IJNRD.ORG ISSN : 2456-4184

INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD) | IJNRD.ORG

An International Open Access, Peer-reviewed, Refereed Journal

Quantum Gates Implementation for Combinational Circuits using Bloch Sphere

[1]Kanaga Nandhini P, [2]Sandhya Devi M, [3]Dr.V.Jeyalakshmi

[1]PG Student, [2]PG Student, [3]Professor

[1]Electronics and Communication Engineering,

[1]College of Engineering, Guindy, Chennai, Tamil Nadu - 600025

Abstract : Quantum is an emerging technology in future computers. It depends upon the basic properties of quantum physics and principles of classical systems. This leads a way to develop systems to solve complex problems that a classical system cannot do. In this project, a web page is designed that will visualize the position of qubits in quantum gates within a Bloch sphere. After that, the basic quantum gates are implemented in Visual Studio Code (VSC) using qiskit and the best gate suitable for classical gate implementation is selected, which is Pauli X gate. In VSC, the programming language used is Python. Then, by using that gate, the classical gates and combinational circuits such as half adder and full adder are implemented in qiskit and the outputs were verified using the truth table.

***IndexTerms* - Classical gates, Quantum, Quantum gates, Qiskit**

I. Introduction

All the complex variables, objects and data structures used in modern software are basically all just big piles of bits called as classical variables and are executed in classical computers. In quantum computers, the basic variable isthe qubit*:* a quantum variant of the bit [1]. A qubit (or quantum bit) is the quantummechanical analogue of a classical bit. A qubit is a two-level quantum system where the two basis qubit states are usually written as $|0\rangle$and $|1\rangle$ [2]. A qubit can bein state $|0\rangle$, $|1\rangle$ or (unlike a classical bit) in a linear combination of both states (Superposition) [3].

In quantum computing, the Bloch sphere is a geometrical representationof the pure state space of a two-level quantum mechanical system (qubit) [4]. The Bloch sphere is a unit 2-sphere. The north and south poles of the Bloch sphere are typically chosen to correspond to the standard basis vectors |0> and |1>respectively, which in turn might correspond e.g. to the spin-up and spin- down states of an electron. The points on the surface of the sphere correspond to the pure states of the system, whereas the interior points correspond to the mixed states [5].

One consequence of having a universal set of quantum gates is the abilityto reproduce any classical computation [6]. We simply need to compile the classicalcomputation down into the Boolean logic, and then reproduce these on a quantum computer [7]. This demonstrates that they can do anything that a classicalcomputer can do, and they can do so with at least the same computational complexity.

II. LITERATURE REVIEW

A. Implementation of Reversible Logic Gates with Quantum Gates

Dr. Mummadi Swathi and Dr. Bhawana Rudra have presented the basic reversible logic gates and quantum gates. Quantum is an emerging technology in future computers. Reversibility is the main advantage of quantum computers.. In this paper, they have discussed various reversible logic gates like Feynman,Toffoli, R, Peres and TR gates using basic quantum gates like CNOT, Pauli, Swap gates and implemented them using IBM quantum experience. Each quantum circuitis implemented using basic gates like CNOT, Swap, Pauli gates and state vectors for each circuit are obtained and results are plotted..

B. Implementation of Quantum Gates based Logic Circuits using IBM Qiskit

Quantum computing is an emerging field that depends upon the basic propertiesof quantum physics and principles of classical systems. In this article, Enaul haq Shaik and Nakkeeran Rangaswamy have presented simple methods to implement logic circuits using quantum gates. Logic gates and circuits are defined with quantum gates using Qiskit in Python. Later, they are verified with quantum circuits created by using IBM Quantum. In this article, the implementation of logic circuits with quantum gates was presented. Logic circuits defined using Qiskit were verified with IBM Quantum simulator. Further, a simple method of instantiating these quantum gate based logic circuits was proposed to realize high-end logic functions.

III. PROPOSED METHODOLOGY

3.1 Visualization of qubits in Bloch sphere

The Bloch sphere is a useful visualization tool in quantum mechanics for understanding the behavior of qubits. It is a three-dimensional sphere, where the north pole and south pole represent the states |0⟩ and |1⟩, respectively, and any pointon the surface of the sphere represents a superposition of the two states. The Bloch sphere provides an intuitive way to visualize the behavior of qubits in quantum computing and is a useful tool for understanding the effects ofquantum gates on qubit states. Here, the GUI window will be the front end which contains the notations of all the quantum gates that we can visualize and also the other additional notations. Fig.1(a) shows the GUI window created using tkinter and Fig.1 (b) shows the appearance of a bloch sphere at its initial state.

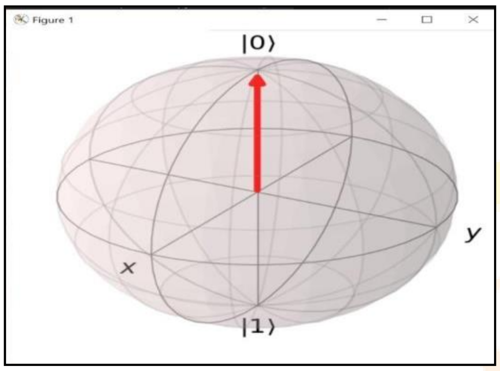
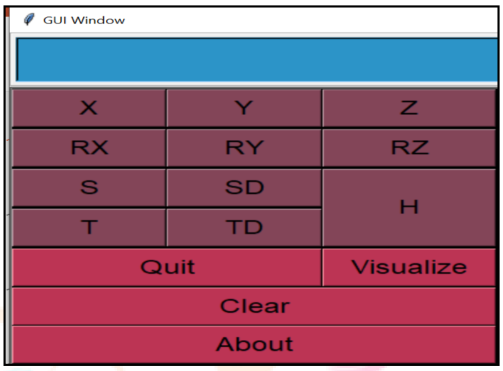


