Slide Set 2

for ENEL 353 Fall 2019

Steve Norman, PhD, PEng

Electrical & Computer Engineering
Schulich School of Engineering
University of Calgary

Fall Term, 2019

Contents

Outline of Slide Set 2

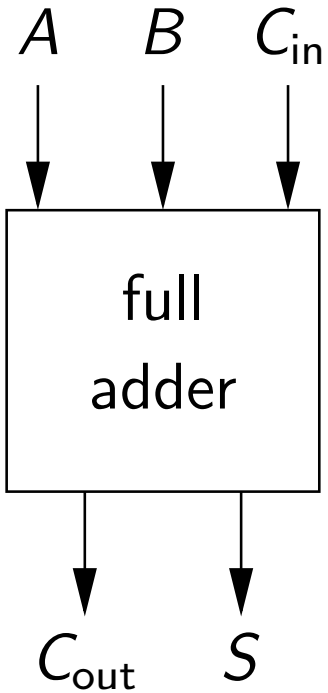
Combinational versus *Sequential* Logic

It's IMPORTANT to know what these words mean.

The outputs of a *combinational* logic circuit depend only the **current** values of its inputs.

The outputs of a *sequential* logic circuit depend on the **history** of its input values.

Example combinational circuit: 1-bit full adder



The values of C_{out} and S depend on what the values of A, B, and C_{in} are RIGHT NOW, not what they were a few minutes ago, or even a few nanoseconds ago.

(The sentence above describes **ideal** combinational logic. In **real, physical** combinational circuits, there will be tiny time delays between changes in input values and changes in output values.)

Outline of Slide Set 2

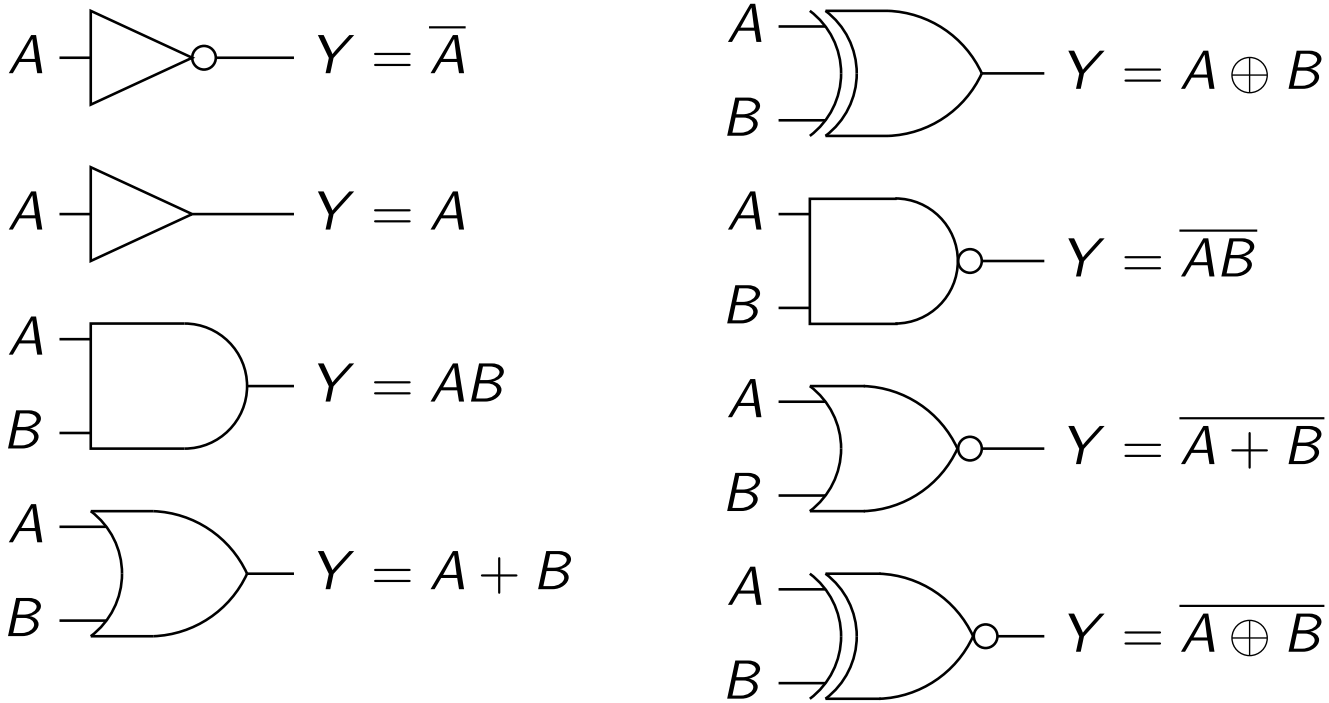
What does the term *logic gate* mean?

