UNIVERSIDAD
COMPLUTENSE
MADRID

Universidad Complutense de Madrid

Facultad de Ciencias Físicas

Mecánica Cuántica

Proyecto de Innovación Educativa en Computación Cuántica

Autor: Álvaro Mandado Díaz

Profesor: Felipe J. LLanes Estrada

1 de mayo de 2025

Índice

1. Introducción

Este documento presenta lo requerido en las diferentes sesiones del Proyecto de Innovación Educativa en Computación Cuántica.

2. Sesión 1: Introducción visual a circuitos cuánticos con QUIRK

En esta sesión utilizaremos QUIRK, un software libre dedicado a la presentación de circuitos cuánticos con una interfaz sencilla para introducirnos a la computación cuántica.

2.1. Ejercicio 2.1.1

Los equivalentes de los siguientes productos de puertas cuánticas son:

a. $H \cdot H = \mathbb{1}$

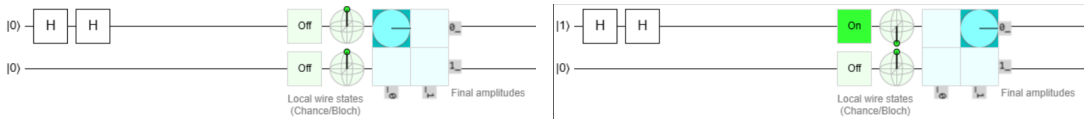


Figura 1: Salida del circuito $H \cdot H$ para la base computacional (fijarse solo en el primer qubit).

b. $X \cdot X = \mathbb{1}$

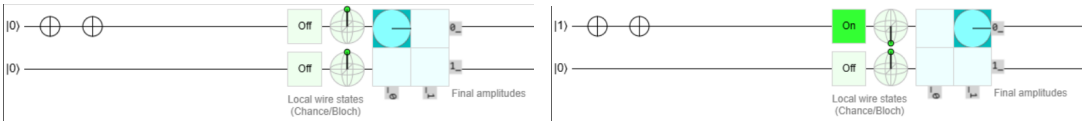


Figura 2: Salida del circuito $X \cdot X$ para la base computacional (fijarse solo en el primer qubit).

c. $Y \cdot Y = \mathbb{1}$

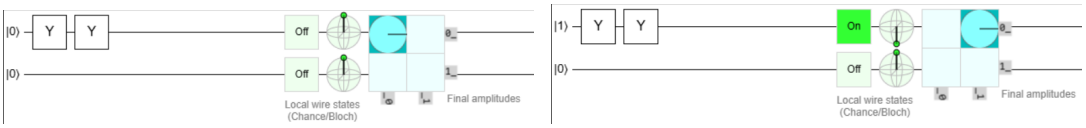


Figura 3: Salida del circuito $X \cdot X$ para la base computacional (fijarse solo en el primer qubit).

d. $Z \cdot Z = \mathbb{1}$

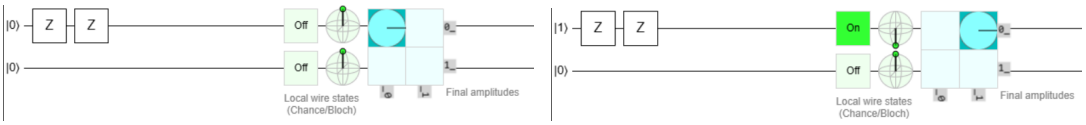


Figura 4: Salida del circuito $Z \cdot Z$ para la base computacional (fijarse solo en el primer qubit).

e. $X^{1/2} \cdot X^{1/2} = X$

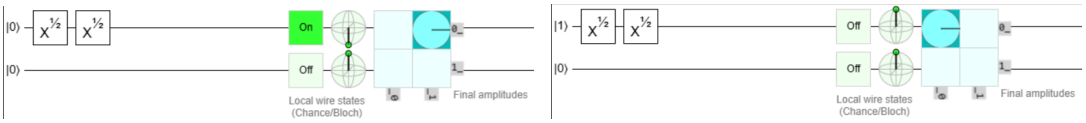


Figura 5: Salida del circuito $X^{1/2} \cdot X^{1/2}$ para la base computacional (fijarse solo en el primer qubit).

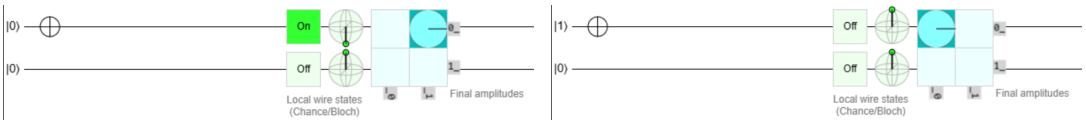


Figura 6: Salida del circuito X para la base computacional (fijarse solo en el primer qubit).

f. $X^{1/4} \cdot X^{1/4} = X^{1/2}$

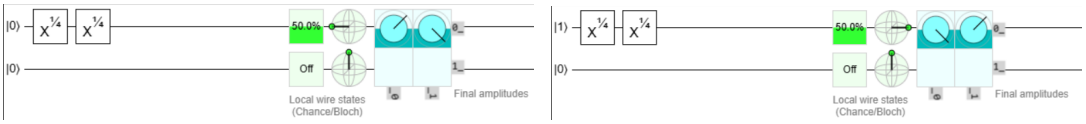


Figura 7: Salida del circuito $X^{1/4} \cdot X^{1/4}$ para la base computacional (fijarse solo en el primer qubit).

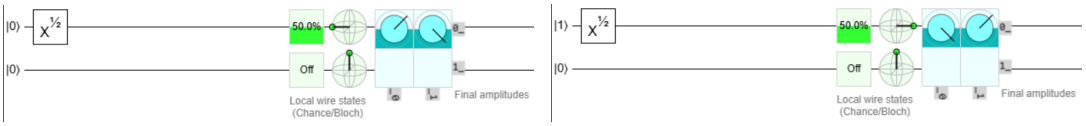


Figura 8: Salida del circuito $X^{1/2}$ para la base computacional (fijarse solo en el primer qubit).

g. $H \cdot Y \cdot H = -Y$, introduciendo una fase global $e^{i\delta} = -1$ respecto del operador Y.

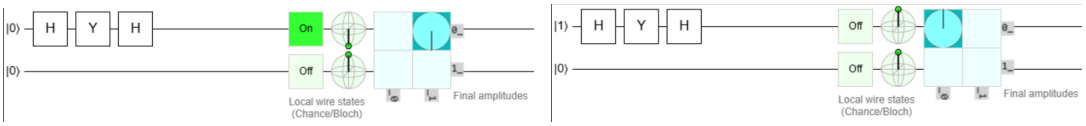


Figura 9: Salida del circuito $H \cdot Y \cdot H$ para la base computacional (fijarse solo en el primer qubit).

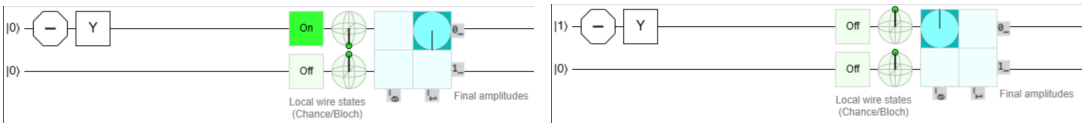


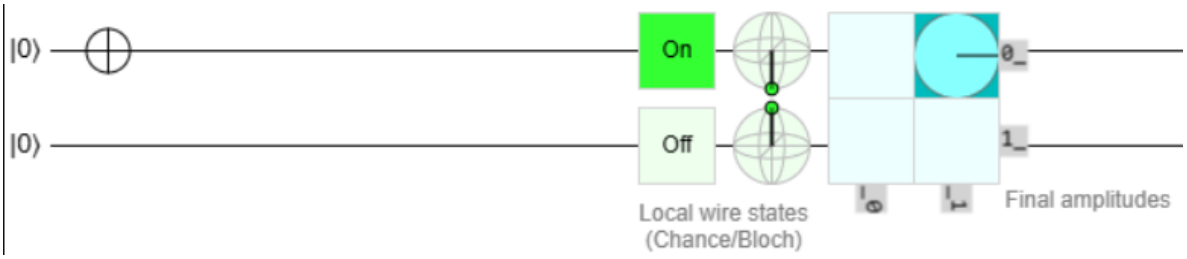
Figura 10: Salida del circuito $-Y$ para la base computacional (fijarse solo en el primer qubit).

En las figuras se ven las imágenes de los vectores de la base, $|0\rangle$ y $|1\rangle$, a través de las puertas. El resultado es la matriz densidad del estado tras las puertas, donde el azul intenso indica la intensidad de ese escalar (su módulo al cuadrado) y la aguja indica su fase. Por ejemplo, en el apartado a) observamos cómo el estado $|0\rangle$ tras las puertas $H \cdot H$ va otra vez a sí mismo (intensidad total con fase nula en la coordenada $|0\rangle$).

