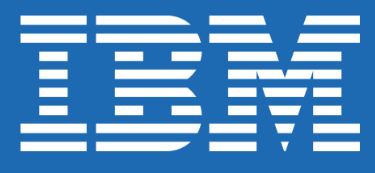
**COIMBATORE INSTITUTE OF TECHNOLOGY**

**COIMBATORE - 014**

**CIT QUANTUM HACKATHON 2022**

**  **

***IDEA – BOOLEAN POSTULATES***

**TEAM MEMBERS:-**

1. Boobalaragavan. P – 5th year Integrated MSc Software Systems
2. Vijay Narendran. V – 3rd year Integrated MSc Software Systems
3. Naren Karthick. T.P – 3rd year Integrated MSc Software Systems
4. Adithyan. A – 3rd year Integrated MSc Software Systems
5. Lakshman Karthick.T – 3rd year Integrated MSc Software Systems

**ABSTRACT:-**

Digital Circuits from the branch of electronics in which it deals with digital signals to perform the various task to meet various requirement. The input signal applied to these circuits is of digital form, which is represented in 0’s and 1’s binary language format.  Digital circuits are made up of logic gates. Logic gates are a fundamental digital component that performs simple logical operations. Boolean algebra is a useful and clever way to simplify digital circuits. Boolean algebra is a branch of Algebra introduced by George Boole. It is basically the use of propositions to solving logical problems and building circuits. Logical Implications can be solved using the Laws of Boolean Algebra. Quantum computers harness the unique behaviour of quantum physics—such as superposition, entanglement and quantum interference—and apply it to computing. Quantum bits (or qubits) harness the quantum-mechanical property of superposition to occupy the states of both 1 and 0 at the same time. Quantum computers to process problems that classical computers can’t even comprehend at a fraction of the time, even with fewer bits. Our idea is to build set of reversible quantum circuits for the Boolean postulates with corresponding truth tables and different visualizations using Quantum theories.

**LIST OF BOOLEAN POSTULATES:-**

DISJUNCTION [OR GATE]:-

    1) A+B = C

CONJUNCTION [AND GATE]:-

    1) A.B = C

IDENTITY LAW:-

    1) A+0 = A 2) A.1 = A

ANNULMENT LAW:-

    1) A+1 = A 2) A.0 = 0

IDEMPOTENT LAW:-

    1) A+A = A 2) A.A = A

INVERSE LAW [NEGOTIATION]:-

1. Not (A) = A'

DOUBLE INVERSE LAW [DOUBLE NEGOTIATION]:-

    1) A'' = A

COMPLEMENT LAW:-

    1) A+A' = 1 2) A.A' = 0

COMMUTATIVE LAW:-

    1) A.B = B.A  2) A+B = B+A

ASSOCIATIVE LAW:-

    1) (A.B).C = A. (B.C)

DISTRIBUTIVE LAW:-

    1) A. (B+C) = (A.B) + (A.C)

ABSORPTION LAW:-

    1) B+ (B.A) = B

    2) B. (B+A) = B

DE-MORGANs THEOREM:-

    1) (A.B)' = A'+B'

    2) (A+B)' = A'.B'

TRANSPOSITION THEOREM:-

    1) AB + A'C = (A + C) (A' + B)

REDUNDANCY THEOREM:-

    1) AB + BC' + AC = AC + BC'

DUALITY THEOREM:-

    1) A (B+C) = A+ (B.C) = (A+B) (A+C)

COMPLEMENTARY THEOREM:-

    1) A (B+C) = A'+ (B'.C') = (A'+B') (A'+C')

KARNAUGH-MAP [K-MAP]:-

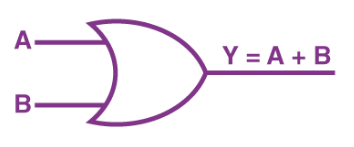
    1) SUM OF PRODUCTS [SOP] => F = X’Y’Z + X’YZ’ + X’YZ + XY’Z

    2) PRODUCTS OF SUM [POS] => F = (X+Y+Z) (X’+Y+Z)(X’+Y’+Z)(X’+Y’+Z’)

**LOGIC GATES:-**

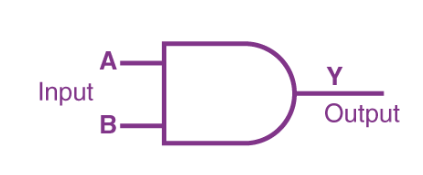
Logic gates are the electronic circuits in a digital system mainly based on the Boolean function. Logic gates are used to carry out logical operations on single or multiple binary inputs and give one binary output.