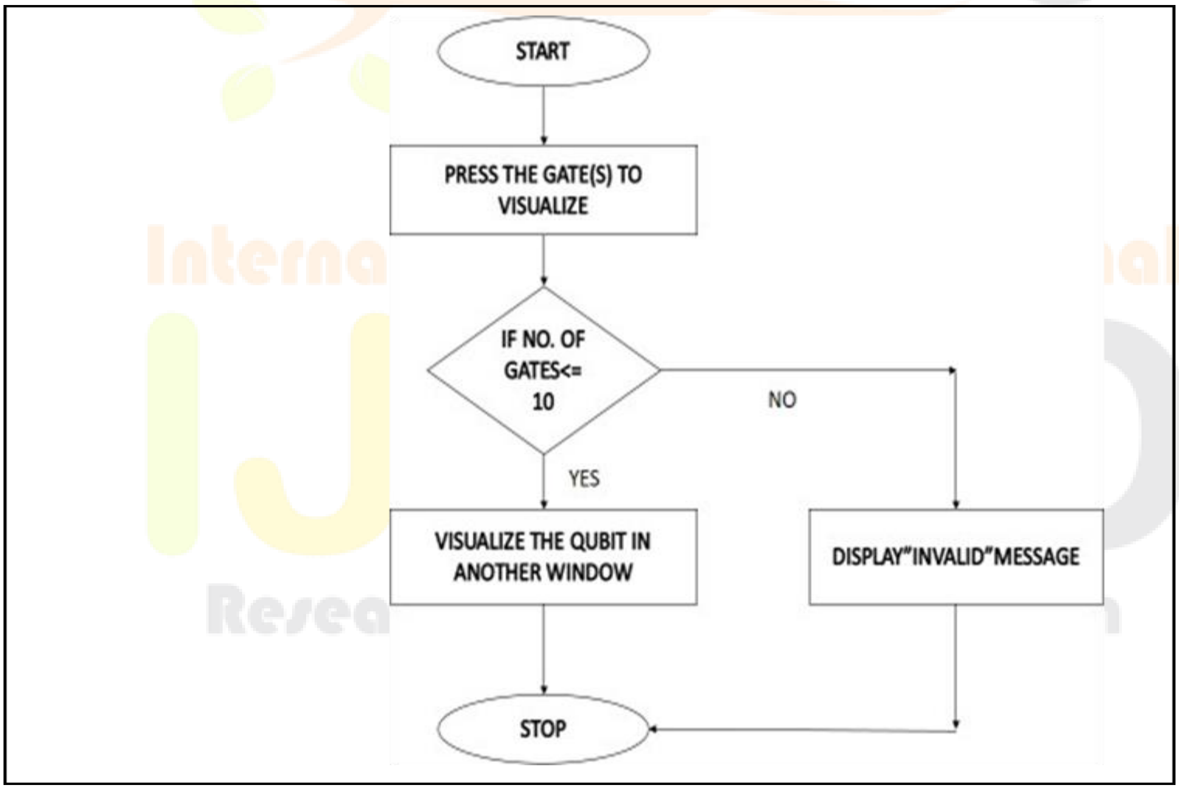
Fig. 1 (a) Bloch sphere



(b) The GUI window

3.1.1 Flow chart for visualization

The work flow of the visualization window can be well described by a flow chartand Fig. 2 shows that flow chart.


START
PRESS THE GATE(S) TO VISUALIZE
IF NO. OF GATES<= 10
NO
YES
VISUALIZE THE QUBIT IN ANOTHER WINDOW
DISPLAY"INVALID"MESSAGE
STOP
Fig. 2 Flowchart for visualization

3.1.2 Gate representation and their functionalities

These are the options available in the GUI window. Table 1 shows its representation and their functionality.

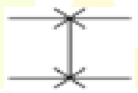
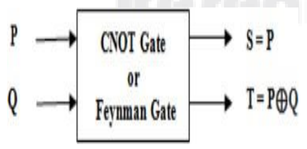
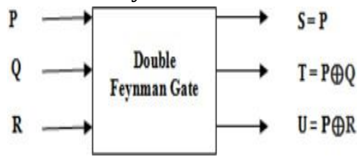
Table 1 Gate representation and their functionalities

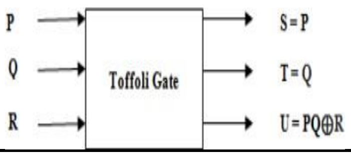
Gate Representation	Functionality
X	Flips the state of qubit
Y	Rotates the state vector about Y-axis
Z	Flips the phase by PI radians
RX	Parameterized rotation about the X axis
RY	Parameterized rotation about the X axis
RZ	Parameterized rotation about the X axis
S	Rotates the state vector about Z axis by PI/2radians
SD	Rotates the state vector about Z axis by -PI/2radians
T	Rotates the state vector about Z axis by PI/4radians
TD	Rotates the state vector about Z axis by -PI/4radians
H	Creates the state of superposition
Quit	Comes out of the program
Visualize	Visualizes the single qubit rotations corresponding to applied gates in a separatetkinter window
Clear	Clears the display. Reinitializes the QuantumCircuit for fresh calculation
About	Displays the info about the project

3.2 Quantum gates: symbols and functionalities

Quantum gates are the basic building blocks of quantum circuits, which are thefundamental building blocks of quantum computing. They are analogous to classicallogic gates, but operate on quantum bits or qubits, which can exist in multiple states at once. Quantum gates are used to manipulate the quantum state of qubits in a controlled way. Table 2 shows the basic quantum gates, their symbol and their functionalities.

Table 2 Quantum gates – Symbols and functionalities

GATE & SYMBOL	FUNCTIONALITY
Swap gate 	It simply swaps the states i.e. ifthe input states are |0>, |1> thenits output will be |1>, |0>
CNOT gate 	If control bit =1,target bit willbe flipped. Also known as Feynman gate
Double FeynmanGate 	If inputs are P, Q and R, andOutputs are S, T, U. Output equations are S=P,T=P⨁Q and U= P⨁R

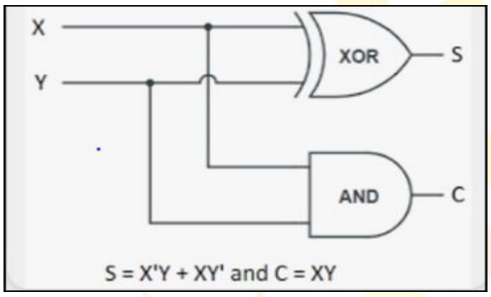
Toffoli Gate 	If inputs are P, Q and R, and theOutputs are S,T,U, then the output equations will be S=P, T=Q and U= PQ⊕R

3.3 Combinational circuits implementation using quantum gates

Combinational circuits using quantum gates have the potential to perform certain computational tasks much faster than classical digital circuits, due to the abilityof qubits to exist in superpositions of states. However, building and operating quantum circuits requires a highly controlled and precise experimental setup, which makes quantum computing still in its infancy, with limited practical.

