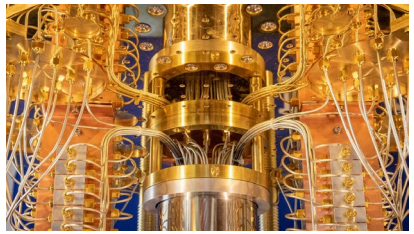
# Lectures 26-27: Classical Quantum Computation

CS 401: Quantum Computing
Dr. Kell, Spring 2023

# Classical Computation on Quantum Computer?

**So far:** we've seen for certain problems:



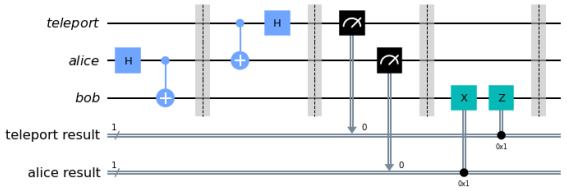
Quantum Computation

>

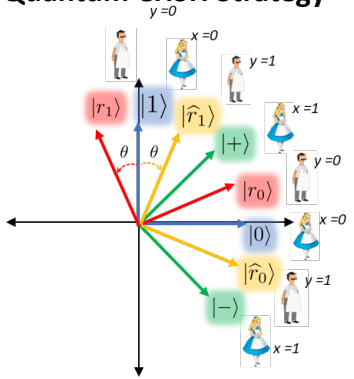
Classical Computation

**Examples**

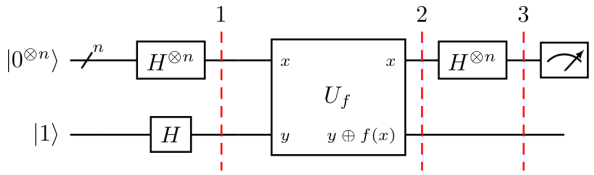
**Quantum Teleportation**




**Quantum CHSH Strategy**

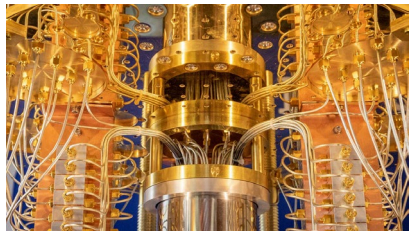



**Deutsch-Jozsa**



# Classical Computation on Quantum Computer?

**So far:** we've seen for certain problems:



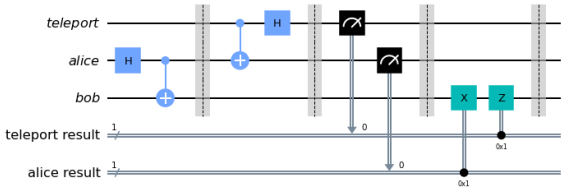
Quantum Computation

>

Classical Computation

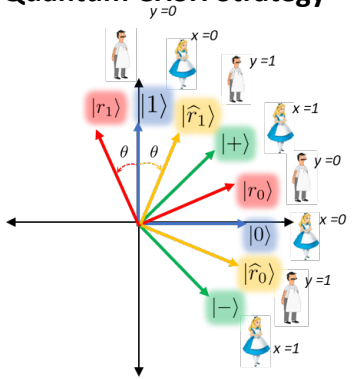
**Examples**

**Quantum Teleportation**

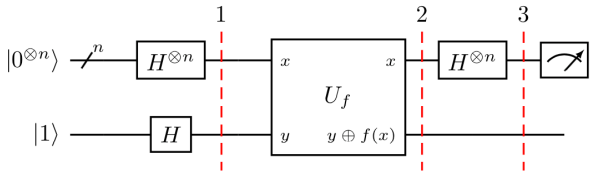



Classically Impossible Information Transfer

**Quantum CHSH Strategy**



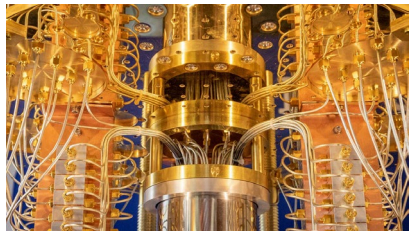

**Deutsch-Jozsa**




Faster algorithm

# Classical Computation on Quantum Computer?

**So far:** we've seen for certain problems:

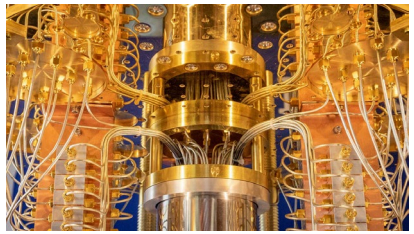


Quantum Computation

>

Classical Computation

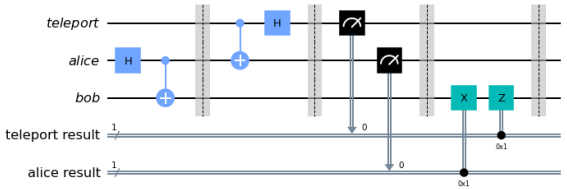
**Natural Question:** Is this true for all problems?



≥

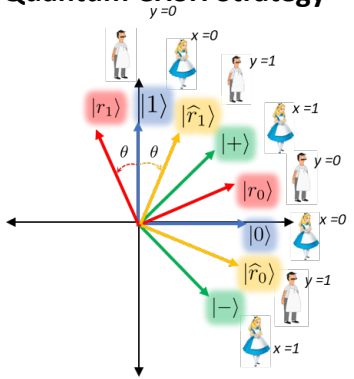
**Examples**

**Quantum Teleportation**

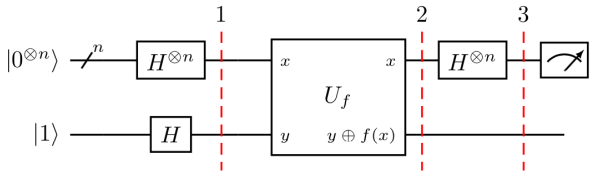



Classically Impossible Information Transfer

**Quantum CHSH Strategy**



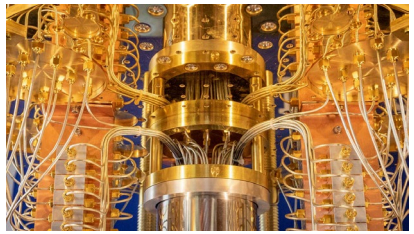

**Deutsch-Jozsa**




Faster algorithm

# Classical Computation on Quantum Computer?

**So far:** we've seen for certain problems:

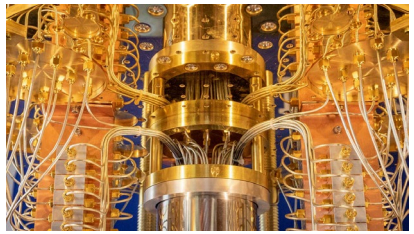


Quantum Computation

$>$

Classical Computation

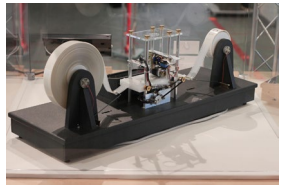
**Natural Question:** Is this true for all problems?

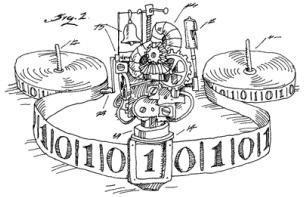


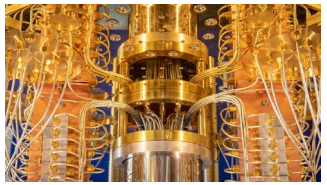
$\geq$

**In other words:**

- Are all Turing computable problems also solvable by a quantum computer?

