Phys 396 2018

Lecture 18

~~Thurs~~ ~~Thurs~~

Tues: HW by 5pm

Classical computation with quantum gates

Recall that classical computation entails evaluating functions of multiple bits. For example two n bit numbers can be represented by

$$X = x_{n-1} x_{n-2} \ldots x_2 x_1 x_0$$

$$y = y_{n-1} y_{n-2} \ldots y_2 y_1 y_0$$

and we might aim to add these to form

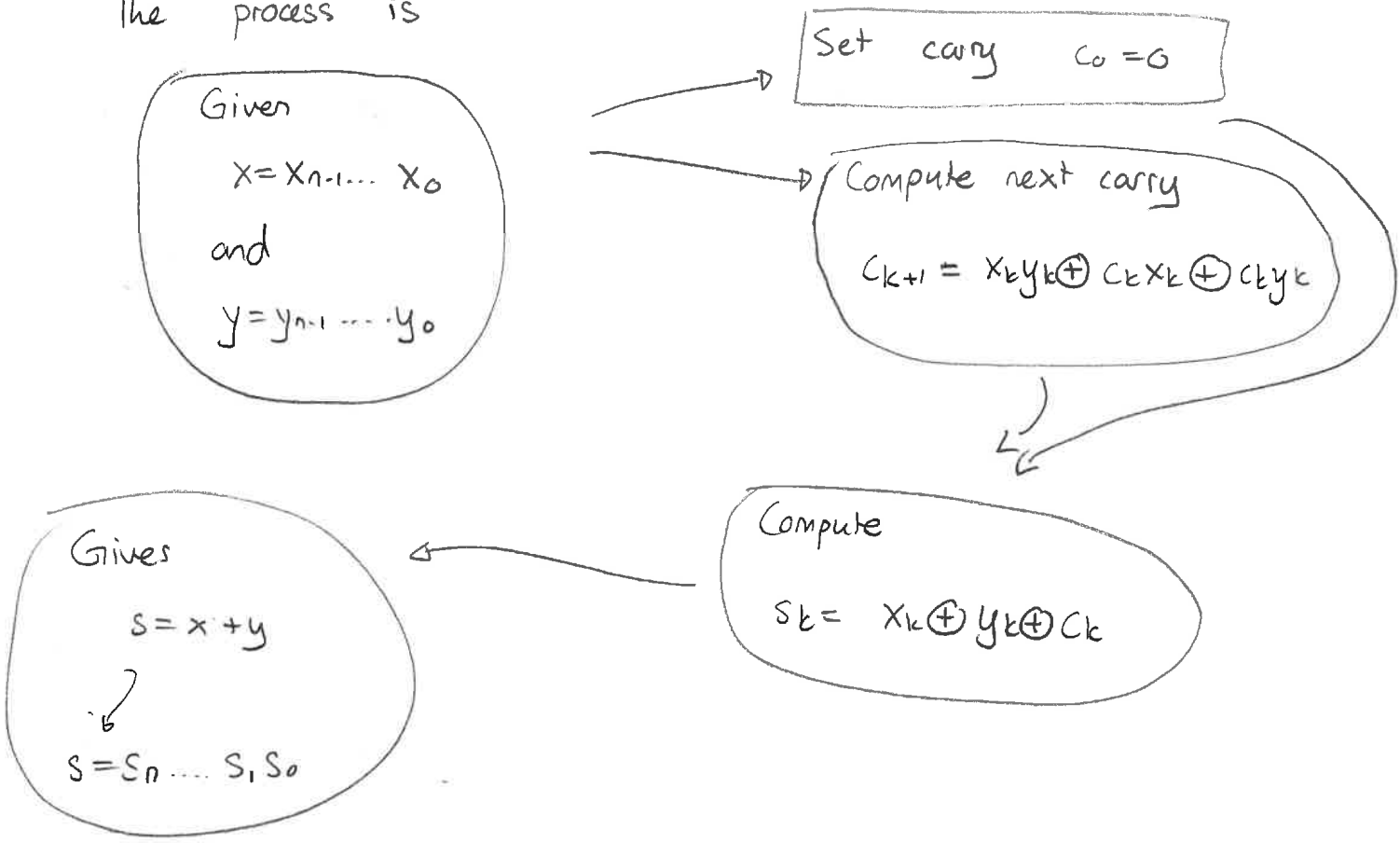
$$S = x + y$$

which has binary representation:

$$S = S_n S_{n-1} \ldots S_2 S_1 S_0$$

The digits of the sum are obtained via a process of bitwise addition that requires repeated computation of carry bits.