3.3.1 Quantum half adder

A quantum half adder can be implemented using a combination of quantum gates, such as the CNOT gate, Hadamard gate, and Toffoli gate. Fig 3 (a) and (b) shows the circuit of half-adder implementation using classical gates and quantum gates respectively.



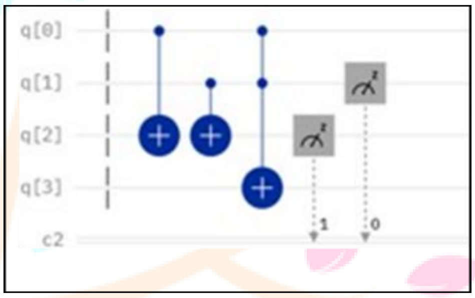
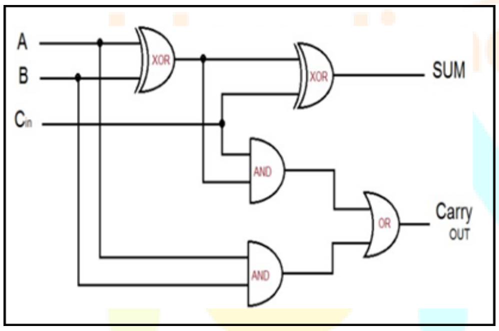


Fig.3(a) Classical half-adder circuit **(b)**Quantum half-adder circuit

3.3.2 Quantum Full Adder

A quantum full adder is a quantum circuit that performs addition of two binary numbers, including a carry-in bit, and produces a sum and a carry-out bit. The circuitis based on the principles of quantum mechanics and uses quantum gates to perform the addition. Fig 3.6 shows the classical circuit of full-adder. Fig. 4 (a) and (b) shows the full-adder circuit implemented using classical gates and quantum gates respectively.



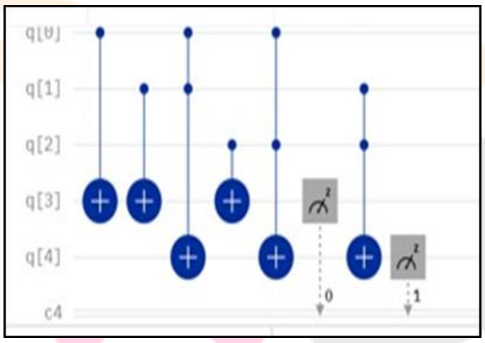


Fig. 4 (a) Classical full-adder circuit **(b)** Quantum full-adder circuit

IV. RESULTS AND DISCUSSIONS

4.1 Visualization of qubits in quantum gates using Bloch sphere

4.1.1 The GUI window: Pauli X gate is applied

When we execute the program in VSC, the GUI window will appear. Then, we have to give the required options/gates that are to be visualized. Here, in Fig 4.2, the GUI window with Pauli X gate applied is shown.

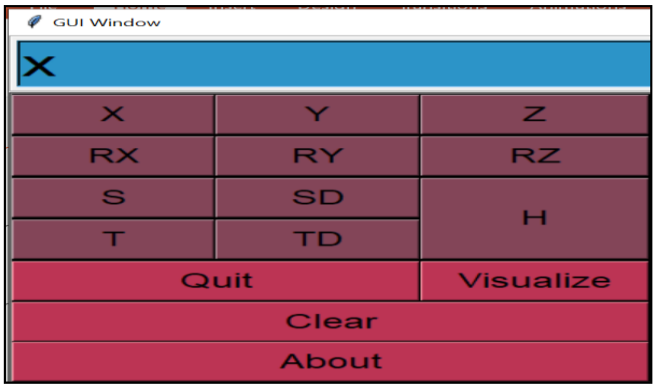


Fig. 5 Pauli X gate applied

Fig. 6 (a) and (b) shows the orientation of qubit in bloch sphere before and after the execution of Pauli X gate.

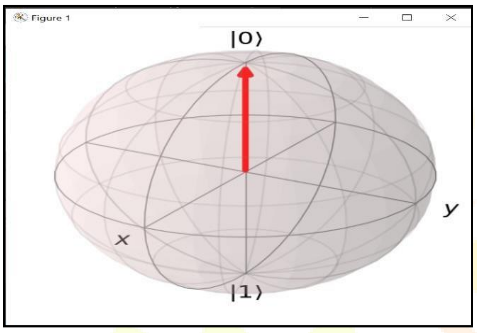
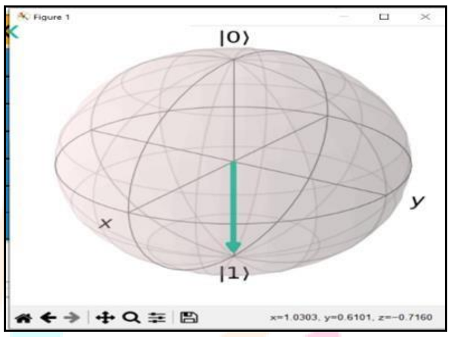


Fig. 6 (a) Before executing X gate



(b) After executing X gate

4.1.2 Visualization Window: RY gate is applied

Unlike Pauli gates, when RY gate is applied, ‘Get Theta’ window will appear.There we have to give the required angle of rotation in the range of -2*Pi to 2*Pi. Fig. 7 (a) and (b) shows the GUI window with RY gate applied for visualization and the ‘Get Theta’ window respectively. Fig. 7(c) and (d) represents the output for RY = π/4 and the output for RY = π/2 respectively.

