PHYC90045 Introduction to Quantum Computing

Lab Session 4

Welcome to Lab 4 of PHYC90045 Introduction to Quantum Computing.

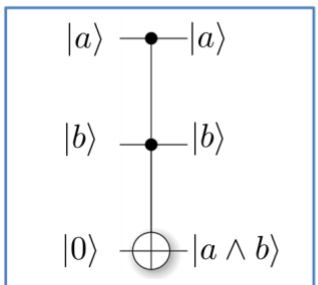
The purpose of this first lab session is to:

- implement classical logic using Toffoli gates
- program arithmetical operations (addition)
- program and analyse simple quantum algorithms

4.1 Classical logic using the Toffoli gate

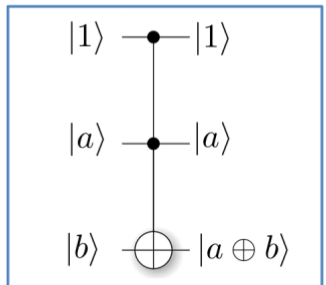
Here we will verify, by direct calculation, how the 3-qubit Toffoli gate allows us to perform classical logic. In the figure below examples are given together with the truth tables.

a AND b



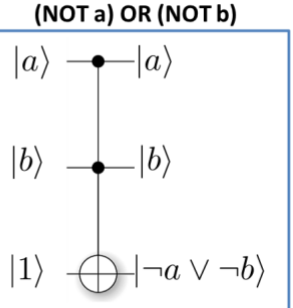
a	b	a ∧ b
0	0	0
0	1	0
1	0	0
1	1	1

a XOR b



a	b	a ⊕ b
0	0	0
0	1	1
1	0	1
1	1	0

(NOT a) OR (NOT b)

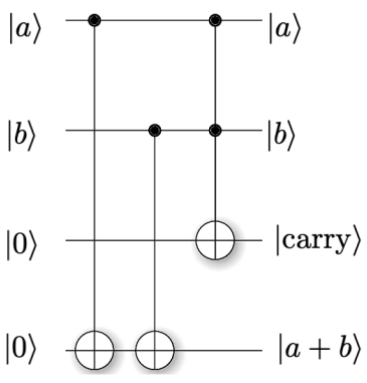


a	b	¬a ∨ ¬b
0	0	1
0	1	1
1	0	1
1	1	0

Exercise 4.1.1 In the QUI set up a 3 qubit instance and program a Toffoli gate, as above with qubit-1 and qubit-3 as the controls and qubit-3 as the target. By setting the input states appropriately verify each entry in the truth tables.

4.2 One-bit adder circuit

Exercise 4.2.1 In the lectures we went through the one-bit adder circuit in detail.



a) With reference to the one-bit adder circuit, review and understand the table below.

Input		Output			
a	b	a+b	carry	Output string	Output string: decimal
0	0	0	0	[0 0]	0
0	1	1	0	[0 1]	1
1	0	1	0	[0 1]	1
1	1	0	1	[1 0]	2

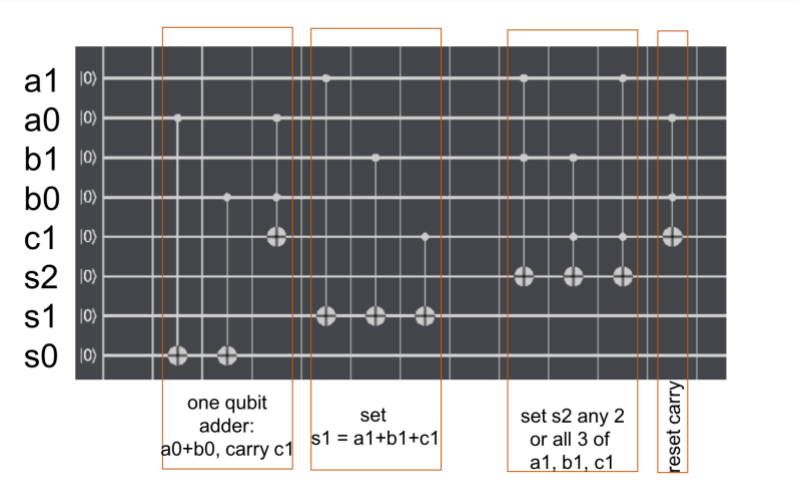
b) Program the circuit in the QUI, leaving space at the start of the circuit to add gates to specify the input states. For classical bit inputs, run the circuit and complete the following table:

Input		Output		
|a>	|b>	|a>|b>|carry>|a+b>	Answer (binary)	Answer (decimal)
|0>	|0>	|0>|0>|0>|0>	[0 0]	0

c) Explore the adder circuit with various superpositions as inputs, and understand what you are seeing.

4.3 Two-bit adder circuit

Consider the two-bit adder circuit below for S = A + B, where A = a0 + 2a1, B = b0 + + 2b1 and S = s0 + 2s1.

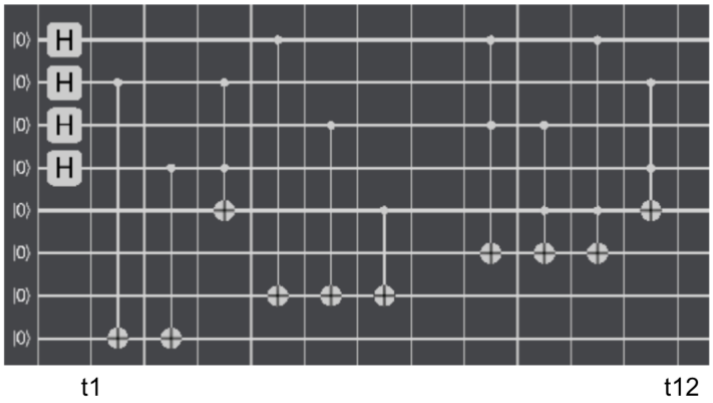


Exercise 4.3.1

a) Program the two-bit adder in the QUI. Run the circuit (with measurements at the end) and verify the arithmetic by filling in the following table (least significant bit last).

[a1 a0]	[b1 b0]	[s2 s1 s0]	S = A + B (decimal)
[0 0]	[0 0]	[0 0 0]	0 = 0 + 0
[0 1]	[0 0]		
[1 0]	[0 0]		
[1 1]	[0 0]		
[0 0]	[0 1]		
[0 1]	[0 1]		
[1 0]	[0 1]		
[1 1]	[0 1]		
[0 0]	[1 0]		
[0 1]	[1 0]		
[1 0]	[1 0]		
[1 1]	[1 0]		
[0 0]	[1 1]		
[0 1]	[1 1]		
[1 0]	[1 1]		
[1 1]	[1 1]		

b) Now edit the circuit so that all possible binaries A and B are presented to the adder in a superposition:

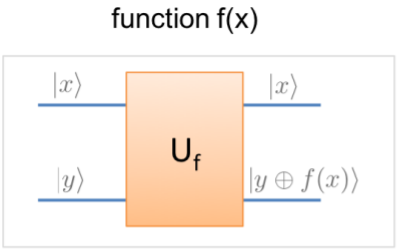


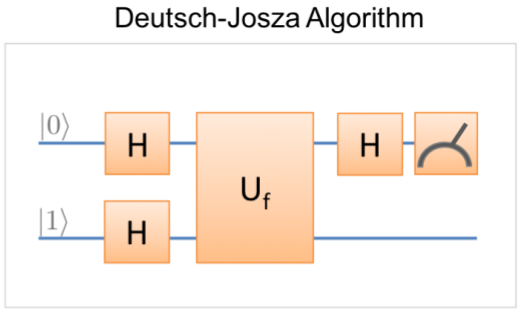
Explain the difference between the input state at t1 and the output state at t12 (also, note the level of entanglement generated in this circuit).