The process is



We can see that <u>two essential operations</u> are:

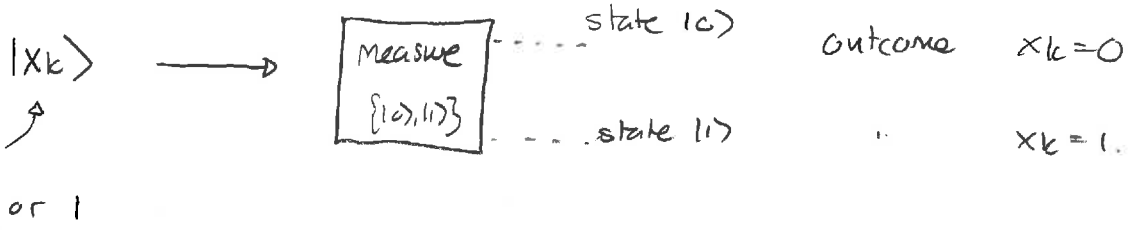
1) bitwise XOR addition $x_k \oplus y_k$

2) bitwise multiplication $x_k y_k$.

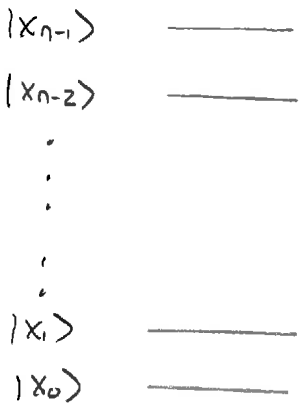
Such function evaluations arise in quantum information processing and we need to establish how they can be implemented using qubits and unitary operations.

Representing classical bits via qubit states

Any classical qubit takes values $x_k = 0$ or $x_k = 1$ and thus it can be represented via qubit states $|0\rangle$ or $|1\rangle$. If we measure in the $\{|0\rangle, |1\rangle\}$ basis then the outcome of the measurement returns the bit value:



Thus a string of bits can be represented by a string of qubits



This is the state

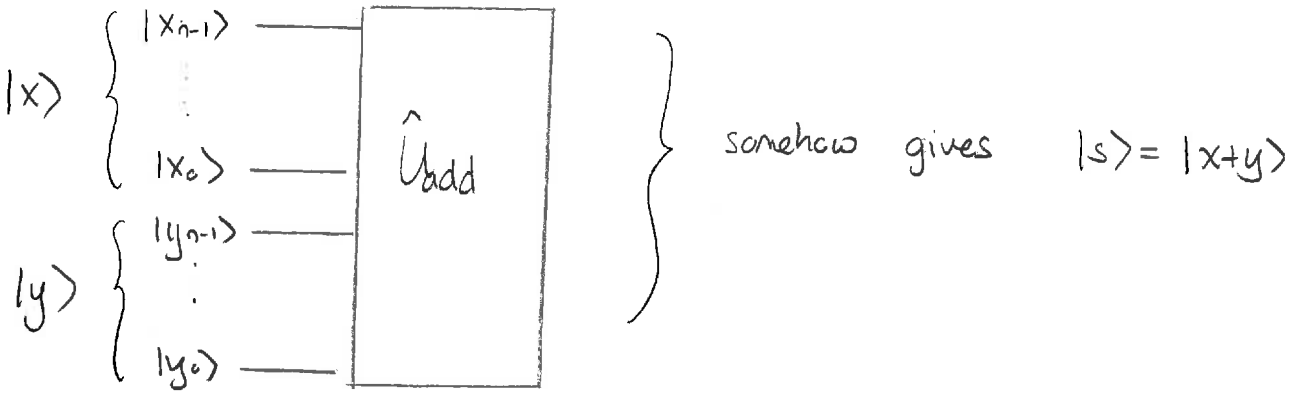
$$|x_{n-1}\rangle|x_{n-2}\rangle \cdots |x_1\rangle|x_0\rangle \equiv \underbrace{|x_{n-1}\, x_{n-2} \cdots x_1 x_0\rangle}_{\text{binary rep}}$$

