BENHA UNIVERSITY
FACULTY OF ENGINEERING AT SHOUBRA

**ECE-291**
**Electronic Engineering**

# Lecture #5
# Basics of Digital Logic Design

**Instructor:**
**Dr. Ahmad El-Banna**



SPRING 2015

# Agenda

- Basic Concepts
- Logic Gates
- Examples of Logic Circuits

Digital logic is the foundation for digital computers!

# BASIC CONCEPTS

# Digital Electronics

- **Digital electronic circuits**, are electronics that represent signals by **discrete** bands of analog **levels**, rather than by continuous ranges (as used in analogue electronics).

- In most cases the number of **states** is **two**.
- They are represented by **two voltage bands**: one near a reference value (typically termed as "***ground***" or zero volts), and the other a value near the ***supply*** voltage.
- These correspond to the "false" ("**0**") and "true" ("**1**") values of the **Boolean** domain, respectively, yielding **binary** code.

| Boolean Algebra | Boolean Logic | Voltage State |
|---|---|---|
| Logic "1" | True (T) | High (H) |
| Logic "0" | False (F) | Low (L) |

# Digital Electronics..

**Advantages:**

- Digital techniques are useful because it is *easier* to get an electronic device to switch into one of a number of known states than to accurately reproduce a continuous range of values.
- Signals represented digitally can be *transmitted without degradation due to noise*.
- Computer-controlled digital systems can be *controlled by software*, allowing new functions to be added without changing hardware.

**Disadvantage:**

- In some cases, digital circuits *use more energy* than analog circuits to accomplish the same tasks, thus *producing more heat* which *increases the complexity* of the circuits such as the inclusion of heat sinks.

# Digital Electronics…

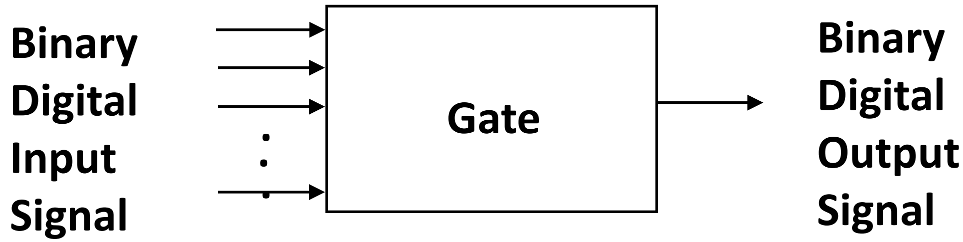
- A digital circuit is often constructed from small electronic circuits called **logic gates** that can be used to create combinational logic.
- Each logic gate represents a **function of boolean logic**.
- A logic gate is an arrangement of electrically controlled switches, better known as **transistors**.
- Logic gates often use the **fewest number of transistors** in order to reduce their size, power consumption and cost, and increase their reliability.
- **Integrated circuits** are the **least expensive** way to make logic gates in large volumes. Integrated circuits are usually designed by engineers using electronic design automation software

# Classification of Integrated Circuits

- Small Scale Integration or (**SSI**) – Contain up to 10 transistors or a few gates within a single package such as AND, OR, NOT gates.
- Medium Scale Integration or (**MSI**) – between 10 and 100 transistors or tens of gates within a single package and perform digital operations such as adders, decoders, counters, flip-flops and multiplexers.
- Large Scale Integration or (**LSI**) – between 100 and 1,000 transistors or hundreds of gates and perform specific digital operations such as I/O chips, memory, arithmetic and logic units.
- Very-Large Scale Integration or (**VLSI**) – between 1,000 and 10,000 transistors or thousands of gates and perform computational operations such as processors, large memory arrays and programmable logic devices.
- Super-Large Scale Integration or (**SLSI**) – between 10,000 and 100,000 transistors within a single package and perform computational operations such as microprocessor chips, micro-controllers, basic PICs and calculators.
- Ultra-Large Scale Integration or (**ULSI**) – more than 1 million transistors – the big boys that are used in computers CPUs, GPUs, video processors, micro-controllers, FPGAs and complex PICs.

