МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”

Кафедра ИИТ

ОТЧЁТ
По лабораторной работе №4
«Сумматор в квантовых схемах»

Выполнил:
Студент группы ИИ-22
Кузьмич В.Н.
Проверила:
Хацкевич А.С.

Брест 2024

Цель работы: ознакомление с выполнением простого сложения с помощью квантовых схем.

Задачи:

1. Изучить теоретический материал.
2. Произвести регистрацию на сайте https://quantum.ibm.com/ для получения токена API или дальнейшей работы с интерактивной средой IBM Quantum Experience
3. Средствами Qiskit или используя средства интерактивной среды IBM Quantum Experience https://quantum.ibm.com/composer/ , создать квантовую схему полного сумматора.

Полный сумматор принимает на вход два двоичных числа плюс бит переполнения, который мы назовем X. Создайте полный сумматор с входными данными:

A=1, B=0, X=1.

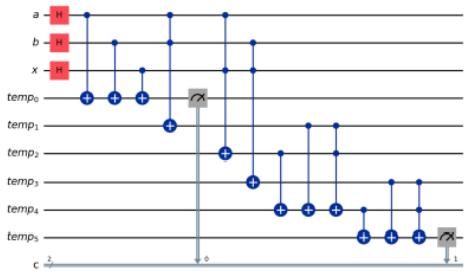
Ход работы

Код программы:

```
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
from qiskit_aer import Aer
import matplotlib
import matplotlib.pyplot as plt

matplotlib.use('QtAgg')


def get_result(a, b, x, shots_count = 1024, test=False) -> dict:

    a_reg = QuantumRegister(1, 'a')
    b_reg = QuantumRegister(1, 'b')
    x_reg = QuantumRegister(1, 'x')

    qt = QuantumRegister(6, 'temp')

    cr = ClassicalRegister(2, 'c')

    qc = QuantumCircuit(a_reg, b_reg, x_reg, qt, cr)
    if test:
        qc.h(a_reg[0])
        qc.h(b_reg[0])
        qc.h(x_reg[0])
    else:
        if a:
            qc.x(a_reg[0])

        if b:
            qc.x(b_reg[0])

        if x:
            qc.x(x_reg[0])

    a = a_reg[0]
    b = b_reg[0]
    x = x_reg[0]


    qc.cx(a, qt[0]) # A -> T[0]
    qc.cx(b, qt[0]) # A xor B -> T[0]
    qc.cx(x, qt[0]) # A xor B xor X-> T[0]

    qc.measure(qt[0], cr[0]) # T[0] -> C[0]

    qc.ccx(a, b, qt[1]) # A and B -> T[1]
    qc.ccx(a, x, qt[2]) # A and X -> T[2]
    qc.ccx(b, x, qt[3]) # B and X -> T[3]

    qc.cx(qt[2], qt[4])
```

```
    qc.cx(qt[1], qt[4])
    qc.ccx(qt[1], qt[2], qt[4])  # (A and B) or (A and X) -> T[4]

    qc.cx(qt[4], qt[5])
    qc.cx(qt[3], qt[5])
    qc.ccx(qt[3], qt[4], qt[5]) # (A and B) or (A and X) or (B and X) -> T[5]


    qc.measure(qt[5], cr[1]) # T[5] -> C[1]


    simulator = Aer.get_backend('aer_simulator')
    result = simulator.run(qc, shots=shots_count).result()
    counts = result.get_counts(qc)
    if test:
        qc.draw(output='mpl')
        plt.draw()
        plt.show()
    return counts

def main():
    shots = 10000
    temp = get_result(0,0,0,shots, True)

    print(f'{" A":3s}|{" B":3s}|{" X":3s}{" S":3s}|{" C":3s}|')
    print(f'{"---":3s}|{"---":3s}|{"---":3s}{"---":3s}|{"---":3s}|')

    for i in range(8):
        arg_a = (i & 0x4) >> 2
        arg_b = (i & 0x2) >> 1
        arg_x = i & 0x1

        res =  get_result(arg_a, arg_b, arg_x, 1)

        t = list(res.keys())[0]

        s = (list(res.keys())[0][1])
        c = (list(res.keys())[0][0])

        print(f' {str(arg_a):2s}| {str(arg_b):2s}| {str(arg_x):2s}|| {s:2s}| {c:2s}|  {(temp[t] / shots):1.4f}')


if __name__ == "main":
    main()
```

Результат работы:



A	B	X	S	C	
0	0	0	0	0	0.1239
0	0	1	0	1	0.3772
0	1	0	0	1	0.3772
0	1	1	1	0	0.3705
1	0	0	0	1	0.3772
1	0	1	1	0	0.3705
1	1	0	1	0	0.3705
1	1	1	1	1	0.1284

Вывод: ознакомился с выполнением простого сложения с помощью квантовых схем.