QCOMPPW100: Quantum computing kickoff practical work

Lotus Noir Quantum Computing Research Group

Based on:
MITx: 8.370.1x Quantum Information Science I, Part I

Practical work Outline: This practical work has the goal of teaching the core concept of quantum computing. There is no quantum computing requirement, knowledge in classical computing will be sufficient.

Contents

1 Reversible gate

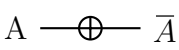
Notion goal 1.1. In this section, we will get familiar with boolean algebra's concept of **reversible gates**.

Definition 1.1. *A **reversible gate** is a logic gate from which you can deduce the input from the output.*

There are many examples of **reversible gates**:

- The **NOT** gate.

  Distinctive shape[1]:

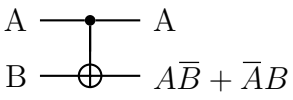
  Truth table:

Input	Output
0	1
1	0

  Indeed if the output is 1, the input must be 0 and vice-versa.

- The **CNOT** gate.

  Distinctive shape:

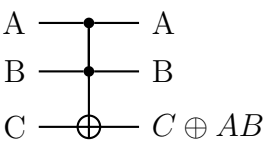
  Truth table:

Input	Output
00	00
01	01
10	11
11	10

  The **CNOT** gate is called this way for control **NOT** gate. The second bit is negated only if the first bit is true.

- The Toffoli gate (CCNOT).

  Distinctive shape:

  Truth table:

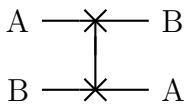
[1] Each of this gate has a distinctive shape in classical computing and in quantum computing. For this workshop, we will only work on classical computing, yet for convenience, we will use their quantum computing distinctive shape.

Input	Output
000	000
001	001
010	010
011	011
100	100
101	101
110	111
111	110

The Toffoli gate is also called the CCNOT gate because it is a doubly controlled **NOT** gate.

- The SWAP gate.

  Distinctive shape:



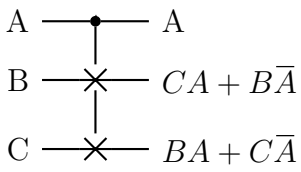
  Truth table:

Input	Output
00	00
01	10
10	01
11	11

  The SWAP gate inverts the value of two bits.

- The Fredkin gate (CSWAP).

  Distinctive shape:



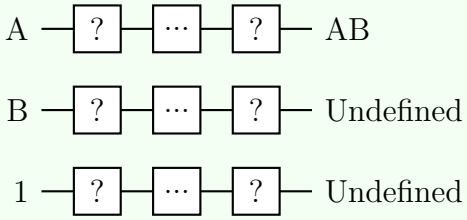
  Truth table:

Input	Output
000	000
001	001
010	010
011	011
100	100
101	101
110	111
111	110

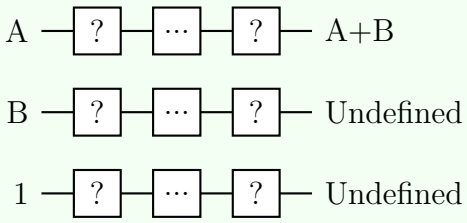
The Fredkin gate is also called the CSWAP gate because it is a controlled SWAP gate.

Unlike classical gates, reversible gates need to have at least as much output as input. The **AND** and **OR** gates have less input than output, so they cannot be reversible gates. For the rest of the practical work, we will only work with logic gate that has as much input as output(like the **NOT** gate).

Exercise 1.1. Make a boolean circuit only using the reversible gates shown previously. This circuit has to take in input 3 boolean A, B, and 1 and has to output AB, undefined and undefined. You do not have the right to use extra input. (Undefined means that every output is accepted)



Exercise 1.2. Same exercise as the previous one, but with the OR gate

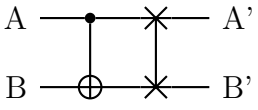


What we have learned 1.1. In this part, we have learned that a reversible gate was a logical gate from which you can deduce the input from the output and we have practiced how to build logic circuits only using reversible gates.

2 Logic circuit and probabilities

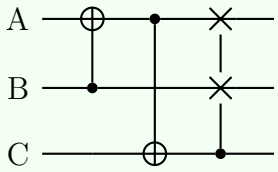
Notion goal 2.1. The goal of this section is to give you a first look at the notion of superposition in quantum computing. To do so we will mix logical circuits and basic probabilities notions.

