

# Boolean Quantum Circuit

By: Ahmed Saad El Fiky

# Boolean Function

## Boolean Functions:

$$f: \{0,1\}^n \rightarrow \{0,1\}$$

e.g.:

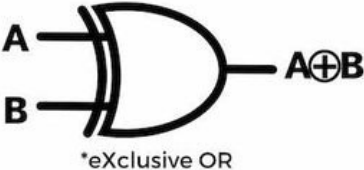
$$f(x_0, x_1, x_2) = \bar{x}_0x_1 + x_0x_2$$

**Aim:** Implement a Quantum Circuit for the given Boolean function.

## Properties of XOR:

- $x \oplus 0 = x$
- $x \oplus 1 = \bar{x}$
- $x \oplus x = 0$

XOR Truth Table

			2 input XOR gate		
A	B	A⊕B	A	B	A⊕B
0	0	0	0	0	0
0	1	1	0	1	1
1	0	1	1	0	1
1	1	0	1	1	0

$x_0$	$x_1$	$x_2$	$f(x_0, x_1, x_2)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

# CNOT & Toffoli gate (CCNOT);

Properties of XOR:

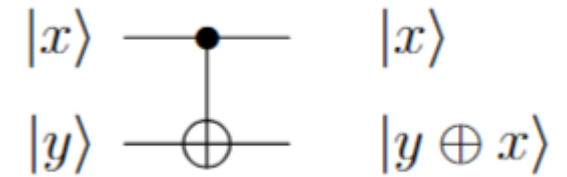
- $x \oplus 0 = x$
- $x \oplus 1 = \bar{x}$
- $x \oplus x = 0$

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

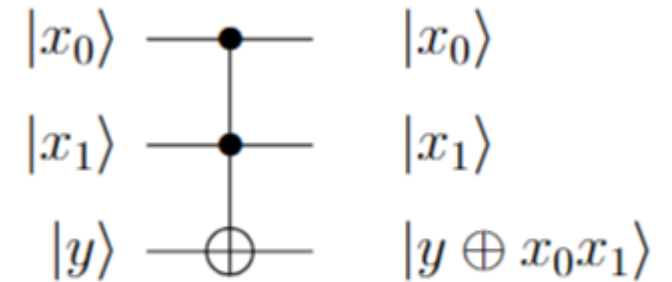
$$\bullet \quad U_{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}; \quad \begin{array}{l} |11\rangle \xrightarrow{\text{maps}} |10\rangle \\ |10\rangle \xrightarrow{\text{maps}} |11\rangle \\ \text{Otherwise do nothing.} \end{array}$$

$$\bullet \quad U_{CCNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad \begin{array}{l} |111\rangle \xrightarrow{\text{maps}} |110\rangle \\ |110\rangle \xrightarrow{\text{maps}} |111\rangle \\ \text{Otherwise do nothing.} \end{array}$$

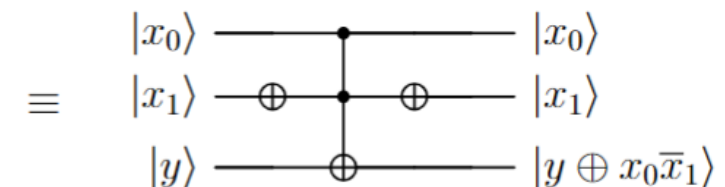
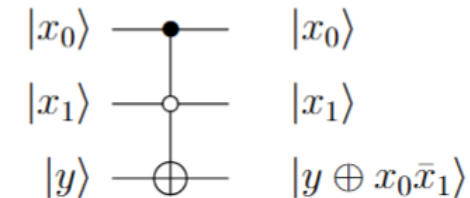
CNOT Gate



Toffoli Gate



Generalized Toffoli Gate



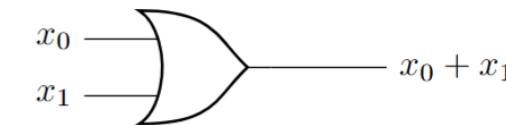
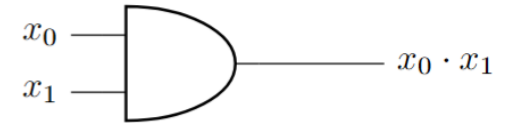
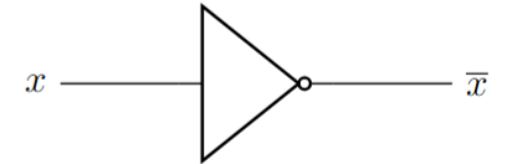
# Classical Logic gates:

- Any Boolean function  $f: \{0,1\}^n \rightarrow \{0,1\}$  can be implemented classically in a logic gate circuit using the universal gate set (AND, OR, NOT).
- NAND gate is a universal classical gate, currently used in processors.