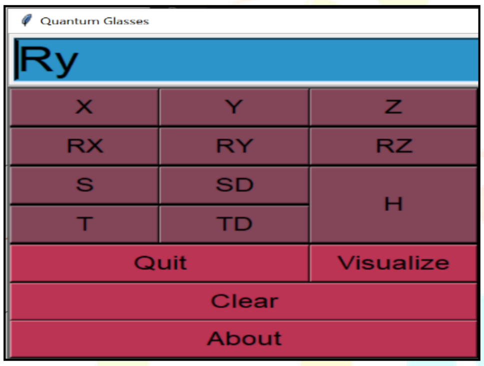
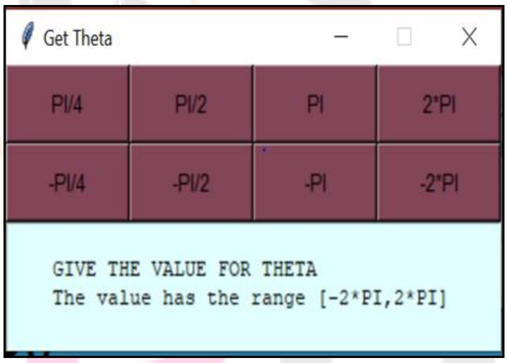
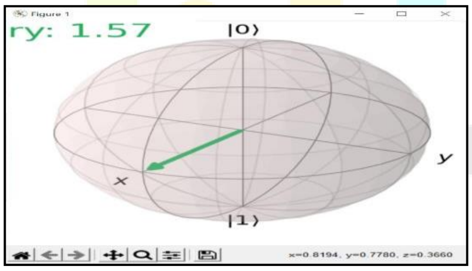


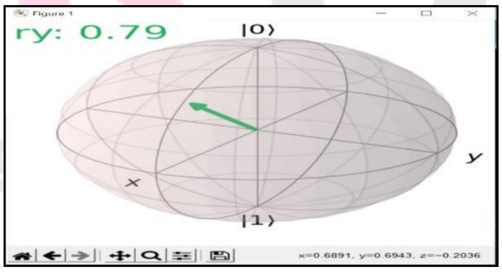
Fig. 7 (a) Applying RY gate



```
GIVE THE VALUE FOR THETA
The value has the range [-2*PI,2*PI]
```

(b) Theta selection window





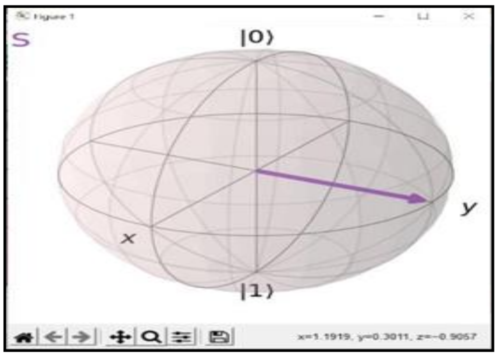
(c) Output for RY = π/4

(d) Output for RY = π/2

4.1.3 Visualization window : S,T,H gate

S,T and H gates are also available in the visualization window. Fig. 8 showsthe outputs of S,H gates respectively.

- S gate - Rotates the state vector about Z axis by PI/2 radians
- T gate - Rotates the state vector about Z axis by PI/4 radians
- H gate – Creates the state of superposition



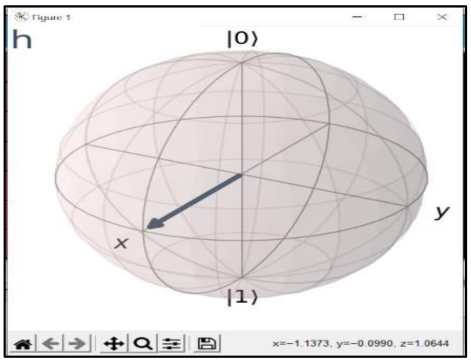
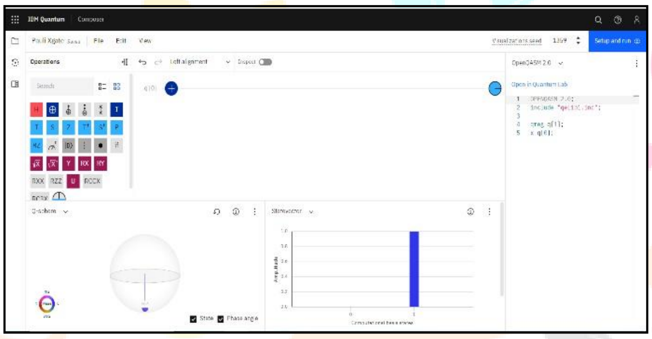


Fig. 8 (a) Output of S gate **(b)** Output of H gate

4.2 Quantum gates implementation in qiskit

4.2.1 Pauli X Gate

The basic gates of quantum were implemented in qiskit. Pauli X gate will do a rotation of 180 degrees in X axis in clockwise direction. Fig. 10 shows the output of Pauli X gate with single X applied.

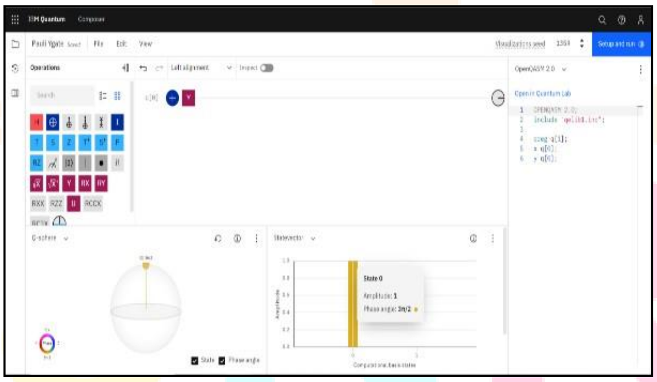


```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[1];
x q[0];
```

Fig. 9 Single Pauli X gate applied

4.2.2 Pauli Y Gate

Pauli Y gate will do a rottion of 180 degrees in Y axis in clockwise direction.Fig. 10 shows the output of Pauli Y gate for inputs '0'.

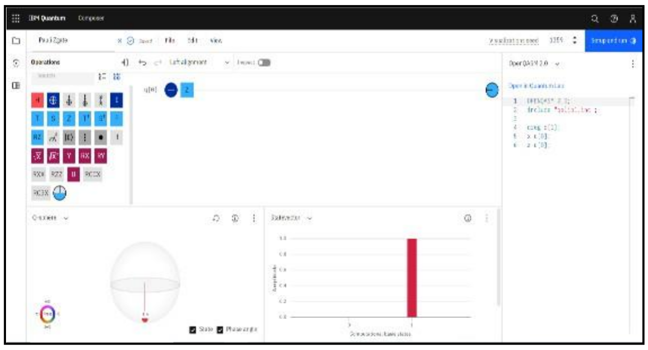


```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[1];
x q[0];
y q[0];
```

Fig. 10 Pauli Y gate- '0' input

4.2.3 Pauli Z Gate

Pauli Z gate will do 180 degree rotation around Z axis in clockwise direction. Fig. 11 shows the output of Pauli Z gate '1' input.