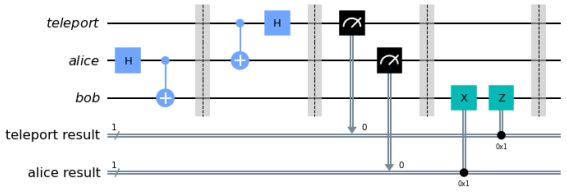






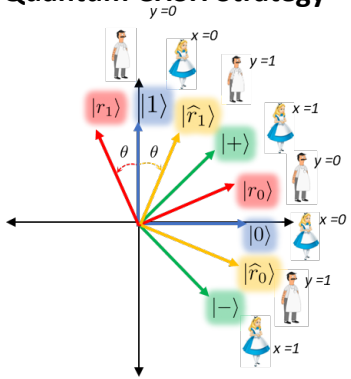
**Examples**

**Quantum Teleportation**

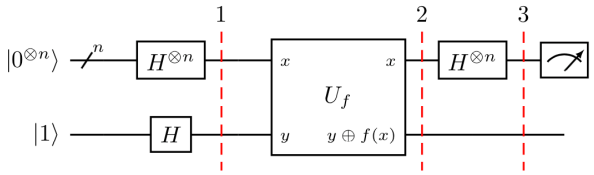



Classically Impossible Information Transfer

**Quantum CHSH Strategy**



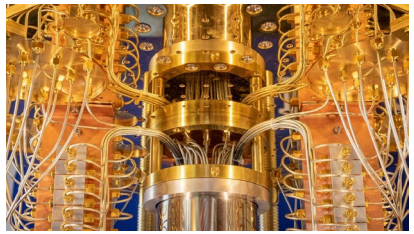

**Deutsch-Jozsa**




Faster algorithm

# Classical Computation on Quantum Computer?

**So far:** we've seen for certain problems:

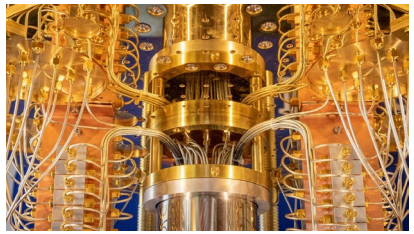


Quantum Computation

$>$

Classical Computation

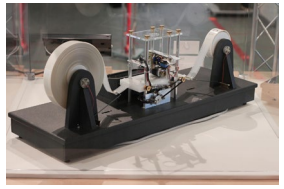
**Natural Question:** Is this true for all problems?

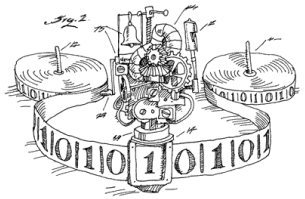


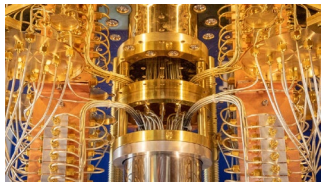
$\geq$

**In other words:**

- Are all Turing computable problems also solvable by a quantum computer?







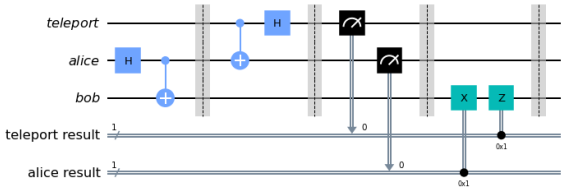
- Can all poly-time solvable problems also be solved on a QC in poly-time?

Solves problem in time T

Can solve same problem in O(T) time.

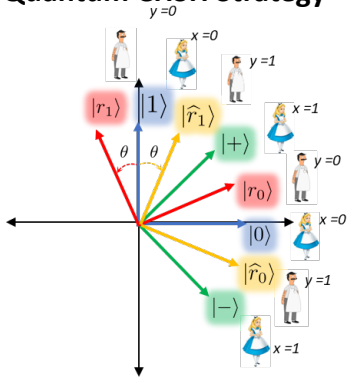
**Examples**

**Quantum Teleportation**

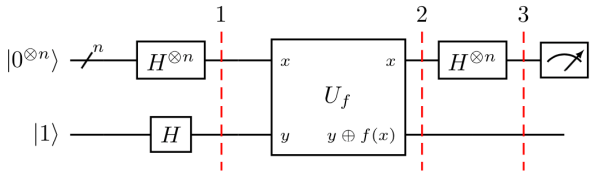



Classically Impossible Information Transfer

**Quantum CHSH Strategy**



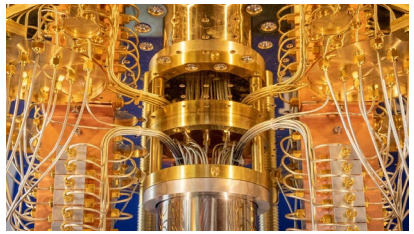

**Deutsch-Jozsa**




Faster algorithm

# Classical Computation on Quantum Computer?

**So far:** we've seen for certain problems:



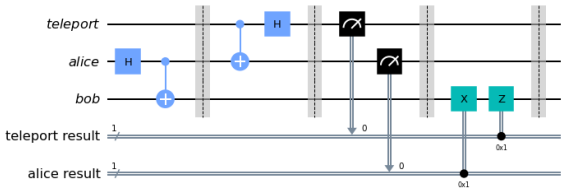
Quantum Computation

>

Classical Computation

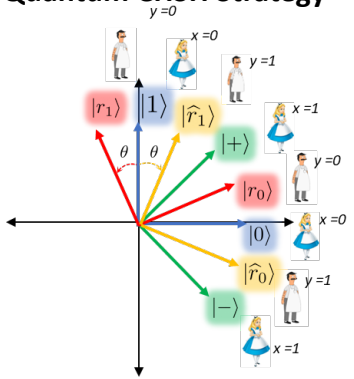
**Examples**

**Quantum Teleportation**

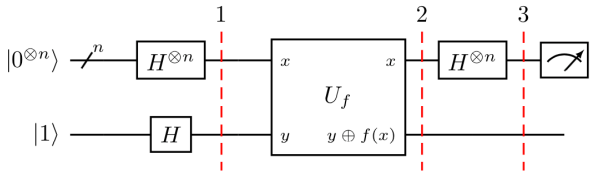



Classically Impossible Information Transfer

**Quantum CHSH Strategy**

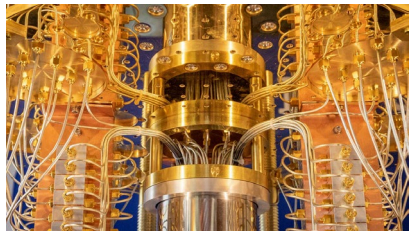


**Deutsch-Jozsa**



Faster algorithm

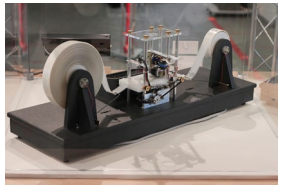
**Natural Question:** Is this true for all problems?

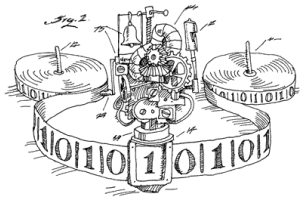


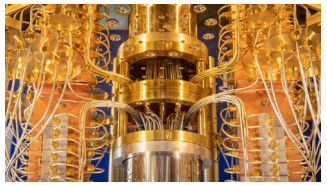
≥

**In other words:**

- Are all Turing computable problems also solvable by a quantum computer?







- Can all poly-time solvable problems also be solved on a QC in poly-time?

Solves problem in time T

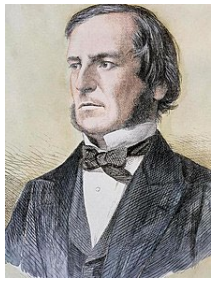
Can solve same problem in O(T) time.

