An excursion to the Quantum Realm

(An article for the Core Team Articles section in Beyond 2024.)

Soumyadeep Bose
Department of Electronics and Communications Engineering,
Heritage Institute of Technology, Anandapur, Kolkata

Welcome to the last Core Team Article! You've made it till here, fellow adventurer! However, this last one is, I must warn you, a little more mathematically inclined. As you might have already guessed from the title itself, this article is going to delve into the domain of Quantum Computing. I could have written something from my own primary domain- ai/ml, but I am an explorer by nature. I like to explore new domains and tread uncharted territories. And somehow, the domains that interest me lie at or near the point of intersection of Computer Science and Mathematics. So, voila! Here it is! A math-heavy article from a math-heavy domain.

Okay, enough background! Now let's get the engine started.

What are we going to discuss in this article?

We will firstly go over classical bits and quantum bits (also known as qubits) and in the process also learn about tensors and tensor product, then slowly transition into learning about a few basic quantum gates, i.e., operations that can be performed on qubits. We will follow that with understanding quantum superposition from the perspective of computation. And finally, we will discuss the Deutsche Oracle problem. (Hang on till the last because the last problem that I mentioned is somewhat of a magic. There, you'll get to really understand why people say quantum computers are way faster than classical computers.)

Pre-requisites:

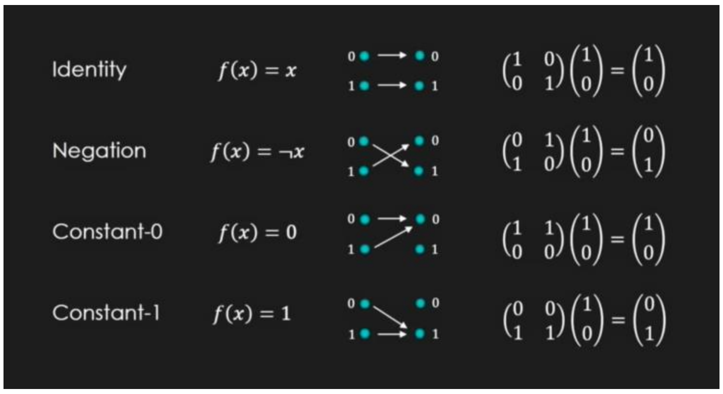
- A basic understanding of quantum mechanics from beforehand will be very handy.
- Basic knowledge of Linear Algebra will be essential.
- Besides the aforesaid, you'll also need a ton of interest in learning new stuff, because that's ultimately what drives humans to gain knowledge.
- Oh! Also, you mustn't fear Maths. For "fear is the mind-killer." (Pop-culture ref.)

Classical Bits:

Classical bits are the fundamental units of classical computing, representing binary digits that can hold values of either 0 or 1. They serve as the building blocks for processing and storing information in traditional computers, with each bit corresponding to a specific state or condition. The classical 0 bit can be represented by the vector $\begin{pmatrix}1\\0\end{pmatrix}$, and the classical 1 bit by $\begin{pmatrix}0\\1\end{pmatrix}$. Now, we won't be referring to the classical bits by their expanded matrix forms in this article, instead we will use the Dirac Vector Notation. According to Dirac Vector Notation, we represent the classical 0 bit by $|0\rangle$ and the classical 1 bit by $|1\rangle$. This notation is also called the *"bra-ket"* notation.

Now, let’s move on and introduce the 4 different basic functions we can perform on the classical bits.

1. **Constant 1**: This function always returns 1 for any input given to it. This is a constant function, as the name suggests.
2. **Constant 0**: This function always returns 0 for any input provided and this is also a constant function.
3. **Identity**: This function is defined as f(x)=x, which means it just gives the input provided to it as the output. This is a variable function as the output of the function depends on the input.
4. **Negation**: This function is defined as f(x)=-x, where the output is just the input times -1. This is also a variable function as the output depends on the input given.