$x$	$\bar{x}$
0	1
1	0

$x_0$	$x_1$	$x_0 \cdot x_1$
0	0	0
0	1	0
1	0	0
1	1	1

$x_0$	$x_1$	$x_0 + x_1$
0	0	0
0	1	1
1	0	1
1	1	1

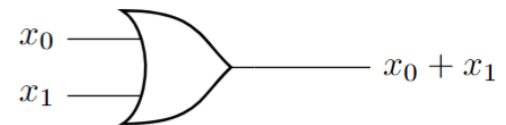
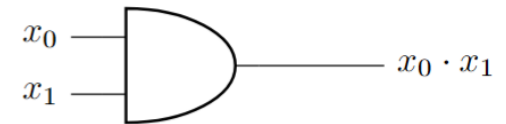
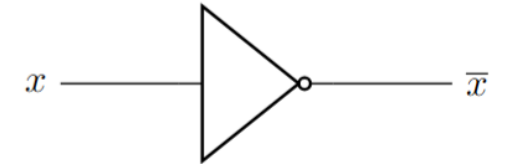


NAND	A	B	Output
	0	0	1
	1	0	1
	0	1	1
	1	1	0



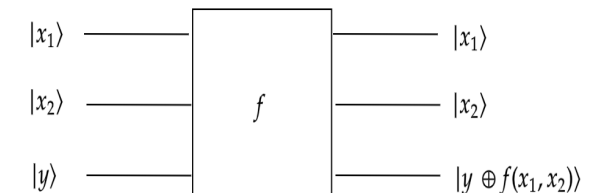
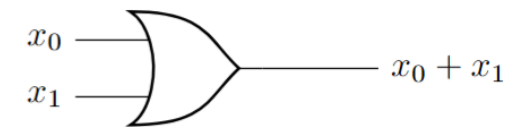
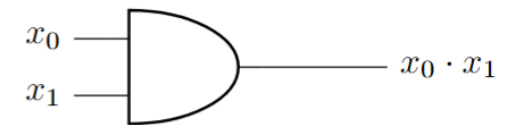
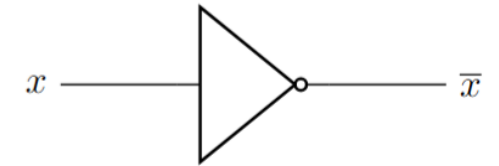
# Reversible Computing

- By looking at the output column of the tables of the **AND** and **OR** gates, we can not guess what the input is. We can say that the information or the entropy is lost by applying those gates and those operations are called **irreversible**.
  - Irreversible computation dissipates heat to the environment.
- On the other hand, this is not the case for the **NOT** gate as the input can be constructed by looking at the output. Such gates are called reversible and a computation which consists of only reversible operations is called a **reversible computation**.



# Reversible Computing

- A set of gates is called **universal** if it is possible to implement any other gate using the gates in the set. Theoretically, it is possible to build a universal computer that only uses reversible gates. For instance, **AND** and **NOT** gates or the Toffoli gate (**CCNOT**) itself are universal sets of gates for classical computing.
  - Note that since **CCNOT** is also a quantum gate, we conclude that a quantum computer can simulate any classical operation.
- Since quantum computing is reversible according to the laws of physics **AND** and **NOT** gates should be implemented in a reversible way as well. The idea is to create a 3-qubit circuit, which does not modify the input bits and writes the output to the third bit. When the output bit is set to 0, then you exactly get the same output.

