

IMPLEMENTATION AND SIMULATION OF A QUANTUM HALF ADDER USING IBM QISKIT

Krishnaveni Guduru¹, Devi Prasanna Vempa²,
Manideepa Mekala³, Anusha Pondugula⁴

¹Assistant Professor, Department of ECE, Bapatla Women's Engineering College, Andhra Pradesh, India.

^{2,3,4}B.Tech Scholar, Department of ECE, Bapatla Women's Engineering College, Andhra Pradesh, India.

Correspondence should be addressed to veni.guduru29@gmail.com

Abstract

Quantum computing is an emerging computational paradigm that represents and processes information using quantum states. This paper presents the implementation and simulation of a Quantum Half Adder using the IBM Qiskit framework. The proposed circuit performs one-bit binary addition and produces two outputs, SUM and CARRY, through reversible quantum operations. Classical input values are encoded into quantum states using Pauli-X gates. Controlled-NOT (CNOT) gates are used to realize the XOR relationship required for SUM generation, while a Toffoli (CCNOT) gate realizes the AND relationship required for CARRY generation. The complete design is implemented in Python and executed using the Qiskit Aer Simulator. Four possible input combinations are evaluated and the measured outputs are compared with the expected Half Adder truth table. Circuit visualization, terminal output, and measurement histograms are used to verify the operation. The results show that the simulated circuit produces the expected SUM and CARRY values for all input combinations. The work provides a simple practical model for understanding reversible quantum computation, quantum gates, and Qiskit-based quantum circuit simulation.

Index Terms— Quantum computing, Quantum Half Adder, IBM Qiskit, quantum gates, CNOT, Toffoli, reversible computation, Qiskit Aer.

I. INTRODUCTION

The rapid growth of digital technology has created a continuous demand for faster and more capable computing systems. Classical computers have successfully supported applications ranging from communication and banking to scientific simulation, artificial intelligence, healthcare, and industrial automation. In a conventional digital system, information is represented by binary bits. Each bit is associated with one of two logical states, 0 or 1, and Boolean logic gates are used to process these states. This approach is highly reliable and remains the foundation of present-day computing, but it also motivates research into alternative models when computational requirements become extremely demanding.

Quantum computing provides a fundamentally different way of representing and manipulating information. Its basic unit is the qubit, which is described using quantum states rather than only classical binary values. A qubit can be prepared in either of the computational basis states $|0\rangle$ and $|1\rangle$, or in a superposition of them. Quantum gates operate on these states and form the elementary operations of a quantum circuit. Measurement is used at the end of a computation to obtain classical information from the quantum state. These concepts make quantum circuit design different from conventional digital circuit design [1], [2], [7].

An important property of quantum computation is reversibility. Quantum operations are represented by reversible transformations, which means that the evolution of the state does not simply discard information [3], [4]. Consequently, arithmetic functions that are normally implemented using irreversible Boolean gates must be mapped to suitable reversible quantum operations [5], [6]. Studying small arithmetic circuits is therefore useful because it connects

familiar digital logic with the principles of quantum information processing.

A Half Adder is one of the simplest arithmetic circuits. It accepts two one-bit inputs, A and B, and produces SUM and CARRY outputs. The SUM output is obtained from the XOR operation and the CARRY output is obtained from the AND operation. The conventional equations are $SUM = A \oplus B$ and $CARRY = A \wedge B$. In a quantum implementation, these functions can be constructed from reversible gates [6], [7]. This makes the Quantum Half Adder an appropriate starting point for understanding quantum arithmetic.

In the present work, a Quantum Half Adder is implemented using IBM Qiskit. The design uses Pauli-X gates for encoding the binary input values, CNOT gates for SUM generation, and a Toffoli gate for CARRY generation. The circuit contains four qubits and four classical bits. The implementation is executed using the Qiskit Aer Simulator, and the outputs are verified for all four possible input combinations. The use of Qiskit provides a practical software environment for constructing and simulating quantum circuits [11], [12]. The study focuses on practical circuit construction, simulation, and verification rather than on physical quantum-hardware implementation.