Identity
$f(x) = x$
$\begin{pmatrix}1 & 0\\0 & 1\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix}=\begin{pmatrix}1\\0\end{pmatrix}$
Negation
$f(x) = \neg x$
$\begin{pmatrix}0 & 1\\1 & 0\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix}=\begin{pmatrix}0\\1\end{pmatrix}$
Constant-0
$f(x) = 0$
$\begin{pmatrix}1 & 1\\0 & 0\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix}=\begin{pmatrix}1\\0\end{pmatrix}$
Constant-1
$f(x) = 1$
$\begin{pmatrix}0 & 0\\1 & 1\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix}=\begin{pmatrix}0\\1\end{pmatrix}$

So, with these basic functions known to us, lets proceed to learn what is a tensor and what is meant by a tensor product. We will need this

Tensors and Tensor Product:

Okay, so according to Wikipedia, in mathematics, a tensor is an algebraic object that describes a multilinear relationship between sets of algebraic objects related to a vector space. Tensors may map between different objects such as vectors, scalars, and even other tensors. (Ref. https://en.wikipedia.org/wiki/Tensor) But thankfully, for our adventure we won’t need to delve deep into the mathematical significance of tensors. Instead, we will take a look at what tensors signify in the domain of computing.

Imagine you have a bunch of numbers organized in a grid-like structure, where you can access each number using coordinates. Now, think of these numbers as representing something, like temperature at different points on a map. This grid of numbers is like a tensor. But unlike simple grids, tensors can have more dimensions, not just rows and columns. They can also represent more complex relationships between things. For example, in physics, tensors might represent how forces act in different directions at each point in space. So, in plain language, tensors are just a way to organize and understand lots of numbers that relate to each other in a more intricate way than just in a list or a grid. Or, in even simpler words, tensors are multi-dimensional vectors. There, you have it now!

So, now let's understand what is a tensor product. And is it like a vector product then, since tensors are just multi-dimensional vectors?

Now, given two tensors $\binom{a}{b}$ and $\binom{c}{d}$, we obtain the tensor product as follows:

$$\binom{a}{b} \otimes \binom{c}{d} = \begin{pmatrix} a\binom{c}{d} \\ b\binom{c}{d} \end{pmatrix} = \begin{pmatrix} ac \\ ad \\ bc \\ bd \end{pmatrix}$$

And similarly, you can find the tensor product of n tensors too. One thing to note here is that, if we multiply a tensor with p elements, with a tensor with q elements, the resulting tensor product will have $p*q$ number of elements. In our generalization, since we have taken tensors with the same number of elements, i.e. 2, the number of elements in the tensor product will be 2^n, where n is the number of tensors we are finding the product of.

Multiple Classical-Bit Representation:

We have earlier seen how to represent a classical 0 bit and 1 bit, but now let's talk about the representation of multiple classical bits, like $|00\rangle$ or $|01\rangle$ or so on and so forth. With our knowledge of tensor products, let's see how we represent $|00\rangle$ classical bit:

$$|00\rangle = |0\rangle \otimes |0\rangle = \binom{1}{0} \otimes \binom{1}{0} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Here we call $|00\rangle$ the product state, and the individual tensors that are getting multiplied, the individual state. The interesting thing to note here is that if we are given just the product state, we can factorize it and obtain the individual states. Another important point is, if the product state has n bits, then its vector representation will have 2^n elements.

Qubits:

Now with our fresh understanding of classical bits, we move on to understanding what quantum bits, or as they are popularly called "Qubits," are. In classical bits we saw the value is always binary: either you have a 0 bit or a 1 bit. But what if you could get the infinite decimal numbers in between 0 and 1 to help in computation? That's where Qubits come.

Qubits are bits that can have values that are not necessarily 0 or 1, but are somewhere between the two. We call this superposition. We say the qubit is in a state of superposition, that collapses when we measure the value of the qubit.

Let's take a generalized qubit $\begin{pmatrix} a \\ b \end{pmatrix}$. Here $|a|^2$ is the probability that the qubit will collapse to 0 when measured, and $|b|^2$ is the probability that it will collapse to 1. Therefore, from this we can easily infer that $|a|^2 + |b|^2 = 1$. For example, let's take the qubit $\begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix}$. Here, the probability that our example qubit will collapse to 1 is 0.5, and to 0 is 0.5 too. Thus, we see that is does satisfy our condition of $|a|^2 + |b|^2 = 1$. Applying this understanding of ours to the classical bits we encountered earlier, we see that $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ because the probability of this bit collapsing to 0 is 1, and to 1 is 0. Similarly, the classical 1 bit can be explained too. But wait! Why am I talking about classical bits collapsing? Well, classical bits are special quantum bits, that are not in superposition.

