

Computer Architecture
Lecture Week 3 (Part III)

Logic Gates

Combinational Logic Circuits Intro

Artem Burmyakov

September 18, 2026



INNOPOLIS
UNIVERSITY

Recap: Boolean Algebra Terminology

Recap: Boolean Algebra Terminology

Boolean Function	Function taking one or multiple binary inputs and producing exactly one binary output
-------------------------	--

Recap: Boolean Algebra Terminology

Boolean Function	Function taking one or multiple binary inputs and producing exactly one binary output
-------------------------	--

Sample boolean variables:

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

One notation:

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

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

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

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

Recap: Logic Gate

Boolean Function	Function taking one or multiple binary inputs and producing exactly one binary output
Logic Gate	?

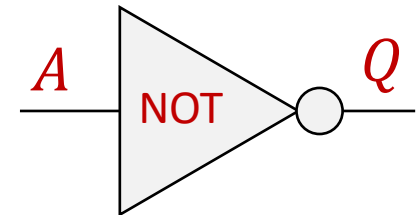
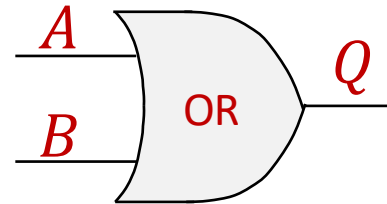
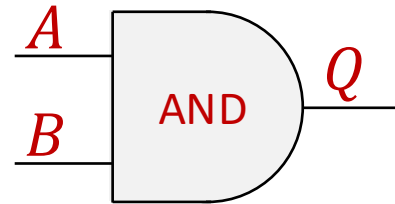
Recap: Logic Gate

Boolean Function	Function taking one or multiple binary inputs and producing exactly one binary output
Logic Gate	Device to implement some basic boolean function (e.g. AND, OR, NOT)

Recap: Logic Gate

Boolean Function	Function taking one or multiple binary inputs and producing exactly one binary output
Logic Gate	Device to implement some basic boolean function (e.g. AND, OR, NOT)

Examples:



[Symbolic representation according to the Standard IEEE Std 91/91a-1991]

Recap: Logic Circuit

Boolean Function	Function taking one or multiple binary inputs and producing exactly one binary output
Logic Gate	Device to implement some basic boolean function (e.g. AND, OR, NOT)
Logic Circuit	?

Recap: Logic Circuit

Boolean Function	Function taking one or multiple binary inputs and producing exactly one binary output
Logic Gate	Device to implement some basic boolean function (e.g. AND, OR, NOT)
Logic Circuit	Interconnection of logic gates to implement some function

Recap: Logic Circuit

Boolean Function	Function taking one or multiple binary inputs and producing exactly one binary output
Logic Gate	Device to implement some basic boolean function (e.g. AND, OR, NOT)
Logic Circuit	Interconnection of logic gates to implement some function

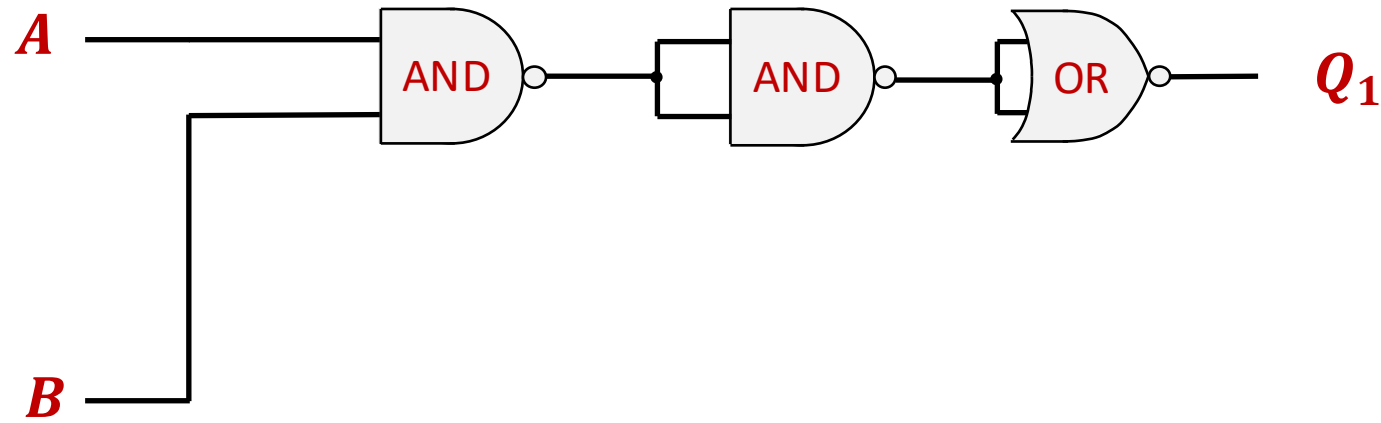
Example: ***A*** —

B —

Recap: Logic Circuit

Boolean Function	Function taking one or multiple binary inputs and producing exactly one binary output
Logic Gate	Device to implement some basic boolean function (e.g. AND, OR, NOT)
Logic Circuit	Interconnection of logic gates to implement some function

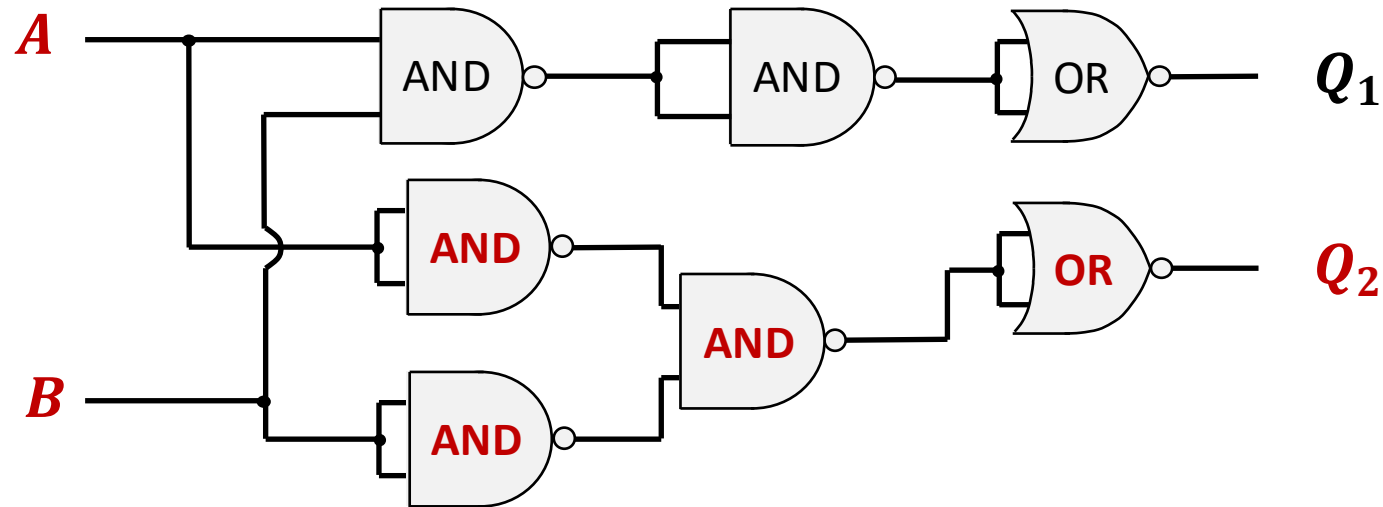
Example:



Recap: Logic Circuit

Boolean Function	Function taking one or multiple binary inputs and producing exactly one binary output
Logic Gate	Device to implement some basic boolean function (e.g. AND, OR, NOT)
Logic Circuit	Interconnection of logic gates to implement some function

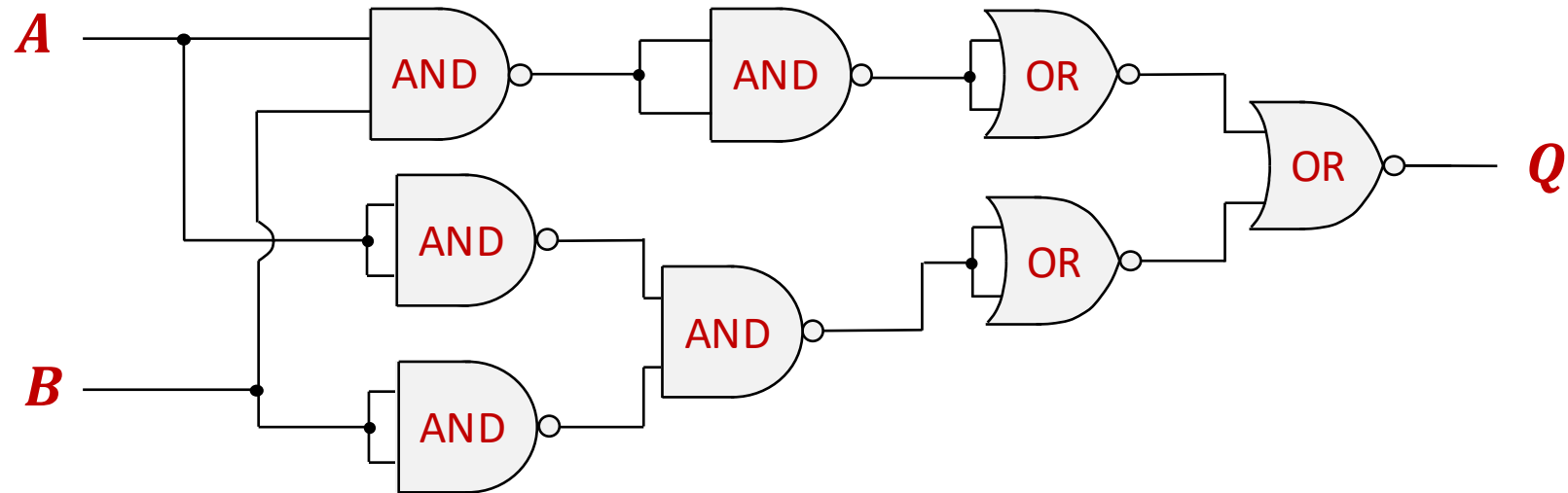
Example:



Recap: Logic Circuit

Boolean Function	Function taking one or multiple binary inputs and producing exactly one binary output
Logic Gate	Device to implement some basic boolean function (e.g. AND, OR, NOT)
Logic Circuit	Interconnection of logic gates to implement some function

Example:



Some boolean function implementation $Q=F(A,B)$ 13

Recap: Truth Table

Boolean Function	Function taking one or multiple binary inputs and producing exactly one binary output
Logic Gate	Device to implement some basic boolean function (e.g. AND, OR, NOT)
Logic Circuit	Interconnection of logic gates to implement some function
Truth Table	Table of boolean function outputs for possible combinations of inputs



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

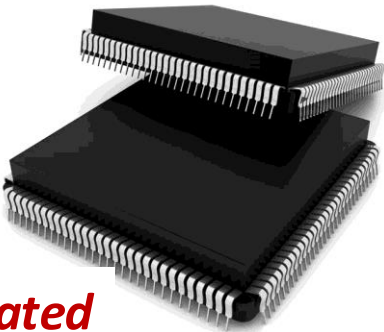
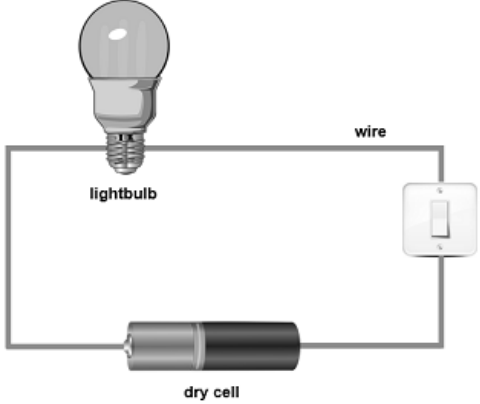
Integrated Circuits

Boolean Function	Function taking one or multiple binary inputs and producing exactly one binary output
Logic Gate	Device to implement some basic boolean function (e.g. AND, OR, NOT)
Logic Circuit	Interconnection of logic gates to implement some function
Truth Table	Table of boolean function outputs for possible combinations of inputs
Integrated Circuit <i>(chip, microchip)</i>	?

