

Computer Architecture
Lecture Week 3 (Part II)

Combinational Logic Circuits

Introduction

Artem Burmyakov

September 19, 2025



INNOPOLIS
UNIVERSITY

Terminology

A Boolean function

(introduced by George Boole)

A function taking one or multiple binary inputs and producing exactly one binary output

Sample boolean variables:

$$A, B, C, D, F \in \{0,1\}$$

Functions in one notation:

$$F(A, B) = A * B$$

$$F(A, B, C) = A * B + C$$

$$F(A, B, C, D) = A * (B + C) + D$$

Functions in another notation:

$$F(A, B) = \text{AND}(A, B)$$

$$F(A, B, C) = \text{OR}(\text{AND}(A, B), C)$$

$$F(A, B, C, D) = \text{OR}(\text{AND}(A, \text{OR}(B, C)), D)$$

Other notations are used as well

Terminology

A Boolean function (introduced by George Boole)	A function taking one or multiple binary inputs and producing exactly one binary output
Truth table	A table specifying the output of a Boolean function for every combination of input values

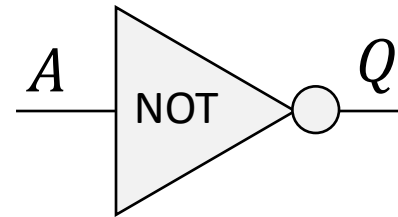
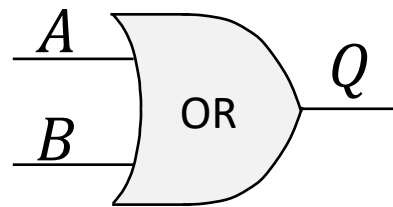
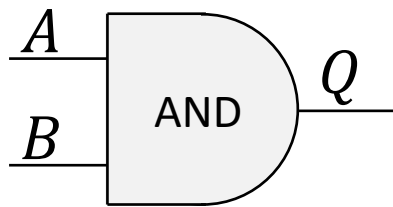
Truth table for function $F = A * B$:

<i>A</i>	<i>B</i>	<i>F = A * B</i>
0	0	0
1	0	0
0	1	0
1	1	1

Terminology

A Boolean function (introduced by George Boole)	A function taking one or multiple binary inputs and producing exactly one binary output
Truth table	A table specifying the output of a Boolean function for every combination of input values
A logic gate	An abstract or physical device implementing a Boolean function

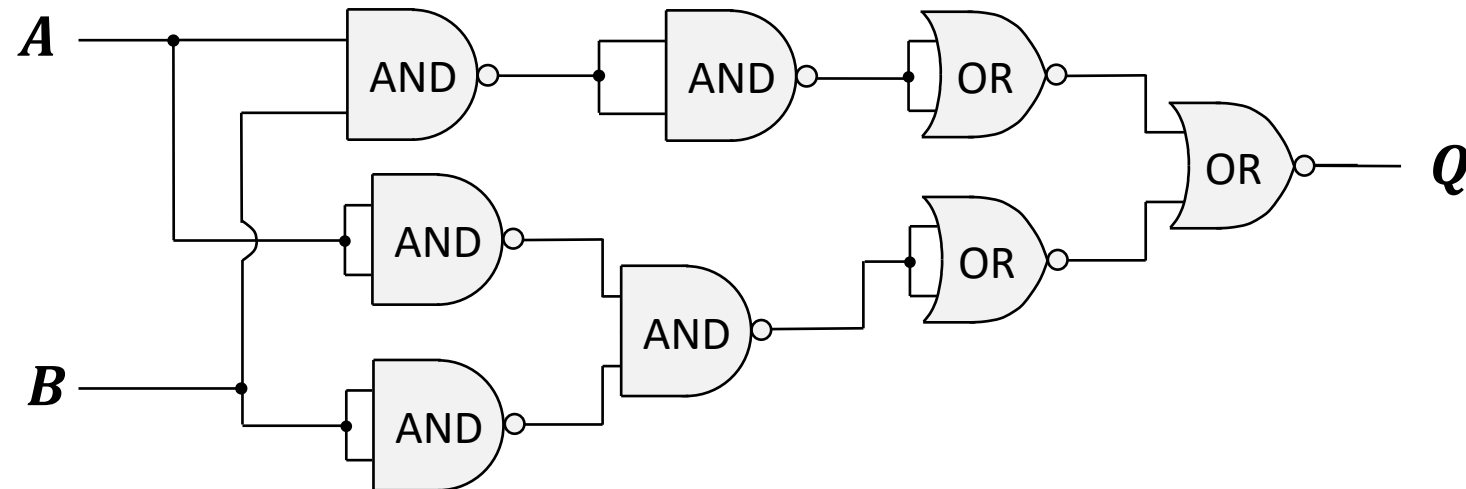
Sample logic gates:

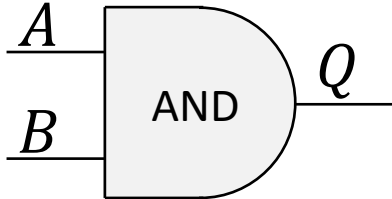


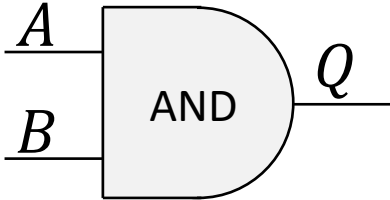
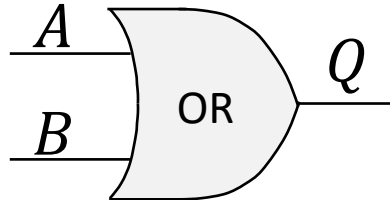
Terminology

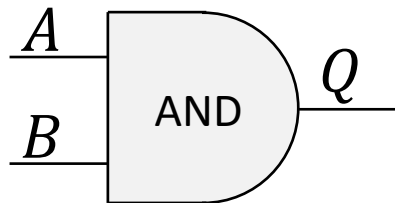
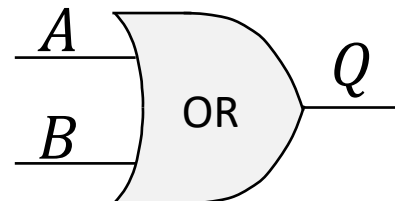
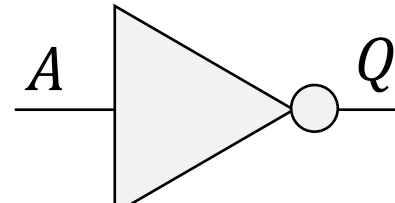
A Boolean function (introduced by George Boole)	A function taking one or multiple binary inputs and producing exactly one binary output
Truth table	A table specifying the output of a Boolean function for every combination of input values
A logic gate	An abstract or physical device implementing a Boolean function
A logic circuit	Composition of logic gates, to implement a certain boolean function

Sample logic circuit:



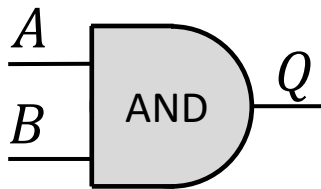
Logic Gate	Symbolic Representation	Truth Table															
AND		<table><tr><th><i>A</i></th><th><i>B</i></th><th><i>Q=AB</i></th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	<i>A</i>	<i>B</i>	<i>Q=AB</i>	0	0	0	1	0	0	0	1	0	1	1	1
<i>A</i>	<i>B</i>	<i>Q=AB</i>															
0	0	0															
1	0	0															
0	1	0															
1	1	1															

Logic Gate	Symbolic Representation	Truth Table															
AND		<table><tr><th><i>A</i></th><th><i>B</i></th><th><i>Q=AB</i></th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	<i>A</i>	<i>B</i>	<i>Q=AB</i>	0	0	0	1	0	0	0	1	0	1	1	1
<i>A</i>	<i>B</i>	<i>Q=AB</i>															
0	0	0															
1	0	0															
0	1	0															
1	1	1															
OR		<table><tr><th><i>A</i></th><th><i>B</i></th><th><i>Q=A+B</i></th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	<i>A</i>	<i>B</i>	<i>Q=A+B</i>	0	0	0	1	0	1	0	1	1	1	1	1
<i>A</i>	<i>B</i>	<i>Q=A+B</i>															
0	0	0															
1	0	1															
0	1	1															
1	1	1															

