Project Title

Enhanced Data Transmission Efficiency

Student Details

Dandamudi.Geeta Priya

2348512

5 MSAIM, Dept. of Computer Science

Guide Details

Dr.Somnath Sinha

Dept.of Computer Science

Institution Details:

Christ(Deemed to be University)

Date of Submission

05-12-24

Table of Contents

1. Abstract

This project demonstrates the use of Superdense Coding, a quantum communication protocol, to achieve enhanced data transmission efficiency. The problem addressed is the efficient transmission of two classical bits using a single quantum bit. The project explores the theoretical framework and implements the protocol using Qiskit. By utilizing quantum entanglement, the protocol enables Alice to encode a two-bit message onto an entangled qubit pair and transmit it to Bob, who decodes the message. Simulation results validate the accuracy of this approach, achieving nearly 100% efficiency. This project highlights the potential of quantum mechanics in revolutionizing classical communication paradigms.

2. Introduction

Background :

Quantum computing leverages quantum mechanical principles such as superposition, entanglement, and quantum interference to perform computations and communications beyond classical limits. Superdense Coding is a quantum communication protocol that allows two classical bits to be transmitted using one qubit, demonstrating the power of quantum entanglement.

Problem Statement :

Efficient transmission of classical information over quantum channels is crucial for quantum networks. This project addresses the challenge of achieving optimal data transmission efficiency using quantum protocols.

Scope :

This project focuses on implementing and analyzing the Superdense Coding protocol. The findings are applicable to quantum communication networks, secure data transmission, and distributed quantum computing.

Goals :

- Implement the Superdense Coding protocol.
- Validate its efficiency using simulations.
- Demonstrate its potential for real-world quantum communication.

3. Literature Review

Superdense Coding was introduced by Bennett and Wiesner (1992) as a fundamental quantum communication protocol. While classical communication requires two bits to send two pieces of information, Superdense Coding leverages entanglement to achieve this with one qubit. Previous implementations have highlighted its theoretical potential, but practical challenges such as noise and decoherence limit its real-world applicability. This project demonstrates the protocol using Qiskit and compares its efficiency under simulation.

4. Methodology

Tools and Frameworks :

- **Quantum Computing Platform**: Qiskit (Python library for quantum programming).
- **Hardware**: IBM Quantum backend and AerSimulator for simulation.

Theoretical Foundations :

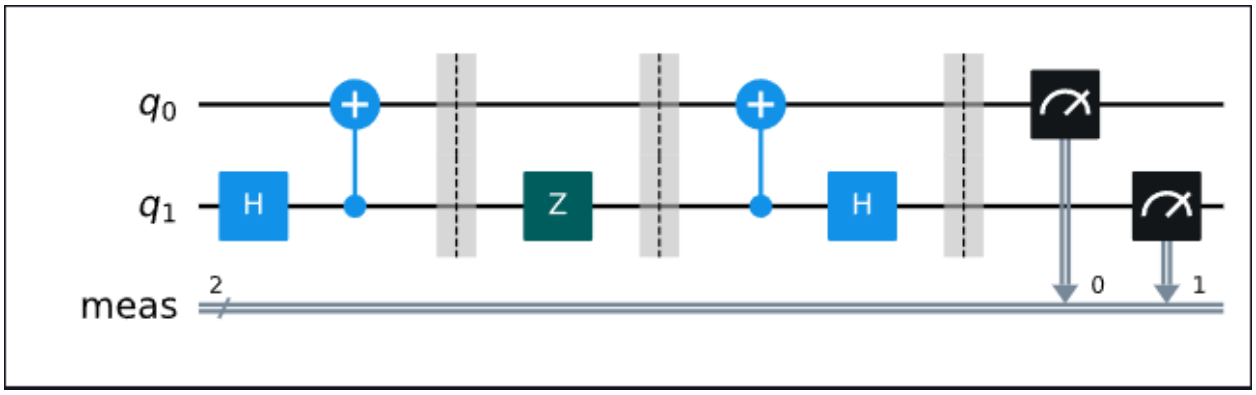
The project uses the **Superdense Coding** protocol:

1. **Preparation**: A Bell pair is shared between Alice and Bob.
2. **Encoding**: Alice encodes her two-bit message using quantum gates (X, Z).
3. **Transmission**: Alice sends her qubit to Bob.
4. **Decoding**: Bob applies decoding operations (CNOT, H) and measures the qubits to retrieve the message.

