Digital Logic

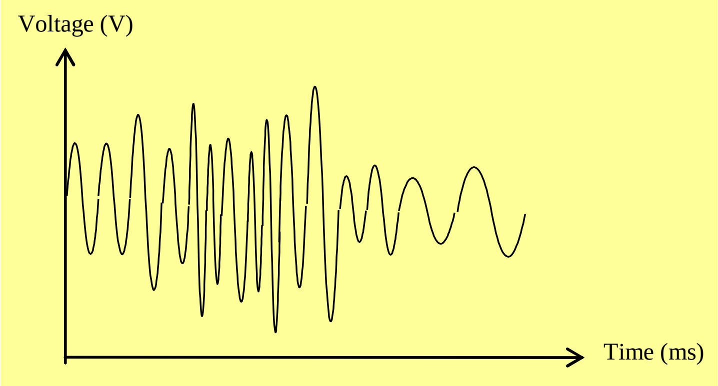
CO 2206 Computer Organization

Topics

- Analog vs Digital
- Digital Systems
- Logic Signal
- Integrated Circuit (IC)
- Logic Gates
- Boolean Algebra

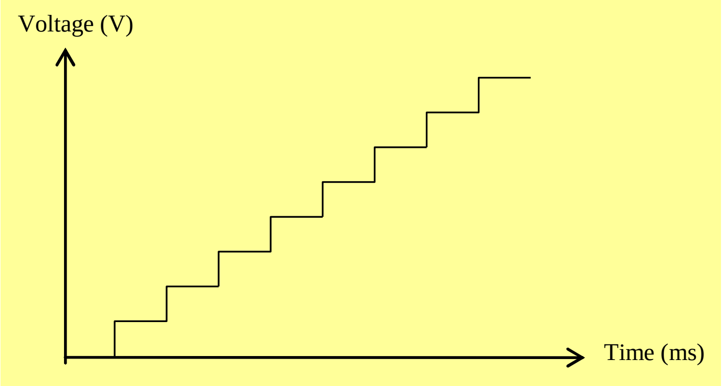
Our Nature is Analog

- Real world is mainly *analog*: size, weight, color, illumination, feeling, senses, voltage, current, etc
  - we do not have a finite number of values; there is always a value in between two: what's between 2.001 and 2.002?
- Properties of ***analog*** signals:
  - varies smoothly between two extremes
  - has continuous range of values, i.e., it has infinite number of possible levels
  - is time-continuous



Analog vs Digital

- Dealing with *analog* signals is demanding
  - infinite values to process (including compares)
  - how accurate can we get to?
  - how reliable (repetitive) can we get to?
- It will be easier to deal with signals having finite values
- ***Digital*** signal has the following properties:
  - has finite levels, i.e. discrete values
  - is time-discrete



Digital Systems

- Computer is a ***digital system***
- *Advantages* of digital system:
  - Digital systems are easier to design because exact level/value is not important
  - Digital systems are less affected by noise. Digital signal can be reconstructed. Digital signals are more reliable.
  - The operation of a digital system/network is programmable
- *Disadvantage* of digital system :
  - The real world is mainly analog. There is a need of analogue-to-digital conversion.

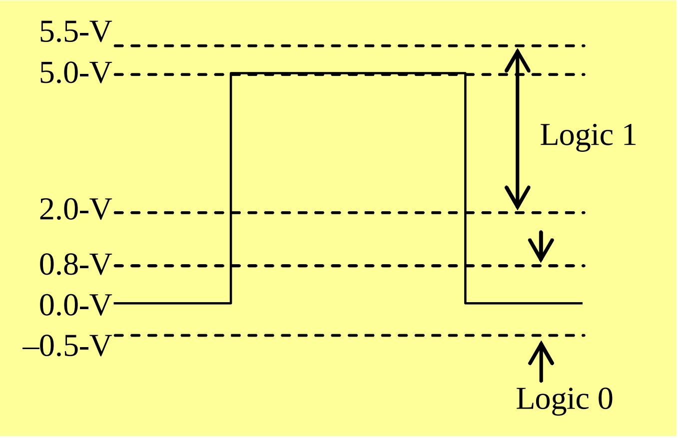
Digital Logic

- Most digital system uses digital signal with two levels — called ***logic level*** :
  - *Logic 1* = High level
  - *Logic 0* = Low level
- This is called a ***binary signal*** or ***logic signal***
  - *HIGH* and *LOW* are referred to as *Logic States*
- ***Positive Logic*** - *logic 1* has a voltage level more positive than that of *logic 0*
- ***Negative Logic*** - *logic 0* has a voltage more positive than that of *logic 1*
- We use *positive logic* throughout this course. It is most widely used.

