Deutsch's Algorithm

Ioannis Savvaidis

General Analysis of Deutsch's Algorithm

Deutch's Problem explores determining whether a function $f(x)$ with $x \in \{0, 1\}$ is constant or balanced. Table 1 shows the four possible versions of this function.

Table 1: The four versions of $f(x)$

x	f_0
0	0
1	0

x	f_1
0	1
1	1

x	f_2
0	0
1	1

x	f_3
0	1
1	0

Based on the table we can determine that $f_0()$ and $f_1()$ are constant while $f_2()$ and $f_3()$ are balanced functions since:
$f_0(0) = f_0(1)$
$f_1(0) = f_1(1)$
$f_2(0) \neq f_2(1)$
$f_3(0) \neq f_3(1)$

We can model the problem with binary by checking the output of $f(x)$ for $f(0)f(1)$:
$f_0 \rightarrow 00$
$f_1 \rightarrow 11$
$f_2 \rightarrow 01$
$f_3 \rightarrow 10$

From the output bits, an XOR pattern appears, and we can use the binary addition to distinguish them from their outputs $f(0) \oplus f(1)$:
$f_0(0) \oplus f_0(1) = 0$
$f_1(0) \oplus f_1(1) = 0$
$f_2(0) \oplus f_2(1) = 1$
$f_3(0) \oplus f_3(1) = 1$

Thus, when $f(0) = f(1) = 0, f$ is constant and when $f(0) \neq f(1) = 1, f$ is balanced. In classical computers, we need to query $f(x)$ twice for 0 and 1, but in quantum, we can query $f(x)$ one time for both values of x.
From the Table 1, we can transpose the ket of row into the bra of each column, constructing the unary single-qubit gate M of an input $|x\rangle$:

$$M_{f_0} = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} \quad M_{f_1} = \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix} \quad M_{f_2} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad M_{f3} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Figure 1: M Gate

But, the first two matrices, M, are not unitary since, $M_{f0}^{\dagger} M_{f0} \neq I$ and $M_{f1}^{\dagger} M_{f1} \neq I$

We need another qubit to generate a Unitary Matrix for the Oracle with all f functions. By adding another qubit from the CNOT pattern, we can have the first qubit as input and the second as output of f(x).

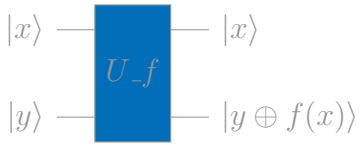


Figure 2: Unitary Gate

The binary addition on the second qubit, acting as an output, makes the oracle unitary and reversible since:

$U_f|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle$
$(U_f)(U_f)|x\rangle|y\rangle = (U_f)|x\rangle|y \oplus f(x)\rangle = |x\rangle|y \oplus f(x) \oplus f(x)\rangle = |x\rangle|y\rangle$

To find the oracle, we need to build the transformation matrix U_f based on the outcome of the oracle of each function:

Table 2: The truth table for all $f(x)$

$\|x\rangle\|y\rangle$	$\|x\rangle\|y \oplus f_0(x)\rangle$	$\|x\rangle\|y\rangle$	$\|x\rangle\|y \oplus f_1(x)\rangle$	$\|x\rangle\|y\rangle$	$\|x\rangle\|y \oplus f_2(x)\rangle$	$\|x\rangle\|y\rangle$	$\|x\rangle\|y \oplus f_3(x)\rangle$
00	00	00	01	00	00	00	01
01	01	01	00	01	01	01	00
10	10	10	11	10	11	10	10
11	11	11	10	11	10	11	11

From the Table 2, we can transpose the ket of row into the bra of each column, constructing the Oracle's matrix:

$$U_{f_0} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} \quad U_{f_1} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} X & 0 \\ 0 & X \end{bmatrix} \quad U_{f_2} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & X \end{bmatrix} \quad U_{f3} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} X & 0 \\ 0 & I \end{bmatrix}$$

In formulating the problem, we can use 0 as the first part of the binary addition since it will leave the second part un-flipped (XOR). We need to execute the circuit classically twice, $x \in \{0, 1\}$.

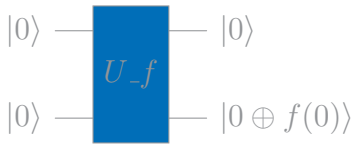


Figure 3: f(0)

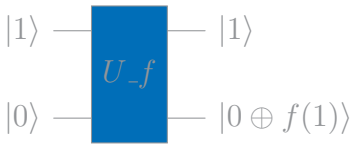


Figure 4: f(1)

Using Quantum Parallelism in the target qubit, which queries the f function, we can calculate both steps in one:

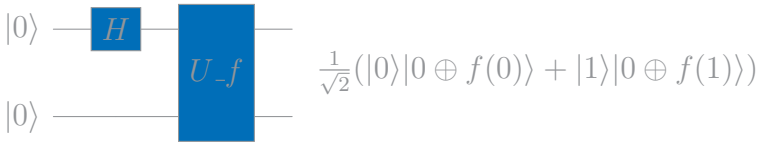


Figure 5: f(0) and f(1)

But we cannot derive the tensored state from the individual qubits states and the state will collapse to one of the equally distributed states. So we cannot know the f values.