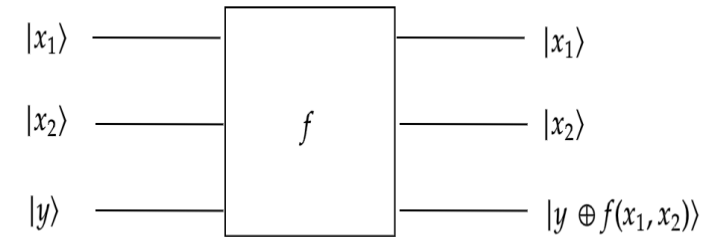
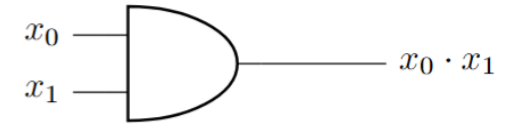
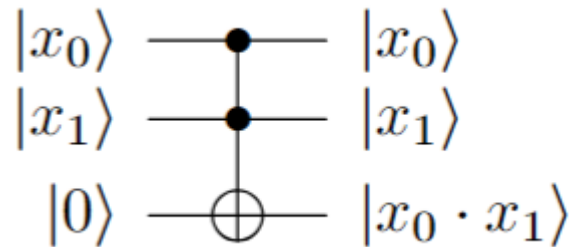


# Example:

Complete the following table that corresponds to reversible AND gate, where  $|x_1\rangle$  and  $|x_2\rangle$  are the inputs of the AND gate and the  $|y\rangle = |0\rangle$  is the output.

Which three-qubit quantum gate can we use to implement the AND operator in a reversible manner?

IN			OUT		
$x_0$	$x_1$	$y$	$x_0$	$x_1$	$x_0 \cdot x_1$
0	0	0	0	0	0
0	1	0	0	1	0
1	0	0	1	0	0
1	1	0	1	1	1



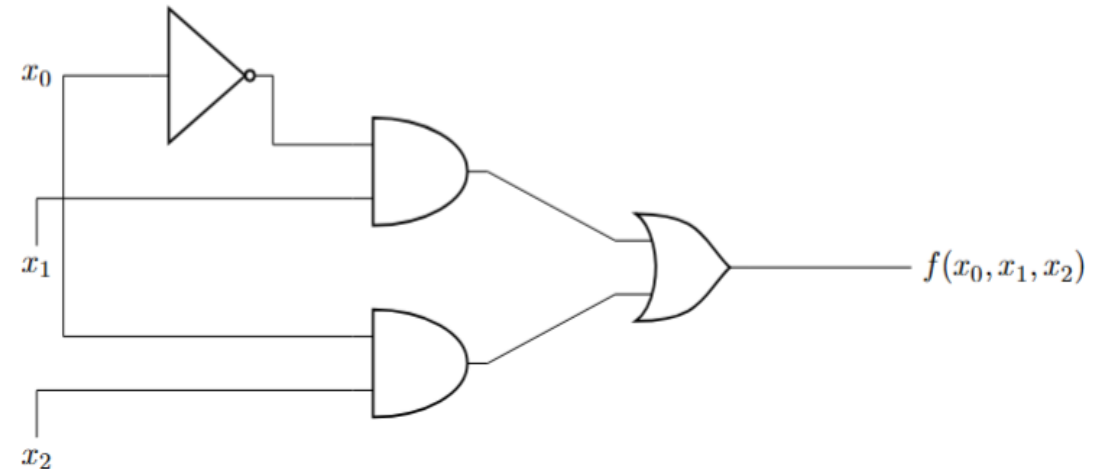
We can use CCNOT (Toffoli) gate.

# Boolean Quantum Circuit

- **Given:**

$$f(x_0, x_1, x_2) = \bar{x}_0x_1 + x_0x_2$$

- Implement the reversible quantum circuit.



- Reed-Muller logic is an alternative representation of Boolean functions that **uses Exclusive OR (XOR,  $\oplus$ )** and **AND** operations, implemented using **CNOT** gates in quantum computing, instead of the traditional sum-of-products (SOP) or product-of-sums (POS) forms, which rely on **AND**, **OR**, and **NOT** logic gates.



# Reed-Muller logic



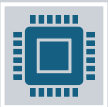
Reed-Muller logic expresses Boolean functions using XOR ( $\oplus$ ) and AND ( $\cdot$ ) instead of AND-OR-NOT.



Provides a canonical form using mod-2 sums.



Efficient for error detection and cryptography.



Reed-Muller logic is ideal for quantum computing because:

XOR maps directly to CNOT gates.

Supports reversible computation.

Reduces quantum circuit complexity.