Digital Electronic
(Digital Logic Circuits)

- *Electronic circuits* are analog in nature
- ***Digital Logic Circuits*** are electronic circuits designed to work on digital logic signals (voltage levels)
  - Computers are mainly build from logic circuits
- Usually, a band of voltage is used to represent each *logic level*. Commonly used logic signals are:
  - *TTL logic level* (for circuits using TTL technology):
    - Logic 1 = +2.0V — +5.5V
    - Logic 0 = –0.5V — +0.8V
  - *CMOS logic level* (for circuits using CMOS technology):

    (at supply voltage of +5.0V):
    - Logic 1 = +3.5V — +5.5V
    - Logic 0 = –0.5V — +1.5V



Integrated Circuit (IC) - 1

- *Digital logic circuit*, including *logic gates* are build from basic electronic components like resistors, capacitors, diodes and transistors
- Digital ***Integrated-Circuits (ICs)*** are a collection of resistors, diodes and transistors fabricated on a single piece of *semiconductor* material (usually *Silicon*) called a *Substrate*, which is commonly referred to as a "*chip*"
- *Logic gates* and *logic systems* usually come in *IC* forms

Integrated Circuit (IC) - 2

- *Advantages* of *IC*:
  - *IC* packs a lot more circuitry than discrete-component-circuit in a small package, so that the overall *size* of almost any digital system is reduced
  - *IC* is *cheaper* to produce, than discrete-component-circuit, in large volume
  - *IC* is more *reliable* than discrete-component-circuit because it reduces the number of external interconnections (soldered connections) from one device to another
  - *IC* has *reduced power consumption* and so it will reduce power supply costs and requirement for cooling

Levels of Integration

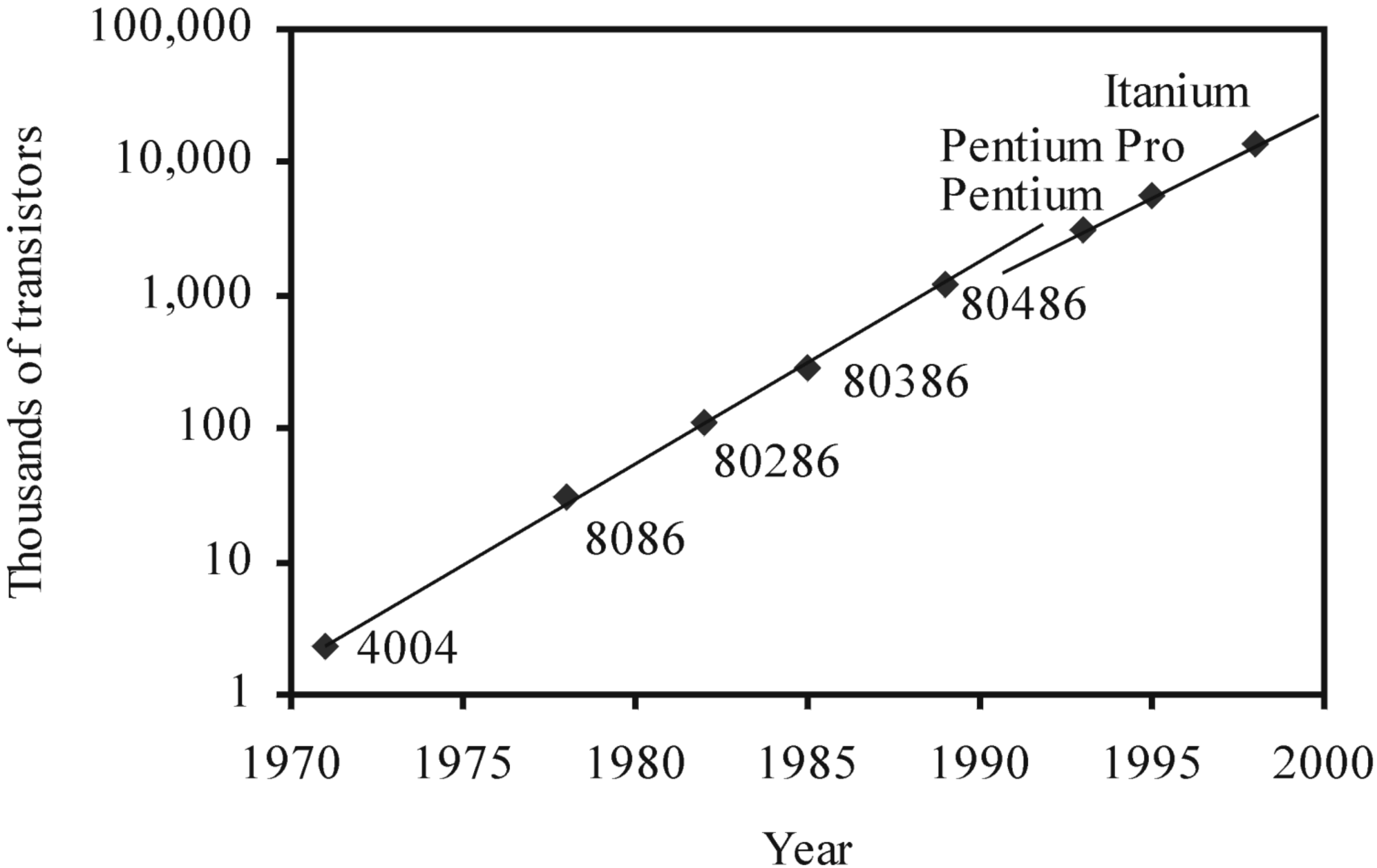
SSI	Small-scale	10+ transistors	Basic gates, flip-flops
MSI	Medium-scale	100+	Decoder, register, counter
LSI	Large-scale	10,000+	Small memory, microprocessor
VLSI	Very large-scale	100,000+	microprocessor, memory
ULSI	Ultra large-scale	1,000,000+	latest processors