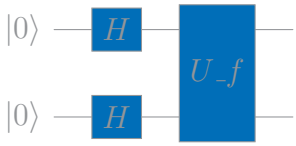


Figure 6

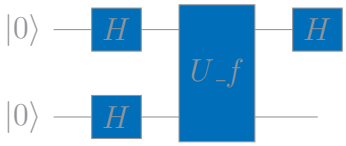


Figure 7

The circuits in Fig. 6 and Fig. 7, are not helpful in calculating the problem since the final states are :

$$\frac{1}{\sqrt{2}}\Big(|0\rangle(|f(0)\rangle + |f(1)\rangle) + |1\rangle(|f(0)\rangle - |f(1)\rangle)$$

and:

$$H|+\rangle|+\rangle$$

Deutch's Algorithm simultaneously queries the function $f(x)$ for both possible values using quantum parallelism and interference, with phase-kickback to encode the function result in qubit phase, initiating the target qubit in $|1\rangle$ basis state:

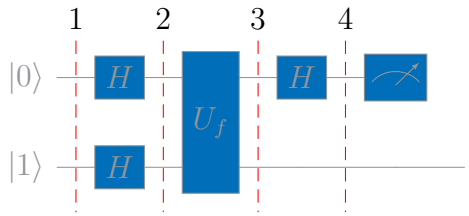


Figure 8: Duetch's Algorithm with Oracle

The circuit in operator form is: $(H \otimes I)U_f(H \otimes H)|0\rangle|1\rangle$
The oracle U_f transformation is: $|x\rangle|y\rangle \xrightarrow{U_f} |x\rangle|y \oplus f(x)\rangle$.

$$|q_2\rangle = (H \otimes H)|0\rangle|1\rangle = \frac{1}{2}(|0\rangle|0\rangle - |0\rangle|1\rangle + |1\rangle|0\rangle - |1\rangle|1\rangle = |+\rangle|-\rangle$$

$$\begin{aligned}
&|q_3\rangle = U_f|q_2\rangle \\
&= \frac{1}{2}\Big(|0\rangle|0 \oplus f(0)\rangle - |0\rangle|1 \oplus f(0)\rangle + |1\rangle|0 \oplus f(1)\rangle - |1\rangle|1 \oplus f(1)\rangle\Big) \\
&= \frac{1}{2}\Big[|0\rangle(|0 \oplus f(0)\rangle - |1 \oplus f(0)\rangle) + |1\rangle(|0 \oplus f(1)\rangle - |1 \oplus f(1)\rangle)\Big] \qquad (1)
\end{aligned}$$

Both parts have a common factor, which can be abstracted:

$$|0 \oplus f(x)\rangle - |1 \oplus f(x) = (-1)^{f(x)}(|0\rangle - |1\rangle) \qquad (2)$$

since:

$$|0 \oplus f(x)\rangle - |1 \oplus f(x) \xrightarrow{f(x)=0} |0\rangle - |1\rangle = 1(|0\rangle - |1\rangle) = (-1)^0(|0\rangle - |1\rangle)$$

$$|0 \oplus f(x)\rangle - |1 \oplus f(x) \xrightarrow{f(x)=1} |1\rangle - |0\rangle = -1(|0\rangle - |1\rangle) = (-1)^1(|0\rangle - |1\rangle)$$

Applying (2) in (1):

$$\begin{aligned}
&|q_3\rangle = \frac{1}{2}\Big[|0\rangle(-1)^{f(0)}(|0\rangle - |1\rangle) + |1\rangle(-1)^{f(1)}(|0\rangle - |1\rangle)\Big] \\
&= \frac{1}{2}\Big[(-1)^{f(0)}|0\rangle(|0\rangle - |1\rangle) + (-1)^{f(1)}|1\rangle(|0\rangle - |1\rangle)\Big]
\end{aligned}$$

Factoring out the common tensor product factor $(|0\rangle - |1\rangle)$*:*

$$|q_3\rangle = \frac{1}{2}\Big[(-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle\Big] \otimes \Big[(|0\rangle - |1\rangle)\Big]$$

The second term looks like $|-\rangle$*, we can decompose* $\frac{1}{2}$ *and use tensor product properties to:*

$$|q_3\rangle = \frac{1}{\sqrt{2}}\Big[(-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle\Big] \otimes \frac{1}{\sqrt{2}}\Big[(|0\rangle - |1\rangle)\Big] = \frac{1}{\sqrt{2}}\Big[(-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle\Big] \otimes |-\rangle$$