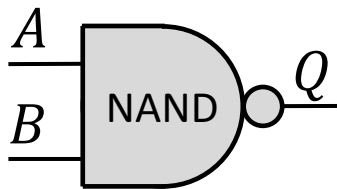
Logic Gate	Symbolic Representation	Truth Table															
AND		<table><tr><th><i>A</i></th><th><i>B</i></th><th><i>Q=AB</i></th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	<i>A</i>	<i>B</i>	<i>Q=AB</i>	0	0	0	1	0	0	0	1	0	1	1	1
<i>A</i>	<i>B</i>	<i>Q=AB</i>															
0	0	0															
1	0	0															
0	1	0															
1	1	1															
OR		<table><tr><th><i>A</i></th><th><i>B</i></th><th><i>Q=A+B</i></th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	<i>A</i>	<i>B</i>	<i>Q=A+B</i>	0	0	0	1	0	1	0	1	1	1	1	1
<i>A</i>	<i>B</i>	<i>Q=A+B</i>															
0	0	0															
1	0	1															
0	1	1															
1	1	1															
NOT		<table><tr><th><i>A</i></th><th><i>Q=A'</i></th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	<i>A</i>	<i>Q=A'</i>	0	1	1	0									
<i>A</i>	<i>Q=A'</i>																
0	1																
1	0																

Logic gates correspond to physical devices implemented by using transistors

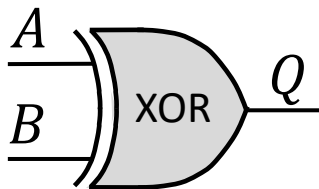
Recap: Basic Logic Gates



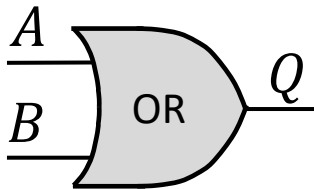
A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1



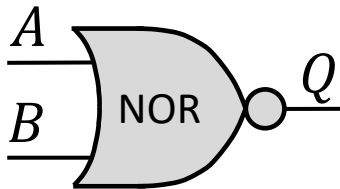
A	B	Q
0	0	1
0	1	1
1	0	1
1	1	0



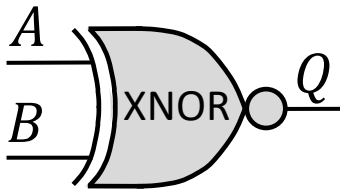
A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0



A	B	Q
0	0	0
0	1	1
1	0	1
1	1	1

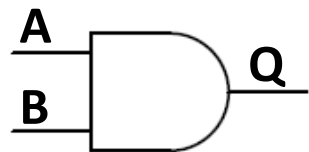


A	B	Q
0	0	1
0	1	0
1	0	0
1	1	0



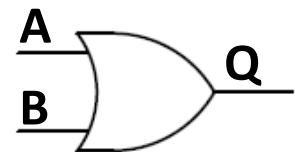
A	B	Q
0	0	1
0	1	0
1	0	0
1	1	1

Basic Logic Gates



AND

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



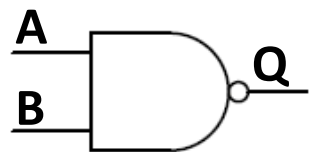
OR

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



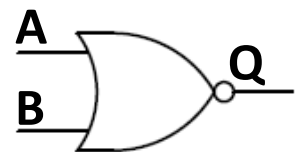
XOR

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



NAND

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



NOR

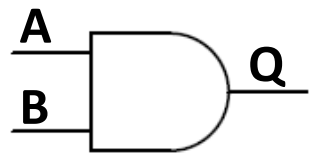
A	B	Output
0	0	1
0	1	0
1	0	0
1	1	0



XNOR

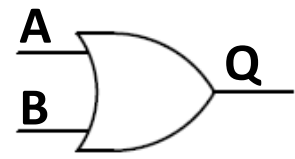
A	B	Output
0	0	1
0	1	0
1	0	0
1	1	1

Basic Logic Gates



AND

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



OR

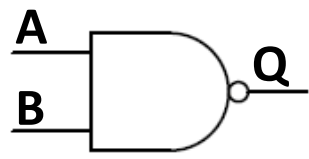
A	B	Output
0	0	0
0	1	1
1	0	1
1	1	1

A logic diagram of an XOR gate. It has two input lines labeled 'A' and 'B' entering from the left into a pointed gate with a curved front. A single output line labeled 'Q' exits from the right. The entire diagram is enclosed in a red rounded rectangle.