A *logic gate* is a **combinational logic circuit** that has

- **one or more inputs**, each of which is a **bit**;
- **exactly one output**, which is a **bit**.

According to this definition, is the full adder a logic gate?

Graphical and algebraic notations for some common 1- and 2-input logic gates

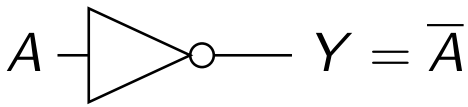


We'll now examine these gates one or two at a time ...

Outline of Slide Set 2

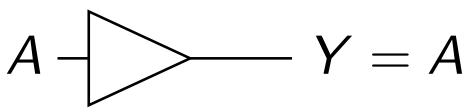
Basic gates: We'll start with 1-input gates

A NOT gate, also called an *inverter*:



In A is the input and Y is the output. The "bubble" in the symbol indicates inversion: 0 converted to 1, and 1 converted to 0. The equation $Y = \overline{A}$ is read, "Y equals NOT A."

A *buffer*:



The output Y is just a copy of the input A. Logically this is no different from a wire, but electrically, a buffer can help with circuit timing, making sure a device gets enough input current, and so on.

An alternate notation for the NOT operation

The "overline" is probably the most common notation for the NOT operation, but there is another symbol in wide use.

Many textbooks and other documents use the $'$ symbol ("prime") instead. This includes the textbook used up to and including Fall 2012 in ENEL 353.

This year we will use an "overline" for the NOT operation, to be consistent with this year's textbook.

If you see A' in course material from past years, translate that in your head to $\overline{A}$.

Truth tables

A *truth table* describes a combinational logic element by making a list of the values of the output for all possible combinations of input bits.

(We've already seen the truth table for the 1-bit full adder.)

Let's write out the very simple truth tables for a NOT gate and a buffer.

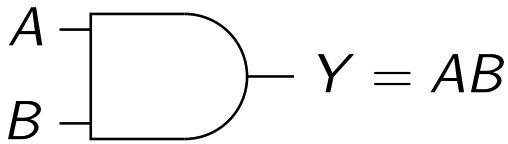
When writing in English about logic gates, use ALL CAPS for the names of gates

Confusing: "For this design you may use or or and gates but not not gates."

Better: "For this design you may use OR or AND gates but not NOT gates."

Our first multi-input gate: the AND gate

Here is the symbol and the algebraic notation:



One way to remember the symbol is to remember that the word AND contains a D, and the symbol looks like a D.

An alternate algebraic notation for AND uses the $\cdot$ symbol, as in $Y = A \cdot B$.

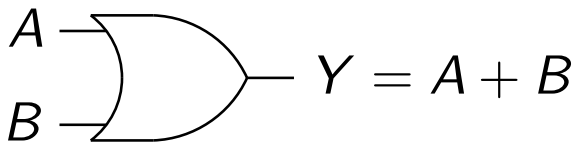
Here is how the AND operation is defined:

$$AB = \begin{cases} 1 & \text{if both } A \text{ and } B \text{ are 1} \\ 0 & \text{otherwise} \end{cases}$$

Let's write the truth table for AND.

The OR gate

Symbol and algebraic notation . . .



Here is how the OR operation is defined:

$$A + B = \begin{cases} 0 & \text{if both } A \text{ and } B \text{ are } 0 \\ 1 & \text{otherwise} \end{cases}$$

In other words, in digital logic, OR means "one or the other **or both** are true."

Let's write the truth table for OR.

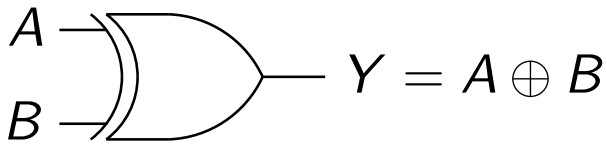
In discussion of digital logic, + does not mean arithmetic addition

- *In decimal arithmetic, what is* $1 + 1$*?*
- *In binary arithmetic, what is* $1 + 1$*?*
- *In digital logic, what is* $1 + 1$*?*

This can be confusing, especially when discussing digital logic circuits that are designed to do arithmetic!

The XOR gate

XOR is short for *exclusive OR*.