Both qubits are in superposition, the qubit $|1\rangle$ remained in $|-\rangle$, but the other qubit changed its phase from $|+\rangle$, to the one mentioned above. While the f started in the second qubit, qubits interfered, and the second kicked back the phase of the first qubit. This phase encodes the calculation of functions during superposition. Since the circuits measure only the $|x\rangle$ qubit, we will omit the tensor product:

$$|q_4\rangle = (H \otimes I)|q_3\rangle = (H \otimes I)\Big(\frac{1}{\sqrt{2}}\Big[(-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle\Big] \otimes |-\rangle\Big) \xrightarrow{omit}$$
$$H\Big(\frac{1}{\sqrt{2}}\Big[(-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle\Big]\Big) = \frac{1}{\sqrt{2}}\Big[(-1)^{f(0)}H|0\rangle + (-1)^{f(1)}H|1\rangle\Big] \tag{3}$$

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = |+\rangle \tag{4}$$
$$H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = |-\rangle \tag{5}$$

Applying (4) and (5) in (3):

$$|q_4\rangle = \frac{1}{2}\Big[(-1)^{f(0)}(|0\rangle + |1\rangle) + (-1)^{f(1)}(|0\rangle - |1\rangle)\Big]$$
$$= \frac{1}{2}\Big[(-1)^{f(0)}|0\rangle + (-1)^{f(0)}|1\rangle + (-1)^{f(1)}|0\rangle - (-1)^{f(1)}(|1\rangle\Big] =$$
$$= \frac{1}{2}\Big[[(-1)^{f(0)} + (-1)^{f(1)}]|0\rangle + [(-1)^{f(0)} - (-1)^{f(1)}]|1\rangle\Big]$$
$$= \frac{1}{2}\Big((-1)^{f(0)} + (-1)^{f(1)})\Big)|0\rangle + \frac{1}{2}\Big((-1)^{f(0)} - (-1)^{f(1)}\Big)|1\rangle$$

Now, we can extend this further using the identity of powers:

$$1 = (-1)^{f(x)} * (-1)^{-f(x)} = (-1)^{f(x)-f(x)} = (-1)^{f(x)\oplus f(x)} = (-1)^0 = 1.$$

Since the common part is $(-1)^{f(0)}$, we will factor our this term:

$$|q_4\rangle = \frac{1}{2}\Big((-1)^{f(0)} + (-1)^{f(1)}(-1)^{f(0)}(-1)^{-f(0)})\Big)|0\rangle + \frac{1}{2}\Big((-1)^{f(0)} - (-1)^{f(1)}(-1)^{f(0)}(-1)^{-f(0)}\Big)|1\rangle$$
$$= \frac{(-1)^{f(0)}}{2}(1 + (-1)^{f(1)-f(0)})|0\rangle + \frac{(-1)^{f(0)}}{2}(1 - (-1)^{f(1)-f(0)})|1\rangle$$
$$= \frac{(-1)^{f(0)}}{2}\Big[(1 + (-1)^{f(1)-f(0)})|0\rangle + (1 - (-1)^{f(1)-f(0)})|1\rangle\Big]$$
$$= \frac{(-1)^{f(0)}}{2}\Big[(1 + (-1)^{f(1)\oplus f(0)})|0\rangle + (1 - (-1)^{f(1)\oplus f(0)})|1\rangle\Big]$$
$$= \frac{(-1)^{f(0)}}{2}\Big[(1 + (-1)^{f(1)\oplus f(0)})|0\rangle\Big] + \frac{(-1)^{f(0)}}{2}\Big[(1 - (-1)^{f(1)\oplus f(0)})|1\rangle\Big] = \alpha|0\rangle + \beta|1\rangle$$

Since we have the probability amplitudes, we can calculate the final states.

For $f(0) = f(1)$:

- $(-1)^{f(1)\oplus f(0)} = (-1)^0 = 1$
- $\alpha = (-1)^{f(0)}$ and $\beta = 0$
- $|q_4\rangle = (-1)^{f(0)}|0\rangle$

For $f(0) \neq f(1) \rightarrow (-1)^{f(1)\oplus f(0)} = -1$:

- $(-1)^{0\oplus 1} = -1$ and $(-1)^{1\oplus 0} = -1 \rightarrow (-1)^{f(1)\oplus f(0)} = -1$
- $\alpha = 0$ and $\beta = (-1)^{f(0)}$
- $|q_4\rangle = (-1)^{f(0)}|1\rangle$

Based on $|q_4\rangle$, $f(0)$ the exponent $f(1) \oplus f(0)$ dictates the probability amplitude of each basis state, giving the needed XOR bit, and the exponent $f(0)$ dictates the control qubit's phase, adding a layer of further binary encoding for both function of each category.

