QCSimulator

Simulating a Quantum Computer in C++

MID TERM REPORT

Meghana Ayyala Somayajula

Gitlab

GitHub

INTRODUCTION

This project aims to simulate a quantum computer using the C++ Programming Language. The target outcome is to implement a simple Quantum Program through implementation of Quantum Gates from scratch.

Midterm Deliverables

Detailed code for implementing methods for Quantum Operators-Hadamard,NOTZ-gate,Rotations on the unit circle with the specified angle and CNOT operator.

Final Deliverables

Code for methods for simulating the Circuit:read_unitary(), read_state,observing_probabilities(),execute(the_number_of_shots)and Testing object. Presentation of the final Project

Code is available on the GitHub repository
GitHub Repo

Quantum Program Class

This class contains all the required methods and variables to implement Quantum Gates and efficiently design a circuit.
A maximum of 3 qubits can be simulated. The following are the list of methods and their syntax.

QuantumProgram(int n);

- Creates an object of QuantumProgram class with number of qubits being initialised by the constructor

create_state();

- Creates a default quantum state according to the number of qubits.

read_state();

- Return the current quantum state of circuit after application of all the defined quantum operators until this point.

read_unitary();

- Prints the unitary matrix (quantum operator) equivalent to all defined quantum operators until this point, i.e., the multiplication of all quantum operators in the defined order.

draw_circuit();

- Draws the quantum circuit at any point after application of the defined quantum gates in a schematic way.
  Representation:
  - Hadamard Gate ‘H’
  - CNOT Gate ‘C’ (Control Qubit), ‘T’ (Target Qubit)
  - NOT Gate ‘X’
  - Z Gate ‘Z’

not_gate(int qubitposition);

- Implements the NOT Quantum Gate on the given qubit position argument

hadamard_gate(int qubitposition)

- Implements the Hadamard Quantum Gate on the given qubit position argument

z_gate(int qubitposition);

- Implements the Z Quantum Gate on the given qubit position argument

cnot_gate(int controlqubit,int targetqubit);

- Implements the multiqubit CNOT Quantum Gate on the given qubits accordingly taken as control qubit and target qubit

Kronecker_Product(double *C, double *A, int nrows, int ncols, double *B, int mrows, int mcols);

- Performs the tensor product of Matrix A and Matrix B in that order and stores the resultant in Matrix C

Matrix_Product(double *C, double *A, int nrows, int ncols, double *B, int mcols);

- Performs matrix multiplication of Matrix A and Matrix B in that order and stores the resultant in Matrix C

Multiply_Matrix_by_Vector(double U[], double *A, int nrows, int ncols,double V[]);