"Exclusive or" means "one or the other but **not both**."

$$A \oplus B = \begin{cases} 1 & \text{if one of } A \text{ and } B \text{ is 1 and the other is 0} \\ 0 & \text{otherwise} \end{cases}$$

Let's write the truth table for XOR.

Note: XOR should be pronounced as "ex-or", not as "zor".

Convention for ordering of truth table input combinations

- Start with the row for all inputs = 0.
- Alternate 0's and 1's most slowly on the left.
- Alternate 0's and 1's more quickly as you move left to right.
- Alternate 0's and 1's most quickly—every row—on the right.
- In other words, write the input bit patterns in unsigned binary ordering.

Convention for ordering of truth table input combinations: Examples

This follows the convention ...

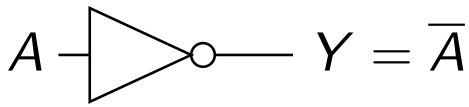
A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

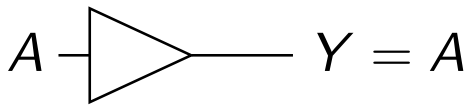
These two tables convey the same information, but should be AVOIDED!

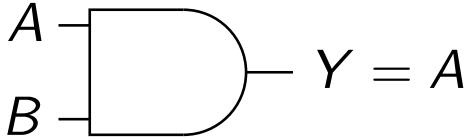
A	B	C	Y
1	1	1	1
1	1	0	1
1	0	1	1
1	0	0	0
0	1	1	0
0	1	0	0
0	0	1	0
0	0	0	0

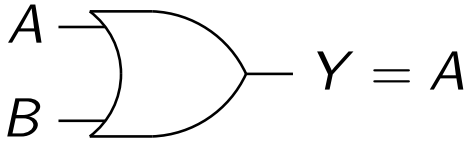
A	B	C	Y
0	0	0	0
1	0	0	0
0	1	0	0
1	1	0	1
0	0	1	0
1	0	1	1
0	1	1	0
1	1	1	1

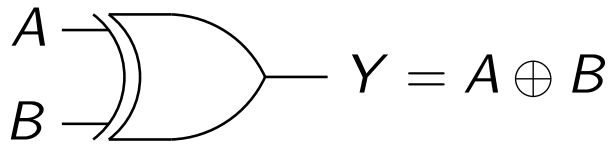
Lots of 1- and 2-input logic gates

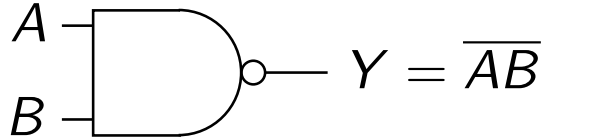
(1) 

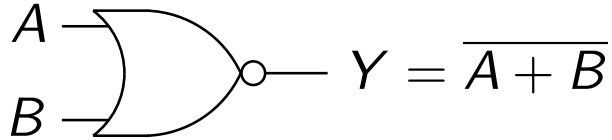
(2) 

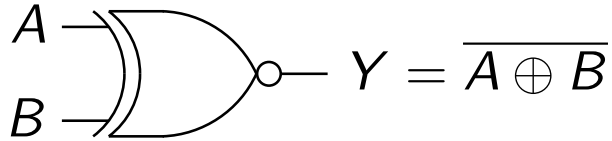
(3) 

(4) 

(5) 

(6) 

(7) 

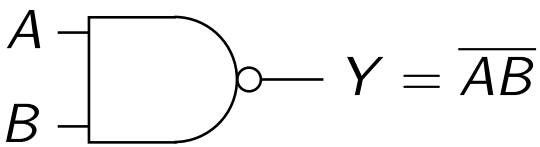
(8) 

Which gates have we named and described so far?

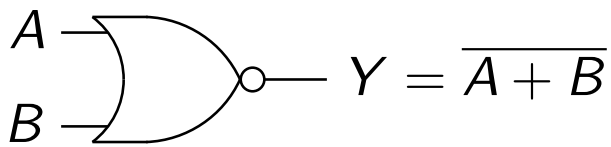
Note: The (1)–(8) numbering is just for this slide and is *not* any kind of standard part numbering!

The NAND and NOR gates

NAND



NOR



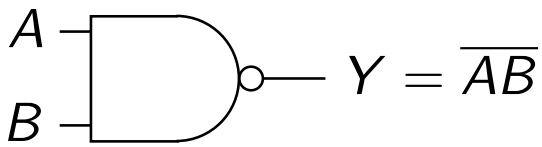
NAND is short for NOT-AND and NOR is short for NOT-OR. Here are the definitions:

- NAND output is the **opposite** of what AND output would be with the same input values.
- NOR output is the **opposite** of what OR output would be with the same input values.