Table 3: Final States

Function	Final State	Phase
f_0	$\|q_{final}\rangle = \|0\rangle$	0
f_1	$\|q_{final}\rangle = -\|0\rangle$	π
f_2	$\|q_{final}\rangle = \|1\rangle$	0
f_3	$\|q_{final}\rangle = -\|1\rangle$	π

I. Case: U_{f_1}

Calculate the circuit

$$|q_2\rangle \ = |+\rangle|-\rangle = \frac{1}{2}(|0\rangle|0\rangle - |0\rangle|1\rangle + |1\rangle|0\rangle - |1\rangle|1\rangle$$

$$|q_3\rangle = U_{f_1}|q_2\rangle = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{2} \\ -\frac{1}{2} \\ \frac{1}{2} \\ -\frac{1}{2} \end{bmatrix} = \begin{bmatrix} -\frac{1}{2} \\ \frac{1}{2} \\ -\frac{1}{2} \\ \frac{1}{2} \end{bmatrix} = \frac{1}{2}(-|0\rangle|0\rangle + |0\rangle|1\rangle - |1\rangle|0\rangle + |1\rangle|1\rangle$$

$$|q_4\rangle = (H \otimes I)|q_3\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \frac{1}{2} \begin{bmatrix} -1 \\ 1 \\ -1 \\ 1 \end{bmatrix} = \frac{1}{2\sqrt{2}} \begin{bmatrix} -2 \\ 2 \\ 0 \\ 0 \end{bmatrix}$$

$$= \frac{1}{\sqrt{2}}(-|0\rangle|0\rangle + |0\rangle|1\rangle) \tag{6}$$

$$= \frac{1}{\sqrt{2}}[-|0\rangle \otimes (|0\rangle - |1\rangle)] = [-|0\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)] = -|0\rangle|-\rangle \tag{7}$$

In equation (7) we can verify the final state with the previous calculations from Table 3.

Executing the circuit in QCS

Figure 9 depicts the Oracle of Deutch's Algorithm, using an X Gate. Figure 10 depicts the Quantum Circuit in QCS Environment, by using 2 qubits and 5 steps.

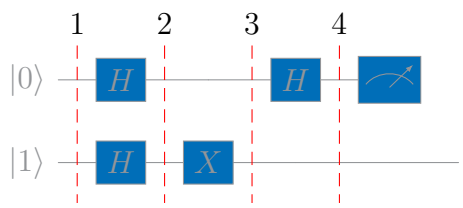


Figure 9: Duetch's Algorithm for f_1

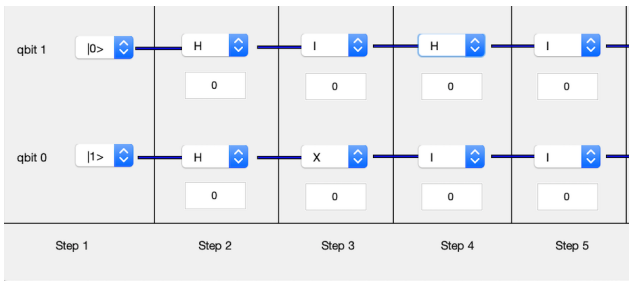


Figure 10: Duetch's Algorithm for f_1 in QCS

Figure 11 depicts the probabilities of the final states $|00\rangle$ and $|01\rangle$, with equal probability 0.5. In Figure 12 we can observe the phase π of state $|00\rangle$, which verifies the equation (6).