Now we will consider logical circuit, that has as input a variable that has a probability to be equal to 0 and probability to be equal to one. Like this circuit:



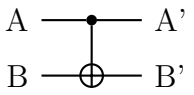
Where $P(A = 0) = \frac{1}{2}$ and where B = 1 Si if we do the computation we have $P(A' = 0) = \frac{1}{2}$

Exercise 2.1. Find the probabilities of every possible output of this circuit:



Where $P(A = 0) = \frac{1}{2}$, $P(B = 0) = \frac{2}{3}$ and $C = 1$

Consider this logical circuit:



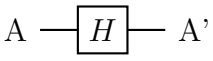
Lets say that $P(A = 0) = \frac{1}{2}$ and that $B = 1$. What is the value of $P(B' = 0)$? Of course $\frac{1}{2}$. Yet now if $A' = 0$ then $B' = 1$ and if $A' = 1$ then $B' = 0$. Therefore, by knowing the value of A you can deduce the value of B. They are linked somehow, this concept is called entanglement in quantum computing.

What we have learned 2.1. During this section, we have understood how logic circuits interacts with probabilities. Indeed a logic circuit only assigned the probabilities of the input to the probabilities of the output.

3 The Hadamard gate

Notion goal 3.1. During this section, we will add the Hadamard gate to our model of probabilistic logic circuit. This will allow us to understand why quantum computers are stronger than classical ones.

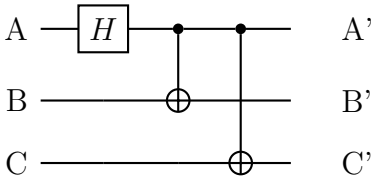
In our model, the Hadamard gate is special because it plays with the probability of the bit that goes through. Indeed if we consider this circuit:



Where $P(A = 0) = p$ with $p \in 0, 1$ we have:

$$P(A' = 0) = \frac{1}{2} + \sqrt{p - p^2}$$

Example 3.1. Imagine you had to simulate a simple logic circuit with only reversible gates and no probability in the input. If the circuit as n bit as input, you would also output n bit[2]. Now consider this circuit:



[2]Simple, Basic

Where $A = 0$, $B = 0$ and $C = 0$. After the Hadamard gate we have:

$$P(A = 0) = \frac{1}{2} + \sqrt{0 - 0^2} = \frac{1}{2}$$

So if we compute the all circuit, we find that: $P(A' = 0) = P(B' = 0) = P(C' = 0) = \frac{1}{2}$. If we had to simulate this circuit one naive answer would be too output 3 floats that represent the probabilities of A', B' and C'. It will informs us that $P(A' = 0) = P(B' = 0) = P(C' = 0) = \frac{1}{2}$, which is true. Yet we will miss that A',B', and C' are untangled. Indeed If $A' = 1$, then we must have $B' = 1$ and $C' = 1$. Same thing if $A' = 0$. So if we want to simulate such circuits we will have to output the probability of each output. Like this way:

Input	Probability
000	$\frac{1}{2}$
001	0
010	0
011	0
100	0
101	0
110	0
111	$\frac{1}{2}$

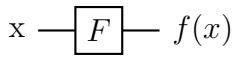
So now we will have to output 7 floats! So for n inputs, we have to output $2^n - 1$ floats. We have jumped from a linear scale to an exponential scale. This is why the probabilities and logic circuits mixed is a big deal!

What we have learned 3.1. During this section, we got an introduction of the Hadamard gate, which is a gate that plays with probability with the special bit of our model. Moreover we got a feeling of how much information we could find in reversible circuit having such Hadamard gate.

4 A first step in understanding the Deutsch algorithm

Notion goal 4.1. The goal of this section, is to make you catch a glimpse of the power of our model, and introducing to you a famous quantum computing problem.

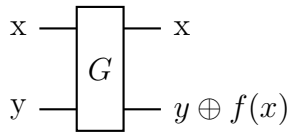
We saw earlier with the exercise's 1.1 and 1.2 that computing a not reversible function in a reversible circuit was a pain in the neck. Is there a way we could do it automatically ? Imagine a gate named F and function f that takes in input one bit and output one bit. The gate F works this way:



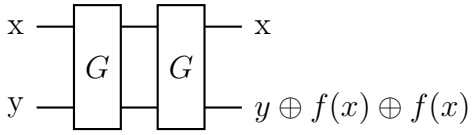
The function f could have, among all the possibilities, a such truth table:

Input	Output
0	0
1	0

So F is unfortunately not necessarily reversible. Now imagine a Gate G, that takes two bits, x and y in input and has two bits in output, x and $y \oplus f(x)$, like this way:



Is G reversible ? As a first step to find out, lets use G two times serially:



So as output we have among others $y \oplus f(x) \oplus f(x)$, yet $f(x) = f(x)$, so we have $y \oplus 0 = y$. So doubling G, bring us right where we have started. G is it own inverse. So G is reversible. Therefore we have find a way to incorporate every possible one bit function in a reversible circuit.

