Programming Assignment Report

COE 530 – Programming Assignment 1: Half Adder, Full Adder, and 3-bit Ripple-Carry Adder (Qiskit)

Quantum Computer & Architecture
A report submitted by

Feras Alsayigh
Student ID: 202515470
Email: g202515470@kfupm.edu.sa

Under the supervision of

Dr. Muhamad Felemban
Email: mfeleman@kfupm.edu.sa

Department of Computer Engineering
King Fahd University of Petroleum and Minerals
Dhahran, Saudi Arabia

Due date: Saturday 28-2-2026

Contents

1 Project Overview

Objective

The objective of this programming assignment is to get familiar with a quantum SDK by implementing a **3-bit ripple-carry adder** that adds two 3-bit binary numbers. The assignment is divided into tasks and should be completed individually.

Reading Reference

Chapter 5, "Quantum Programming" in *Introduction to Classical and Quantum Computing.*

Assignment Description

It is recommended to complete the tasks in order after becoming familiar with a quantum SDK. IBM Qiskit may be used to implement and execute the required circuits.

2 Project Methodology

Tools and Technologies

- IBM Qiskit (for implementing circuits programmatically)
- IBM Quantum Composer (optional: for circuit drawing and screenshots)
- IBM Quantum Lab (for executing circuits, especially larger circuits)
- IBM Quantum hardware backend (for real-device execution)

Methods

This work was carried out by implementing the required circuits, verifying them using simulation (Statevector and Probabilities), and then executing selected circuits on real quantum hardware. Screenshots were collected as required in each task.

3 Project Implementation

Task 1: Half Adder (HA)

Classical Half Adder (Textbook Reference: Page 29)

According to Page 29 of the reference textbook, the classical Half Adder takes two input bits a and b and produces:

$$S = a \oplus b$$

$$C = a \cdot b$$

where:

- $\oplus$ is the XOR operation (Sum output),
- $\cdot$ is the AND operation (Carry output).

The classical circuit consists of one XOR gate and one AND gate as shown in Figure 1.

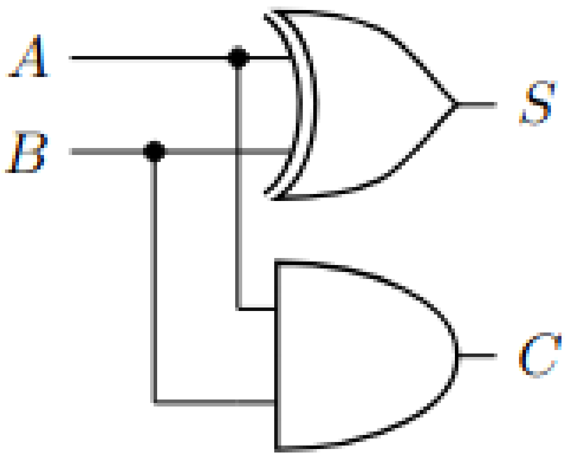


Figure 1: Classical Half Adder (Page 29)

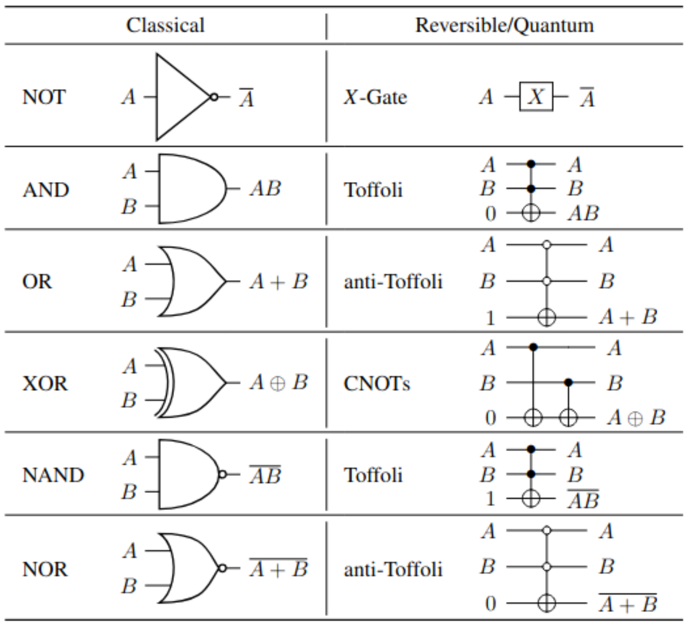
Reversible Gate Mapping (Textbook Reference: Page 170)

As shown in the reversible gate equivalence table on Page 170:

- XOR can be implemented using two CNOT gates.
- AND can be implemented using a Toffoli (CCX) gate.

Because quantum circuits must be reversible, additional qubits initialized to $|0\rangle$ are introduced to store the outputs without destroying the inputs.

The reversible mapping table used is shown in Figure 2.



Classical		Reversible/Quantum	
NOT	$A \to \overline{A}$	X-Gate	$A \to \overline{A}$
AND	$A, B \to AB$	Toffoli	$A, B, 0 \to A, B, AB$
OR	$A, B \to A+B$	anti-Toffoli	$A, B, 1 \to A, B, A+B$
XOR	$A, B \to A \oplus B$	CNOTs	$A, B, 0 \to A, B, A \oplus B$
NAND	$A, B \to \overline{AB}$	Toffoli	$A, B, 1 \to A, B, \overline{AB}$
NOR	$A, B \to \overline{A+B}$	anti-Toffoli	$A, B, 0 \to A, B, \overline{A+B}$

Figure 2: Reversible Gate Equivalence Table (Page 170)

Quantum Half Adder Implementation

Based on the mapping:

- Apply CNOT gates to compute $S = a \oplus b$.
- Apply a Toffoli gate to compute $C = a \cdot b$.

The resulting quantum circuit preserves the input qubits and writes the outputs to additional qubits, resulting in the state:

$$|a\,b\,S\,C\rangle$$

The implemented quantum circuit is shown in Figure 3.

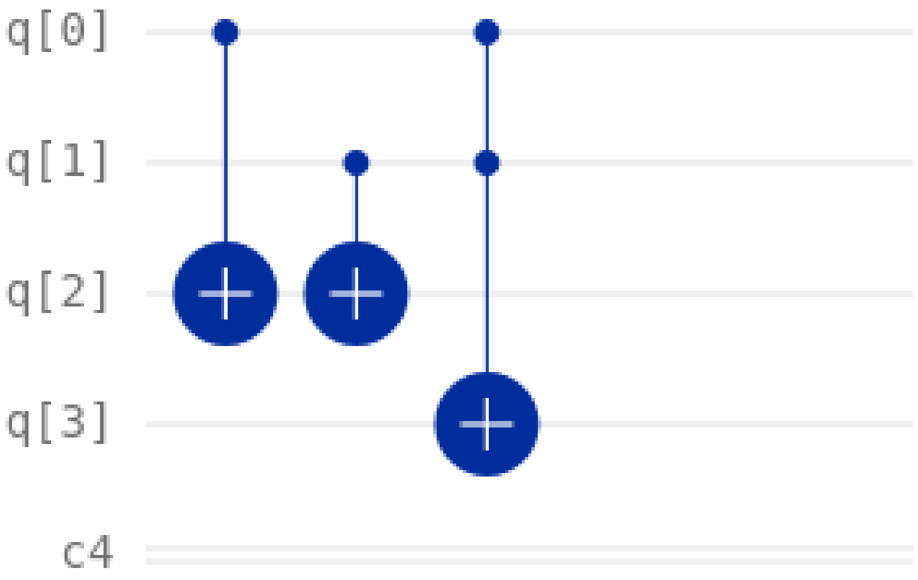


Figure 3: Quantum Half Adder Implementation

Screenshots for Required Input Combinations

As required in Task 1, the circuit was executed for all input combinations:

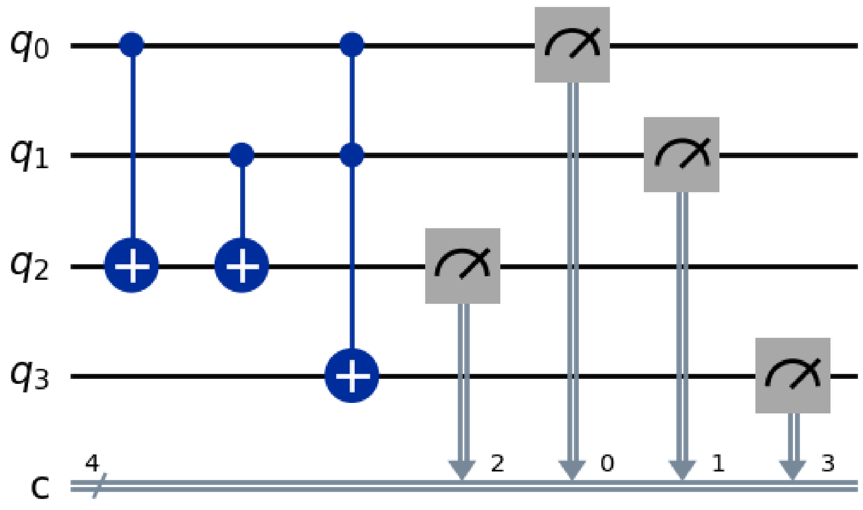


Figure 4: Half Adder Circuit for $a = 0, b = 0$

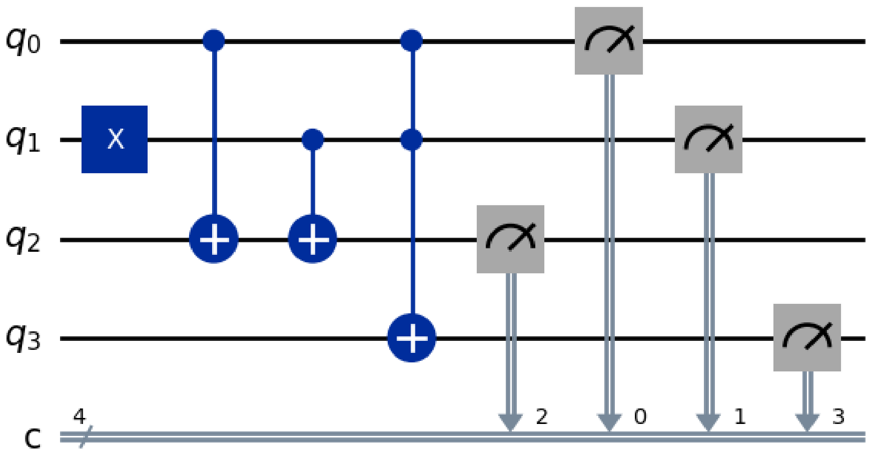


Figure 5: Half Adder Circuit for $a = 0, b = 1$

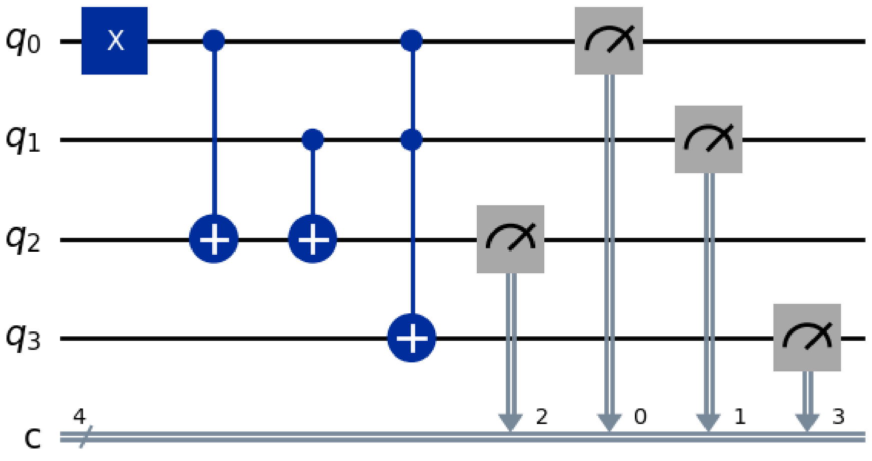


Figure 6: Half Adder Circuit for $a = 1, b = 0$

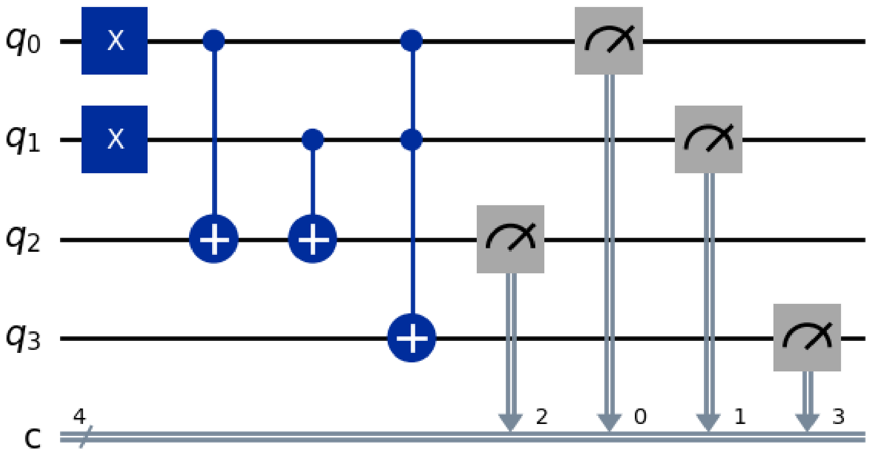


Figure 7: Half Adder Circuit for $a = 1, b = 1$

For example, for input $a = 1, b = 1$, the resulting state was:

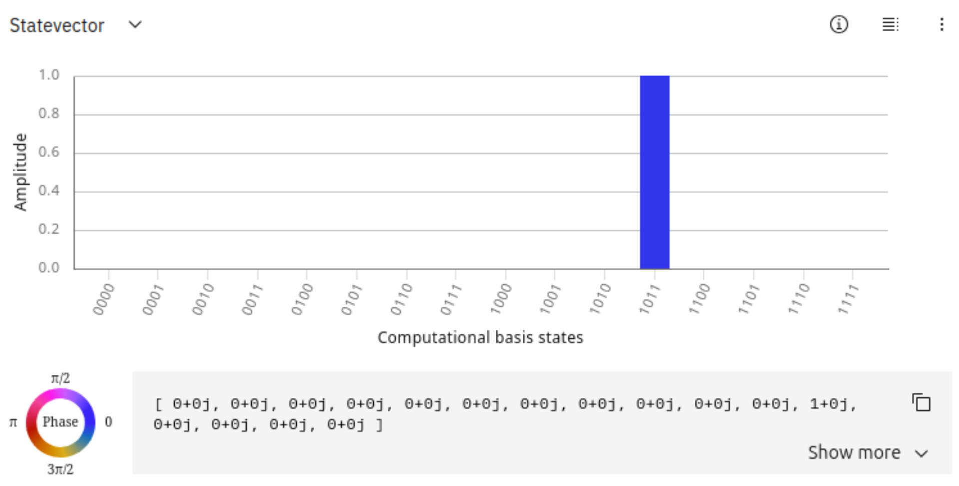
$$|1\,0\,1\,1\rangle$$

which corresponds to:

- Sum $= 0$
- Carry $= 1$

Task 2: Interpreting HA Outputs

Take a screenshot of the outputs of **one** of the circuits in Task 1. Explain how to interpret the output from the **Statevector** and **Probabilities** figures in IBM Q.
Selected case: $a = 1, b = 1$
Screenshots:



```
[ 0+0j, 0+0j, 0+0j, 0+0j, 0+0j, 0+0j, 0+0j, 0+0j, 0+0j, 0+0j, 0+0j, 1+0j,
0+0j, 0+0j, 0+0j, 0+0j ]
```

Figure 8: Statevector output for Half Adder with $a = 1, b = 1$

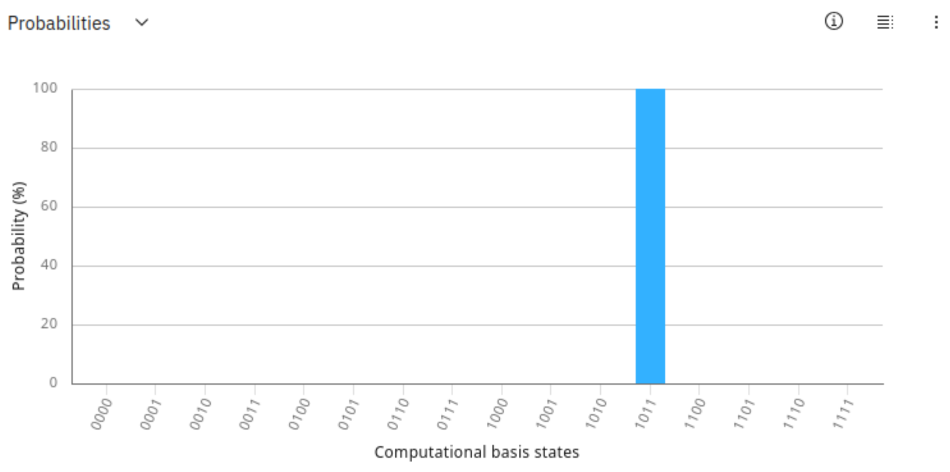


Figure 9: Probability histogram for Half Adder with $a = 1, b = 1$

Explanation:

For the selected case $a = 1, b = 1$, the classical Half Adder computes:

$$S = a \oplus b = 1 \oplus 1 = 0$$

$$C = a \cdot b = 1 \cdot 1 = 1$$

Since the quantum circuit is reversible, the input qubits are preserved and the outputs are written to additional qubits. Therefore, the expected final state is:

$$|C\,S\,b\,a\rangle = |1\,0\,1\,1\rangle$$

Statevector Interpretation:

A general quantum state is written as:

$$|\psi\rangle = \sum_i \alpha_i |i\rangle$$

where α_i are complex amplitudes corresponding to each computational basis state.

From the Statevector figure, the simulator shows that the only non-zero amplitude corresponds to the basis state $|1011\rangle$, and its amplitude is equal to 1. This means:

$$|\psi\rangle = 1|1011\rangle$$

All other basis states have amplitude 0. Therefore, the system is entirely in the state $|1011\rangle$.

Given the qubit ordering $|q_3\, q_2\, q_1\, q_0\rangle$, where:

- $q_0 = a$
- $q_1 = b$
- $q_2 = S$
- $q_3 = C$

the bitstring 1011 corresponds to:

- Carry $(q_3) = 1$
- Sum $(q_2) = 0$
- $b\ (q_1) = 1$
- $a\ (q_0) = 1$

which matches the expected Half Adder result.

Probabilities Interpretation:

Measurement probabilities are calculated as:

$$P(i) = |\alpha_i|^2$$

Since the amplitude of $|1011\rangle$ is 1:

$$P(1101) = |1|^2 = 1$$

The probability histogram therefore displays a single bar at 1011 with probability 1, while all other basis states have probability 0. This confirms that the circuit behaves deterministically for classical inputs and correctly computes the Half Adder output.

Task 3: Full Adder (FA)

The Full Adder takes three inputs: a, b, and c_{in}, and produces:

$$S = a \oplus b \oplus c_{in}$$

$$C_{out} = ab + c_{in}(a \oplus b)$$

1. Classical Full Adder (Two Half Adders)

From the reference (Page 33), the classical Full Adder can be constructed using two Half Adders and one OR gate, as shown in Figure 10.

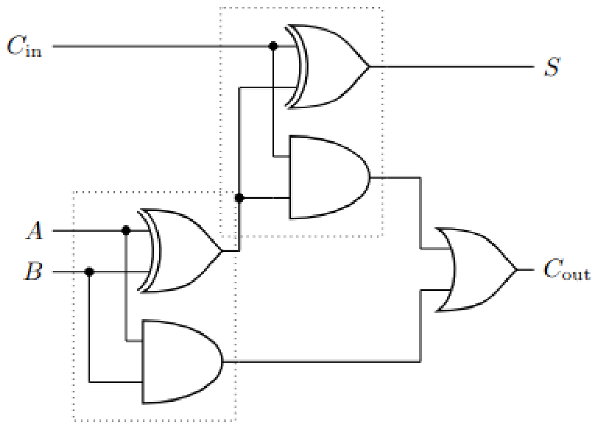


Figure 10: Classical Full Adder implemented using two Half Adders (Page 33)

The first Half Adder computes:

$$A \oplus B$$

The second Half Adder adds c_{in}:

$$S = A \oplus B \oplus c_{in}$$

The carry output is computed as:

$$C_{out} = AB + c_{in}(A \oplus B)$$

However, this circuit is not reversible because the outputs do not uniquely determine the inputs.

2. Making the Full Adder Reversible (Page 168)

As explained in Section 4.5.2 (Page 168), the classical Full Adder is not reversible because information is lost. To make it reversible, extra bits are introduced and XORed with the outputs, producing a reversible mapping.

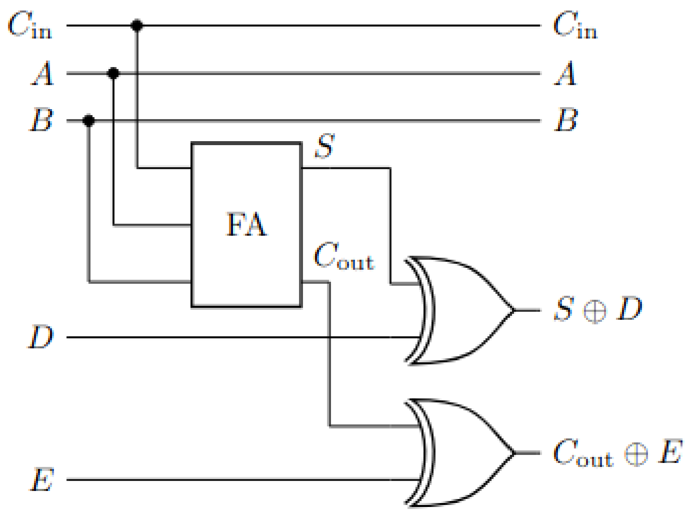


Figure 11: Reversible Full Adder Construction (Page 168)

This reversible construction ensures that the number of inputs equals the number of outputs, making it a valid quantum gate.

3. Transforming to a Quantum Circuit (Page 170)

Using the reversible gate equivalence table (Page 170):

- XOR → CNOT
- AND → Toffoli (CCX)
- OR → Implemented using Toffoli + X gates

Replacing each classical gate with its reversible quantum equivalent results in the quantum Full Adder circuit shown in Figure 12.

