speed up the adding process by positioning them close to the center, while slower adders can be employed close to the regions with the least and greatest significance. As a result, the speed will be preserved, and less hardware and power will be consumed. So here we are using an efficient control adder which already existing adders implemented in Deutsch-Jozsa algorithm. When compared to classical algorithms, quantum algorithms were developed to solve classical problems with fewer steps the Shor algorithm, the Grover algorithm, and others are a few of the highly well-known algorithms.

Grover's method is essentially a search technique that is used to discover the input in a collection of functions in order to produce a specific output. It has several uses while looking over a big database.

The Deutsch-Jozsa algorithm is an exponentially quicker quantum deterministic algorithm than the conventional algorithm (Figure 16). There is a drastic change in speed and performance (Figure 17).

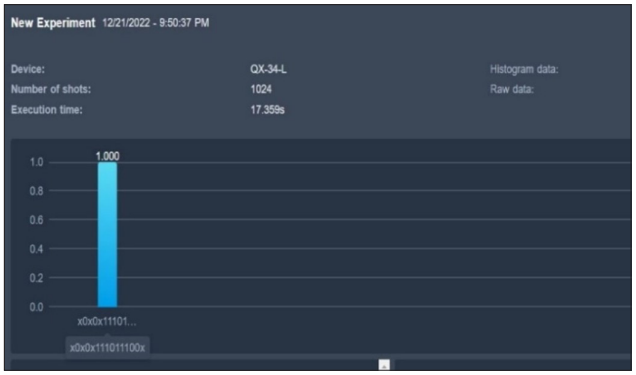

Figure 10: The output of the 6-bit quantum multiplier.

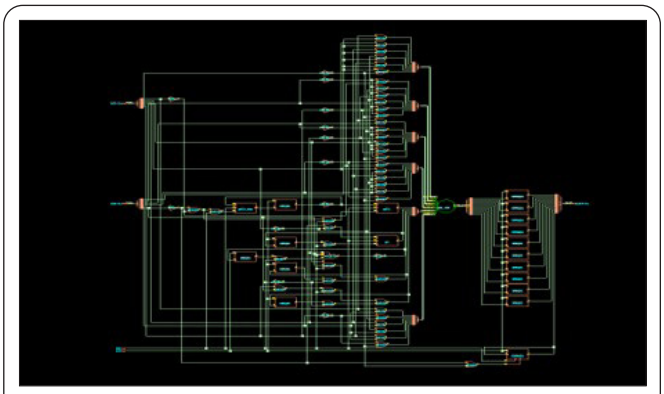

Figure 11: There are displayed the schematic diagram for the 6-bit quantum multiplier.

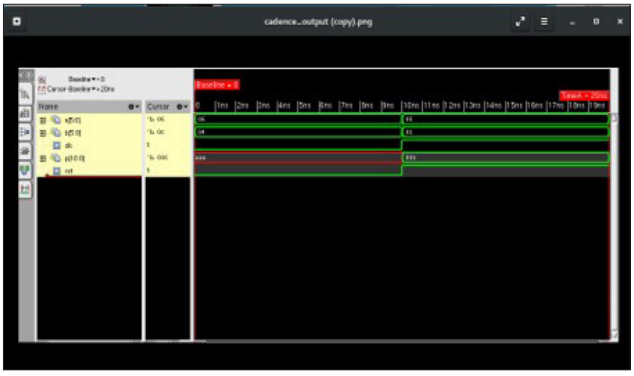

Figure 12: There are displayed the simulation.

```
============================================================
  Generated by:           Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:           Dec 03 2022  01:54:39 pm
  Module:                 multiplier
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================

                                  Leakage    Dynamic     Total
        Instance          Cells Power(nW)  Power(nW)  Power(nW)
------------------------------------------------------------------
multiplier                  135  5103.997 164184.056 169288.053
  csa_tree_add_40_6_groupi   64  2683.279  39289.658  41972.937
```

Figure 13: Power report.

```
============================================================
  Generated by:           Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:           Dec 03 2022  01:54:39 pm
  Module:                 multiplier
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================

Path 1: MET (5 ps) Setup Check with Pin p_reg[10]/CK->D
          Group: clk
     Startpoint: (F) rst
          Clock: (R) clk
       Endpoint: (R) p_reg[10]/D
          Clock: (R) clk

                    Capture       Launch
      Clock Edge:+     3000            0
      Src Latency:+       0            0
      Net Latency:+       0 (I)        0 (I)
          Arrival:=    3000            0

            Setup:-     196
      Uncertainty:-      10
    Required Time:=    2794
     Launch Clock:-       0
     Input Delay:-     1000
       Data Path:-     1789
             Slack:=      5

Exceptions/Constraints:
  input_delay             1000            constraints_top.sdc_line_5
```