**Answer:** Yes! (goal of today's lecture is to see how)

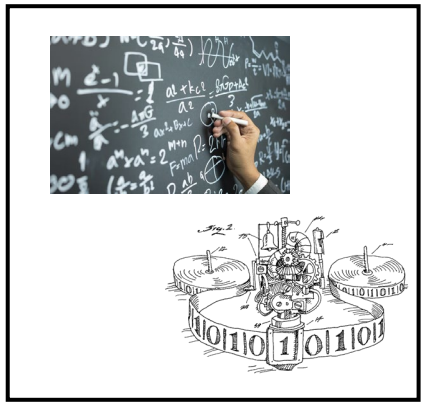
(Pictures and Diagrams Courtesy of Wikipedia)

# Step 1: Universal Classical Logic Gate

Leibnitz (1705)



Boole (1847)



Arithmetic/Logic/Computation

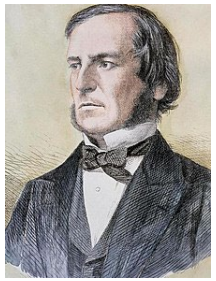
| Annulment | Complement | Associative |
|---|---|---|
| $A \wedge 0 = 0$ | $A \vee \neg A = 1$ | $(A \vee B) \vee C = A \vee (B \vee C)$ |
| $A \vee 1 = 1$ | $A \wedge \neg A = 0$ | $(A \wedge B) \wedge C = A \wedge (B \wedge C)$ |
| **Identity** | **Double Negation** | **Commutative** |
| $A \wedge 1 = A$ | $\neg(\neg A) = A$ | $A \vee B = B \vee A$ |
| $A \vee 0 = A$ | | $A \wedge B = B \wedge A$ |
| **Idempotent** | **De Morgan's** | **Distributive** |
| $A \vee A = A$ | $\neg(A \wedge B) = \neg A \vee \neg B$ | $A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$ |
| $A \wedge A = A$ | $\neg(A \vee B) = \neg A \wedge \neg B$ | $A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C)$ |
| | | **Absorptive** |
| | | $A \vee (A \wedge B) = A$ |
| | | $A \wedge (A \vee B) = A$ |

Boolean Algebra

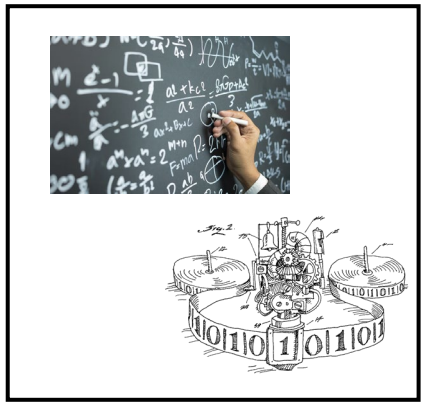
(Pictures and Diagrams Courtesy of Wikipedia)

# Step 1: Universal Classical Logic Gate

Leibnitz (1705)



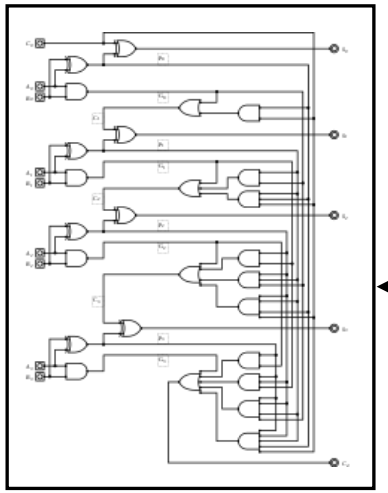
Boole (1847)



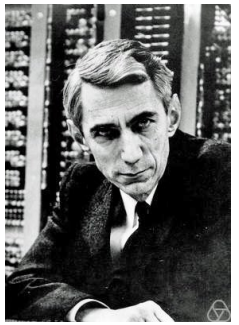
Arithmetic/Logic/Computation

| | | |
|---|---|---|
| Annulment | Complement | Associative |
| $A \wedge 0 = 0$ | $A \vee \neg A = 1$ | $(A \vee B) \vee C = A \vee (B \vee C)$ |
| $A \vee 1 = 1$ | $A \wedge \neg A = 0$ | $(A \wedge B) \wedge C = A \wedge (B \wedge C)$ |
| Identity | Double Negation | Commutative |
| $A \wedge 1 = A$ | $\neg(\neg A) = A$ | $A \vee B = B \vee A$ |
| $A \vee 0 = A$ | | $A \wedge B = B \wedge A$ |
| Idempotent | De Morgan's | Distributive |
| $A \vee A = A$ | $\neg(A \wedge B) = \neg A \vee \neg B$ | $A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$ |
| $A \wedge A = A$ | $\neg(A \vee B) = \neg A \wedge \neg B$ | $A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C)$ |
| | | Absorptive |
| | | $A \vee (A \wedge B) = A$ |
| | | $A \wedge (A \vee B) = A$ |

Boolean Algebra



Logic Gates

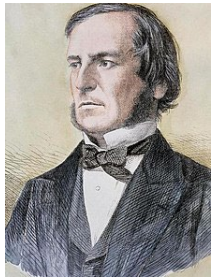


Shannon (1936)

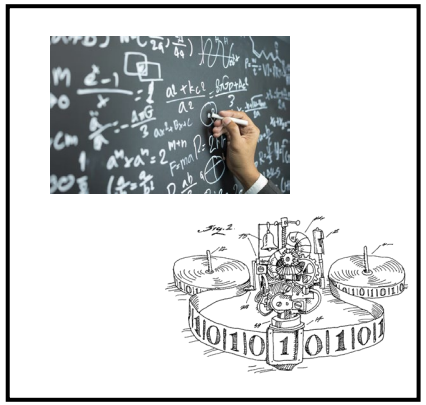
(Pictures and Diagrams Courtesy of Wikipedia)

# Step 1: Universal Classical Logic Gate

Leibnitz (1705)



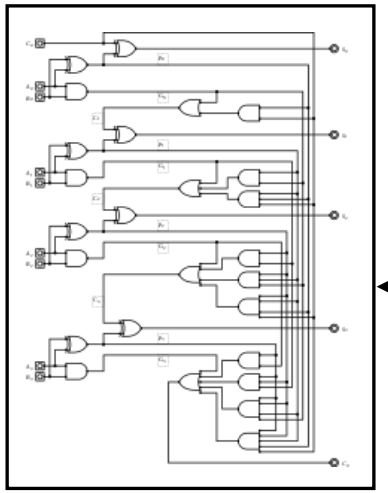
Boole (1847)



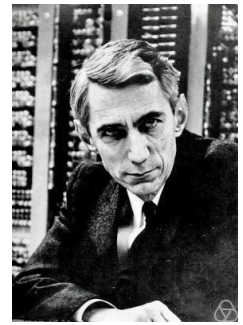
Arithmetic/Logic/Computation

| | | |
|---|---|---|
| Annulment | Complement | Associative |
| $A \wedge 0 = 0$ | $A \vee \neg A = 1$ | $(A \vee B) \vee C = A \vee (B \vee C)$ |
| $A \vee 1 = 1$ | $A \wedge \neg A = 0$ | $(A \wedge B) \wedge C = A \wedge (B \wedge C)$ |
| Identity | Double Negation | Commutative |
| $A \wedge 1 = A$ | $\neg(\neg A) = A$ | $A \vee B = B \vee A$ |
| $A \vee 0 = A$ | | $A \wedge B = B \wedge A$ |
| Idempotent | De Morgan's | Distributive |
| $A \vee A = A$ | $\neg(A \wedge B) = \neg A \vee \neg B$ | $A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$ |
| $A \wedge A = A$ | $\neg(A \vee B) = \neg A \wedge \neg B$ | $A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C)$ |
| | | Absorptive |
| | | $A \vee (A \wedge B) = A$ |
| | | $A \wedge (A \vee B) = A$ |

Boolean Algebra



Logic Gates



Shannon (1936)

## NOT


| Input | Output |
|---|---|
| A | Q |
| 0 | 1 |
| 1 | 0 |

## AND


| Input | | Output |
|---|---|---|
| A | B | Q |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

## OR


| Input | | Output |
|---|---|---|
| A | B | Q |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

## XOR


| Input | | Output |
|---|---|---|
| A | B | Q |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

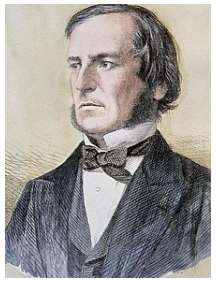
## NAND


| Input | | Output |
|---|---|---|
| A | B | Q |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

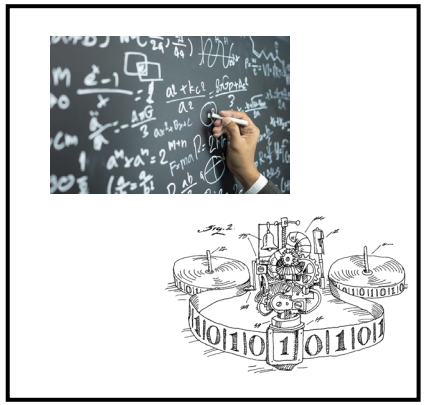
(Pictures and Diagrams Courtesy of Wikipedia)

# Step 1: Universal Classical Logic Gate

Leibnitz (1705)