XOR

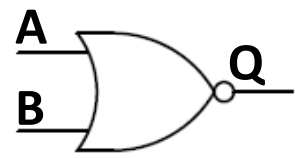
A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0

XOR = eXclusive OR:
unlike OR, if both inputs are "1"s, the output is "0"



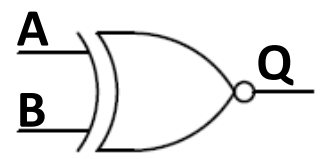
NAND

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



NOR

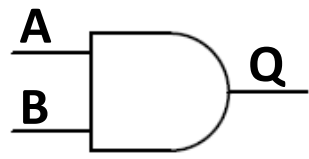
A	B	Output
0	0	1
0	1	0
1	0	0
1	1	0



XNOR

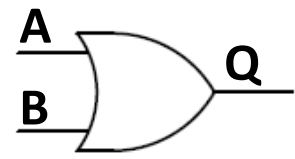
A	B	Output
0	0	1
0	1	0
1	0	0
1	1	1

Basic Logic Gates



AND

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



OR

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

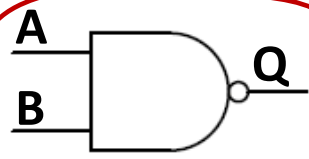
A logic diagram of an XOR gate. It has two input lines labeled 'A' and 'B' on the left, and one output line labeled 'Q' on the right. The gate is represented by a symbol with a curved left side and a pointed right side, similar to an OR gate but with a different internal shape. The entire diagram is enclosed in a rounded rectangle.

XOR

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

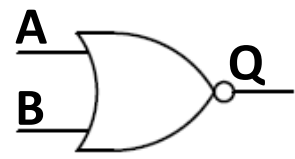
XOR = eXclusive OR:
unlike OR, if both inputs are "1"s, the output is "0"

NAND = Not AND
Widely used in practice



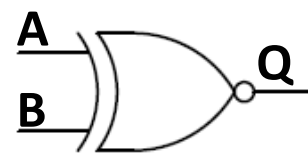
NAND

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



NOR

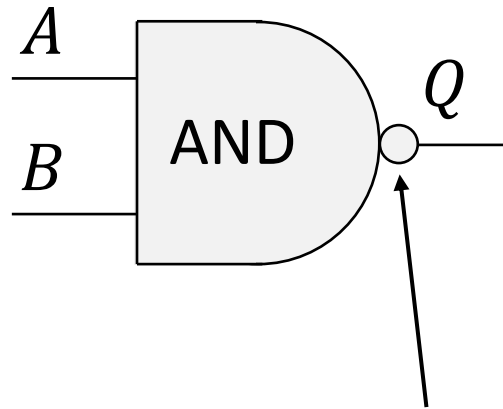
A	B	Output
0	0	1
0	1	0
1	0	0
1	1	0



XNOR

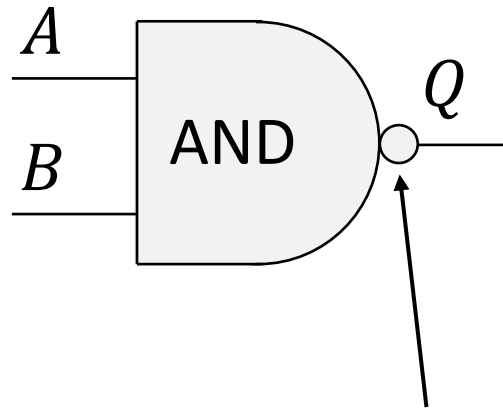
A	B	Output
0	0	1
0	1	0
1	0	0
1	1	1

NAND Logic Gate

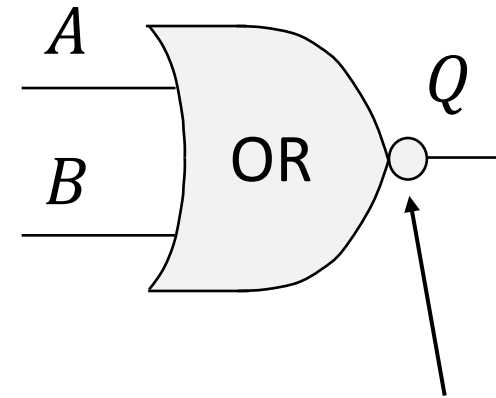


NAND = AND + NOT

NAND and NOR Logic Gates



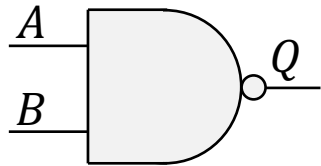
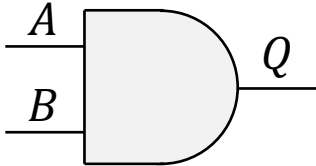
NAND = AND + NOT



