METHOD OF IMPLEMENTATION

QISKIT

Qiskit is an open-source framework for quantum computing. It provides tools for creating and manipulating quantum programs and running them on prototype quantum devices on IBM Quantum Experience or on simulators on a local computer [15].

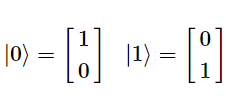
Qiskit was founded by IBM to allow software development for their cloud quantum computing service.

To perform computation in qiskit one can use Jupyter lab or Jupyter notebook.

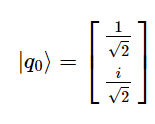
Qubit or also called as Quantum bit is a basic unit of quantum information. Qubit can store a single binary piece of information. A single qubit can store bit 1 or 0, but they can be manipulated in many other ways. From qubit we can capitalize on two phenomena, superposition and entanglement. Superposition is the ability of quantum systems to be in multiple state at the same time. Entanglement is an extremely strong correlation that exists between two quantum particles, in other words two or more quantum particles are linked in perfect unison even when separated by great distance [15].

The state 0 of qubit is represented as |0>

The state 1 of qubit is represented as |1>



The linear combination of the two states is also possible and is represented as |q0>



IMPLEMENTATION OF FULL ADDER AND FULL SUBTRACTOR IN QISKIT

HNG gate

HNG gate is a four input reversible gate. The inputs are A, B, C and D and the outputs are P, Q, R, S [16].

P output is A. Q output is B. R output is A ⊕ B ⊕C. S output is (A

XOR B) AND C XOR (A AND B) XOR D

P = A,

Q = B,

R = A⊕B⊕C,

S = (A⊕B)C ⊕AB⊕D

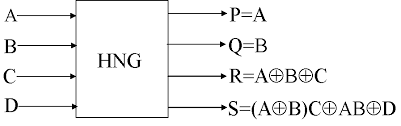


fig. Block diagram of HNG gate

The HNG gate can be computed in the qiskit and below are some terms that are used to perform computations [15].

1. QuantumCircuit – Algorithms and operations are stored in this class

2. Aer – This is a class that handles classical simulator backend

3. Execute – This is a function that we must import in order to run quantum algorithms

from qiskit import QuantumCircuit, Aer, execute

def hng(inp1, inp2, inp3, inp4):

qc = QuantumCircuit(4, 4)

if(inp1 == '1'):

qc.x(0)

if(inp2 == '1'):

qc.x(1)

if(inp3 == '1'):

qc.x(2)

if(inp4 == '1'):

qc.x(3)

qc.ccx(0, 2, 3)

qc.cx(0, 2)

qc.ccx(1, 2, 3)

qc.cx(1, 2)

qc.measure(0, 3)

qc.measure(1, 2)

qc.measure(2, 1)

qc.measure(3, 0)

backend= Aer.get\_backend('qasm\_simulator')

job = execute(qc, backend, memory = True )

output = job.result().get\_memory()[0]

return qc, output

for inp1 in ['0', '1']:

for inp2 in ['0', '1']:

for inp3 in ['0', '1']:

for inp4 in ['0', '1']:

qc, output = hng(inp1, inp2, inp3, inp4)

print('{} {} {} {}'.format(inp1, inp2, inp3, inp4), '=', output)

display(qc.draw('mpl'))

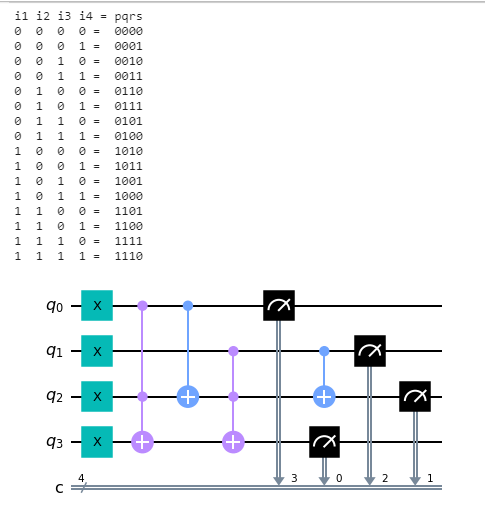


fig HNG gate

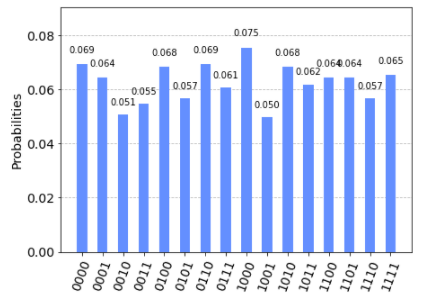


fig. histogram plot of hng gate

The above plot shows the probablilty of the getting the output for the specific input. The output is calculated by passing the circuit to the ibm cloud.