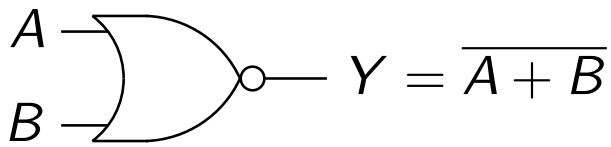
Let's write truth tables.

NAND and NOR: Graphical and algebraic notation

NAND



NOR



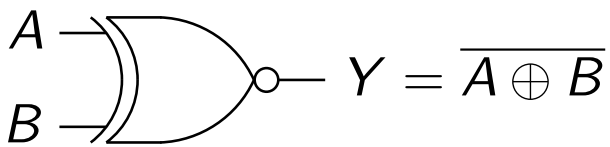
Graphical notation. Note the use of a **bubble** to indicate that NAND is like AND with NOT applied to the AND output. The same kind of use of a bubble relates NOR to OR.

Algebraic notation.

- The expression $\overline{AB}$ says: Generate the AND of A and B, then generate the NOT of the AND result.
- The expression $\overline{A + B}$ says: Generate the OR of A and B, then generate the NOT of the OR result.

The XNOR gate

XNOR is defined as XOR followed by NOT. Note that the symbol for an XNOR gate is the symbol for XOR with a **bubble** applied to the output.



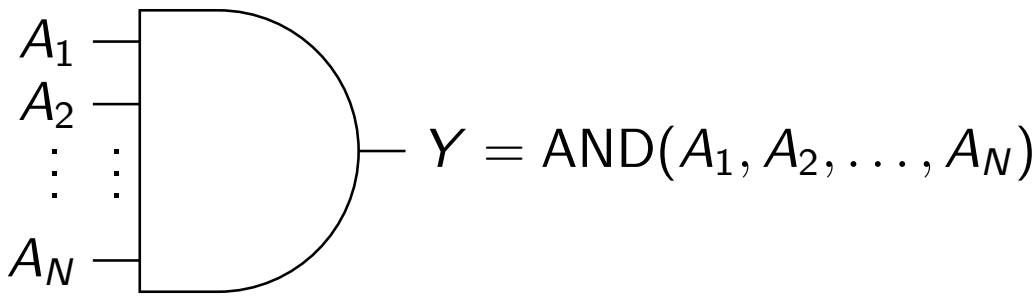
Let's write the truth table for XNOR.

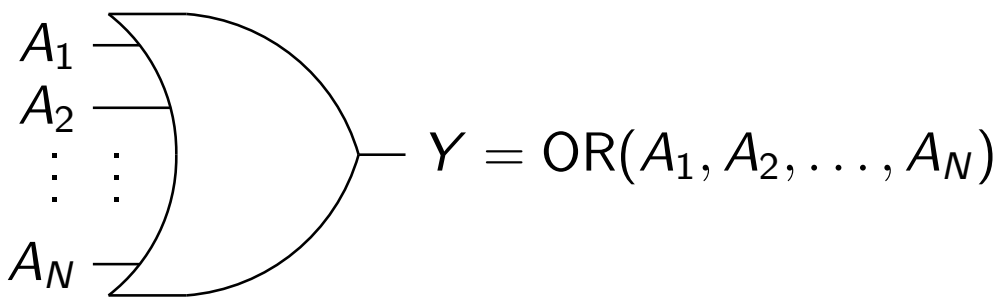
Remark: NXOR would make more sense as a name, but "ex-nor" is much easier to pronounce than "nxor"!

Gates with 3 or more inputs

1- and 2-input gates are the most common, but gates with 3 or more inputs also have lots of practical uses.

Examples: Let's look at the general concept of N-input AND gates and N-input OR gates ...