The major objective is to demonstrate a complete workflow from classical input to quantum-state preparation, gate-level arithmetic, measurement, and output verification. The approach also provides a simple basis for extending the design to larger reversible arithmetic circuits such as Quantum Full Adders and Ripple Carry Adders.

The remainder of this paper is organized as follows. Section II presents the motivation and objectives of the proposed Quantum Half Adder. Section III reviews the relevant literature and identifies the research gap. Section IV provides the background of quantum computing and the basic concepts required for the proposed design. Section V describes the proposed Quantum Half Adder, including input encoding, SUM and CARRY generation, measurement mapping, and expected functional behavior. Section VI presents the software environment, methodology, implementation algorithm, and circuit interpretation. Section VII presents the experimental results and discussion, followed by the verification summary, advantages, limitations, comparison with a classical Half Adder, practical significance, future scope, and conclusion.

II. MOTIVATION AND OBJECTIVE

The motivation for selecting a Quantum Half Adder is its importance as a basic arithmetic building block. Larger arithmetic systems are formed by combining elementary operations, and an understanding of a one-bit quantum adder helps in studying more complex quantum arithmetic architectures. The project also provides practical exposure to qubits, reversible gates, circuit measurement, and quantum simulation using a widely used software framework.

The specific objectives are to understand the working principles of a Quantum Half Adder; encode binary inputs into quantum states; generate SUM and CARRY using reversible gates; implement the

complete circuit in Python using IBM Qiskit; execute the circuit with Qiskit Aer; visualize the circuit and measurement results; and verify the outputs against the standard Half Adder truth table.

III. LITERATURE REVIEW AND RESEARCH GAP

A. Literature Review

Research in quantum computing developed from the idea that computation is a physical process and that quantum systems can be used to represent information. Feynman discussed the difficulty of efficiently simulating quantum systems using conventional computers and proposed quantum systems as a natural platform for such simulation. Deutsch later introduced the concept of a universal quantum computer and described computation using quantum-mechanical operations. These early works established the conceptual basis for treating quantum systems as computational devices [1], [2].

Bennett investigated logical reversibility and showed that computation can be formulated so that information is preserved. Reversible computation is particularly relevant to quantum circuits because quantum evolution is reversible. Benioff also developed a quantum-mechanical model of computation, contributing to the theoretical connection between physical quantum systems and computational processes. These studies provide the conceptual background for using reversible gates in quantum arithmetic [3], [4].

Toffoli introduced a reversible three-input logic gate, now commonly called the Toffoli or CCNOT gate. The gate has two controls and one target and changes the target only when both controls are active. This property makes it particularly useful for implementing conditional arithmetic operations such as carry generation. Barenco et al. investigated elementary quantum gates and universal gate constructions, showing how complex operations can be constructed from elementary reversible quantum gates [5], [6].

Nielsen and Chuang provided a comprehensive treatment of quantum computation and quantum information, including qubits, quantum states, measurement, quantum gates, circuits, and algorithms. Their work is widely used as a theoretical reference for understanding the mathematical and physical principles underlying quantum circuits [10]. Shor's algorithm and Grover's search algorithm later demonstrated important computational applications of quantum principles [7], [8], [9].

The development of quantum software frameworks has made these concepts more accessible for practical implementation. IBM Qiskit provides a Python-based environment for constructing quantum circuits, applying gates, measuring qubits, and visualizing circuits. Qiskit Aer enables users to simulate quantum circuits on classical computers. These capabilities make it possible to verify a circuit before attempting execution on a physical quantum processor.

Existing literature therefore provides strong theoretical foundations for quantum computation, reversible logic, and quantum gates, while software documentation provides practical mechanisms for circuit implementation. However, for beginners and small-scale experimental studies, there is value in combining these concepts into a single arithmetic example. The present work uses a Quantum Half Adder to demonstrate how input encoding, CNOT-based XOR, Toffoli-based AND, measurement, and histogram-based verification can be connected in one implementation.