# Boolean Quantum Circuit

$$f(x_0, x_1, x_2) = \bar{x}_0x_1 + x_0x_2$$

Step 1:

- Express the Function as a Truth Table
- List all possible inputs  $(x_0, x_1, x_2)$  and compute  $f(x_0, x_1, x_2)$ .
- Write the sum of all 1 output terms.

$x_0$	$x_1$	$x_2$	$f(x_0, x_1, x_2)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

## Step 2: Find the Reed-Muller Expansion

- Write the Boolean function  $f$  in its Normal Form “Sum of Product (SOP) form”:

$$f(x_0, x_1, x_2) = \bar{x}_0x_1\bar{x}_2 + \bar{x}_0x_1x_2 + x_0\bar{x}_1x_2 + x_0x_1x_2$$

- Write the Boolean function  $f$  in its Reed-Muller Expansion form by replacing OR operator by XOR operator – only allowed in Normal form:

$$f(x_0, x_1, x_2) = \bar{x}_0x_1\bar{x}_2 \oplus \bar{x}_0x_1x_2 \oplus x_0\bar{x}_1x_2 \oplus x_0x_1x_2$$

# Step 3: Quantum Circuit Design

$$f(x_0, x_1, x_2) = \bar{x}_0 x_1 \bar{x}_2 \oplus \bar{x}_0 x_1 x_2 \oplus x_0 \bar{x}_1 x_2 \oplus x_0 x_1 x_2$$

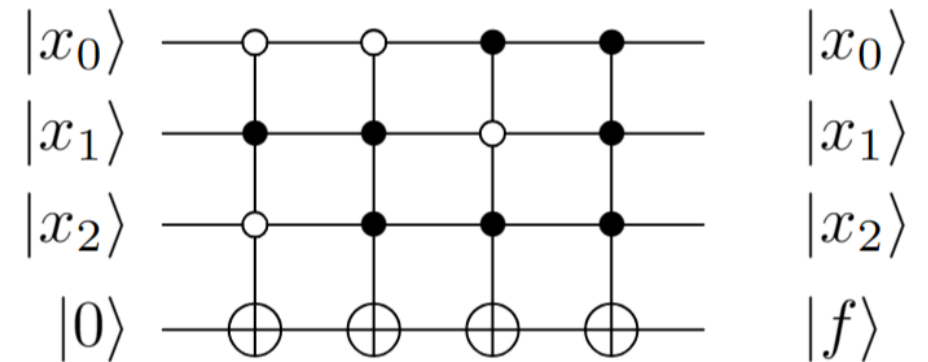
To implement this function in a quantum circuit:

**1. Input qubits**  $x_0, x_1, x_2$  .

**2. Auxiliary qubit:**  $f$  (initialized to  $|0\rangle$ ).

**3. Operations used:**

1. A **negation** ( $\bar{x}_0$ )  $\rightarrow$  Implemented using an **X** gate “**Open control**”.
2. **XOR** ( $\oplus$ )  $\rightarrow$  Implemented using **CNOT (CX)** gates.
3. **AND** ( $\cdot$ )  $\rightarrow$  Implemented using **Toffoli (CCX)** gates.



## Step 4: Find the Reed-Muller Canonical Form

- Use the XOR properties to reduce the complexity:

$$x \oplus 0 = x \quad \& \quad x \oplus 1 = \bar{x} \quad \& \quad x \oplus x = 0$$

$$f(x_0, x_1, x_2) = \bar{x}_0 x_1 \bar{x}_2 \oplus \bar{x}_0 x_1 x_2 \oplus x_0 \bar{x}_1 x_2 \oplus x_0 x_1 x_2$$

$$f(x_0, x_1, x_2) = (x_0 \oplus 1)x_1(x_2 \oplus 1) \oplus (x_0 \oplus 1)x_1 x_2 \oplus x_0(x_1 \oplus 1)x_2 \oplus x_0 x_1 x_2$$

$$= \cancel{x_0 x_1 x_2} \oplus x_0 x_1 \oplus \cancel{x_1 x_2} \oplus x_1 \oplus \cancel{x_0 x_1 x_2} \oplus \cancel{x_1 x_2} \oplus \cancel{x_0 x_1 x_2} \oplus x_0 x_2 \oplus \cancel{x_0 x_1 x_2}$$

$$f(x_0, x_1, x_2) = x_0 x_1 \oplus x_0 x_2 \oplus x_1$$

➤ This form call Positive (Zero) Polarity Reed-Muller Expansion.