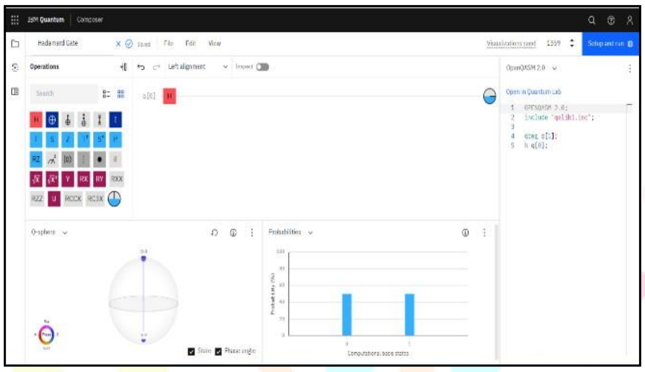


```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[1];
x q[0];
z q[0];
```

Fig. 11 Pauli Z gate – ‘1’ input

4.2.4 Hadamard Gate

Hadamard gate will distribute the probability of getting ‘0’ and ‘1’ equally. Fig. 12 shows the output of Hadamard gate.

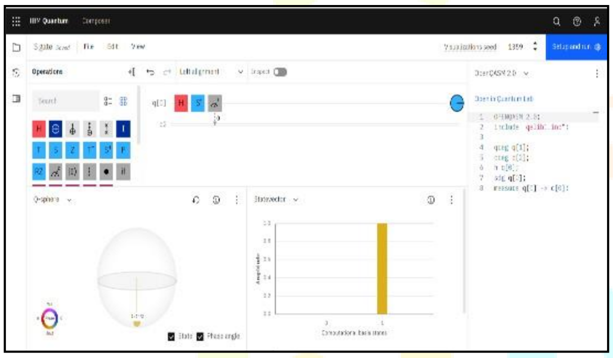


```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[1];
h q[0];
```

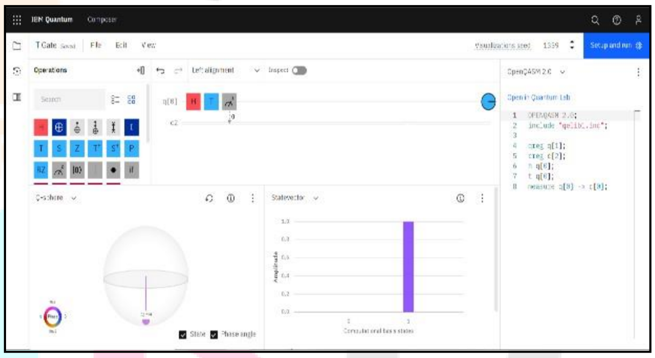
Fig. 12 Output of Hadamard gate

4.2.5 S and T Gate

S gate rotates the state vector about Z axis by PI/2radians. SD gate rotates the state vector about Z axis by -PI/2radians. T and TD gates also do the same functionality as S and SD respectively, but the angle is PI/4 radians. Fig. 13 and Fig. 14 shows the output of S gate and T gate respectively.



```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[1];
creg c[1];
h q[0];
sdg q[0];
measure q[0] -> c[0];
```

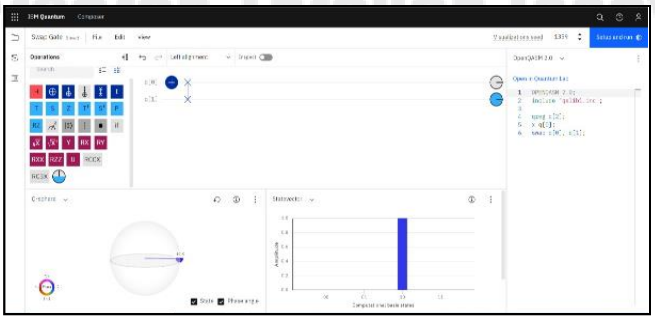


```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[1];
creg c[2];
h q[0];
t q[0];
measure q[0] -> c[0];
```

Fig. 13 Output of S gate **Fig. 14** Output of T gate

4.2.6 Swap Gate

Swap gate will swap the inputs. Fig. 15 shows the output of swap gate.

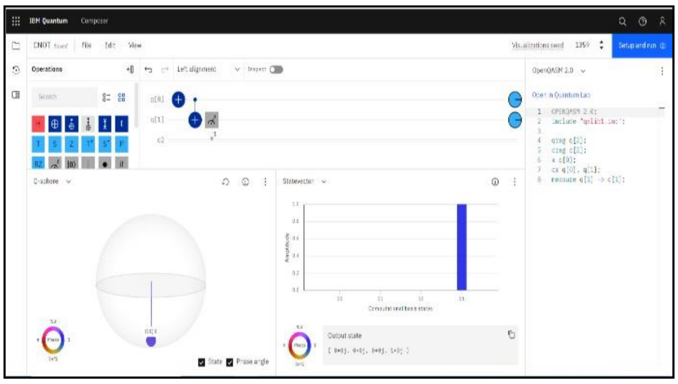


```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[2];
x q[0];
swap q[0], q[1];
```

Fig. 15 Output of Swap gate

4.2.7 CNOT Gate

For CNOT gate, if the control bit is ‘1’ , then the target bit will be flipped. Fig. 16 shows the outputs of CNOT gate with control input ‘1’.

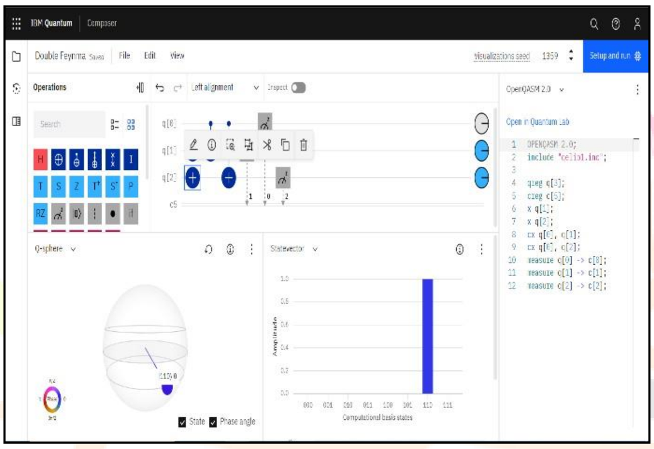


```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[2];
creg c[2];
x q[0];
cx q[0], q[1];
measure q[1] -> c[1];
```

Fig. 16 Output of CNOT gate-‘1’ input

4.2.8 Double Feynman Gate

In Double Feynman gate, if inputs are P, Q and R, and Outputs are S, T, U, thenthe output equations will be S=P, T=P⨁Q and U= P⨁R. Fig 4shows the output of DF gate.

