

Quantum Tic Tac Toe

Course Project for PHY 318

Mah Noor Jamil
2023-10-0002

April 28, 2023

Abstract

In this project, we propose a game of Quantum Tic Tac Toe: a classical game of Tic Tac Toe, but with a quantum twist. With the implementation of quantum moves of “Collapse” and “Entanglement” in an accompanying, interactive quantum circuit, players can use the power of quantum mechanics to throw off an opponent or win the entire game. Such a quantum game aims to show the player (who may have a penchant for quantum physics) to see the inherently probabilistic nature of quantum mechanics and its other properties in real-time - and in a fun, interactive way.

Contents

1	Introduction	2
2	Mathematical Formulation	2
2.1	Gates Used	2
2.1.1	Hadamard Gate	2
2.1.2	CNOT Gate	3
2.2	Initial Configuration of the Board	3
2.3	Rules	3
2.4	Step-by-step Game Play	4
3	Implementation Methodology	8
3.1	Circuit Set-up	8
3.2	Implementation of “Collapse”	9
3.3	Tripping Your Opponent using “Entanglement”	12
4	Results	15
4.1	Probabilistic Analysis	15
4.2	IBM Results	17
5	Conclusion	19

1 Introduction

Quantum mechanics is a complex field; concepts like entanglement, superposition and the like can all get quite abstract even for experts. Considering how popular this field has become, it is imperative that the fundamentals of quantum mechanics be made more palatable to the avidly interested student - and what better way to do so than through a quantum mechanical game. For this project, we shall illustrate and explain a game of 3×3 tiled Quantum Tic Tac Toe, through its implementation on a quantum circuit. Through this, the goal is to show all the important fundamental aspects of [3]:

1. Superposition: the ability of a quantum state to be in two states at once
2. Entanglement: the phenomenon by which the state of one component can be influenced by the state of another component, through some kind of correlation
3. Collapse: the process through which a quantum state reduces - or “collapses” - to a classical state.

2 Mathematical Formulation

In this section, we shall explore the mathematics behind our quantum version of Tic Tac Toe, which will accompany the main component of the game: the quantum circuit, with which both players are supposed to interact with. However, considering that the rules of the game have been kept deliberately simple, the aim in this portion of the paper is to provide the players with a kind of sketch as to how states within the tile are evolving as a game proceeds. Therefore, instead of keeping the players completely out of the loop about all the quantum mechanics hidden behind the game-board, instead the players are encouraged to explore the consequences of these quantum gates; through this, they may even be able to create ever-changing strategies following every step in order to win a game. [2]

First, we will present the gates that will be used in the quantum circuit during the game-play. Then, the initial configuration of the game-board will be presented, wherein each tile contains a superposition state of **X** and **O**. Finally, to show the gates inter-playing with the superposition states within the tiles (in accordance with legal moves) a step-by-step hypothetical game-play will be shown, wherein two versions of the game-board will be shown: one will be a “classical” game-board, which will show the positions of **X**’s and **O**’s after each move, and another will be a “quantum” game-board, which contains all the information about the gates used and states existing in each tile.

2.1 Gates Used

2.1.1 Hadamard Gate

This gate is a quantum logic gate that acts on a single qubit. Its sole responsibility is to create equal superposition of $|0\rangle$ and $|1\rangle$, the final form of which depends on what kind of qubit is it acting on:

$$\begin{aligned}U_H |0\rangle &= \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle), \\U_H |1\rangle &= \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle).\end{aligned}$$

Its matrix representation is as follows [1]:

$$U_H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{2.1}$$

For the purpose of our quantum game of Tic Tac Toe, the Hadamard gate is used to create equal superposition of $|0\rangle$ and $|1\rangle$ in all the tiles at the beginning of the game.

2.1.2 CNOT Gate

This is a quantum logic gate that acts on two qubits instead of one. One qubit acts as the control, while the other as a target; therefore, these qubits are also called control and target qubits respectively. Its purpose is to flip the state of the target qubit when the control qubit is precisely $|1\rangle$. Its matrix form can be written as:

$$U_{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad (2.2)$$

For the purpose of our game, this gate will be used to create entanglement with two states, such that they form two bell states (explained in more detail ahead).

2.2 Initial Configuration of the Board

We will start the way any Tic Tac Toe game begins: by considering a 3×3 tiled game-board - except, in this quantum counterpart, each tile is not necessarily empty. Instead, each tile contains an equal superposition state of $|0\rangle$ and $|1\rangle$:

$$|\psi\rangle = \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle). \quad (2.3)$$

$ \psi\rangle$	$ \psi\rangle$	$ \psi\rangle$
$ \psi\rangle$	$ \psi\rangle$	$ \psi\rangle$
$ \psi\rangle$	$ \psi\rangle$	$ \psi\rangle$

2.3 Rules

The end goal is the same as classical Tic Tac Toe: the first one to get a complete row, column or diagonal filled with uninterrupted **X**'s or **O**'s wins. How to get there, though, requires a little more than just the simple rules of the classical game.