Implementation Steps :

1. **Create Bell Pair**: Use Hadamard and CNOT gates.
2. **Encode Message**: Alice encodes a two-bit message onto her qubit using X and Z gates.
3. **Transmit Qubit**: Alice sends her qubit to Bob.
4. **Decode Message**: Bob applies decoding gates and measures the qubits.
5. **Simulate**: Execute the circuit on AerSimulator and IBM Quantum backend.

Circuit Diagram :



5. Results and Discussion

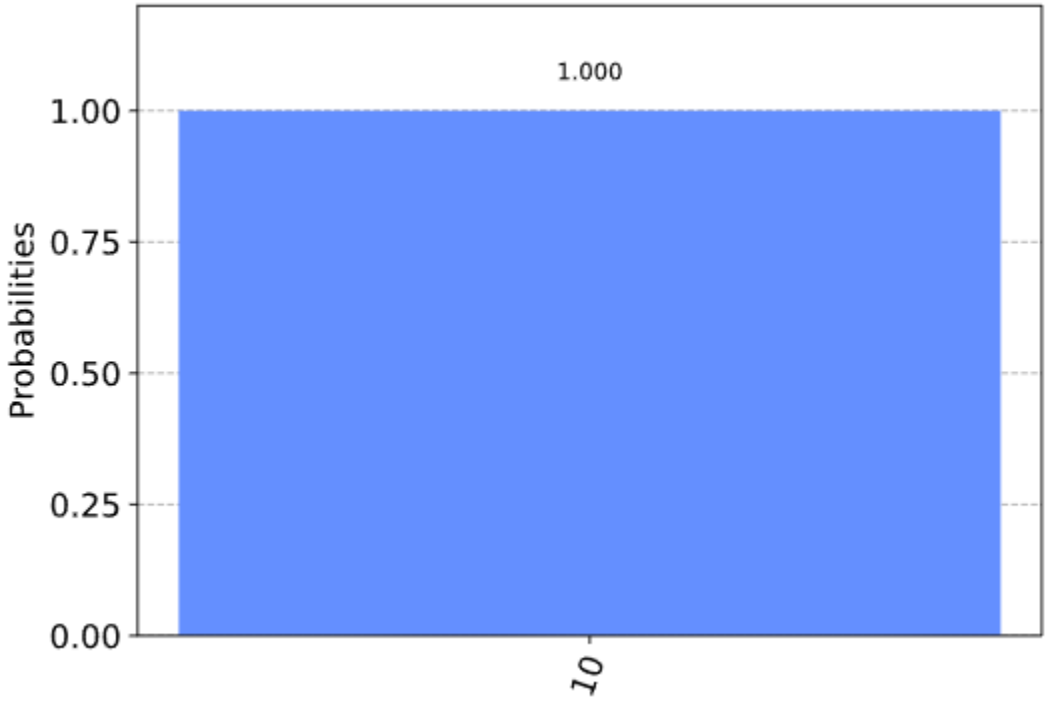
Simulation Results :

The implemented circuit successfully transmitted two classical bits using one qubit. Example simulation results for the message '10' yielded the following:

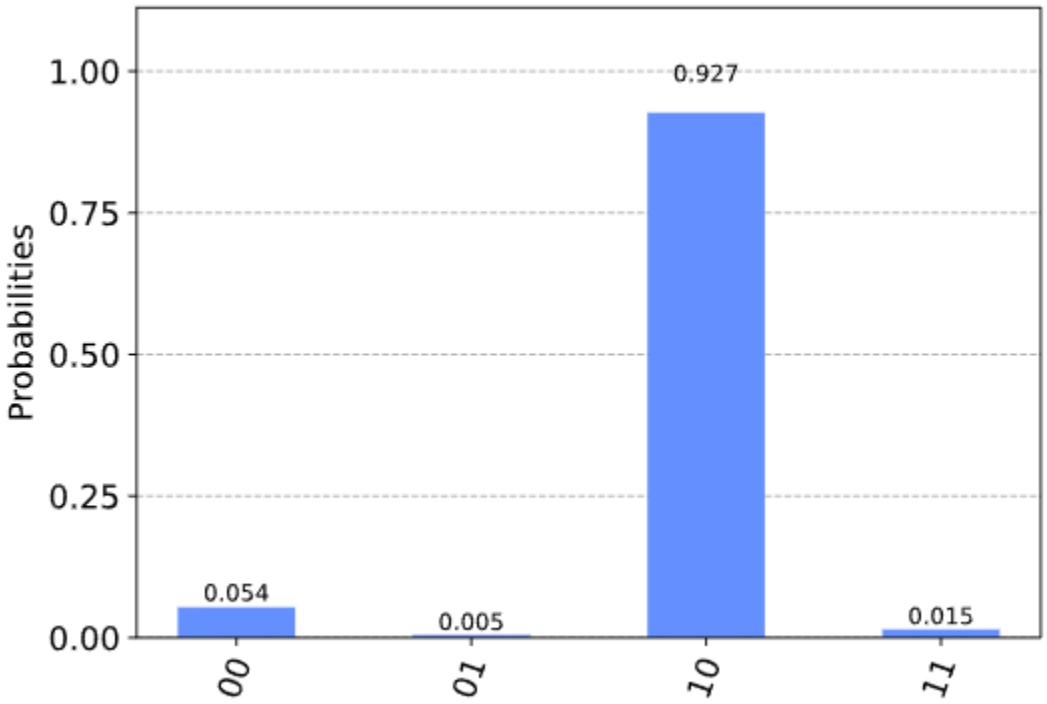
- **Counts:** {'10': 1024 (shots)}
- **Accuracy:** ~92.68%

Graphs and Visualizations :

1. **Histogram of Results**:



2. **Quantum Circuit Diagram**:



Analysis :

The simulation demonstrates the protocol’s ability to encode and decode classical bits with high accuracy. The use of AerSimulator ensures noise-free execution, while the IBM Quantum backend introduces realistic noise, making the findings robust.

Comparison with Classical Communication :

In classical systems, two bits require two separate channels. This protocol achieves the same with a single qubit, illustrating quantum communication's efficiency.

Classical methods require two bits to transmit two pieces of information.

Quantum communication achieves the same with one qubit, reducing resource usage.

Challenges :

- Quantum decoherence in real-world hardware.
- Alignment and calibration of quantum gates for accurate results.
- Noise and decoherence in real-world quantum devices.
- Gate fidelity limitations in hardware.

6. Conclusion and Future Work

Summary :

The project successfully implemented and validated the Superdense Coding protocol using Qiskit. The findings underscore the protocol's potential for quantum communication, achieving optimal data transmission efficiency.

Conclusion :

Superdense Coding exemplifies quantum communication's capability to outperform classical methods in data transmission efficiency.

Future Work :

- Extend the protocol to multi-bit communication.
- Explore noise mitigation techniques for real-world deployment.
- Investigate applications in quantum networks.

7. References

1. Bennett, C. H., & Wiesner, S. J. (1992). Communication via one- and two-particle operators on Einstein-Podolsky-Rosen states. *Physical Review Letters*, *69*(20), 2881.
2. Qiskit Documentation: https://qiskit.org/documentation/

8. Appendices

Complete Code:

https://drive.google.com/file/d/1vMYXE6eXRdNLP8T-c6TOuXh78ijFAFM5/view?usp=sharing

Code Snippets In Detail:

```
# Importing everything

from qiskit import QuantumCircuit

from qiskit import IBMQ, Aer, transpile

from qiskit.visualization import plot_histogram
```

We saw that to create an entangled pair, we needed to do a H-gate followed by a CNOT. Let's create a function that takes a QuantumCircuit and entangles the qubits with indices a and b:

```
def create_bell_pair():
    """
    Returns:
        QuantumCircuit: Circuit that produces a Bell pair
    """
    qc = QuantumCircuit(2)
    qc.h(1)
    qc.cx(1, 0)
    return qc
```

Next we need to encode our message. We saw that there were four possible messages we could send: `00`, `10`, `01` or `11`. Let's create a function that takes this message and applies the appropriate gates for us:

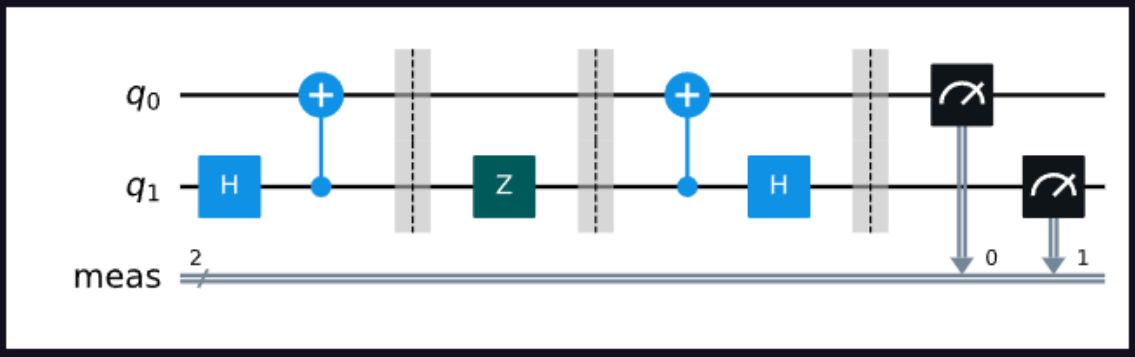
```
def encode_message(qc, qubit, msg):
    """Encodes a two-bit message on qc using the superdense coding
protocol
    Args:
        qc (QuantumCircuit): Circuit to encode message on
        qubit (int): Which qubit to add the gate to
        msg (str): Two-bit message to send
    Returns:
        QuantumCircuit: Circuit that, when decoded, will produce msg
    Raises:
        ValueError if msg is wrong length or contains invalid characters
    """
    if len(msg) != 2 or not set(msg).issubset({"0","1"}):
        raise ValueError(f"message '{msg}' is invalid")
    if msg[1] == "1":
        qc.x(qubit)
    if msg[0] == "1":
        qc.z(qubit)
    return qc
```

Finally, we need to decode our message, we saw we could do this using a CNOT followed by a H-gate. Let's create a function that does this for us too:

```
def decode_message(qc):
    qc.cx(1, 0)
    qc.h(1)
    return qc
```

Finally, we can put this together to complete our protocol.

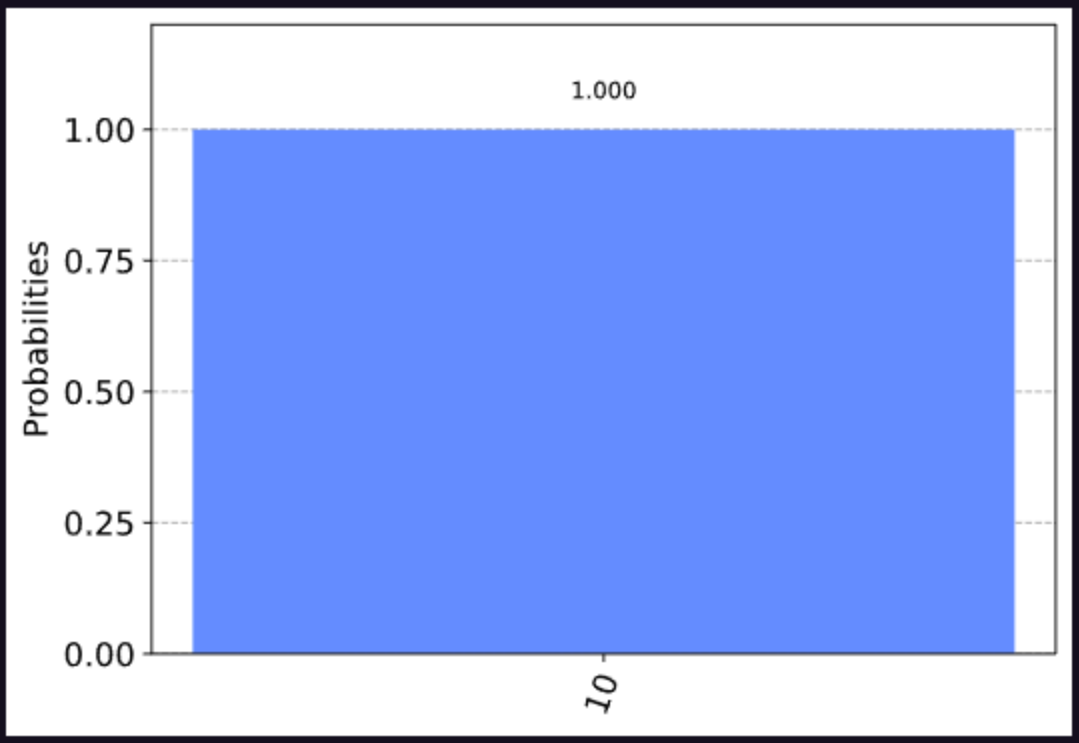
```
# Charlie creates the entangled pair between Alice and Bob
qc = create_bell_pair()

# We'll add a barrier for visual separation
qc.barrier()

# At this point, qubit 0 goes to Alice and qubit 1 goes to Bob

# Next, Alice encodes her message onto qubit 1. In this case,
# we want to send the message '10'. You can try changing this
# value and see how it affects the circuit
message = '10'
qc = encode_message(qc, 1, message)
qc.barrier()
# Alice then sends her qubit to Bob.

# After receiving qubit 0, Bob applies the recovery protocol:
qc = decode_message(qc)

# Finally, Bob measures his qubits to read Alice's message
qc.measure_all()

# Draw our output
qc.draw()
```