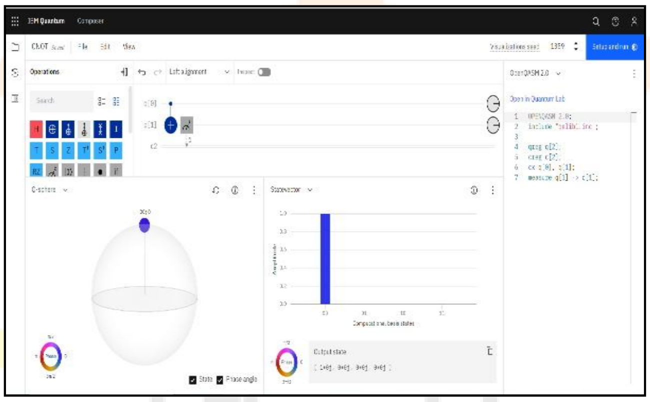


```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[3];
creg c[3];
x q[1];
x q[2];
cx q[0], q[1];
cx q[0], q[2];
measure q[0] -> c[0];
measure q[1] -> c[1];
measure q[2] -> c[2];
```

Fig. 17 Output of double Feynman gate

4.2.9 Toffoli Gate

In Toffoli gate, if inputs are P, Q and R, and the Outputs are S,T,U, then the output equations will be S=P, T=Q and U= PQ⨁R.

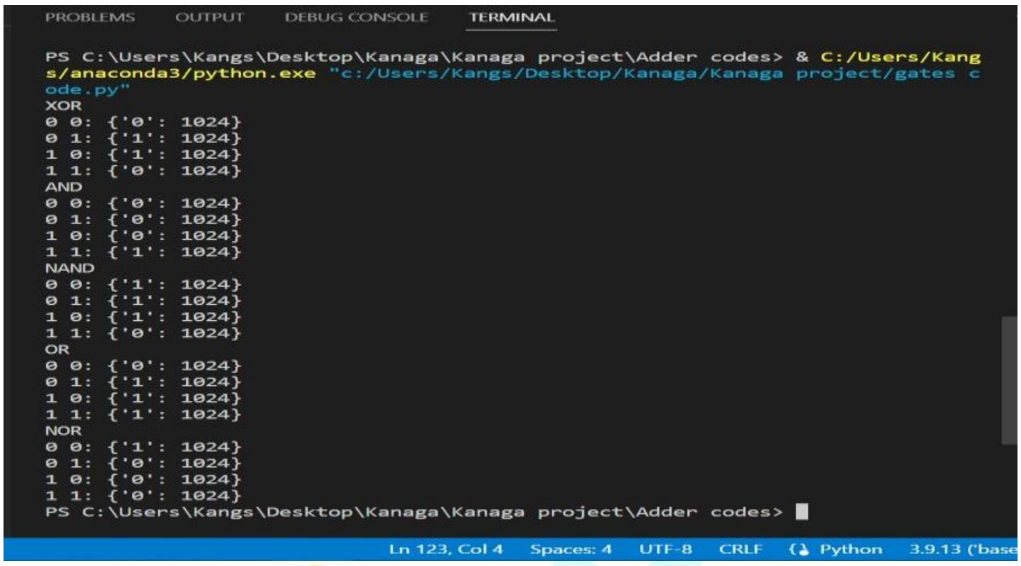


```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[2];
creg c[2];
cx q[0], q[1];
measure q[1] -> c[1];
```

Fig. 18 Output of Toffoli gate – input is ‘10’

4.3 CLASSICAL GATES IMPLEMENTATION USING QUANTUM GATES IN VSC AND QISKIT

The classical gates such as AND gate, NAND gate, OR gate, NOR gate, EXORgate were implemented in VSC using qiskit. Fig. 19 shows the output of classical gates implemented using quantum gates in VSC and qiskit.



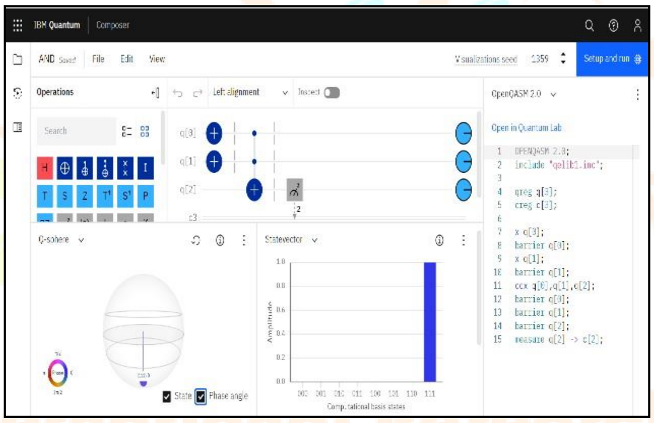
```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

PS C:\Users\Kangs\Desktop\Kanaga\Kanaga project\Adder codes> & C:/Users/Kang
s/anaconda3/python.exe "c:/Users/Kangs/Desktop/Kanaga/Kanaga project/gates c
ode.py"
XOR
0 0: {'0': 1024}
0 1: {'1': 1024}
1 0: {'1': 1024}
1 1: {'0': 1024}
AND
0 0: {'0': 1024}
0 1: {'0': 1024}
1 0: {'0': 1024}
1 1: {'1': 1024}
NAND
0 0: {'1': 1024}
0 1: {'1': 1024}
1 0: {'1': 1024}
1 1: {'0': 1024}
OR
0 0: {'0': 1024}
0 1: {'1': 1024}
1 0: {'1': 1024}
1 1: {'1': 1024}
NOR
0 0: {'1': 1024}
0 1: {'0': 1024}
1 0: {'0': 1024}
1 1: {'0': 1024}
PS C:\Users\Kangs\Desktop\Kanaga\Kanaga project\Adder codes> 

Ln 123, Col 4    Spaces: 4    UTF-8    CRLF    {} Python    3.9.13 ('base
```