Multiple Qubit Representation:

This is very similar to what we learnt earlier with the representation of multiple classical bits. For example, let's take two qubits $\begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix}$ and $\begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix}$, that are in superposition. As we can see, both qubits have equal chances of collapsing to either 0 or 1. Now if we multiply these two tensors according to the tensor product rule we learnt, we will get the tensor product to be $\begin{pmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \end{pmatrix}$, and you can verify yourself that it does satisfy the rule of squaring and adding all the elements of the product to get unity. But what is the interpretation of this tensor product? We can say that the product means the resultant qubit, after multiplying, has a probability of 0.25 of collapsing to either $|00\rangle$, $|01\rangle$, $|10\rangle$ or $|11\rangle$.

Now, with our understanding of qubits, let's proceed to understanding the three basic quantum operators that we will need to continue on our adventure. Those three operators are:

- X Gate
- CNOT Gate
- Hadamard Gate

Quantum Gates:

1. **X Gate (Bit Flip):** This Quantum Gate, represented by a X, is perhaps the easiest one to learn. The X Gate just flips the qubit given to it as input. For example, if we input the qubit $\begin{pmatrix}1\\0\end{pmatrix}$ which is $|0\rangle$ the X Gate will give us the output $\begin{pmatrix}0\\1\end{pmatrix}$ which is $|1\rangle$. So, we see that the X Gate has flipped the bit that we gave. In tensor form, this gate can be represented as:

$$X = \begin{pmatrix}0 & 1\\1 & 0\end{pmatrix}$$

So, if we write our example mathematically, in tensor form, it will be:

$$X\,(|0\rangle) = X\begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}0 & 1\\1 & 0\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}0\\1\end{pmatrix} = |1\rangle$$

This operation is reversible in nature. By this I mean that if you have the output bit of a X Gate, you can get the input bit easily by just passing the output bit through the X Gate again. From our previous example,

$$X\,(|1\rangle) = X\begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix}0 & 1\\1 & 0\end{pmatrix}\begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix}1\\0\end{pmatrix} = |0\rangle$$

2. **CNOT Gate (Controlled-NOT):** This gate, represented by a C, takes in bits like $|00\rangle$ or $|11\rangle$ or so on. If we consider the input to be $|ab\rangle$, then the CNOT Gate flips the bit 'b' if 'a' is 1, else acts as identity if 'a' is 0. So, if the input is $|00\rangle$ it'll give $|00\rangle$ as the output, but if the input is $|11\rangle$ then the output will be $|10\rangle$.

Earlier in this article we found the value of $|00\rangle$ to be $\begin{pmatrix}1\\0\\0\\0\end{pmatrix}$. Now if we pass this through the CNOT Gate, the output will remain the same. Let's verify this:

$$C\,(|00\rangle) = C\begin{pmatrix}1\\0\\0\\0\end{pmatrix} = \begin{pmatrix}1 & 0 & 0 & 0\\0 & 1 & 0 & 0\\0 & 0 & 0 & 1\\0 & 0 & 1 & 0\end{pmatrix}\begin{pmatrix}1\\0\\0\\0\end{pmatrix} = \begin{pmatrix}1\\0\\0\\0\end{pmatrix} = |00\rangle$$

But, if we input $|10\rangle$, the gate will output $|11\rangle$, and vice versa. This can also be verified as:

$$C\,(|10\rangle) = C\begin{pmatrix}0\\0\\1\\0\end{pmatrix} = \begin{pmatrix}1 & 0 & 0 & 0\\0 & 1 & 0 & 0\\0 & 0 & 0 & 1\\0 & 0 & 1 & 0\end{pmatrix}\begin{pmatrix}0\\0\\1\\0\end{pmatrix} = \begin{pmatrix}0\\0\\0\\1\end{pmatrix} = |11\rangle$$

The reverse of this operation is possible too, i.e. you can get the input back by applying the gate again to the output.

3. **Hadamard Gate:** This gate, represented by a H, takes a classical 0 bit or 1 bit as the input, and puts them in superposition. How? Well, let's see.