Integrated and Discrete Circuits

Boolean Function	Function taking one or multiple binary inputs and producing exactly one binary output
Logic Gate	Device to implement some basic boolean function (e.g. AND, OR, NOT)
Logic Circuit	Interconnection of logic gates to implement some function
Truth Table	Table of boolean function outputs for possible combinations of inputs
Integrated Circuit <i>(chip, microchip)</i>	Antonym to a “discrete” circuit; can be assembled and reassembled

Integrated vs. Discrete Circuits

Boolean Function	Function taking one or multiple binary inputs and producing exactly one binary output
Logic Gate	Device to implement some basic boolean function (e.g. AND, OR, NOT)
Logic Circuit	Interconnection of logic gates to implement some function
Truth Table	Table of boolean function outputs for possible combinations of inputs
Integrated Circuit <i>(chip, microchip)</i>	Antonym to a “discrete” circuit; can be assembled and reassembled  <i>Integrated</i>  <i>discrete</i>

Advantages of Integrated over Discrete Circuits

Boolean Function	Function taking one or multiple binary inputs and producing exactly one binary output
Logic Gate	Device to implement some basic boolean function (e.g. AND, OR, NOT)
Logic Circuit	Interconnection of logic gates to implement some function
Truth Table	Table of boolean function outputs for possible combinations of inputs
Integrated Circuit <i>(chip, microchip)</i>	Antonym to a “discrete” circuit; can be assembled and reassembled Advantages of “integrated” over “discrete” circuits?

Advantages of Integrated over Discrete Circuits

Boolean Function	Function taking one or multiple binary inputs and producing exactly one binary output
Logic Gate	Device to implement some basic boolean function (e.g. AND, OR, NOT)
Logic Circuit	Interconnection of logic gates to implement some function
Truth Table	Table of boolean function outputs for possible combinations of inputs
Integrated Circuit <i>(chip, microchip)</i>	Antonym to a “discrete” circuit; can be assembled and reassembled Advantages over “discrete” circuits: - Compact size

Advantages of Integrated over Discrete Circuits

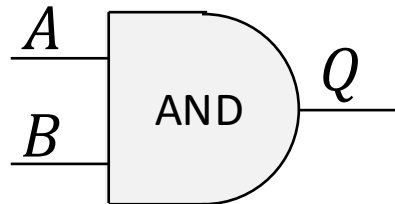
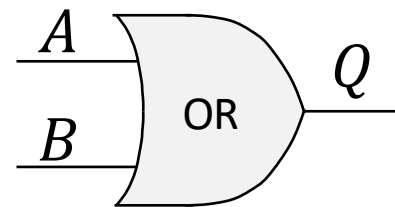
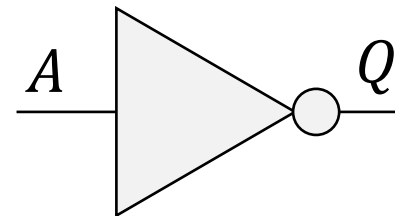
Boolean Function	Function taking one or multiple binary inputs and producing exactly one binary output
Logic Gate	Device to implement some basic boolean function (e.g. AND, OR, NOT)
Logic Circuit	Interconnection of logic gates to implement some function
Truth Table	Table of boolean function outputs for possible combinations of inputs
Integrated Circuit <i>(chip, microchip)</i>	Antonym to a “discrete” circuit; can be assembled and reassembled Advantages over “discrete” circuits: - Compact size; - Performance (e.g. lower propagation delays)

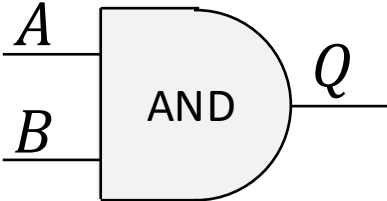
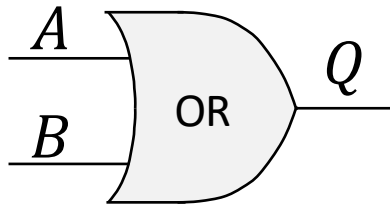
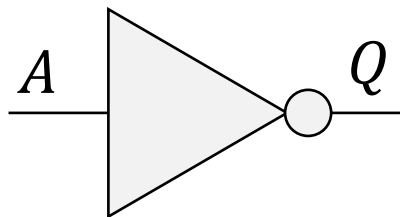
Advantages of Integrated over Discrete Circuits

Boolean Function	Function taking one or multiple binary inputs and producing exactly one binary output
Logic Gate	Device to implement some basic boolean function (e.g. AND, OR, NOT)
Logic Circuit	Interconnection of logic gates to implement some function
Truth Table	Table of boolean function outputs for possible combinations of inputs
Integrated Circuit <i>(chip, microchip)</i>	Antonym to a “discrete” circuit; can be assembled and reassembled Advantages over “discrete” circuits: - Compact size; - Performance (e.g. lower propagation delays); - Lower manufacturing price

Summary

Boolean Function	Function taking one or multiple binary inputs and producing exactly one binary output
Logic Gate	Device to implement some basic boolean function (e.g. AND, OR, NOT)
Logic Circuit	Interconnection of logic gates to implement some function
Truth Table	Table of boolean function outputs for possible combinations of inputs
Integrated Circuit (<i>chip, microchip</i>)	Antonym to a “discrete” circuit; can be assembled and reassembled Advantages over “discrete” circuits: - Compact size; - Performance (e.g. lower propagation delays); - Lower manufacturing price

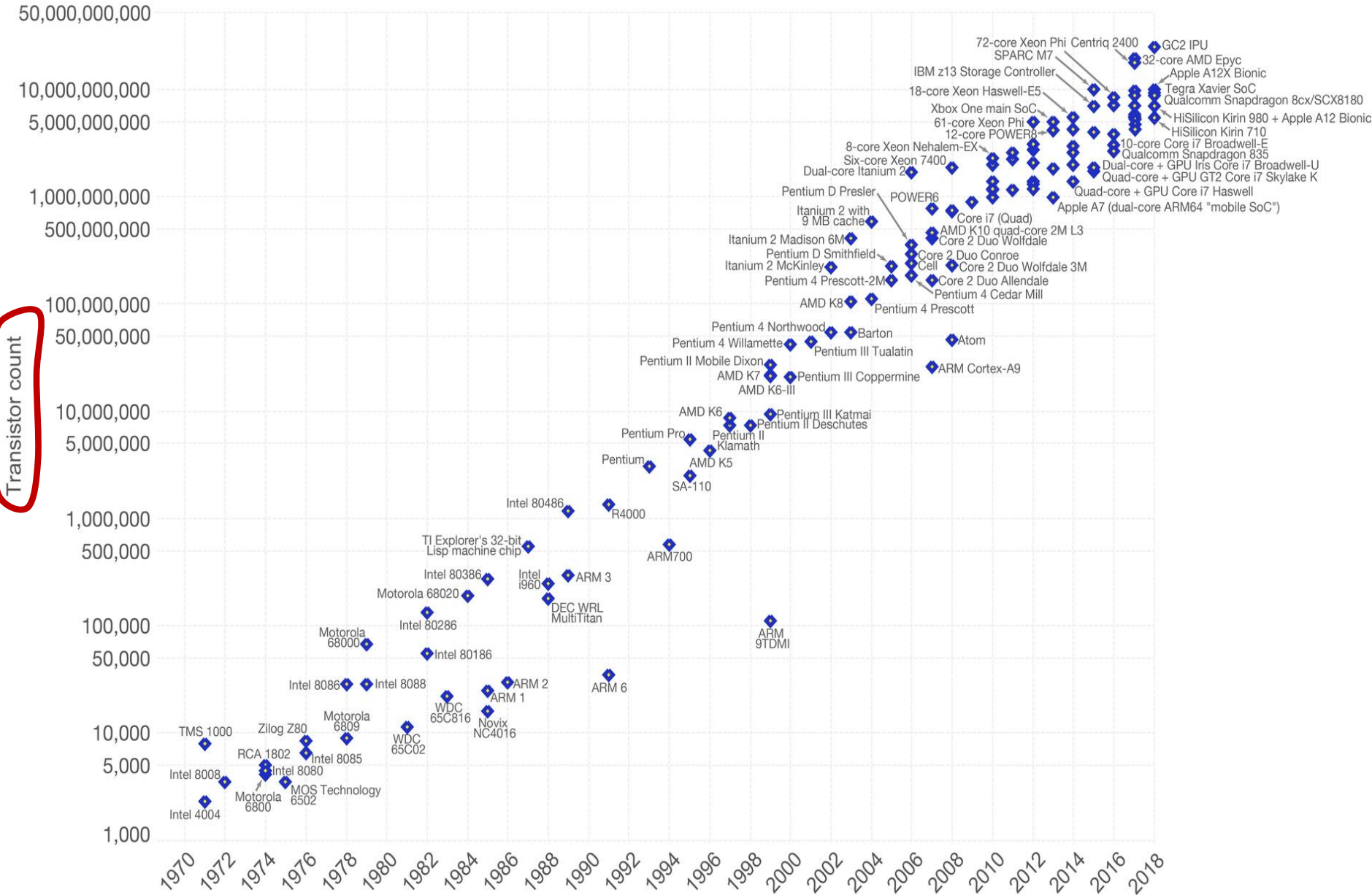
Logic Gate	Symbolic Representation	Truth Table															
AND		<table><tr><th>A</th><th>B</th><th>Q=AB</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	Q=AB	0	0	0	1	0	0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

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

Logic gates correspond to physical circuits based on **transistors**

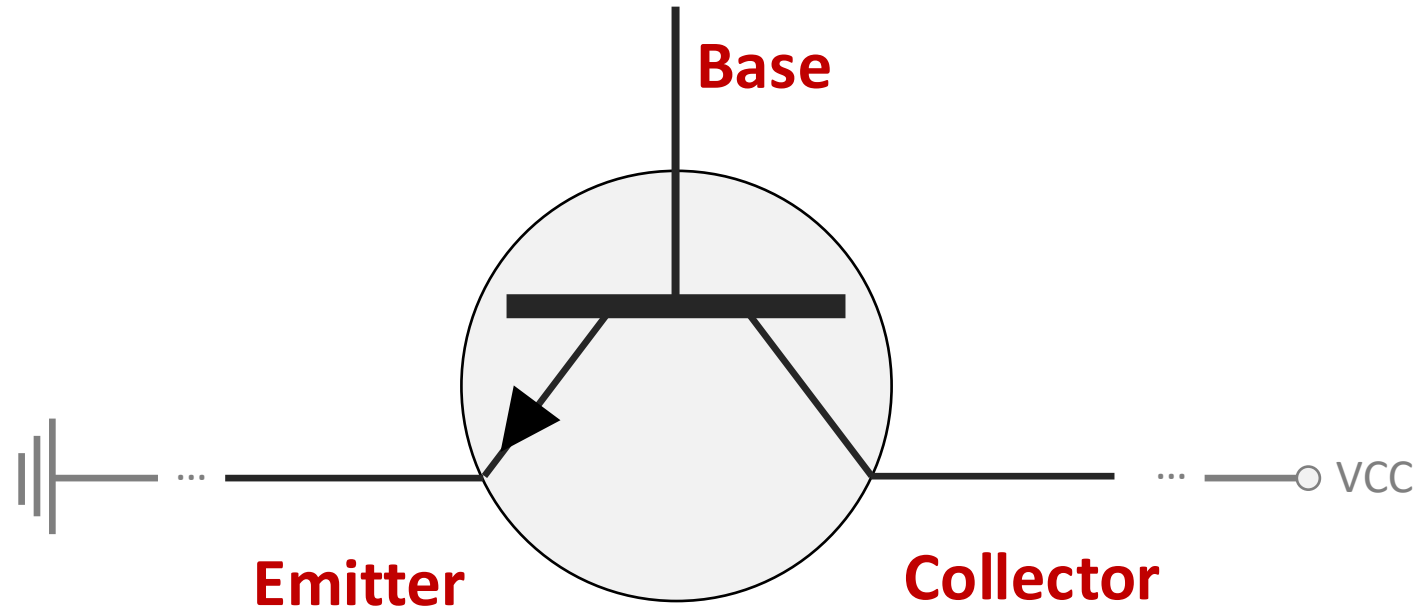
Moore's Law – The number of transistors on integrated circuit chips (1971-2018)

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important as other aspects of technological progress – such as processing speed or the price of electronic products – are linked to Moore's law.