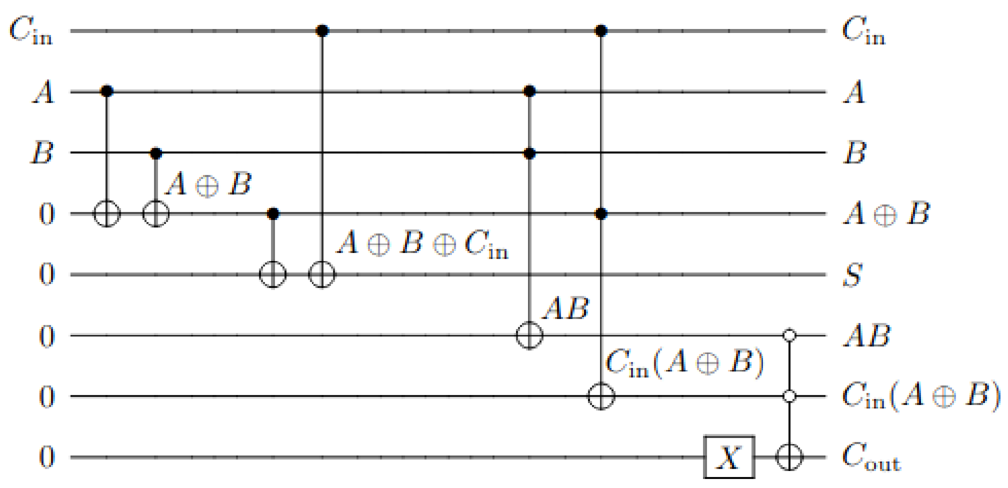


Figure 12: Quantum Full Adder using CNOT and Toffoli Gates (Page 170)

The circuit performs:

- Two CNOTs to compute $A \oplus B$
- Two additional CNOTs to compute $S = A \oplus B \oplus c_{in}$
- Toffoli gates to compute AB and $c_{in}(A \oplus B)$
- Final combination to compute C_{out}

4. Cleaning Ancilla Bits (Page 171)

The implementation introduces ancilla (temporary) qubits for intermediate computations such as:

$$A \oplus B, \quad AB, \quad c_{in}(A \oplus B)$$

In quantum circuits, ancilla bits must be reset to $|0\rangle$ to avoid unintended entanglement and to allow reuse.

This is achieved through **uncomputation**, where the inverse sequence of gates is applied to return the ancilla qubits to zero.

The cleaned Full Adder circuit is shown in Figure 13.

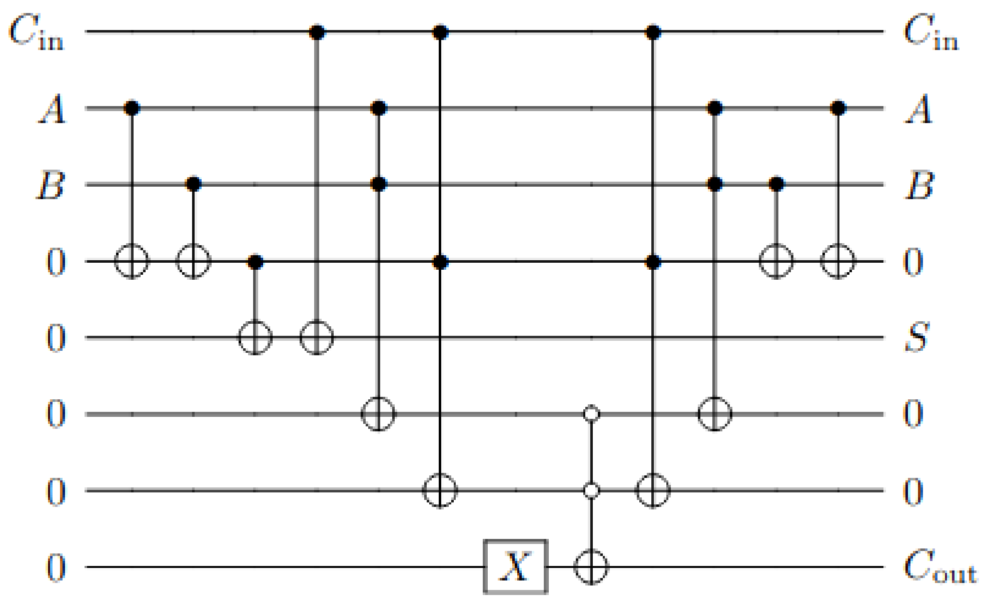


Figure 13: Full Adder with Ancilla Uncomputation (Page 171)

Since CNOT and Toffoli gates are self-inverse, applying them in reverse order restores the ancilla qubits to zero while preserving the final outputs S and C_{out}.

5. Implementation and Required Screenshots

The Full Adder was implemented using IBM Qiskit following the reversible transformation described earlier.

Initially, the circuit was constructed and visualized using IBM Quantum Composer. However, when attempting to display the **Statevector** and **Probabilities**, the Composer did not allow statevector simulation because the circuit uses multiple ancilla qubits, increasing the total qubit count beyond the Composer visualization limit.

Figure 14 shows the Full Adder circuit drawn in IBM Composer.

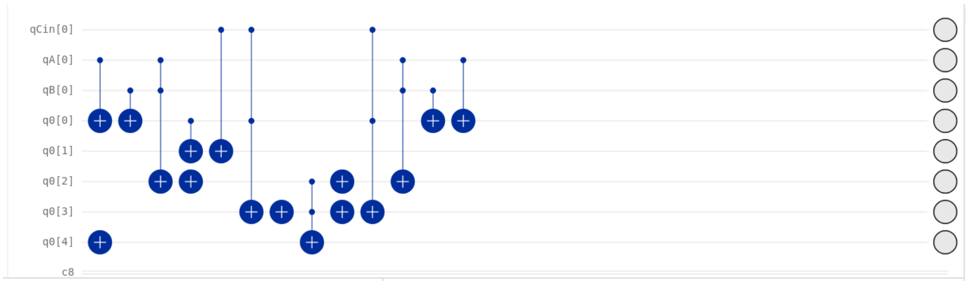
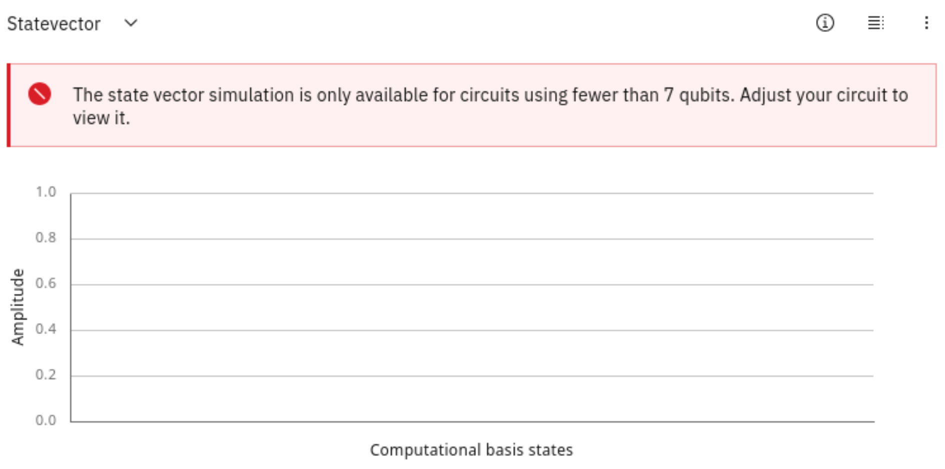


Figure 14: Full Adder circuit constructed in IBM Quantum Composer

Figure 15 shows the Composer message indicating that statevector visualization is not available for circuits of this size.



The state vector simulation is only available for circuits using fewer than 7 qubits. Adjust your circuit to view it.

Figure 15: IBM Quantum Composer limitation for statevector visualization

To overcome this restriction, the circuit was simulated locally using Qiskit. The statevector was obtained using the `Statevector` class, and the probability distribution was generated and visualized using `matplotlib`. This approach provides equivalent theoretical information without qubit-number limitations.

It is important to note that IBM Quantum Lab was not used at this stage, since it is explicitly required later in Task 6.

After verifying the correctness of the implementation through local simulation, the circuit was executed for all input combinations:

$$(a, b, c_{in}) \in \{0, 1\}^3$$

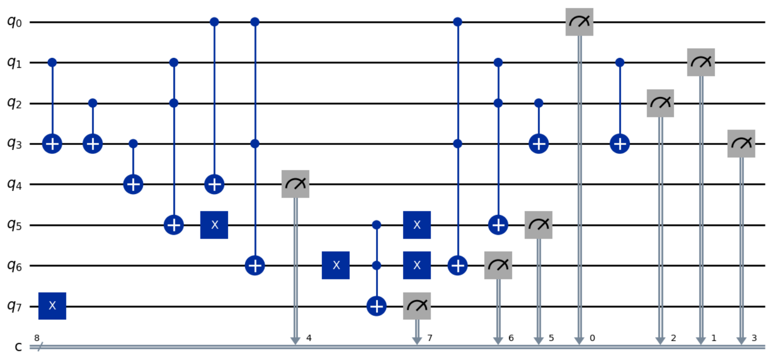


Figure 16: Full Adder circuit for $(cin, a, b) = (0, 0, 0)$

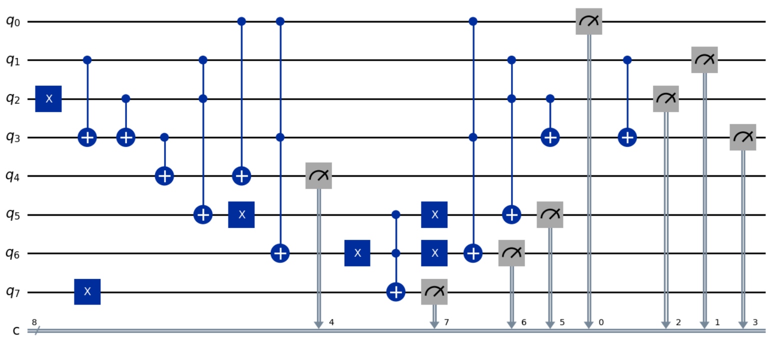


Figure 17: Full Adder circuit for $(cin, a, b) = (0, 0, 1)$

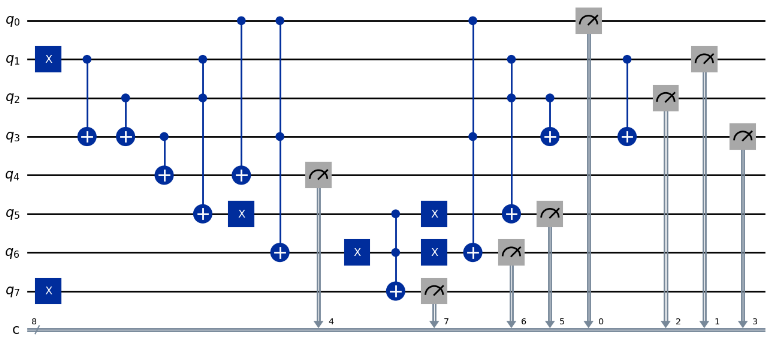


Figure 18: Full Adder circuit for $(cin, a, b) = (0, 1, 0)$

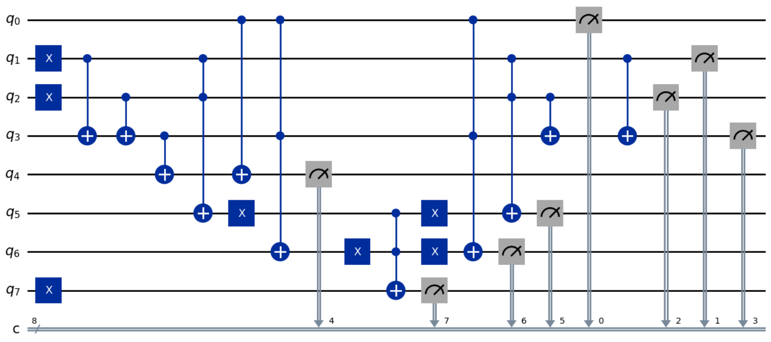


Figure 19: Full Adder circuit for $(cin, a, b) = (0, 1, 1)$

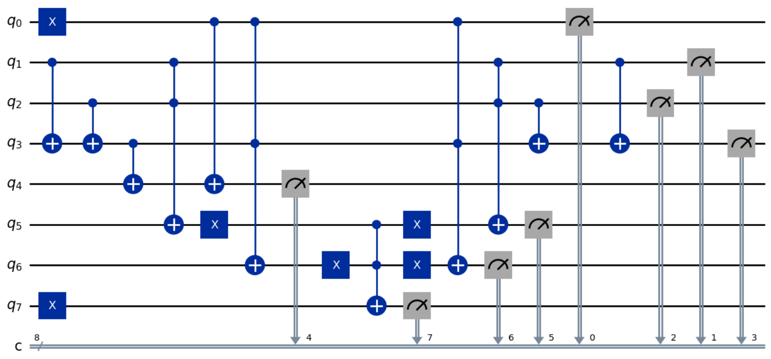


Figure 20: Full Adder circuit for $(cin, a, b) = (1, 0, 0)$

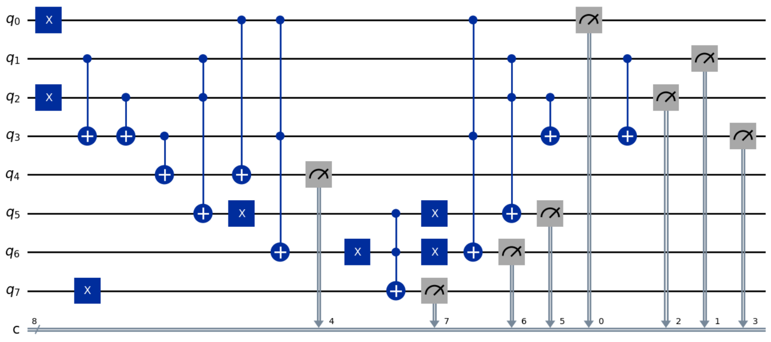


Figure 21: Full Adder circuit for $(cin, a, b) = (1, 0, 1)$

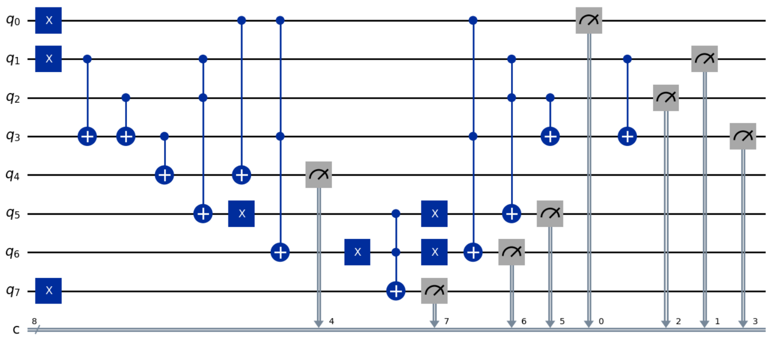


Figure 22: Full Adder circuit for $(cin, a, b) = (1, 1, 0)$

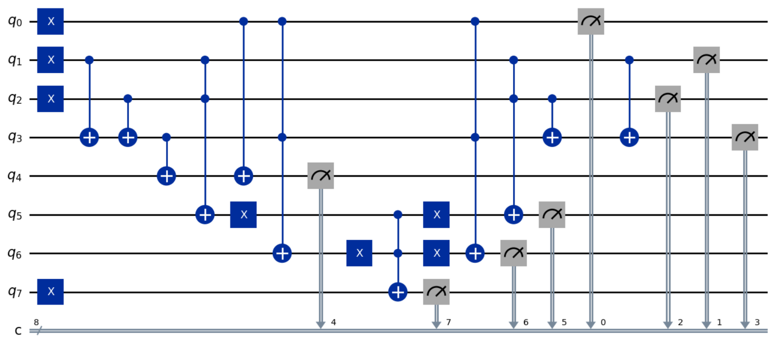


Figure 23: Full Adder circuit for $(cin, a, b) = (1, 1, 1)$

6. Output Interpretation (Example Case)

For example, for input $(a, b, c_{in}) = (1, 1, 1)$:

$$S = 1 \oplus 1 \oplus 1 = 1$$

$$C_{out} = 1 \cdot 1 + 1(1 \oplus 1) = 1$$

Since the circuit includes ancilla qubits and follows the ordering

$$|q_7\, q_6\, q_5\, q_4\, q_3\, q_2\, q_1\, q_0\rangle,$$

the resulting quantum state after execution is:

$$|10010111\rangle$$

which corresponds to:

- $q_0 = 1 \Rightarrow c_{in} = 1$
- $q_1 = 1 \Rightarrow a = 1$
- $q_2 = 1 \Rightarrow b = 1$
- $q_3 = 0 \Rightarrow$ ancilla
- $q_4 = 1 \Rightarrow S = 1$
- $q_5 = 0 \Rightarrow$ ancilla

- $q_6 = 0 \Rightarrow$ ancilla
- $q_7 = 1 \Rightarrow C_{out} = 1$

Statevector Interpretation The Statevector visualization shows that the amplitude of the computational basis state $|10010111\rangle$ is equal to 1, while all other basis states have amplitude 0. Therefore,

$$|\psi\rangle = |10010111\rangle$$

indicating that the system is entirely in a single deterministic state.

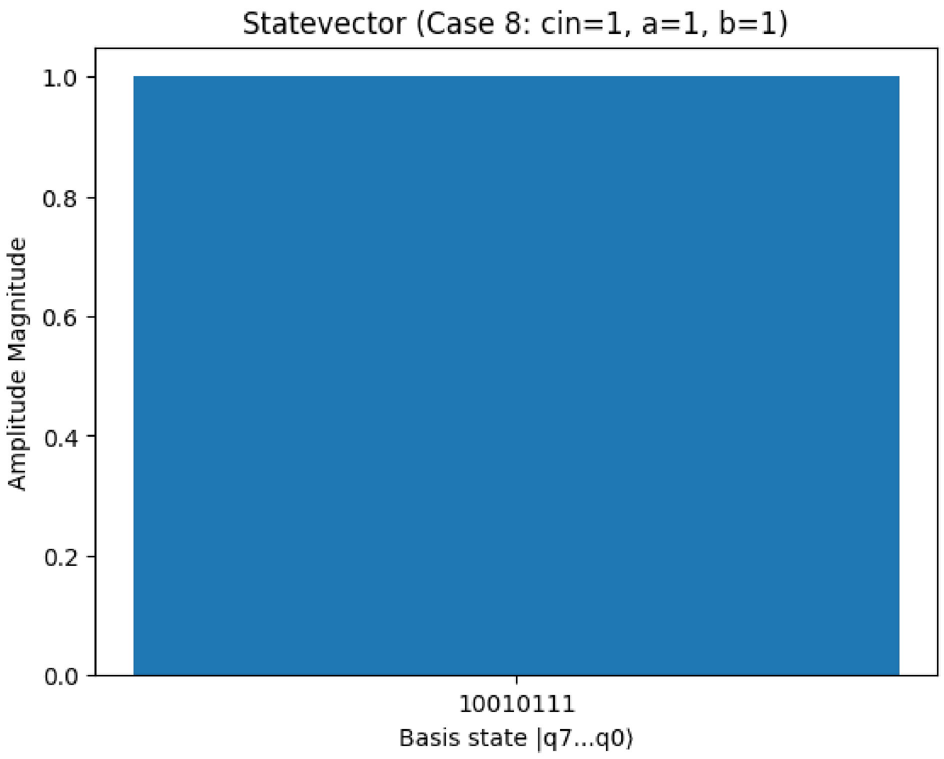


Figure 24: Statevector visualization for $(a, b, c_{in}) = (1, 1, 1)$

Probability Interpretation Measurement probabilities are given by the squared magnitude of the amplitudes:

$$P(i) = |\alpha_i|^2$$

Since the amplitude of $|10010111\rangle$ is 1, the probability is:

$$P(10010111) = |1|^2 = 1$$

The probability histogram therefore shows a single bar at $|10010111\rangle$ with probability

1, confirming deterministic classical behavior and correct ancilla uncomputation.

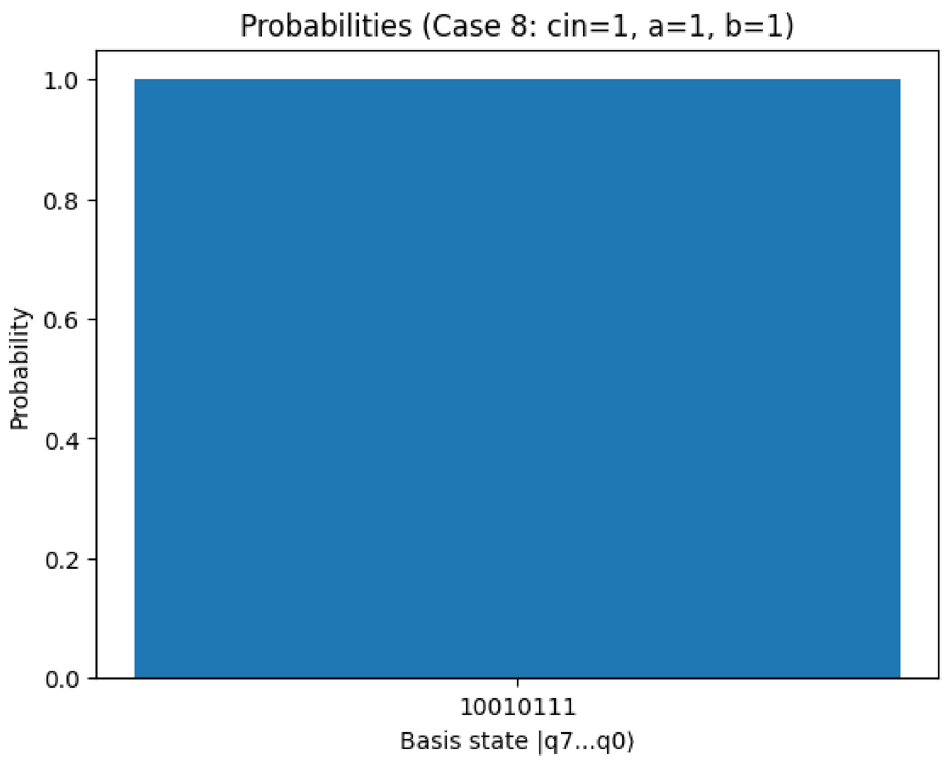


Figure 25: Probability distribution for $(a, b, c_{in}) = (1, 1, 1)$

Task 4: Superposition Input

In this task, instead of assigning a fixed classical value to the input a, the qubit representing a was placed in a superposition state using a Hadamard gate:

$$H|0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}$$

This was implemented in the circuit by applying:

$$H(q_a)$$

before the Full Adder operations.

As a result, the input to the Full Adder becomes:

$$\frac{1}{\sqrt{2}}\Big(|0, b, c_{in}\rangle + |1, b, c_{in}\rangle\Big)$$

Due to the linearity of quantum operations, the Full Adder computes on both branches simultaneously, producing a superposition of two classical results.