Let's take the classical 0 bit first, which is denoted by $\begin{pmatrix}1\\0\end{pmatrix}$, and the tensor form of the Hadamard Gate, which is $\begin{pmatrix}1/\sqrt{2} & 1/\sqrt{2}\\ 1/\sqrt{2} & -1/\sqrt{2}\end{pmatrix}$. Now mathematically the operation of the Hadamard Gate is represented as:

$$\mathrm{H}\,(|0\rangle) = \mathrm{H}\begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}1/\sqrt{2} & 1/\sqrt{2}\\ 1/\sqrt{2} & -1/\sqrt{2}\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}1/\sqrt{2}\\ 1/\sqrt{2}\end{pmatrix}$$

And similarly, we can calculate that if we pass $|1\rangle$ through the Hadamard Gate, the output will be $\begin{pmatrix}1/\sqrt{2}\\ -1/\sqrt{2}\end{pmatrix}$. Now, you might have a question that why is the H44 element negative? Well, had it been positive too, we would have gotten $\begin{pmatrix}1/\sqrt{2}\\ 1/\sqrt{2}\end{pmatrix}$ as output regardless of we give $|0\rangle$ or $|1\rangle$ as the input. This would have violated the reversible nature of these gates, as we would not be able to determine the input given the output in such a scenario.

Now, one interesting fact about Hadamard Gate is that, just like it can take a bit and put it into superposition, it can also take the superposition and convert it back into $|0\rangle$ or $|1\rangle$. Like, from the previous example,

$$\mathrm{H}\begin{pmatrix}1/\sqrt{2}\\ 1/\sqrt{2}\end{pmatrix} = \begin{pmatrix}1/\sqrt{2} & 1/\sqrt{2}\\ 1/\sqrt{2} & -1/\sqrt{2}\end{pmatrix}\begin{pmatrix}1/\sqrt{2}\\ 1/\sqrt{2}\end{pmatrix} = \begin{pmatrix}1\\0\end{pmatrix} = |0\rangle$$

So, as it seems, with the Hadamard Gate, we can easily transition out of superposition even without measurement, which would lead to our qubit collapsing. Therefore, we can use this property to structure quantum computation deterministically rather than probabilistically.

Okay, so finally the training phase of our protagonist is over! Now we can get to fighting the Dragon head-on.

The Deutsche Oracle Problem:

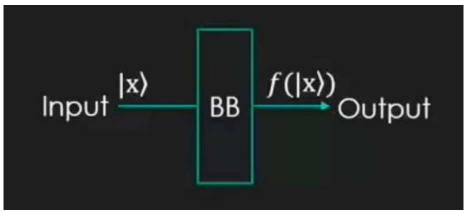
First of all, let's define the problem. Picture this. You are given a black box that's performing some kind of function on the input bit and giving you an output bit. This function is one of the four functions that we discussed in the 'Classical Bits' section near the beginning of this article. The function can either be a constant function or a variable function. Now, you are asked what kind of function is the black box applying to our input: Is it constant or variable? How many attempts do you think it'll take to determine that?

Well, let me assist you in coming to a definitive answer.

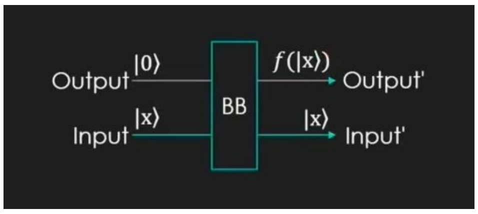
1. If we have input 0, the output is also 0. Now we give input 1 and the output still remains 0, so we conclude that there is a constant 0 function inside the black box. Something very similar is done for constant 1 too.
2. Now if we give input 0 and get 0 as output, and then input 1 and get 1 back as the output, we can conclude that there is an identity function within the black box.
3. Lastly when we give input 0 and get the output as 0, and then we give input 1 and get the output as -1, we know that the black box has a negation function inside.

So, what did you observe? Each time we would need two attempts to decide which function it is. Then how many attempts would we need to determine whether the function is constant or variable, and not the exact function? Any guesses?

ONE! We would need only one attempt to correctly predict if the black box has a constant or variable function inside. Let's see how.