* OR GATE :
* **I**n an OR gate, the output of an OR gate attains state 1 if one or more inputs attain state 1
* The Boolean expression of the OR gate is Y = A + B, read as Y equals A ‘OR’ B



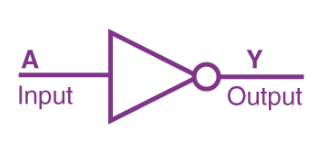
|  |  |  |
| --- | --- | --- |
| **A** | **B** | **Y** |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

* AND GATE :
  + In the AND gate, the output of an AND gate attains state 1 if and only if all the inputs are in state 1
  + The Boolean expression of AND gate is Y = A.B



|  |  |  |
| --- | --- | --- |
| **A** | **B** | **Y** |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

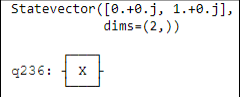
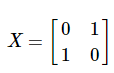
* NOT GATE :
  + In a NOT gate, the output of a NOT gate attains state 1 if and only if the input does not attain state 1.
  + The Boolean expression of NOT gate is Y = A’



|  |  |
| --- | --- |
| **A** | **Y** |
| 0 | 1 |
| 1 | 0 |

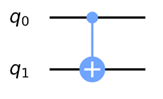
**QUANTUM GATES WE USED:-**

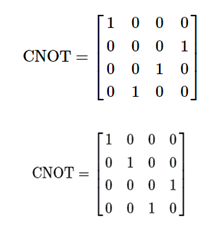
* **SINGLE QUBIT GATE : X**
  + X gate switches the amplitudes of the state |0> and |1>



|  |  |
| --- | --- |
| **Input q0** | **Output** |
| 0 | 1 |
| 1 | 0 |

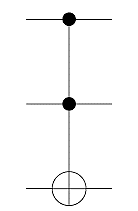
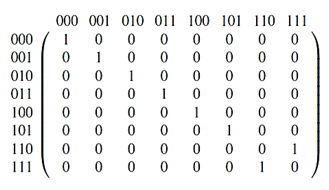
* **MULTI QUBIT GATE : CX**
  + Also kown as CNOT gate or Feynman gate which operates on 2 qubits and where q0 is the control qubit and q1 is the target qubit and it entangles the qubit.

****

****

|  |  |
| --- | --- |
| **Input (t, c)** | **Output (t, c)** |
| 00 | 00 |
| 01 | 11 |
| 10 | 10 |
| 11 | 01 |

* **MULTI QUBIT GATE : CCX**
  + It is a three qubit control gate, takes three qubits as inputs.
  + If q0 is 0 then q1 and q2 remains same, if q0 is 1 then x gate is applied to q1 and q2.

|  |  |
| --- | --- |
| **Input (t, c)** | **Output (t, c)** |
| 000 | 000 |
| 001 | 001 |
| 010 | 010 |
| 011 | 011 |
| 100 | 100 |
| 101 | 110 |
| 110 | 101 |
| 111 | 111 |

**SIMULATOR WE USED:-**

SIMULATOR – AER

The backend simulator emulates the execution of a quantum circuits. [**Qiskit Aer 0.1**](https://qiskit.org/aer). Qiskit Aer is a high-performance simulator framework for studying quantum computing algorithms.

**TOOLS AND TECHNOLOGIES:-**

**Tools** = Google Collaboratory and IBM Quantum Experience

* **Google Collaboratory:** A data analysis and machine learning tool that allows you to combine executable Python code. Using colab, we executed the programs by setting up the corresponding backend objects (Simulators).
* **IBM Quantum Experience:**  IBM offers cloud access to the most advanced quantum computers available using IBM Quantum Experience. Using IBM quantum experience, we executed the programs by setting up the corresponding real quantum devices.

**Technology** = Python

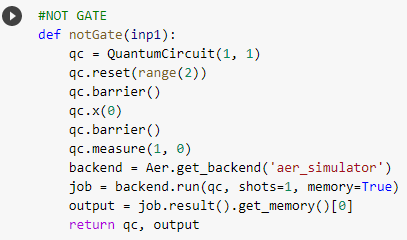
**Libraries** = Qiskit, matplotlib and pylatexenc

* **Qiskit Library: Qiskit** is an open-source SDK for working with quantum computers at the level of pulses, circuits, and application modules.
* **Matplotlib: Matplotlib** is a comprehensive **library**for creating static, animated, and interactive visualizations in Python**. Matplotlib** makes easy things easy and hard things possible.
* **Pylatexenc:** The **pylatexenc**.latexencode module provides a function unicode\_to\_latex () which converts a unicode string into LaTeX text and escape sequences.