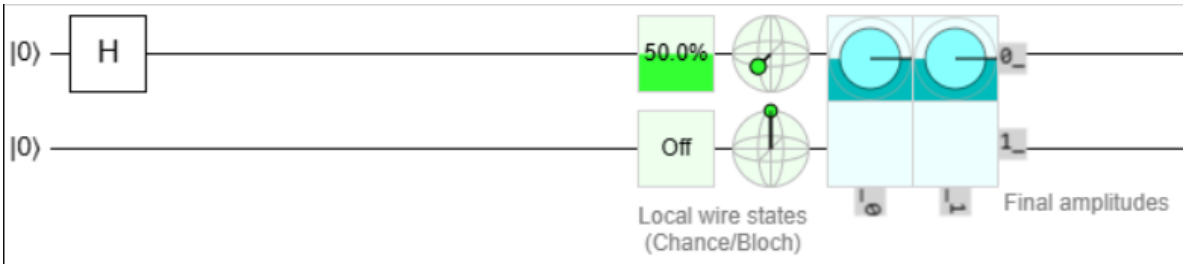
2.2. Ejercicio 2.1.2

Los circuitos que obtienen esas salidas son:

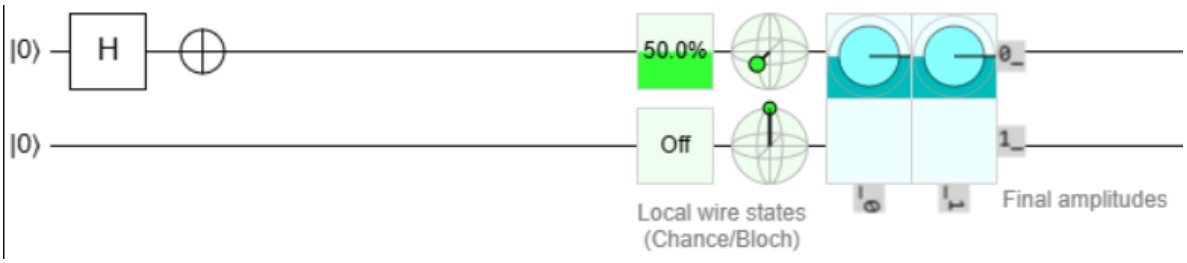
a. $X|0\rangle = |1\rangle$



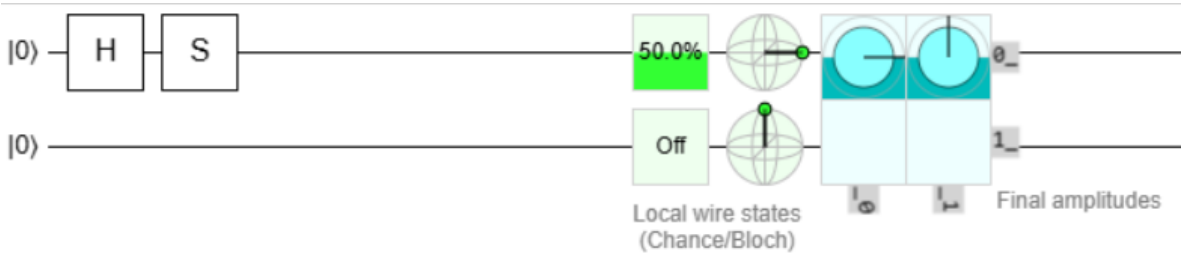
b. $H|0\rangle = |+\rangle$



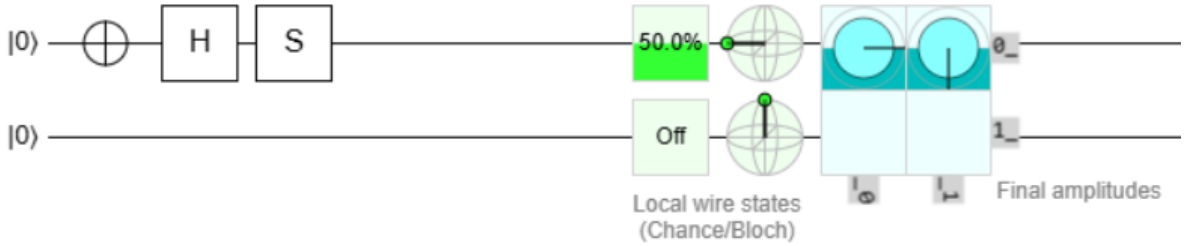
c. $X \cdot H|0\rangle = |-\rangle$



d. $S \cdot H\,|0\rangle = |R\rangle$



e. $S \cdot H \cdot X\,|0\rangle = |L\rangle$



2.3. Ejercicio 3.1

Veamos qué transformaciones son unitarias:

a. Ordenar alfabéticamente **no** es unitaria: $(\langle a|\,\langle b|)\,(|b\rangle\,|a\rangle) = 0$, pero al transformarlos el segundo par se ordena y obtenemos $(\langle a|\,\langle b|)\,(|a\rangle\,|b\rangle) = 1$.

b. Permutar **sí** es unitaria: Como el producto escalar se hace subsistema a subsistema, únicamente estamos reordenando esos subsistemas, luego el producto escalar no cambia.

c. Sumar dos elementos módulo d y perder ambos **no** es unitaria: Si lo fuese sería invertible, pero por ejemplo $|a\rangle\,|b\rangle$ y $|a+1\rangle\,|b-1\rangle$ ambos dan lugar a $|a \oplus b\rangle\,|0\rangle$, luego no es inyectiva.

d. Sumar dos elementos módulo d y guardar alguno **sí** es unitaria: Ahora, teniendo uno de los elementos guardado, podemos recuperar el otro como $a \equiv (a \oplus b) \ominus b$, pues ese elemento a es único salvo congruencia módulo d, y solo tomamos valores entre 0 y $d-1$, lo cual asegura la unicidad. Así, el operador es invertible. Además, es fácil ver lo siguiente:
Veamos que $\langle x, y|x', y'\rangle = \langle x \oplus y, y|x' \oplus y', y'\rangle$. Como estamos en el espacio producto, tenemos lo siguiente:

$$\begin{aligned} \langle x, y|x', y'\rangle &= \langle x|x'\rangle\,\langle y|y'\rangle \\ \langle x \oplus y, y|x' \oplus y', y'\rangle &= \langle x \oplus y|x' \oplus y'\rangle\,\langle y|y'\rangle \end{aligned} \tag{2.1}$$

Nótese que si $y \neq y'$ módulo d, entonces ambos productos escalares son automáticamente cero. Así en ese caso coinciden. Si ahora tomamos $y = y'$, con lo que $\langle y|y'\rangle = 1$, se tiene que $\langle x \oplus y|x' \oplus y'\rangle = \langle x \oplus y|x' \oplus y\rangle$.

Si ahora $x \neq x'$ módulo d, entonces $x \oplus y \neq x' \oplus y$ módulo d y por tanto se vuelven a anular ambos productos. Sin embargo, si $x = x'$ módulo d, entonces $\langle x \oplus y|x' \oplus y\rangle = \langle x \oplus y|x \oplus y\rangle =$

$1 = \langle x|x\rangle = \langle x|x'\rangle$, luego vuelven a coincidir.

Por tanto, concluimos que en efecto $U^\dagger U = \mathbb{1}$, luego U (el operador en cuestión) es unitario.

e. Simetrizar (o antisimetrizar) **sí** es unitaria: El factor $1/\sqrt{2}$ lo asegura.

f. La operación del ejemplo **no** es unitaria: Si nos quedamos solo con números positivos, $(\langle 1|\,\langle -1|)\,(|1\rangle\,|-2\rangle) = 0$, pero al borrar los números negativos tendremos $\langle 1|1\rangle = 1$. De alguna manera, al no guardar una copia podemos estar perdiendo información si las instrucciones no generan una operación invertible.

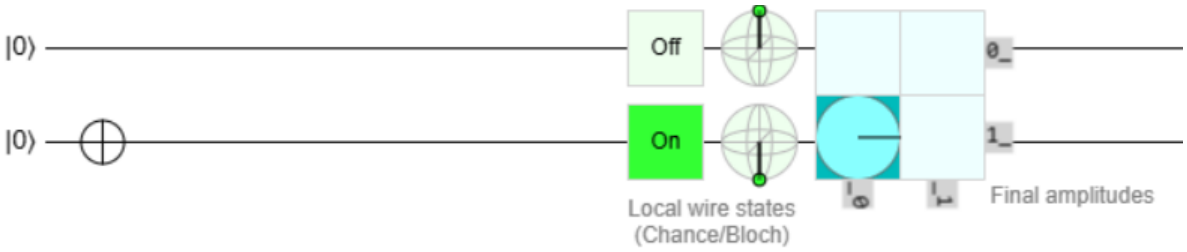
2.4. Ejercicio 3.2

Para cada una de estas puertas, en el ejercicio 2.1.1 hemos visto que se tiene $A \cdot A = \mathbb{1}$, luego concluimos que $A^{-1} = A$. Para comprobar que son unitarias basta con ver $A = A^\dagger$, pero eso se tiene puesto que las matrices de Pauli son autoadjuntas.

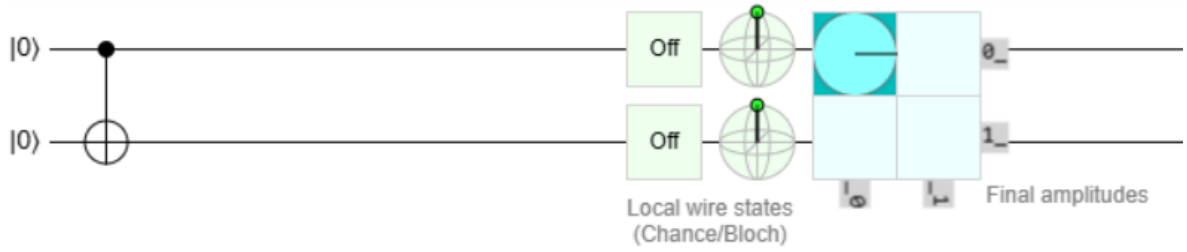
2.5. Ejercicio 4.1

Los cambios tras los sucesivos pasos son:

- Tras 2), el segundo qubit pasa del estado $|0\rangle$ al estado $|1\rangle$, el primero no cambia.

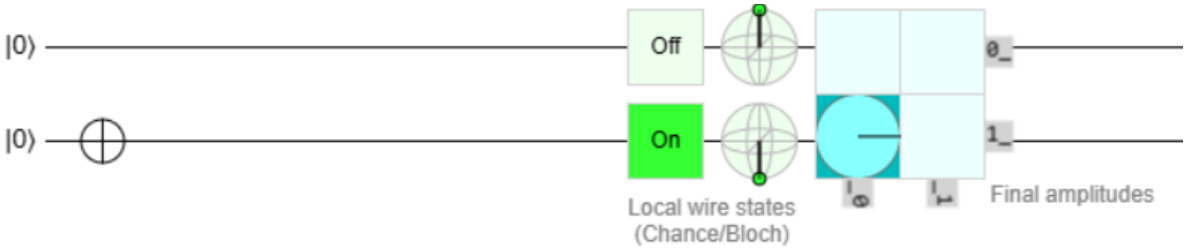


- Tras 3), el segundo qubit no cambia, el primero tampoco.