- (a) Each player can do one of two legal plays: (1) collapse, and (2) entanglement.
- (b) For the "Collapse" move, the player will have to chose one of the tiles, and then it will either get assigned $|0\rangle$ or $|1\rangle$, each having a 50% probability. (And yes, both players will be well aware of that risk beforehand!) This move can happen legally in one of two types of tiles: (1) one that has not been occupied yet, and (2) a tile locked in a control terminal of a CNOT gate. A tile collapsed once cannot be collapsed again.

- (c) For the “Entanglement” move, the player will apply the CNOT gate across two tiles. However, to make the game simpler, the target terminal must be placed at an already “collapsed” tile for it to be a legal move. That way, there are only two types of entangled bell states that are possible:

$$|B_1\rangle = U_{\text{CNOT}}(|\psi\rangle|0\rangle) = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle), \quad (2.4)$$

$$|B_2\rangle = U_{\text{CNOT}}(|\psi\rangle|1\rangle) = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle). \quad (2.5)$$

It must be noted that a state that has already been entangled cannot be entangled further with any other tile (classical or not).

This inclusion of “entanglement” allows a player to be able to present an obstacle for the opponent: instead of allowing them to maintain their tile, the other player can entangle their chosen tile with their opponent’s, so as to render their occupied tile “smeared”. The best way to show the efficacy of this move is by showing it in a real game.

2.4 Step-by-step Game Play

As mentioned in the beginning of this section, the following step-by-step game play is supposed to show the reader how our newly modified “quantum” moves can be applied by players competing to win. The right side contains the “classical” board - i.e. the board that the players will be able to see their results on. On the left hand side, there is the “quantum” board, which represents all the gates used and quantum states evolved as a result. Note that C in each tile represents a “Collapse” move placed there, and E represents an “Entanglement” move; for the sake of maintaining a history of all moves made in a complete game, all such C and E notations are retained in each tile. Gates that are colored black are operational, and when they turn grey, that is when they become obsolete or ineffective. In each move, we shall show each player’s strategy as we go along; more precisely, we will show how knowledge of quantum states can motivate players to strategise their next move, so as to secure a win.

1. *Move 1 by Player X: “Collapse” on Tile 4.*

0.	1.	2.
3.	4. X	5.
6.	7.	8.

0.	1.	2.
3.	4. 1⟩ <i>C</i>	5.
6.	7.	8.

2. *Move 2 by Player O: “Collapse” on Tile 2.*

0.	1.	2. O
3.	4. X	5.
6.	7.	8.

0.	1.	2. $ 0\rangle$ <i>C</i>
3.	4. $ 1\rangle$ <i>C</i>	5.
6.	7.	8.

3. *Move 3 by Player X: “Entanglement” of Tile 6 (Control) with Tile 2 (Target).* At this stage, Player **X** wishes to get the upper-hand by securing the corners, while simultaneously attempt to topple Player **O**’s hold in Tile 2. Thus, he chooses the “Entanglement” move, which smears the information of **O** in Tile 2. The resulting state one gets is:

$$|\psi_{6\rightarrow 2}\rangle = U_{CNOT} |\psi\rangle |0\rangle = \frac{1}{\sqrt{2}} (|00\rangle + |11\rangle).$$

Notationally, $|\psi_{6\rightarrow 2}\rangle$ represents the entangled bell state wherein the qubit in Tile 6 is the control qubit and the qubit in Tile 2 is the target qubit. This new entangled state exists in both Tiles 6 and 2; what exact state resides in each tile is decided when a “Collapse” move is made in Tile 6, in which case there is a 50% probability of both tiles getting $|0\rangle$ or $|1\rangle$. Until then, both are in an entangled superposition of sorts, as shown in the figure below.

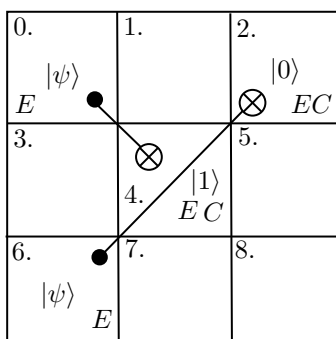
0.	1.	2. $\frac{\mathbf{O}}{\mathbf{X}}$
3.	4. X	5.
6. $\frac{\mathbf{O}}{\mathbf{X}}$	7.	8.

0.	1.	2. $ 0\rangle$ \otimes <i>EC</i>
3.	4. $ 1\rangle$ <i>C</i>	5.
6. $ \psi\rangle$ <i>E</i>	7.	8.

4. *Move 4 by Player O: “Entanglement” of Tile 0 (Control) with Tile 4 (Target).* Now, Player **O** notices **X**’s trick, and so goes for a similar trick to loosen **X**’s grasp in the center tile; he entangles Tiles 0 and 4, such that the qubit in Tile 0 is the control qubit, and Tile 4 is the target qubit. This new entangled state is as follows:

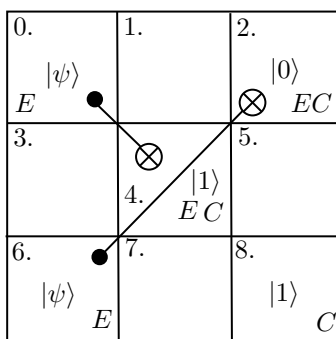
$$|\psi_{0\rightarrow 4}\rangle = U_{CNOT} |\psi\rangle |1\rangle = \frac{1}{\sqrt{2}} (|01\rangle + |10\rangle).$$

0.	$\frac{\text{O}}{\text{X}}$	1.		2.	$\frac{\text{O}}{\text{X}}$
3.		4.	$\frac{\text{O}}{\text{X}}$	5.	
6.	$\frac{\text{O}}{\text{X}}$	7.		8.	