Figure 14: Delay report.

```
Generated by:          Genus(TM) Synthesis Solution 17.22-s017_1
Generated on:          Dec 03 2022  01:54:39 pm
Module:                multiplier
Operating conditions:  slow (balanced_tree)
Wireload mode:         enclosed
Area mode:             timing library

      Instance              Module          Cell Count  Cell Area  Net Area  Total Area  Wireload
multiplier                                        135    1027.113     0.000    1027.113  <none> (D)
  csa_tree_add_40_6_groupi csa_tree_add_40_6_group_7       64     584.327     0.000     584.327  <none> (D)

(D) = wireload is default in technology library
```

Figure 15: Area report.

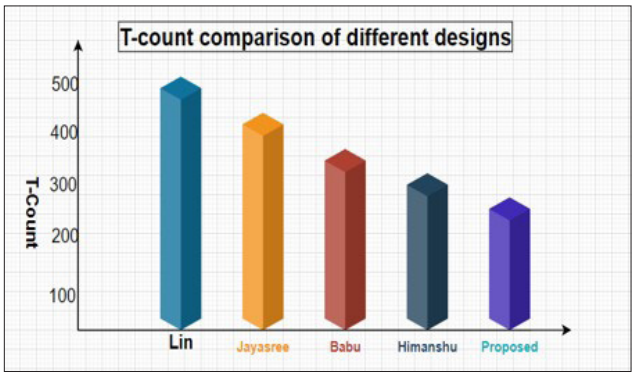

Figure 16: Analysis of previous works.

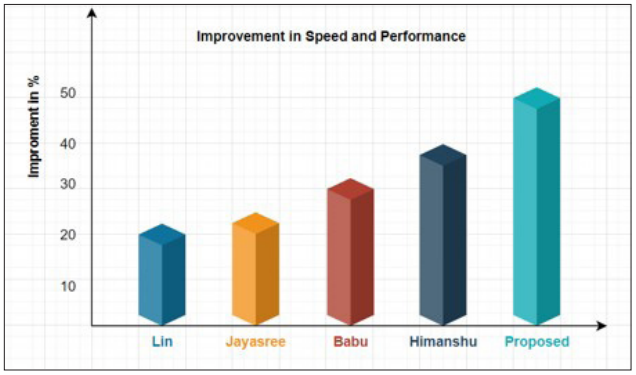

Figure 17: Improvement in speed and perform.

Conclusion

In this paper, we offer two different forms of enhancements for integer quantum circuit multiplication, one that utilizes Clifford+T gates and the other on a Deutsch-Jozsa algorithm, which both increase the multiplier's speed and efficiency. It also illustrates the architecture of the quantum conditional add-on circuit functionality and the quantum adder, two subcomponents utilized in the suggested quantum integer multiplier circuits. In count of T-depth and T-count, the suggested quantum multiplier for integers circuits is demonstrated to outperform already existing models. It is also a quick multiplier. We get to the conclusion that the suggested multiplier circuit may be incorporated into a wider quantum data route system design, where creating a Deutsch-Jozsa algorithm and focusing on T-depth and T-count are very important parameters.

Acknowledgements

None.

Conflict of Interest

None.