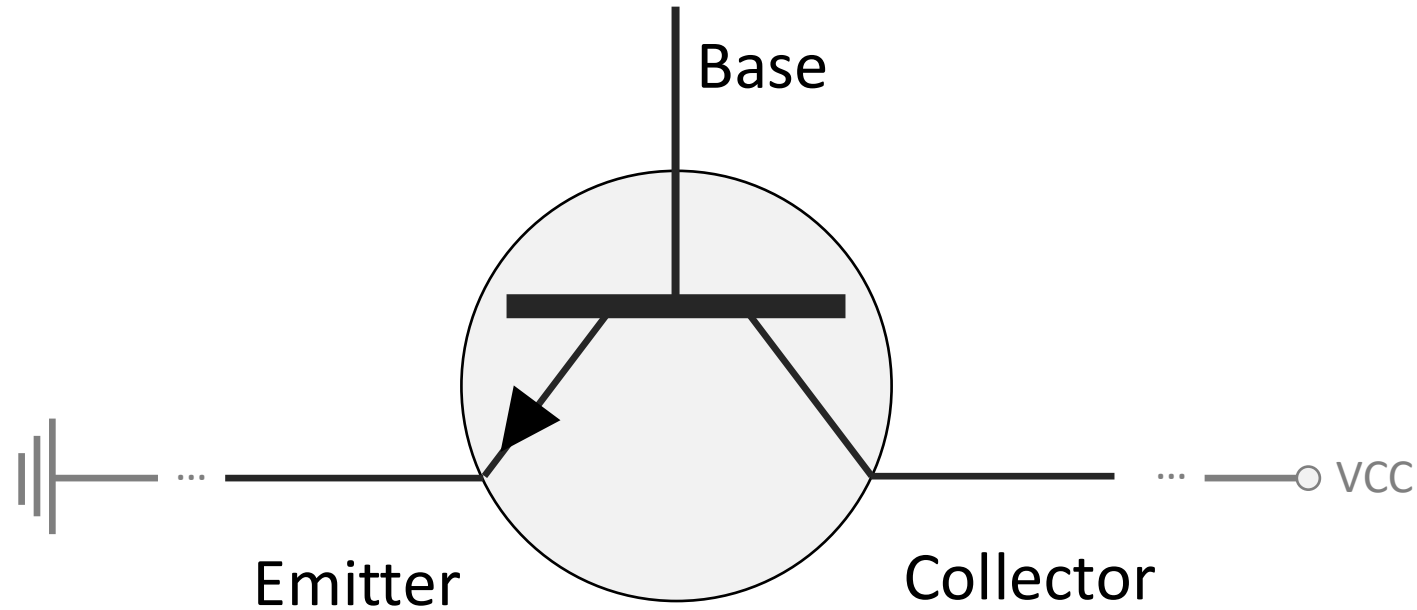


Transistor: Building Block for Logic Gates

3 transistor terminals



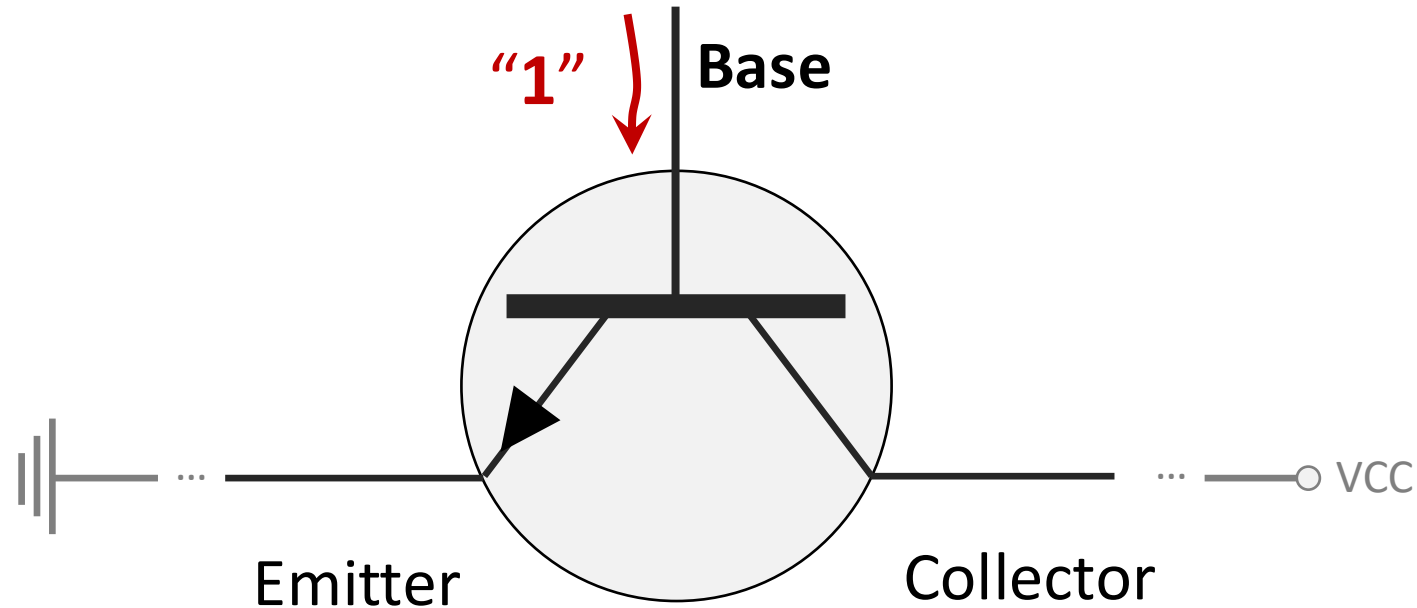
Transistor: Building Block for Logic Gates



Transistor is a fast switch:

If voltage is provided at Base terminal (Base = "1"),
then current flows between Emitter and Collector

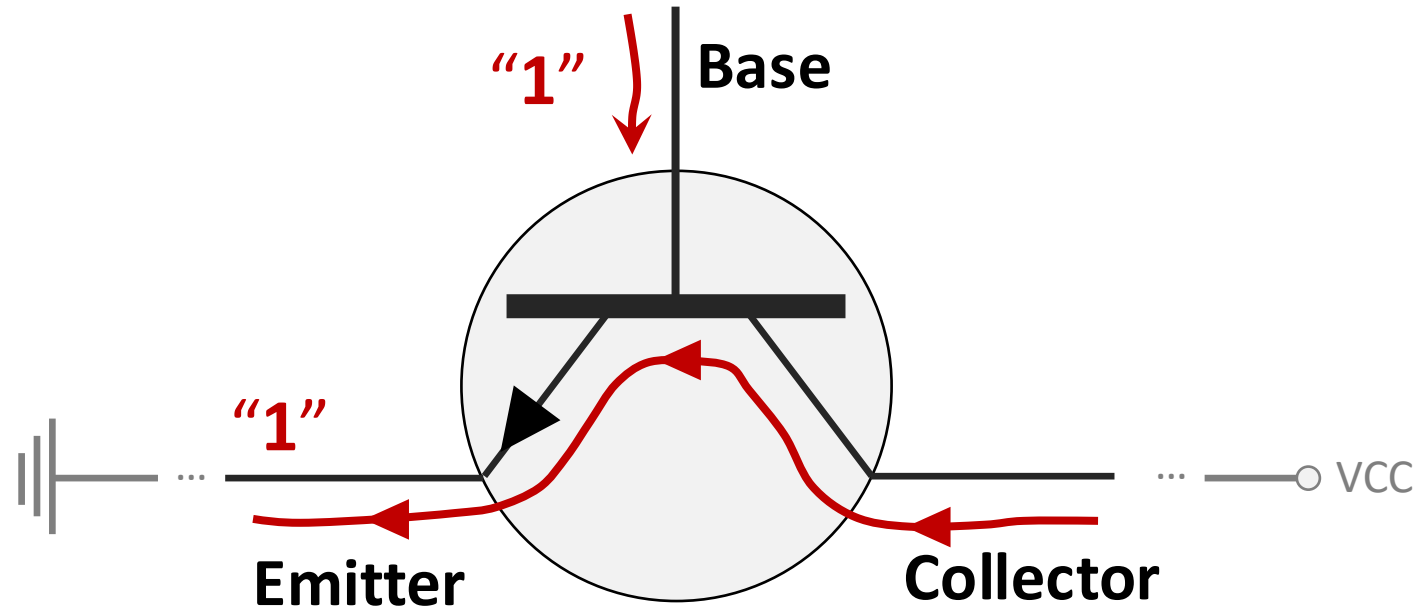
Transistor: Building Block for Logic Gates



Transistor is a fast switch:

If voltage is provided at Base terminal (Base = "1"),
then current flows between Emitter and Collector