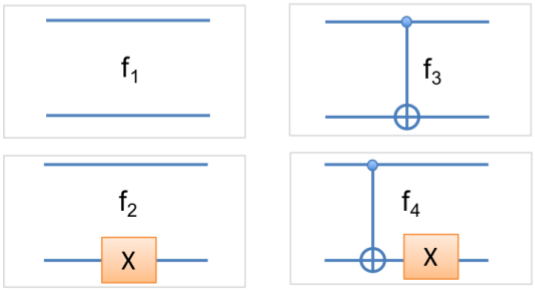
c) Add SWAP gates to the input and output parts of the circuit so that the bit ordering of A, B and S are reversed (i.e. most significant bit last).

4.4 Deutsch-Josza Circuits (1-qubit)

As we saw in lectures, the Deutsch-Josza (DJ) algorithm tests whether a binary function is constant or balanced – as per the summary below (for the one-bit functions).





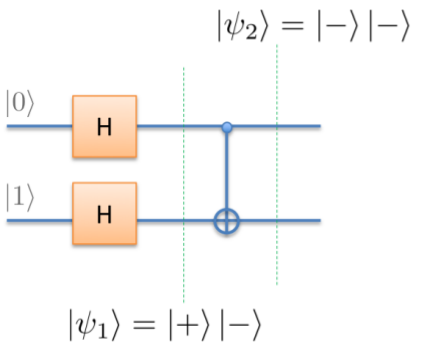


		Measured
CONSTANT	$f_1(0) = 0$ $f_1(1) = 0$	0
	$f_2(0) = 1$ $f_2(1) = 1$	0
BALANCED	$f_3(0) = 0$ $f_3(1) = 1$	1
	$f_4(0) = 1$ $f_4(1) = 0$	1

Exercise 4.4.1

a) Program the Deutsch-Josza circuit in the QUI as per above, and verify the algorithm outputs for the four function types.

b) Examine phase kickback in the QUI – program the following circuit (make sure you understand the X-basis states shown) and fill in the QUI states as indicated (ket/binary representation). Can you see the phase kickback in the QUI representation of the states?



QUI states

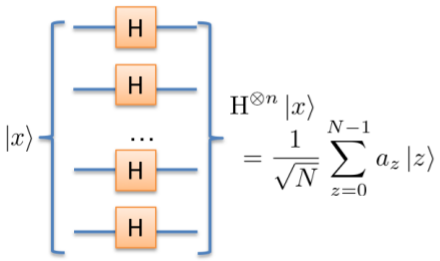
$|\psi_1\rangle$ _____

$|\psi_2\rangle$ _____

4.5 Deutsch-Josza Circuits (n-qubit)

Exercise 4.5.1 Verify the expression for the state after application of multiple Hadamards to a general state in the x-register:

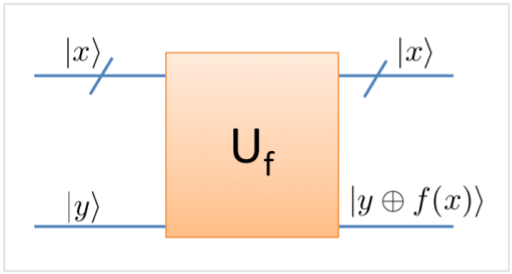
Hadamards applied to a general state (n qubits, N = 2^n): $\mathrm{H}^{\otimes n}|x\rangle = \frac{1}{\sqrt{N}}\sum_{z=0}^{N-1}(-1)^{x\cdot z}|z\rangle$

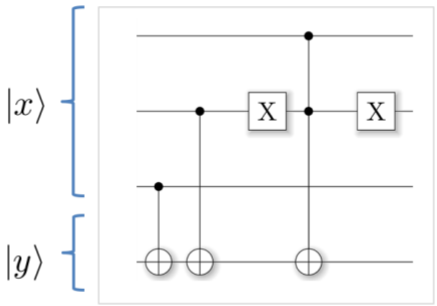


When 1's in the same location, we get a sign change: $(-1)^{x\cdot z}$

$$x \cdot z = \sum_{j=0}^{n} x_j z_j$$

Exercise 4.5.2 Consider the balanced function example given in lectures (see below).