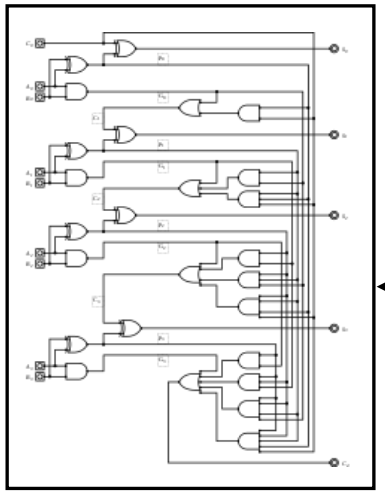
Boole (1847)



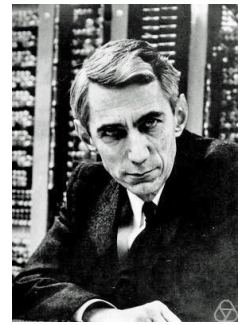
Arithmetic/Logic/Computation

| | | |
|---|---|---|
| **Annulment** | **Complement** | **Associative** |
| $A \wedge 0 = 0$ | $A \vee \neg A = 1$ | $(A \vee B) \vee C = A \vee (B \vee C)$ |
| $A \vee 1 = 1$ | $A \wedge \neg A = 0$ | $(A \wedge B) \wedge C = A \wedge (B \wedge C)$ |
| **Identity** | **Double Negation** | **Commutative** |
| $A \wedge 1 = A$ | $\neg(\neg A) = A$ | $A \vee B = B \vee A$ |
| $A \vee 0 = A$ | | $A \wedge B = B \wedge A$ |
| **Idempotent** | **De Morgan's** | **Distributive** |
| $A \vee A = A$ | $\neg(A \wedge B) = \neg A \vee \neg B$ | $A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$ |
| $A \wedge A = A$ | $\neg(A \vee B) = \neg A \wedge \neg B$ | $A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C)$ |
| | | **Absorptive** |
| | | $A \vee (A \wedge B) = A$ |
| | | $A \wedge (A \vee B) = A$ |

Boolean Algebra



Logic Gates



Shannon (1936)

## NOT


| Input | Output |
|---|---|
| A | Q |
| 0 | 1 |
| 1 | 0 |


## AND


| Input | | Output |
|---|---|---|
| A | B | Q |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |


## OR


| Input | | Output |
|---|---|---|
| A | B | Q |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |


## XOR


| Input | | Output |
|---|---|---|
| A | B | Q |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |


## NAND

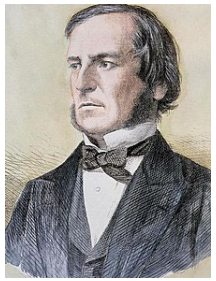

| Input | | Output |
|---|---|---|
| A | B | Q |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Universal Gate: all other gates can be expressed using NANDs

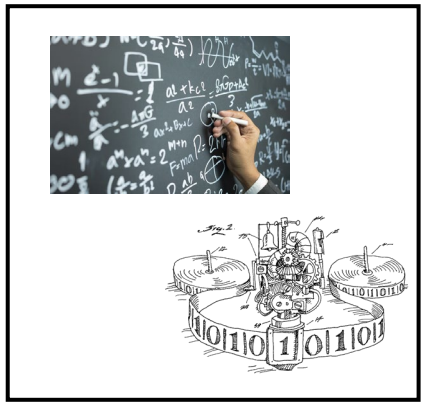
(Pictures and Diagrams Courtesy of Wikipedia)

# Step 1: Universal Classical Logic Gate

Leibnitz (1705)



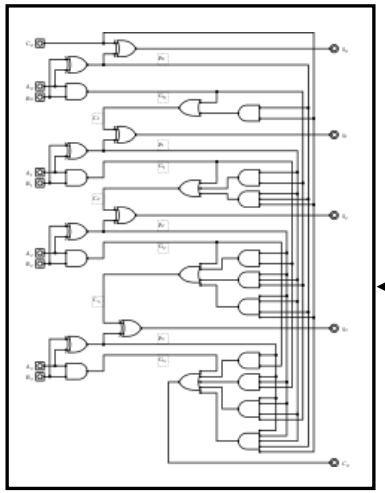
Boole (1847)



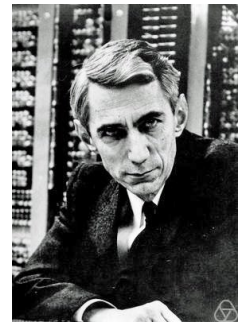
Arithmetic/Logic/Computation

| | | |
|---|---|---|
| **Annulment** | **Complement** | **Associative** |
| $A \wedge 0 = 0$ | $A \vee \neg A = 1$ | $(A \vee B) \vee C = A \vee (B \vee C)$ |
| $A \vee 1 = 1$ | $A \wedge \neg A = 0$ | $(A \wedge B) \wedge C = A \wedge (B \wedge C)$ |
| **Identity** | **Double Negation** | **Commutative** |
| $A \wedge 1 = A$ | $\neg(\neg A) = A$ | $A \vee B = B \vee A$ |
| $A \vee 0 = A$ | | $A \wedge B = B \wedge A$ |
| **Idempotent** | **De Morgan's** | **Distributive** |
| $A \vee A = A$ | $\neg(A \wedge B) = \neg A \vee \neg B$ | $A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$ |
| $A \wedge A = A$ | $\neg(A \vee B) = \neg A \wedge \neg B$ | $A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C)$ |
| | | **Absorptive** |
| | | $A \vee (A \wedge B) = A$ |
| | | $A \wedge (A \vee B) = A$ |

Boolean Algebra



Logic Gates



Shannon (1936)

## NOT


| Input | Output |
|---|---|
| A | Q |
| 0 | 1 |
| 1 | 0 |


## AND


| Input | | Output |
|---|---|---|
| A | B | Q |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |


## OR


| Input | | Output |
|---|---|---|
| A | B | Q |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |


## XOR


| Input | | Output |
|---|---|---|
| A | B | Q |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

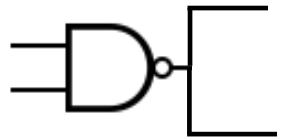

## NAND


| Input | | Output |
|---|---|---|
| A | B | Q |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Universal Gate: all other gates can be expressed using NANDs

## Practice Exercise

Show that NOT, AND, OR, can XOR can be implemented using only NAND gates.



(permitting you can 'fan out' wires)

# Solutions

**NAND**


| Input | | Output |
|---|---|---|
| A | B | Q |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

**NOT**


| Input | Output |
|---|---|
| A | Q |
| 0 | 1 |
| 1 | 0 |

**AND**


| Input | | Output |
|---|---|---|
| A | B | Q |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

**OR**


| Input | | Output |
|---|---|---|
| A | B | Q |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

**XOR**


| Input | | Output |
|---|---|---|
| A | B | Q |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

# Solutions

**NAND**


| Input | | Output |
|---|---|---|
| A | B | Q |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

| NOT | AND | OR | XOR |
|---|---|---|---|

**NOT**


| Input | Output |
|---|---|
| A | Q |
| 0 | 1 |
| 1 | 0 |


**AND**


| Input | | Output |
|---|---|---|
| A | B | Q |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

**OR**


| Input | | Output |
|---|---|---|
| A | B | Q |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

**XOR**


| Input | | Output |
|---|---|---|
| A | B | Q |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

# Solutions

**NAND**


| Input | | Output |
|---|---|---|
| A | B | Q |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

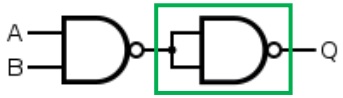
## NOT


| Input | Output |
|---|---|
| A | Q |
| 0 | 1 |
| 1 | 0 |


## AND


| Input | | Output |
|---|---|---|
| A | B | Q |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

NOT (A NAND B)




## OR


| Input | | Output |
|---|---|---|
| A | B | Q |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

## XOR


| Input | | Output |
|---|---|---|
| A | B | Q |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

# Solutions

## NAND


| Input | | Output |
|---|---|---|
| A | B | Q |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

