

**L03-CS8421-08-27-08**

# Architecture Overview

## CS8421

### Computing Systems

### Dr. Ken Hoganson

*Class*

*Will*

*Start*

*Momentarily...*



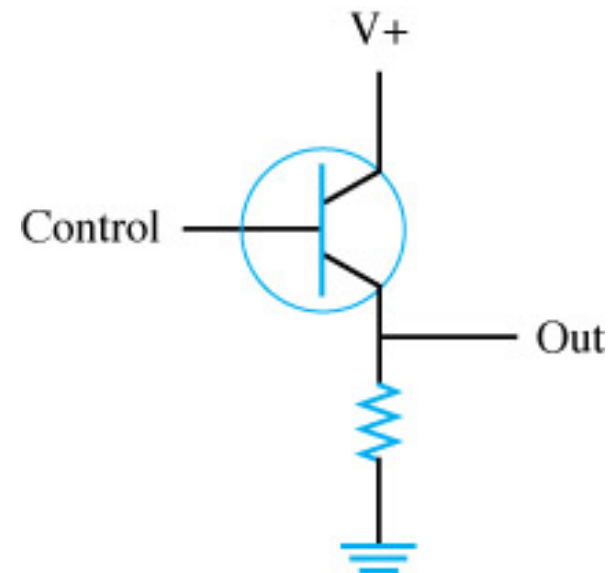
## Layers Discussed

- Layer 0 – transistors
- Layer 1 – Logic gates
  - AND/OR
  - NAND/NOR
  - Complete Set of Operations

# Transistor

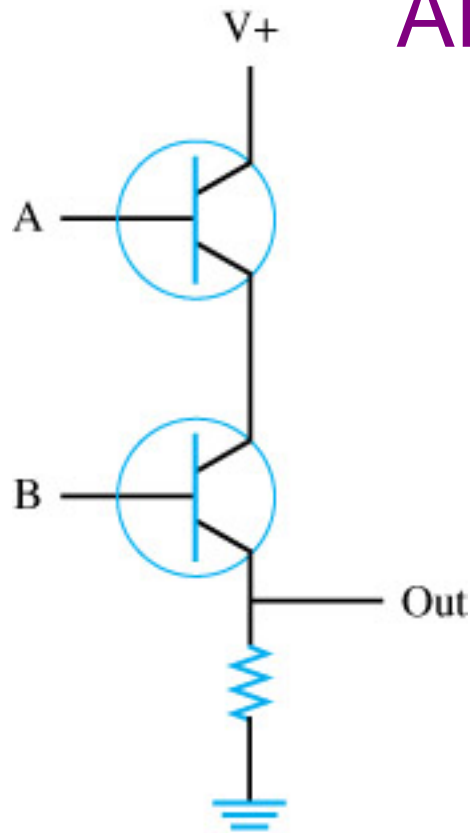
Transistor used as a switch.

The transistor in other applications, is also used as an amplifier or detector.