# LOGIC GATES

# BASIC LOGIC BLOCK (GATE )




- **Types of Basic Logic Blocks**

  **- Combinational Logic Block**

  Logic Blocks whose output logic value depends only on the input logic values

  **- Sequential Logic Block**

  Logic Blocks whose output logic value depends on the input values and the state (stored information) of the blocks

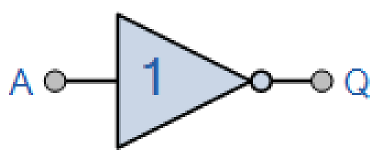
- **Functions of Gates can be described by**

  - Truth Table
  - Boolean Function
  - Karnaugh Map

# NOT Gate -- Inverter

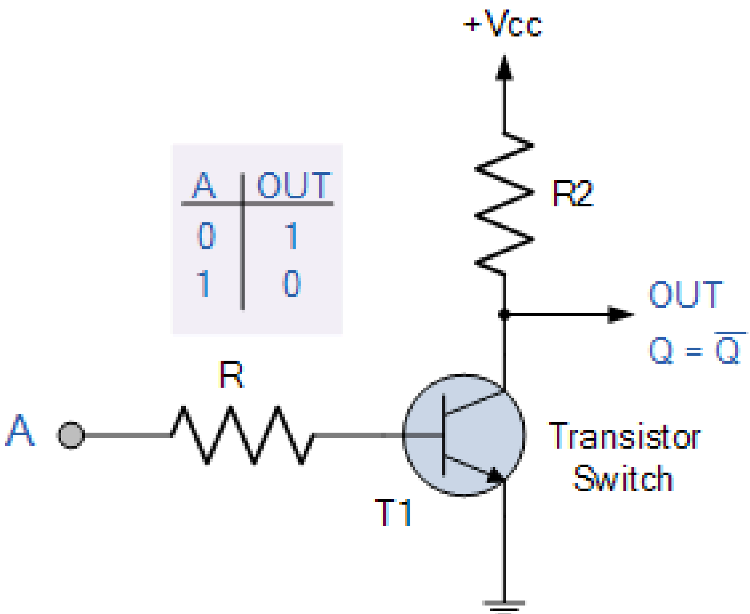
If A is NOT true, then Q is true

- The Logic NOT Gate Truth Table

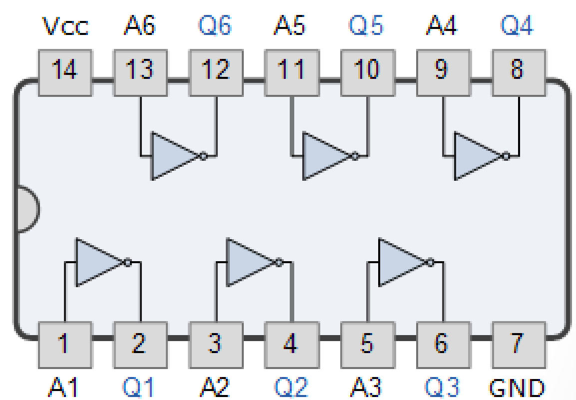
| Symbol | Truth Table | |
| --- | --- | --- |
|   Inverter or NOT Gate | A | Q |
| | 0 | 1 |
| | 1 | 0 |
| **Boolean Expression Q = not A or $\overline{A}$** | **Read as inverse of A gives Q** | |

