Introduction to Digital Logic

- Digital systems use binary numbers
  - Decimal numbers are base 10 (digits 0, 1, … 9)
    - Example: $893 = 8 \times 10^2 + 9 \times 10^1 + 3 \times 10^0$
  - Binary numbers are base 2 (digits 0, 1)
  - Example: binary $1101 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 13$
  - Example: write decimal 23 in binary

$$
\begin{aligned}
23 &= 16 + 4 + 2 + 1 \\
&= 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 \\
&= 10111
\end{aligned}
$$

Binary numbers represent information: numbers, text, colors, etc.

Numbers

Binary	***Decimal***
“000”	0
“001”	1
“010”	2
“011”	3
“100”	4
“101”	5
“110”	6
“111”	7

Text

Binary	***Character***
“0100 0000”	@
“0100 0001”	A
“0100 0010”	B
“0100 0011”	C
“0100 0100”	D
“0100 0101”	E
“0100 0110”	F
“0100 0111”	G

Colors

Binary	***Color***
“0 0 0”	Black
“1 0 0”	Red
“0 1 0”	Green
“0 0 1”	Blue
“0 1 1”	Cyan
“1 0 1”	Magenta
“1 1 0”	Yellow
“1 1 1”	White

Where do we use digital logic in electronics?

- Transistors act as switches
  - Apply high voltage → transistor conducts
  - Apply low voltage → transistor doesn’t conduct
  - Can switch very fast
- Digital or *Boolean* logic describes everything in 1’s and 0’s
  - 1 = high voltage, say > 4 V, means “True” or “On”
  - 0 = low voltage, say < 1 V, means “False” or “Off”
- Logic operations such as NOT, OR, AND act on these logic variables and are easily implemented in transistor circuits called **Logic Gates**
- Logic operations are represented by **Truth Tables** which define every possible combination of inputs

Logic Operations: NOT

- Example: A safety light is normally always on, unless you push a button to turn it off.
- Input: 1 = push button, 0 = don't push button
- Output: 1 = light on, 0 = light off
- Truth table:

button	light
0	1
1	0

- Called **NOT** operation, output = opposite of input

Logic Operations: OR

- Example: 2-door car, light goes on if either or both doors are open
- Inputs: 1 = door open, 0 = door closed
- Output: 1 = light on, 0 = light off
- Truth table:

door 1	door 2	light
0	0	0
0	1	1
1	0	1
1	1	1

- Called **OR** operation, output = 1 if either or both inputs = 1

Logic Operations: AND

- Example: passenger car window with driver master switch, window only opens if both switches on
- Inputs: 1 = switch on, 0 = switch off
- Output: 1 = window operates, 0 = doesn't operate
- Truth table:

switch 1	switch 2	window
0	0	0
0	1	0
1	0	0
1	1	1

- Called **AND** operation, output = 1 only if both inputs = 1

- Can have more inputs and more complicated logic
- Example: Joe drives instead of bikes to school if its cold and/or raining, and he has an early class
  - Inputs: A = 1 cold B = 1 raining C = 1 early class
    0 warm 0 dry 0 no early class
  - Output: 1 = drives, 0 = bikes

Truth table:

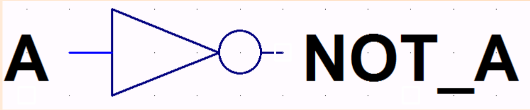
A	B	C	Out
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Out = (A OR B) AND C

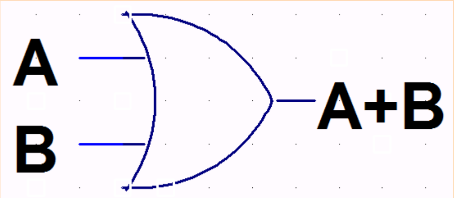
Math Symbols and Circuit (Logic Gate) Symbols

- Math symbols: NOT = $\overline{A}$, OR = A + B, AND = A·B (or AB)
- Circuits that perform these operations called **Logic Gates**

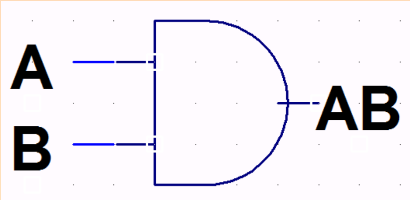
NOT:



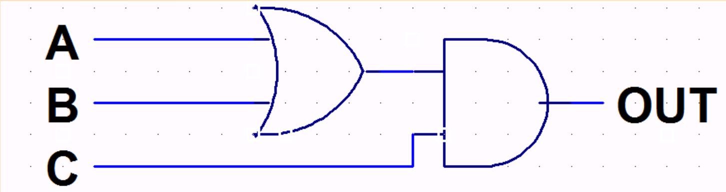
OR:



AND:



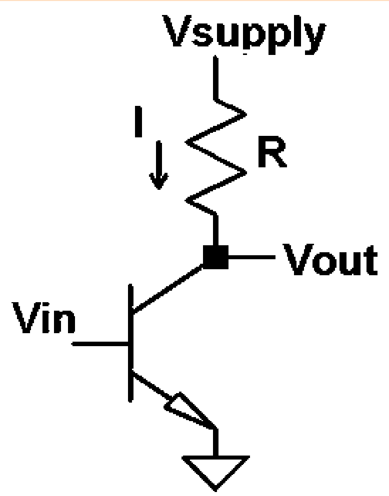
- Joe example: Out = (A + B) · C



Transistors behave as inverters

- V_{in} = Low, no I flows, V_{out} = High (= V_{supply})
- V_{in} = High, large I flows, V_{out} = Low (large voltage dropped across R)
- V_{out} = NOT V_{in}

in	out
1	0
0	1

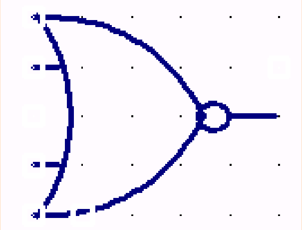


- More convenient to build circuits which use *inverse* of OR, AND – called **NOR, NAND**

NOR and NAND Gates

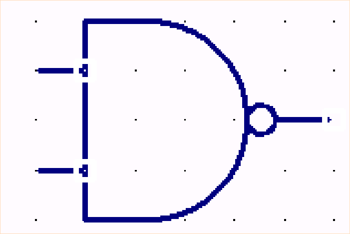
- OR + NOT = NOR: output low if any input high

A	B	A+B	$\overline{A+B}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0



- AND + NOT = NAND: output low if all inputs high; demo on logic.ly

A	B	AB	$\overline{AB}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0



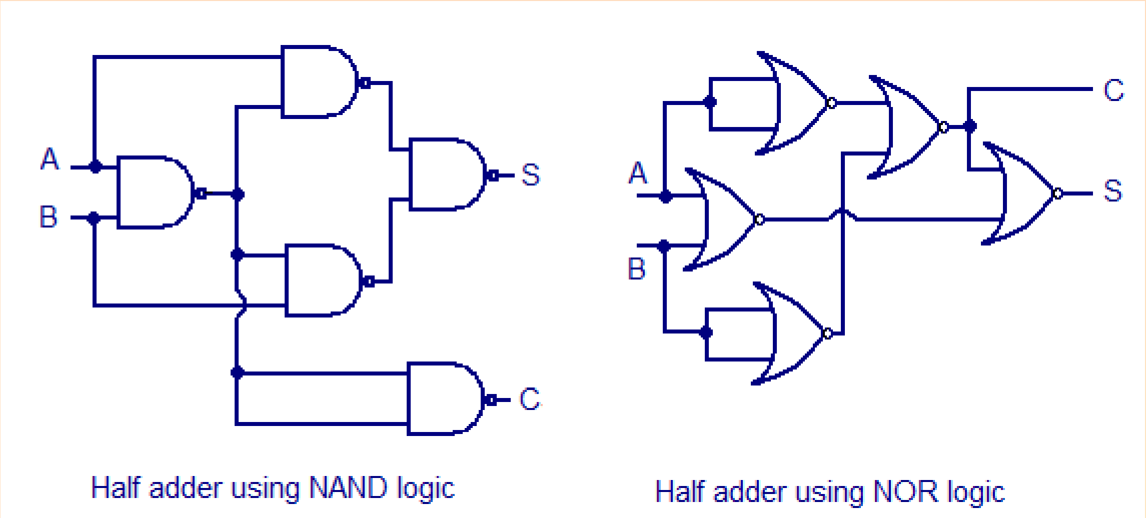
Simulators

- Logic.ly
- Examples of lab circuit – do it in class

More circuits

- Logic.ly
- Example of half-adder circuit – use logic.ly to construct

Inputs		Outputs	
A	*B*	*C*	*S*
0	0	0	0
1	0	0	1
0	1	0	1
1	1	1	0



More circuits