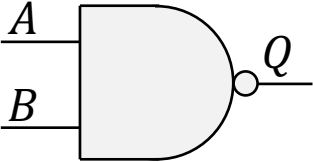
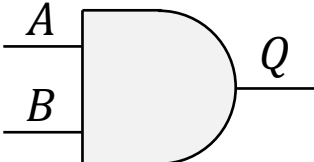
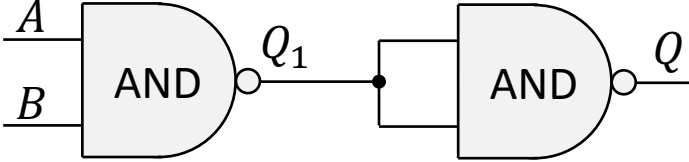
NOR = OR + NOT

**These logic gates are “universal”:
Any other logic gate can be implemented through one of these!**

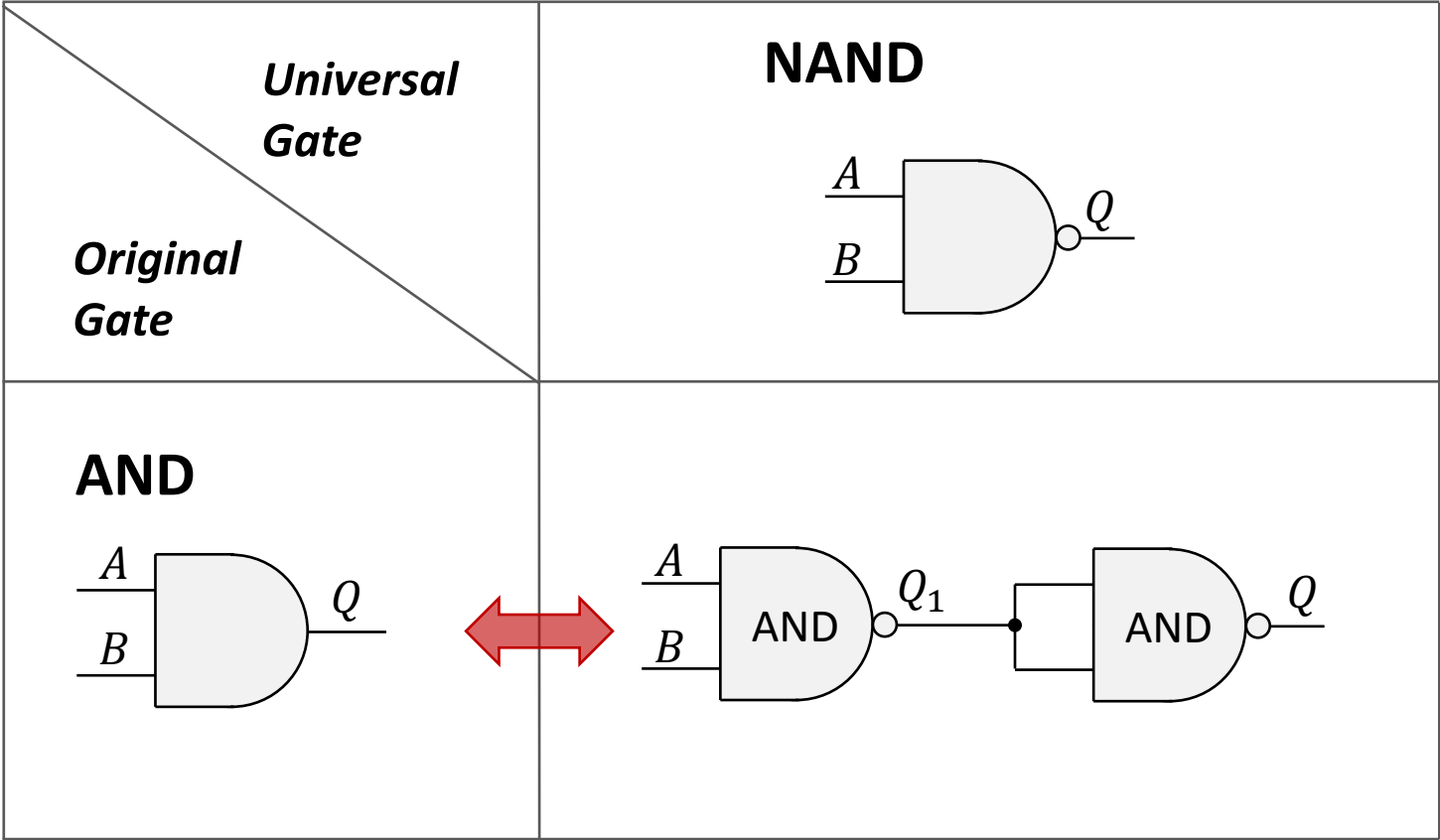
Universal Logic Gates: Implementation of AND Through NAND