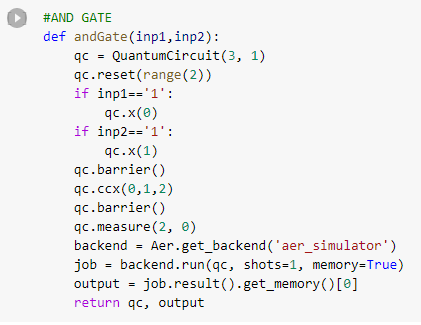
**REPORTS GENERATED:-**

* Truth Table
* Quantum Circuit
* State vector
* Visualizations
  + Qsphere representation
  + Bloch sphere representation
  + Bloch vector representation
  + State city representation
  + State paulivec representation
  + State hinton representation

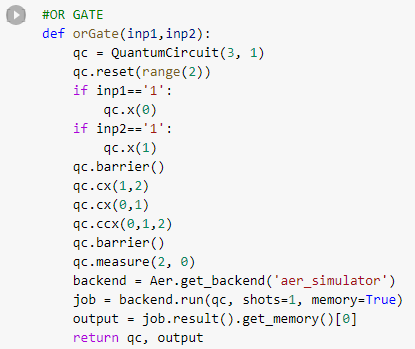
**SAMPLE CODE:-**

****

Classical NOT gate implementation using Quantum single qubit X gate

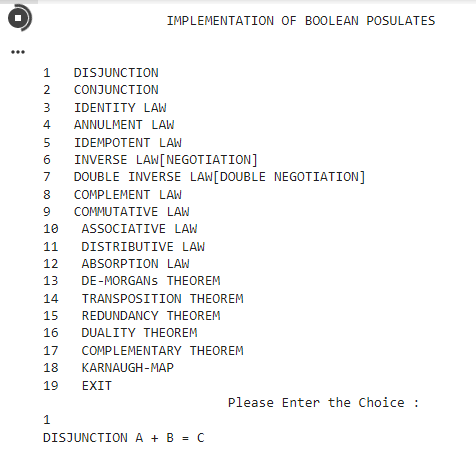
****

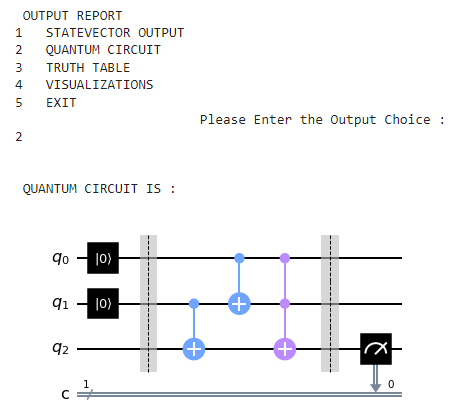
Classical AND gate implementation using Quantum multi qubit CCX gate

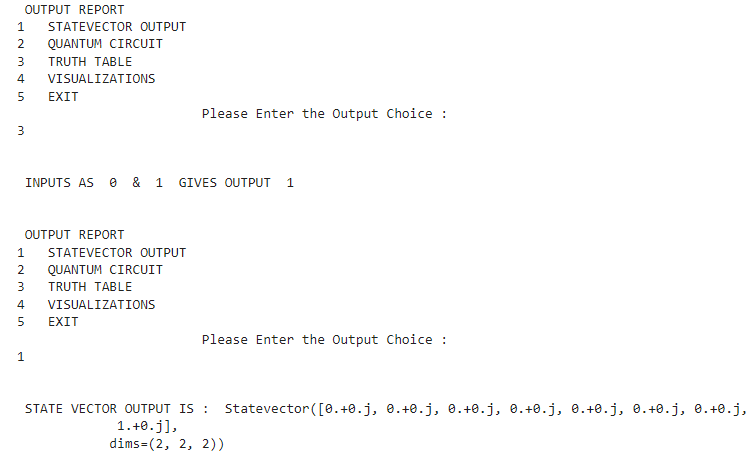
****

Classical OR gate implementation using Quantum multi qubit CX and CCX gate

**SAMPLE REPORT:-**

****

****

****

**CONCLUSION:-**

Our idea works for all two bit Boolean postulates but not for more than two qubits. Our code generates the reports such as Quantum Circuits, Statevector output, Truth table and also other visualizations. Due to lack of qubits availability, we have implemented the reversible circuits for the postulates which possess two qubits or less than that. In future, we like to enhance this project for many numbers of qubits by implementing the reversible circuits for other qubits also.