- Being the most important component in *IC*, ***transistors*** are *electronic switches*, responsible for generating the *logic levels* (*ON* or *OFF*)
- *Intel 8086* has about *29k* transistors; *Pentium >3M* transistors

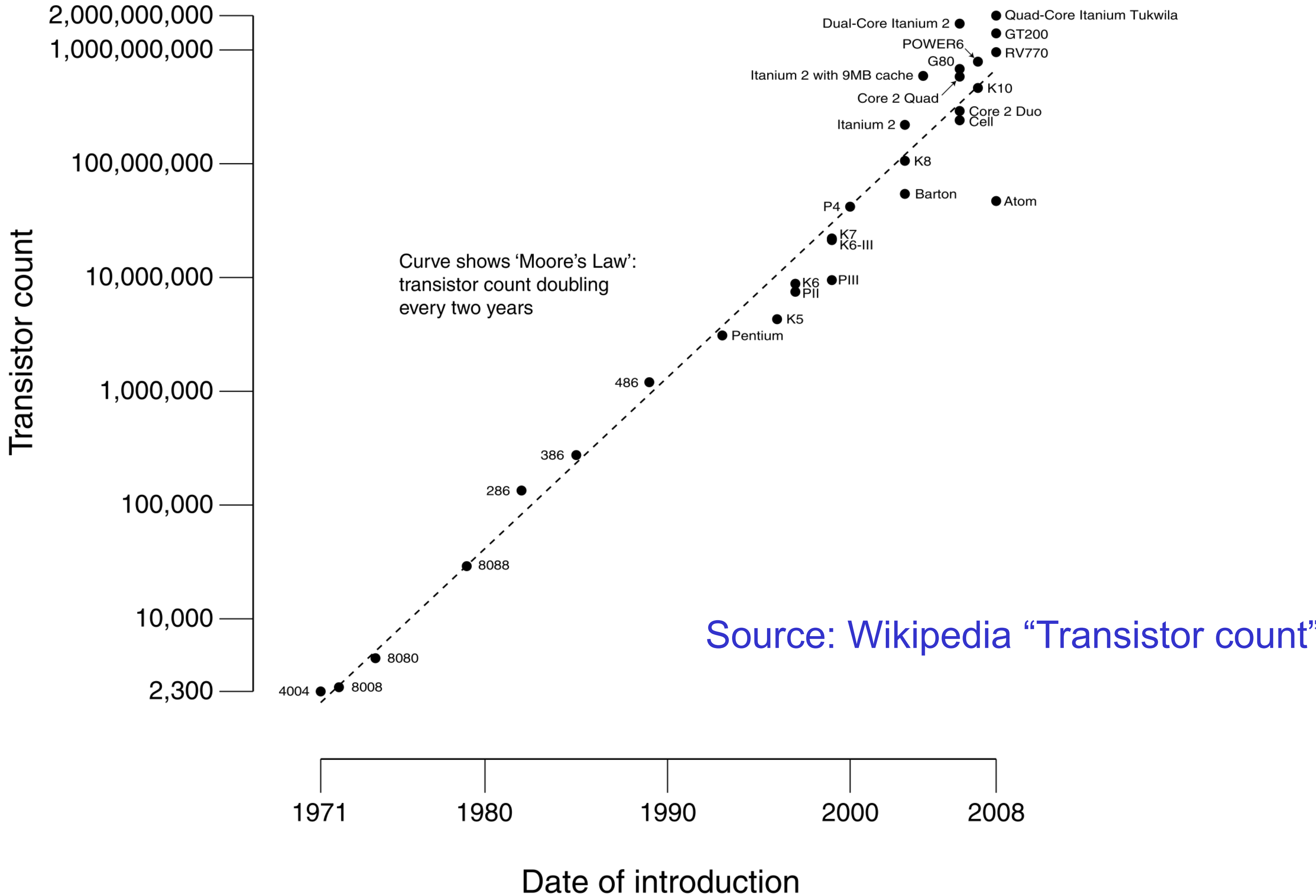
Moore's Law

- Gordon Moore observed in 1965 that the no. of transistors on a chip was doubling every year
- Development slowed down since 1970s
  - Until 1990s, transistor count doubled every 18-24 months
  - In 1990s, doubling about every 2.5 years
- Cost of a chip has remained almost unchanged

Growth in CPU transistors count



CPU Transistor Counts 1971-2008 & Moore's Law



Logic Gates

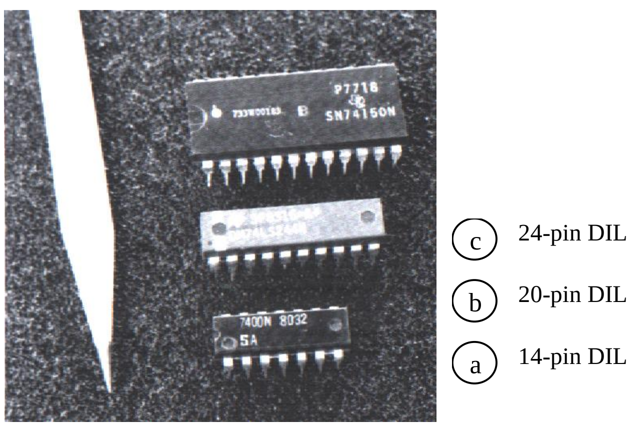
- ***Logic Gates*** are basic building blocks for *digital logic circuits*
  - *Logic gate* is a circuit that may have a number of inputs but has only one output that will be either logic 1 or logic 0 as determined by the condition of the inputs
- Two types of *representation* for *logic gates*:
  - *International Symbol* (MIL, ANSI)