# Step 5: Quantum Circuit of Canonical Form

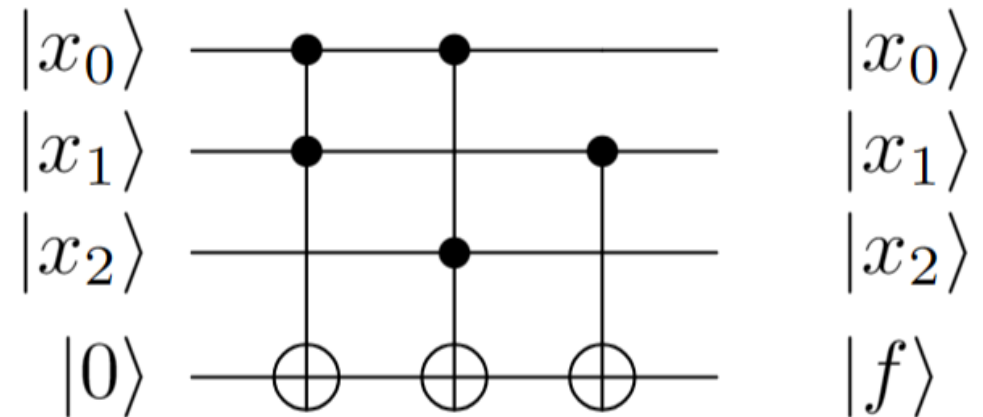
$$f(x_0, x_1, x_2) = x_0x_1 \oplus x_0x_2 \oplus x_1$$

To implement this function in a quantum circuit:

1. **Input qubits**  $x_0, x_1, x_2$  .
2. **Auxiliary qubit:**  $f$  (initialized to  $|0\rangle$ ).
3. **Operations used:**
  1. **XOR ( $\oplus$ )**  $\rightarrow$  Implemented using **CNOT (CX)** gates.
  2. **AND ( $\cdot$ )**  $\rightarrow$  Implemented using **Toffoli (CCX)** gates.

## Quantum Circuit Breakdown:

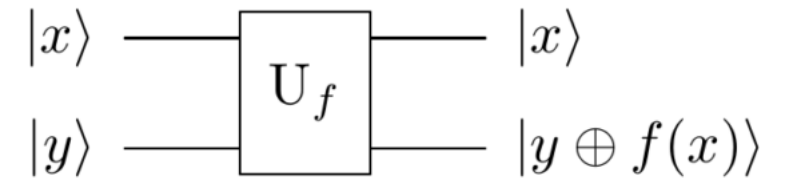
1. Compute  $x_0x_1$  using a **CCX gate** and store it in an auxiliary qubit.
2. Compute  $x_1 \oplus (x_0x_1)$  using a **CNOT gate**.





# Implementing any Boolean Function

- We can implement a set of universal gates on a quantum computer, provided that we make them "reversible", we can say that it is possible to implement any Boolean function  $f: \{0,1\}^n \rightarrow \{0,1\}$ . So given any Boolean function  $f(x)$ , we propose that the following circuit will implement it in a quantum computer.



- Here  $U_f$ , the corresponding quantum operator, is defined as follows:

$$U_f: |x\rangle|y\rangle \mapsto |x\rangle|y \oplus f(x)\rangle$$

- The symbol  $\oplus$  denotes bitwise addition modulo 2 (*XOR*). This mapping is reversible although  $f$  might not be invertible.
- $x \oplus x = 0$  for any bit  $x$  and  $\oplus$  operation is associative.

# Oracle model of computation

- An oracle in the quantum circuit model is a unitary operator  $U_f$  that implements a Boolean function on data and target registers without revealing its internal structure, often called a “black box.”
- Suppose that your friend picks such a function  $f$  and you try to guess whether it is constant or balanced. You are only allowed to ask questions like “What is  $f(0)$ ?”
- Each question you ask, is a query to the function  $f$ . In quantum computing, many algorithms rely on this oracle model of computation and the aim is to solve some problem making as minimum queries as possible.

“We can evaluate it for an input by making queries but we can't look inside.”