HNG gate as a full adder

One of the prominent functionalities of the HNG gate is that it can work singly as a reversible full adder [16]. Reversible full adder circuit uses only one reversible HNG gate. It produces only two garbage outputs. It requires only one constant input. The inputs are A, B, C and constant input 0. The outputs are two garbage output at P and Q, output at R is considered as sum and output at S is the carry.

#hng gate as a full adder

def hng\_adder(inp1, inp2, inp3, inp4):

qc = QuantumCircuit(4, 2)

if(inp1 == '1'):

qc.x(0)

if(inp2 == '1'):

qc.x(1)

if(inp3 == '1'):

qc.x(2)

qc.ccx(0, 2, 3)

qc.cx(0, 2)

qc.ccx(1, 2, 3)

qc.cx(1, 2)

qc.measure(2, 1)

qc.measure(3, 0)

backend=Aer.get\_backend('qasm\_simulator')

job=execute(qc,backend, memory = True )

output = job.result().get\_memory()[0]

return qc, output

print('{} {} {} '.format('i1', 'i2', 'i3'), '=', 's carry')

for inp1 in ['0', '1']:

for inp2 in ['0', '1']:

for inp3 in ['0', '1']:

qc,output=hng\_adder(inp1,inp2,inp3,inp4)

print('{} {} {}'.format(inp1, inp2, inp3), '=', output)

display(qc.draw('mpl'))

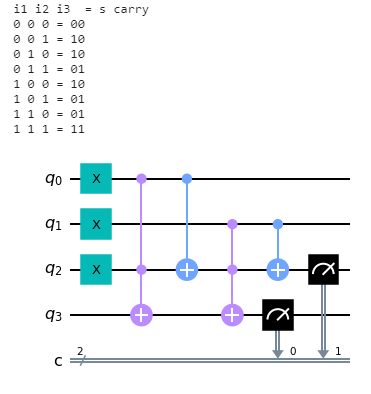


fig hng gate as full adder

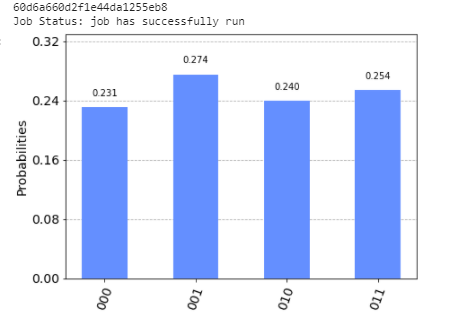


fig. histogram plot of full adder

The above plot shows the probablilty of the getting the output for the specific input for a full adder circuit. The output is calculated by passing the circuit to the ibm cloud.

HNG gate as full subtractor

HNG gate can work as a reversible full subtractor [10]. Reversible full subtractor circuit uses one reversible HNG gate and two not gates. It produces only two garbage outputs. It requires only one constant input. The inputs are A, B, C and constant input 0. The outputs are two garbage output at P and Q, output at R is considered as difference and output at S is the borrow.

#hng gate as a full subtractor

def hng\_subtractor (inp1, inp2, inp3, inp4):

qc = QuantumCircuit(4, 2)

if(inp1 == '1'):

qc.x(0)

if(inp2 == '1'):

qc.x(1)

if(inp3 == '1'):

qc.x(2)

qc.x(0)

qc.ccx(0, 2, 3)

qc.cx(0, 2)

qc.ccx(1, 2, 3)

qc.cx(1, 2)

qc.x(2)

qc.measure(2, 1)

qc.measure(3, 0)

backend=Aer.get\_backend('qasm\_simulator')

job=execute(qc,backend, memory = True )

output = job.result().get\_memory()[0]

return qc, output

for inp1 in ['0', '1']:

for inp2 in ['0', '1']:

for inp3 in ['0', '1']:

qc,output=hng\_adder(inp1,inp2,inp3,inp4)

print('{} {} {}'.format(inp1, inp2, inp3), '=', output)

display(qc.draw(‘mpl’))