# NOT Gate – Inverter..

- Transistor NOT Gate



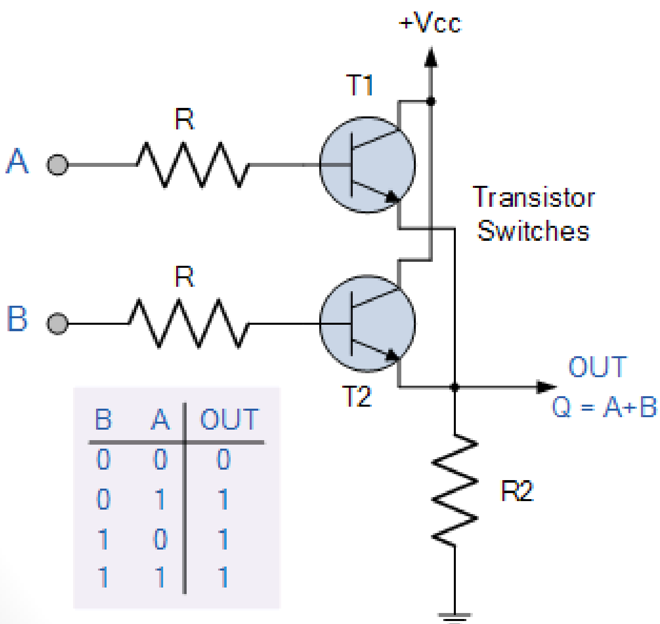

- 7404 NOT Gate or Inverter



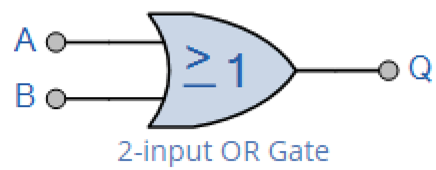
# OR Gate

If either A or B is true, then Q is true

- 2-input Transistor OR Gate



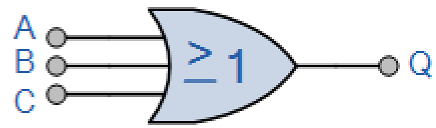

- The 2-input Logic OR Gate

| Symbol | Truth Table | | |
|---|---|---|---|
|  | B | A | Q |
| | 0 | 0 | 0 |
| | 0 | 1 | 1 |
| | 1 | 0 | 1 |
| | 1 | 1 | 1 |
| Boolean Expression Q = A+B | Read as A OR B gives Q | | |

# OR Gate..

- The 3-input Logic OR Gate

**Symbol**



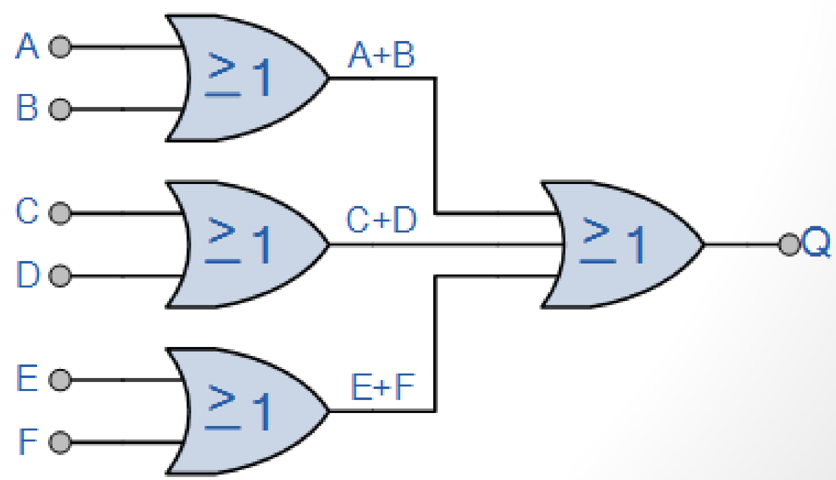

3-input OR Gate

**Truth Table**

| C | B | A | Q |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

**Boolean Expression Q = A+B+C** — **Read as A OR B OR C gives Q**

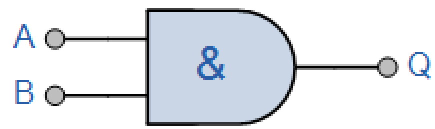
- Multi-input OR Gate

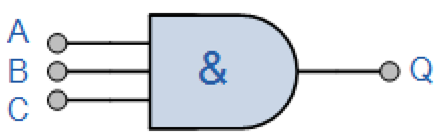



6-input “OR” Function

# AND Gate

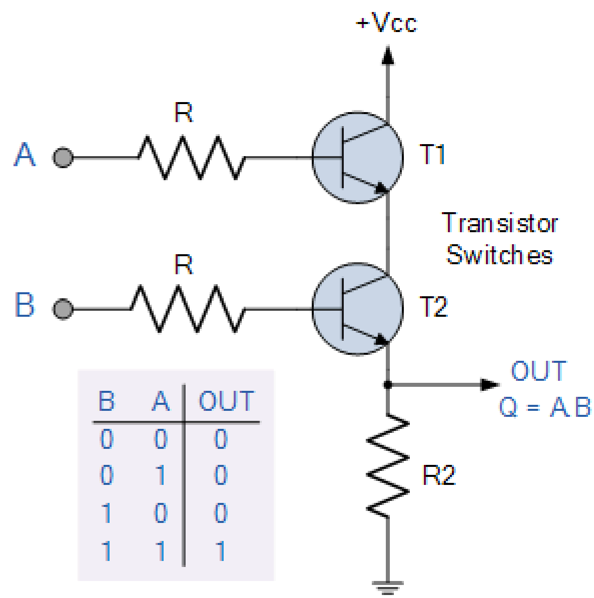
If both A and B are true, then Q is true

| Symbol | Truth Table | | |
|---|---|---|---|
|  2-input AND Gate | B | A | Q |
| | 0 | 0 | 0 |
| | 0 | 1 | 0 |
| | 1 | 0 | 0 |
| | 1 | 1 | 1 |
| **Boolean Expression Q = A.B** | **Read as A AND B gives Q** | | |

