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
Варицкий М.И.
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

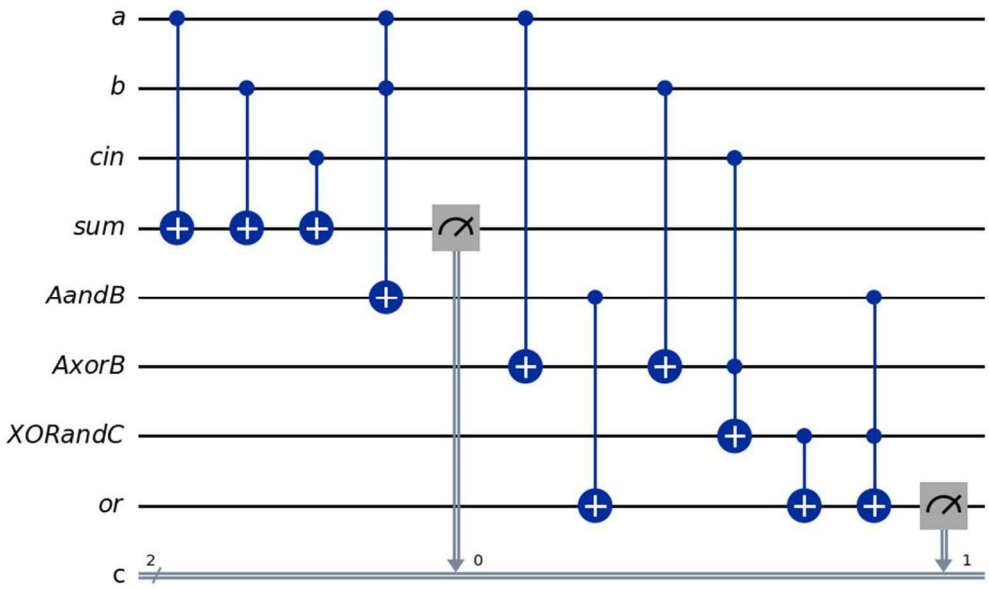
Код программы:

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


def compute(a, b, cin, shots=1024, test=False) -> dict:
    q_a = QuantumRegister(1, 'a')
    q_b = QuantumRegister(1, 'b')
    q_cin = QuantumRegister(1, 'cin')

    q_sum = QuantumRegister(1, 'sum')
    q_AandB = QuantumRegister(1, 'AandB')
    q_AxorB = QuantumRegister(1, 'AxorB')
    q_XORandC = QuantumRegister(1, 'XORandC')
    q_or = QuantumRegister(1, 'or')

    c = ClassicalRegister(2, 'c')
    qc = QuantumCircuit(q_a, q_b, q_cin, q_sum, q_AandB, q_AxorB, q_XORandC, q_or, c)

    if a:
        qc.x(q_a[0])
    if b:
        qc.x(q_b[0])
    if cin:
        qc.x(q_cin[0])

    # СУММА S
    qc.cx(q_a[0], q_sum[0])
    qc.cx(q_b[0], q_sum[0])
    qc.cx(q_cin[0], q_sum[0])
    qc.measure(q_sum[0], c[0])  # измеряем бит для определения суммы

    # ПЕРЕХОДЯЩИЙ БИТ Cout
    qc.ccx(q_a[0], q_b[0], q_AandB[0])  # A and B
    qc.cx(q_a[0], q_AxorB[0])  # A xor B
    qc.cx(q_b[0], q_AxorB[0])  # A xor B
    qc.ccx(q_AxorB[0], q_cin[0], q_XORandC[0])  # AxorB and Cin
    qc.cx(q_AandB[0], q_or[0])  # OR
    qc.cx(q_XORandC[0], q_or[0])  # OR
    qc.ccx(q_AandB[0], q_XORandC[0], q_or[0])  # OR
    qc.measure(q_or[0], c[1])  # измеряем бит для определения переходящего бита

    simulator = Aer.get_backend('aer_simulator')
    result = simulator.run(qc, shots=shots).result()
    counts = result.get_counts(qc)
    if test:
        qc.draw(output='mpl')  # Рисуем квантовую схему
```

```
        plt.show()  # Показываем рисунок
    return counts


def main():
    print(f" {'A'} | {'B'} | {'X'} || {'S'} | {'C'} |")

    for i in range(8):
        a = (i & 0b100) >> 2
        b = (i & 0b10) >> 1
        x = i & 0b1

        result = compute(a, b, x, 1)

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

        print(f' {str(a)} | {str(b)} | {str(x)} || {s} | {c:} |')

    compute(0, 0, 0, 1, True)  # Визуализируем последнюю квантовую схему


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

Результат работы:

```
A | B | X || S | C |
0 | 0 | 0 || 0 | 0 |
0 | 0 | 1 || 1 | 0 |
0 | 1 | 0 || 1 | 0 |
0 | 1 | 1 || 0 | 1 |
1 | 0 | 0 || 1 | 0 |
1 | 0 | 1 || 0 | 1 |
1 | 1 | 0 || 0 | 1 |
1 | 1 | 1 || 1 | 1 |
```



Схема

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