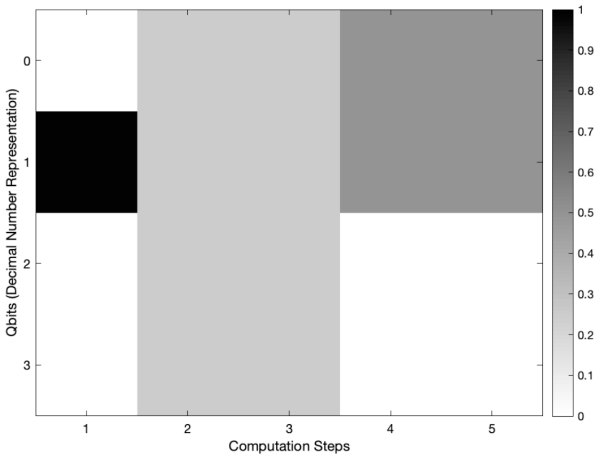


Figure 11: QCS:Qubits States

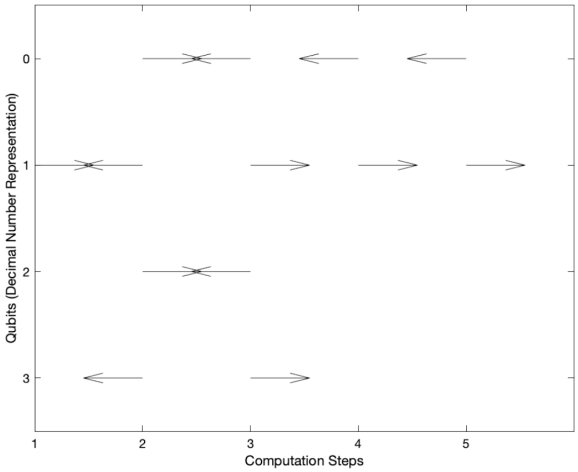


Figure 12: QCS:Qubits Phases

II. Case: U_{f_2}

Calculate the circuit

$$|q_2\rangle = |+\rangle|-\rangle = \frac{1}{2}(|0\rangle|0\rangle - |0\rangle|1\rangle + |1\rangle|0\rangle - |1\rangle|1\rangle$$

$$|q_3\rangle = U_{f_2}|q_2\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{2} \\ -\frac{1}{2} \\ \frac{1}{2} \\ -\frac{1}{2} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} \\ -\frac{1}{2} \\ -\frac{1}{2} \\ \frac{1}{2} \end{bmatrix} = \frac{1}{2}(|0\rangle|0\rangle - |0\rangle|1\rangle - |1\rangle|0\rangle + |1\rangle|1\rangle$$

$$|q_4\rangle = (H \otimes I)|q_3\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \frac{1}{2} \begin{bmatrix} 1 \\ -1 \\ -1 \\ 1 \end{bmatrix} = \frac{1}{2\sqrt{2}} \begin{bmatrix} 0 \\ 0 \\ 2 \\ -2 \end{bmatrix}$$

$$= \frac{1}{\sqrt{2}}(|1\rangle|0\rangle - |1\rangle|1\rangle) \tag{8}$$

$$= \frac{1}{\sqrt{2}}[|1\rangle \otimes (|0\rangle - |1\rangle)] = [|1\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)] = |1\rangle|-\rangle \tag{9}$$

In equation (9) we can verify the final state with the previous calculations from Table 3.

Executing the circuit in QCS

Figure 13 depicts the Oracle of Deutch's Algorithm, using an CNOT Gate. Figure 14 depicts the Quantum Circuit in QCS Environment, by using 2 qubits and 5 steps.

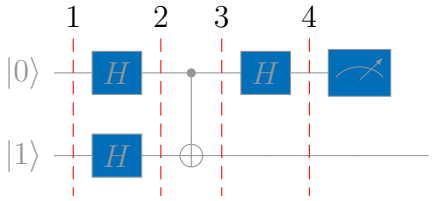


Figure 13: Duetch's Algorithm for f_2

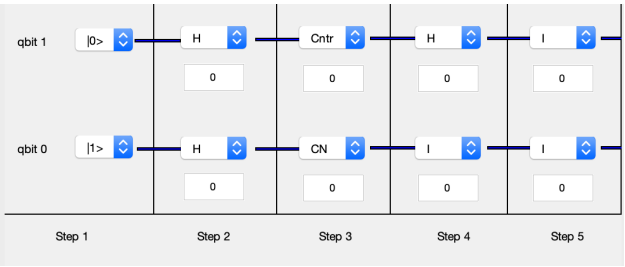


Figure 14: Duetch's Algorithm for f_2 in QCS

Figure 15 depicts the probabilities of the final states $|10\rangle$ and $|11\rangle$, with equal probability 0.5. In Figure 16 we can observe the phase π of state $|11\rangle$, which verifies the equation (9).