| Symbol | Truth Table | | | |
|---|---|---|---|---|
|  3-input AND Gate | C | B | A | Q |
| | 0 | 0 | 0 | 0 |
| | 0 | 0 | 1 | 0 |
| | 0 | 1 | 0 | 0 |
| | 0 | 1 | 1 | 0 |
| | 1 | 0 | 0 | 0 |
| | 1 | 0 | 1 | 0 |
| | 1 | 1 | 0 | 0 |
| | 1 | 1 | 1 | 1 |
| **Boolean Expression Q = A.B.C** | **Read as A AND B AND C gives Q** | | | |

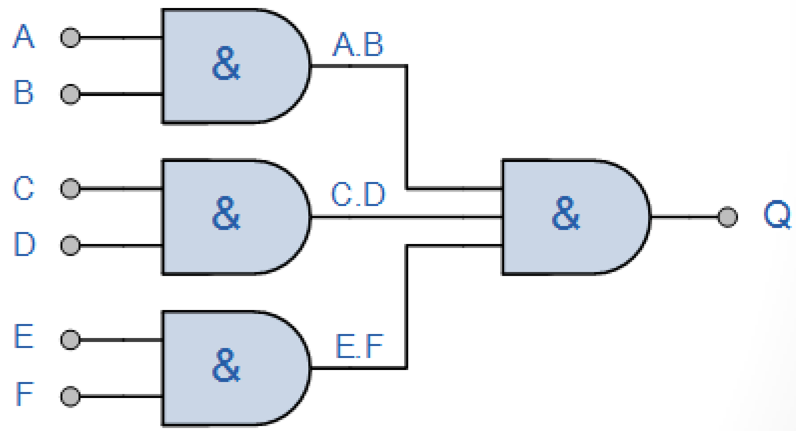
# AND Gate..