References

1. Cheung D, Maslov D, Mathew J, Pradhan DK. 2008. On the Design and Optimization of a Quantum Polynomial-Time Attack on Elliptic Curve Cryptography. In Kawano Y, Mosca M (eds) Theory of Quantum Computation, Communication, and Cryptography. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, pp 96-104.
2. van Dam W, Shparlinski IE. 2008. Classical and Quantum Algorithms for Exponential Congruences. In Kawano Y, Mosca M (eds) Theory of Quantum Computation, Communication, and Cryptography. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, pp 1-10.
3. Zilic Z, Radecka K. 2007. Scaling and better approximating quantum Fourier transform by higher radices. *IEEE Trans Comput* 56(2): 202-207. https://doi.org/10.1109/TC.2007.35
4. Fredkin E, Toffoli T. 1982. Conservative logic. *Int J Theor Phys* 21(3-4): 219-253. https://doi.org/10.1007/BF01857727
5. Webster P, Bartlett SD, Poulin D. 2015. Reducing the overhead for quantum computation when noise is biased. *Phys Rev A* 92(6): 062309. https://doi.org/10.1103/PhysRevA.92.062309
6. Zhou X, Leung DW, Chuang IL. 2000. Methodology for quantum logic gate construction. *Phys Rev A* 62(5): 052316. https://doi.org/10.1103/PhysRevA.62.052316
7. Polian I, Fowler AG. 2015. Design automation challenges for scalable quantum architectures. In Proceedings of the 52nd Annual Design Automation Conference, San Francisco, California, USA.
8. Amy M, Maslov D, Mosca M. 2014. Polynomial-time T-depth optimization of Clifford+T circuits via matroid partitioning. *IEEE Trans Comput Aid Des Integr Circuits Syst* 33(10): 1476-1489. https://doi.org/10.1109/TCAD.2014.2341953
9. Paler A, Polian I, Nemoto K, Devitt SJ. 2017. Fault-tolerant, high-level quantum circuits: form, compilation and description. *Quantum Sci Technol* 2(2): 025003. https://doi.org/10.1088/2058-9565/aa66eb
10. Devitt SJ, Stephens AM, Munro WJ, Nemoto K. 2013. Requirements for fault-tolerant factoring on an atom-optics quantum computer. *Nat Commun* 4(1): 2524. https://doi.org/10.1038/ncomms3524
11. Lin CC, Chakrabarti A, Jha NK. 2014. Qlib: quantum module library. *ACM J Emerg Technol Comput Syst* 11(1): 1-20. https://doi.org/10.1145/2629430
12. Florio G, Picca D. 2004. Quantum implementation of elementary arithmetic operations. *arXiv preprint* 1-11. https://doi.org/10.48550/arXiv.quant-ph/0403048
13. Gosset D, Kliuchnikov V, Mosca M, Russo V. 2013. An algorithm for the T-count. *arXiv preprint* 1-12. https://doi.org/10.48550/arXiv.1308.4134
14. Sowmya G, Sowmya SS, Ravindra JVR. 2023. Efficient RNS Realization of High-Speed Arithmetic Multiplier with Respect to Cryptographic Computation. In Ranganathan G, Fernando X, Rocha Á (eds) Inventive Communication and Computational Technologies. Lecture Notes in Networks and Systems. Springer, Singapore, pp 13-21.
15. Ruiz-Perez L, Garcia-Escartin JC. 2017. Quantum arithmetic with the quantum Fourier transform. *Quantum Inf Process* 16: 152. https://doi.org/10.1007/s11128-017-1603-1
16. Jayashree HV, Thapliyal H, Arabnia HR, Agrawal VK. 2016. Ancilla-input and garbage-output optimized design of a reversible quantum integer multiplier. *J Supercomput* 72: 1477-1493. https://doi.org/10.1007/s11227-016-1676-0

17. Kotiyal S, Thapliyal H, Ranganathan N. 2014. Circuit for reversible quantum multiplier based on binary tree optimizing ancilla and garbage bits. In 27th International Conference on VLSI Design and 13th International Conference on Embedded Systems, Mumbai, Maharashtra, India.

18. Babu HMH. 2017. Cost-efficient design of a quantum multiplier–accumulator unit. *Quantum Inf Process* 16: 30. https://doi.org/10.1007/s11128-016-1455-0

19. Montanaro A. 2017. Quantum pattern matching fast on average. *Algorithmica* 77: 16-39. https://doi.org/10.1007/s00453-015-0060-4

20. Jones NC, Van Meter R, Fowler AG, McMahon PL, Kim J, et al. 2012. Layered architecture for quantum computing. *Phys Rev X* 2(3): 031007. https://doi.org/10.1103/PhysRevX.2.031007

21. Thapliyal H, Muñoz-Coreas E, Khalus V. 2021. Quantum circuit designs of carry lookahead adder optimized for T-count T-depth and qubits. *Sustain Comput Inf Syst* 29: 100457. https://doi.org/10.1016/j.suscom.2020.100457

22. Markov IL, Shi Y. 2008. Simulating quantum computation by contracting tensor networks. *SIAM J Comput* 38(3): 963-981. https://doi.org/10.1137/050644756