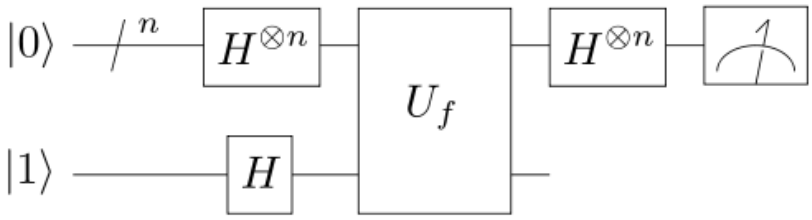




x	f(x)
000	0
001	1
010	1
011	0
100	1
101	0
110	1
111	0

Program this in the QUI and verify the table given using $|y\rangle = |0\rangle$. Now set $|y\rangle = |1\rangle$. Do you understand what is happening? Put the x-register into an equal superposition over all 8 binaries and run the circuit again. What is the circuit now producing as output? Place a measurement on the y-register after the function and explain what happens.

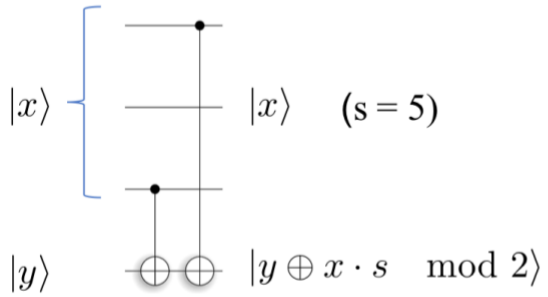
Exercise 4.5.3 Program this function into a circuit performing the Deutsch-Josza algorithm (as shown), run it and verify that the correct conclusion about the function is obtained (i.e. measurements on x-register imply a balanced function). Can you verify in the QUI that the conclusion is deterministic, i.e. that the circuit will never produce an output corresponding to a wrong conclusion?



4.6 Bernstein-Vazirani Algorithm

In the Bernstein-Vazirani problem we have a function $f = x \cdot s \bmod 2$ which maps the product of two n-bit numbers to {0,1}.

Exercise 4.6.1 Program the following (oracle) function in the QUI for the case s = 5 and verify the table of results.

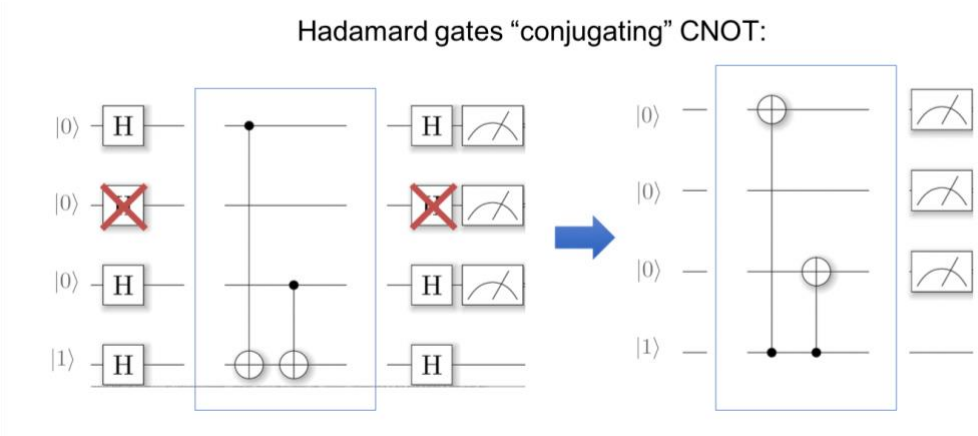


x	f(x)
000	0
001	1
010	0
011	1
100	1
101	0
110	1
111	0

Exercise 4.5.2 Modify the circuit to compute $f = x \cdot s \bmod 2$ for s = 3 and complete the table of results:

x	f(x)	x	f(x)
000		100	
001		101	
101		110	
011		111	

Exercise 4.6.3 Program the following circuits and verify that the equivalence holds. Superposition?