- 2-input Transistor AND Gate



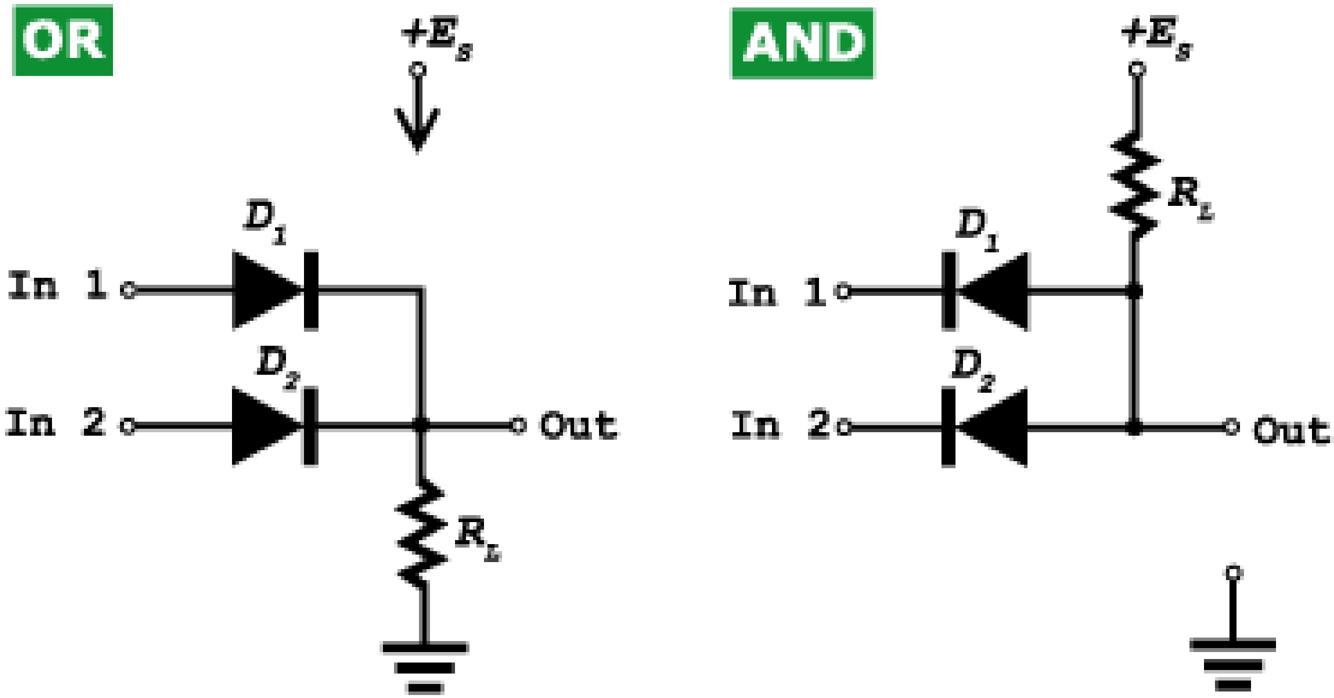

| B | A | OUT |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

- Multi-input AND Gate




6-input "AND" Function

# OR/AND Gates using diodes




**OR**

| In 1 | In 2 | Out |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

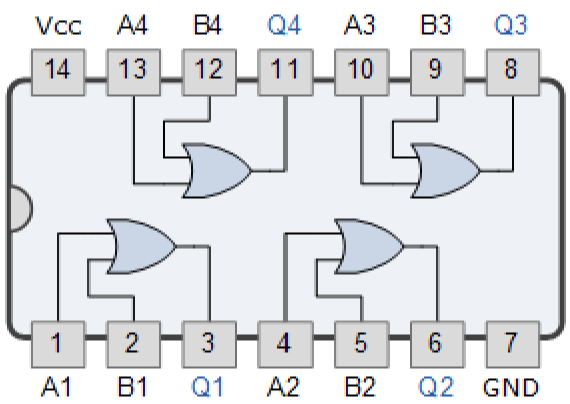
0 - 0V
1 - Es

**AND**

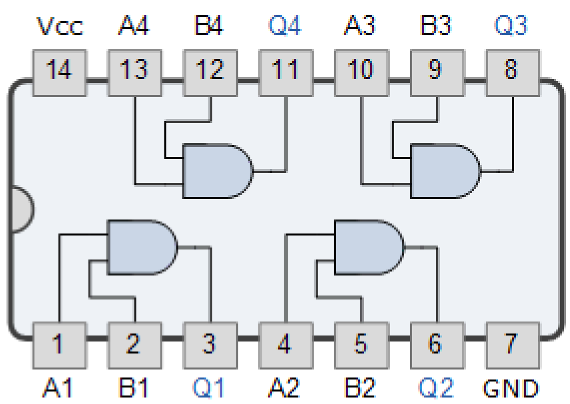
| In 1 | In 2 | Out |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