B. Research Gap

Many foundational studies concentrate on quantum theory, universal computation, algorithms, or individual reversible gates. General-purpose software frameworks provide the tools needed for circuit implementation but do not by themselves demonstrate the complete reasoning behind a small arithmetic circuit. A practical gap therefore exists between learning individual quantum gates and

understanding how those gates cooperate to perform a familiar arithmetic operation.

This work addresses that gap by implementing a complete one-bit Quantum Half Adder and explicitly mapping each functional requirement to a quantum operation. The design identifies the input qubits, output qubits, encoding operation, SUM operation, CARRY operation, and measurement stage. The circuit is then evaluated for every possible binary input combination, making the verification process transparent and reproducible.

C. Contribution of the Present Work

The main contribution is an implementation-oriented quantum arithmetic model using IBM Qiskit. The work does not propose a new quantum gate; instead, it demonstrates a clear combination of standard gates for a useful arithmetic function. The implementation is intentionally compact so that each gate can be traced to a specific operation. The simulation output is compared with the classical truth table, providing a direct functional check.

The project also demonstrates how a classical simulator can be used as a development and verification tool. This is useful when physical quantum hardware is unavailable or when the primary goal is to understand circuit behavior before introducing hardware noise and connectivity constraints.

IV. BACKGROUND OF QUANTUM COMPUTING

A. Qubit and Quantum States

A qubit is represented by a quantum state that can be expressed using the basis states $|0\rangle$ and $|1\rangle$. Superposition allows a qubit to be represented as a weighted combination of these basis states. When measurement is performed, a classical outcome is obtained according to the probability associated with the state. In the present Half Adder implementation, the inputs are deliberately prepared in computational basis states so that the circuit performs deterministic binary logic while still using quantum gates.

B. Quantum Gates

Quantum gates are represented as operations acting on one or more qubits. The Pauli-X gate is a single-qubit operation, while the CNOT gate and Toffoli gate are multi-qubit controlled operations. Controlled gates are especially useful in arithmetic because the output can be changed conditionally according to input states. By connecting these gates in the correct order, a quantum circuit can reproduce the functional behavior of familiar arithmetic logic.

C. Measurement

Measurement is the interface between quantum computation and conventional digital interpretation. Before measurement, the circuit contains quantum states; after measurement, the result is recorded as classical bits. In the proposed system, measurement is performed only after all arithmetic gates have completed their operations. The resulting classical information is then interpreted using the known mapping of qubits and classical bits in the program.

V. PROPOSED QUANTUM HALF ADDER

A. Overall Architecture

The proposed Quantum Half Adder is organized into four stages: input acquisition, quantum-state encoding, reversible arithmetic processing, and measurement/output. Two one-bit values A and B are supplied as inputs. The second stage prepares the input qubits according to these values. The processing stage applies the quantum gates required to produce SUM and CARRY. The final stage measures the output qubits and presents the results in classical form. Four qubits are used in the implementation. q0 represents input A and q1 represents input B. q2 is reserved for SUM and q3 is reserved for CARRY. Four classical bits are used to store the measurement results.

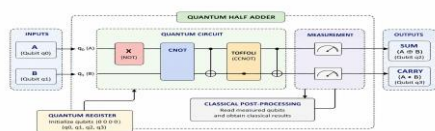


Fig. 1. Proposed architecture showing input encoding, quantum arithmetic, measurement, and output verification.

B. Input Encoding Using Pauli-X

At circuit initialization, the qubits are in the $|0\rangle$ state. The binary inputs are encoded conditionally. If $A=1$, the X gate is applied to q_0 ; if $A=0$, q_0 remains unchanged. Similarly, if $B=1$, the X gate is applied to q_1 . This produces the computational-basis representation of the selected input pair.

The encoding stage is important because a quantum circuit does not directly accept a conventional integer or Boolean value as a qubit state. The program must prepare the desired quantum state explicitly. Using the Pauli-X gate for this purpose provides a simple connection between classical binary input and quantum circuit representation.

