Super dense coding protocol in Python

Indranil Ghosh
Jadavpur University, Physics Department
Email: indranilg49@gmail.com

January 3, 2018

Abstract

This Project is to simulate the Superdense Coding, a simple application of elimentary Quantum mechanics and computations. A Python program has been designed, available in Github (https://github.com/indrag49/Quantum-SimuPy) , to accompany this project. Packages used: Numpy, Pandas and Mathplotlib. The quantum gates and the functions required for this simulation are available in the program and can be easily implemented. The moto of this project is to use python packages to design a quantum simulator for performing simple quantum computations.

1 Introduction

Super Dense Coding protocol was first introduced by Bennet and Wiesner in 1992. In this protocol a number of classical bits is shared between two parties by sharing a few qubits. It consists of two parts: 1) A set of instructions and 2) the deseired results. The protocol is considered to be correct if and only if the set of instructions leads to the desired results. Quantum Entanglement, also described by Einstein as "Spooky action at a distance" is the key resource in Super Dense Coding.

This protocol involves two parties *Alice* and *Bob* who are located at a long distance from one another. Alice has in possession two classical bits of information which she tends to share with Bob by sending only a single qubit to Bob.

2 Procedure

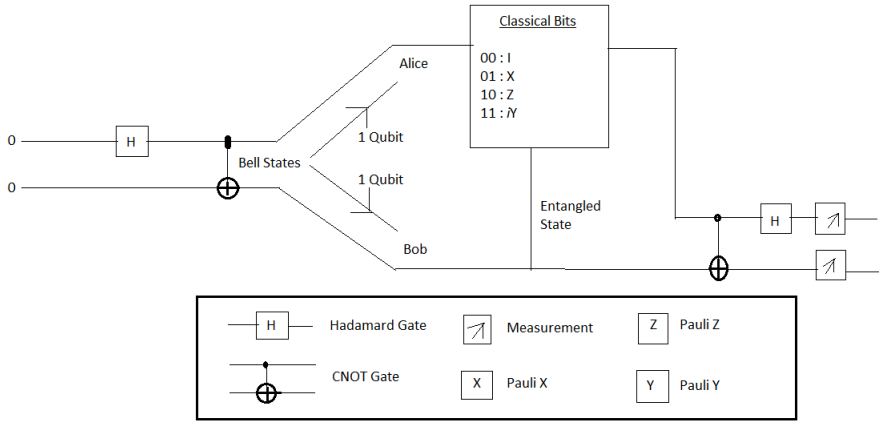

Figure 1: SuperDense Coding Protocol

At first, we prepare the entangled Bell states. We apply $H \otimes I$ unitary operation on the input qubits followed by the CNOT unitary operation(control=0 , target=1). The output we get is the entangled Bell state. We can have 4 Bell states corresponding to inputes: $|00\rangle$, $|01\rangle$, $|10\rangle$ and $|11\rangle$

Input	Output
$\|00\rangle$	$\frac{(\|00\rangle+\|11\rangle)}{\sqrt{2}}$
$\|01\rangle$	$\frac{(\|01\rangle+\|10\rangle)}{\sqrt{2}}$
$\|10\rangle$	$\frac{(\|00\rangle-\|11\rangle)}{\sqrt{2}}$
$\|11\rangle$	$\frac{(\|01\rangle-\|10\rangle)}{\sqrt{2}}$

While computing the Bell states we should remember that to calculate the equivalent gate matrix of two parallel gate matrices, we should compute their tensor product. Using my Python program,

```
def B(Qubit):
        H=Hadamard(I2)
        b=CNOT(np.kron(H, I2))
        return b.dot(Qubit)
b00=B(Q00)
```

b01=B(Q01)
b10=B(Q10)
b11=B(Q11)

The bell states *b00*, *b01*, *b10* and *b11* are prepared. In the Super dense coding protocol we use the Bell state $\frac{(|00\rangle+|11\rangle)}{\sqrt{2}}$, i.e., we input $|00\rangle$ to prepare , as shown in Figure 1. Alice and Bob share this pair of qubits in the entangled state, initially. Alice possesses the first qubit, while Bob, the second. By sending the single qubit in her possession, Alice can easily transfer two bits of classical information to Bob.

Now, Alice follows the following rule:

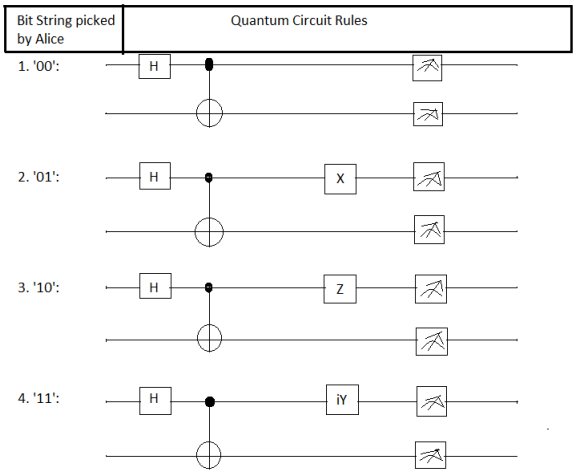

Figure 2: Rules followed by Alice

Input: $\frac{(|00\rangle+|11\rangle)}{\sqrt{2}}$

Output:

Bit String picked by Alice	Gate Applied	Output
'00'	I	$\frac{(\|00\rangle+\|11\rangle)}{\sqrt{2}}$
'01'	X	$\frac{(\|10\rangle+\|01\rangle)}{\sqrt{2}}$
'10'	Z	$\frac{(\|00\rangle-\|11\rangle)}{\sqrt{2}}$
'11'	iY	$\frac{(\|01\rangle-\|10\rangle)}{\sqrt{2}}$

```
def Alice(Q):
        if Q=="00": L=np.kron(I2, I2)
        elif Q=="10":
                z=PauliZ(I2)
                L=np.kron(z, I2)
        elif Q=="01":
                x=PauliX(I2)
                L=np.kron(x, I2)
        else:
                y=PauliY(I2)
                L=np.kron(1j*y, I2)
        return L.dot(b00)
```

The four resulting states are again the Bell states/ *EPR* pairs. They are mutually orthonormal to one another and can be determined by appropriate quantum measurement. To check whether two of the Bell states are orthonormal we calculate their inner product and obtain the result 0. We check some of these examples, using python program:

```
>>> np.inner(b00, b01)
0.0
>>> np.inner(b01, b10)
0.0
>>> np.inner(b00, b10)
0.0
>>> np.inner(b01, b11)
0.0
```

Now, Bob already has a qubit and after receiving the qubit from Alice, he does a measurement in the Bell basis and extracts the result, i.e., determines which of the four classical bit strings Alice sent him. Bob at first applies the CNOT unitary operation followed by $H \otimes I$ unitary operations, on the *EPR* pairs i.e, the reverse of the operations used while preparing the Bell states, to obtain the origibnial classiical bits sent by Alice.

Bell States	Output
$\|b00\rangle$	$\|00\rangle$
$\|b01\rangle$	$\|01\rangle$
$\|b10\rangle$	$\|10\rangle$
$\|b11\rangle$	$\|11\rangle$

```
def Bob(Qubit):
        s=CNOT(Qubit)
        return s.dot(np.kron(Hadamard(I2), I2))
```

If a hacker, Ada, intercepts Alice's qubit en route to Bob, all she obtains is a part of the entangled state and no useful information is stolen by her, untill and unless she interacts with Bob's qubit.

References

[1] http://www-reynal.ensea.fr/docs/iq/QC10th.pdf

[2] https://github.com/indrag49/Quantum-SimuPy