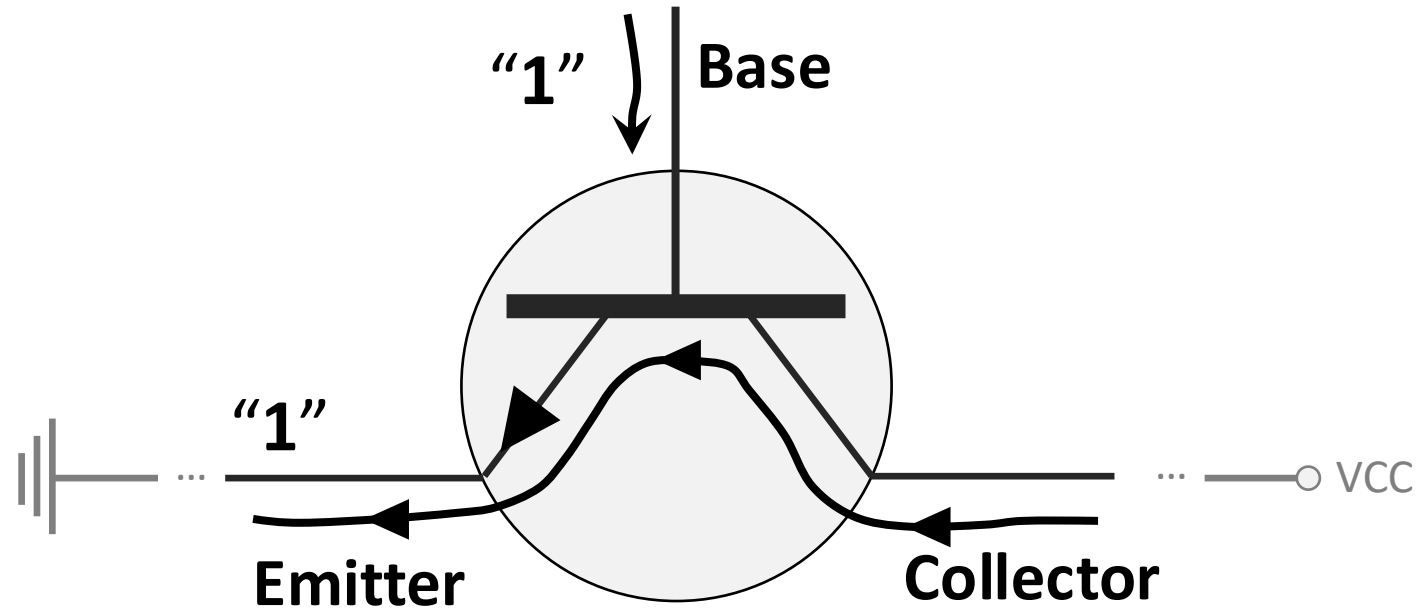
Transistor: Building Block for Logic Gates



Transistor is a fast switch:

If voltage is provided at Base terminal (Base = "1"),
then current flows between Emitter and Collector

Transistor: Building Block for Logic Gates

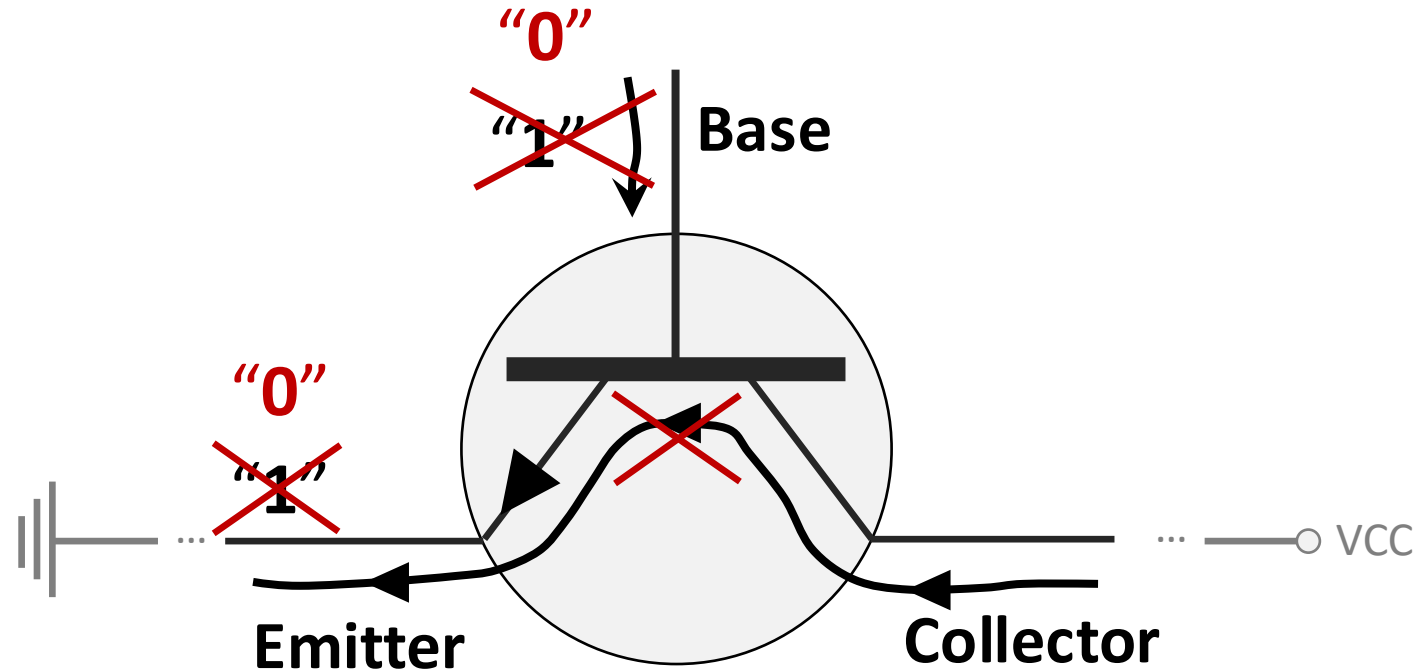


Transistor is a fast switch:

If voltage is provided at Base terminal (Base = "1"),
then current flows between Emitter and Collector

Note: Current direction depends on the transistor type (e.g. NPN or PNP)

Transistor: Building Block for Logic Gates

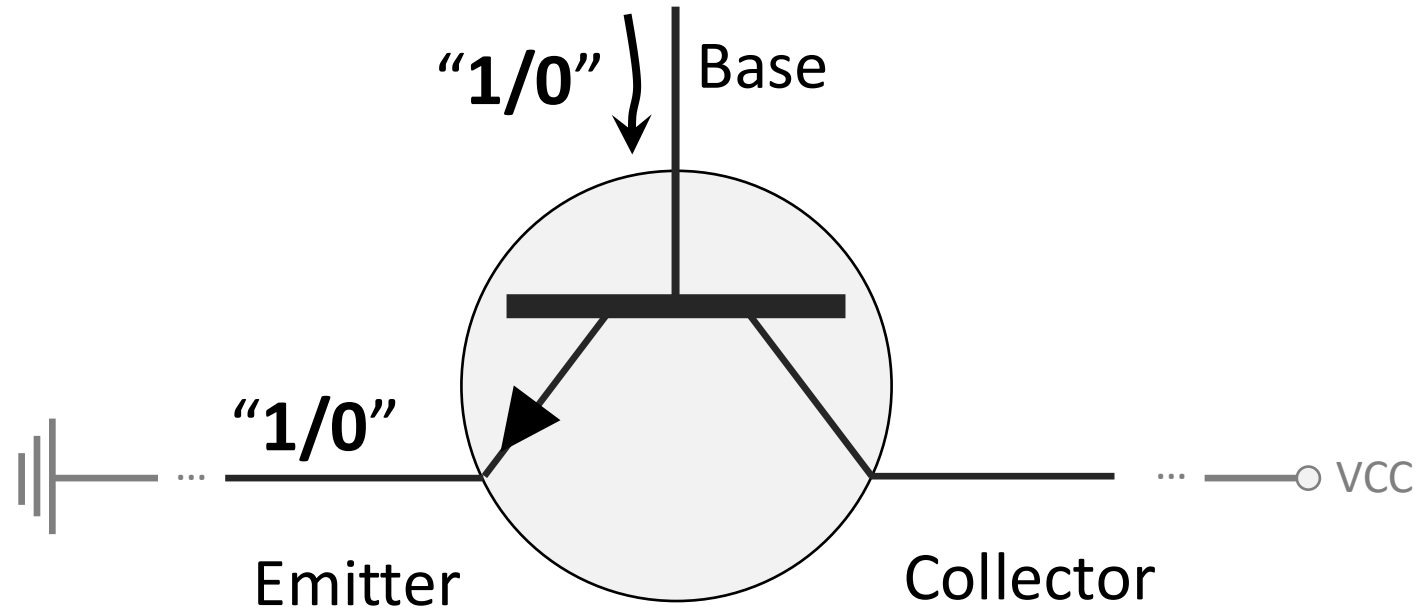


Transistor is a fast switch:

If voltage is provided at Base terminal (Base = "1"),
then current flows between Emitter and Collector;

Otherwise there is no current

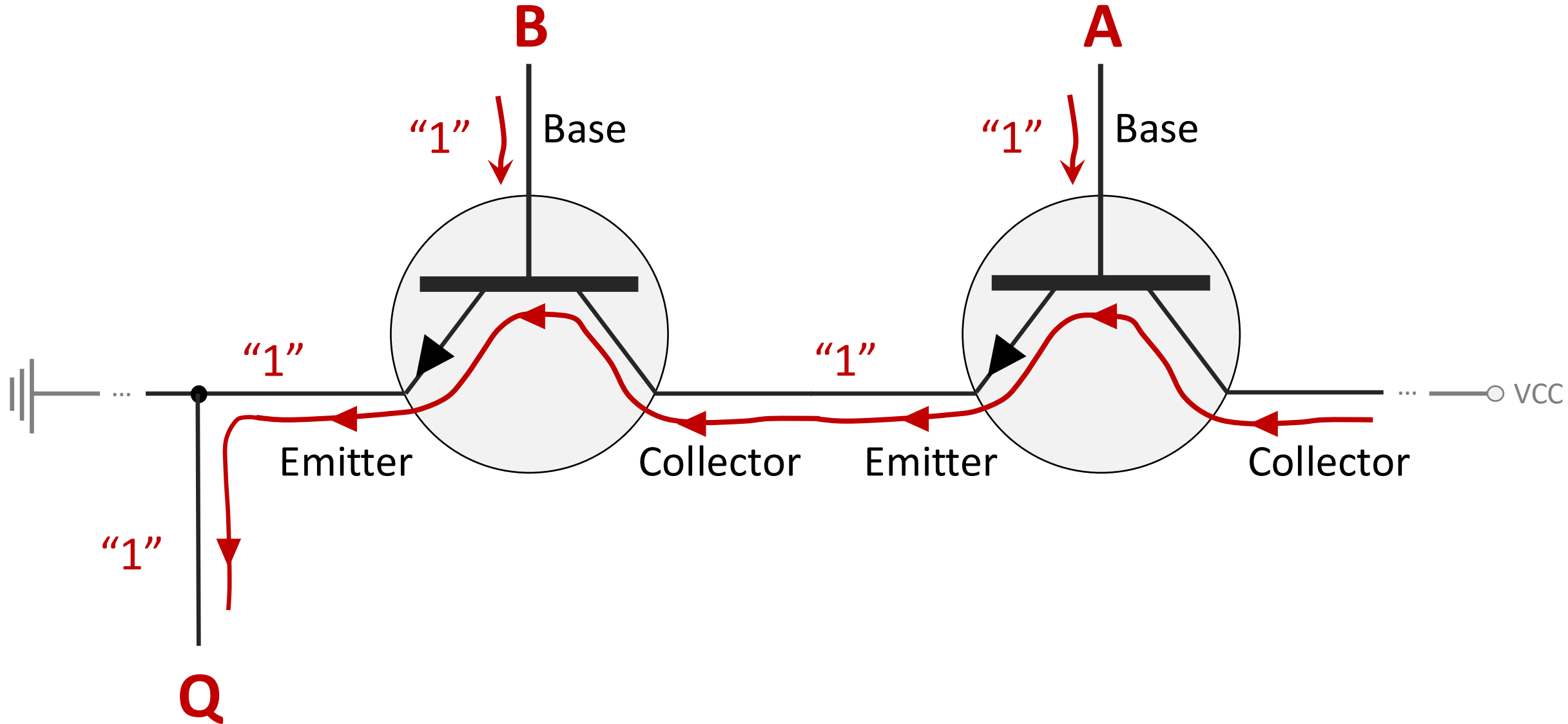
Transistor: Building Block for Logic Gates