C. CNOT Gate and SUM Generation

The SUM output of a Half Adder is the XOR of A and B. The CNOT gate is suitable for this operation because it conditionally flips its target according to the state of its control. By applying CNOT operations to the SUM qubit using the two input qubits, the target records the parity of A and B. Equal inputs result in $SUM=0$, while unequal inputs result in $SUM=1$.

For input 00, neither control activates the SUM target and the output remains 0. For input 01, one control is active and the SUM becomes 1. For input 10, the other control is active and the SUM also becomes 1. For input 11, both controlled operations act on the SUM target, causing two inversions and returning the target to 0.

D. Toffoli Gate and CARRY Generation

The CARRY output is the AND of the two input bits. A Toffoli gate is a three-qubit controlled operation with two controls and one target. In the proposed circuit, q_0 and q_1 act as the control qubits and q_3 acts as the target. The target is inverted only when q_0 and q_1 are both in $|1\rangle$. Consequently, only the input combination 11 produces $CARRY=1$.

The Toffoli gate is especially appropriate because it preserves reversibility while providing the exact conditional behavior required for the carry function. It demonstrates how a familiar Boolean AND relationship can be realized using a reversible quantum operation rather than a conventional irreversible AND gate.

E. Measurement Mapping

After the arithmetic operations, the SUM and CARRY qubits are measured into the classical register. The program interprets the measured positions according to the selected classical-bit mapping. This mapping must be kept consistent when reading terminal output or plotting a histogram because quantum software may display classical bit strings in register order rather than in the same left-to-right order used in the circuit diagram.

F. Expected Functional Behavior

When both inputs are zero, no output is generated. When exactly one input is one, SUM becomes one and CARRY remains zero. When both inputs are one, SUM becomes zero and CARRY becomes one. These four cases completely describe the behavior of a one-bit Half Adder and form the basis for experimental verification.

Table I. Truth table of the proposed Quantum Half Adder.

A	B	SUM	CARRY
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

VI. SOFTWARE AND IMPLEMENTATION ENVIRONMENT

Python is used as the programming language because Qiskit provides a Python programming interface for quantum circuit development. The Qiskit library supplies the QuantumCircuit functionality required to create qubits, classical bits, and gate operations. Qiskit Aer provides the simulator used for executing the circuit on a classical computer. Matplotlib can be used to display measurement histograms. A Python development environment such as Visual Studio Code can be used to write and execute the program. The software workflow consists of importing the required libraries, creating the quantum circuit, accepting or defining binary inputs, applying conditional Pauli-X gates, performing CNOT and Toffoli operations, measuring the output qubits, executing the circuit through Aer, and displaying the results. This workflow makes each stage easy to inspect during debugging and verification.

A. Methodology and Implementation

1. Input Acquisition and Validation

The program begins by accepting two binary input values A and B. Since the proposed circuit represents one-bit addition, each value must be either 0 or 1. The accepted inputs are used to determine which input qubits require state preparation. This stage provides the interface between conventional user input and the quantum circuit [11].

2. Circuit Initialization

A QuantumCircuit is created with four qubits and four classical bits. The qubits are initially in the computational basis state $|0000\rangle$. Two qubits are assigned to the inputs, while two additional qubits are reserved for SUM and CARRY. Classical bits are associated with the measurement stage so that the final quantum state can be represented as a classical bit string [11].

3. State Encoding

For each input whose value is 1, a Pauli-X gate is applied to the associated input qubit. An input of 0 leaves the corresponding qubit unchanged. This conditional encoding creates the desired basis state for the input pair.

4. SUM Operation

The SUM output is generated through the CNOT network. The input qubits act as controls and the SUM qubit acts as the target. The sequence produces the XOR relationship without using an irreversible classical XOR gate. As a result, the SUM qubit represents the required binary addition parity.

5. CARRY Operation

The CARRY output is generated with a Toffoli operation. The two input qubits serve as controls and the CARRY qubit serves as the target. The target is flipped only when both inputs are 1. This directly realizes the AND condition required for binary carry generation.