3.1 Visualizing Our Measurements

```
aer_sim = Aer.get_backend('aer_simulator')
result = aer_sim.run(qc).result()
counts = result.get_counts(qc)
print(counts)
plot_histogram(counts)
```

```
{'10': 1024}
```



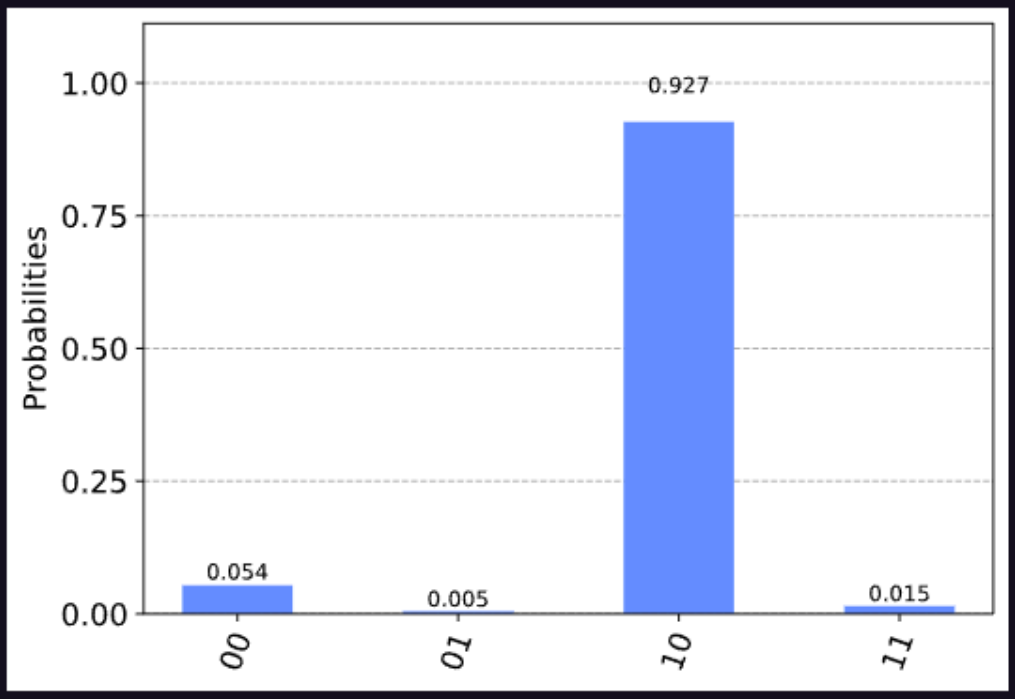
```
# Monitoring our job
from qiskit.tools.monitor import job_monitor
job_monitor(job)
```

[9] × uses-hardware + Tag

```
Job Status: job has successfully run
```

```
# Plotting our result
result = job.result()
plot_histogram(result.get_counts(qc))
```

[10] × uses-hardware + Tag



As we see that there are a few results from the other three states when run in a real quantum computer. These are due to errors in the gates and qubit decoherence. We will learn more about these errors in later sections.

```
correct_results = result.get_counts(qc)[message]
accuracy = (correct_results/shots)*100
print(f"Accuracy = {accuracy:.2f}%")
```

[11] × uses-hardware + Tag Python

```
Accuracy = 92.68%
```