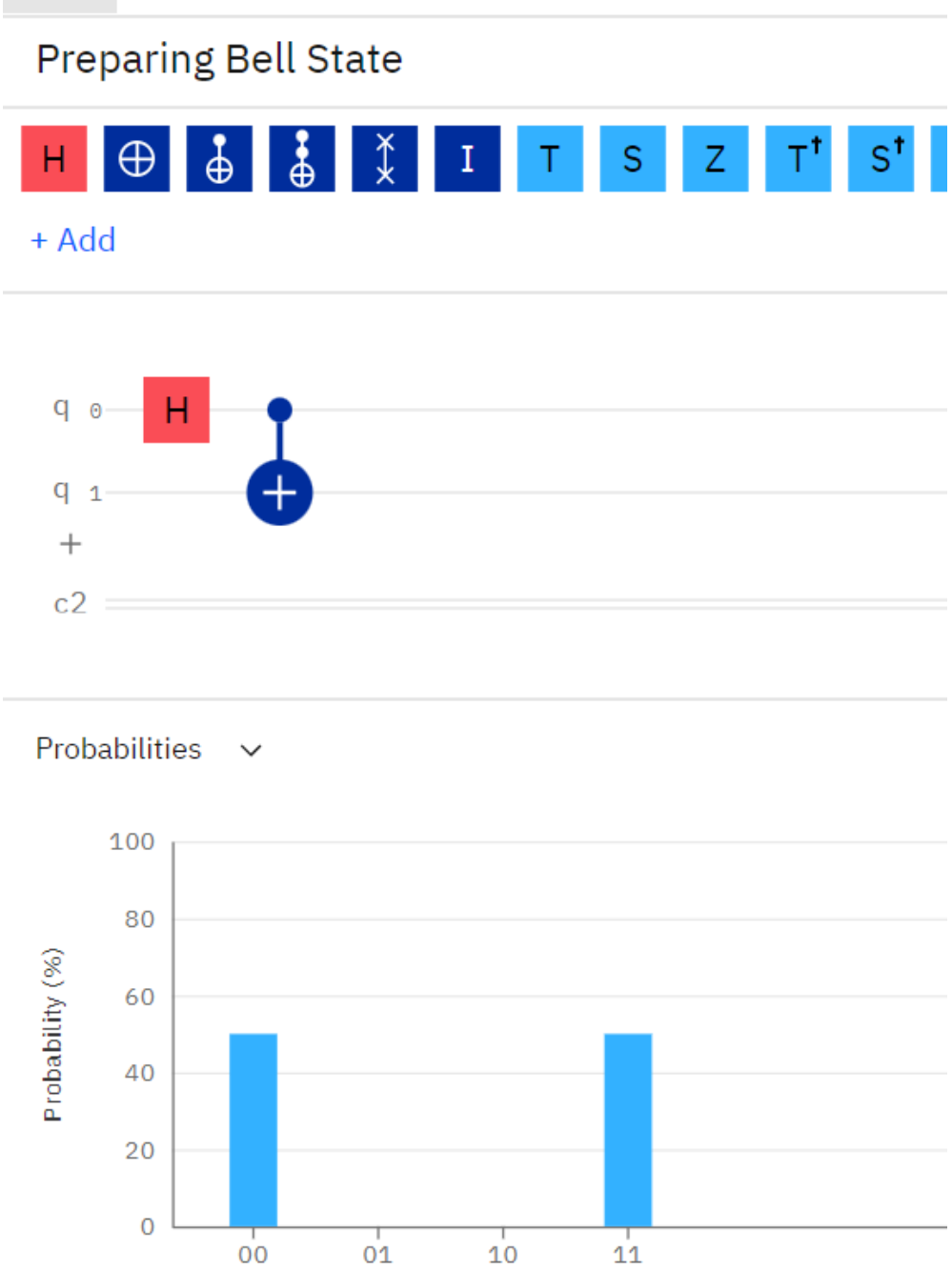
- Performs multiplication of a Matrix A with vector V and stores resultant in U.

Example: Creating a Bell State

Qiskit Implementation

```
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
from numpy import pi

qreg_q = QuantumRegister(2, 'q')
creg_c = ClassicalRegister(2, 'c')
circuit = QuantumCircuit(qreg_q, creg_c)
circuit.h(qreg_q[0])
circuit.cx(qreg_q[0], qreg_q[1])
```



QCSimulator Implementation

```
#include<math.h>
#include<complex>
#include<iostream>
using namespace std;
#include "QuantumProgram.h"


int main()
{
QuantumProgram q1(2);

q1.hadamard_gate(0);
q1.cnot_gate(0,1);
q1.create_state();
q1.read_state();
cout<<endl;
q1.draw_circuit();
}
```

Code compiled and executed in Replit Online C++ Repl
Quantum State representation convention is taken as
[00
01
10
11]
Quantum Circuit is also shown in the output.

```
Console  Shell

> clang++-7 -pthread -std=c++17 -o main QuantumProgram.cpp main.cpp
> ./main
Quantum State is:
0.707107
0
0
0.707107

Quantum Circuit:
Q0 H-C------------------
Q1 T------------------->
```