6. Measurement

After all gate operations are completed, measurement instructions are added. The SUM and CARRY qubits are mapped to classical bits. Once the circuit is measured, the quantum information is converted into classical values. The program can then print the resulting state and use the counts to generate a histogram.

7. Qiskit Aer Simulation

The Aer simulator executes the circuit repeatedly according to the selected number of shots. In this project, 1024 shots are used. The counts dictionary returned by the simulator indicates how many times each classical output state was observed. For the ideal deterministic circuit, the expected logical state should appear overwhelmingly or exclusively, depending on the exact circuit and simulator configuration.

8. Result Display and Verification

The final stage displays the quantum circuit, measurement histogram, and decoded output values. The program compares the

obtained SUM and CARRY values with the expected Half Adder truth table. A PASS result is reported when the measured logical outputs agree with the expected outputs. Repeating this procedure for all four combinations provides complete functional verification.

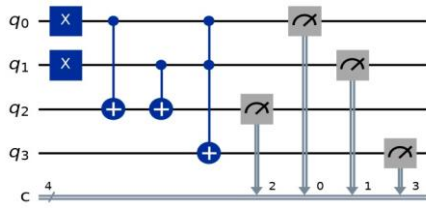


Fig. 2. Quantum Half Adder circuit generated using IBM Qiskit.

B. Implementation Algorithm

The implementation algorithm starts by importing the required Qiskit and simulator libraries. The program then reads the two binary input values A and B. Four qubits and four classical bits are created and initialized. If A is 1, a Pauli-X gate is applied to the A input qubit; if B is 1, a Pauli-X gate is applied to the B input qubit. The CNOT operations are then applied to generate SUM, followed by the Toffoli operation for CARRY. The output qubits are measured into the classical register. The circuit is executed using Qiskit Aer with 1024 shots. Measurement counts are collected, the circuit and histogram are displayed, and the decoded SUM and CARRY values are compared with the expected truth-table values.

C. Circuit Interpretation

The circuit diagram provides a visual representation of the sequence of quantum operations. The horizontal lines represent qubits, while the symbols placed on the lines indicate the gates acting on them. X symbols indicate Pauli-X input encoding. Controlled operations contain control markers and target symbols. The measurement symbols at the end transfer the selected qubit states into the classical register. The diagram therefore provides a compact view of the complete computation.

When explaining the circuit, each operation can be related directly to the arithmetic requirement. The X gates prepare the input states, the CNOT sequence performs the XOR relationship for SUM, the Toffoli gate performs the AND relationship for CARRY, and measurement converts the final quantum states into classical outputs. This gate-by-gate interpretation makes the design easier to verify and explain.

VII. EXPERIMENTAL RESULTS AND DISCUSSION

A. Verification Procedure

Verification is performed exhaustively because there are only four possible input pairs. Each input pair is encoded, simulated, measured, and compared with the corresponding row of the truth table. This avoids relying on a partial test set. A correct implementation must satisfy both output functions for every input combination, not merely for one representative case.

The verification process also helps detect practical implementation errors. Incorrect qubit mapping may produce apparently reversed output strings, an incorrectly placed CNOT may change the SUM relationship, and an incorrectly connected Toffoli may produce an incorrect CARRY. Comparing decoded outputs rather than only visual circuit appearance provides a stronger functional check.

B. Experimental Results and Discussion

1. Experimental Setup

The proposed Quantum Half Adder was tested using the Python implementation and Qiskit Aer Simulator. The simulator was selected because it allows the quantum circuit to be executed on a classical computer while preserving the gate-level structure of the proposed design. A total of 1024 shots were used for each input combination. The same circuit architecture was maintained while the encoded input values were changed.

The experiment focuses on logical correctness rather than hardware performance. The primary evaluation criteria are the SUM and CARRY values obtained after measurement, agreement with the expected truth table, and consistency of the measurement histogram with the generated output state.