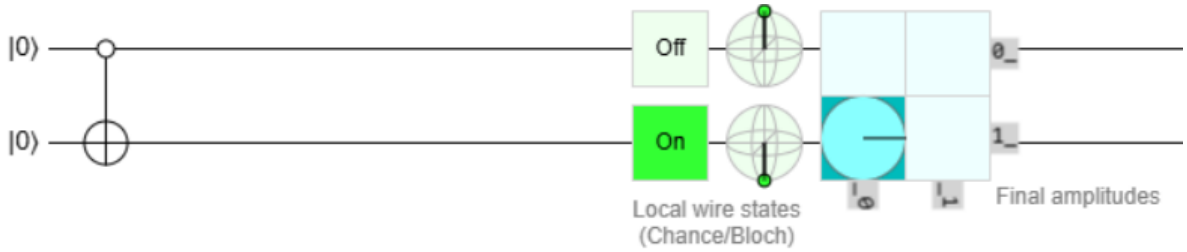


- Tras 4), el segundo qubit pasa del estado $|0\rangle$ al estado $|1\rangle$, el primero no cambia, pero indica a la puerta que el segundo debe cambiar.

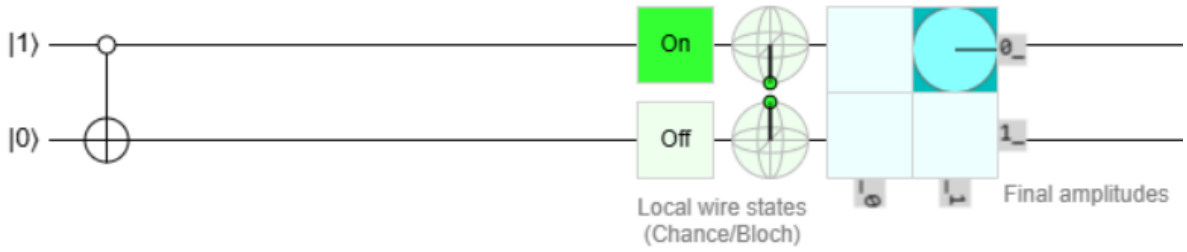
- Cambiando *control* por *anti-control* y tras 2), el segundo qubit pasa del estado $|0\rangle$ al estado $|1\rangle$, el primero no cambia.



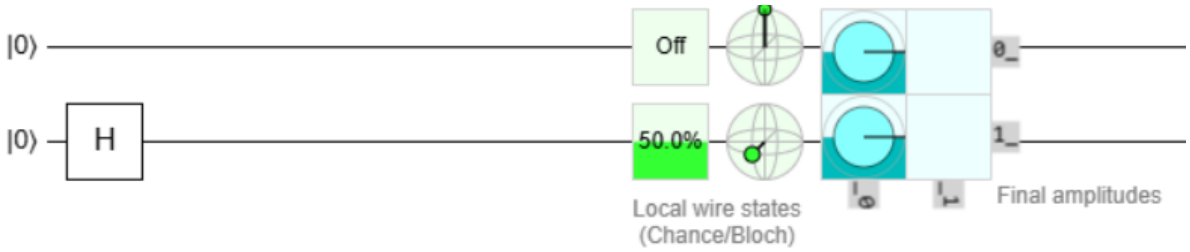
- Cambiando *control* por *anti-control* y tras 3), el segundo qubit pasa del estado $|0\rangle$ al estado $|1\rangle$, el primero no cambia, pero indica a la puerta que el segundo debe cambiar.



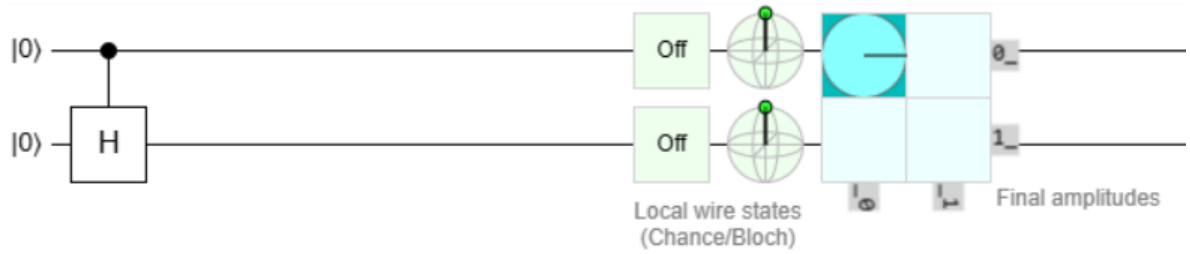
- Cambiando *control* por *anti-control* y tras 4), el segundo qubit no cambia, el primero tampoco, pues la instrucción es contraria a la de la puerta con *control*.



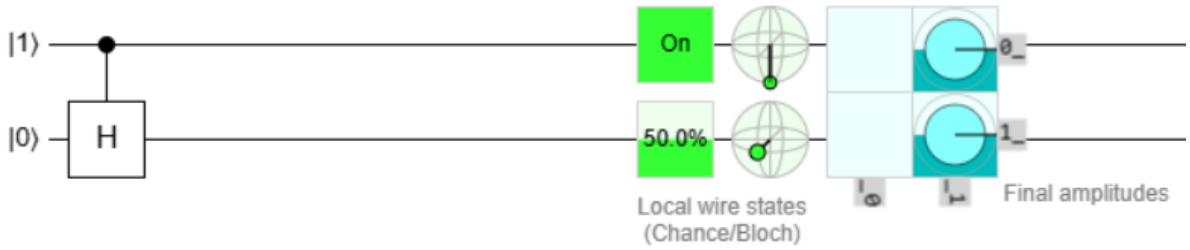
- Cambiando X por H y tras 2), el segundo qubit pasa del estado $|0\rangle$ al estado $|+\rangle$, el primero no cambia.



- Cambiando X por H y tras 3), el segundo qubit no cambia, el primero tampoco.



- Cambiando X por H y tras 4), el segundo qubit pasa del estado $|0\rangle$ al estado $|+\rangle$, el primero no cambia, pero indica con *control* que el segundo debe cambiar.

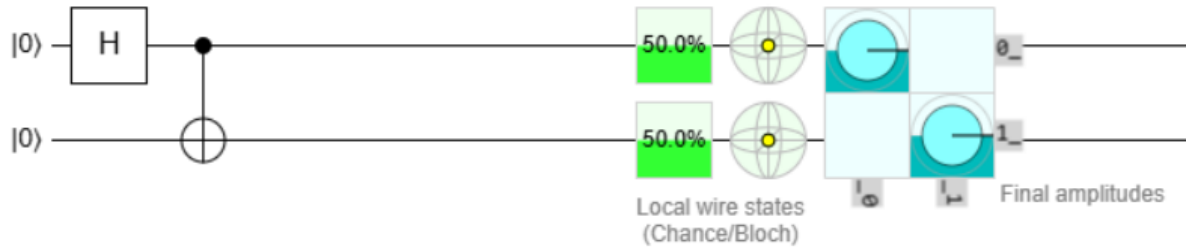


Se podría usar también *anti-control* con H, pero el resultado es análogo y esperable.

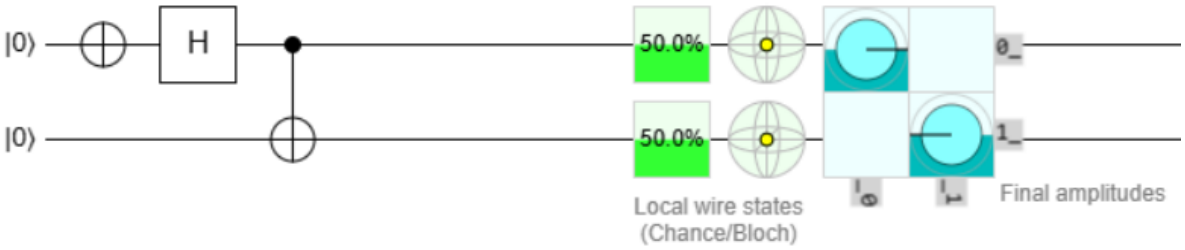
2.6. Ejercicio 4.2

Veamos cómo obtener los distintos estados:

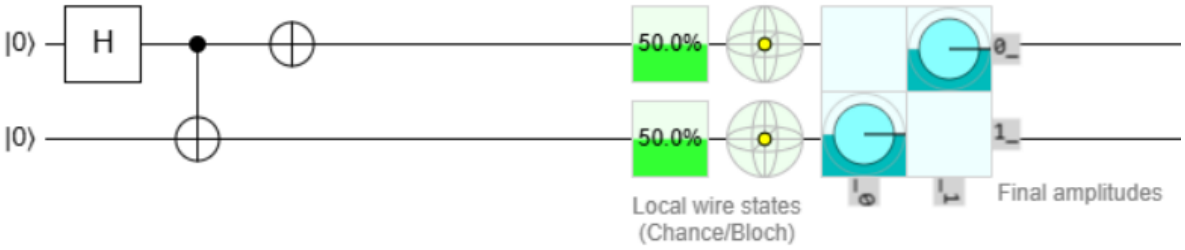
a. Para obtener $|\Phi^+\rangle$, usamos la puerta que nos indica el enunciado.



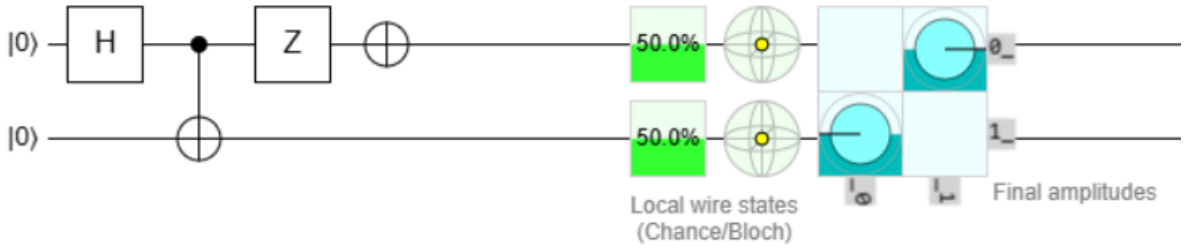
b. Para obtener $|\mathbf{\Phi}^-\rangle$, usamos la puerta que nos indica el enunciado, previa negación del primer qubit.



c. Para obtener $|\mathbf{\Psi}^+\rangle$, usamos la puerta que nos indica el enunciado, posterior negación del primer qubit. También podríamos negar el segundo, pues ahora están entrelazados.



d. Para obtener $|\mathbf{\Psi}^-\rangle$, usamos la puerta que nos indica el enunciado, posterior paso por Z y luego por X del primer qubit, cambiando el signo del término en $|10\rangle$ respecto del caso anterior.