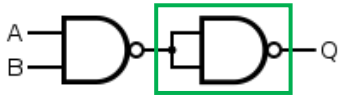
## NOT


| Input | Output |
|---|---|
| A | Q |
| 0 | 1 |
| 1 | 0 |


## AND


| Input | | Output |
|---|---|---|
| A | B | Q |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

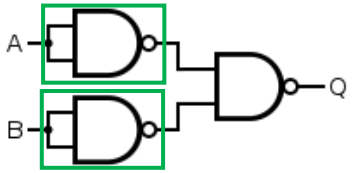
NOT (A NAND B)




## OR


| Input | | Output |
|---|---|---|
| A | B | Q |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

(NOT A) NAND (NOT B)
(De Morgan's Law)




## XOR


| Input | | Output |
|---|---|---|
| A | B | Q |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

# Solutions

## NAND


| Input | | Output |
|---|---|---|
| A | B | Q |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

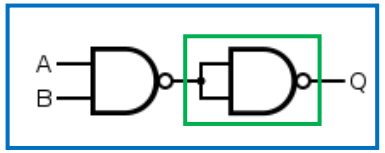
## NOT


| Input | Output |
|---|---|
| A | Q |
| 0 | 1 |
| 1 | 0 |


## AND


| Input | | Output |
|---|---|---|
| A | B | Q |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

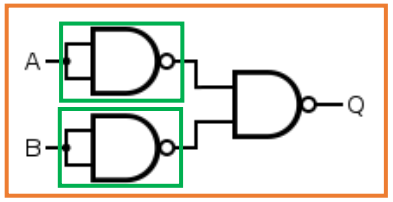
NOT (A NAND B)




## OR


| Input | | Output |
|---|---|---|
| A | B | Q |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

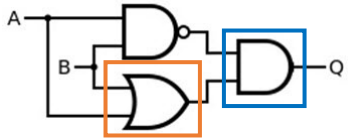
(NOT A) NAND (NOT B)
(De Morgan's Law)




## XOR


| Input | | Output |
|---|---|---|
| A | B | Q |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

(A NAND B) AND (A OR B)



# Solutions

**NAND**


| Input | | Output |
|---|---|---|
| A | B | Q |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

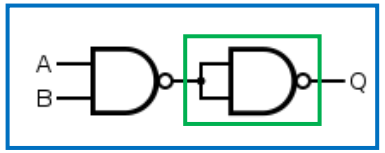
## NOT


| Input | Output |
|---|---|
| A | Q |
| 0 | 1 |
| 1 | 0 |


## AND


| Input | | Output |
|---|---|---|
| A | B | Q |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

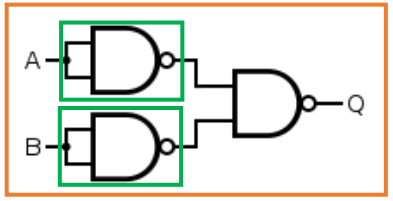
NOT (A NAND B)




## OR


| Input | | Output |
|---|---|---|
| A | B | Q |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

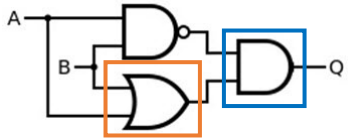
(NOT A) NAND (NOT B)
(De Morgan's Law)



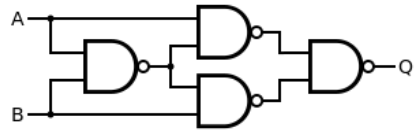

## XOR


| Input | | Output |
|---|---|---|
| A | B | Q |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

(A NAND B) AND (A OR B)




fewer NANDs:



# Solutions

**NAND**


| Input | | Output |
|---|---|---|
| A | B | Q |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

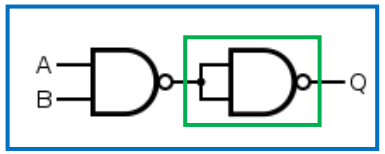
**NOT**


| Input | Output |
|---|---|
| A | Q |
| 0 | 1 |
| 1 | 0 |


**AND**


| Input | | Output |
|---|---|---|
| A | B | Q |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

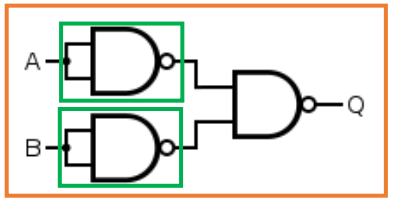
NOT (A NAND B)




**OR**


| Input | | Output |
|---|---|---|
| A | B | Q |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

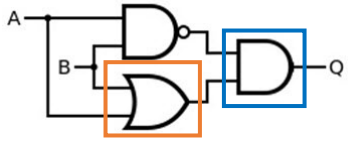
(NOT A) NAND (NOT B)
(De Morgan's Law)



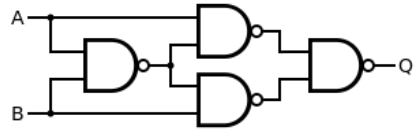

**XOR**


| Input | | Output |
|---|---|---|
| A | B | Q |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

(A NAND B) AND (A OR B)

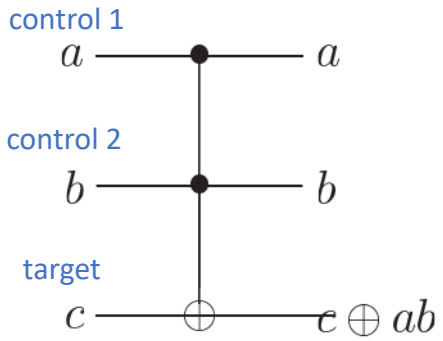



fewer NANDs:




# Step 2: Quantum Gate Implementing NAND

| Inputs | | | Outputs | | |
|---|---|---|---|---|---|
| $a$ | $b$ | $c$ | $a'$ | $b'$ | $c'$ |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 |




**Toffoli Gate**

# Solutions

**NAND**


| Input | | Output |
|---|---|---|
| A | B | Q |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

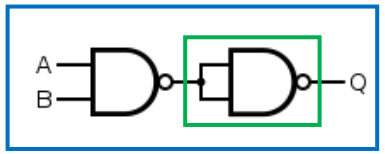
**NOT**


| Input | Output |
|---|---|
| A | Q |
| 0 | 1 |
| 1 | 0 |


**AND**


| Input | | Output |
|---|---|---|
| A | B | Q |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

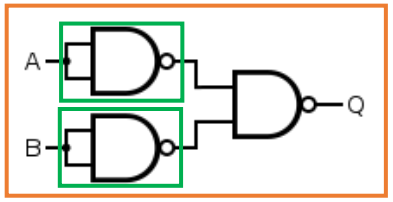
NOT (A NAND B)




**OR**


| Input | | Output |
|---|---|---|
| A | B | Q |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

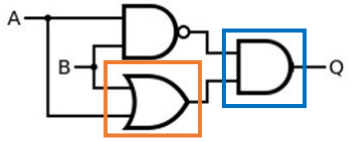
(NOT A) NAND (NOT B)
(De Morgan's Law)



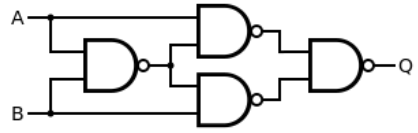

**XOR**


| Input | | Output |
|---|---|---|
| A | B | Q |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

(A NAND B) AND (A OR B)

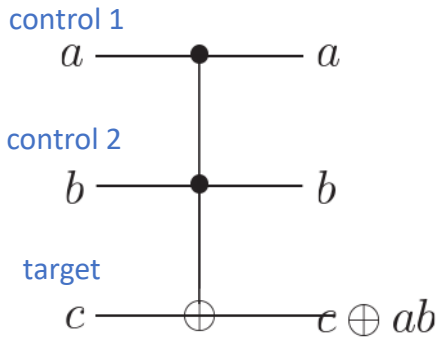



fewer NANDs:




# Step 2: Quantum Gate Implementing NAND

| Inputs | | | Outputs | | |
|---|---|---|---|---|---|
| $a$ | $b$ | $c$ | $a'$ | $b'$ | $c'$ |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 |