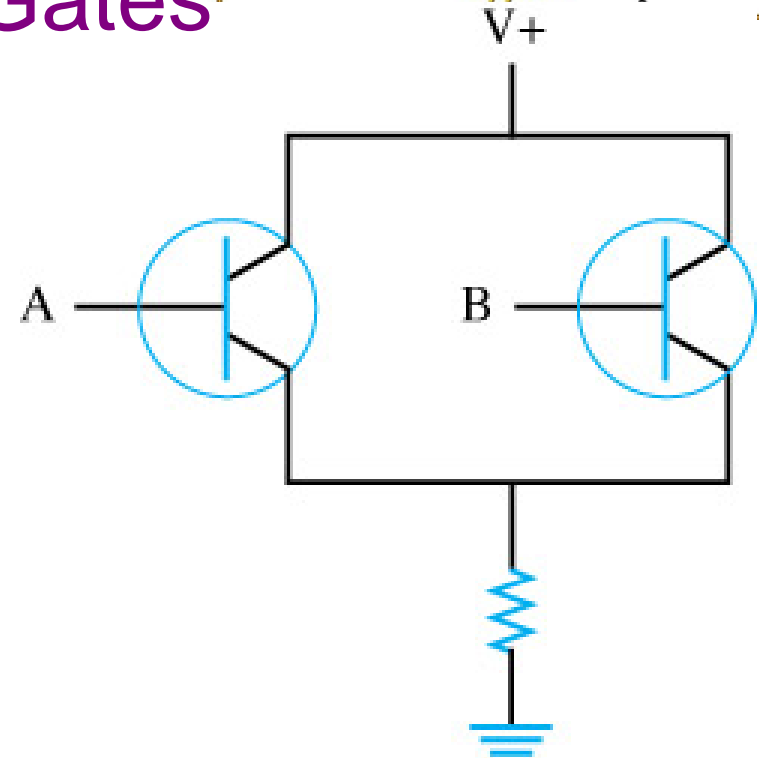


Control	Out
0	0
1	1

# AND & OR Gates



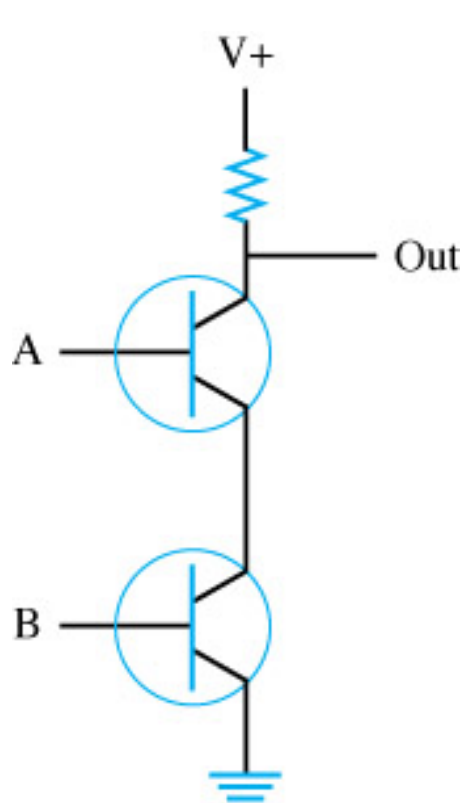
A	B	Out
0	0	0
0	1	0
1	0	0
1	1	1



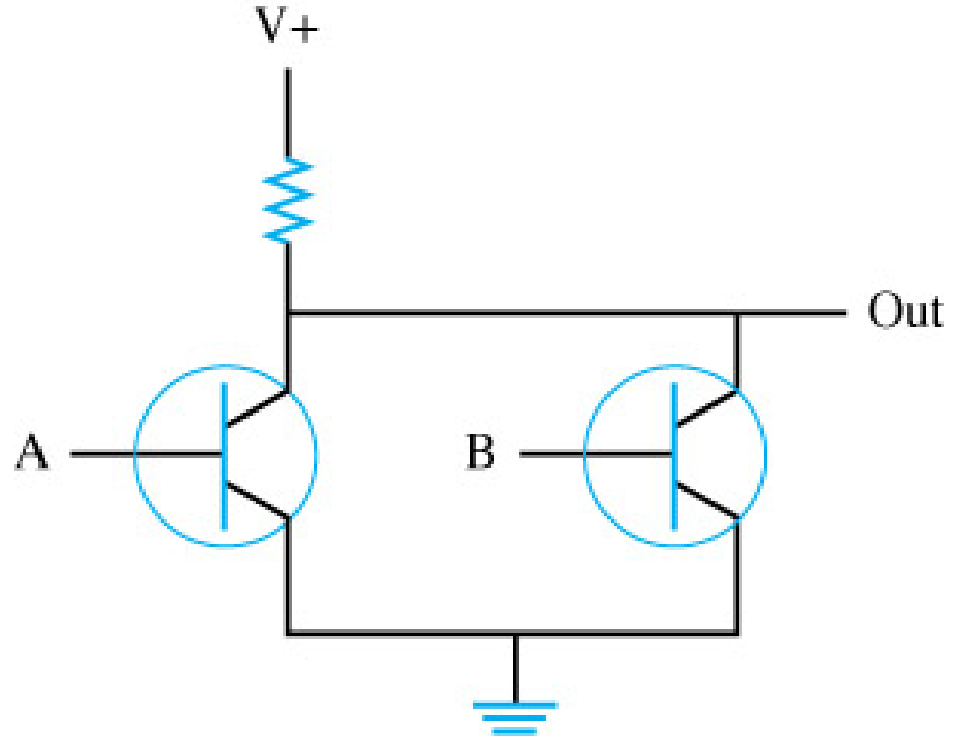
A	B	Out
0	0	0
0	1	1
1	0	1
1	1	1



# NAND & NOR Gates



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

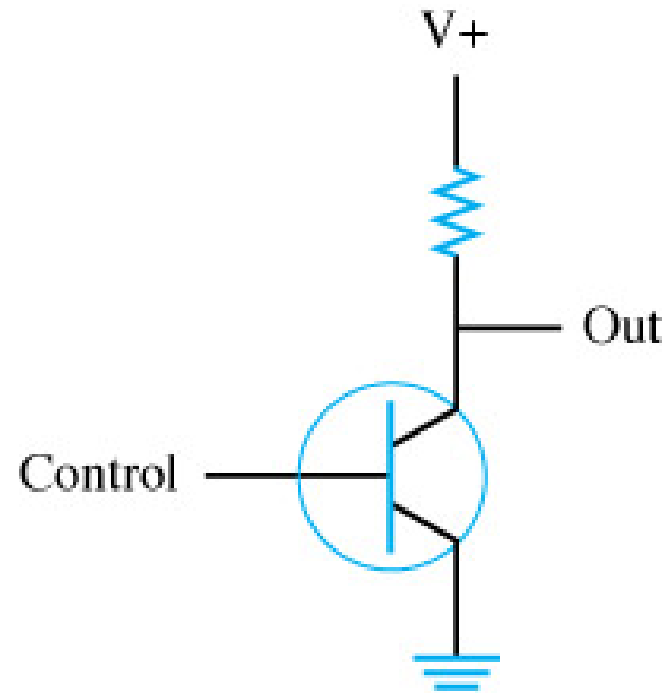


A	B	Out
0	0	1
0	1	0
1	0	0
1	1	0








# Transistor as an Inverter

Transistor used to reverse or complement its input.



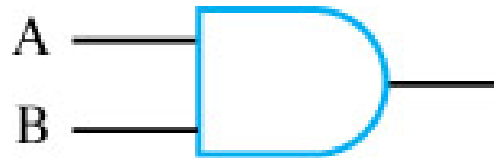
Control	Out
0	1
1	0

Name	Symbol	Function	Truth Table															
AND		$F = AB$ or $F = A \cdot B$	<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>F</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	A	B	F	0	0	0	0	1	0	1	0	0	1	1	1
A	B	F																
0	0	0																
0	1	0																
1	0	0																
1	1	1																
OR		$F = A + B$	<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>F</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	A	B	F	0	0	0	0	1	0	1	0	0	1	1	1
A	B	F																
0	0	0																
0	1	0																
1	0	0																
1	1	1																
NOT		$F = \bar{A}$	<table border="1"> <thead> <tr> <th>B</th> <th>F</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> </tr> </tbody> </table>	B	F	0	1	1	0									
B	F																	
0	1																	
1	0																	
NAND		$F = \overline{AB}$	<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>F</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </tbody> </table>	A	B	F	0	0	1	0	1	1	1	0	1	1	1	0
A	B	F																
0	0	1																
0	1	1																
1	0	1																
1	1	0																
NOR		$F = \overline{A + B}$	<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>F</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </tbody> </table>	A	B	F	0	0	1	0	1	0	1	0	0	1	1	0
A	B	F																
0	0	1																
0	1	0																
1	0	0																
1	1	0																