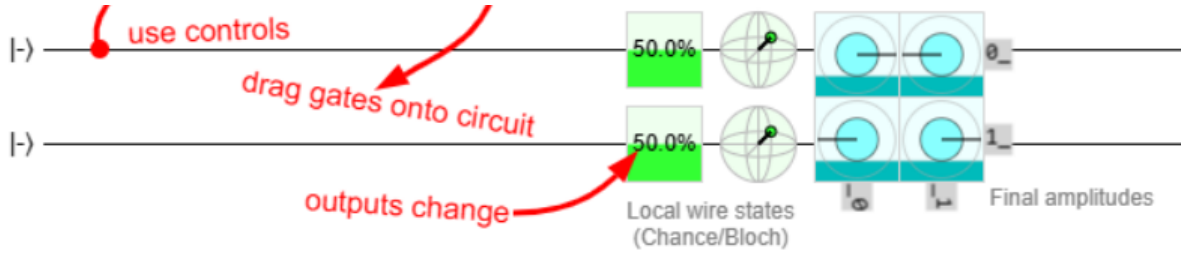


2.7. Ejercicio 4.3.1

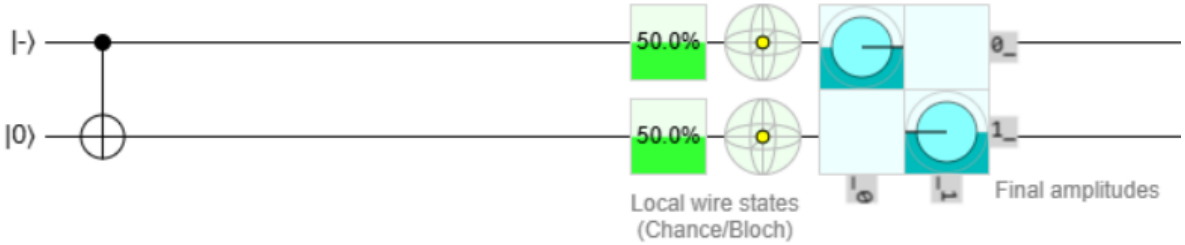
a. Para calcularlos simplemente hacemos lo siguiente:

$$\begin{aligned} |-\rangle \otimes |-\rangle = \frac{1}{2}\left(|0\rangle - |1\rangle\right) \otimes \left(|0\rangle - |1\rangle\right) = \\ = \frac{1}{2}\left(|0\rangle \otimes |0\rangle - |0\rangle \otimes |1\rangle - |1\rangle \otimes |0\rangle + |1\rangle \otimes |1\rangle\right) = \frac{1}{2}\left(1, -1, -1, 1\right)^T \end{aligned} \tag{2.2}$$

En efecto, lo comprobamos con QUIRK:



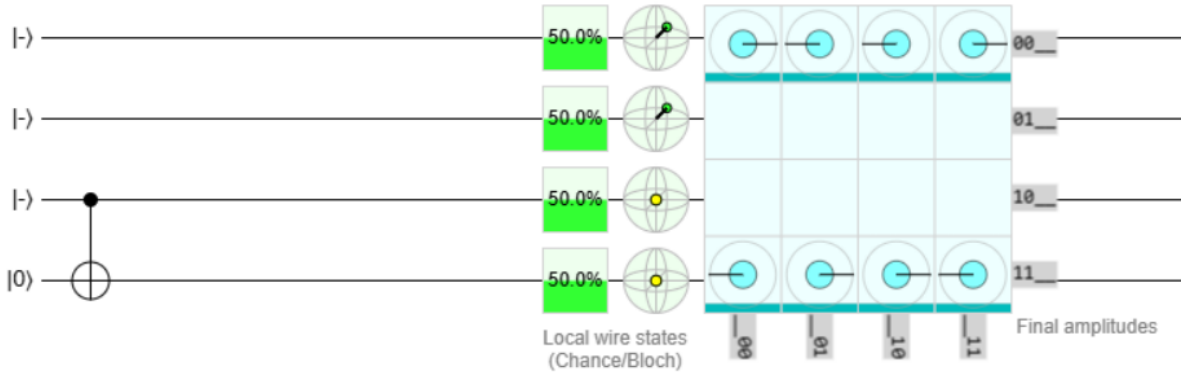
b. Haciendo uso de QUIRK, obtenemos el estado $|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle \otimes |0\rangle - |1\rangle \otimes |1\rangle) = \frac{1}{\sqrt{2}}(1, 0, 0, 1)$.



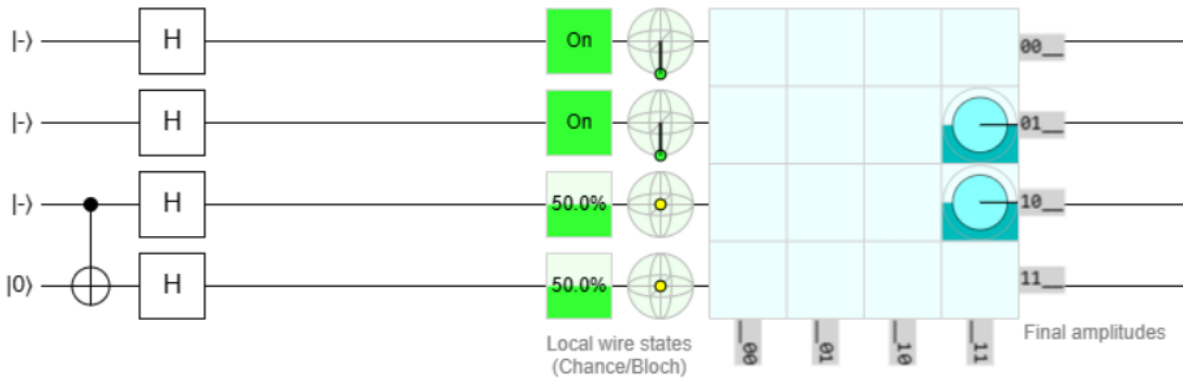
c. Parece que hemos clonado el primer qubit en el segundo, puesto que no tenemos coherencias, pero en realidad no hemos enviado el estado $|-\rangle$ del primer qubit al segundo, pues nótese que los estados de los apartados a) y b) no coinciden.

2.8. Ejercicio 4.3.2

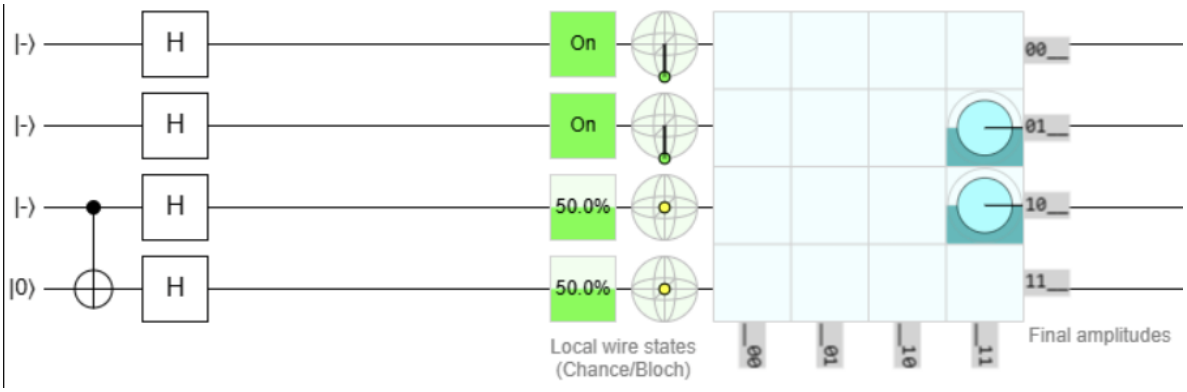
a. En la siguiente figura se puede observar que la probabilidad de encontrar $|1\rangle$ en cada uno de los qubits es $1/2$.



b. Las dos primeras son estados puros, mientras que las dos últimas son estados mezcla, luego se encuentran en el interior de la esfera de Bloch. De hecho, el vector que los representa es $\vec{r} = (0, 0, 0)$, y la matriz densidad asociada en el subsistema de un qubit es $\frac{1}{2}\mathbb{1}$, correspondiente al estado $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$.



c. Como se puede observar, el tercer qubit, correspondiente al primero del registro B, pasa del estado $|-\rangle$ al estado $|1\rangle$, que es lo esperable al pasar por la puerta H puesto que es su propia inversa. Así, se ve que la puerta $CNOT$ del registro B no ha afectado al qubit de control.



d. La matriz asociada al operador $H \otimes H$ es la siguiente:

$$H \otimes H = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \tag{2.3}$$

El estado que nos dan es $|\mathbf{\Phi}^-\rangle = \frac{1}{\sqrt{2}}(1,0,0,-1)^T$, luego la imagen es la siguiente:

$$H \otimes H \left|\mathbf{\Phi}^-\right\rangle = \frac{1}{2\sqrt{2}}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \\ 0 \\ -1 \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} \tag{2.4}$$

Así, se explica lo observado en la figura anterior.

2.9. Ejercicio 4.4.1

En las siguientes figuras se observa cómo construir los estados $|\mathbf{\Phi}^{\pm}\rangle$.

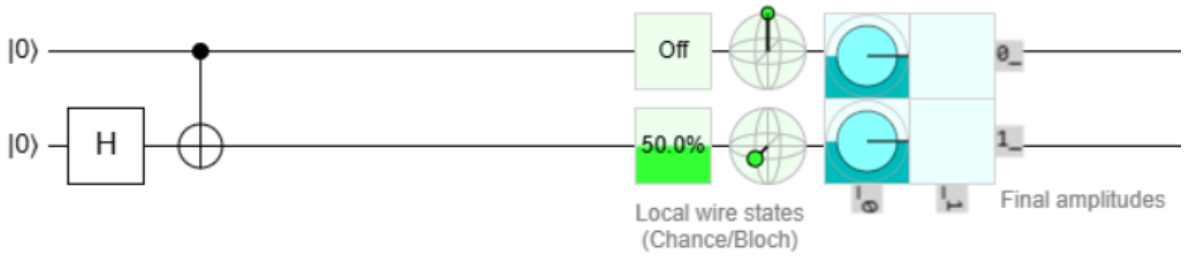


Figura 11: Preparación del estado $|\Phi^+\rangle$.