Definitions for N-input AND, OR, NAND and NOR

$$\text{AND}\,(A_1, A_2, \ldots, A_N) = \begin{cases} 1 & \text{if } \textbf{all} \text{ of } A_1, A_2, \ldots, A_N \text{ are } 1 \\ 0 & \text{otherwise} \end{cases}$$

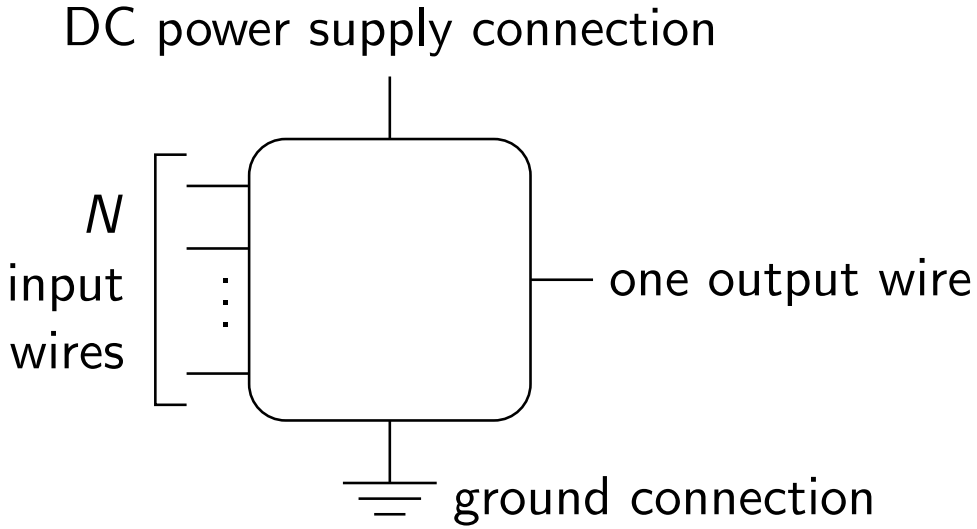
$$\text{OR}\,(A_1, A_2, \ldots, A_N) = \begin{cases} 0 & \text{if } \textbf{all} \text{ of } A_1, A_2, \ldots, A_N \text{ are } 0 \\ 1 & \text{otherwise} \end{cases}$$

N-input NAND and NOR are formed by applying NOT to the output of N-input AND and OR, respectively.

- *Let's draw a symbol and write a truth table for an OR3 (3-input OR) gate.*
- *Let's draw a symbol and write a truth table for a NAND4 (4-input NAND) gate.*

Outline of Slide Set 2

Electrical connections for a generic logic gate



Ground is assumed to be 0 V.

The DC power supply voltage—called $V_{\mathbf{DD}}$ in our textbook—depends on what **logic family** the gate belongs to.

For logic families designed in the 1970's and 80's, $V_{\mathrm{DD}} = +5.0\,\mathrm{V}$ was just about standard. Since then, newer families have had lower and lower values. V_{DD} of $+1.2\,\mathrm{V}$ or less is common in 2019.

Rough guide to electrical signalling in logic circuits

These rules are good enough to get a general idea about how signalling works in digital circuits:

- Voltages fairly close to V_{DD} represent a logic level of 1.
- Voltages fairly close to ground represent a logic level of 0.
- Voltages somewhere near the middle between ground and V_{DD} are **ambiguous** (might be read as 0, might be read as 1), and should be avoided.

However, language like "fairly close" and "somewhere near the middle" is **not nearly precise enough** to guide the design of **reliable** digital circuits.

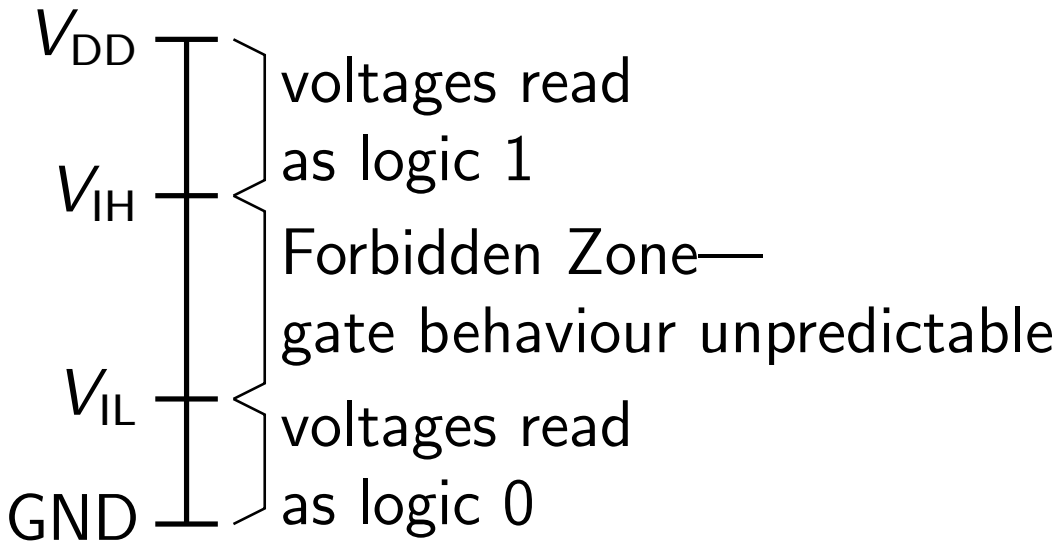
V_{IL}, V_{IH}, V_{OL}, and V_{OH}

These four voltage levels are specifications that are properties of a **logic family**. The concepts are the same for all logic families, but the exact numbers vary from one family to another.