Definition 4.1. *The **Deutsch problem** is a problem where you are given a black box implementing a one bit function. To solve the problem you have to guess whether the function inside is constant or balanced (in this case, balanced means not constant).*

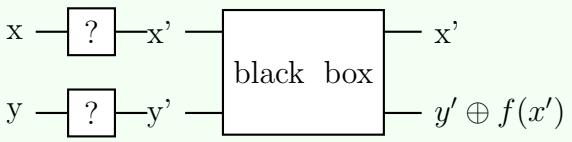
The black box is a one bit function, therefore it can implements four functions: **set 0**, **set 1**, **identity** and **negation**. Here are their truth table:

Input	set 0	set 1	identity	negation
0	0	1	0	1
1	0	1	1	0

The function **set 1** and **set 0** are constant, and the function **identity** and **negation** are balanced. How many call to the black box would you need to do on a classical computer? Actually two, here an algorithm in pseudo-code that solves it:

```
def deutsch(f):
    if f(0) == f(1):
        return "constant"
    else:
        return "balanced"
```

Exercise 4.1. Consider that we incorporate a such black box to our reversible circuits and that we had access to the probability of each combination at each time. Solve the Deutsch problem with one call to the black box.



With $x = 0$ and $y = 0$

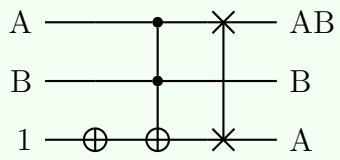
What we have learned 4.1. In this section we have learned how to incorporate any one bit function in a reversible circuit and we get feeling of how reversible circuit having Hadamard get could the solve the Deutsch problem in one call to the black box.

5 Exercise correction

5.1 Reversible gate

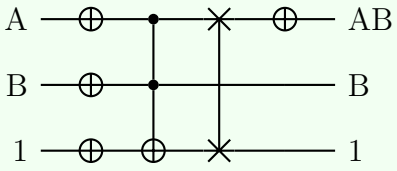
5.1.1 Exercise 1.1

This circuit is meet the requirements of the exercise:



5.1.2 Exercise 1.2

This circuit is meet the requirements of the exercise:



5.2 Logic circuit and probabilities

5.2.1 Exercise 2.1

A method to solve this exercise is to make the truth table of the circuit, then compute the probability of each input, and then finally link each output to its probability thanks to the truth table.
The truth table of circuit is:

Input	Output
000	000
001	001
010	111
011	110
100	011
101	100
110	010
111	101

Tips: The circuit is made only of reversible gate therefor every output must only come once in the truth table!
The probability of each input is:

Input	Probability
000	P(A=0)$\times P(B=0) \times P(C=0) = \frac{1}{2}0 \times \frac{2}{3} = 0$
001	$\frac{1}{3}$
010	0
011	$\frac{1}{6}$
100	0
101	$\frac{1}{3}$
110	0
111	$\frac{1}{6}$

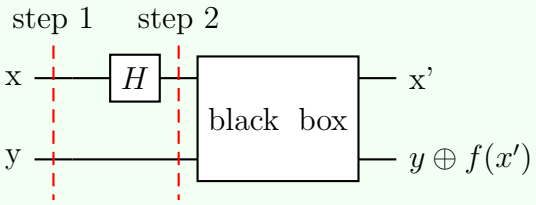
So now thanks to the truth table and the previous array we can solve the exercise:

Input	Probability
000	0
001	$\frac{1}{3}$
010	0
011	0
100	$\frac{1}{3}$
101	$\frac{1}{6}$
110	$\frac{1}{6}$
111	0

5.3 A first step in understanding the Deutsch algorithm

5.3.1 Exercise 4.1

The solution is to put an Hadamard gate the circuit before the black box, like this way:



Indeed lets compute the probabilities of the output,like we have learned to do. First lets compute the truth table of the circuit, according to the input given to the black box at step 2:

Input [a]	set 0	set 1	identity	negation
00	00	01	01	01
10	10	01	11	10

Here is the probability of each input:

Input	Probability
00	$\frac{1}{2}$
01	0
10	$\frac{1}{2}$
11	0

Here are the probabilities of each output for each function:

Output	set 0	set 1	identity	negation
00	$\frac{1}{2}$	0	$\frac{1}{2}$	0
01	0	$\frac{1}{2}$	0	$\frac{1}{2}$
10	$\frac{1}{2}$	0	0	$\frac{1}{2}$
11	0	$\frac{1}{2}$	$\frac{1}{2}$	0

The probabilities of the output of each function are different. Therefore if we have access to the probabilities of the circuit we can differentiate the functions.
In a quantum computer you do not have access to the probabilities like we do. Hence we still have things to learn to solve this problem on a real quantum computer, yet we will do that together ;).

[a] We did take into account the inputs: 01 and 11, because they had no chance to came out.