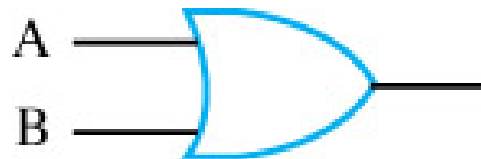


- We can use transistors to build AND, OR, NAND, NOR, and Invertors
- Manufacturing is simplified with NAND/NOR
- NAND/NOR avoid problem with “transistor bleed-through”
- Converting between AND/OR circuits and NAND/NOR circuits is easy with Boolean Algebra.
- (named for mathematician George Boole)
- Boolean Variable:
  - can be in just two states: 0 or 1
- Represent Boolean Variables with letters

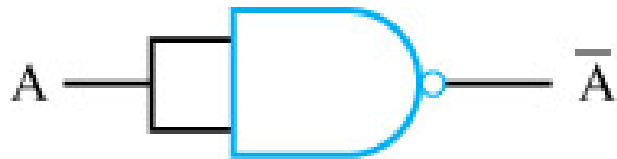
- Two inputs (Boolean variables) A and B are ANDed together.
- The AND operation is represented in a Boolean equation with the multiplication symbol (overloading of operator!)
- (The OR operation is represented with the addition symbol +)
- A AND B can be represented as  $A*B$  (usually AB)