**Toffoli Gate**

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

**Unitary Matrix**

# Solutions

**NAND**


| Input | | Output |
|---|---|---|
| A | B | Q |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

| NOT | AND | OR | XOR |
|---|---|---|---|

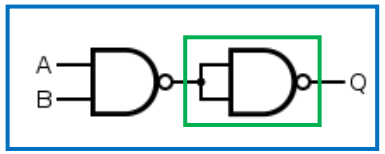
**NOT**


| Input | Output |
|---|---|
| A | Q |
| 0 | 1 |
| 1 | 0 |


**AND**


| Input | | Output |
|---|---|---|
| A | B | Q |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

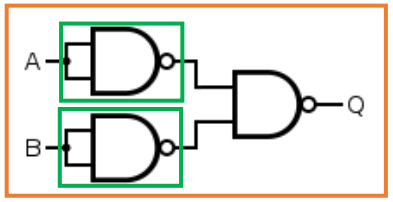
NOT (A NAND B)




**OR**


| Input | | Output |
|---|---|---|
| A | B | Q |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

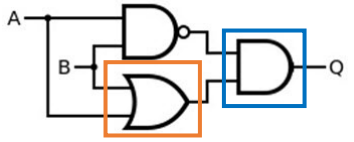
(NOT A) NAND (NOT B)
(De Morgan's Law)



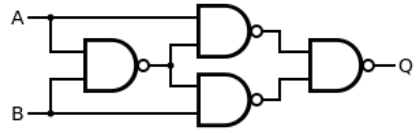

**XOR**


| Input | | Output |
|---|---|---|
| A | B | Q |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

(A NAND B) AND (A OR B)

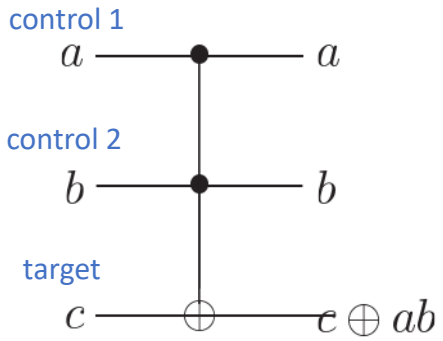



fewer NANDs:




# Step 2: Quantum Gate Implementing NAND

| Inputs | | | Outputs | | |
|---|---|---|---|---|---|
| $a$ | $b$ | $c$ | $a'$ | $b'$ | $c'$ |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 |

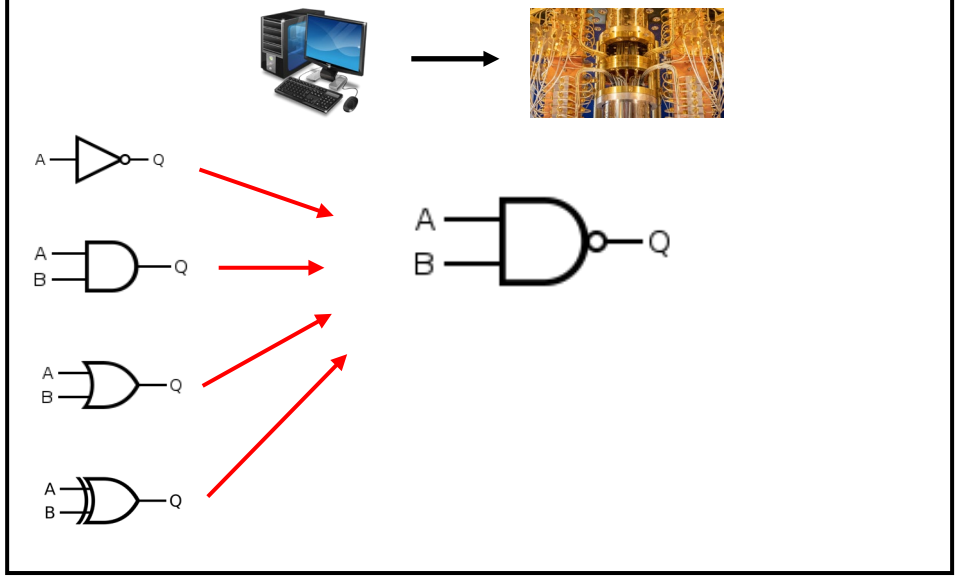



**Toffoli Gate**

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

**Unitary Matrix**

**Implementing NAND with Toffoli**



# Solutions

**NAND**


| Input | | Output |
|---|---|---|
| A | B | Q |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

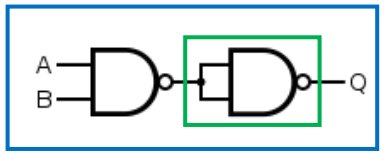
**NOT**


| Input | Output |
|---|---|
| A | Q |
| 0 | 1 |
| 1 | 0 |


**AND**


| Input | | Output |
|---|---|---|
| A | B | Q |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

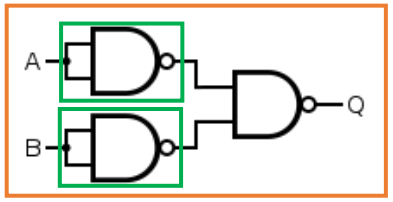
NOT (A NAND B)




**OR**


| Input | | Output |
|---|---|---|
| A | B | Q |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

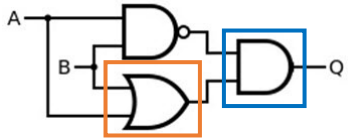
(NOT A) NAND (NOT B)
(De Morgan's Law)



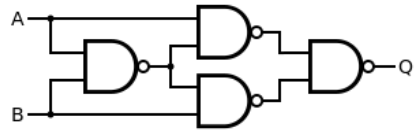

**XOR**


| Input | | Output |
|---|---|---|
| A | B | Q |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

(A NAND B) AND (A OR B)

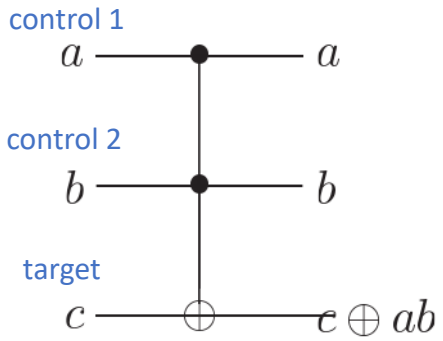



fewer NANDs:




# Step 2: Quantum Gate Implementing NAND

| Inputs | | | Outputs | | |
|---|---|---|---|---|---|
| $a$ | $b$ | $c$ | $a'$ | $b'$ | $c'$ |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 |

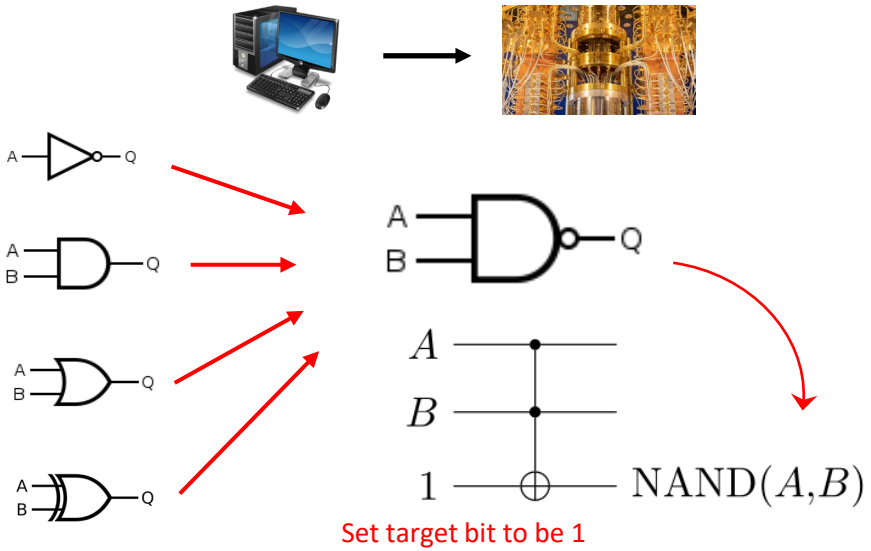



**Toffoli Gate**

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

**Unitary Matrix**

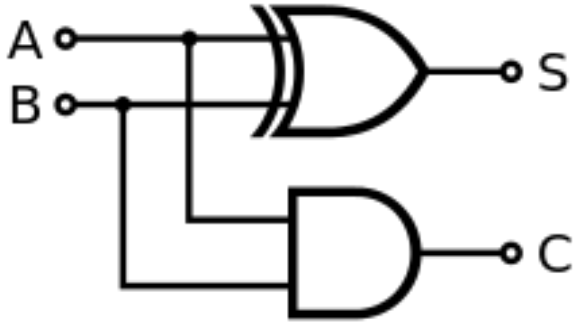
**Implementing NAND with Toffoli**




Set target bit to be 1

# Quantum Half Adder

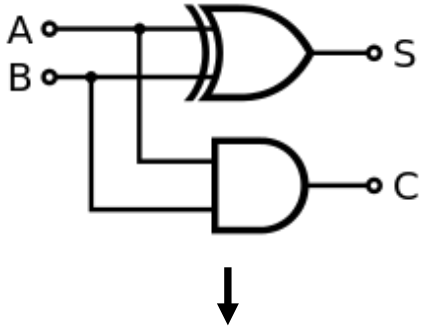
### Circuit for Adding Two Bits Together