Fig. 19 Output of classical gates implemented using quantum gates in VSC and Qiskit

4.4 Classical gates implementation in qiskit

4.4.1 AND Gate

The output of AND gate for the input '11' is shown in Fig 4.20

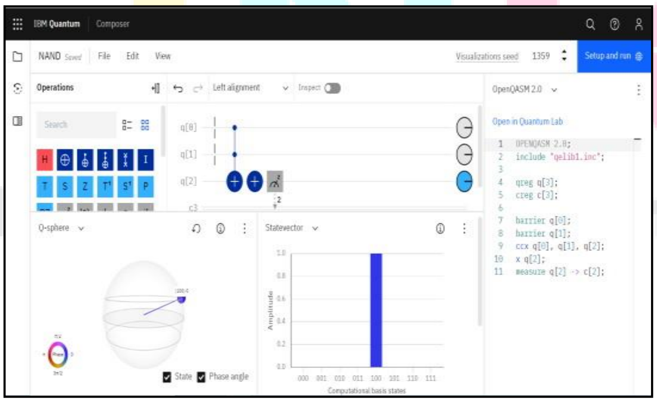


```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[3];
creg c[3];

x q[0];
barrier q[0];
x q[1];
barrier q[1];
ccx q[0],q[1],q[2];
barrier q[0];
barrier q[1];
barrier q[2];
measure q[2] -> c[2];
```

Fig. 20 AND gate output-'11' input

4.4.2 NAND Gate

The output of NAND gate for the input '00' is shown in Fig. 21.

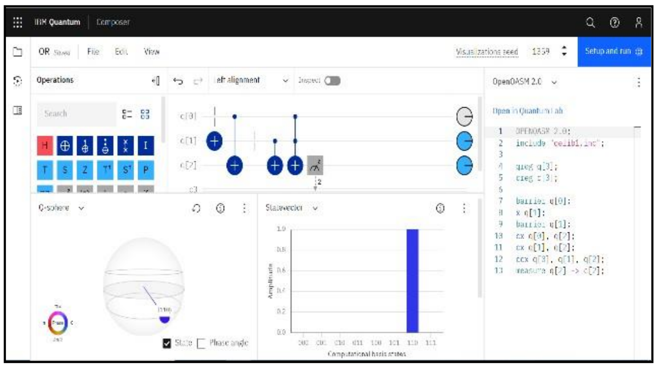


```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[3];
creg c[3];

barrier q[0];
barrier q[1];
ccx q[0], q[1], q[2];
x q[2];
measure q[2] -> c[2];
```

Fig. 21 NAND gate output-'00' input

4.4.3 OR Gate

The outputs of OR gate for input '01' is shown in Fig. 22

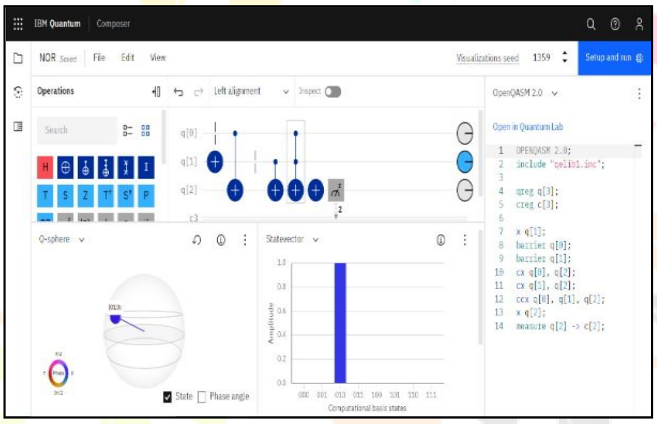


```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[3];
creg c[3];

barrier q[0];
x q[1];
barrier q[1];
cx q[0], q[2];
cx q[1], q[2];
ccx q[0], q[1], q[2];
measure q[2] -> c[2];
```

Fig. 22 OR gate output-'01' input

4.4.4 NOR Gate

The output of NOR gate for the input '01' is shown in Fig. 23

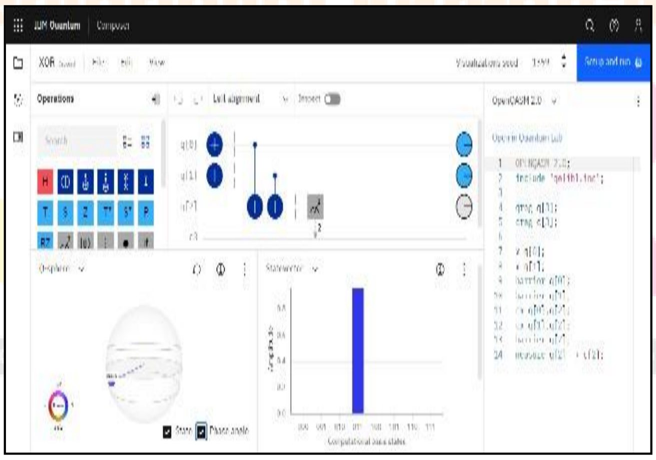


```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[3];
creg c[3];

x q[1];
barrier q[0];
barrier q[1];
cx q[0], q[2];
cx q[1], q[2];
ccx q[0], q[1], q[2];
x q[2];
measure q[2] -> c[2];
```

Fig. 23 NOR gate output-'01' input

4.4.5 EXOR Gate

The output of EXOR gate for input '11' is shown in Fig. 24.