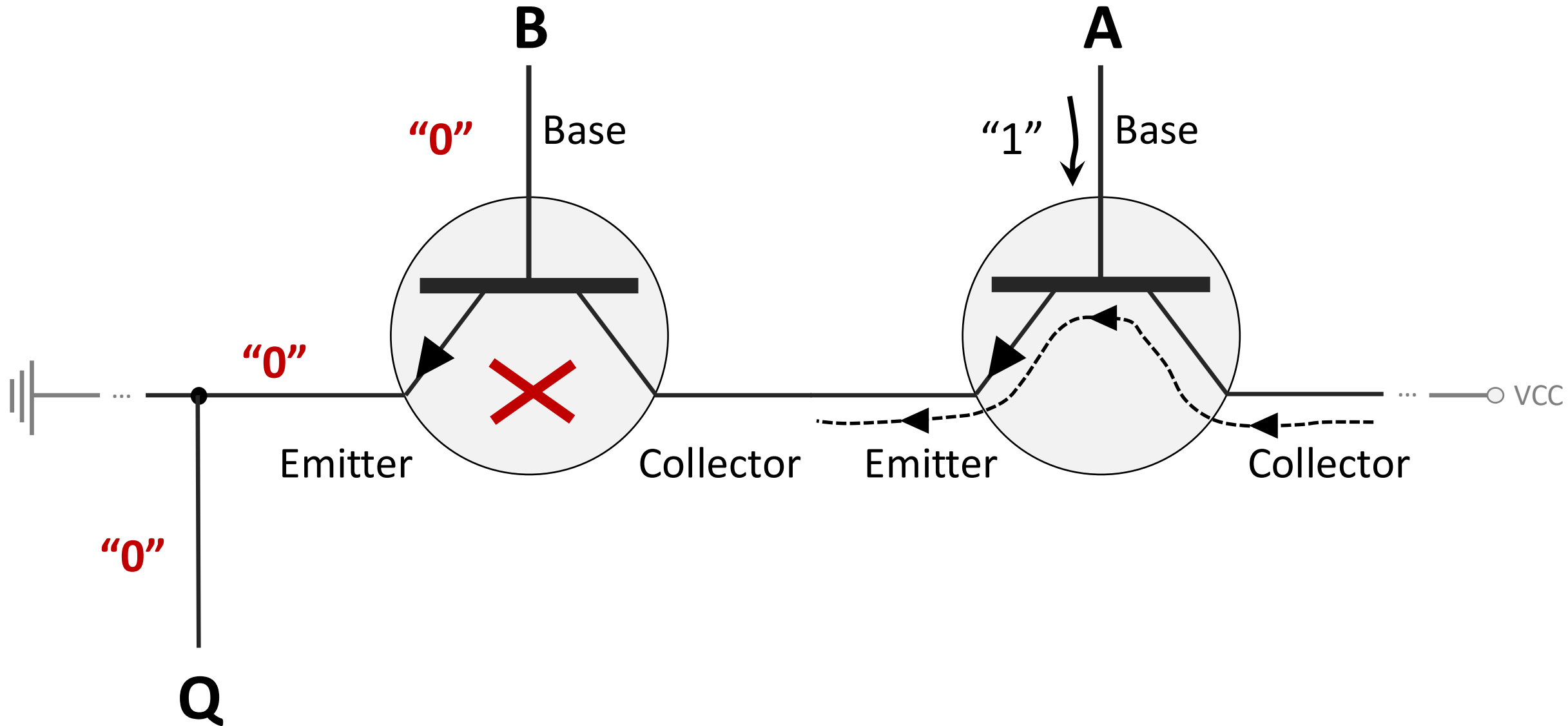
Transistor is a fast switch:

If voltage is provided at Base terminal (Base = "1"),
then current flows between Emitter and Collector;
Otherwise there is no current

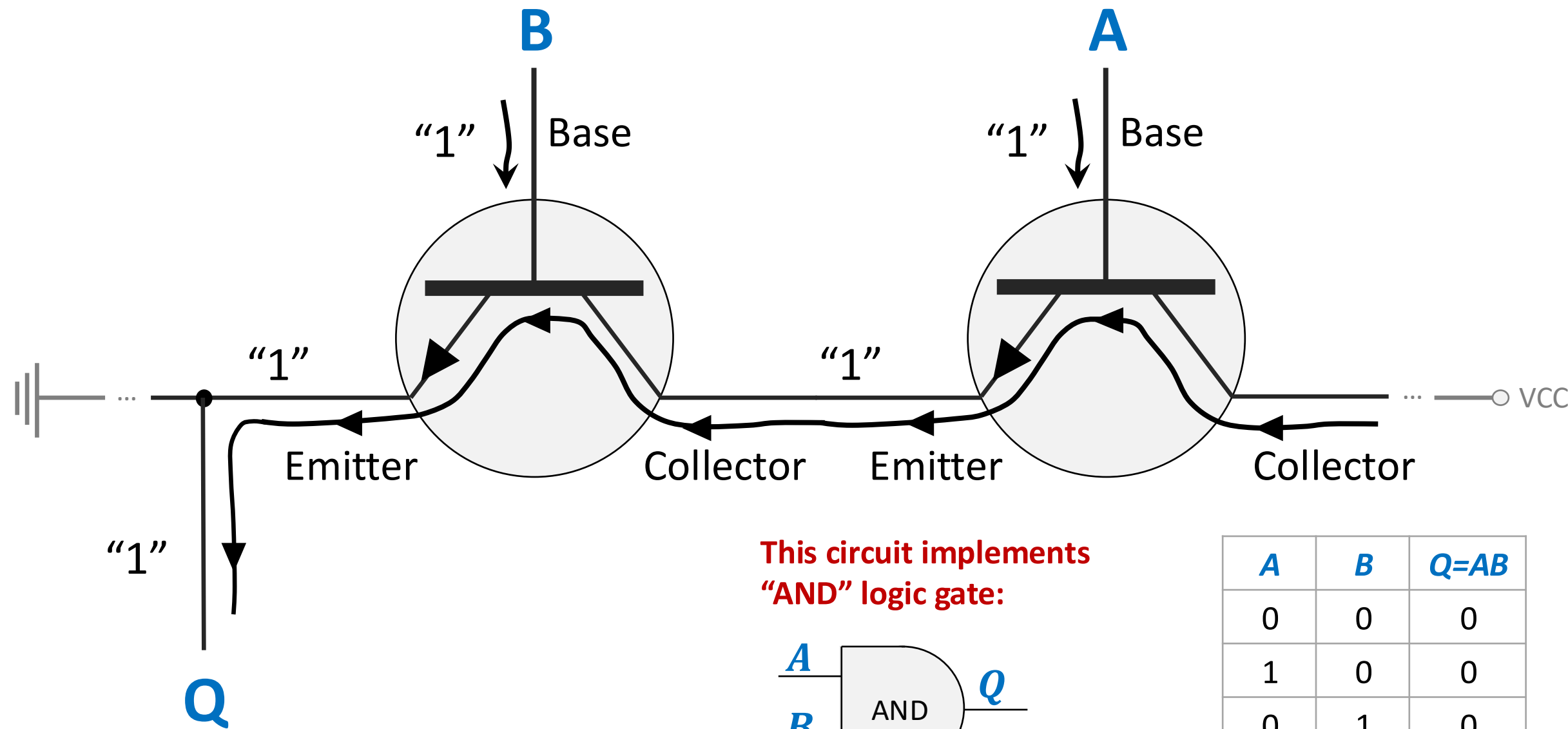
Transistor: Building Block for Logic Gates



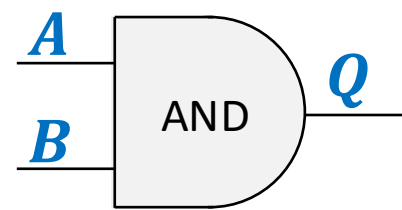
Transistor: Building Block for Logic Gates



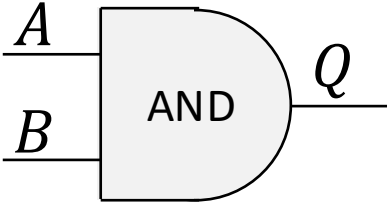
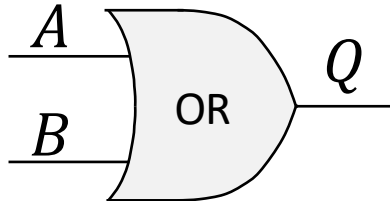
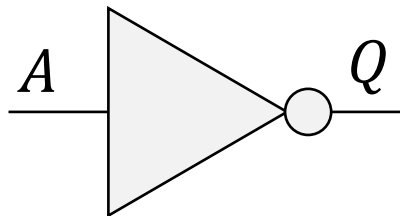
Transistor: Building Block for Logic Gates



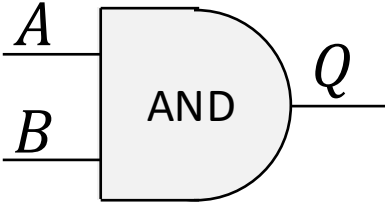
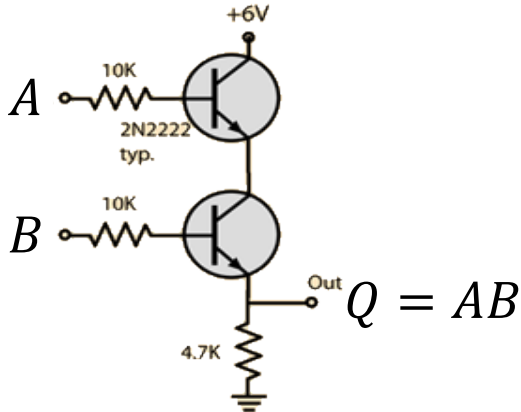
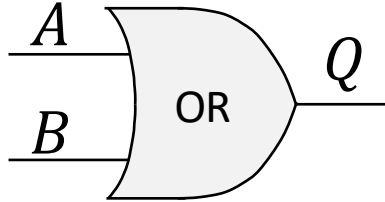
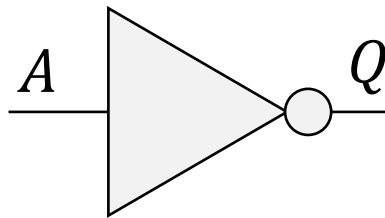
This circuit implements
"AND" logic gate:

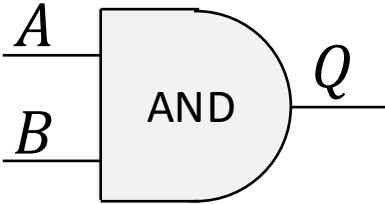
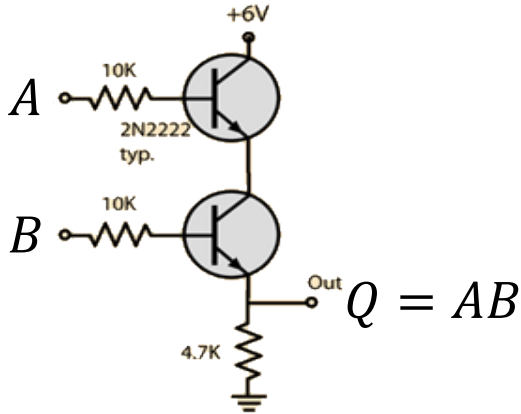
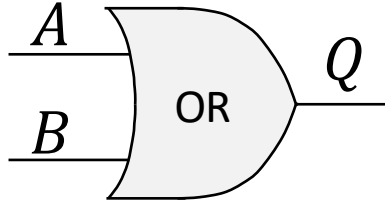
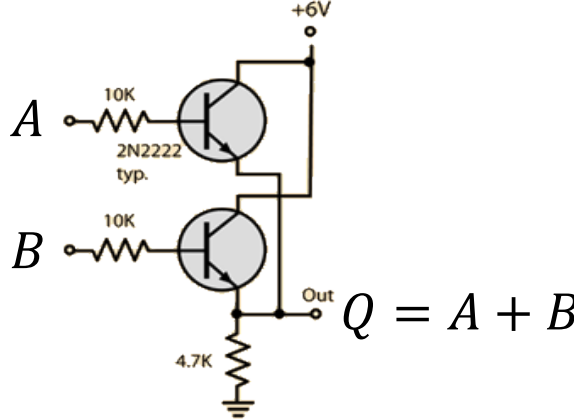
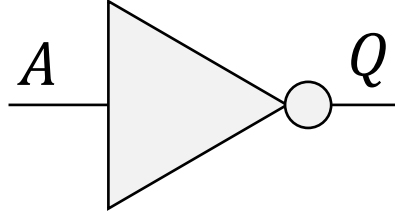


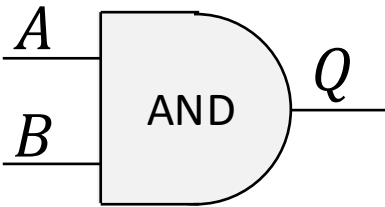
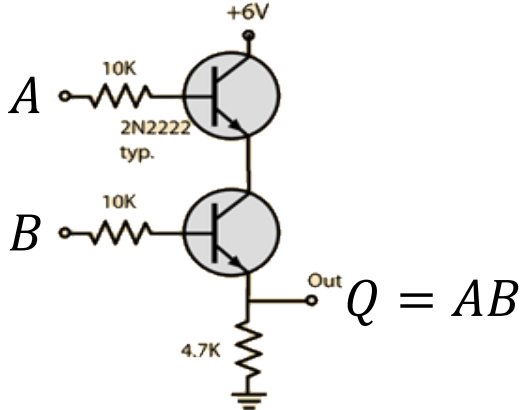
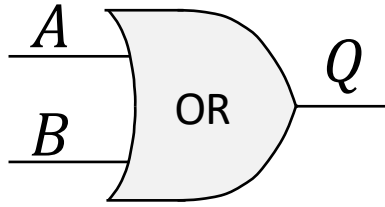
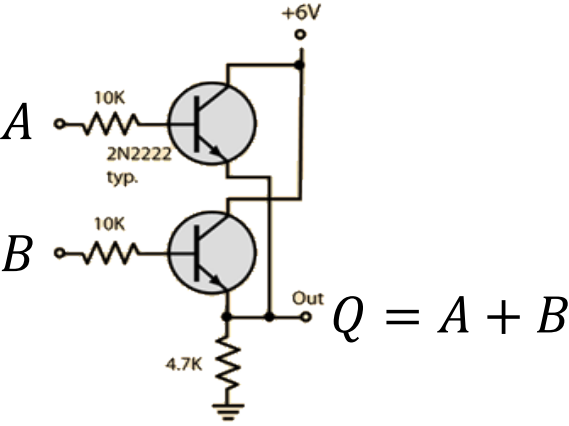
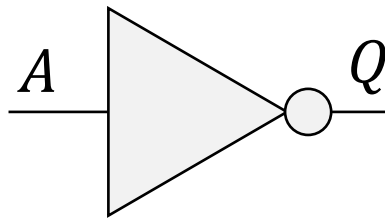
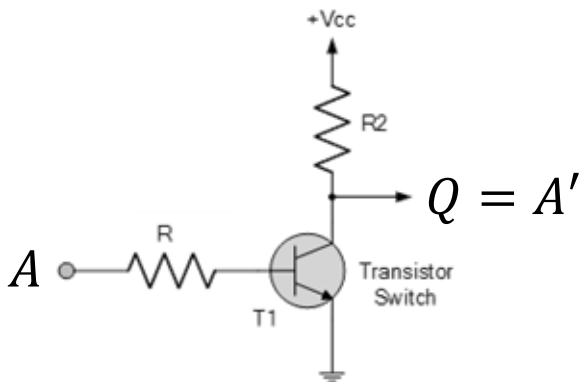
A	B	Q=AB
0	0	0
1	0	0
0	1	0
1	1	1

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															
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																

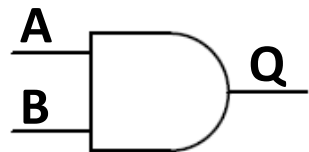
Logic gates correspond to physical devices implemented by using **transistors**

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

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

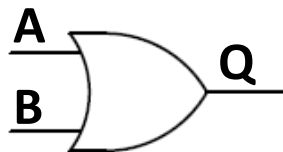
Logic Gate	Symbolic Representation	Truth Table	Implementation with Transistors															
AND		<table><tr><th>A</th><th>B</th><th>Q=AB</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr>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

Basic Logic Gates



AND

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



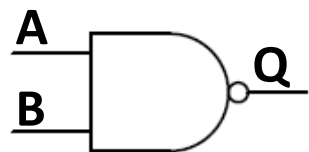
OR

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



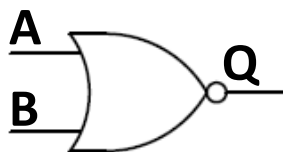
XOR

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

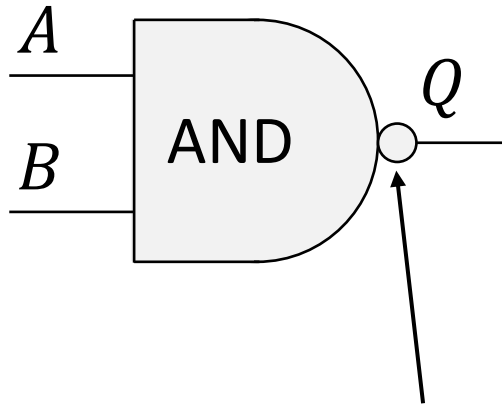
A	B	Output
0	0	1
0	1	0
1	0	0
1	1	0



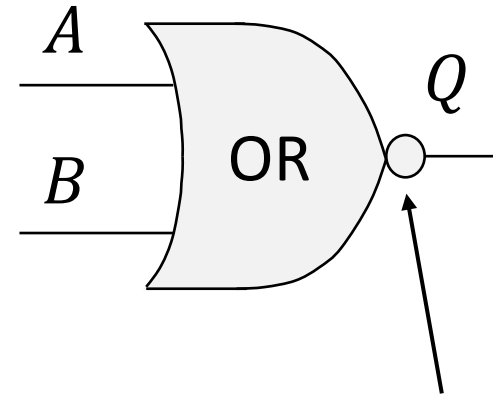
XNOR

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

NAND and NOR Universal Logic Gates

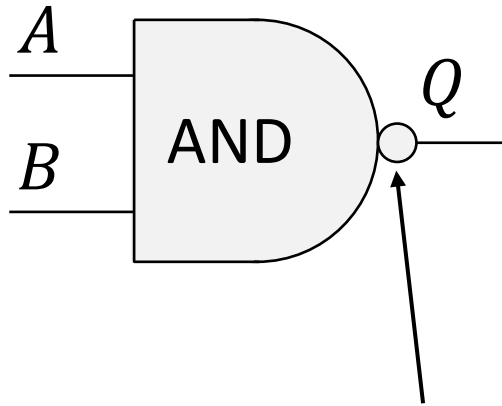


NAND = AND + NOT

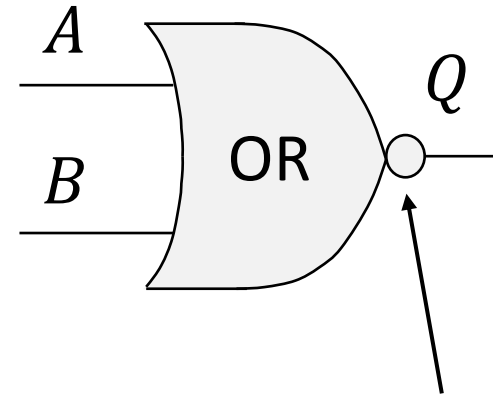


NOR = OR + NOT

NAND and NOR Universal Logic Gates



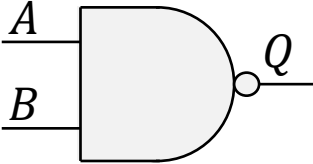
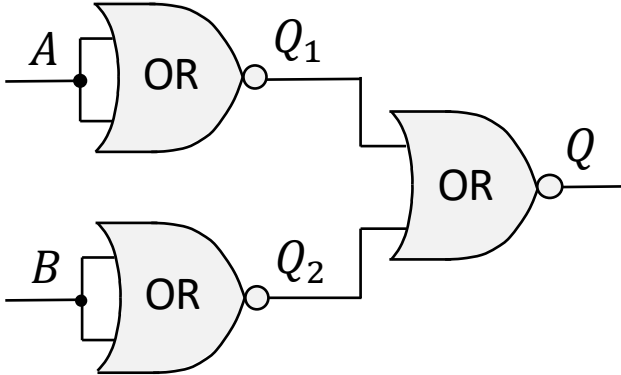
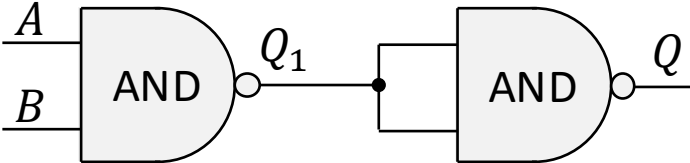
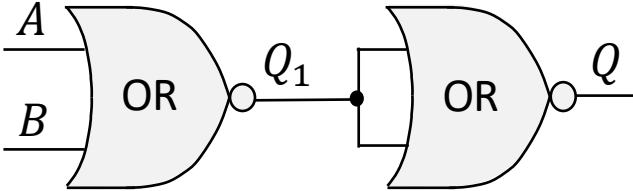
NAND = AND + NOT



NOR = OR + NOT

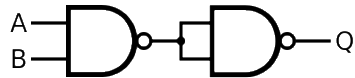
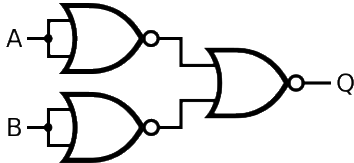
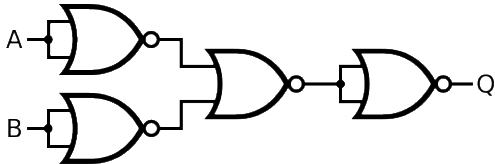
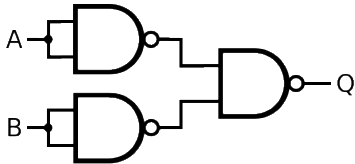
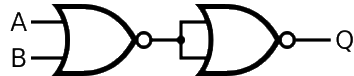
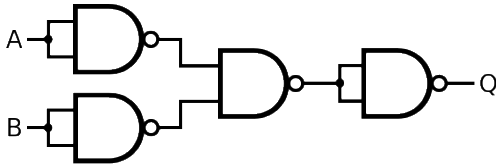
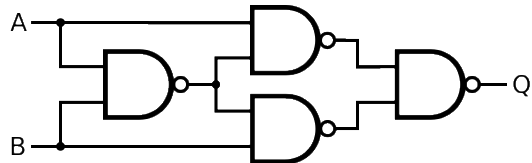
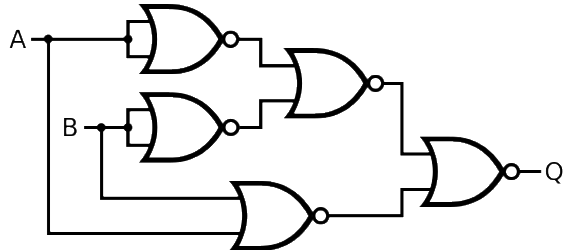
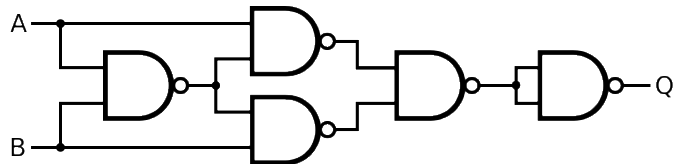
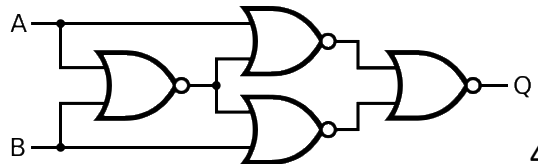
Any other logic gate can be implemented through one of these!