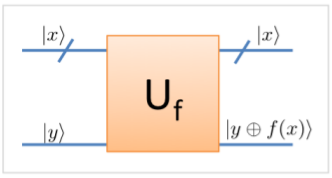


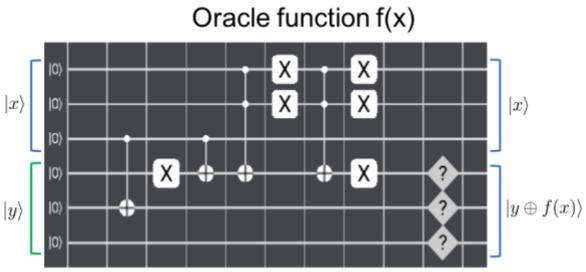
Exercise 4.6.4 Program the oracle function $f = x \cdot s \bmod 2$ which into the BV circuit to solve for the parameter s (i.e. using the DJ circuit) and verify the solution for a range of values of s (changing the oracle function as required).

4.7 Simon's Algorithm

Here we will implement a 3-bit example of finding the period of a function, i.e. a: $f(x) = f(x \oplus a)$, using Simon's algorithm.

Exercise 4.7.1 Consider the following circuit which implements an oracle function $f(x)$ over 3-bit numbers, with a period a.

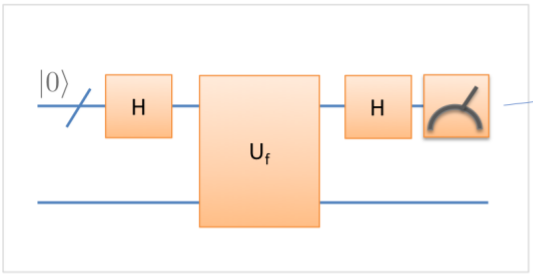




x	f(x)
000	
001	
010	
011	
100	
101	
110	
111	

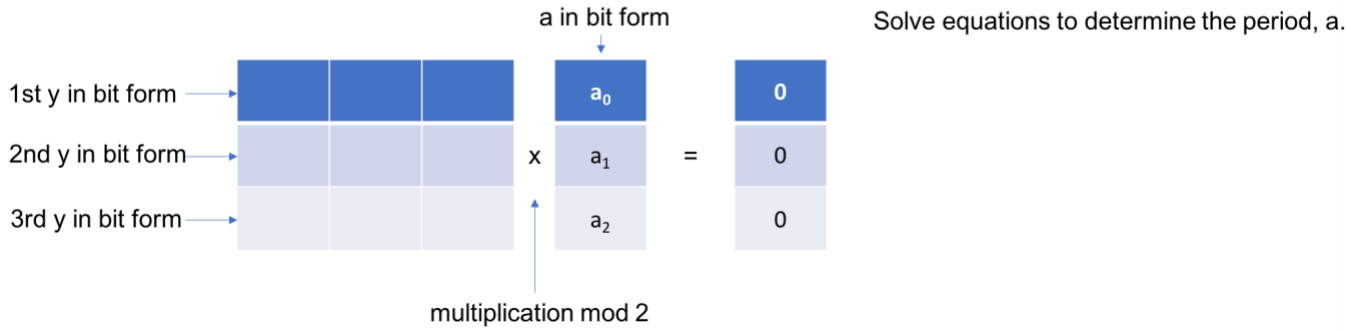
a) Program and run the circuit and fill in the table. By (classical) inspection, what is the period of the function we wish to determine (both bit and decimal forms)?

b) Incorporate the oracle into a circuit in the QUI to determine the period using Simon's algorithm. Run the circuit several times to obtain three distinct integers (y_a, y_b, y_c) in the upper register. From the system of equations (as per schematic below) and solve by hand (a classical step) to obtain the period a in bit form, and hence determine the decimal (integer) form of a and compare with the classical solution above.



Measurement on upper register gives results, y:

$$y : \; a \cdot y = 0 \mod 2$$



Run the time scrubber through the circuit and see if you can understand how it's working.