symbol	**name and meaning**
V_{IL}	low-level input voltage; voltages $\leq V_{IL}$ reliably read as logic 0
V_{IH}	high-level input voltage; voltages $\geq V_{IH}$ reliably read as logic 1
V_{OL}	low-level output voltage; gate output for logic 0 must be $\leq V_{OL}$
V_{OH}	high-level output voltage; gate output for logic 1 must be $\geq V_{OH}$

The "Forbidden Zone"

The range of voltages between V_{IL} and V_{IH} is called the "forbidden zone."



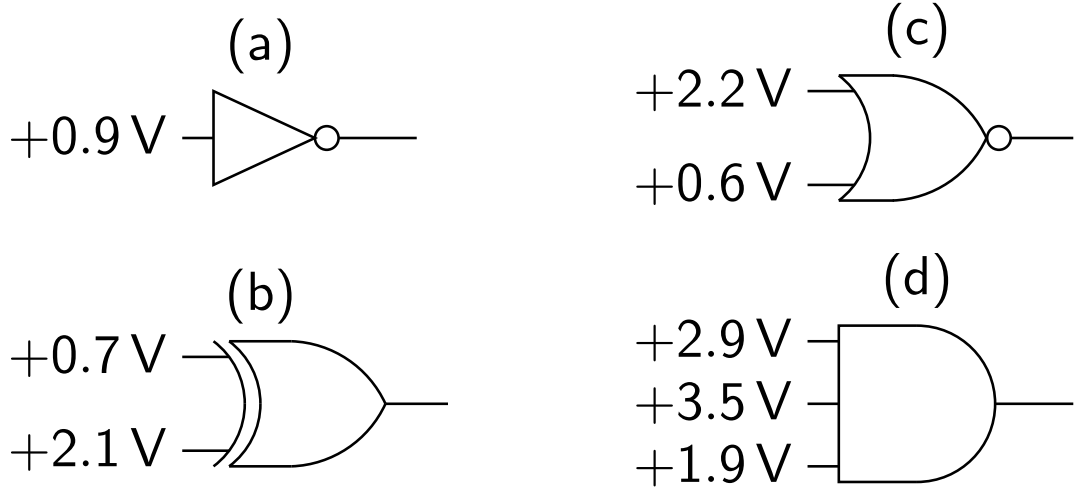
If one or more input wires of a logic gate has a voltage within the forbidden zone, the output voltage of the gate **may be unpredictable**.

V_{IL}, V_{IH}, V_{OL}, V_{OH}: TTL examples

Here are numbers for TTL logic families . . .

item	voltage
V_{DD}	5.0 V
V_{IL}	0.8 V
V_{IH}	2.0 V
V_{OL}	0.4 V
V_{OH}	2.4 V

What can we say about the output voltages of these TTL gates? (Note: V_{DD} and GND connections are often omitted from diagrams to reduce clutter.)