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		

Binary Functions Implementation

Any binary 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

Binary Functions Implementation

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

1) AND, OR, NOT

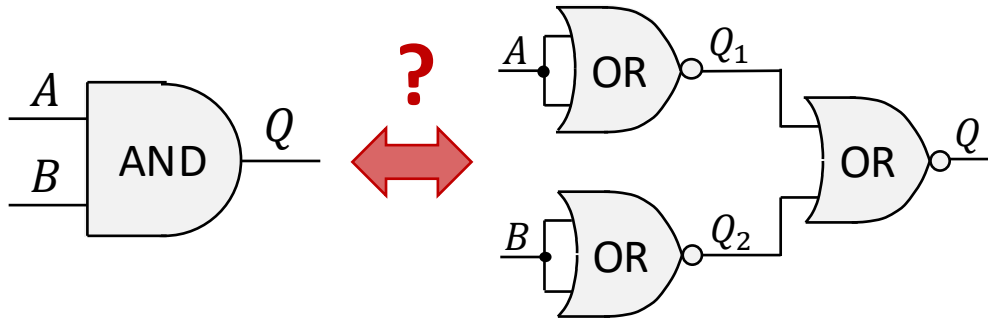
2) NAND – universal logic gate

3) NOR – universal logic gate



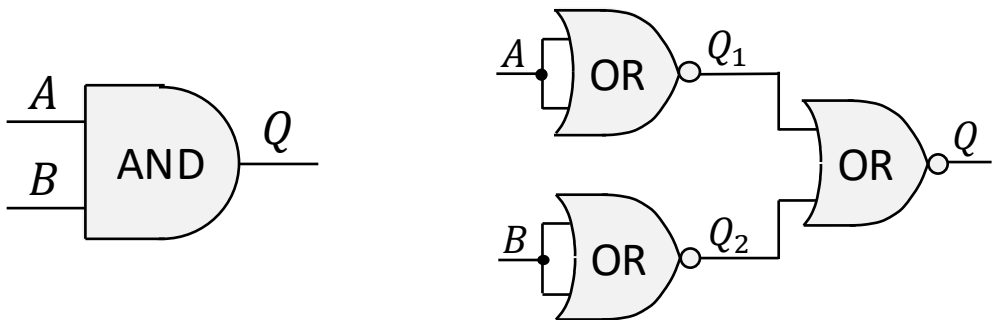
**Proved in Discrete Math as
“Logic Completeness Theorems”**

Proof for Logic Circuits Equivalence



Proof for Logic Circuits Equivalence

Way 1: Truth Tables

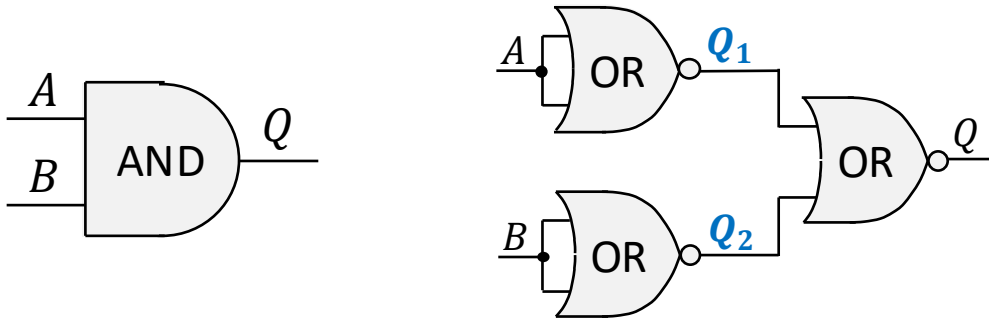


<i>A</i>	<i>B</i>	<i>Q=AB</i>
0	0	0
1	0	0
0	1	0
1	1	1

<i>A</i>	<i>B</i>	<i>Q</i> ₁	<i>Q</i> ₂	<i>Q</i>
0	0	1	1	0
1	0	0	1	0
0	1	1	0	0
1	1	0	0	1

Proof for Logic Circuits Equivalence

Way 1: Truth Tables

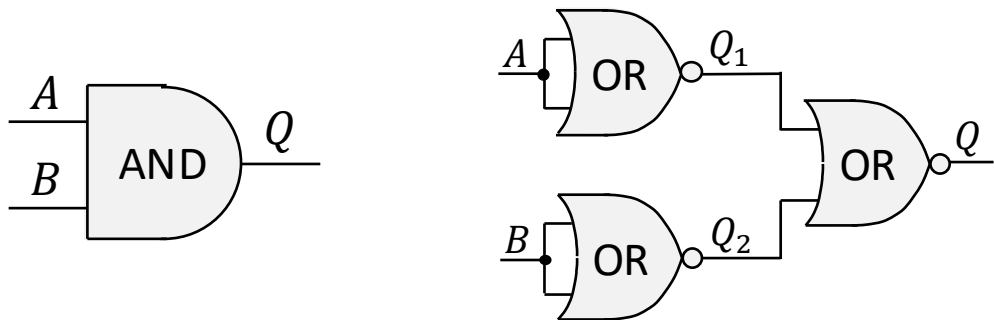


<i>A</i>	<i>B</i>	<i>Q=AB</i>
0	0	0
1	0	0
0	1	0
1	1	1

<i>A</i>	<i>B</i>	<i>Q</i> ₁	<i>Q</i> ₂	<i>Q</i>
0	0	1	1	0
1	0	0	1	0
0	1	1	0	0
1	1	0	0	1

Proof for Logic Circuits Equivalence

Way 1: Truth Tables



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

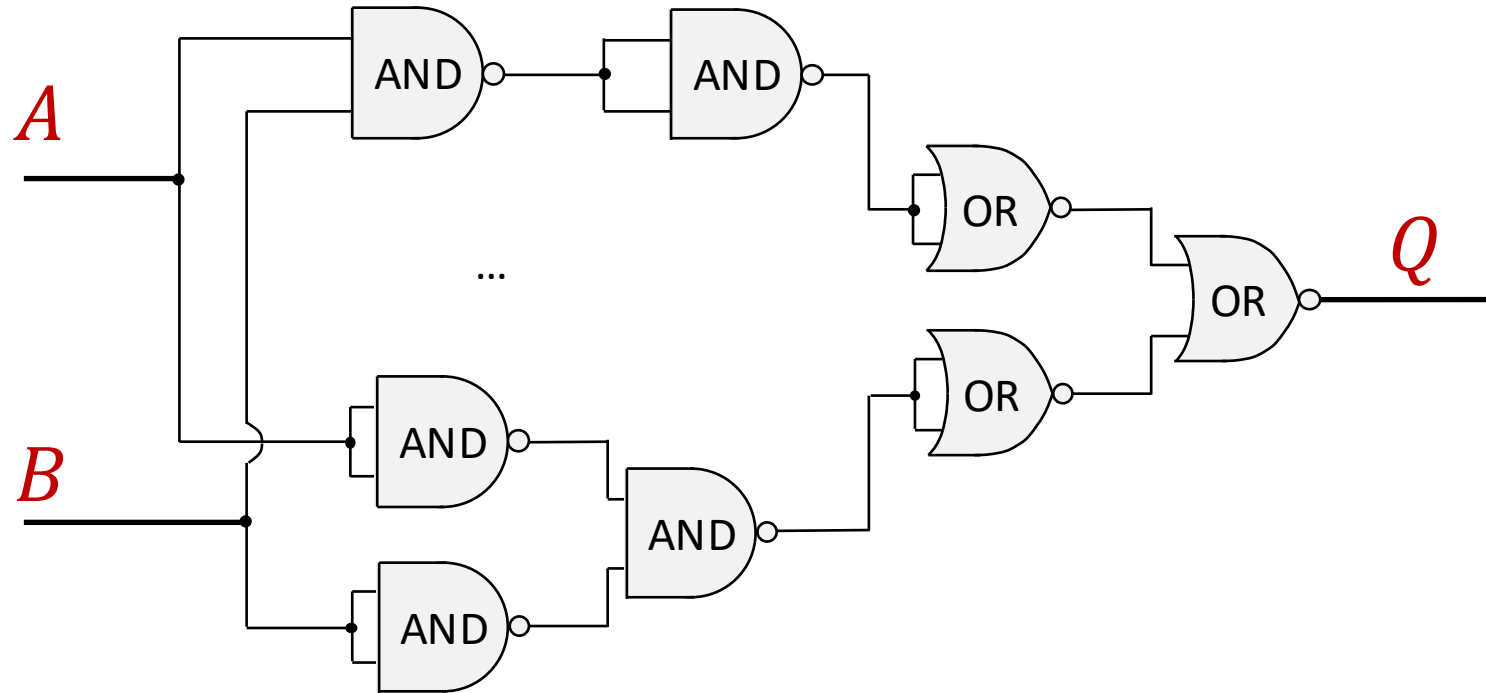
A	B	Q ₁	Q ₂	Q
0	0	1	1	0
1	0	0	1	0
0	1	1	0	0
1	1	0	0	1

Way 2: Boolean Algebra Laws

Name	AND Form	OR Form
Identity Law	$1A = A$	$0 + A = A$
Null Law	$0A = 0$	$1 + A = 1$
Idempotent Law	$AA = A$	$A + A = A$
Inverse Law	$AA' = 0$	$A + A' = 1$
Commutative Law	$AB = BA$	$A + B = B + A$
Associative Law	$(AB)C = A(BC)$	$(A + B) + C = A + (B + C)$
Distributive Law	$A + BC = (A + B)(A + C)$	$A(B + C) = AB + AC$
Absorption Law	$A(A + B) = A$	$A + AB = A$
De Morgan's Law	$\overline{AB} = \overline{A} + \overline{B}$	$\overline{A + B} = \overline{A} \overline{B}$