Screenshots (All Cases):

Case 1: $c_{in} = 0$**,** $b = 0$

Circuit

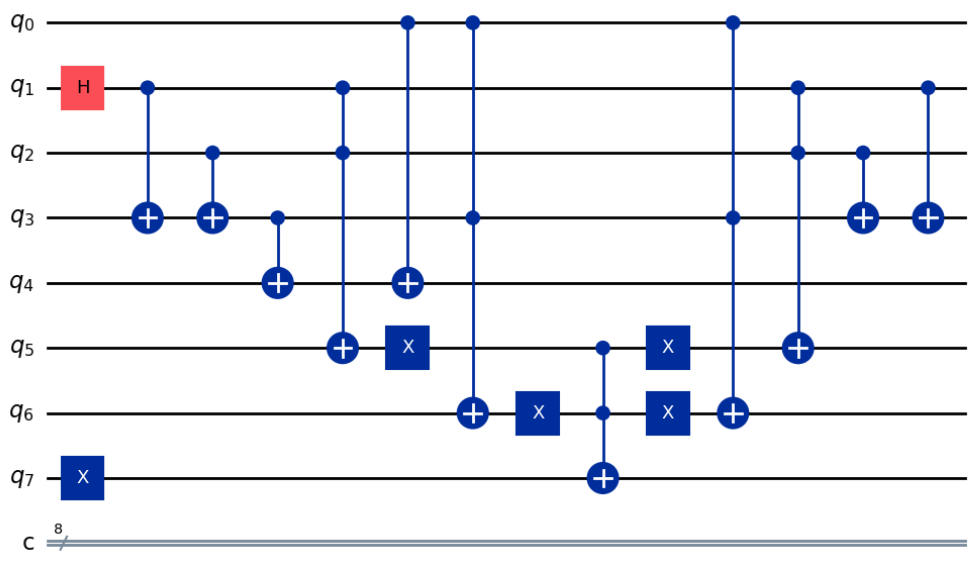


Figure 26: Full Adder circuit with a in superposition, $c_{in} = 0$, $b = 0$

Statevector Amplitudes

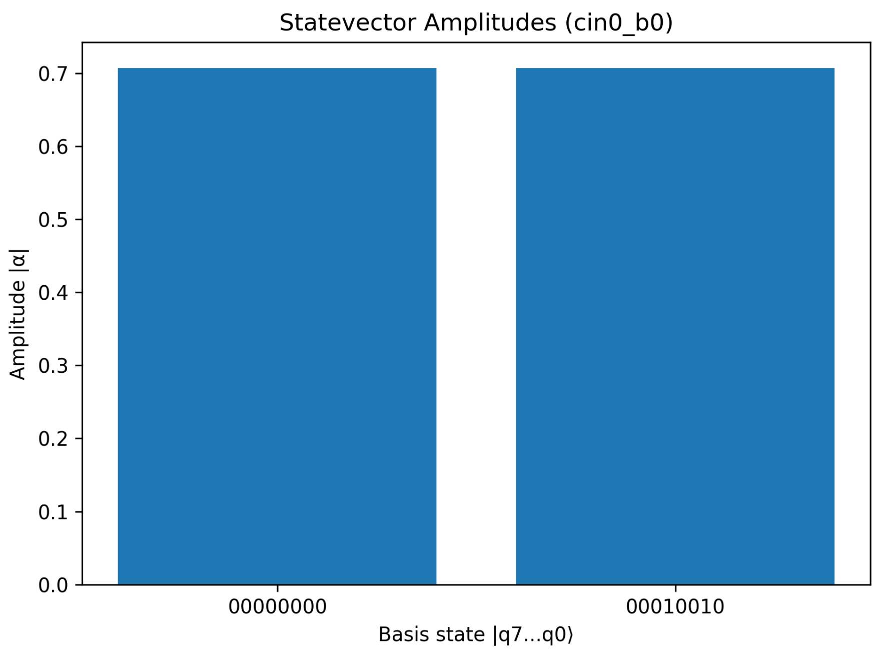


Figure 27: Statevector amplitudes for $c_{in} = 0$, $b = 0$

Q-Sphere Representation

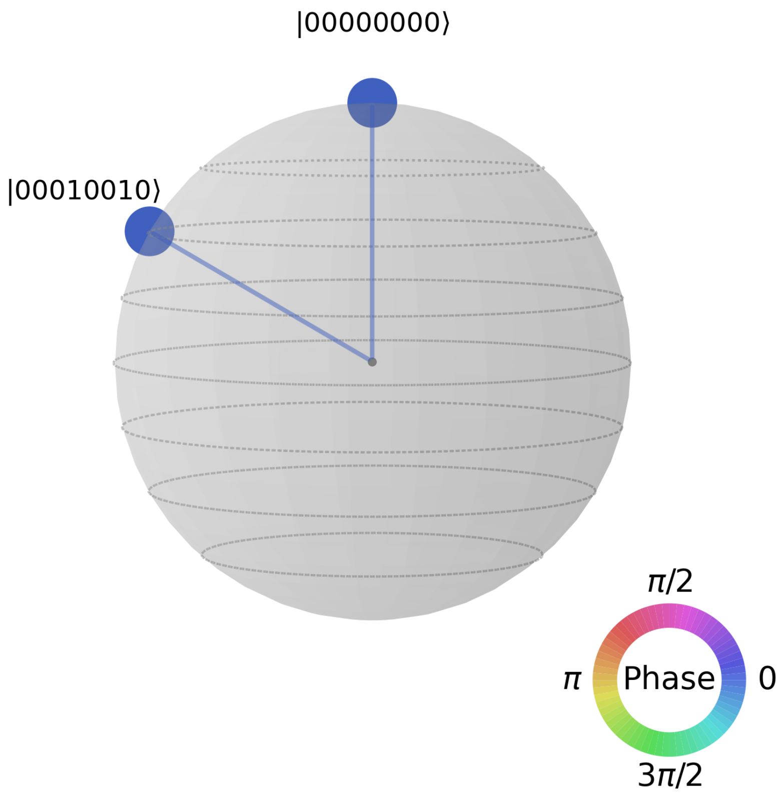


Figure 28: Q-sphere representation for $c_{in} = 0$, $b = 0$

Probabilities

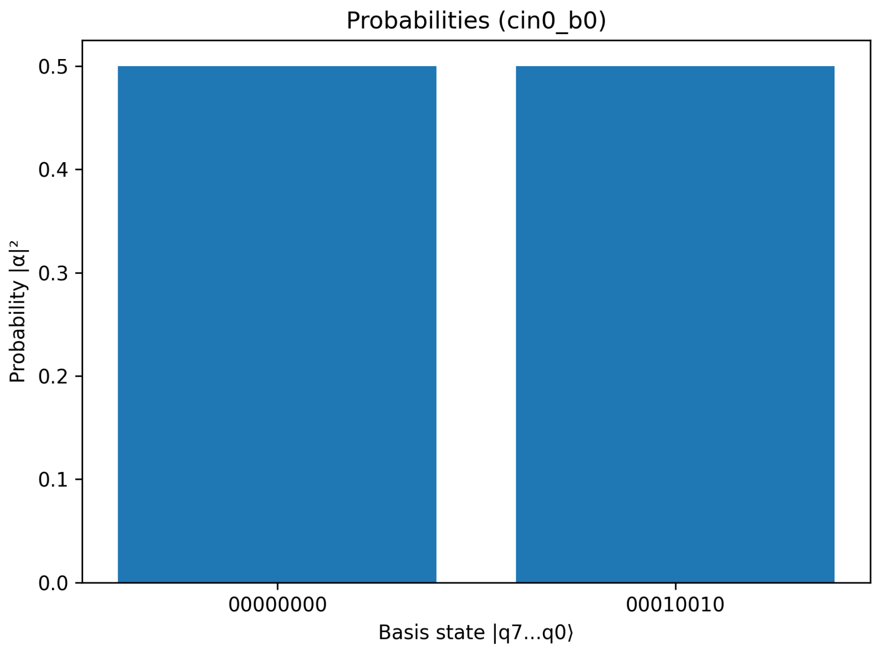


Figure 29: Measurement probabilities for $c_{in} = 0$, $b = 0$

Case 2: $c_{in} = 0$, $b = 1$

Circuit

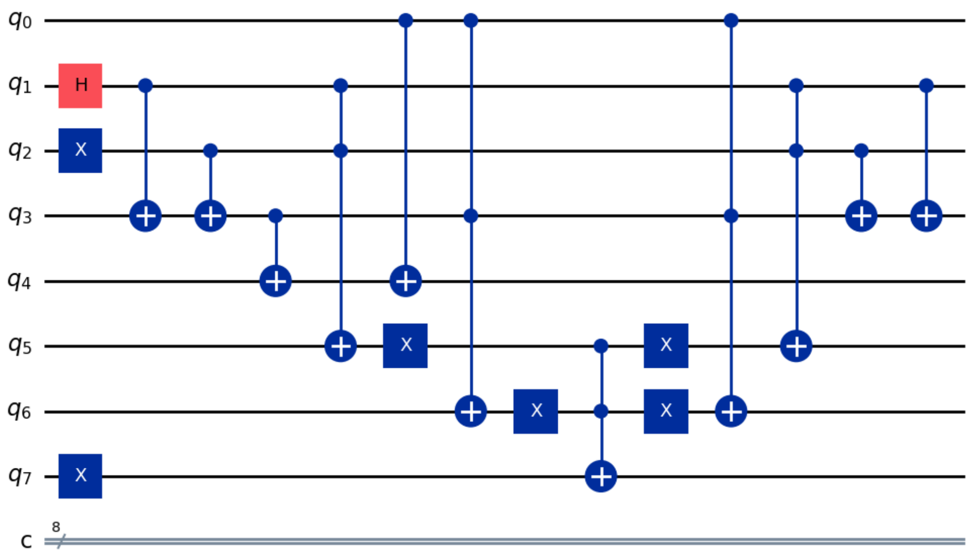


Figure 30: Full Adder circuit with a in superposition, $c_{in} = 0$, $b = 1$

Statevector Amplitudes

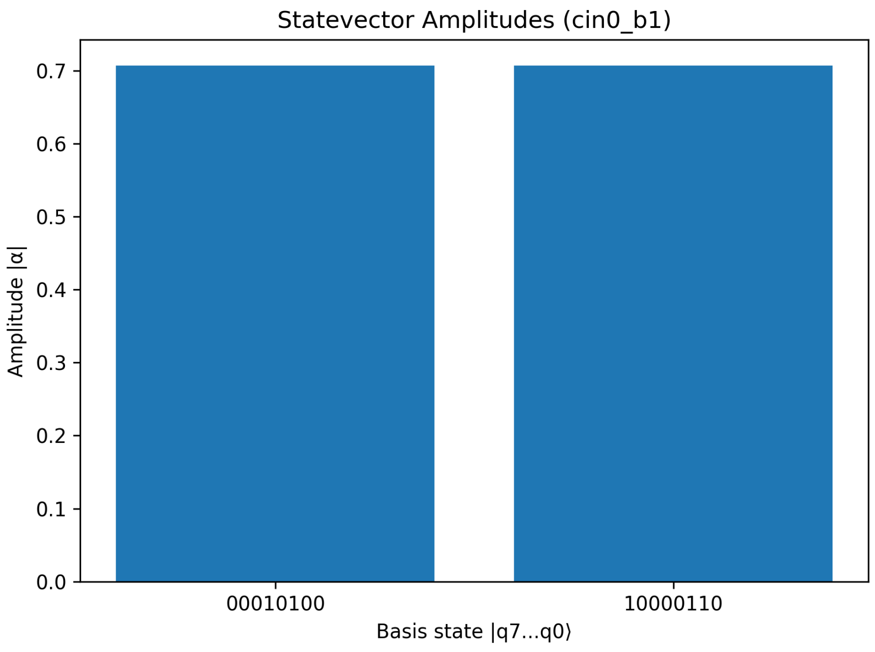


Figure 31: Statevector amplitudes for $c_{in} = 0$, $b = 1$

Q-Sphere Representation

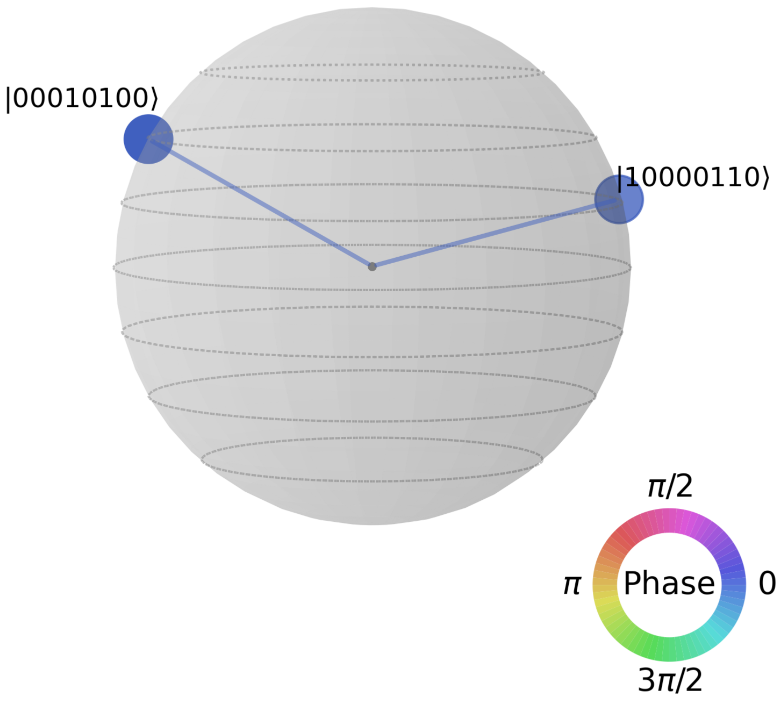


Figure 32: Q-sphere representation for $c_{in} = 0$, $b = 1$

Probabilities

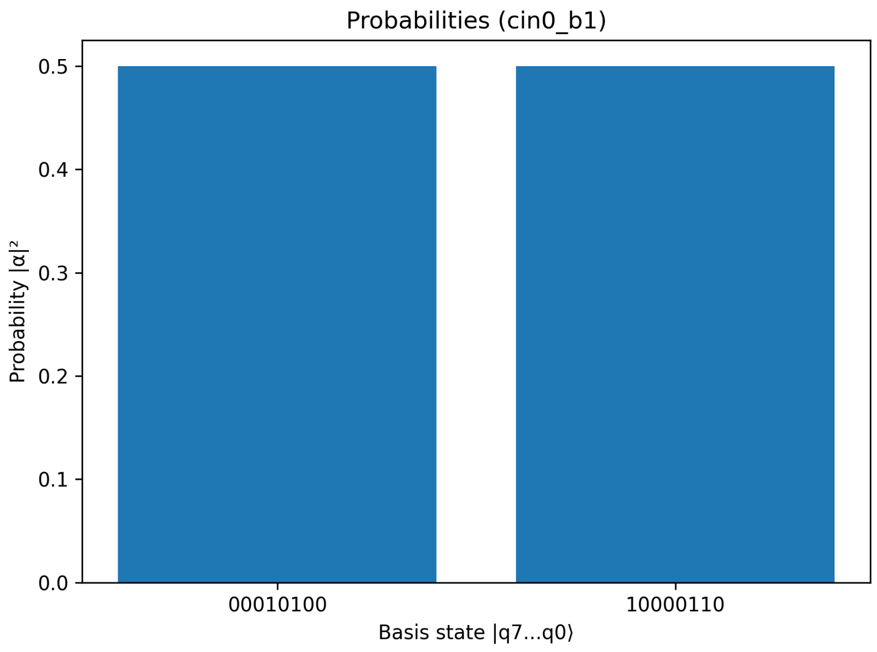


Figure 33: Measurement probabilities for $c_{in} = 0$, $b = 1$

Case 3: $c_{in} = 1,\ b = 0$

Circuit

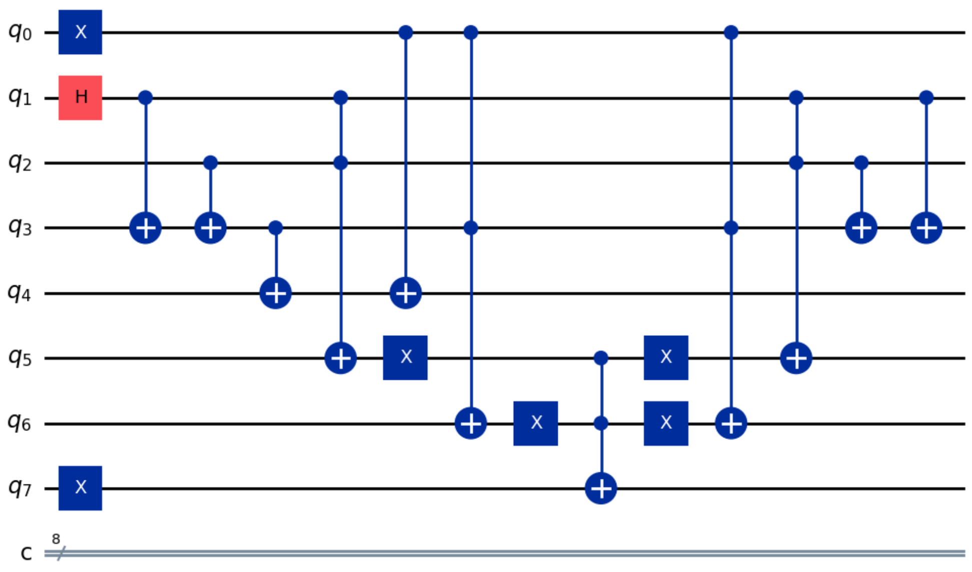


Figure 34: Full Adder circuit with a in superposition, $c_{in} = 1$, $b = 0$

Statevector Amplitudes

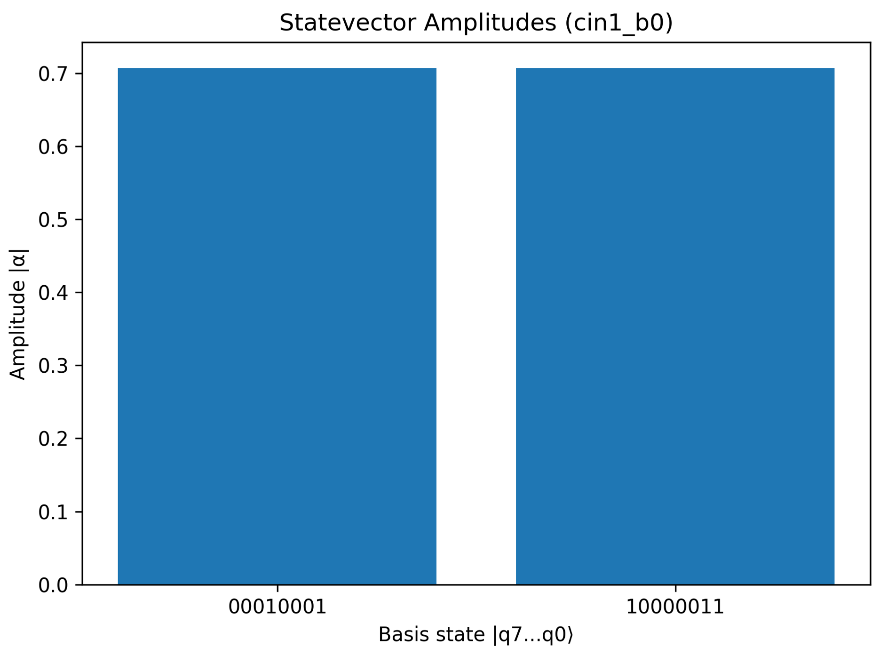


Figure 35: Statevector amplitudes for $c_{in} = 1$, $b = 0$

Q-Sphere Representation

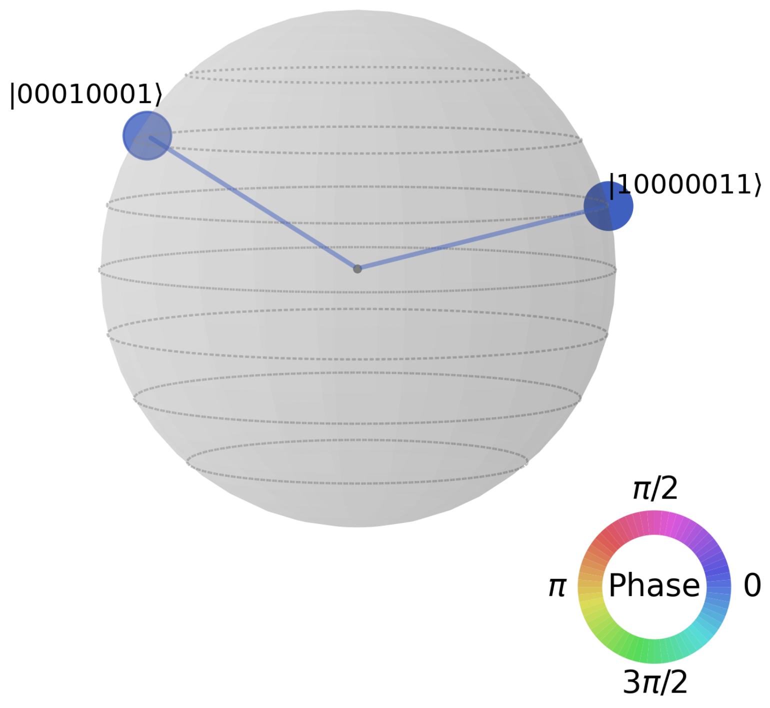


Figure 36: Q-sphere representation for $c_{in} = 1$, $b = 0$

Probabilities

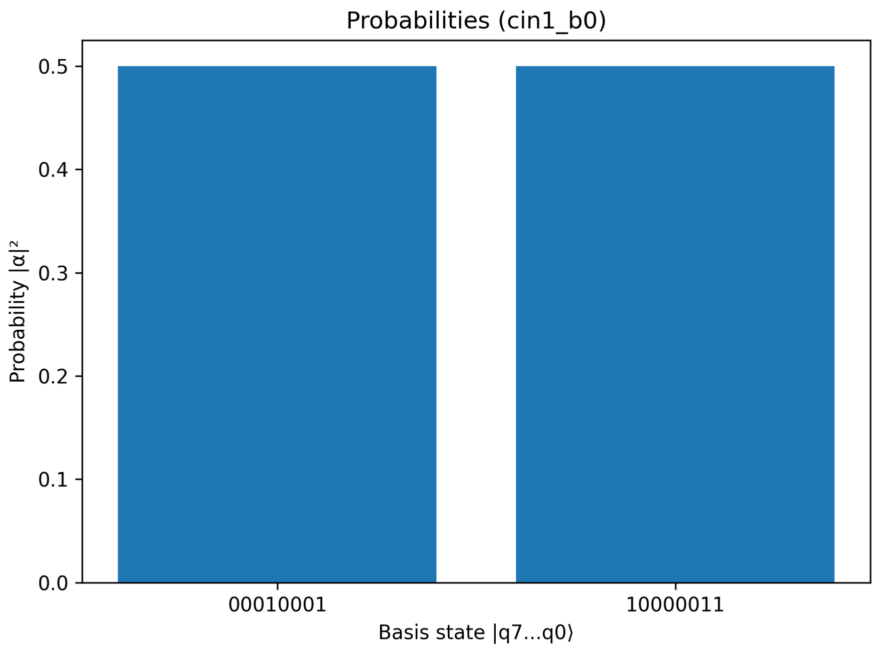