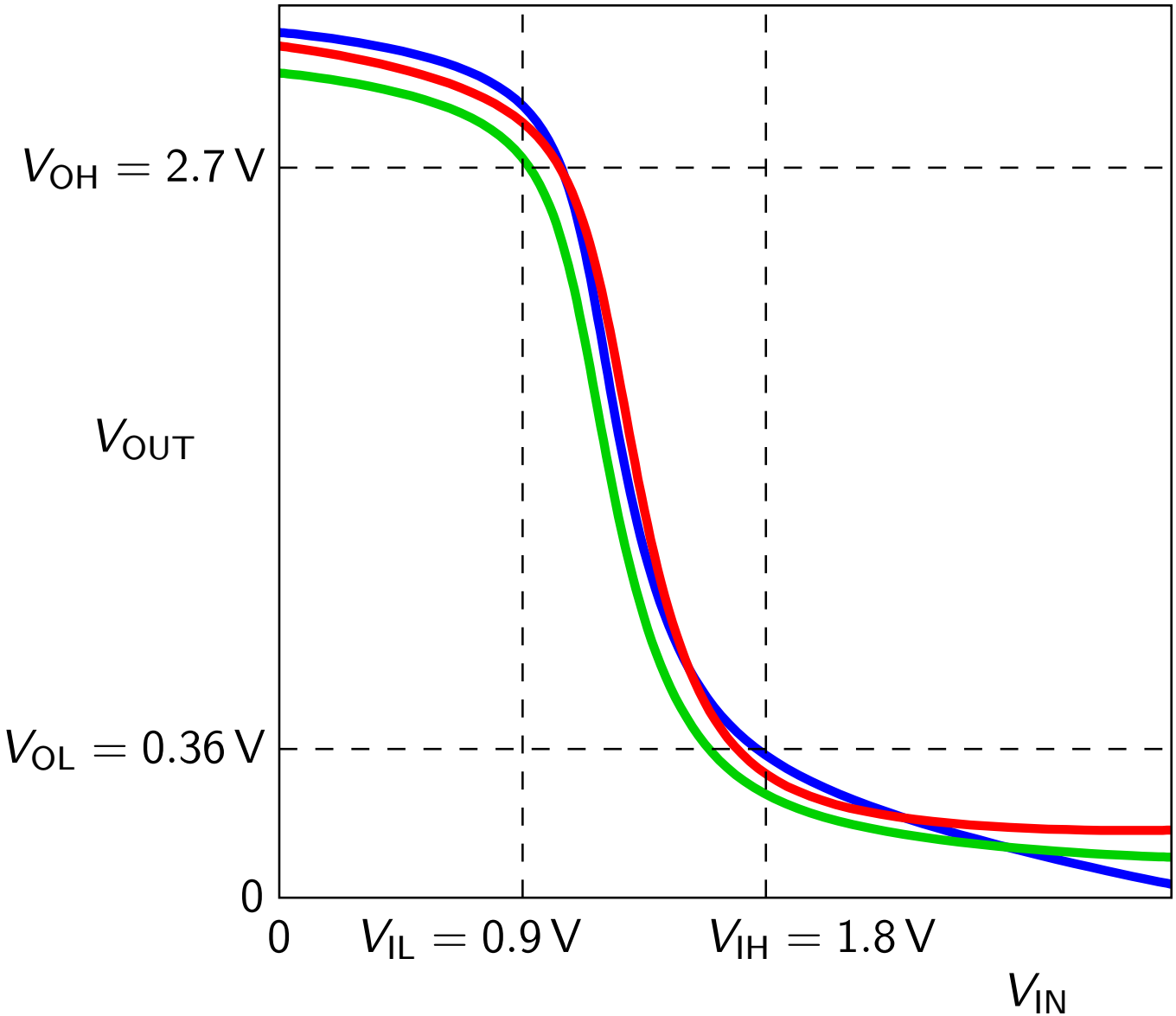
V_{IL}, V_{IH}, V_{OL}, V_{OH}: LVCMOS examples