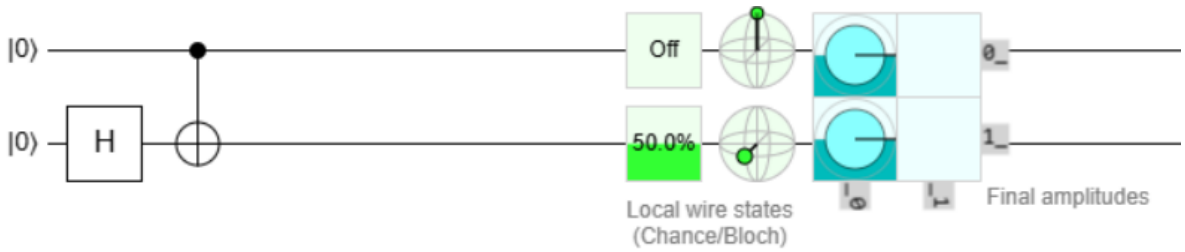


Figura 12: Preparación del estado $|\Phi^-\rangle$.

Así, solo tendríamos que cambiar el qubit 2 por el qubit 3:

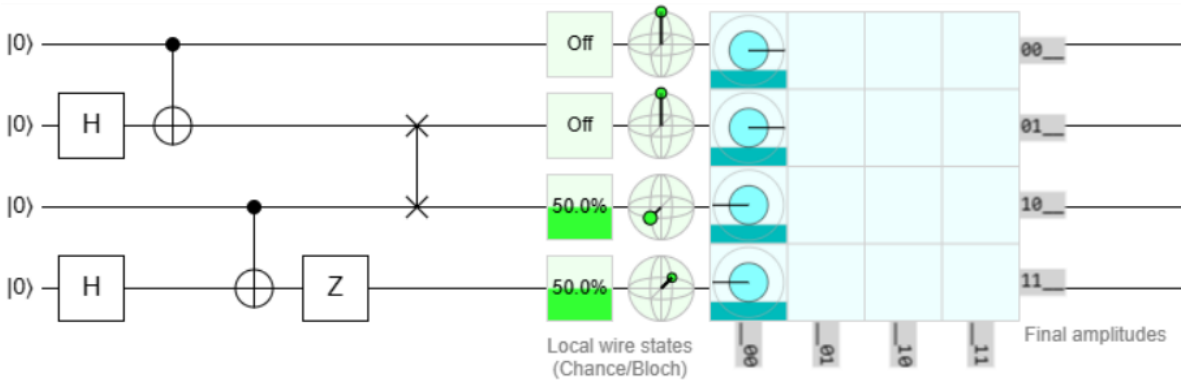
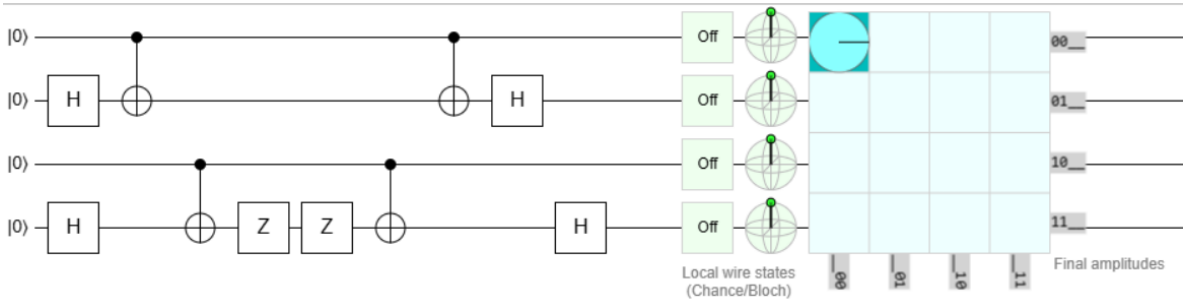


Figura 13: Preparación del estado pedido.

2.10. Ejercicio 4.3.2

Los qubits $(1, 3)$ y los qubits $(2, 4)$ están entrelazados a través de las puertas $CNOT$, que entrelazaron $(1, 2)$ y $(3, 4)$ y luego intercambiaron 2 y 3.

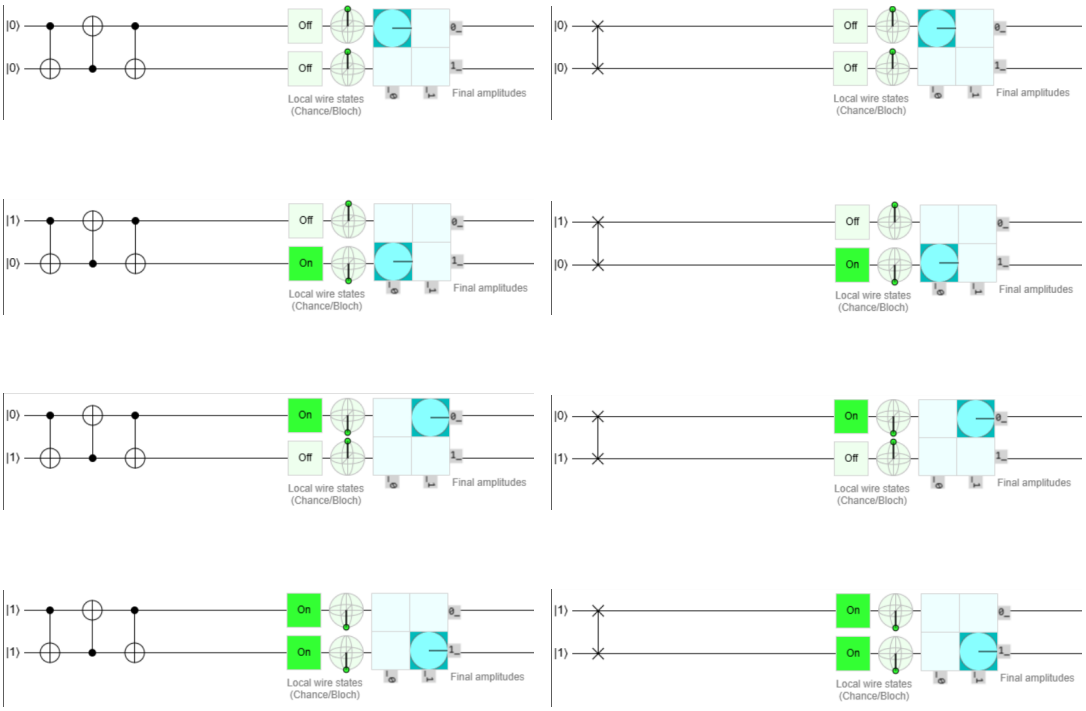
Si preparamos el circuito del enunciado obtenemos lo siguiente:



Esto era esperable, pues $(\mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \otimes Z)^2 = (\mathbb{1} \otimes \mathbb{1} \otimes (CNOT))^2 = ((CNOT) \otimes \mathbb{1} \otimes \mathbb{1})^2 = (\mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \otimes H)^2 = (\mathbb{1} \otimes H \otimes \mathbb{1} \otimes \mathbb{1})^2 = \mathbb{1}_4$, donde $\mathbb{1}_4 = \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1}$.

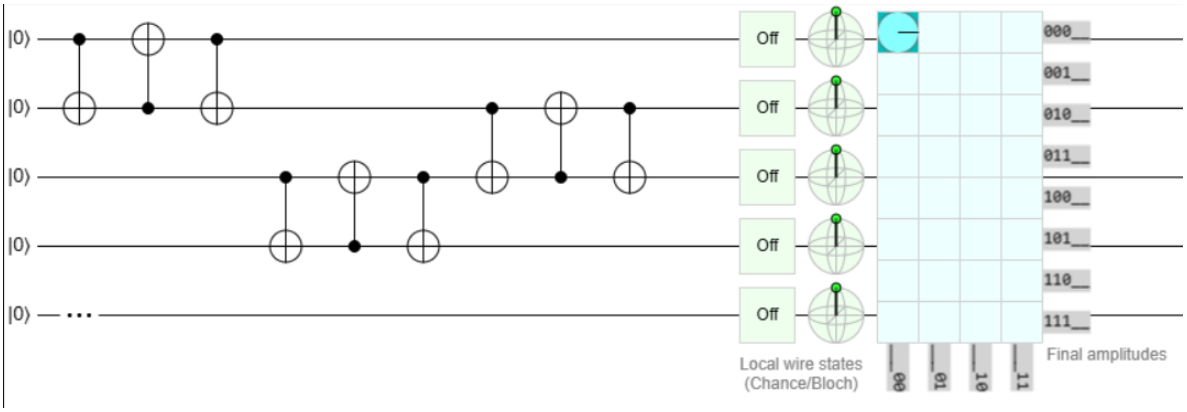
2.11. Ejercicio 4.4.3

En los siguientes pares de figuras se encuentran las salidas para la puerta $C_1NOT_2 \cdot C_2NOT_1 \cdot C_1NOT_2$ y $SWAP$, para comprobar que en efecto coinciden.



2.12. Ejercicio 4.4.4

La permutación $(1, 2, 3, 4, 5) \to (2, 4, 1, 3, 5)$ se puede implementar siguiendo los siguientes pasos: $(1, 2, 3, 4, 5) \to (2, 1, 3, 4, 5) \to (2, 1, 4, 3, 5) \to (2, 4, 1, 3, 5)$. En QUIRK lo escribimos como sigue:



3. Sesión 2: Introducción a la programación en Qiskit

En esta sección se resolverán las preguntas que se encuentran en el notebook de Jupyter.

3.1. ¿Por qué el histograma tiene una única barra correspondiente al ket $|1\rangle$?

Hemos partido de un estado de la base computacional, $|1\rangle$, luego al medir sobre esa misma base, evidentemente solo obtendremos la medida 1, asociada a ese estado (la amplitud de probabilidad de $|0\rangle$ es nula).

3.2. ¿Puedes escribir explícitamente el estado dado?