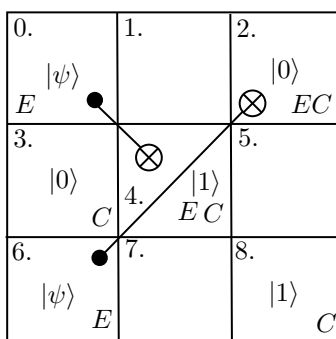
5. Move 5 by Player X: "Collapse" at Tile 8.

0.	$\frac{\text{O}}{\text{X}}$	1.		2.	$\frac{\text{O}}{\text{X}}$
3.		4.	$\frac{\text{O}}{\text{X}}$	5.	
6.	$\frac{\text{O}}{\text{X}}$	7.		8.	X



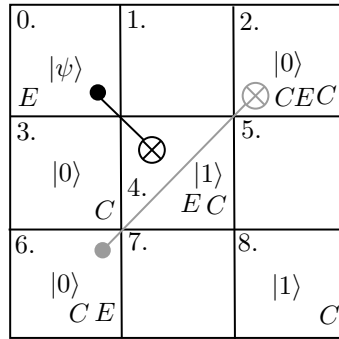
6. Move 6 by Player O: "Collapse" at Tile 3.

0.	$\frac{\text{O}}{\text{X}}$	1.		2.	$\frac{\text{O}}{\text{X}}$
3.	O	4.	$\frac{\text{O}}{\text{X}}$	5.	
6.	$\frac{\text{O}}{\text{X}}$	7.		8.	X



7. Move 7 by Player X: "Collapse" at Tile 6. In an attempt to finally secure the corners, Player X takes the risk and collapses $|\psi_{6 \rightarrow 2}\rangle$. But to his dismay, the states chosen are $|0\rangle$ on both Tiles 6 and 2. The entanglement placed before is now rendered obsolete.

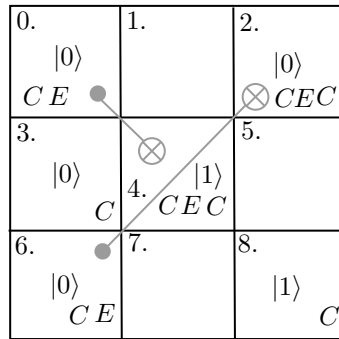
0.	$\frac{O}{X}$	1.		2.	O
3.	O	4.	$\frac{O}{X}$	5.	
6.	O	7.		8.	X



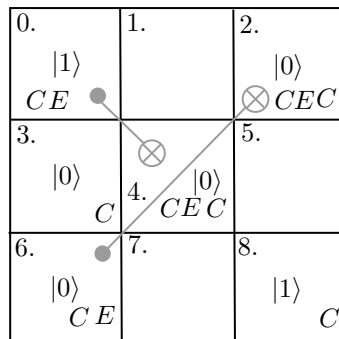
8. *Move 8 by Player O: "Collapse" at Tile 0.* Player **O** now goes for the kill and chooses the "Collapse" move in Tile 0, for which there is a 50% probability of either of the following:

- (a) Tile 0: $|0\rangle$ and Tile 4: $|1\rangle$. In this scenario, Player **O** wins.
- (b) Tile 0: $|1\rangle$ and Tile 4: $|0\rangle$. In this scenario, Player **O** wins again.

0.	O	1.		2.	O
3.	O	4.	X	5.	
6.	O	7.		8.	X



0.	X	1.		2.	O
3.	O	4.	O	5.	
6.	O	7.		8.	X



3 Implementation Methodology

3.1 Circuit Set-up

To implement the use of quantum moves in an otherwise classical game of Tic Tac Toe, we shall ascribe to each tile two qubits: a *goti-bit* and an *occupation-bit*. The goti-bit records the final move made on the gameboard: $|0\rangle$ represents “O” on the tile, and $|1\rangle$ represents “X” on the tile. The occupation-bit records the presence of either an “X” or an “O” on the tile by any player; thus, if an occupation bit carries $|0\rangle$, then that means that corresponding tile is empty, and if it carries $|1\rangle$, then that corresponding tile is occupied by either an “X” or an “O”. Thus, our circuit has a total of 18 qubits, where the first 9 qubits represent the goti-bits for each tile, and the next 9 qubits represent the corresponding occupation-bits.