$$\equiv |x\rangle$$

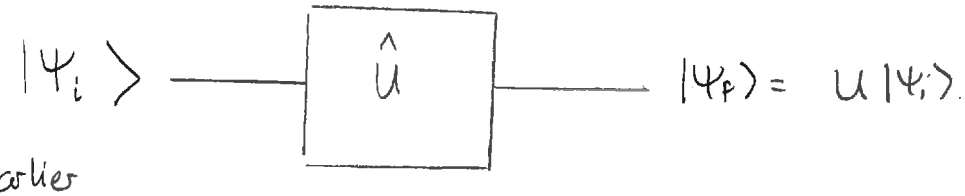
x: decimal rep

Reversible computing

We know that evolution of qubit states is described by a unitary operation. In order to perform any classical function evaluation we need an appropriate unitary operation. For example, to add two n bit numbers:



There is one constraint that has to do with unitary transformations which is that they are reversible. So suppose generally:



earlier

So if we know $\hat{U}$ and $|\psi_i\rangle$ we can obtain $|\psi_f\rangle$. But if we know $|\psi_f\rangle$ can we obtain $|\psi_i\rangle$? To see that this is true consider:

$$\hat{U}^\dagger |\psi_f\rangle = \underbrace{\hat{U}^\dagger \hat{U}}_{\hat{I}} |\psi_i\rangle$$

$$\Rightarrow |\psi_i\rangle = \hat{U}^\dagger |\psi_f\rangle$$

So if we can construct $\hat{U}^\dagger$, then we can reverse the process. This is always true for unitary evolution processes.

Thus we will need to check whether any classical computing is reversible and, if not whether it can be reversed.

We will see that we have to modify classical computations to render them reversible

1 Reversible classical computation

Consider modular addition of two single bits, which is a map from two bits to one bit:

$$(x, y) \mapsto x \oplus y.$$

a) Suppose that $s \equiv x \oplus y$ and you are told that $s = 1$. Is it possible to *uniquely* reconstruct the input (x, y) that gives this output? Is this computation reversible?

b) Consider a modified version of the modular addition defined as a map from two bits to two bits via

$$(x, y) \mapsto (x, x \oplus y).$$

Construct a table with all possible inputs to this function. For each evaluate all possible outputs. Verify that each ouput is uniquely associated with one single input. Is this computation reversible?

Consider bitwise multiplication

$$(x, y) \mapsto xy.$$

c) Is the map $(x, y) \mapsto xy$ reversible?

d) Consider the modified map from two bits to two bits

$$(x, y) \mapsto (x, xy).$$

Is this map reversible?

e) Consider the modified map from three bits to three bits

$$(x, y, 0) \mapsto (x, y, xy).$$

Is this map reversible?

Answer: a) We could have $x=0$ $y=1$ $\rightarrow$ $x \oplus y = s = 1$

$x=1$ $y=0$ $\rightarrow$ $x \oplus y = s = 1$

There are two distinct inputs that give the same output. The computation is not reversible.

b)

(x, y)	$(x, x \oplus y)$
$(0,0)$	$(0, 0)$
$(0,1)$	$(0, 1)$
$(1,0)$	$(1, 1)$
$(1,1)$	$(1, 0)$

Given any output we can determine the input from which it came. This is reversible.

c)

(x, y)	xy
$(0,0)$	0
$(0,1)$	0
$(1,0)$	0
$(1,1)$	1

This is clearly not reversible - several different inputs give 0.

d)

(x, y)	(x, xy)	
$(0,0)$	$(0, 0)$	← not reversible
$(0,1)$	$(0, 0)$	← not reversible
$(1,0)$	$(1, 0)$	
$(1,1)$	$(1, 1)$	