  - *British Standards Institute Symbol* (BSI)
- ANSI is more commonly used compared to BSI
  - ANSI representation will be used throughout the course

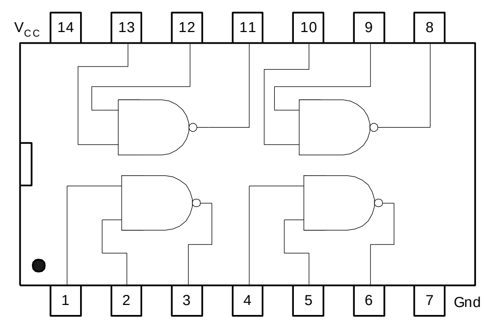
Logic Families

- There are different technologies used to fabricate digital *ICs*. They are referred to as ***logic families***. Popular logic families are:
  - *Transistor-transistor logic* (*TTL*)
  - *Complementary metal oxide semiconductor* (*CMOS*)
  - *Emitter coupled transistor logic* (*ECL*)
- Each *logic family* has a different set of characteristics. For example:
  - *ECL* is the fastest
  - *CMOS* has low power consumption but slow
  - *TTL* is a compromise between the two

Logic Gate ICs

- Logic gates come in multiple form in commercial *ICs*, i.e. an *IC* contains more than one gate.
  - e.g. *OR* gate can come in the form of *Quad 2-input OR*, *Dual 4-input OR* or *Triple 3-input OR*.
- Digital *ICs* normally come in the form *of 14 pins* *Dual-In-Line (DIL)* package or *16 pins DIL ICs*.