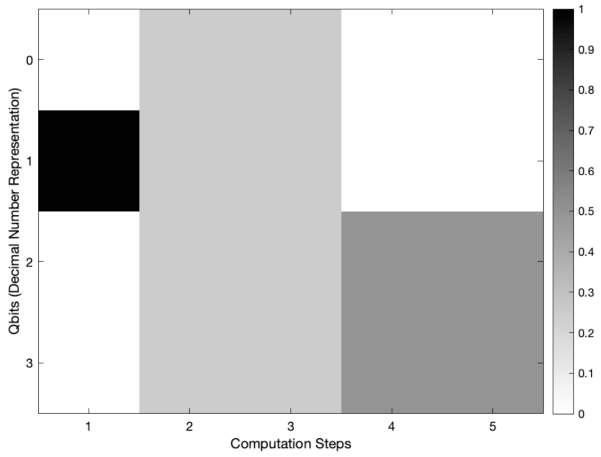


Figure 15: QCS:Qubits States

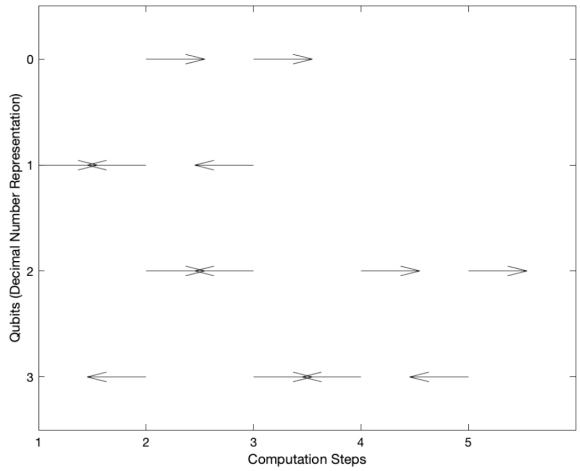


Figure 16: QCS:Qubits Phases

Executing in Qiskit Aer

The code was developed on Python 3.11.1 with Qiskit 1.1.1, Qiskit IBM Runtime 0.27.0 and Qiskit Aer 0.14.2 packages. Since Qiskit has a reversed bit-ordering, I wanted to measure the exact circuit as developed in QCS. Thus, I developed it in reverse order, and I used the draw(reverse_bits=True) parameter to draw it in reverse, from the already reversed, is it unitary? :)

Figure 17 depicts the Quantum Circuit simulated in Qiskit. Figure 18 depicts the Q-Shere of the final state in superposition before the measurement, which matches our previous results and depicts the phase π of $|11\rangle$ state. Figure 19 depicts the measurement, to the collapsed basis state $|1\rangle$.

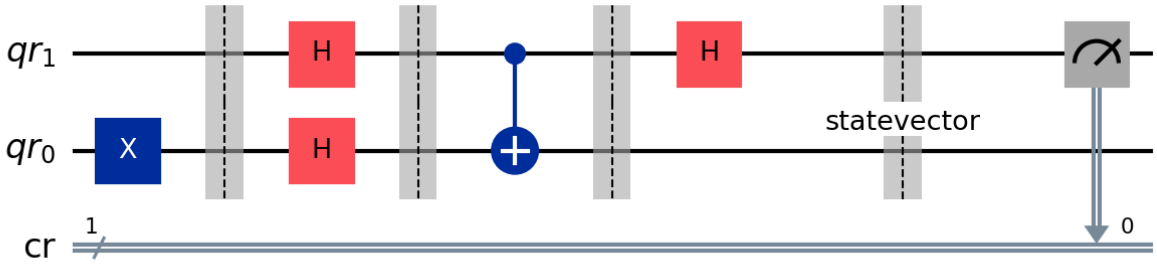


Figure 17: Qiskit Quantum Circuit

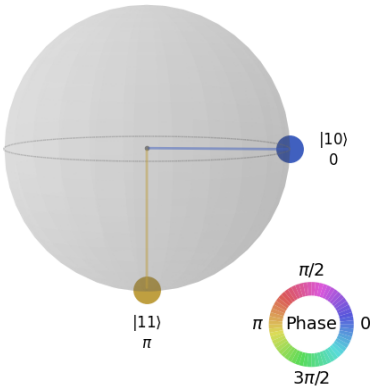


Figure 18: Q-Sphere

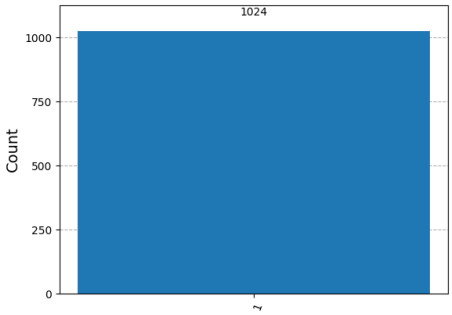


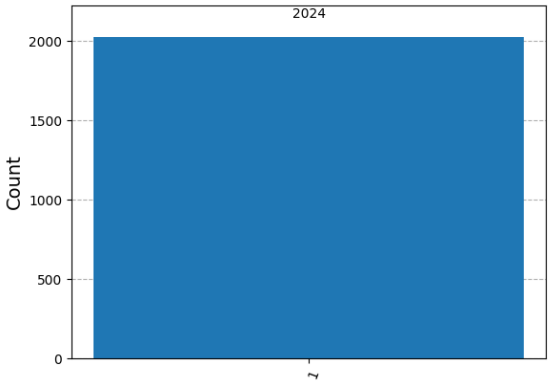
Figure 19: Plot Histogram

Executing in Real Quantum Computer

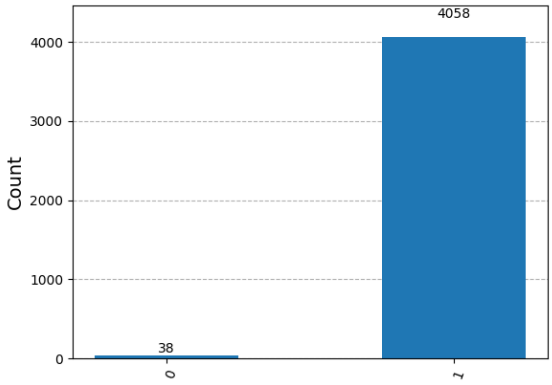
The Qiskit 1.+ version introduced breaking changes and a few suggestions. They mention the transpilation process and the usage of primitives to sample and estimate the circuit. During my experiments, I tried to first understand and simulate a real backend and observe a few metrics. After this, I ran it on a real Quantum Computer. It was very interesting to observe that Deutch's Algorithm took **3s** on an ibm_sherbrooke. I think this is a big execution time, and there is a latency from somewhere, during circuit execution or measurement.