2. Case 1: A=0, B=0

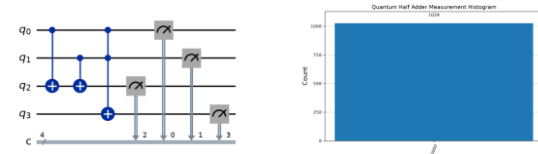


Fig. 3. Quantum circuit and measurement histogram for A = 0, B = 0.

For the 00 input, both input qubits remain in $|0\rangle$ because neither requires an X operation. Since both CNOT controls are zero, the SUM qubit remains zero. The Toffoli gate also does not activate because its two controls are not both one. Therefore, the expected result is SUM=0 and CARRY=0.

3. Case 2: A=0, B=1

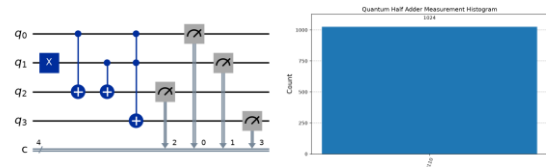


Fig. 4. Quantum circuit and measurement histogram for A = 0, B = 1.

For the 01 input, the B qubit is changed to $|1\rangle$ using the X gate. Because the inputs are different, the CNOT network produces SUM=1. The Toffoli condition is not satisfied because A=0, so CARRY remains 0. The expected output is therefore SUM=1 and CARRY=0.

4. Case 3: A=1, B=0

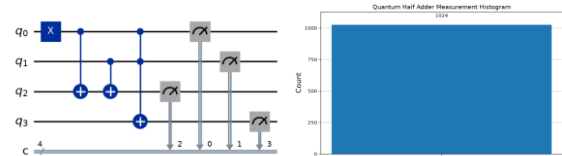


Fig. 5. Quantum circuit and measurement histogram for A = 1, B = 0.

For the 10 input, the A qubit is prepared in $|1\rangle$ while B remains in $|0\rangle$. Again, exactly one input is active, so the SUM qubit becomes 1 and the CARRY qubit remains 0. The expected output is SUM=1 and CARRY=0. This case also verifies the symmetry of the Half Adder with respect to its two inputs.

5. Case 4: A=1, B=1

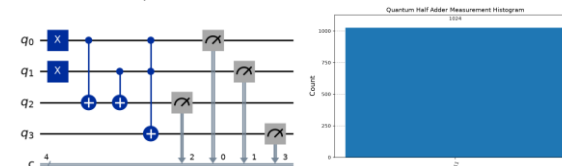


Fig. 6. Quantum circuit and measurement histogram for A = 1, B = 1.

For the 11 input, both input qubits are prepared in $|1\rangle$. The CNOT sequence applies two controlled inversions to the SUM target, resulting in SUM=0. Both Toffoli controls are active, so the CARRY target changes from 0 to 1. The final logical result is SUM=0 and CARRY=1. This case demonstrates simultaneous generation of the two Half Adder outputs.

6. Histogram Analysis

The measurement histogram shows the frequency of each measured output state over 1024 shots. Each bar represents a measured classical state and its height represents the number of observations. For the deterministic basis-state calculations used in this implementation, the expected output state should dominate the counts. The histogram therefore provides a visual confirmation of the logical result and complements the terminal output.

The histogram should not be interpreted as evidence that the Half Adder is performing probabilistic arithmetic. The input states and gate sequence in this project are chosen to produce deterministic

computational-basis outputs in an ideal simulator. The purpose of the histogram is to show how the simulator records repeated measurements and to make the verification result visually clear.

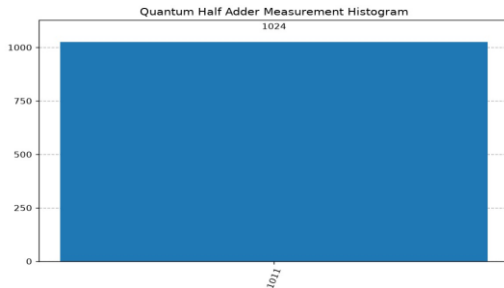


Fig. 7. Measurement histogram obtained from the Qiskit Aer Simulator.