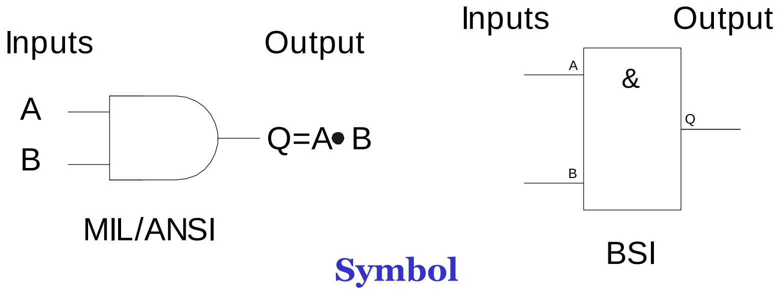




TTL NAND gates ICs

Basic Logic Gates - 1

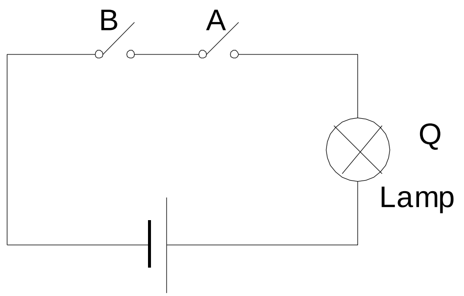
- Basic Logic operations:
  - AND, OR, NOT, Exclusive OR (XOR)
- **AND**