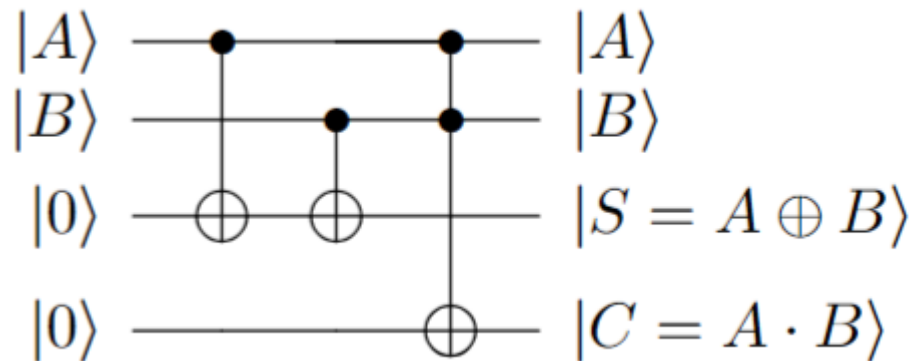
# Example: One-bit Half Adder

- Adding two one-bit numbers:

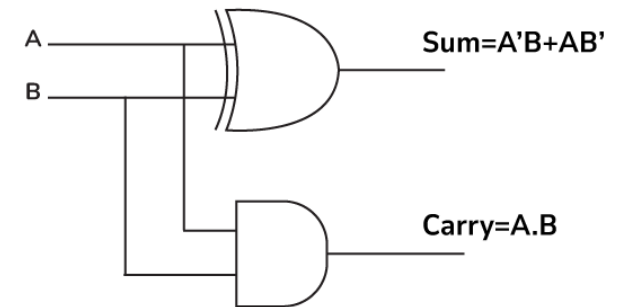
$$S = A \oplus B$$

$$C = A \cdot B$$

- Corresponding quantum circuit:



Half Adder



Truth Table

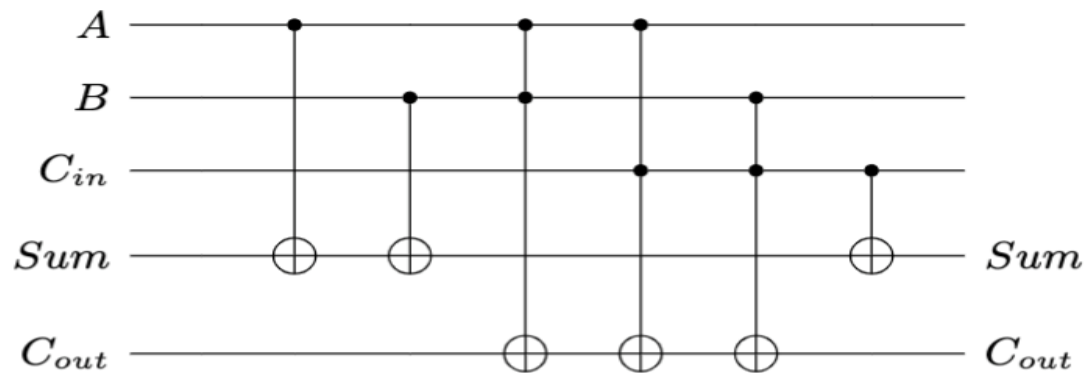
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

# Example: One-bit Full Adder

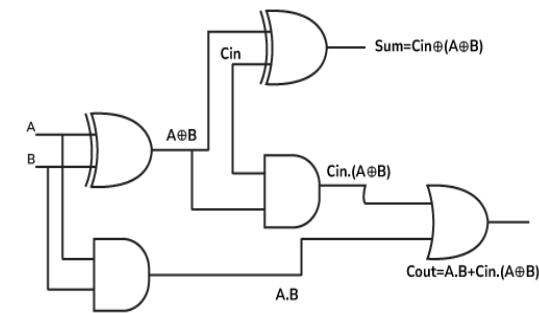
- Adding two one-bit numbers plus Carry in:

$$\begin{aligned} \text{Sum} &= A \oplus B \oplus C \\ C_{out} &= A \cdot B + A \cdot C + B \cdot C \\ &= AB \oplus AC \oplus BC \end{aligned}$$

- Corresponding quantum circuit:



Full Adder



Inputs			Outputs	
A	B	C	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



# Thank you

Ahmed Saad El Fiky

Questions?