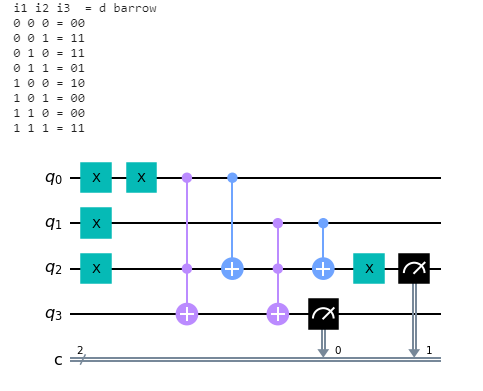


fig hng gate as full subtractor

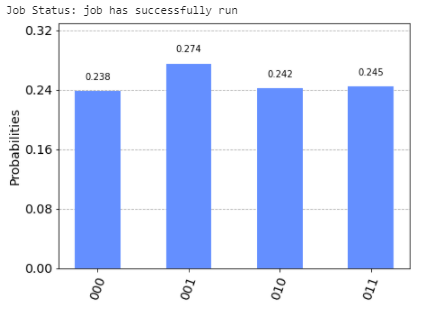


fig. histogram plot of full adder

The above plot shows the probablilty of the getting the output for the specific input for a full subtractor circuit. The output is calculated by passing the full subtractor circuit to the ibm cloud.

CONCLUSION

Classical computing is a convenient tool for performing sequential operations and storing information, but classical computations are based on the linear mathematics. Therefore it is difficult to perform non linear operations. Quantum computing seems to be a suitable candidate in solving nonlinear problems as it has nonlinear properties of nature.

Qiskit is a tool to perform quantum computations. The output of the quantum computation are based on the state vectors and probabilities. The output may vary due to the noise migrations when the data is passed to the ibm cloud.

Qiskit is an emerging tool for quantum computations. Ibm still is running its research in the quantum computers and computations, but the qubits availability is still less in number therefore it is hard to perform many algorithms and quantum computations which has large number of qubits in it.

REFERENCES

1. 1. R. Landauer, “Irreversibility and Heat Generation in the Computational Process”, IBM Journal of Research and Development, 5, pp. 183-191, 1961.

2. C. H. Benett, “Logical Reversibility of Computation”, IBM Journal of Research and Development, pp.525-532, November 1973.

3. Classical and Quantum Logic Gates: “An Introduction to Quantum Computing”, Ashok Muthukrishnan, Rochester Center for Quantum Information (RCQI).

4. Michael Kirkedal Thomsen and Holger Bock Axelsen, “Reversible arithmetic logic unit for quantum arithmetic”, Journal of physics A Mathematical and Theoretical.

5. Alwin Zulehner, Alexander Paler and Robert Wile, “An efficient methodology for mapping quantum circuits to the IBM QX Architectures”.

6. Ranveer Kumar Singh, Bishvanwesha Panda, Bikash K Behra and Prasanta K Panigrahi, “Demonstration of a general fault tolerant quantum error detection code for (2n+1)qubit entangled state on IBM 16-qubit quantum computer.

7. Thomas Alexander, Naoki Kanazawa, Daniel J Egger, “Qiskit Pulse: Programming Quantum Computers through the cloud with Pulses”, IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, USA

8. Stephan Barabasi, James Barrera, Charles C Tappert “Student user experience with the IBM Qiskit Quantum computing interface”, Seidenberg school of CSIS, Pace University, Pleasantville, New York

9. M. Nielsen and I. Chuang, Quantum Computation and Quantum Information. Cambridge Univ. Press, 2000.

10. Full Adder and full Subtractor circuit using reversible logic gates 2016 Pradeesha R. Chandran, Anand kumar, Arthi Noor

11. Design of efficient full adder in quantum dot cellular automata 2013

12. An Improved structure of reversible adder and subtractor

13. A novel design of half and full adder using basic QCA gates – 2014

14. Efficient approaches for designing quantum circuit of various reversible gates – 2017 Quantum representation with calculating.

15. Qiskit textbook learing resources by qiskit.org

16. A reversible design of BCD Multiplier V. Rajmohan, Dr. V. Ranganathan, M. Rajmohan

17. Quantum circuits and algorithms 2019

18.Quantum information processing: A Review on Quantum arithmetic circuits – 2020

19. A Brief overview of reversible logic gates and reversible circuits 2019

20. Implementing logical circuits with four phase quantum dot cellular clock using QCA designer – 2020

21. Novel reversible DS gate for reversible logic synthesis 2016

22. Cost efficient design of reversible adder circuits for low power applications.