<i>Universal Gate</i> <i>Original Gate</i>	NAND 
AND 	

Universal Logic Gates: Implementation of AND Through NAND

<i>Universal Gate</i> <i>Original Gate</i>	NAND 
AND 	

Universal Logic Gates: Implementation of AND Through NAND



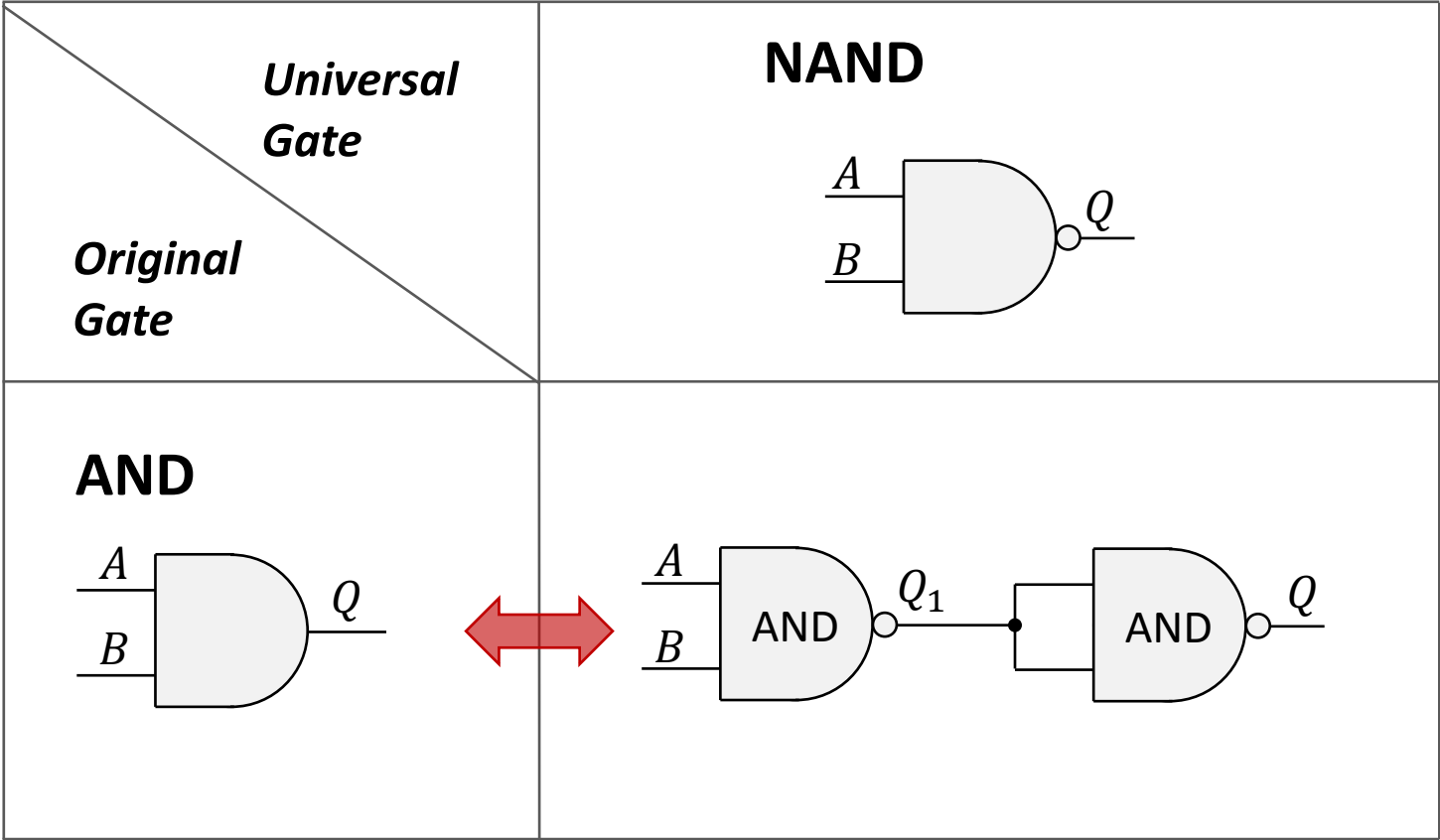
A	B	Q=AB
0	0	0
1	0	0
0	1	0
1	1	1



A	B	Q ₁	Q ₁	Q
0	0	1	1	0
1	0	1	1	0
0	1	1	1	0
1	1	0	0	1

To proof circuits equivalence, we compare truth tables

Universal Logic Gates: Implementation of AND Through NAND



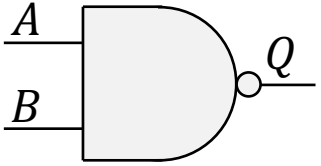
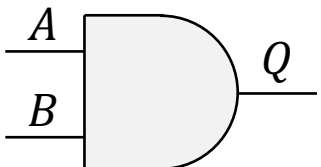