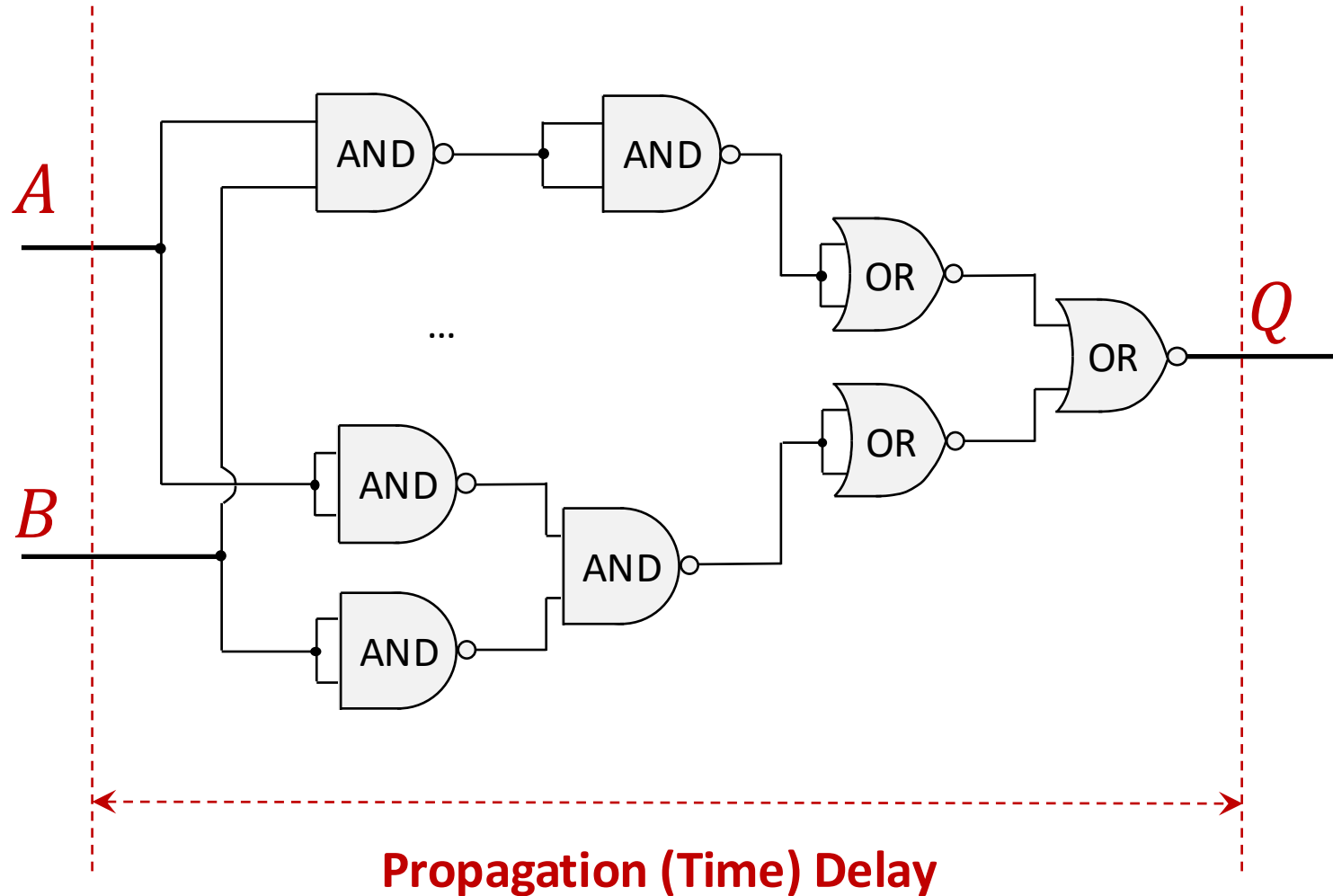
Propagation (Time) Delay

A time delay between the updates of input and output signals



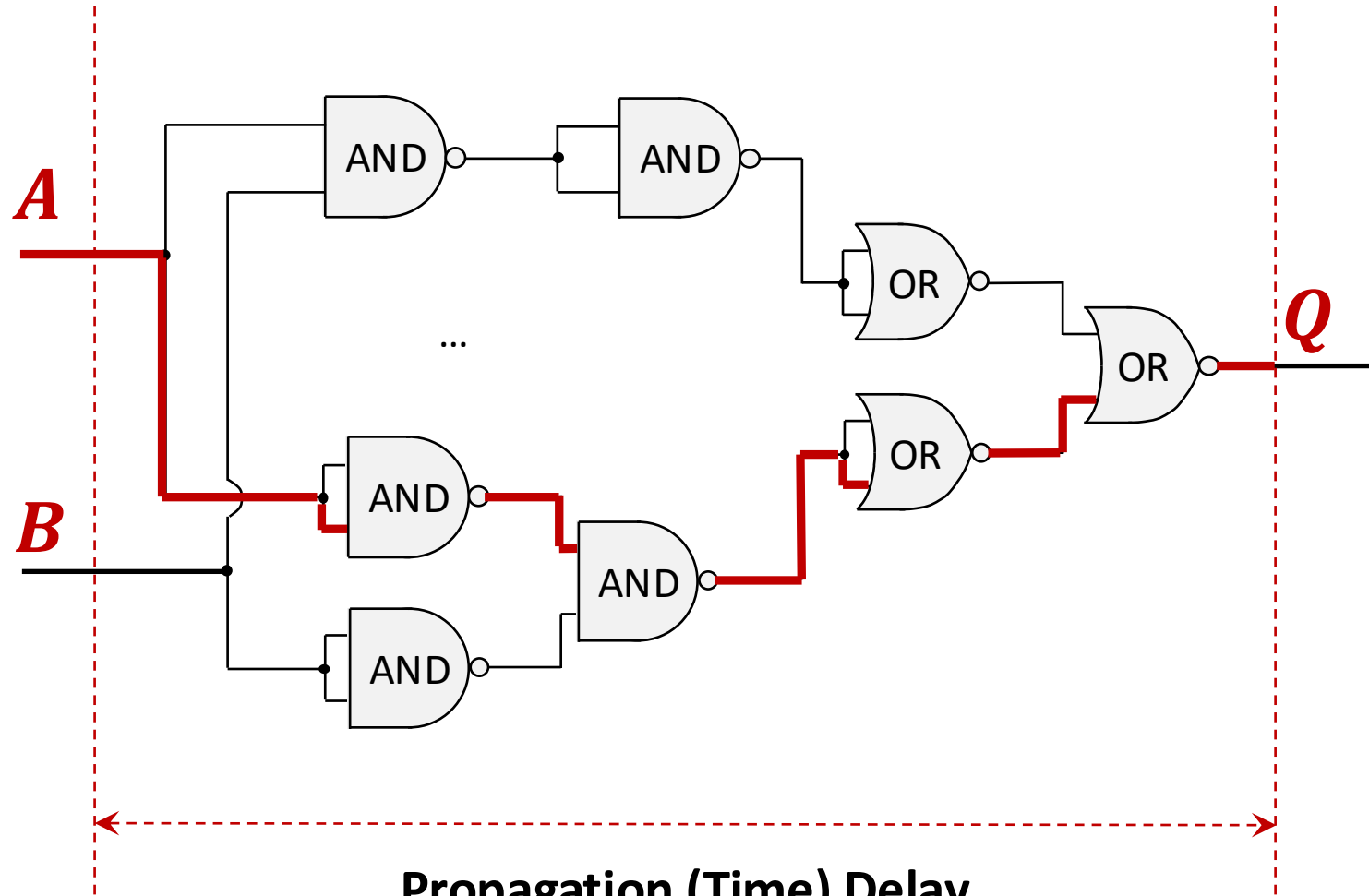
Propagation (Time) Delay

A time delay between the updates of input and output signals



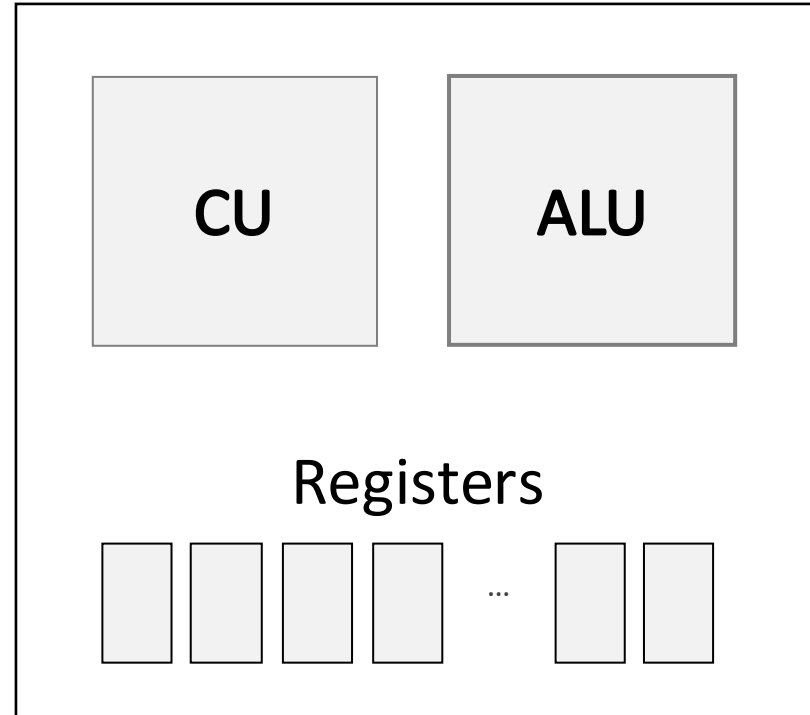
Critical Path in a Circuit

The path with the largest propagation delay

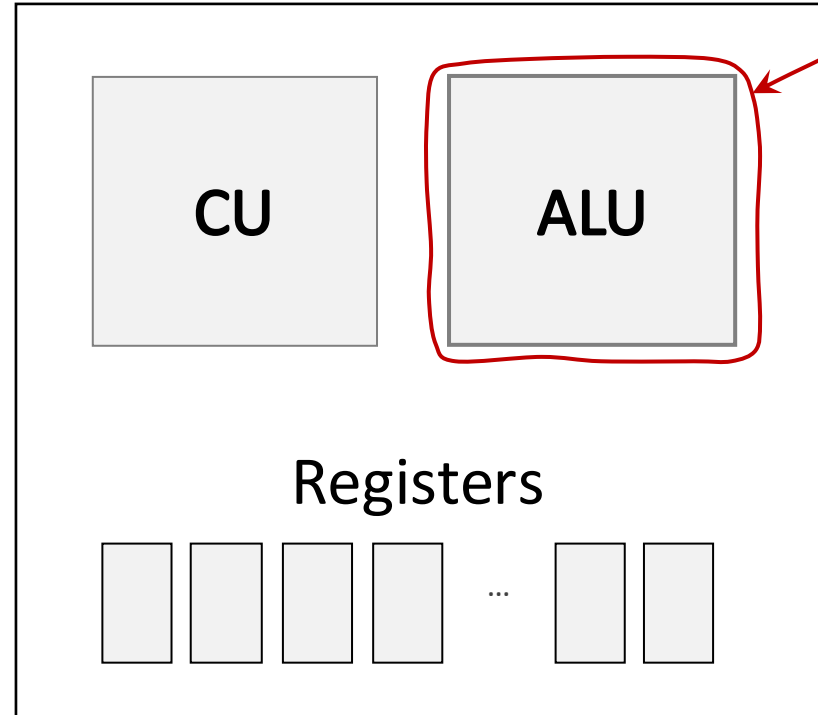


Critical Path – the one taking the longest time

Types of Logic Circuits for CPU Components



Types of Logic Circuits for CPU Components



**Combinational logic
circuit for arithmetical
and logic operations**