El estado se nos da según `init_state = np.kron([1,1],np.kron([1,1],np.kron([1,1],[1,1])))`. Por tanto, escribiendo $|\alpha\rangle \times |\beta\rangle \equiv |\alpha\beta\rangle$, el estado que obtenemos es:

$$\begin{aligned}(|0\rangle + |1\rangle) \times \{(|0\rangle + |1\rangle) \times [(|0\rangle + |1\rangle) \times (|0\rangle + |1\rangle)]\} = \\ (|0\rangle + |1\rangle) \times \{(|0\rangle + |1\rangle) \times [|00\rangle + |01\rangle + |10\rangle + |11\rangle]\}\end{aligned} \tag{3.1}$$

3.3. Ejecute el circuito varias veces y anote los resultados

Al correr la celda utilizando $N = 1$ un total de 5 veces, los resultados obtenidos (en binario) fueron: 1000, 1100, 0000, 0111, 0000.

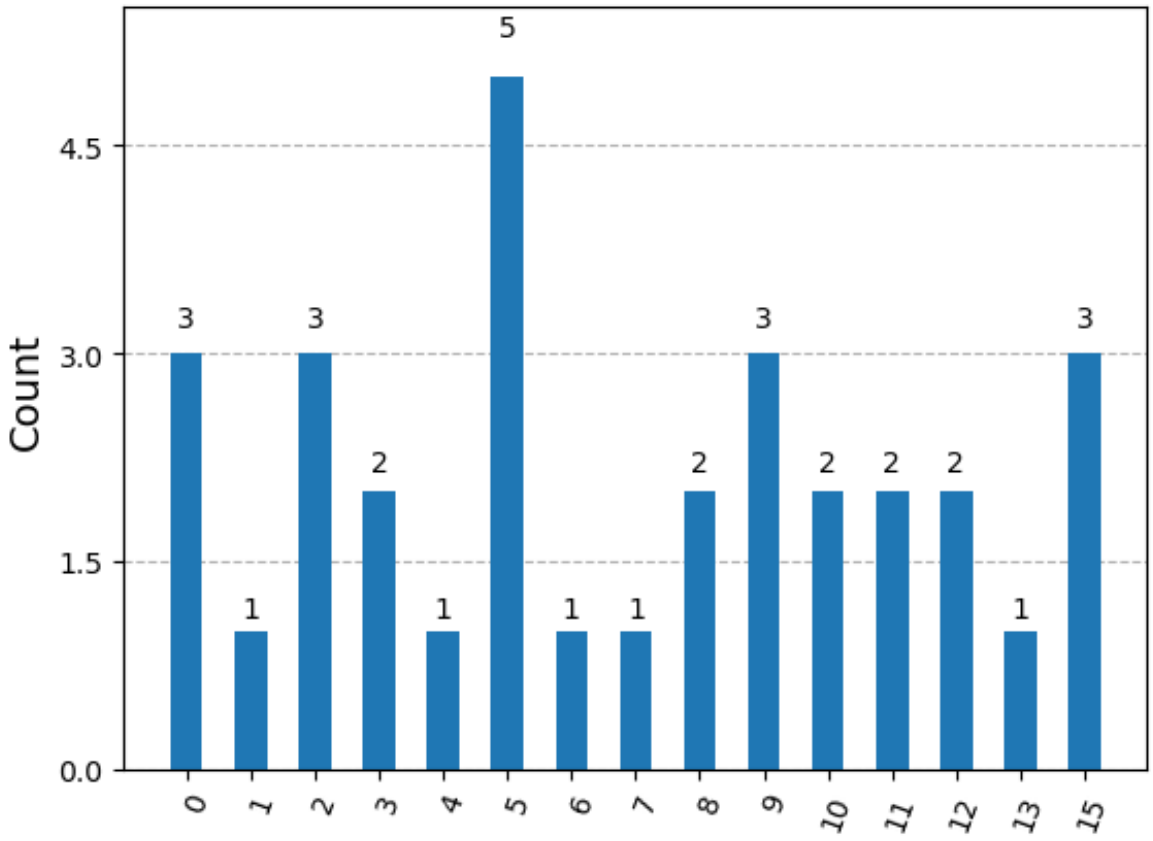
3.4. Escriba un programa que realice 32 medidas y genere un histograma a partir de los resultados

El programa utilizado es el siguiente:

```
    result=sim.run(qrng,shots=32).result()
rn_bin=result.get_counts(qrng)
```

```
rn_dec=\{int(key,2):value
        for key,value in rn_bin.items()\}
plot_histogram(rn_dec)
```

El resultado es el siguiente:



3.5. Escriba todos los métodos que ha aprendido

Hemos generado circuitos aleatorios mediante qubits inicializados con amplitudes de probabilidad iguales para todos los posibles estados, hemos almacenado las medidas en bits clásicos, y tras cambiar el valor binario a decimal, lo hemos dibujado en un histograma.

3.6. Compruebe que la función R_k en efecto implementa la transformación pedida

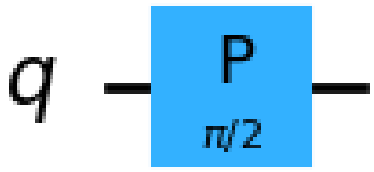
A partir de la definición de la función

```
from qiskit.circuit.library import PhaseGate, XGate
def R(k):
    return PhaseGate((2.*np.pi)/pow(2,k))
```

obtenemos con el código

```
    Example3 = QuantumCircuit(1)
Example3.append(R(2),[0])
Example3.draw(output='mpl')
```

la siguiente figura:



Ahora creamos el circuito, tomando el estado inicial $|1\rangle$ para poder observar el efecto:

```
qc4 = QuantumCircuit(1)
qc4.initialize([0,1], 0)
qc4.append(R(2),[0])
psi=Statevector(qc4)
psi.draw(output='latex')
```

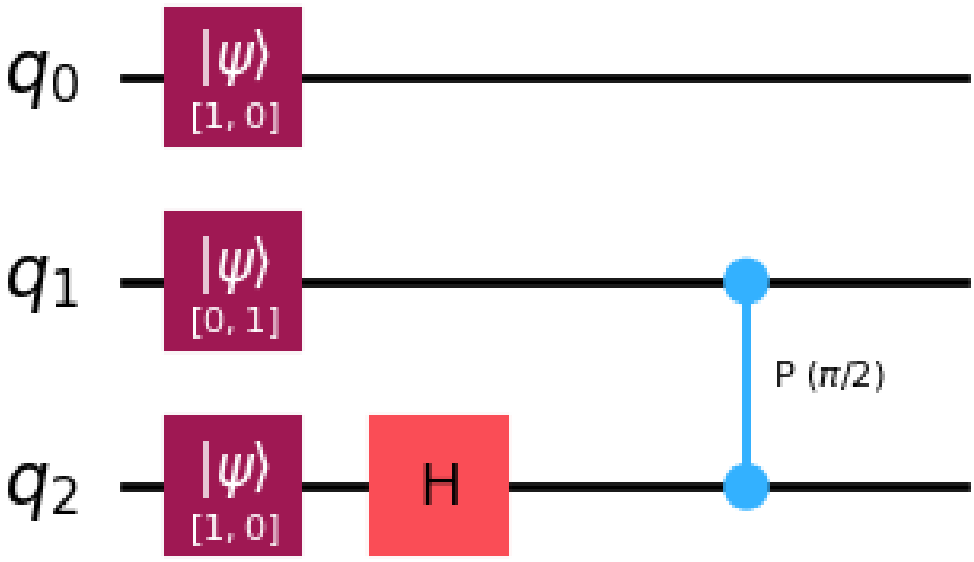
El resultado es $i|1>$.

3.7. Escriba el resultado de aplicar el circuito al $|010\rangle$

El código para el circuito es

```
qft_circuit = QuantumCircuit(3, name = 'QFT')
qft_circuit.initialize([1,0], 0)
qft_circuit.initialize([0,1], 1)
qft_circuit.initialize([1,0], 2)
qft_circuit.h(2)
qft_circuit.append(R(2).control(),[2,1])
qft_circuit.draw(output='mpl')
```

El dibujo es:



El codigo para el estado es

```
phi1 = Statevector(qft_circuit)
phi1
phi1.draw('latex')
```

El resultado es $\frac{1}{\sqrt{2}}|010> + \frac{i}{\sqrt{2}}|110>$.

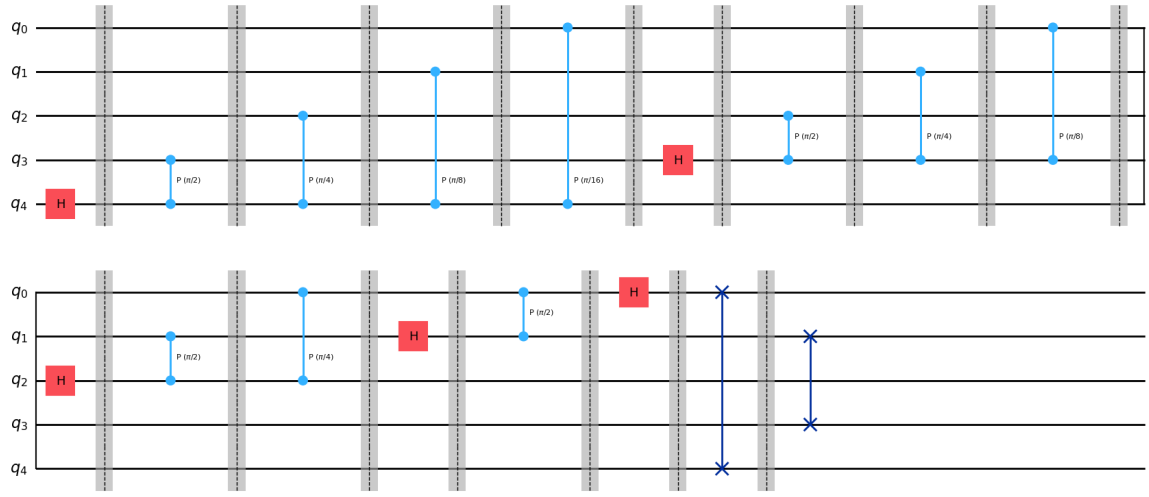
3.8. Implemente la transformada para 5 qubits sobre los siguientes estados