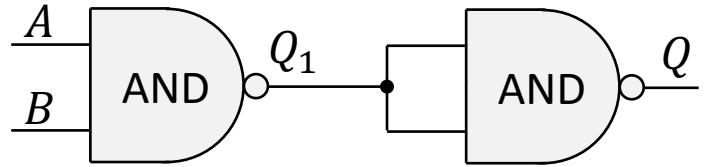
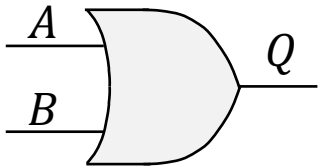
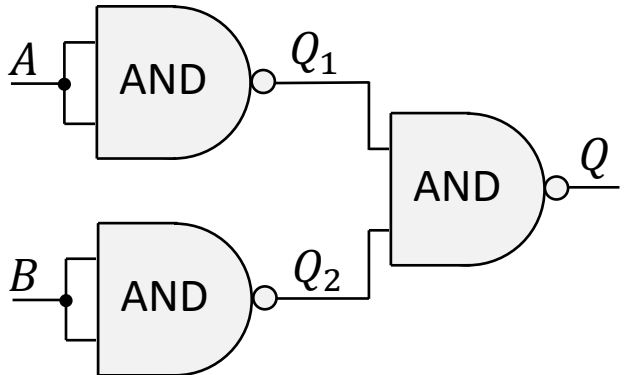
A	B	Q=AB
0	0	0
1	0	0
0	1	0
1	1	1



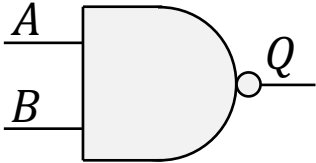
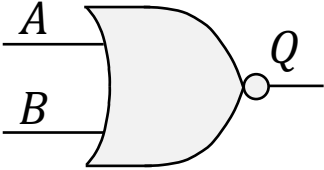
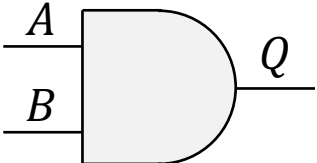
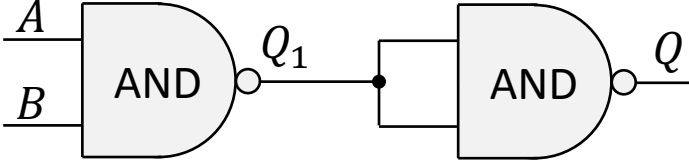
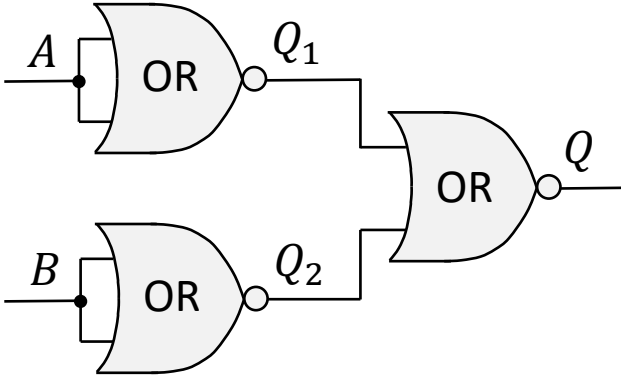
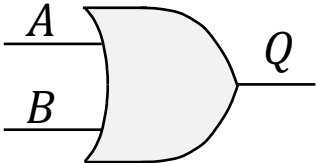
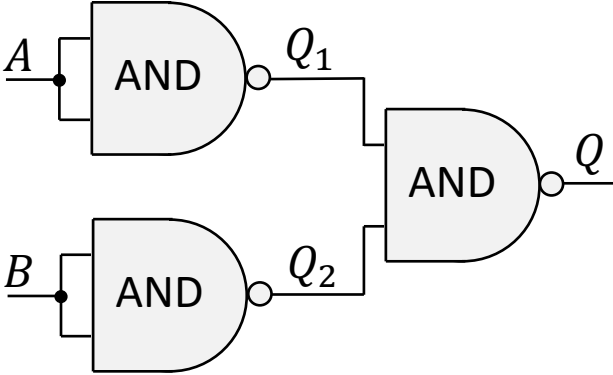
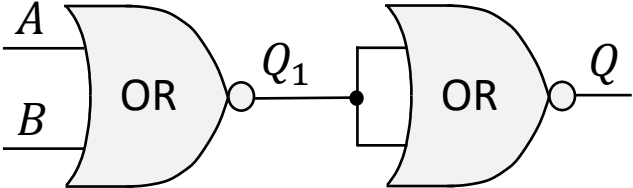
A	B	Q ₁	Q ₁	Q
0	0	1	1	0
1	0	1	1	0
0	1	1	1	0
1	1	0	0	1