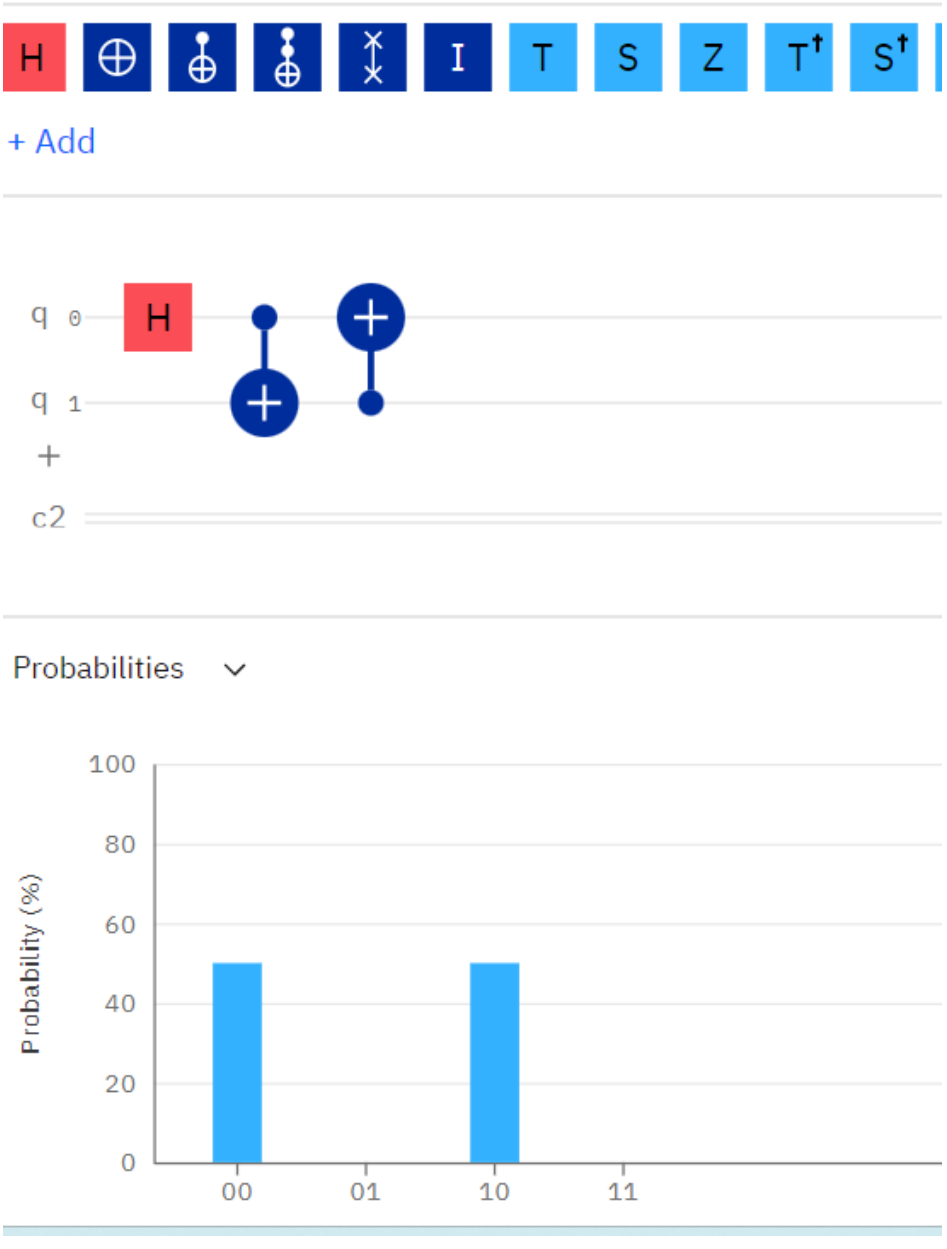
Example:

Qiskit Implementation

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

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

h q[0];
cx q[0],q[1];
cx q[1],q[0];
```



QCSimulator Implementation

```cpp
#include<math.h>
#include<complex>
#include<iostream>
using namespace std;
#include "QuantumProgram.h"


int main()
{
QuantumProgram q1(2);

q1.hadamard_gate(0);
q1.cnot_gate(0,1);
q1.cnot_gate(1, 0);
q1.create_state();
q1.read_state();
cout<<endl;
q1.draw_circuit();
}
```