El código es

```
qft5 = QuantumCircuit(5, name = 'QFT5')
# Transforming qubit q_4:
qft5.h(4)
qft5.barrier()
qft5.append(R(2).control(),[4,3])
qft5.barrier()
qft5.append(R(3).control(),[4,2])
qft5.barrier()
qft5.append(R(4).control(),[4,1])
qft5.barrier()
qft5.append(R(5).control(),[4,0])
qft5.barrier()
```

```
# Transforming qubit q_3:
qft5.h(3)
qft5.barrier()
qft5.append(R(2).control(),[3,2])
qft5.barrier()
qft5.append(R(3).control(),[3,1])
qft5.barrier()
qft5.append(R(4).control(),[3,0])
qft5.barrier()
# Transforming qubit q_2:
qft5.h(2)
qft5.barrier()
qft5.append(R(2).control(),[2,1])
qft5.barrier()
qft5.append(R(3).control(),[2,0])
qft5.barrier()
# Transforming qubit q_2:
qft5.h(1)
qft5.barrier()
qft5.append(R(2).control(),[1,0])
qft5.barrier()
# Transforming qubit q_0:
qft5.h(0)
qft5.barrier()
# SWAP
qft5.swap(0,4)
qft5.barrier()
qft5.swap(1,3)
qft5.draw(output='mpl')
```

El dibujo es:



Las barreras se dibujan porque se desordenan las puertas en la figura, no por ello en el codigo, y de esta forma se colocan en su sitio (entre barreras no se desordenan). El código sin las barreras es el siguiente:

```
qft5 = QuantumCircuit(5, name = 'QFT5')
# Transforming qubit q_4:
qft5.h(4)
qft5.append(R(2).control(),[4,3])
qft5.append(R(3).control(),[4,2])
qft5.append(R(4).control(),[4,1])
qft5.append(R(5).control(),[4,0])
# Transforming qubit q_3:
qft5.h(3)
qft5.append(R(2).control(),[3,2])
qft5.append(R(3).control(),[3,1])
qft5.append(R(4).control(),[3,0])
# Transforming qubit q_2:
qft5.h(2)
qft5.append(R(2).control(),[2,1])
qft5.append(R(3).control(),[2,0])
# Transforming qubit q_2:
qft5.h(1)
qft5.append(R(2).control(),[1,0])
# Transforming qubit q_0:
qft5.h(0)
# SWAP
qft5.swap(0,4)
qft5.swap(1,3)
```

Con `qft5.gate=qft5.to_gate()` creamos la puerta. Debajo se muestran los códigos con sus respectivas salidas:

a.
```
qftcircuit=QuantumCircuit(5)
init_state=np.kron(np.kron(np.kron(np.kron([1,1],[1,1]),[1,1]),[1,1]),[1,-1])
init_state_normal = init_state/np.linalg.norm(init_state)
qftcircuit.initialize(init_state_normal,[0,1,2,3,4])
qftcircuit.append(qft5,[0,1,2,3,4])
PHI=Statevector(qftcircuit)
PHI.draw(output='latex')
```

   Salida: $|10000\rangle$.

b.
```
qftcircuit=QuantumCircuit(5)
init_state=np.kron(np.kron(np.kron(np.kron([1,1],[1,1]),[1,1]),[1,-1]),[1,1])
init_state_normal = init_state/np.linalg.norm(init_state)
qftcircuit.initialize(init_state_normal,[0,1,2,3,4])
qftcircuit.append(qft5,[0,1,2,3,4])
PHI=Statevector(qftcircuit)
PHI.draw(output='latex')
```

   Salida: $\frac{1+i}{2}|01000\rangle + \frac{1-i}{2}|11000\rangle$.

c.
```
qftcircuit=QuantumCircuit(5)
init_state=np.kron(np.kron(np.kron(np.kron([1,1],[1,1]),[1,-1]),[1,1]),[1,1])
init_state_normal = init_state/np.linalg.norm(init_state)
qftcircuit.initialize(init_state_normal,[0,1,2,3,4])
qftcircuit.append(qft5,[0,1,2,3,4])
```

```
PHI=Statevector(qftcircuit)
PHI.draw(output='latex')
```

Salida: $(0{,}25+0{,}60355339i)\,|00100\rangle+(0{,}25+0{,}10355339i)\,|01100\rangle+(0{,}25-0{,}10355339i)\,|00100\rangle+(0{,}25-0{,}60355339i)\,|11100\rangle$.

d.
```
qftcircuit=QuantumCircuit(5)
init_state=np.kron(np.kron(np.kron(np.kron([1,1],[1,-1]),[1,1]),[1,1]),[1,1])
init_state_normal = init_state/np.linalg.norm(init_state)
qftcircuit.initialize(init_state_normal,[0,1,2,3,4])
qftcircuit.append(qft5,[0,1,2,3,4])
PHI=Statevector(qftcircuit)
PHI.draw(output='latex')
```

Salida: $(0{,}125+0{,}62841744i)\,|00010\rangle+(0{,}125+0{,}18707572i)\,|00110\rangle+(0{,}125+0{,}08352233i)\,|01010\rangle+(0{,}125+0{,}02486405i)\,|01110\rangle+(0{,}125-0{,}02486405i)\,|10010\rangle+(0{,}125-0{,}18707572i)\,|11010\rangle+(0{,}125-0{,}18707572i)\,|11010\rangle+(0{,}125-0{,}62841744i)\,|11110\rangle$.

4. Sesión 3: Ejecución en los ordenadores de IBM

4.1. Números aleatorios

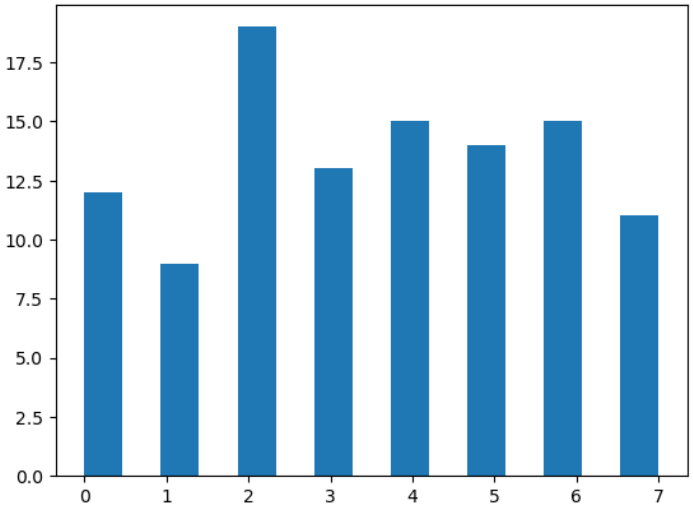
4.1.1. Definir el circuito

Usamos el siguiente código:

```
#How many random numbers will be produced in a single shot:
Nnumbers=33  #must be smaller than [127 qubits/3]=42
# Prepare the input circuit.
QRNG = QuantumCircuit(3*Nnumbers) #four qubits
for j in range(Nnumbers):
  QRNG.h([(3*j+0),(3*j+1),(3*j+2)]) #apply Hadamard gate to each of the qubits

QRNG.measure_all() #measure all qubits
#QRNG.draw(output="mpl") #plot circuit: don't do it with many qubits
```

La última linea genera el dibujo del circuito, pero para 33 números son 99 qubits y es demasiado grande como para incluirlo aquí. Una vez realizado el ejercicio, el resultado obtenido es el siguiente histograma:



4.1.2. Comente los resultados

Hemos obtenido un histograma donde la frecuencia relativa de cada número es aproximadamente igual.

4.2. Algoritmo DEUTSCH

4.2.1. Unitariedad

Para comprobarlo usaremos el mismo razonamiento que en el ejercicio 3.1 de la primera práctica. Veamos que $\langle x, y|x', y'\rangle = \langle x, y \oplus f(x)|x', y' \oplus f(x')\rangle$. Como estamos en el espacio producto, tenemos lo siguiente:

$$\begin{aligned} \langle x, y|x', y'\rangle &= \langle x|x'\rangle \langle y|y'\rangle \\ \langle x, y \oplus f(x)|x', y' \oplus f(x')\rangle &= \langle x|x'\rangle \langle y \oplus f(x)|y' \oplus f(x')\rangle \end{aligned} \tag{4.1}$$

Nótese que si $x \neq x'$ módulo 2, entonces ambos productos escalares son automáticamente cero. Así en ese caso coinciden. Si ahora tomamos $x = x'$, con lo que $\langle x|x'\rangle = 1$, se tiene que $f(x) = f(x')$, y por tanto $\langle y \oplus f(x)|y' \oplus f(x')\rangle = \langle y \oplus f(x)|y' \oplus f(x)\rangle$.

Si ahora $y \neq y'$ módulo 2, entonces $y \oplus f(x) \neq y' \oplus f(x)$ módulo 2 y por tanto se vuelven a anular ambos productos. Sin embargo, si $y = y'$ módulo 2, entonces $\langle y \oplus f(x)|y' \oplus f(x)\rangle = \langle y \oplus f(x)|y \oplus f(x)\rangle = 1 = \langle y|y\rangle = \langle y|y'\rangle$, luego vuelven a coincidir.

Por tanto, concluimos que en efecto $\langle x, y|x', y'\rangle = \langle x, y \oplus f(x)|x', y' \oplus f(x')\rangle = \langle x, y| U^\dagger U |x', y'\rangle$ sean cuales sean los valores de las variables, y por tanto $U^\dagger U = \mathbb{1}$, luego U (el operador en cuestión) es unitario.