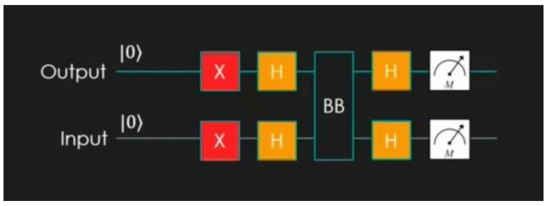
(Non-Reversible)



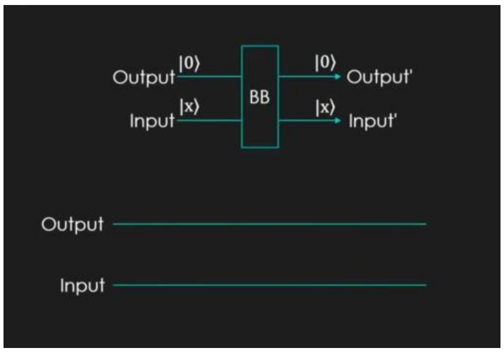
(Reversible)

But first we would need to convert out quantum circuit from that of non-reversible function to a reversible one. See the picture above. In the non-reversible one, there is one input on one side and one output line on the other side of the black box. But in the reversible one, we have input and output lines on either side of the black box. Now you might be wondering why the structure of the reversible quantum circuit looks like that. Well, let me give you an analogy. Suppose your teacher gives you a blank paper and asks you "What is your name, beta?" Here, compare the question of the teacher to the input in the picture |x⟩ and the blank paper to write the output on to the |0⟩ in the output line. Now, you write "My name is Khan" on the paper and go to submit it to your teacher. Compare the act of you writing your response on the blank paper as applying a certain function f inside the black box, and your response to f(|x⟩). Now, assume that is teacher doesn't have any memory. Sad, I know, but it needs to assumed since our circuit also does not have any memory. So, the teacher asks you why you have written what you have, on the paper. That when you remind her that she asked you to write your name on the paper. Compare this act of you reminding her as giving out the input too, from the black box. So, now you kind of know, why we need the two-port sort of structure for our reversible quantum circuit. In summary, we just want our black box to give the input as well as the value when the function is applied to the input, as the two outputs. And the $|0\rangle$ output line is required on the left, to sort of write out function's output on it.

Now, with that clear, let's analyse the circuit itself.

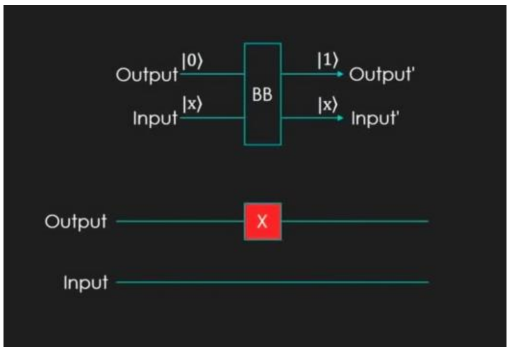


1. First off, let's put a **constant 0** function inside the black box. The reversible quantum circuit for the constant zero function looks like the following:



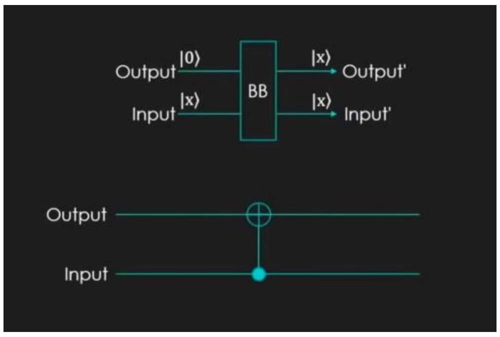
So, now let's begin analysing the deutsche oracle problem with constant 0 function inside the black box. $|0\rangle$ is applied both at the input and the output lines, that go through the X Gate or Bit Flip operator to become $|1\rangle$. Now both $|1\rangle$ and $|1\rangle$ go through the Hadamard Gate that gives $\begin{pmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{pmatrix}$ on both the input and output lines. Now after passing unaltered through the black box, there are two Hadamard Gates at the two lines for some post-processing. Those two Hadamard Gates to the right of the black box in the circuit diagram, convert the $\begin{pmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{pmatrix}$ back into $|1\rangle$, thus giving us $|1\rangle$ on the input line, and $|1\rangle$ on the output line too. Therefore, we end up with $|11\rangle$ as the state after measurement. Please note here that the input line state is the most significant bit, while the output line state is the least one.