# 74xx IC

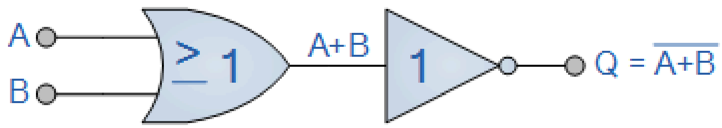
- 7432 Quad 2-input Logic OR Gate



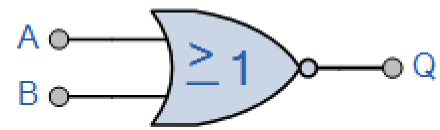

- 7408 Quad 2-input AND Gate

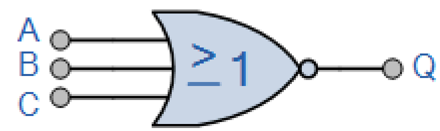


# NOR Gate

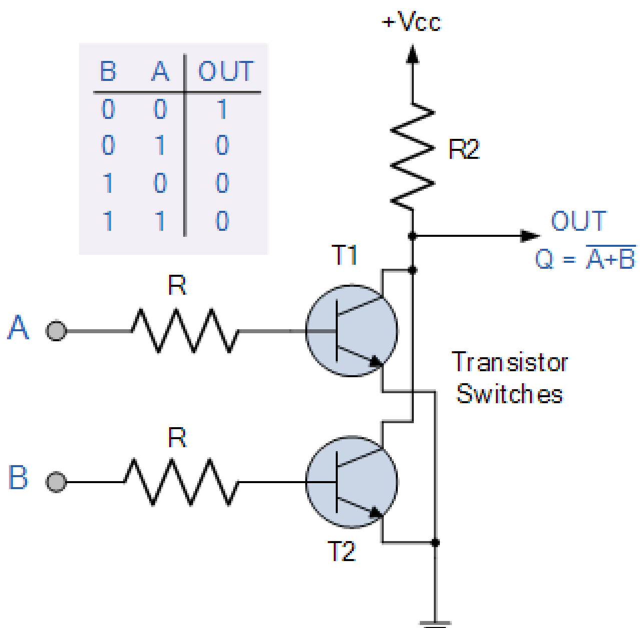



2-input "OR" gate plus a "NOT" gate

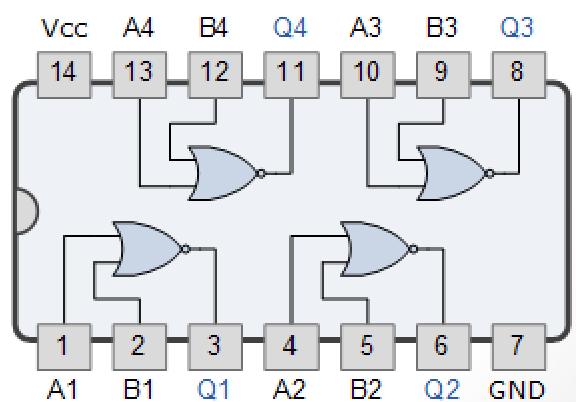


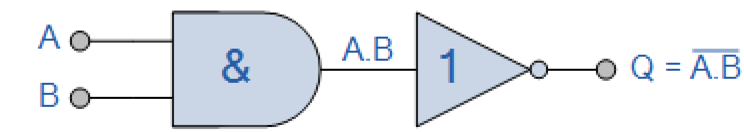

| B | A | OUT |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |



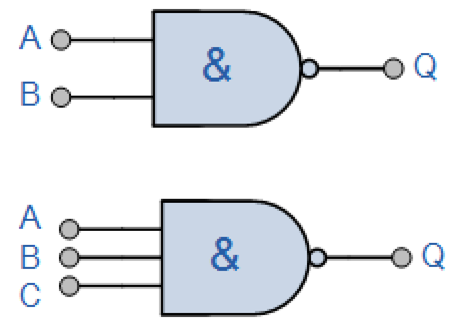

7402 Quad 2-input NOR Gate



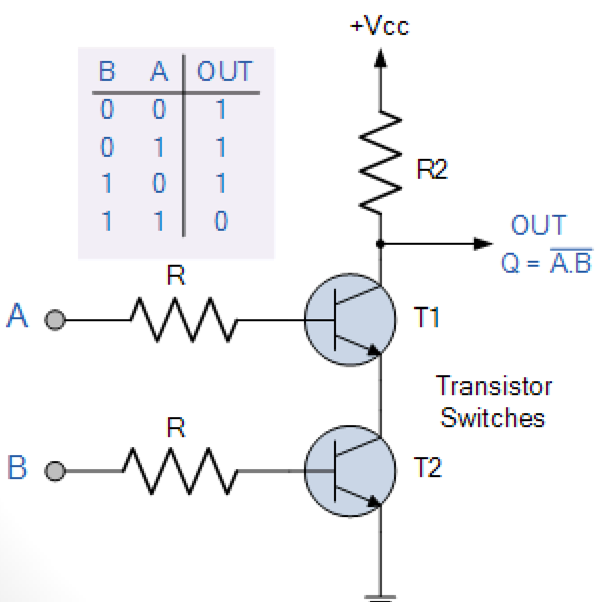
# NAND Gate



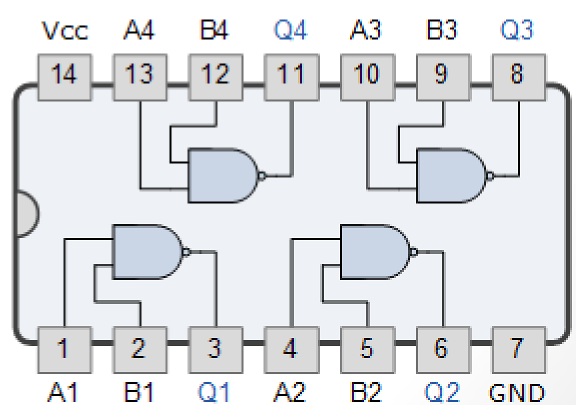

2-input "AND" gate plus a "NOT" gate




| B | A | OUT |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

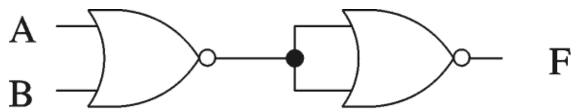



7400 Quad 2-input Logic NAND Gate

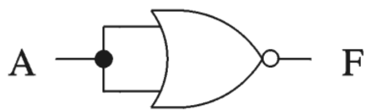


# NOR & NAND Universal Gate

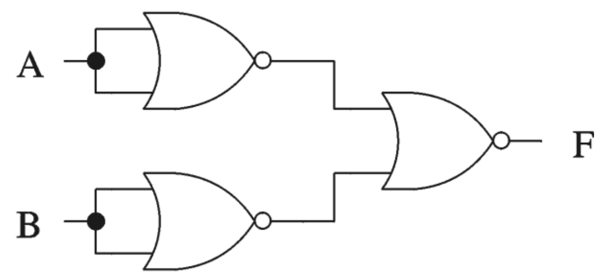
- Any gate can be constructed using them.