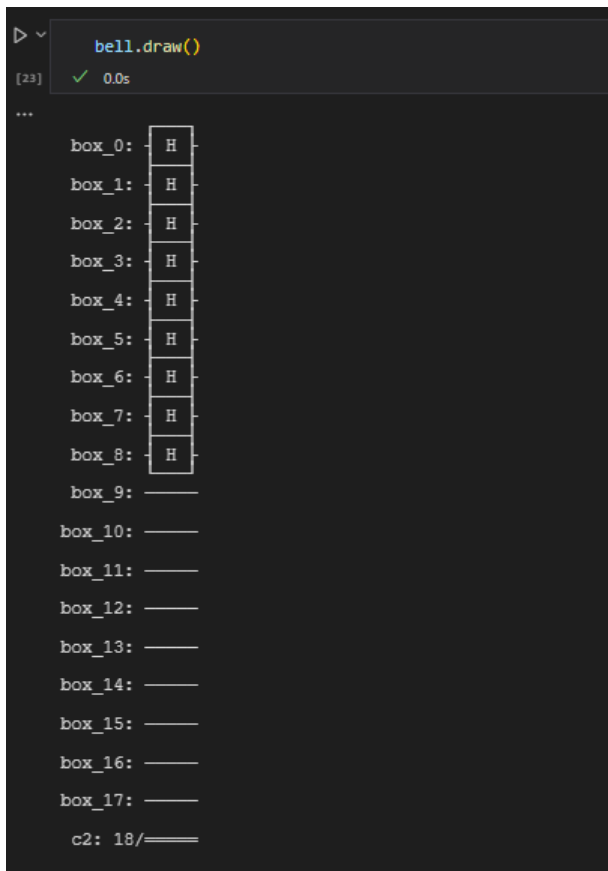


Figure 1: Set-up of the circuit at the start of the game.

Here, the first 9 qubit channels starting from 0 to 8 all represent goti-bits; in the beginning, all tiles are empty, and so each of them carry equal superpositions of $|0\rangle$ and $|1\rangle$ implemented by the Hadamard gate. The next 9 qubits starting from 9 to 17 are all occupation bits; since no move has been made at the start, they are all “empty”. Note that the arrangement of these goti and occupation bits are by no means random; they are arranged in order of channel number. For e.g., the first goti-bit’s (labelled as the 0^{th} qubit in the above circuit) occupation bit is the qubit labelled as the 9^{th} in the circuit - i.e. the first occupation bit out of the 9. Just like this, the second goti-bit’s (labelled 1^{st} in circuit) occupation bit is the 10^{th} qubit (i.e. the second occupation bit), and so on. Diagrammatically they can be paired as follows:

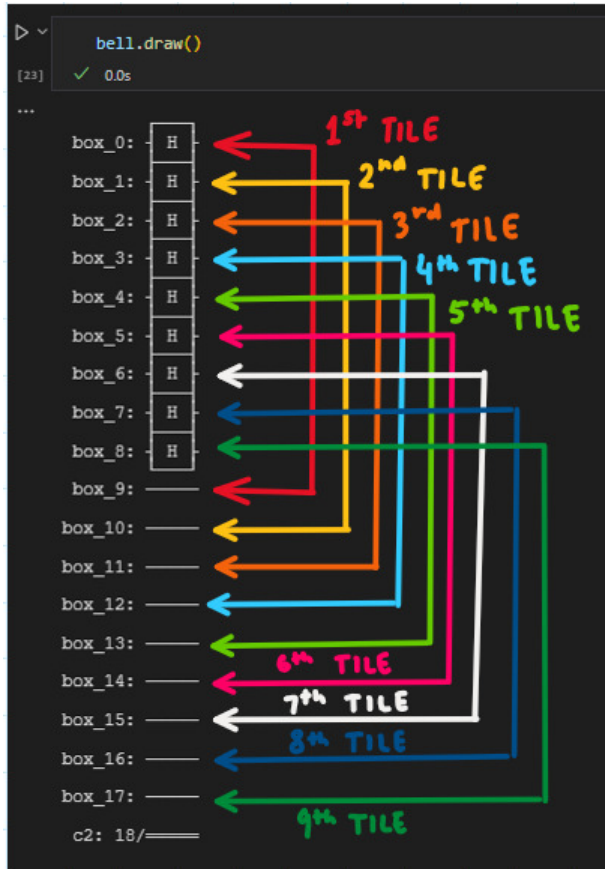


Figure 2: Goti and occupation bit pairings in each tile.

3.2 Implementation of “Collapse”

Consider a player wishing to implement the “Collapse” move on the 0^{th} tile. The code implements the following:

```

counts1,cir = simpleRead(bell,0)
newstate1 = max(counts1,key=counts1.get)
print('New state after collapse: ', newstate1)
[15] ✓ 5.0s
... New state after collapse: 0000000100000000

```

Figure 3: New state after “Collapse” on 0^{th} tile.