Figure 37: Measurement probabilities for $c_{in} = 1$, $b = 0$

Case 4: $c_{in} = 1,\ b = 1$

Circuit

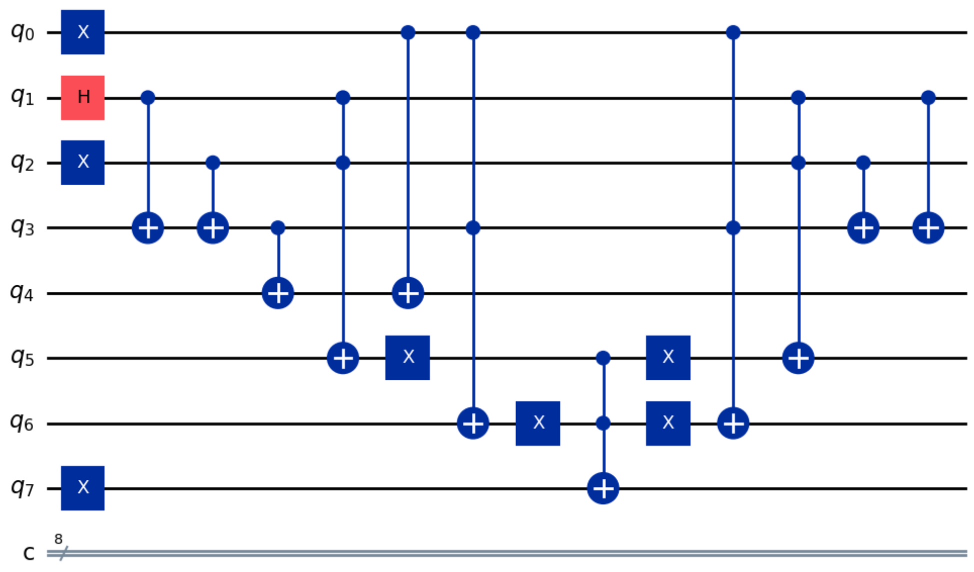


Figure 38: Full Adder circuit with a in superposition, $c_{in} = 1$, $b = 1$

Statevector Amplitudes

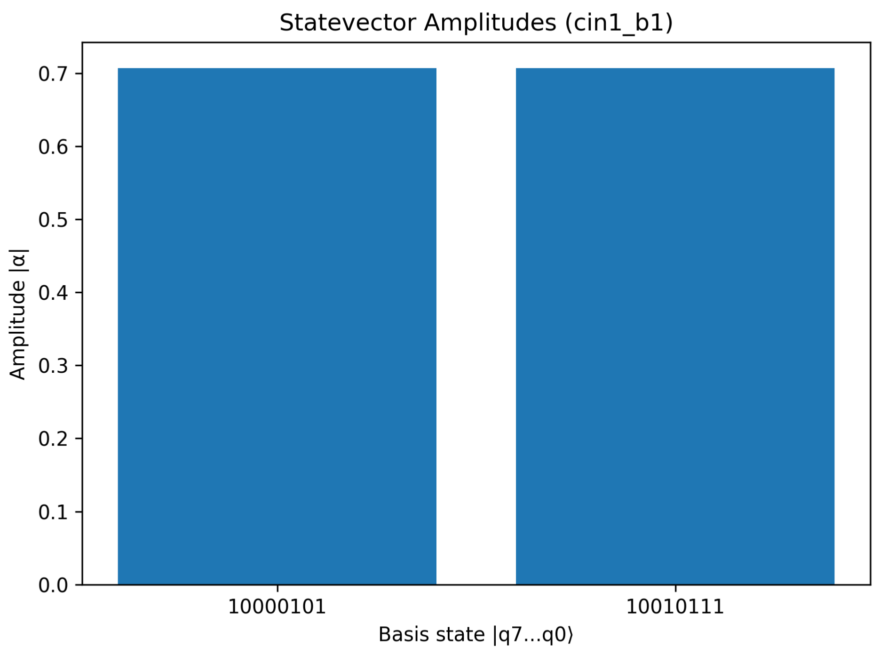


Figure 39: Statevector amplitudes for $c_{in} = 1$, $b = 1$

Q-Sphere Representation

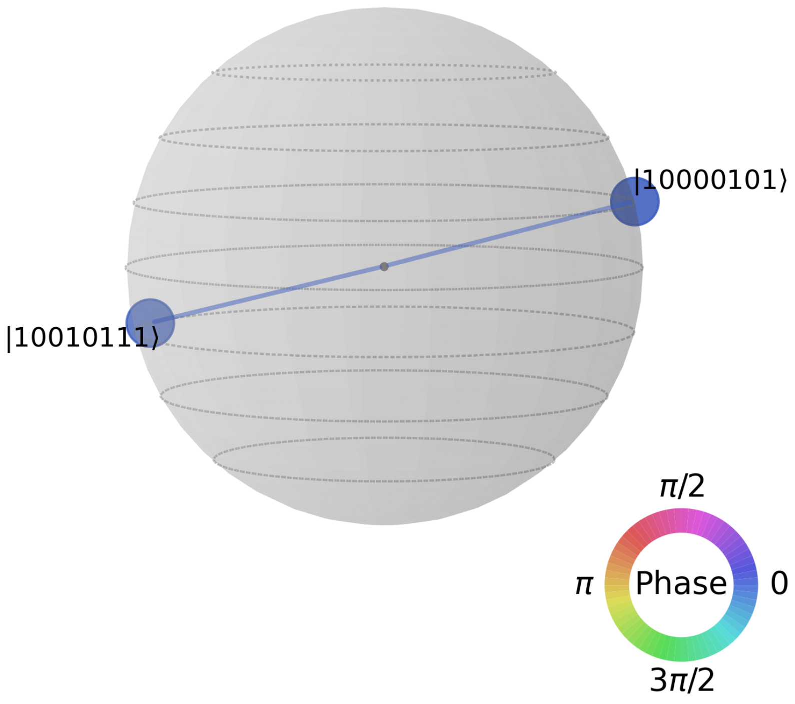


Figure 40: Q-sphere representation for $c_{in} = 1$, $b = 1$

Probabilities

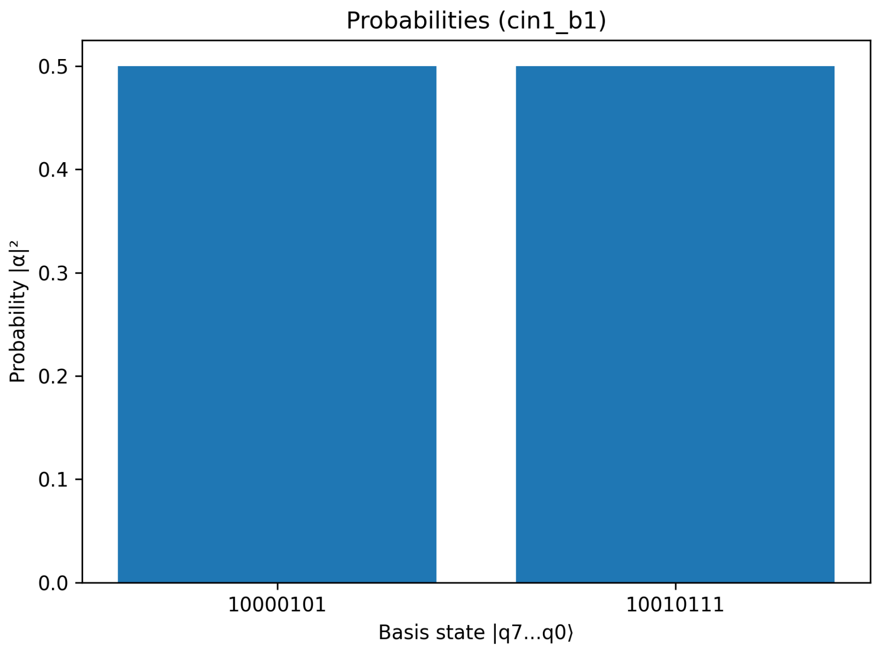


Figure 41: Measurement probabilities for $c_{in} = 1$, $b = 1$

Detailed Interpretation (Example: $c_{in} = 1$, $b = 1$)

Classically, the Full Adder computes:

$$S = a \oplus b \oplus c_{in}$$

$$C_{out} = ab + c_{in}(a \oplus b)$$

Since a is in superposition:

$$|a\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}$$

the Full Adder evaluates both possibilities:

Branch 1: $a = 0$

$$S = 0 \oplus 1 \oplus 1 = 0$$

$$C_{out} = 1$$

Branch 2: $a = 1$

$$S = 1 \oplus 1 \oplus 1 = 1$$

$$C_{out} = 1$$

Therefore, the final quantum state becomes:

$$|\psi\rangle = \frac{1}{\sqrt{2}}\Big(|\text{state corresponding to } a = 0\rangle + |\text{state corresponding to } a = 1\rangle\Big)$$

$$|\psi\rangle = \frac{1}{\sqrt{2}}\Big(|10000101\rangle + |10010111\rangle\Big)$$

Statevector Interpretation:

The Statevector figure shows exactly two non-zero amplitudes, each equal to approximately:

$$\frac{1}{\sqrt{2}}$$

This confirms that the system is in a superposition of two classical Full Adder outputs.

Probabilities Interpretation:

Since measurement probabilities are:

$$P(i) = |\alpha_i|^2$$