4.2.2. Completar celdas

Los códigos son los siguientes:

```
#Classical check Oracle_f: f(0)
Check_f0 = QuantumCircuit(2,1)
#Remember that by default the two qubits are initialized to zero
Check_f0.unitary(Oracle_f, [0, 1],label="Oracle_f")  # Apply Oracle_f
# Measure qubit 1
Check_f0.measure(1,0)
Check_f0.draw(output='mpl')
##Classical check Oracle_f: f(1)
Check_f1 = QuantumCircuit(2,1)
#We initialize the first qubit to 1:
Check_f1.initialize([0,1],0)
Check_f1.unitary(Oracle_f, [0, 1],label="Oracle_f")  # Apply Oracle_f
# Measure qubit 1
Check_f1.measure(1,0)
Check_f1.draw(output='mpl')
# One shot gor each circuit
shots_C=1
# Sampler with aer simulator as backend
sampler_S = Sampler(mode=sim);
# The sampler evaluates probabilities of each input circuit
jobs_f = sampler_S.run([Check_f0,Check_f1],shots=shots_C).result();
result_f_0 = jobs_f[0].data.c.get_counts() # result of Check_f0 circuit
result_f_1 = jobs_f[1].data.c.get_counts() # result of Check_f1 circuit
#Print the measurement results
print(f"f(0)={max(result_f_0)}")
print(f"f(1)={max(result_f_1)}")
#Classical check Oracle_g: g(0)
Check_g0 = QuantumCircuit(2,1)
Check_g0.unitary(Oracle_g, [0, 1],label="Oracle_g")  # Apply Oracle_g
# Measure qubit 1
Check_g0.measure(1,0)
Check_g0.draw(output='mpl')
##Classical check Oracle_g: g(1)
Check_g1 = QuantumCircuit(2,1)
#We initialize the first qubit to 1:
Check_g1.initialize([0,1],0)
Check_g1.unitary(Oracle_g, [0, 1],label="Oracle_g")  # Apply Oracle_g
# Measure qubit 1
Check_g1.measure(1,0)
Check_g1.draw(output='mpl')
jobs_g = sampler_S.run([Check_g0,Check_g1],shots=shots_S);
result_g_0 = jobs_g.result()[0].data.c.get_counts()
result_g_1 = jobs_g.result()[1].data.c.get_counts()
#Print the measurement results
print(f"g(0)={max(result_g_0)}")
print(f"g(1)={max(result_g_1)}")
```

4.2.3. Resultado de medir en f

El resultado de f resulta ser `f(0)=1, f(1)=0`. En efecto, si tomamos el estado inicial $|00\rangle$ y le hacemos pasar por el operador obtenemos lo siguiente:

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = |01\rangle \tag{4.2}$$

Así, metiendo en el primer qubit el valor 0, en el segundo obtenemos el valor 1, coincidiendo con el resultado obtenido por el código. Si ahora tomamos el estado inicial $|10\rangle$ y le hacemos pasar por el operador obtenemos lo siguiente:

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = |10\rangle \tag{4.3}$$

Así, metiendo en el primer qubit el valor 1, en el segundo obtenemos el valor 0, coincidiendo con el resultado obtenido por el código.

4.2.4. Resultado de medir en g

El resultado de g resulta ser `g(0)=1, g(1)=1`. En efecto, si tomamos el estado inicial $|00\rangle$ y le hacemos pasar por el operador obtenemos lo siguiente:

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = |01\rangle \tag{4.4}$$

Así, metiendo en el primer qubit el valor 0, en el segundo obtenemos el valor 1, coincidiendo con el resultado obtenido por el código. Si ahora tomamos el estado inicial $|10\rangle$ y le hacemos pasar por el operador obtenemos lo siguiente:

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = |11\rangle \tag{4.5}$$

Así, metiendo en el primer qubit el valor 1, en el segundo obtenemos el valor 1, coincidiendo con el resultado obtenido por el código. Nótese que estamos entendiendo f y g como funciones del primer qubit con imagen en el segundo, es decir, aunque en el espacio producto los operadores asociados no son constantes, las salidas (segundo qubit) en el caso del operador g sí que lo son.

4.2.5. Estado final

Si $f(0) \neq f(1)$, como estamos en congruencias módulo 2, entonces $1 \oplus f(0) = f(1)$ y $1 \oplus f(1) = f(0)$. Así, tomando la imagen por el operador que nos da el enunciado obtenemos lo siguiente:

$$\begin{aligned}
&\frac{1}{2}\left[|0\rangle|f(0)\rangle - |0\rangle|1 \oplus f(0)\rangle + |1\rangle|f(1)\rangle - |1\rangle|1 \oplus f(1)\rangle\right] = \\
&= \frac{1}{2}\left[|0\rangle|f(0)\rangle - |0\rangle|f(1)\rangle + |1\rangle|f(1)\rangle - |1\rangle|f(0)\rangle\right] = \\
&= \frac{1}{2}\left[|0\rangle\left(|f(0)\rangle - |f(1)\rangle\right) + |1\rangle\left(|f(1)\rangle - |f(0)\rangle\right)\right] = \\
&= \frac{1}{2}\left(|0\rangle - |1\rangle\right)\left(|f(0)\rangle - |f(1)\rangle\right) = \\
&= \frac{|0\rangle - |1\rangle}{\sqrt{2}}\frac{|f(0)\rangle - |f(1)\rangle}{\sqrt{2}} = |-\rangle\frac{|f(0)\rangle - |f(1)\rangle}{\sqrt{2}}
\end{aligned} \tag{4.6}$$

4.2.6. Circuito para el algoritmo Deutsch

El código es el siguiente:

```
# Deutsch's for f
deutsch_f = QuantumCircuit(2,1)
deutsch_f.x(1)  # Apply X gate to qubit 0
deutsch_f.h([0, 1])  # Apply Hadamard gate to both qubits
deutsch_f.unitary(Oracle_f, [0, 1],label="Oracle_f")  # Apply Oracle_f
deutsch_f.h(0)  # Apply Hadamard gate to qubit 0
# Measure qubit 0
deutsch_f.measure(0,0)
deutsch_f.draw(output='mpl')
# Deutsch's for g
deutsch_g = QuantumCircuit(2,1)
deutsch_g.x(1)  # Apply X gate to qubit 0
deutsch_g.h([0, 1])  # Apply Hadamard gate to both qubits
deutsch_g.unitary(Oracle_g, [0, 1],label="Oracle_g")  # Apply Oracle_g
deutsch_g.h(0)  # Apply Hadamard gate to qubit 0
# Measure qubit 0
deutsch_g.measure(0,0)
deutsch_g.draw(output='mpl')
# Execute the circuit
jobs_DeutschS=sampler_S.run([deutsch_f,deutsch_g],shots=shots_S).result()
result_DeutschS_f = jobs_DeutschS[0].data.c.get_counts()
result_DeutschS_g = jobs_DeutschS[1].data.c.get_counts()
# Print the measurement results for f, if 0 the fuction is constant
if max(result_DeutschS_f)=='0':
    print("f is constant")
else:
    print("f is not constant")
# Print the measurement results for g, if 0 the fuction is constant
if max(result_DeutschS_g)=='0':
    print("g is constant")
else:
```

```
    print("g is not constant")
```

Cuyas salidas son:

```
f is not constant
g is constant
```

4.2.7. Comparación

Para el circuito cuántico tenemos el código:

```
shots_Q=3
sampler_Q = Sampler(mode=backend_Q) #to obtain the sampling of the measurement
# Execute the circuit
job_Q=sampler_Q.run([transpile(deutsch_f,backend_Q),transpile(deutsch_g,backend_Q)],
shots=shots_Q)
```

Con salidas:

```
f is not constant
g is constant
```

Efectivamente, coinciden.

4.3. CHSH

4.3.1. Preparar el circuito

Preparamos el circuito con el siguiente código:

```
# Prepare the input circuit:
chsh_circuit = QuantumCircuit(2)
chsh_circuit.h(0)
chsh_circuit.cx(0,1)
chsh_circuit.z(0)
chsh_circuit.x(0)
chsh_circuit.draw(output='mpl')
phi=Statevector(chsh_circuit)
phi.draw(output='latex')
```

Para generar el estado de Bell hacemos uso de lo obtenido en la primera sección de esta práctica, correspondiente al uso de QUIRK. La última línea nos dibuja el estado, que en efecto es:

$$\frac{\sqrt{2}}{2}|01\rangle - \frac{\sqrt{2}}{2}|10\rangle \tag{4.7}$$

4.3.2. Definir los productos de observables

Los definimos con el siguiente código:

```
# Define pairs of observables for maximum violation of the CHSH inequality
A1B1=SparsePauliOp.from_list([('XX', -1/sqrt(2)), ('XY', -1/sqrt(2))])
A1B2=SparsePauliOp.from_list([('XX', -1/sqrt(2)), ('XY', 1/sqrt(2))])
A2B1=SparsePauliOp.from_list([('YX', -1/sqrt(2)), ('YY', -1/sqrt(2))])
A2B2=SparsePauliOp.from_list([('YX', -1/sqrt(2)), ('YY', 1/sqrt(2))])
Obs=[A1B1,A1B2,A2B1,A2B2] #The four correlators are packaged in one list of observables
```

4.3.3. Cálculo de la desigualdad

Los valores esperados para `[A1B1,A1B2,A2B1,A2B2]` fueron `[ 0.707452  0.7067615 0.6991656 -0.7150479]`, y las desviaciones típicas fueron `[0.011, 0.011, 0.011, 0.011047847]`. Usamos el siguiente código para calcular y mostrar el valor esperado de la suma de los productos de observables según indica la desigualdad CHSH, así como su incertidumbre:

```
CHSH_mean=result_S.data.evs[0]+result_S.data.evs[1]+result_S.data.evs[2]-result_S.data.evs[3]
CHSH_uncertainty=sqrt(Standard_errors_S[0]**2+Standard_errors_S[1]**2+
Standard_errors_S[2]**2+Standard_errors_S[3]**2)
print(f"The simulated result is {CHSH_mean} + {CHSH_uncertainty}:
\ndoes it exceed 2 with sufficient statistical certainty?")
```

La respuesta es la siguiente:

```
The simulated result is 2.82842712474619 + 0.022023962480339658:
does it exceed 2 with sufficient statistical certainty?
```

Concluimos que en efecto se viola la desigualdad de Bell.

4.3.4. Cálculo en el caso cuántico

Para el circuito cuántico los valores esperados fueron `[ 0.7092474  0.7049662 0.7056797 -0.6985444]`, y las desviaciones típicas fueron `[0.013, 0.013, 0.012, 0.012363197]`, dando lugar a la siguiente salida:

```
The simulated result is 2.8184377252137867 + 0.025196202889737354:
does it exceed 2 with sufficient statistical certainty?
```

Una vez más, se viola la desigualdad de Bell.

5. Anexos

5.1. Notebooks

Aquí se adjuntan los notebooks de Jupyter obtenidos. De la sección 2 se adjunta un .html en vez de un notebook, debido a un error del autor que no descargó correctamente el archivo. No obstante, se encuentran en él todas las respuestas, y el final del script que no se incluye en el archivo está incluido en este mismo documento en su sección correspondiente.