7. Results Summary

Table II. Functional verification summary. The displayed measured bit-string order depends on the classical-register mapping; SUM and CARRY are interpreted using the program's defined output positions.

A	B	SUM	CARRY	Expected	Status
0	0	0	0	00	PASS
0	1	1	0	10	PASS
1	0	1	0	10	PASS
1	1	0	1	01	PASS

8. Discussion of Results

The four input cases demonstrate that the proposed circuit follows the expected Half Adder behavior. The 00 case produces no output, the 01 and 10 cases produce a sum without a carry, and the 11 case produces a carry without a sum. These cases collectively verify both logical functions implemented by the circuit.

The results show the complementary roles of the two controlled-gate structures. The CNOT network responds to whether the two input states are different, while the Toffoli gate responds only when both input states are one. Because the conditions are different, the circuit can generate SUM and CARRY independently on separate output qubits.

The use of Qiskit Aer provides an accessible way to validate the circuit before hardware execution. A simulator avoids the physical-device noise, calibration errors, limited connectivity, and queueing constraints associated with real quantum hardware. This is suitable for the present project because the main objective is to verify the logical operation of the gate-level design.

At the same time, ideal simulation should not be interpreted as a complete evaluation of real quantum hardware. Physical qubits can experience decoherence, gate errors, readout errors, and connectivity restrictions. Hardware-based testing would provide an additional level of validation beyond the ideal simulator results.

9. Advantages

The implementation uses a small number of qubits and simple gates, making the circuit easy to inspect. Each gate has a clearly identifiable role. Qiskit provides circuit visualization and simulation in a Python environment, and exhaustive verification is straightforward because only four input combinations exist. The design can also serve as a starting point for larger reversible arithmetic circuits.

10. Limitations

The implementation is a small simulation-oriented circuit and does not claim a computational speedup over a classical Half Adder. The experiment does not evaluate physical quantum-hardware noise, hardware execution time, or device-specific resource constraints. The results therefore demonstrate logical correctness under ideal simulation conditions rather than full hardware performance.

11. Comparison with Classical Half Adder

A classical Half Adder generally uses an XOR gate for SUM and an AND gate for CARRY. These are conventional Boolean operations and are normally implemented using irreversible logic. The quantum implementation instead uses reversible gates. CNOT operations provide the XOR relationship for SUM and the Toffoli

gate provides the AND relationship for CARRY while maintaining reversible quantum operations.

The purpose of the quantum implementation is not to claim that a one-bit quantum Half Adder is inherently faster than its classical counterpart. Rather, it demonstrates how conventional arithmetic functionality can be mapped into a quantum circuit. This mapping is important for understanding how larger reversible arithmetic units can be developed for quantum processors.

Another difference is the representation of information. Classical inputs are initially expressed as bits, but the proposed system explicitly prepares qubits and applies quantum gates. The final measurement converts the quantum state back into classical information. Thus, the implementation provides a complete example of the interface between classical control and quantum processing.

12. Practical Significance

The proposed implementation is useful for students and researchers beginning work in quantum circuit design because it combines several essential concepts in a single experiment. Input encoding introduces state preparation, CNOT operations demonstrate controlled quantum logic, the Toffoli gate introduces multi-control computation, and measurement demonstrates the conversion of quantum information into classical output. The use of Qiskit Aer further shows how quantum circuits can be tested without access to physical hardware.

From a design perspective, the circuit illustrates the importance of separating functional requirements from gate implementation. The Half Adder requirements are first expressed as SUM and CARRY functions. Each function is then mapped to an appropriate reversible gate sequence. This design approach can be reused when developing more complex quantum arithmetic circuits.

The verification procedure is similarly reusable. A designer can define an expected truth table, execute each input combination, obtain measurement counts, decode the output, and compare the measured result with the expected value. This provides a systematic method for identifying gate-order errors, incorrect qubit mapping, and measurement-order mistakes.