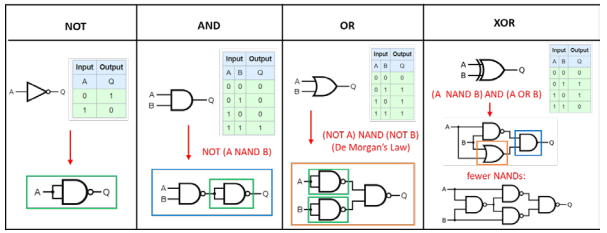


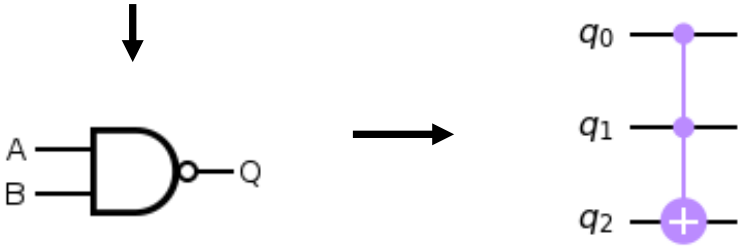

| Input | | Output | |
|---|---|---|---|
| A | B | Sum | Carry |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

### Practice Exercise

Implement a half adder circuit in Qiskit using Toffoli Gates.



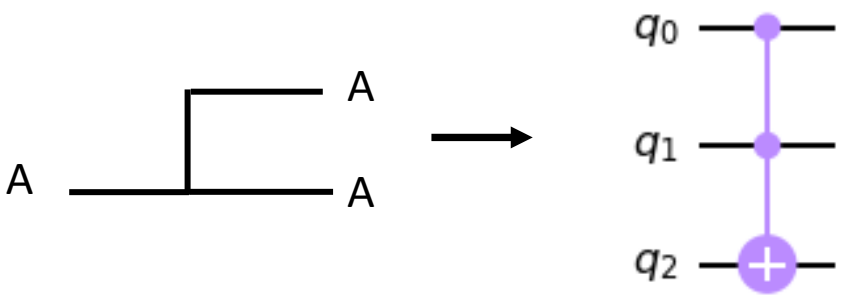

```
circ.ccx(control1, control2, target)
```

**Starter code in** `3-24_quantum_half_adder.ipynb`

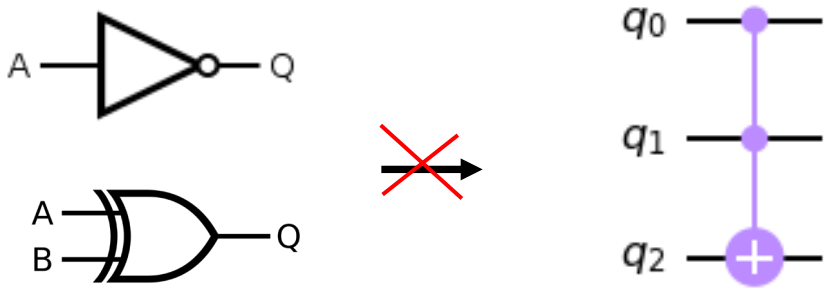
**Solutions in** `3-24_quantum_half_adder_solutions.ipynb`