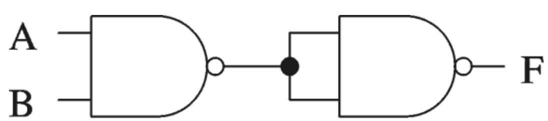

OR gate



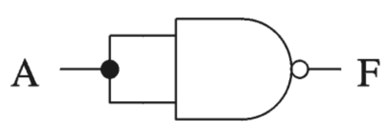

NOT gate



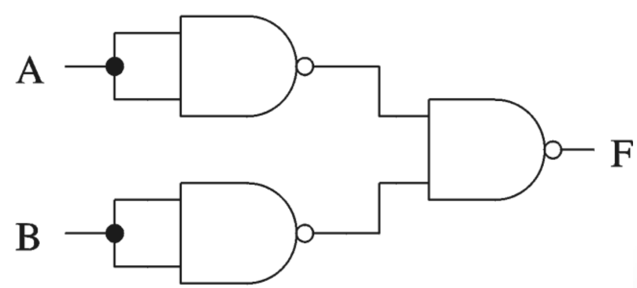

AND gate




AND gate



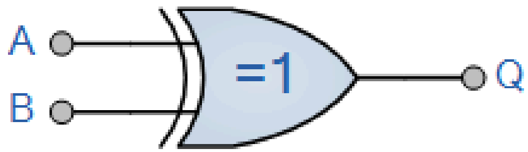

NOT gate




OR gate

# XOR and XNOR Gates

**Symbol**



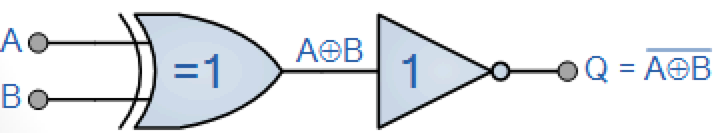

2-input Ex-OR Gate

**Truth Table**

| B | A | Q |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

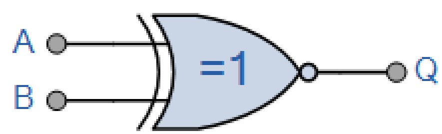
**Boolean Expression** $Q = A \oplus B$

**A OR B but NOT BOTH gives Q**




2-input "Ex-OR" gate plus a "NOT" gate

**Symbol**




2-input Ex-NOR Gate

**Truth Table**

| B | A | Q |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

**Boolean Expression** $Q = \overline{A \oplus B}$

**Read if A AND B the SAME gives Q**

# Truth Table Summary

| Inputs | | Truth Table Outputs For Each Gate | | | | | |
|---|---|---|---|---|---|---|---|
| A | B | AND | NAND | OR | NOR | EX-OR | EX-NOR |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |

| Truth Table Output for Single-input Gates | | |
|---|---|---|
| A | NOT | Buffer |
| 0 | 1 | 0 |
| 1 | 0 | 1 |