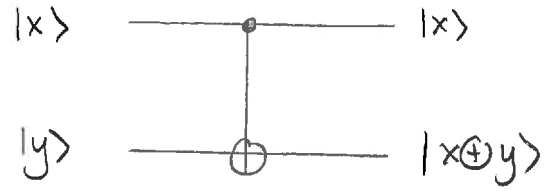
e)

$(x, y, 0)$	(x, y, xy)
$(0,0,0)$	$(0,0,0)$
$(0,1,0)$	$(0,1,0)$
$(1,0,0)$	$(1,0,0)$
$(1,1,0)$	$(1,1,1)$

Yes reversible.

Reversible XOR gate

We have an immediate candidate for a reversible modular addition operation. Consider the controlled-NOT. We can see by supplying all possible inputs that



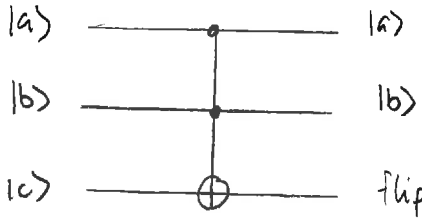
This clearly accomplishes the map

$$(x, y) \rightarrow (x, x \oplus y)$$

So the controlled-NOT can give a classical XOR.

Reversible binary multiplication

The crucial operation needed to implement binary multiplication is the Toffoli gate. This acts on three qubits



We will see that this maps the computational basis as:

$$|a\rangle|b\rangle|c\rangle \rightarrow |a\rangle|b\rangle|c \oplus ab\rangle$$

2 Toffoli gate

a) Construct a truth table for a Toffoli gate.

b) Show that the Toffoli gate maps

$$|a\rangle\,|b\rangle\,|c\rangle \mapsto |a\rangle\,|b\rangle\,|c \oplus ab\rangle\,.$$

c) Construct a circuit using a Toffoli gate so that it outputs ab.

d) Construct a Toffoli gate so that it outputs $\mathsf{NAND}(a, b)$.

Answer: a)

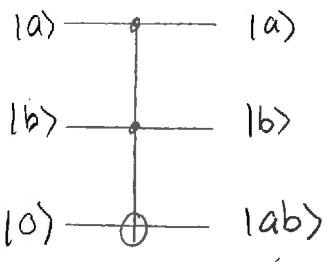
a	b	c	output
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

b)

ab	ab⊕c
0	0
0	1
0	0
0	1
0	0
0	1
1	1
1	0

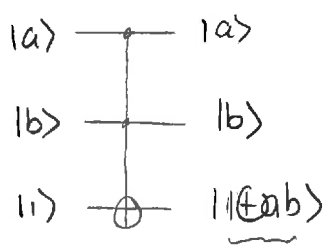
✓

c)



↳ desired output

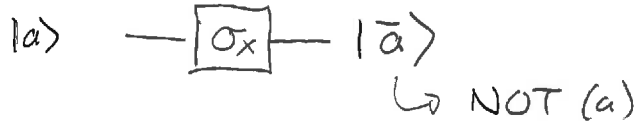
d)



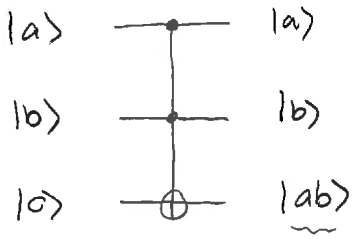
NAND(a,b)

Since the NAND gate is universal we can use the Toffoli gate to construct a reversible version of NAND and therefore any gate.

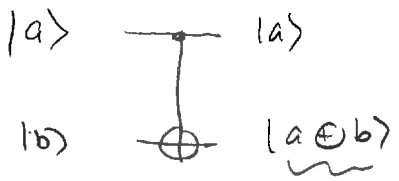
Occasionally there are shortcuts.

1) 

↳ NOT (a)

2) 

AND (a,b)

3) 

XOR (a,b)

Combining these gates ultimately allows any possible function evaluation.