The “counts1” carries a dictionary of the two types of outputs: the 0^{th} tile gets either $|0\rangle$ or $|1\rangle$, with each state getting different number of counts or frequency. “Cir” carries the new circuit after having applied two measurement gates: one on the 0^{th} channel, and one on the 9^{th} channel (i.e. one on the first goti-bit, and one on the first occupation-bit). The quantum circuit will appear as follows:

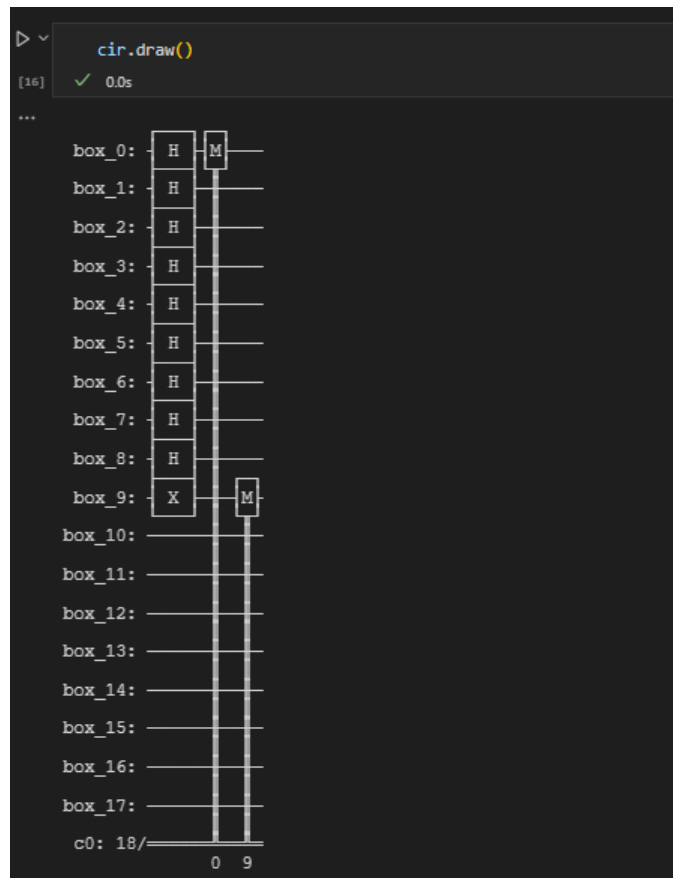


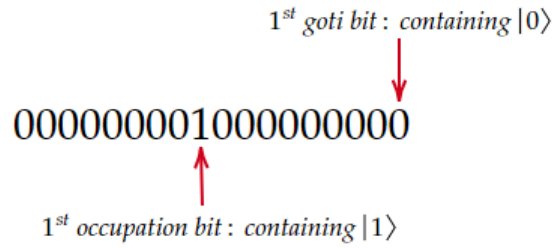
Figure 4: Circuit after “Collapse” on 0^{th} tile.

Note that the occupation bit gets an X gate because the 0^{th} tile gets “occupied”, therefore the $|0\rangle$ must be converted to $|1\rangle$, before measurement. The “newstate1” string printed reads off only that state with the most number of counts (i.e. the most likely). It must be noted that this string is printed in reverse order; for the sake of clarity, we shall present the order of qubit labels shown in the string:

$$\begin{array}{c}
 \text{“ [17] [16] [15] [14] [13] [12] [11] [10] [9] [8] [7] [6] [5] [4] [3] [2] [1] [0] ”} \\
 \underbrace{\hspace{10em}}_{\text{Occupation bits}} \quad \underbrace{\hspace{10em}}_{\text{Goti bits}}
 \end{array}$$

Figure 5: Ordering of digits in the “New state” string, labelled according to their corresponding channels in the quantum circuit.

So, since the player has done the “Collapse” move on the 0^{th} tile, the 1st occupation bit (labelled 9 in the string above) will be set to $|1\rangle$, and the output of either $|0\rangle$ or $|1\rangle$ will reflect in the 1st goti bit (labelled 0 in the string above). In our specific output, we get the following:



In order to save all this information for future moves in the game, this circuit must be recreated with this new state using the `resetCircuit` function:

```

c1: 18/=====
box_17: -----
box_16: -----
box_15: -----
box_14: -----
box_13: -----
box_12: -----
box_11: -----
box_10: -----
box_9:  X
box_8:  H
box_7:  H
box_6:  H
box_5:  H
box_4:  H
box_3:  H
box_2:  H
box_1:  H
box_0:  -----

```

Since the 0th tile has been occupied, the 1st occupation bit labelled 9 gets an X gate to reflect $|1\rangle$; and since the state read is $|0\rangle$, the 1st goti bit gets no gate to retain $|0\rangle$ state. At this stage, the gameboard will appear as follows:

```

board = XorOBoard(newstate1)
print(board)
[216] ✓ 0.1s
...
[['O' 'P1' 'P2']
 ['P3' 'P4' 'P5']
 ['P6' 'P7' 'P8']]

```

3.3 Tripping Your Opponent using “Entanglement”

Implementation of “Entanglement” using the entangledMove function is quite straightforward. It asks for two inputs from the player: the first input is the position of the target qubit (which has to be a state that is already collapsed into $|0\rangle$ or $|1\rangle$) and the second input is the position of the control qubit (which must be a superposition state corresponding to an unoccupied state). Both these inputs are then used to apply the CNOT gate accordingly. An example of such a move is shown below, wherein the target qubit is placed in the channel labelled 0, and the control qubit is placed in the channel labelled 1:

```

entangledMove(cir)
cir.draw()
[17] ✓ 4.4s
...