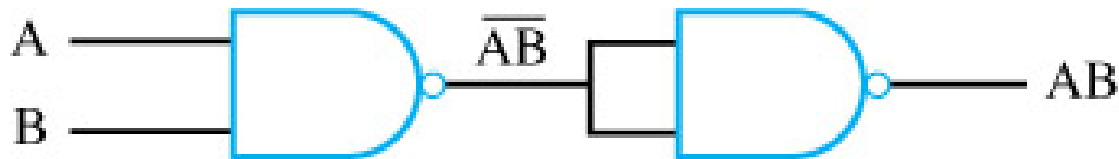
- A OR B can be represented as  $A + B$



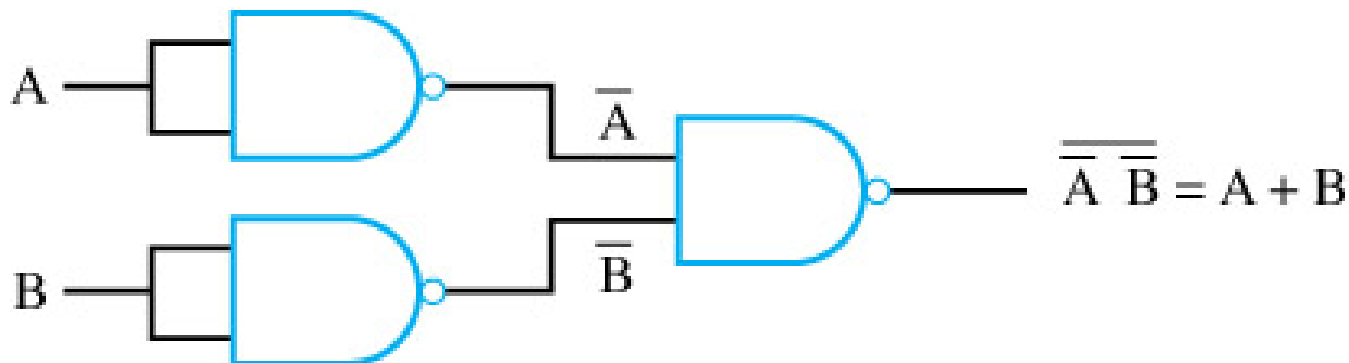
# NAND as a Complete Set of Operations



Inverter

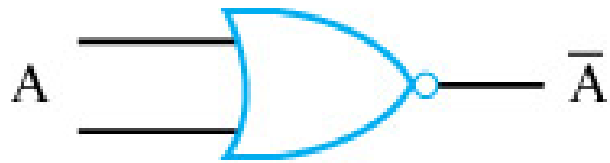


AND

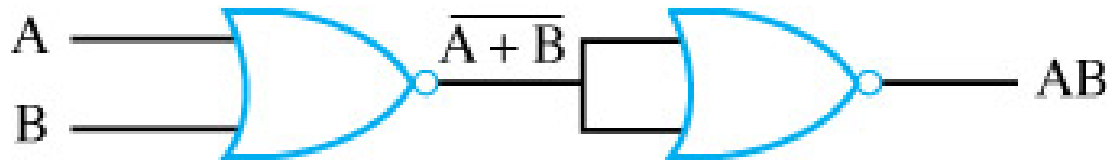


OR

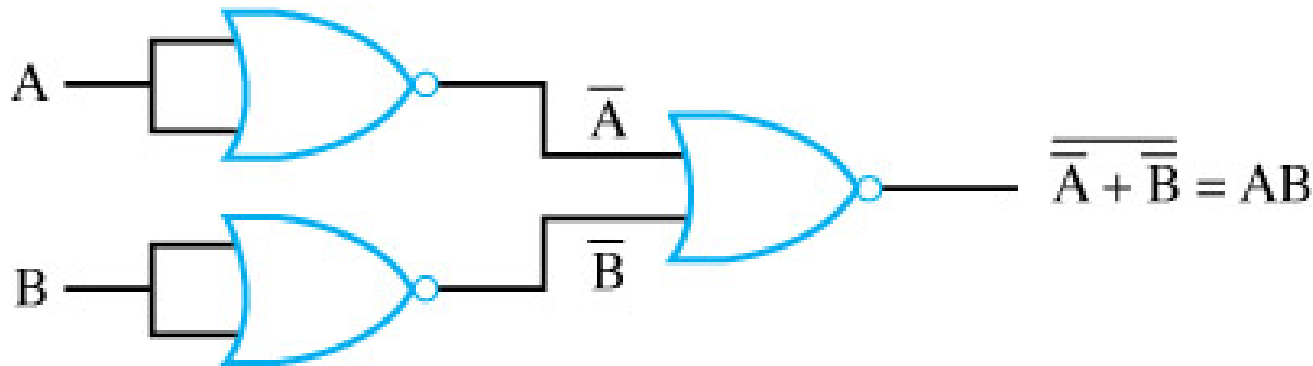
# NOR as a complete set of operations



Invertor



OR



AND

- One of the most useful principles in boolean algebra is **De'Morgan's Theorem.**
- It allows one to switch between ANDs and NORs and OR and NAND
- NOT terms or Inverted terms are represented with a line over the terms

$$\overline{AB} = \overline{A} + \overline{B}$$

$$\overline{A + B} = \overline{A} \overline{B}$$

A	B	A OR B	A'	B'	A' AND B'	A' NAND B'
0	0	0	1	1	1	0
0	1	1	1	0	0	1
1	0	1	0	1	0	1
1	1	1	0	0	0	1

In this notation, A' is NOT A, the same as  $\bar{A}$

# De'Morgan's

To convert  $A + B$  into a form that can be implemented using a NAND gate:

$$A + B = \overline{\overline{AB}} = \overline{\overline{A} \overline{B}}$$

To convert  $AB$  into a form that can be implemented using a NOR gate:

$$AB = \overline{\overline{AB}} = \overline{\overline{A} + \overline{B}}$$



# End of Lecture

**End  
Of  
Today's  
Lecture.**

This slide intentionally left blank