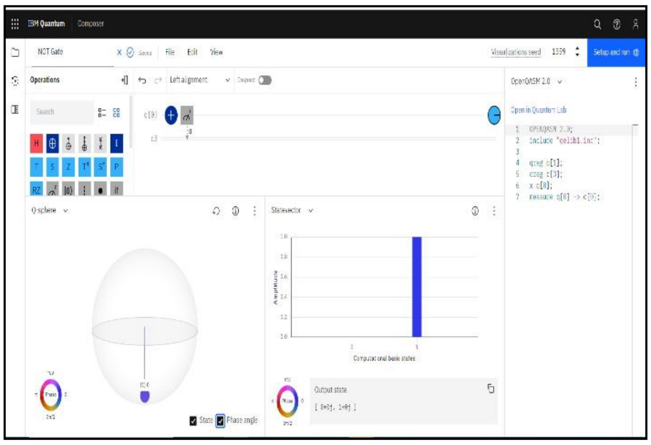


```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[3];
creg c[3];

x q[0];
x q[1];
barrier q[0];
barrier q[1];
cx q[0],q[2];
cx q[1],q[2];
barrier q[2];
measure q[2] -> c[2];
```

Fig. 24 EXOR gate output- '11' input

4.4.6 NOT Gate

The outputs of NOT gate for input ‘0’ is shown in Fig. 25.



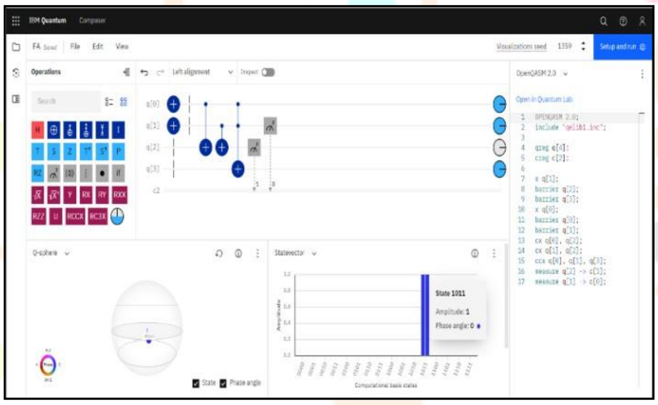
```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[1];
creg c[3];
x q[0];
measure q[0] -> c[0];
```

Fig. 25 NOT gate output- ‘0’ input

4.5 IMPLEMENTATION OF COMBINATIONAL CIRCUITS USING QUANTUM GATES

4.5.1 Implementation of Half Adder using Quantum Gates

The outputs of Half-adder for inputs ‘00’ , ‘01’, ‘10’, ‘11’ are shown in Fig. 26

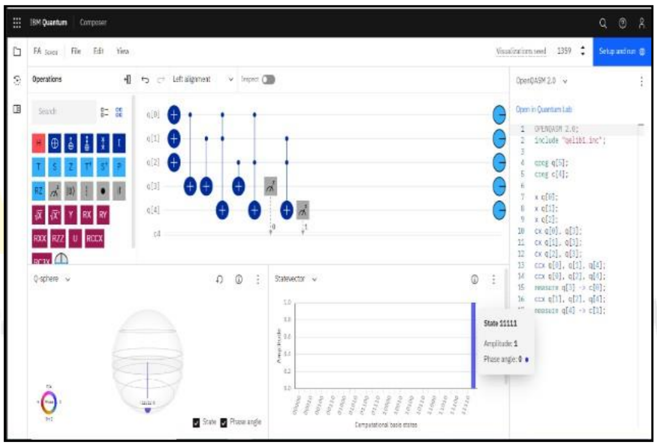


```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[4];
creg c[2];

x q[1];
barrier q[2];
barrier q[3];
x q[0];
barrier q[0];
barrier q[1];
cx q[0], q[2];
cx q[1], q[2];
ccx q[0], q[1], q[3];
measure q[2] -> c[1];
measure q[1] -> c[0];
```

Fig. 26 Output of Half adder- ‘11’ input

4.5.2 Implementation of Full Adder using Quantum Gates

The outputs of full-adder for input ‘111’ is shown in Fig. 27.



```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[5];
creg c[4];

x q[0];
x q[1];
x q[2];
cx q[0], q[3];
cx q[1], q[3];
cx q[2], q[3];
ccx q[0], q[1], q[4];
ccx q[0], q[2], q[4];
measure q[3] -> c[0];
ccx q[1], q[2], q[4];
measure q[4] -> c[1];
```

Fig. 27 Output of full adder- ‘111’ input

V. Conclusion

Thus, a visualization tool that will display the orientation of the qubit in quantumgates in a bloch sphere was developed and it can display upto a maximum of 10 gatesat a time. Then, Quantum gates such as Pauli X gate, Pauli Y gate, Pauli Z gate, H gate, S gate, SD gate, T gate, TD gate, swap gate, CNOT gate, double feynman gate,toffoli gate were implemented in Qiskit. And then, the classical gates such as AND gate, NAND gate, OR gate, NOR gate, EXOR gate were implemented using Quantumgates in qiskit. Finally, the combinational circuits of Half adder and Full adder were constructed using those Quantum gates. Since we know the exact inputs and outputs of the gates, Pauli X gates were used instead of Hadamard gate for controlling the inputs. This removes the chance of distribution of output probabilities and we get theexact '1' probability for the outputs.

References

[1] M. Swathi and B. Rudra, "Implementation of Reversible Logic Gates with Quantum Gates," *2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC), NV, USA, 2021, pp. 1557-1563,doi: 10.1109/CCWC51732.2021.9376060.*

[2] E. h. Shaik and N. Rangaswamy, "Implementation of Quantum Gates based Logic Circuits using IBM Qiskit," *2020 5th International Conference on Computing, Communication and Security (ICCCS), Patna, India, 2020, pp. 1-6, doi: 10.1109/ICCCS49678.2020.9277010.*

[3] G. Nikhil, B. Sharanya, P. B. Reddy and R. P. Vidyadhar, "Reversible 2:4 DecoderUsing Universal Fredkin Gate," *2021 6th International Conference on Communication and Electronics Systems (ICCES), Coimbatre, India, 2021,pp. 173- 176, doi: 10.1109/ICCES51350.2021.9489.*

[4] T.S. Humble, H. Thapliyal, M-C. Edgard, F.A. Mohiyaddin, and R.S. Bennink, "Quantum computing circuits and devices*," IEEE Design and Test, vol. 36, pp. 69-94, April 2019*

[5] Bennett, Charles H. "Logical reversibility of computation." *IBM journal of Research and Development 17.6 (1973): 525-532.*

[6] Kumar, Umesh, Lavisha Sahu, and Uma Sharma. "Performance evaluation of reversible logic gates." *International Conference on ICT in Business Industry & Government (ICTBIG), IEEE (2016): 1-4.*