```

```

box_0: ———— X ————
          |
box_1: ———— H ————
          |
box_2: ———— H ————
          |
box_3: ———— H ————
          |
box_4: ———— H ————
          |
box_5: ———— H ————
          |
box_6: ———— H ————
          |
box_7: ———— H ————
          |
box_8: ———— H ————
          |
box_9: ———— X ————
          |
box_10: ————
box_11: ————
box_12: ————
box_13: ————
box_14: ————
box_15: ————
box_16: ————
box_17: ————
c1: 18/══════════

```

Now, the solitary $|0\rangle$ state in the 0^{th} tile has been entangled with the superposition state in the 1^{st} tile; the information in the previous tile has now been “smeared” with this new tile, such that now there’s a 50-50 chance of both tiles getting either $|0\rangle$ or $|1\rangle$ due to the CNOT transformation (as shown in Equation 2.4).

On the gameboard, this shall be reflected as follows:

```
board = XorOBoard(newstate1)
print(board)
[219] ✓ 0.0s
... [['X/O' 'X/O' 'P2']
      ['P3' 'P4' 'P5']
      ['P6' 'P7' 'P8']]
```

Which one of the two will land on the tiles, is decided whenever the player decides to make a “Collapse” move. We have presented one scenario where, upon application of the simpleRead function and preparing a new state hereafter, we get the following resultant state:

```
counts1,cir = simpleRead(cir,newstate1)
newstate1 = max(counts1,key=counts1.get)
print('New state after collapse: ', newstate1)
[77] ✓ 7.1s
... New state after collapse: 000000011000000000
```

Here, the “Collapse” move has been done on the 1st tile, which caused both the 0th and the 1st tile to get occupied by |0>; that is confirmed when we see the 9th and 10th positions (in accordance with Figure 5) of the string with 1’s, meaning the 0th and 1st tile have been occupied. The reset circuit reflects this as well:

```

cir = resetCircuit(newstate1)
[78] ✓ 0.0s

cir.draw()
[79] ✓ 0.0s
...
box_0: _____
box_1: _____
box_2: | H
box_3: | H
box_4: | H
box_5: | H
box_6: | H
box_7: | H
box_8: | H
box_9: | X
box_10: | X
box_11: _____
box_12: _____
box_13: _____
box_14: _____
box_15: _____
box_16: _____
box_17: _____
c9: 18/=====

```

On the game board, this will appear as follows:

```

board = XorOBoard(newstate1)
print(board)
[324] ✓ 0.1s
...
[['0' '0' 'P2']
 ['P3' 'P4' 'P5']
 ['P6' 'P7' 'P8']]

```

The true significance of this move arises in an alternative scenario, wherein measurement of the entangled 1st tile gives rise to both the 0th and 1st tiles getting $|1\rangle$:

```

counts1,cir = simpleRead(cir,newstate1)
newstate1 = max(counts1,key=counts1.get)
print('New state after collapse: ', newstate1)
[197] ✓ 7.1s
... New state after collapse: 000000011000000011

```

```

cir = resetCircuit(newstate1)
[198] ✓ 0.0s

cir.draw()
[199] ✓ 0.1s
...
box_0: X
box_1: X
box_2: H
box_3: H
box_4: H
box_5: H
box_6: H
box_7: H
box_8: H
box_9: X
box_10: X
box_11: —
box_12: —
box_13: —
box_14: —
box_15: —
box_16: —
box_17: —
c24: 18/==

```

What this has done is that it has legally allowed player **X** to overwrite player **O**'s $|0\rangle$ placed in the 0^{th} tile with $|1\rangle$, thereby giving player **X** a fair advantage in the game. This will reflect in the gameboard as follows:

```

board = XorOBoard(newstate1)
print(board)
[200] ✓ 0.0s
...
[['X' 'X' 'P2']
 ['P3' 'P4' 'P5']
 ['P6' 'P7' 'P8']]

```

4 Results

4.1 Probabilistic Analysis

In the previous section, we have already shown how the use of entanglement within a regular game of Tic Tac Toe makes it to where one player can get an advantage over the other. As a matter of fact, we can

actually quantify this advantage by seeing probabilities of a certain kind of outcome, first in a game with no entanglement, and then in another with entanglement included.

Let us then begin by considering a game of Tic Tac Toe wherein we are focusing on just two tiles, labelled S_1 and S_2 . If we only had the ‘‘Collapse’’ move allowed, then performing the move on both tiles would give us four possible outcomes:

1. Both tiles get \mathbf{O} ($\mathbf{O}_1, \mathbf{O}_2$)
2. Both tiles get \mathbf{X} ($\mathbf{X}_1, \mathbf{X}_2$)
3. S_1 gets \mathbf{O} and S_2 gets \mathbf{X} ($\mathbf{O}_1, \mathbf{X}_2$)
4. S_1 gets \mathbf{X} and S_2 gets \mathbf{O} ($\mathbf{X}_1, \mathbf{O}_2$)

Since both tiles carry an equal superposition state of $|0\rangle$ and $|1\rangle$ as expressed in Equation 2.3, then there’s a half chance of getting $|0\rangle$ or $|1\rangle$ (i.e. \mathbf{O} or \mathbf{X} respectively) in each tile:

$$P(\mathbf{O}_1) = P(\mathbf{X}_1) = P(\mathbf{O}_2) = P(\mathbf{X}_2) = \frac{1}{2}.$$