Symbol

Truth Table

Inputs		*Output*
A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1



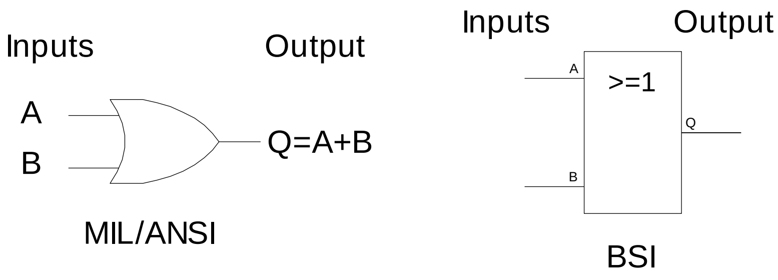
Illustration

Basic Logic Gates - 2

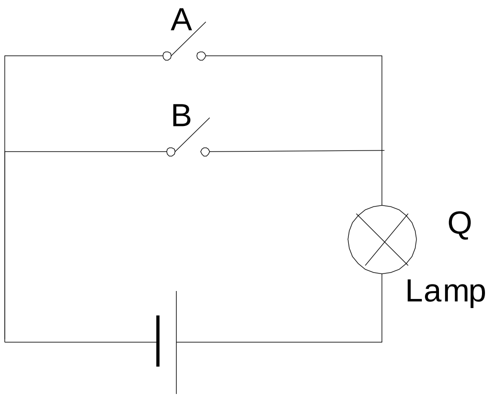
- **OR**

Truth Table

Inputs		*Output*
A	B	Q
0	0	0
0	1	1
1	0	1
1	1	1



Symbol



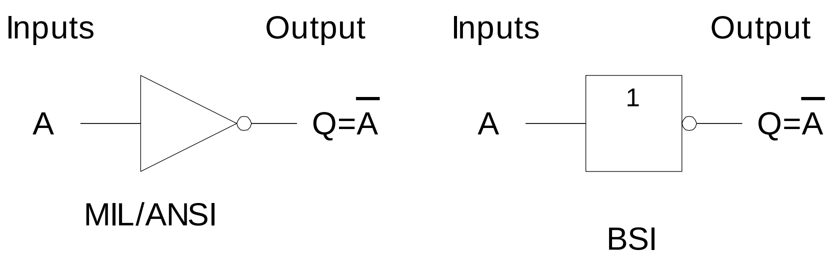
Illustration

Basic Logic Gates - 3

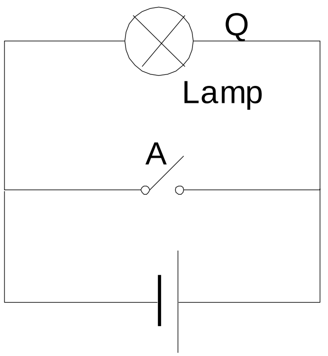
- **NOT**

Truth Table

Input A	Output Q
0	1
1	0



Symbol



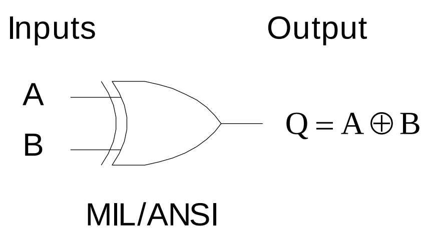
Illustration

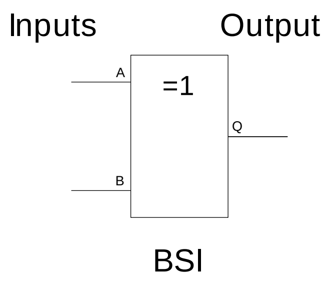
Basic Logic Gates - 4

- **XOR**

Truth Table

Inputs		*Output*
A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0

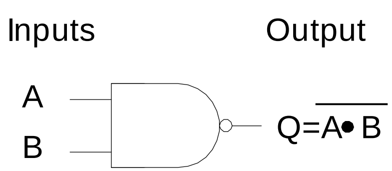




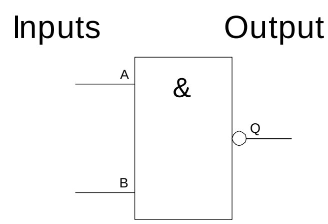
Symbol

Derived Logic Gates

- Derived Logic operations:
  - NAND (Not-AND)
  - NOR (Not-OR)
- **NAND**



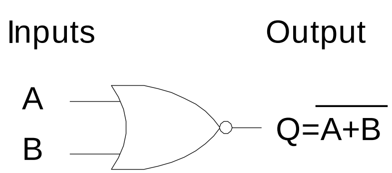
MIL/ANSI



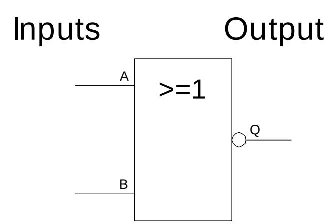
BSI

Inputs		*Output*
A	B	Q
0	0	1
0	1	1
1	0	1
1	1	0

- **NOR**



MIL/ANSI

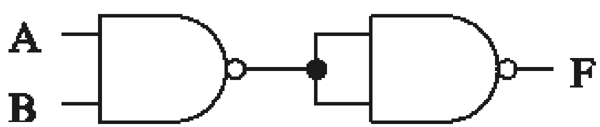


BSI

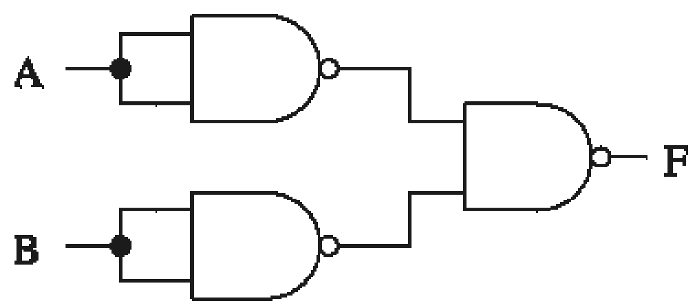
Inputs		*Output*
A	B	Q
0	0	1
0	1	0
1	0	0
1	1	0

Universal Gates - 1

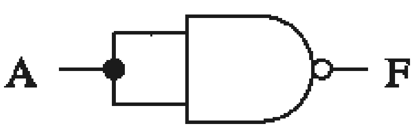
- *NAND* and *NOR* gates are called ***universal gates*** because any logical function can be implemented using only *NAND* or *NOR* gates
- Implementations of *AND*, *OR* and *NOT* gates using only *NAND*:



AND gate



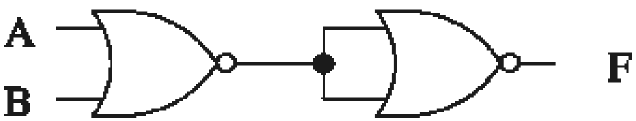
OR gate



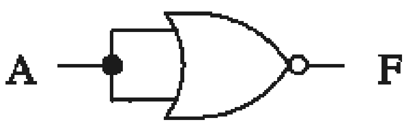
NOT gate

Universal Gates - 2

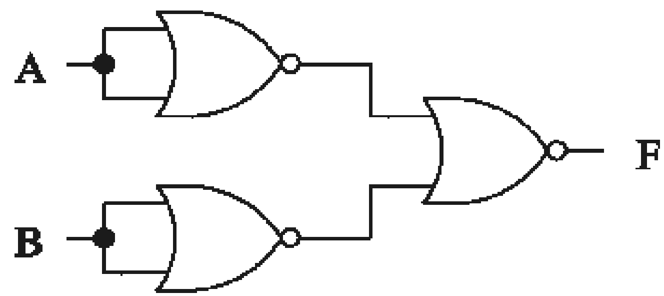
- Implementation of *AND*, *OR* and *NOT* gates using only *NOR*



OR gate



NOT gate



AND gate

Boolean Algebra

- Analysis of logic networks is greatly aided by the logical algebra developed in the last century by *George Boole*, an English mathematician.
- The ***theorems of Boolean algebra*** are used to simplify digital logic networks in much the same fashion that mathematical algebra is used to manipulate ordinary algebraic expressions.
  - the variables in *Boolean expressions* can assume only one of two possible values

Boolean Notation

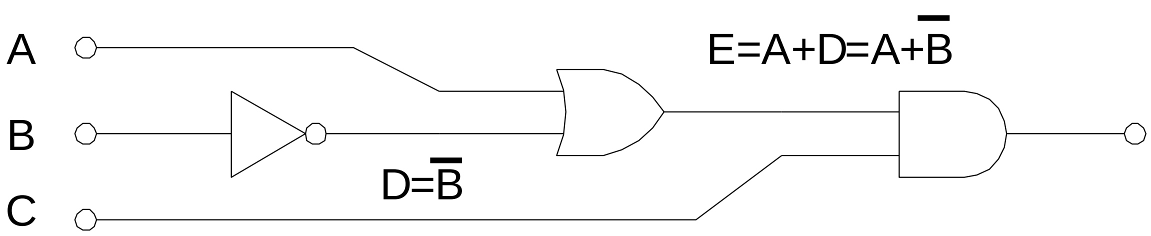
- ***Boolean notation*** is an "arithmetic" symbol used to represent the logic operations *AND*, *OR*, *NOT* and *XOR*
  - *the "•" (AND) notation is sometimes omitted, for example, Q=A•B is sometimes written as Q=AB*

Logic Operation	Boolean Notation
AND	•
OR	+
NOT	bar over the input variable, e.g. $\overline{A}$
XOR	⊕

Describing Logic Circuit

- A logic circuit can be described in *three ways*:
  - *circuit schematic*
  - *truth table*
  - *Boolean expression*

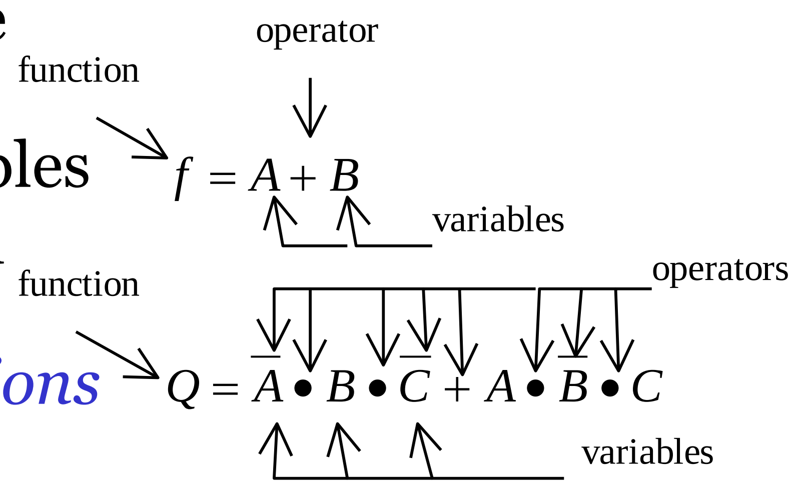
Inputs			*Output*
A	B	C	Q
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1