each branch has probability:

$$\left(\frac{1}{\sqrt{2}}\right)^2 = \frac{1}{2}$$

Therefore, the probability histogram displays two bars, each with probability approximately 0.5.

Placing a in superposition causes the Full Adder to compute both possible classical additions simultaneously. The output is a coherent superposition of two classical results, demonstrating the linearity of quantum circuits.

Task 5: Run FA on Quantum Hardware

In this task, the Full Adder circuit was executed on **real IBM quantum hardware** (not a simulator). To demonstrate a non-classical input, the qubit representing a was prepared in a **superposition** state by applying a Hadamard gate H before the Full Adder computation.

For this experiment, the classical inputs were fixed to:

$$c_{in} = 1, \quad b = 1$$

Thus, only the input qubit a was placed in superposition, while c_{in} and b remained classical.

$$|a\rangle = H|0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}$$

This means the circuit effectively evaluates the Full Adder simultaneously for both cases $a = 0$ and $a = 1$, with $c_{in} = 1$ and $b = 1$, and the measurement results reflect the probabilistic mixture of these two computational branches.

The job was submitted through the IBM Quantum Platform using the backend:

```
Using backend: ibm_torino

Job ID: d6hhfme48nic73an29m0
```

Circuit screenshot (IBM Quantum Platform):

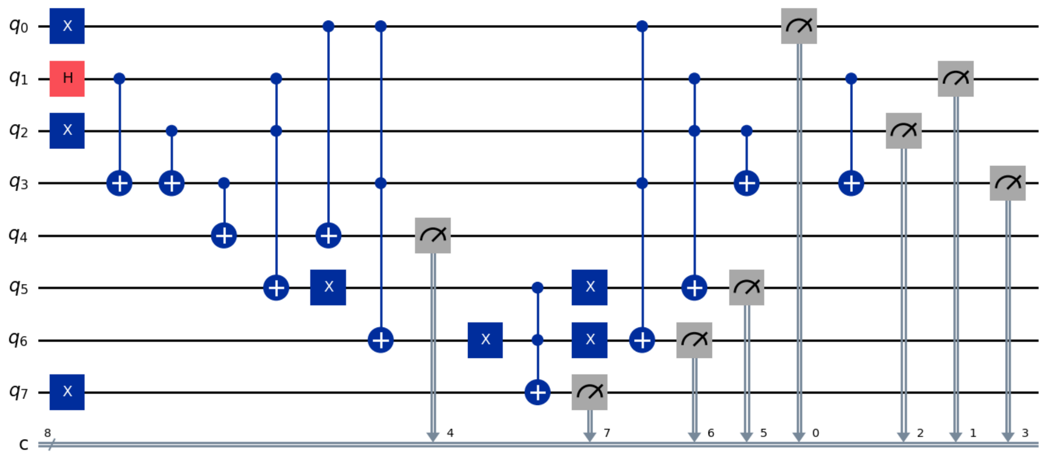
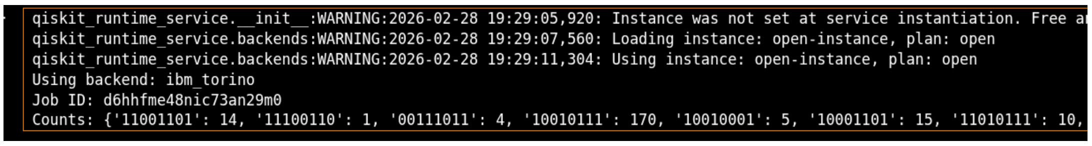


Figure 42: Full Adder circuit with a prepared in superposition (H gate) executed on IBM quantum hardware (`ibm_torino`).