Let us define a random variable X as the number of squares which contain \mathbf{X} in S_1 and S_2 . Then, we have the following probabilities for each allowed value of X :

$$\begin{aligned} P(X = 0) &= P(\mathbf{O}_1, \mathbf{O}_2) = \frac{1}{4}, \\ P(X = 1) &= P(\mathbf{O}_1, \mathbf{X}_2) + P(\mathbf{X}_1, \mathbf{O}_2) = \frac{1}{2}, \\ P(X = 2) &= P(\mathbf{X}_1, \mathbf{X}_2) = \frac{1}{4}. \end{aligned}$$

Using this, we can now evaluate the expectation value of random variable X :

$$E[X] = \sum_x x \times P(X = x) = \frac{1}{2} + \left(2 \times \frac{1}{4}\right) = 1.$$

How does this change when entanglement is added into the mix? We have just seen that the game becomes advantageous when entanglement allowed player \mathbf{X} to overwrite what player \mathbf{O} had placed in a given tile. Lets see how does that change the expectation value of random variable X .

We will be taking into consideration the case where player \mathbf{X} entangles S_2 with S_1 only when S_1 gets collapsed to $|0\rangle \equiv \mathbf{O}$ (according to the rules of the game, S_1 tile is the target qubit and the S_2 tile is the control qubit). Otherwise, if S_1 gets $|1\rangle \equiv \mathbf{X}$, then there will be no entanglement. At the end, S_1 will be measured using the ‘‘Collapse’’ move right after S_2 is measured; this way, if there is any entanglement between the two tiles, then the effect can be seen in S_1 . All the possible outcomes are still the same, except now the likelihood of getting two tiles with \mathbf{X} ’s increases: not just $(\mathbf{X}_1, \mathbf{X}_2)$, but also $(\mathbf{O}_1, \mathbf{X}_2)$, because when S_2 carries $|1\rangle \equiv \mathbf{X}$, the CNOT gate will flip the $|0\rangle$ state to $|1\rangle$ in S_1 if present. The probability distribution changes:

$$\begin{aligned} P(X = 0) &= P(\mathbf{O}_1, \mathbf{O}_2) = \frac{1}{4}, \\ P(X = 1) &= P(\mathbf{X}_1, \mathbf{O}_2) = \frac{1}{4}, \\ P(X = 2) &= P(\mathbf{X}_1, \mathbf{X}_2) + P(\mathbf{O}_1, \mathbf{X}_2) = \frac{1}{2}. \end{aligned}$$

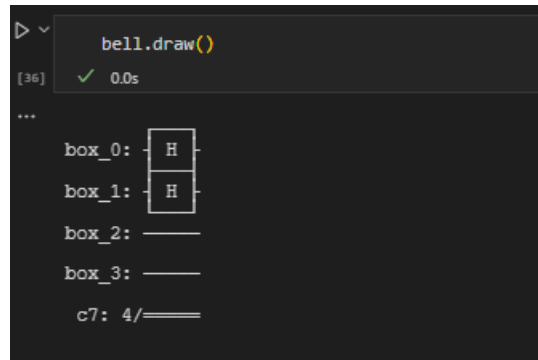
New expectation value of random value X:

$$E[X] = \sum_x x \times P(X = x) = \frac{1}{4} + \left(2 \times \frac{1}{2}\right) = 1.25.$$

This means that the expected number of tiles with X after the two measurements has increased from before. This goes to show mathematically that indeed, the entanglement move proves advantageous [2].

4.2 IBM Results

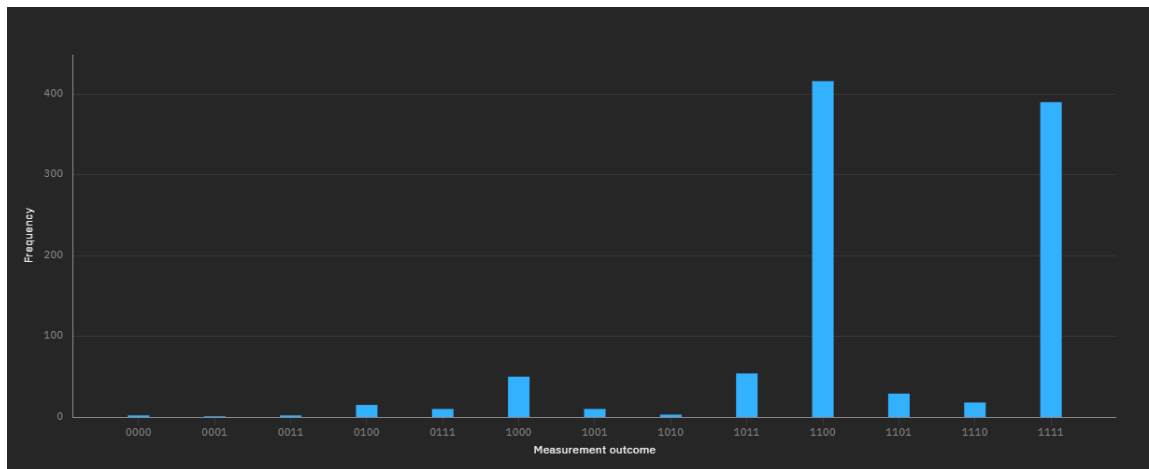
To show the validity of these simulation results, we have chosen to run this game’s algorithm on IBM’s “Nairobi”. However, due to the quantum computer’s limitation of at most 7 qubits, we cannot expect a full quantum Tic Tac Toe game - with all of its 18 qubits - to run on IBM. Instead, we have chosen to send a small 4 qubit circuit instead, wherein we’re considering two goti bits and two corresponding occupation bits; these can be thought of as the two tiles we had been discussing before, namely S_1 and S_2 .



```
bell.draw()
[36] ✓ 0.0s
...
box_0: H
box_1: H
box_2: 
box_3: 
c7: 4/
```