2. Now let's fit a **constant 1** function inside the black box. The quantum circuit for a reversible constant 1 function looks like as follows:



Just like the previous one, in this circuit too, we get $\begin{pmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{pmatrix}$ on the input and output lines to the black box. But here the output line is flipped inside the box, leading it to give $\begin{pmatrix} -1/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix}$ on the output line while the input line output remains the same. Now after applying the Hadamard Gate to both the input and the output line while post-processing we get $\binom{0}{1}$ or $|1\rangle$ on the input line and $\binom{0}{-1}$ on the output line, which is also $|1\rangle$. So, we still end up with $|11\rangle$ as the measurement.

3. So, we are done with the constant functions. Now moving on to the variable functions, we begin with the **identity** function, the rev. quantum circuit for which is:



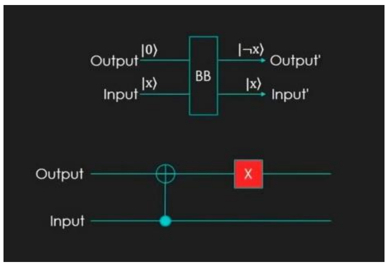
Here too, just like the first point, we are giving $\begin{pmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{pmatrix}$ on the two lines to the black box. Now, the tensor product of the two is calculated and a CNOT Gate is applied. Then the resulting tensor is factored back into two individual states. Let's see mathematically what I mean.

$$\mathrm{C}\left(\begin{pmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{pmatrix} \otimes \begin{pmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{pmatrix}\right) = \mathrm{C}\begin{pmatrix} 0.5 \\ -0.5 \\ -0.5 \\ 0.5 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}\begin{pmatrix} 0.5 \\ -0.5 \\ -0.5 \\ 0.5 \end{pmatrix} = \begin{pmatrix} 0.5 \\ -0.5 \\ 0.5 \\ -0.5 \end{pmatrix}$$

and we can factor $\begin{pmatrix} 0.5 \\ -0.5 \\ 0.5 \\ -0.5 \end{pmatrix}$ into $\begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix}$ and $\begin{pmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{pmatrix}$, which is $|0\rangle$ and $|1\rangle$.

So, we get $|01\rangle$ on measurement.

4. Now finally, we put the variable function **negation** inside the black box. The reversible quantum circuit for negation function is:



Here we can see that there is a CNOT Gate followed by a Bit Flip operator inside the black box. Now just like the previous identity black box, here too we will get $\begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix}$, which is $|0\rangle$ as the value at the input line. But at the output line we will get the bit flipped equivalent of $\begin{pmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{pmatrix}$, which is $\begin{pmatrix} -1/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix}$ or after Hadamard post-processing gives $\begin{pmatrix} 0 \\ -1 \end{pmatrix}$, which is still $|1\rangle$ since $|\,(-1)\,|^2$ is the probability of the qubit collapsing to 1. So, we finally get $|01\rangle$ at the output lines on measurement.

So, what did we observe? One attempt with $|00\rangle$ as the input on the two lines to the black box. This Quantum Circuit will return $|11\rangle$ if the function inside the black box is constant, else $|01\rangle$ if it is variable. So, in just one attempt and some mind-tingling maths, we have determined the type of function inside the black box. Classically, to determine if a function is constant or balanced, we would need to evaluate the function for at least half of the possible inputs. This would require many queries to the circuit, with the number of queries increasing with the size of the input space. For a function with n inputs, a classical computer would need to make $2^{(n-1)} + 1$ function evaluations in the worst case to determine if the function is constant or balanced. But in a quantum computer we need just one function evaluation. So, this is where you see a glimpse of the speedup potential that the quantum computers have.

This algorithm was designed to work on just 2 bits. However, there is a generalized n-bit version of this algorithm called the Deutsch Josza Algorithm. (Ref. https://en.wikipedia.org/wiki/Deutsch%E2%80%93Jozsa_algorithm)

Bravo, adventurer! You have slayed the Dragon! Congratulations on getting through this adventure with me.

You have learned about classical bits, tensors, tensor product qubits, basic Quantum Gates, and the Deutsche Oracle Problem. You really know quite a lot now. So, welcome to the domain of Quantum Computing. If you'd want to have a chit-chat on any of the topics, or would perhaps want me to know about your views or interpretation on any topics, feel free to ping me up on my socials.

Till then this was me Soumyadeep, your friendly neighbourhood ai/ml guy, now signing off.