# EXAMPLES OF LOGIC CIRCUITS

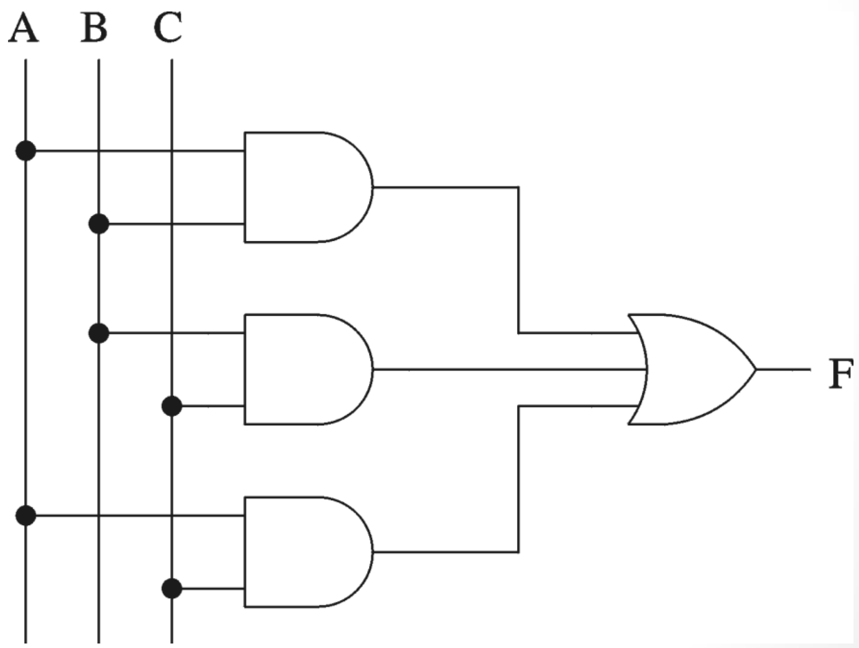
# Example of a Logic Function

3-input majority function

| A | B | C | F |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

- Logical expression form

$$F = A\,B + B\,C + A\,C$$

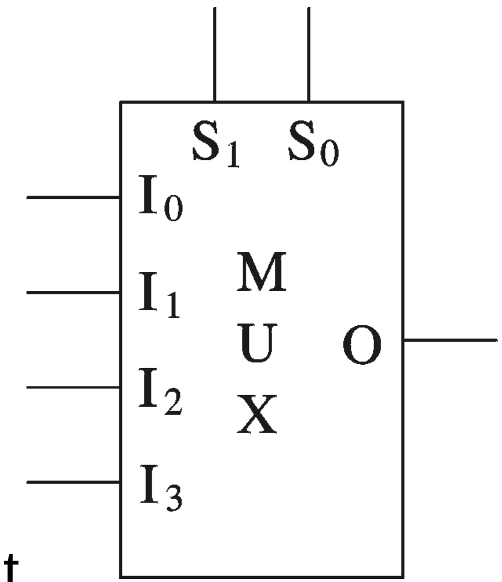


# Multiplexers

- An Example of Combinational Circuit
- Output depends only on the current inputs

4-data input MUX

- Multiplexer
  - $2^n$ data inputs
  - n selection inputs
  - a single output
- Selection input determines the input that should be connected to the output

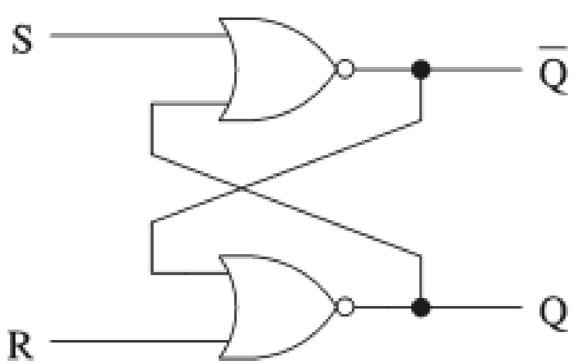



| $S_1$ | $S_0$ | O |
|---|---|---|
| 0 | 0 | $I_0$ |
| 0 | 1 | $I_1$ |
| 1 | 0 | $I_2$ |
| 1 | 1 | $I_3$ |

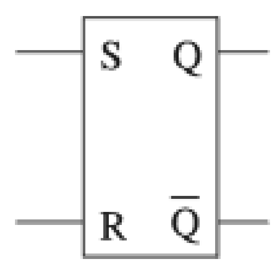
# SR Latch

- A Sequential example
  - Output depends on current as well as past inputs
- Can remember a bit
- Level-sensitive (not edge-sensitive)

A NOR gate implementation of SR latch




(a) Circuit diagram




(b) Logic symbol

| S | R | $Q_{n+1}$ |
|---|---|---|
| 0 | 0 | $Q_n$ |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| ~~1~~ | ~~1~~ | ~~0~~ |

(c) Truth table

- For more details, refer to:
  - **Digital Logic Tutorial**, found online at
    http://www.electronics-tutorials.ws/logic/logic_1.html
  - Other Digital Logic online Tutorials & wiki.
  - Reference books:
    - M. Mano, **Digital-Design**, 4th ed.
    - **Fundamentals of Computer Organization and Design**.
- The lecture is available online at:
  - *http://bu.edu.eg/staff/ahmad.elbanna-courses/12136*
- For inquires, send to:
  - ahmad.elbanna@feng.bu.edu.eg