Since the two tiles are empty, they each contain a superposition of $|0\rangle$ and $|1\rangle$ from the Hadamard gate’s application. We shall recreate what we had discussed before: we will conduct a “Collapse” move on the 0^{th} tile, then entangle the 0^{th} and 1^{st} tiles together, and then finally conduct a “Collapse” move on the 1^{st} tile. While the first collapse will be done using the quantum game’s simulator function, the second one will be done on the IBM ‘Nairobi’ computer.

As already discussed, both S_1 and S_2 should get either $|0\rangle$ or $|1\rangle$ after the entanglement, both with equal probabilities as expressed in Equation 2.4. From the looks of the histogram that follows, we see that the state ‘1100’ (i.e. the output state with $|0\rangle$ on both goti bits, and $|1\rangle$ on both occupation bits) is the most likely of them all:



Also note that the second most frequent output we get is ‘1111’ (i.e. the output state with $|1\rangle$ on both qubit bits, and $|1\rangle$ on both occupation bits). Both of these having such high frequencies is expected because of the nature of the entangled state that we had. However, what is unusual are the non-zero frequencies of the other output states that seem impossible to obtain in the confines of our simulated game (for e.g. ‘0100’, ‘1001’, ‘1011’ etc.). They represent noise inherent within the quantum processor chosen, i.e. the IBM ‘Nairobi’, under the Falcon processor; from its documentation, it can be seen that the error corresponding to the CNOT gate is of the order 10^{-4} , while its readout error is of the order 10^{-2} . These are quite small, which goes to show that their contributions can be considered negligible when applied to a simple 4 qubit circuit. That way, even if we have these other non-zero frequencies, we can choose to ignore them when playing our game.

This output is also reflected in the job results that we get, in the form of the “new state”:

```

#Job has returned from IBM, and is being displayed here

job = provider.backend.retrieve_job('ch42o1011tcjfh1q0br0')
counts_output = job.result().get_counts()
newstate_output = max(counts_output, key=counts_output.get)
print('New state from IBM: ', newstate_output)

[13] ✓ 1.7s
... New state from IBM: 1100

```

The corresponding circuit we get is as follows:

```

newstate_final = max(counts, key=counts.get)
circuit_final = resetCircuit(newstate_final)
circuit_final.draw()

[33]
...
box_0: _____
box_1: _____
box_2: [ X ]
box_3: [ X ]
c6: 4/=====

```

5 Conclusion

In this project, we have successfully formulated and analysed the application of a quantum version of Tic Tac Toe, in accordance with three vital components of quantum mechanics: superposition of two quantum states, its collapse, and the entanglement of quantum states. This allowed us to make this quantum game a physical, interactive reality which works both on a quantum computer, as well as in simulation. To conduct a thorough analysis, we have explained the set-up of the game, its rules, and how it all can be applied on an 18 qubit quantum circuit, which reflects the dynamics of the game play. More importantly, we have illustrated how the notion of entanglement can help one player gain an advantage over the other by “overwriting” their move beforehand; we even showed the efficacy of this using a probabilistic analysis of expectation values, and how they change before and after including entanglement.

Despite our quantum game being extremely simple with a limited set of rules, it is remarkable how the inclusion of just entanglement and superposition changes the classical game dramatically, making it a game that’s as challenging as chess. This may very well lead us to cook up further improvements that are possible, such as the possible inclusion of the NOT gate, the Hadamard gate, etc. Arguably, these will add to the complexity of the game, but it would be interesting to explore their results. Regardless, through this project we have achieved the end goal: we have made a quantum version of a classically simple game, which exemplifies all the fundamental aspects of quantum mechanics, in a way that is palatable to any avid player.

References

- [1] M. A. Nielsen and I. L. Chuang, “*Quantum Computation and Quantum Information*”, Cambridge University Press, (2010), pp. 500-501.
- [2] M. Nagy and N. Nagy, ”Quantum Tic-Tac-Toe: A Genuine Probabilistic Approach,” *Applied Mathematics*, Vol. 3 No. 11A, 2012, pp. 1779-1786.
- [3] W. Maurice, W. Tim, “Quantum Tic-Tac-Toe - learning the concepts of quantum mechanics in a playful way”, *Computers and Education Open* 4, (2022).