# Fanouts and Direct implementation of NOT and XOR

**Fanout using Toffoli?**

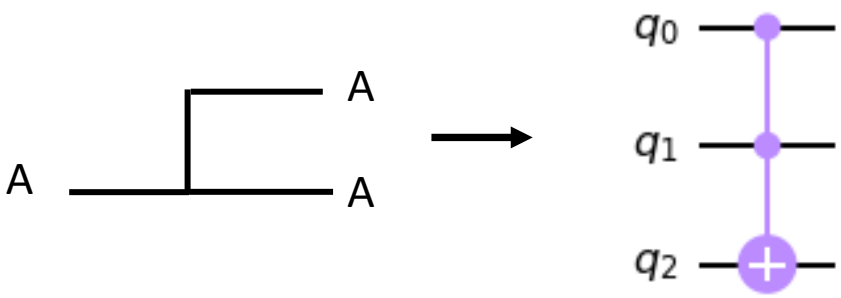



**Don’t really need to use NAND conversion for NOT and XOR gates…**

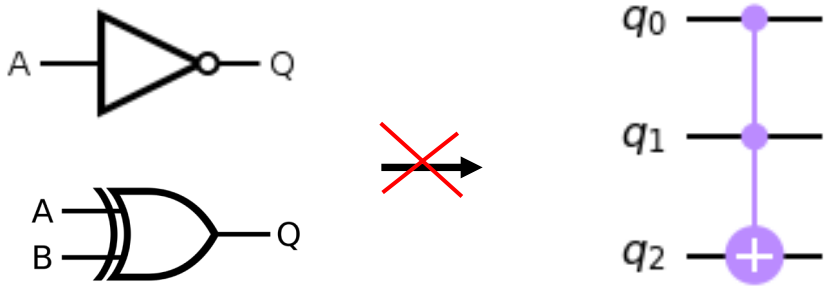


# Fanouts and Direct implementation of NOT and XOR

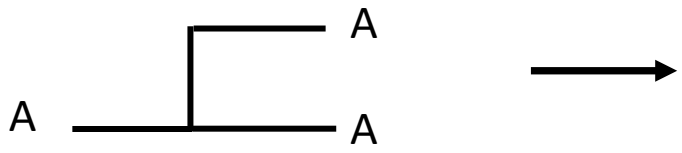
**Fanout using Toffoli?**



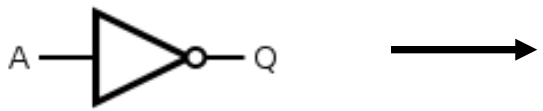

**Don't really need to use NAND conversion for NOT and XOR gates…**



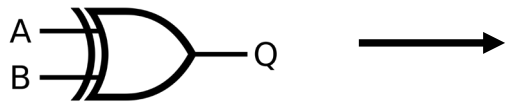

**Fanout can be implemented as…**




**NOT can be implemented as…**

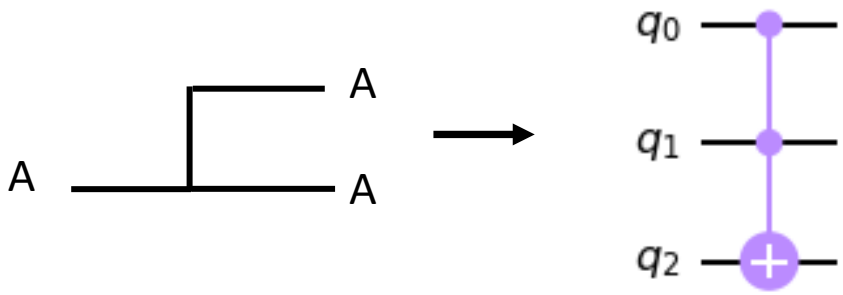



**XOR can be implemented as…**

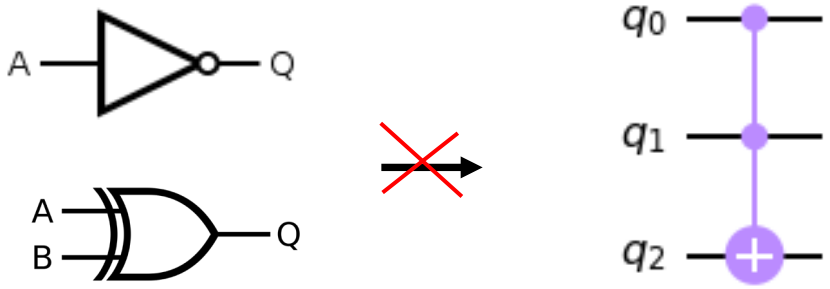


# Fanouts and Direct implementation of NOT and XOR

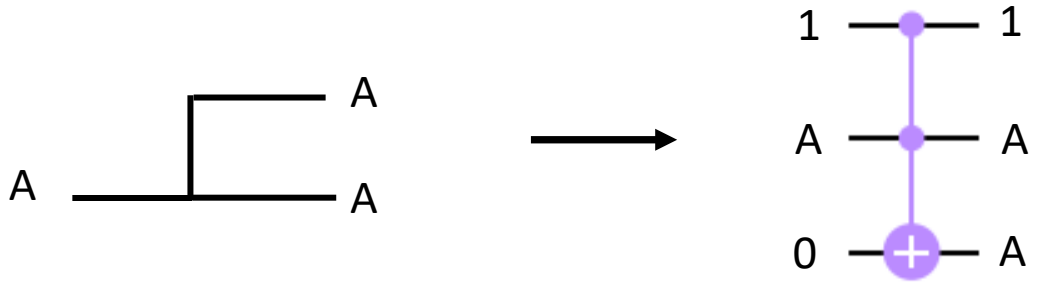
**Fanout using Toffoli?**



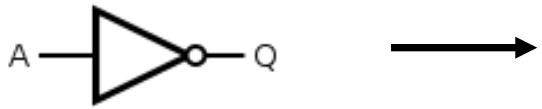

**Don't really need to use NAND conversion for NOT and XOR gates…**



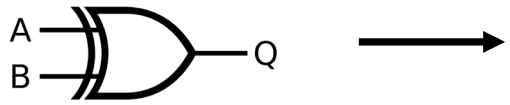

**Fanout can be implemented as…**




**NOT can be implemented as…**

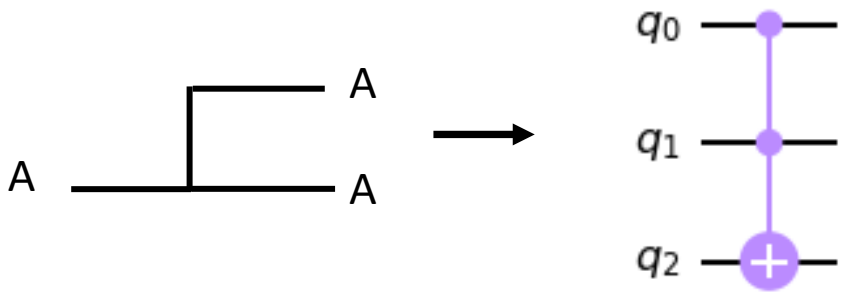



**XOR can be implemented as…**

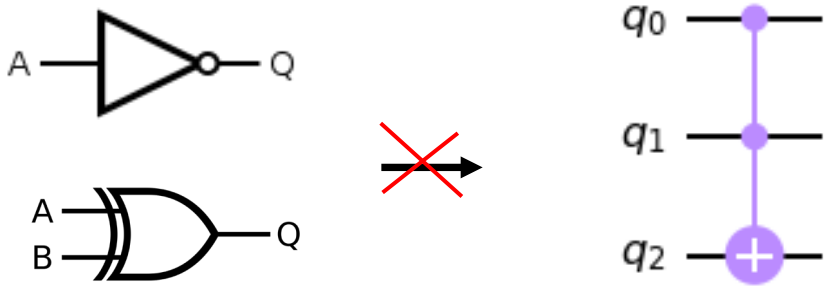


# Fanouts and Direct implementation of NOT and XOR

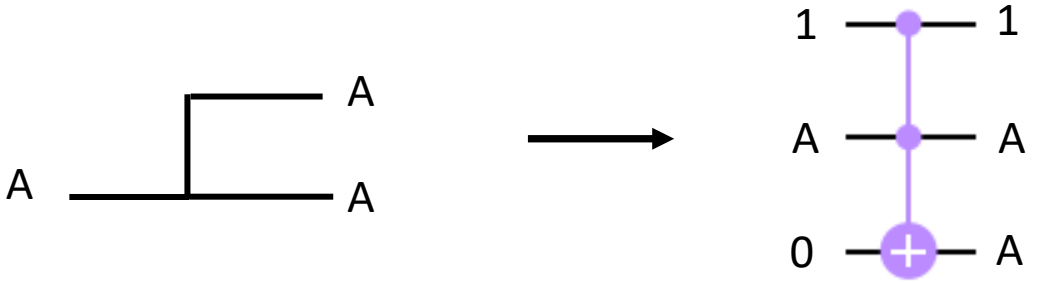
**Fanout using Toffoli?**




**Don't really need to use NAND conversion for NOT and XOR gates…**



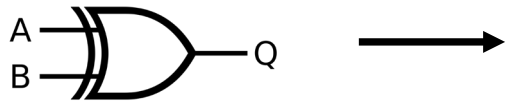

**Fanout can be implemented as…**




**NOT can be implemented as…**

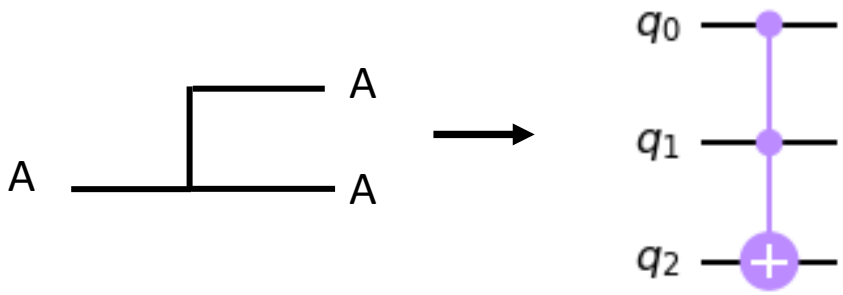
**X gate**


**XOR can be implemented as…**

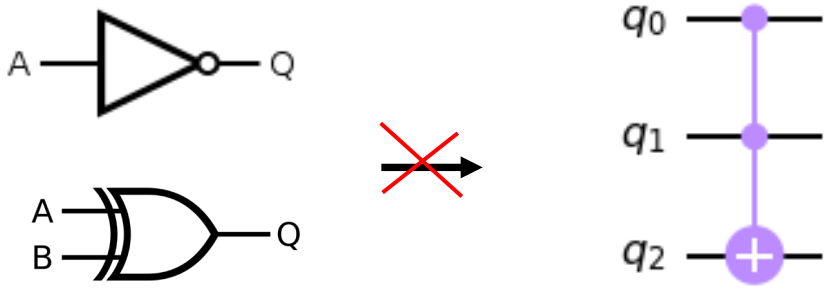


# Fanouts and Direct implementation of NOT and XOR

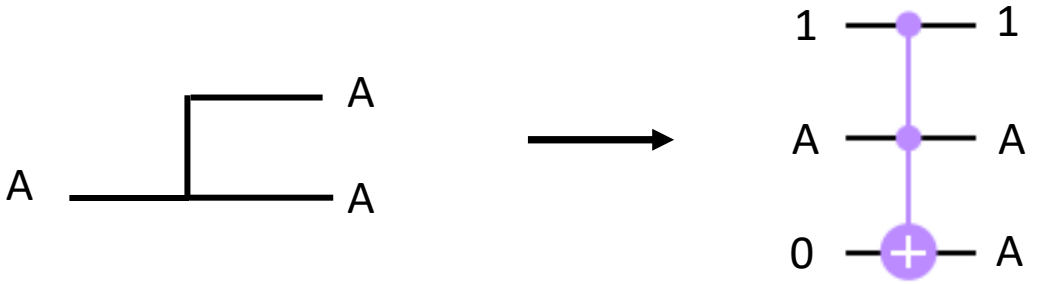
**Fanout using Toffoli?**




**Don't really need to use NAND conversion for NOT and XOR gates…**




**Fanout can be implemented as…**




**NOT can be implemented as…**

**X gate**


**XOR can be implemented as…**

**CX gate**