To proof circuits equivalence, we compare truth tables

Universal Logic Gates: Sample Use Cases

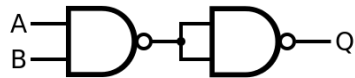
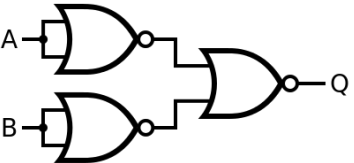
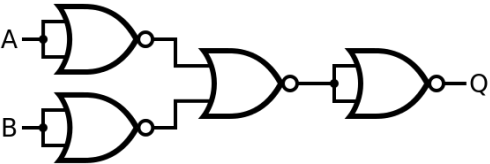
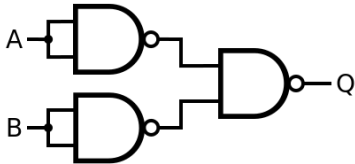
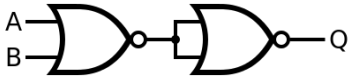
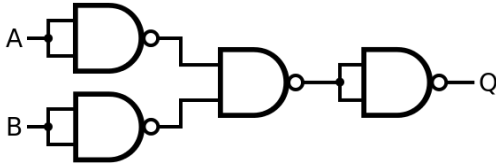
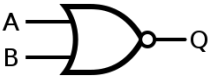
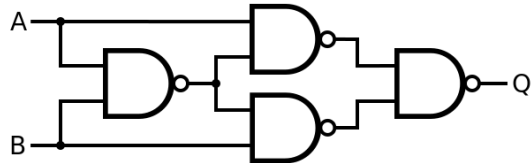
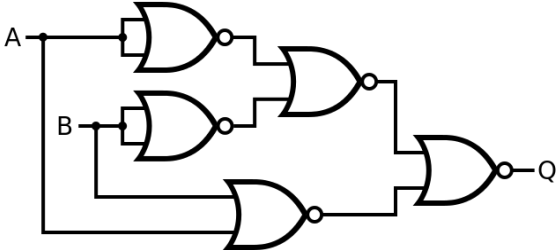
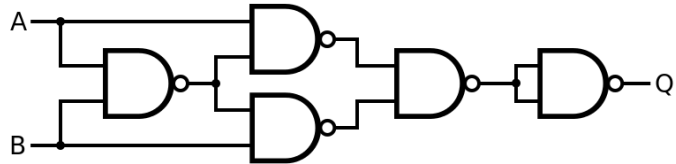
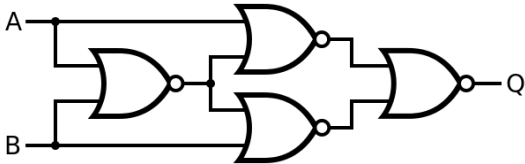
<i>Universal Gate</i> <i>Original Gate</i>	NAND 
AND 	
OR 	

Universal Logic Gates: Sample Use Cases

Universal Gate Original Gate	NAND 	NOR 
AND 		
OR 		

Universal Logic Gates: NAND and NOR

Any other logic gate can be implemented through one of the universal logic gates

type	NAND construction	NOR construction
NOT		
AND		
NAND		
OR		
NOR		
XOR		
XNOR		

Boolean Function Implementation

Any Boolean function can be implemented by using one of these sets of logic gates:

1) AND, OR, NOT

Boolean Function Implementation

Any Boolean function can be implemented by using one of these sets of logic gates:

1) AND, OR, NOT

2) NAND – universal logic gate

Boolean Function Implementation

Any Boolean function can be implemented by using one of these sets of logic gates:

- 1) **AND, OR, NOT**
- 2) **NAND** – universal logic gate
- 3) **NOR** – universal logic gate