13. Future Scope

The most direct extension of the proposed design is a Quantum Full Adder. A Full Adder includes a third input representing carry from a previous stage. By combining reversible arithmetic blocks, larger binary additions can be constructed. Cascading such circuits can lead to Quantum Ripple Carry Adders and other arithmetic architectures.

Another direction is circuit optimization. Quantum hardware resources are limited, so reducing gate count, circuit depth, and auxiliary qubits can improve practical implementability. Equivalent gate decompositions can be investigated to identify circuits that use fewer elementary operations while maintaining the same functional behavior.

Noise-aware simulation is another useful extension. Real quantum processors are affected by relaxation, dephasing, imperfect gates, readout errors, and connectivity constraints. Introducing realistic noise models into the simulation would allow the robustness of the Half Adder to be studied. The final stage of this extension would be execution on real IBM Quantum hardware and comparison between ideal simulation, noisy simulation, and physical-device results.

The design may also be extended beyond simple adders. Quantum arithmetic units, comparators, accumulators, and arithmetic logic units can be constructed from combinations of reversible operations. The present Half Adder can therefore be considered a basic experimental step toward more complex quantum information-processing systems.

Conclusion

This work presented the implementation and simulation of a Quantum Half Adder using the IBM Qiskit framework. The proposed design accepts two one-bit binary inputs and produces

SUM and CARRY outputs using reversible quantum operations. Pauli-X gates provide input-state encoding, CNOT gates implement the XOR-based SUM function, and a Toffoli gate implements the AND-based CARRY function. The circuit was developed in Python and executed using the Qiskit Aer Simulator with 1024 measurement shots.

All four possible input combinations were considered during functional verification. The expected behavior was obtained for 00, 01, 10, and 11, corresponding to SUM/CARRY outputs of 00, 10, 10, and 01 respectively. Circuit visualization and measurement histograms provide additional evidence of the simulation process. The results confirm that the proposed gate sequence correctly reproduces the functionality of a one-bit Half Adder under ideal simulation conditions.

The work demonstrates that a familiar digital arithmetic function can be expressed through reversible quantum operations and implemented using an accessible software framework. Although the circuit is small and does not itself provide a practical speed advantage over a classical Half Adder, it serves as a useful foundation for studying quantum arithmetic. The same design principles can be extended to Full Adders, larger arithmetic circuits, optimized reversible networks, and physical quantum hardware.

REFERENCES

- [1] R. P. Feynman, "Simulating physics with computers," *International Journal of Theoretical Physics*, vol. 21, pp. 467–488, 1982.
- [2] D. Deutsch, "Quantum theory, the Church–Turing principle and the universal quantum computer," *Proceedings of the Royal Society A*, vol. 400, pp. 97–117, 1985.
- [3] C. H. Bennett, "Logical reversibility of computation," *IBM Journal of Research and Development*, vol. 17, no. 6, pp. 525–532, 1973.
- [4] P. Benioff, "The computer as a physical system," *Journal of Statistical Physics*, vol. 22, pp. 563–591, 1980.
- [5] T. Toffoli, "Reversible computing," *MIT Laboratory for Computer Science, Tech. Rep. MIT/LCS/TM-151*, 1980.
- [6] A. Barenco et al., "Elementary gates for quantum computation," *Physical Review A*, vol. 52, no. 5, pp. 3457–3467, 1995.
- [7] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge, U.K.: Cambridge Univ. Press, 2010.
- [8] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proc. 35th Annual Symposium on Foundations of Computer Science*, 1994, pp. 124–134.
- [9] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proc. 28th Annual ACM Symposium on Theory of Computing*, 1996, pp. 212–219.
- [10] J. Preskill, "Quantum computing in the NISQ era and beyond," *Quantum*, vol. 2, p. 79, 2018.
- [11] IBM Quantum, "Qiskit documentation," IBM Quantum, accessed Aug. 2026.
- [12] Qiskit Aer, "Qiskit Aer documentation," IBM Quantum, accessed Aug. 2026.