Thus, I tried to write a few functions and measure what is happening. It still is not clear to me, but I believe that maybe the sampler primitive, along with the IBM runtime layer, can add this latency. The function "metrics()", handles all the plotting, and function "aer_mimic_simulation()" handles the simulations. The main part of the code is for real Quantum Computer. In the following graphs, I am comparing the counts, circuit depth and gate operations, for an aer simulator of an ideal and a fake_sherbrook backend, and a real ibm_sherbrook backend. In the last figures for Scheduling pass, I used the timeline_drawer() function, to analyze in dt units, the duration of each step. While it seems measurement takes more time, I would assume that the QPU time would be $9466 dt\_units * 0.0000000002 dt\_time(s) = 1.8932 * 10^{-6} s$. Interestingly, the circuit depth and the gate operations after transpilation in ibm_sherbrook, increased. Also the default shots of Sample against ibm_sherbrook, is 4k, which may added this QPU time.

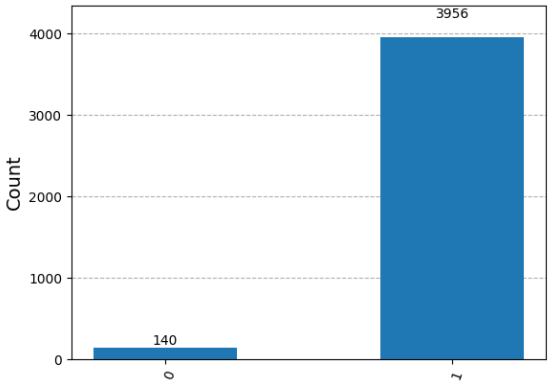
Figure 20: Results/Counts



aer_simulator(ideal)

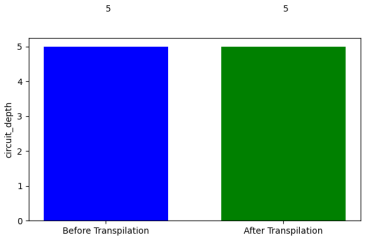


aer_simulator(fake_sherbrooke)

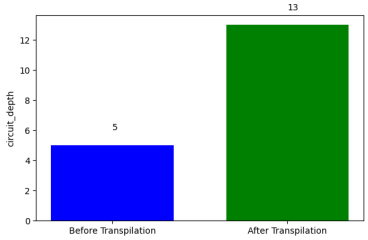


ibm_sherbrooke

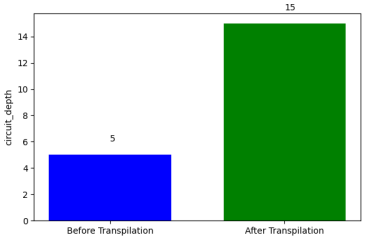
Figure 21: Circuit Depth



aer_simulator(ideal)

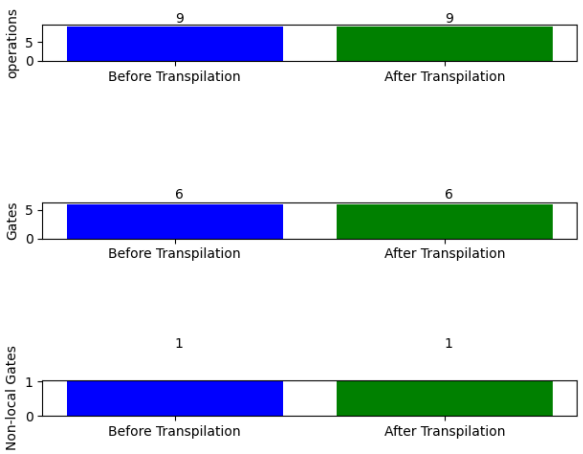


aer_simulator(fake_sherbrooke)