$Q = C \bullet E = C \bullet (A+B) = C(A+\overline{B})$

Boolean Expression
(aka Logical Expression)

- ***Boolean expression*** is an alternative way of describing the behaviour of a logic gate or a logic network besides the *truth table*
- A *Boolean expression* has three parts :
  - ***Function*** - *output* which we want to express
  - ***Variables*** - the *input* variables
    - *literals* are the variables and their inverted form
  - ***Operators*** - *Boolean notations* used to express the logic operations on the inputs



Boolean Algebra Laws – 1

(Theorems of Boolean Algebra)

law	***and* version**	***or* version**
Identity	$x \cdot 1 = x$	$x + 0 = x$
Complement	$x \cdot x' = 0$	$x + x' = 1$
Commutative	$x \cdot y = y \cdot x$	$x + y = y + x$
Idempotent	$x \cdot x = x$	$x + x = x$
Null	$x \cdot 0 = 0$	$x + 1 = 1$

Boolean Algebra Laws – 2

(Theorems of Boolean Algebra)

law	***and*** **version**	***or*** **version**
Distribution	$x \cdot (y+z) = (x \cdot y)+(x \cdot z)$	$x+(y \cdot z) = (x+y) \cdot (x+z)$
Involution	$(x')' = x$	-
Absorption	$x \cdot (x+y) = x$	$x+(x \cdot y) = x$
Associative	$x \cdot (y \cdot z) = (x \cdot y) \cdot z$	$x+(y+z) = (x+y)+z$
De Morgan's	$(x \cdot y)' = x'+y'$	$(x+y)' = x' \cdot y'$

XOR Identities

XOR: $x \oplus y = xy' + x'y$ **XNOR**: $(x \oplus y)' = xy + x'y$

Basic theorems

T1. $x \oplus x = 0$ T2. $x \oplus x' = 1$ T3. $x \oplus 0 = x$ T4. $x \oplus 1 = x'$

Inversion theorems

T5. $(x \oplus y)' = x' \oplus y = x \oplus y'$

T6. $x' \oplus y' = x \oplus y$

T7. $x \oplus y = y \oplus x$ **Commutative law**

T8. $(x \oplus y) \oplus z = x \oplus (y \oplus z)$ **Associative law**

T9. $x(y \oplus z) = xy \oplus xz$ **Distributive law**

T9'. $x(y \oplus z) = (x'+y) \oplus (x'+z)$ **Distributive law with OR function**

T10. If: $f = g \oplus h$ and $gh = 0$, then $f = g + h$ **Disjunction theorem**

T11. If: $f = g \oplus h$, then $g = f \oplus h$ and $h = g \oplus f$ **Transposition theorem**

Duality Principle

- ***De Morgan's Duality Law*** is useful in coming up with *NAND* or *NOR* based design
- ***Duality*** allows us to transform a law from *AND* version to *OR* version by replacing each 1 with 0, 0 with 1, + with ·, and · with +
  - *Duality* principle states that a *Boolean* equation remains valid if we take the dual of the expressions on both sides of the equal sign.
  - Complement of a function can be derived by taking the dual and complement each literal

A = BC'D
A' = B'+C+D'

ABC = (A'+B'+C')'

A+B'+C = (A'BC')'

Logical Equivalence

- If 2 logical circuits are performing the same logical function, they are ***equivalent***
- Establishing equivalence is important because it allows us to pick an efficient design for *implementation*
  - efficient here means using less number of gates
- Equivalence can be proved by *truth table* or by *Boolean algebra rules*
  - proving equivalence can be from left to right, or right to left, whichever is more convenient

Summary

- Digital signals are easier and more reliable to deal with
- Logic signal is the most common digital signal
- Computer deals with digital logic signals and is a digital systems
- Basic building blocks of digital systems are the logic gates
- Logic gates are electronic circuits designed to deal with digital signals
- Logic gates are available in IC forms
- Boolean Algebra is a useful mathematical expression for describing logic functions, as well as manipulating the functions