Hardware job output screenshot:



```
qiskit_runtime_service.__init__:WARNING:2026-02-28 19:29:05,920: Instance was not set at service instantiation. Free ar
qiskit_runtime_service.backends:WARNING:2026-02-28 19:29:07,560: Loading instance: open-instance, plan: open
qiskit_runtime_service.backends:WARNING:2026-02-28 19:29:11,304: Using instance: open-instance, plan: open
Using backend: ibm_torino
Job ID: d6hhfme48nic73an29m0
Counts: {'11001101': 14, '11100110': 1, '00111011': 4, '10010111': 170, '10010001': 5, '10001101': 15, '11010111': 10,
```

Figure 43: IBM Quantum hardware execution output on `ibm_torino` (Job ID: `d6hhfme48nic73an29m0`).

Output visualization (counts histogram):

After execution, the device returned a dictionary of *counts*, where each key is a measured bitstring (computational basis state) and each value is the number of times that bitstring was observed across all shots. Since this circuit is relatively deep (multi-qubit gates such as Toffoli), the hardware results are affected by noise and readout errors, causing the output distribution to spread across multiple states rather than concentrating entirely on the ideal outcomes.

Top 15 outcomes:

For clarity, the 15 most frequent measured states are plotted below.

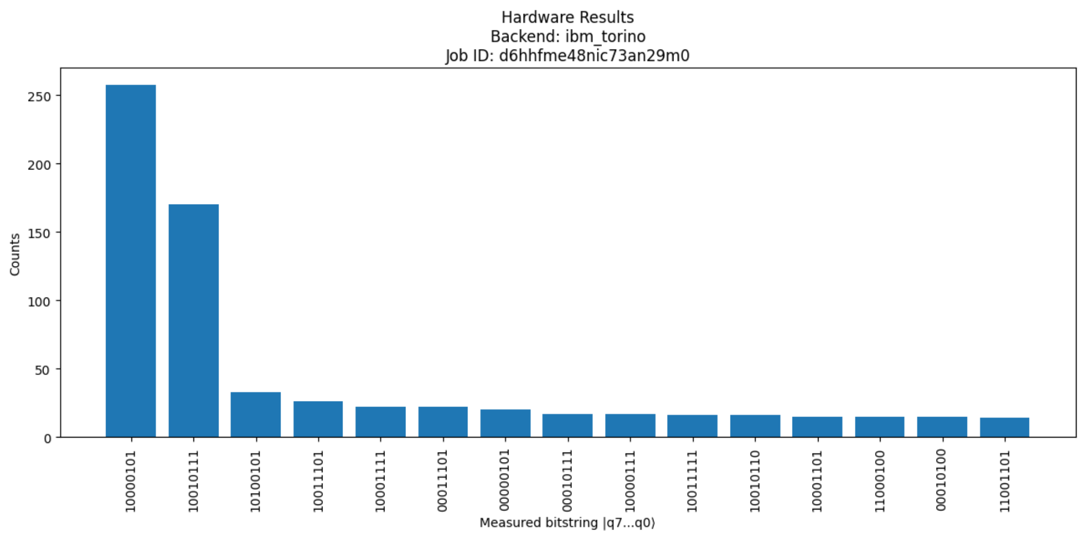


Figure 44: Top 15 most frequent measurement outcomes from the hardware execution.

Task 6: 3-bit Ripple-Carry Adder

A 3-bit Ripple-Carry Adder (RCA) was implemented in IBM Qiskit by cascading three reversible full-adder blocks. Each full-adder stage computes one sum bit and produces a carry-out, which is then used as the carry-in for the next stage. In this way, the carry propagates (ripples) through the three stages:

$$c_0 \to c_1 \to c_2 \to c_3$$

where each stage updates the carry and writes the corresponding sum bit into the B register. The circuit is fully **reversible** and the intermediate carry computations are **uncomputed**, which returns temporary ancilla qubits back to $|0\rangle$ and leaves only the desired output in the designated wires.

Circuit diagram:

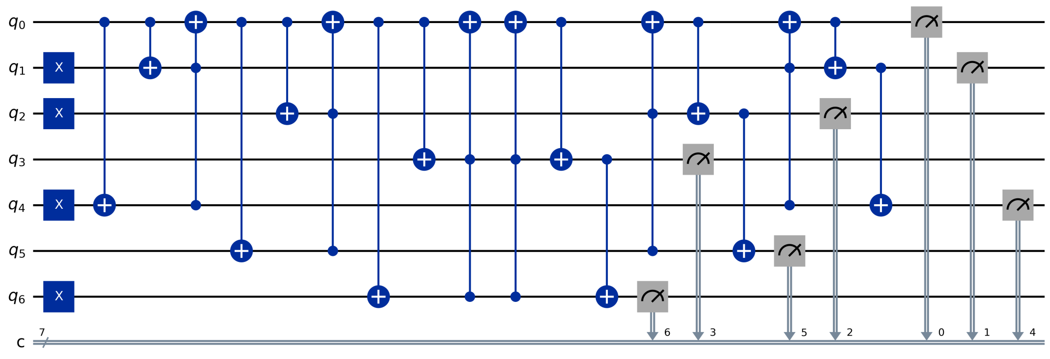


Figure 45: 3-bit reversible ripple-carry adder implemented in Qiskit (7 qubits).

Simulator result (ideal, noiseless):
The circuit was first executed on the Qiskit simulator. Since the inputs are classical and the circuit is deterministic, the simulator returns a single output state with probability 1. For the tested input, the simulator produced:

$$\texttt{`0000110'} : 1024$$

meaning the same bitstring was measured in all 1024 shots.
Hardware execution (IBM Quantum Lab):
The same circuit was then executed on real IBM quantum hardware. Unlike the simulator, the hardware output is affected by noise. As a result, the measurement distribution spreads across multiple bitstrings rather than concentrating entirely on a single outcome.
Hardware output visualization (Top 15 outcomes):

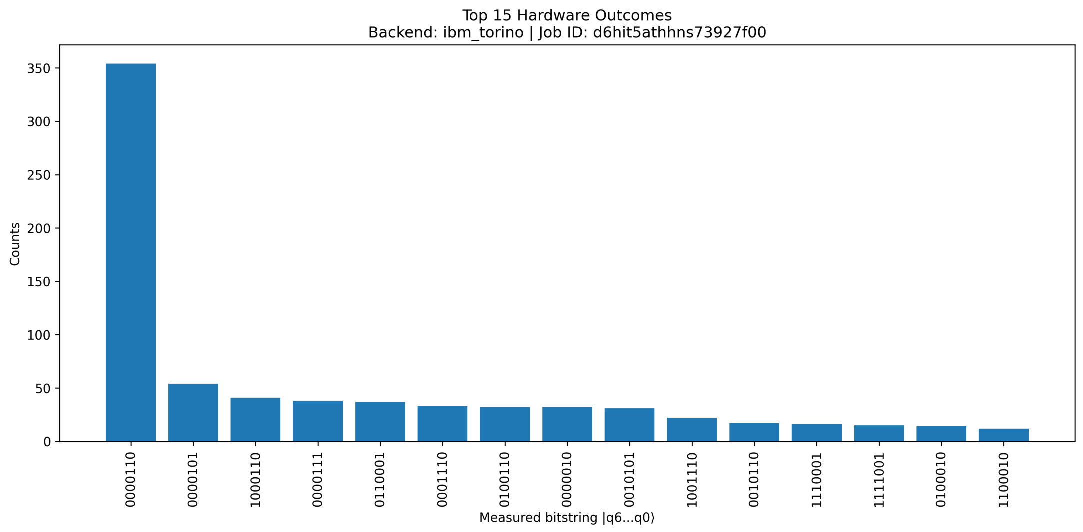


Figure 46: Top 15 measurement outcomes from running the 3-bit ripple-carry adder on real quantum hardware. The dominant outcome corresponds to the ideal result, while the remaining outcomes arise from hardware noise.

Task 7: ID-based Addition Using the 3-bit Adder

The student ID is 202515470. According to the task instructions, the two 3-bit numbers A and B are computed as follows:

$$A = (\text{sum of the rightmost two digits}) \bmod 8$$

$$B = (\text{sum of the two digits preceding them}) \bmod 8$$

The rightmost two digits are 7 and 0, therefore:

$$A = (7 + 0) \bmod 8 = 7$$

The two digits preceding them are 4 and 5, therefore:

$$B = (4 + 5) \bmod 8 = 9 \bmod 8 = 1$$

In 3-bit binary representation:

$$A = 7_{10} = 111_2, \qquad B = 1_{10} = 001_2$$

The ripple-carry adder was then executed with inputs $A = 111_2$ and $B = 001_2$ (with $c_{in} = 0$). The expected classical result is:

$$111_2 + 001_2 = 1000_2$$

which corresponds to sum bits 000_2 with a carry-out of 1.

In the implemented reversible 3-bit adder, the sum is written into the B register, while the carry/ancilla wire is uncomputed back to its original value to preserve reversibility. Therefore, the simulator output is deterministic and returns a single measured bitstring with all shots:

`'0001110'` : 1024

This bitstring is reported in the order $(q_6\, q_5\, q_4\, q_3\, q_2\, q_1\, q_0)$, where $q_1 q_2 q_3$ store the input A, $q_4 q_5 q_6$ store the computed sum, and q_0 is the carry/ancilla wire. The result confirms that the sum stored in the B register is 000_2, which matches the expected addition result.

4 Conclusion

In this lab, classical adders were implemented using reversible quantum circuits in IBM Qiskit. The Half Adder, Full Adder, and 3-bit Ripple-Carry Adder were constructed using CNOT and Toffoli gates while preserving reversibility through uncomputation of ancilla qubits.

Statevector and probability visualizations confirmed correct deterministic behavior for classical inputs, and superposition inputs demonstrated quantum parallelism through multiple output amplitudes.

Running the circuits on real IBM quantum hardware revealed noise effects compared to the ideal simulator results.

Overall, this lab demonstrated the connection between classical reversible logic and quantum computation, as well as the practical challenges of executing arithmetic circuits on real quantum hardware.

Appendix

The complete code, Jupyter notebook, and supplementary material are available in the following public GitHub repository:

`https://github.com/ferasalsayigh/`
`COE530Assignment1HalfAdderFullAdder-and-3-bit-Ripple-Carry-Adder-Qiskit-`

The repository includes:

- **COE530_Assignment1_FerasAlsayigh_g202515470.ipynb** — Full implementation in Jupyter notebook.
- **assignment.pdf** — Original assignment description.
- **documentation.pdf** — Full written report with figures and explanations.

For

References