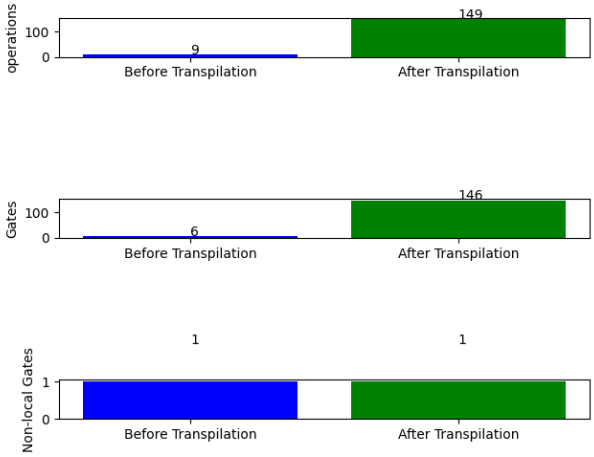


ibm_sherbrooke

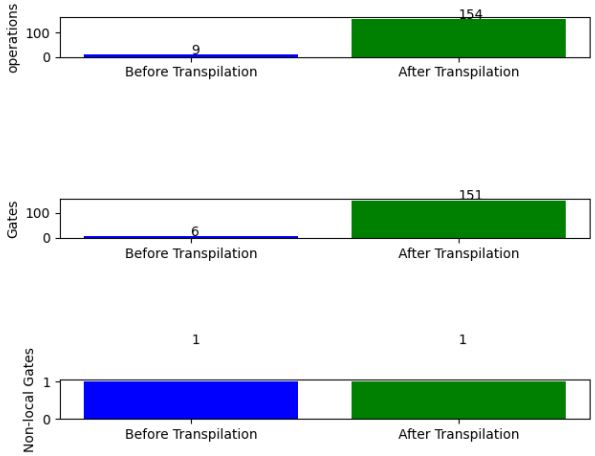
Figure 22: Gate Operations



aer_simulator(ideal)

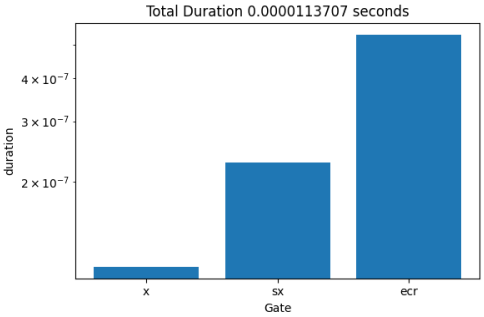


aer_simulator(fake_sherbrooke)

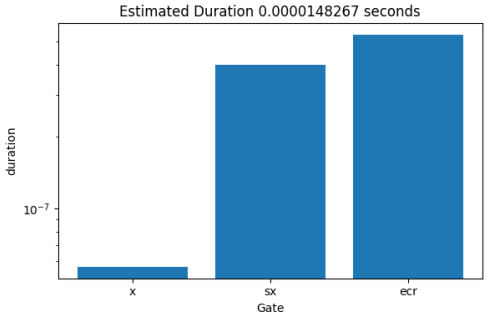


ibm_sherbrooke

Figure 23: Estimated Duration

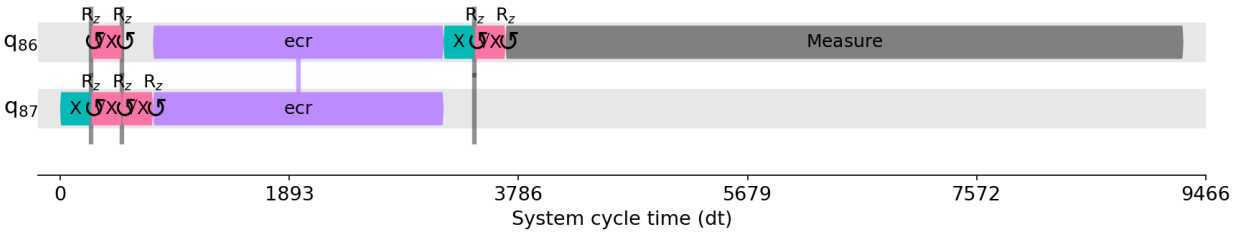


aer_simulator(fake_sherbrooke)

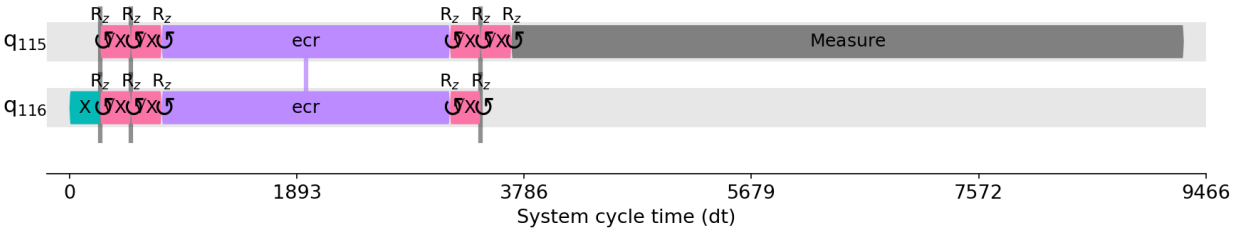


ibm_sherbrooke

Figure 24: Scheduling



aer_simulator(fake_sherbrooke)



ibm_sherbrooke