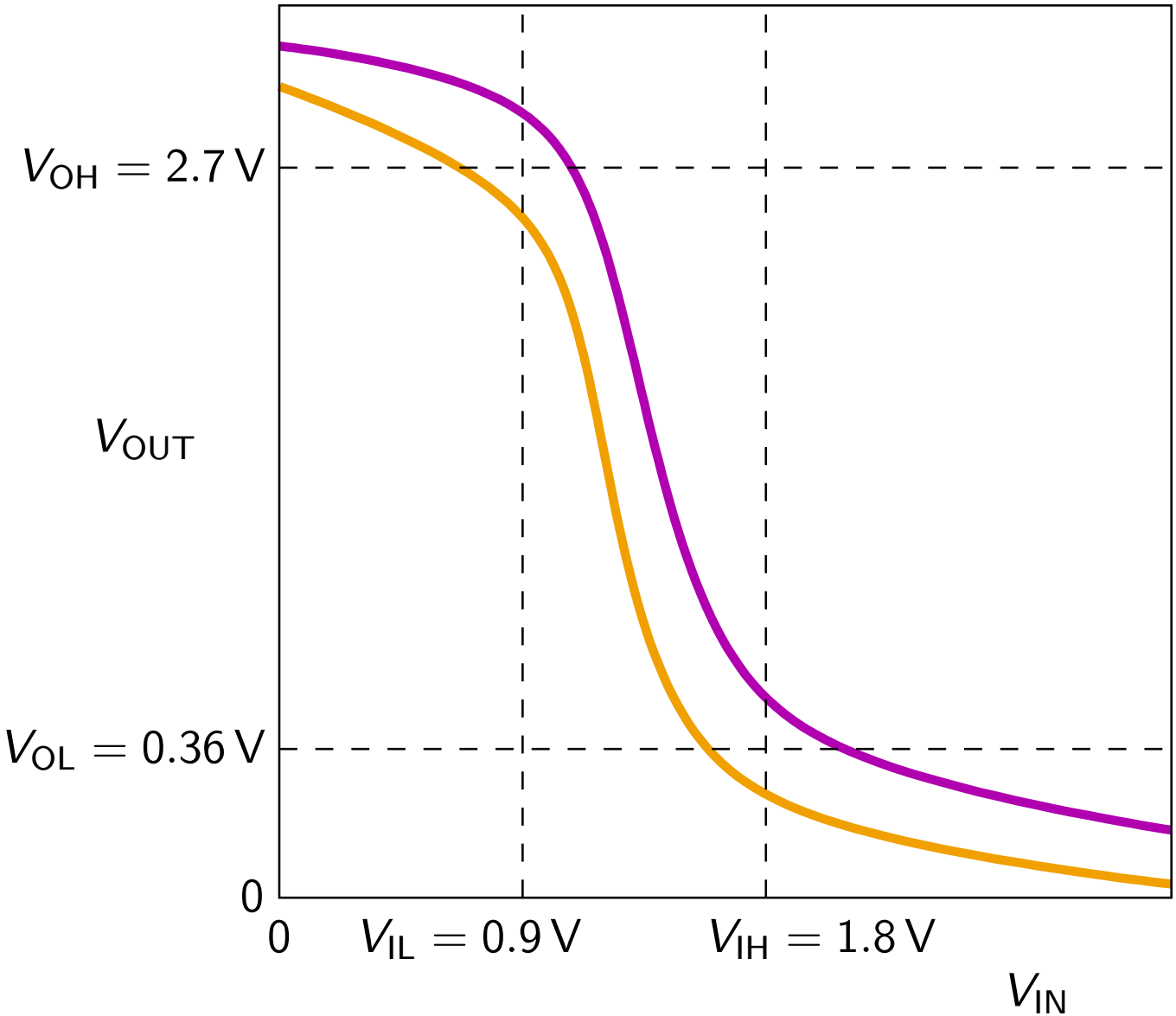
Here are numbers for low-voltage CMOS logic families . . .

item	voltage
V_{DD}	3.3 V
V_{IL}	0.9 V
V_{IH}	1.8 V
V_{OL}	0.36 V
V_{OH}	2.7 V

The next two slides show five plots of output voltage versus input voltage for inverters that are supposed to meet voltage specifications for LVCMOS.

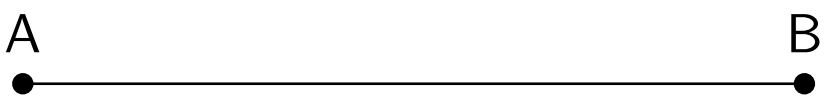
Which of the inverters actually do meet those specifications?


$V_{OH} = 2.7\,V$
V_{OUT}
$V_{OL} = 0.36\,V$
0
0
$V_{IL} = 0.9\,V$
$V_{IH} = 1.8\,V$
V_{IN}


$V_{OH} = 2.7\,V$
V_{OUT}
$V_{OL} = 0.36\,V$
0
0
$V_{IL} = 0.9\,V$
$V_{IH} = 1.8\,V$
V_{IN}

Ideal wires, real wires, and noise

Here is a wire:



In ENGG 225—and many other circuits courses—wires are modeled as **ideal**: The voltage at node B is always the same as the voltage at node A.

In a **real wire** the voltage at node B is **not necessarily the same** as the voltage at node A! One reason for this is **additive noise**: Very roughly speaking, a real wire acts as an antenna and picks up signals from its electromagnetic environment.

(Another reason is that in a real wire it takes some time for voltage changes to propagate from one end of the wire to the other, but we're not going to worry about that in this course.)

Digital circuits can tolerate a significant amount of noise

Noise tolerance is one of many properties that make digital circuits really useful!

Two quantities that can be computed from V_{IL}, V_{IH}, V_{OL}, and V_{OH} are:

- *low-level noise margin*: $NM_L = V_{IL} - V_{OL}$
- *high-level noise margin*: $NM_H = V_{OH} - V_{IH}$

These two numbers describe how much noise can be added to the wire from the output of a gate to the input of another gate without causing unreliable circuit behaviour.

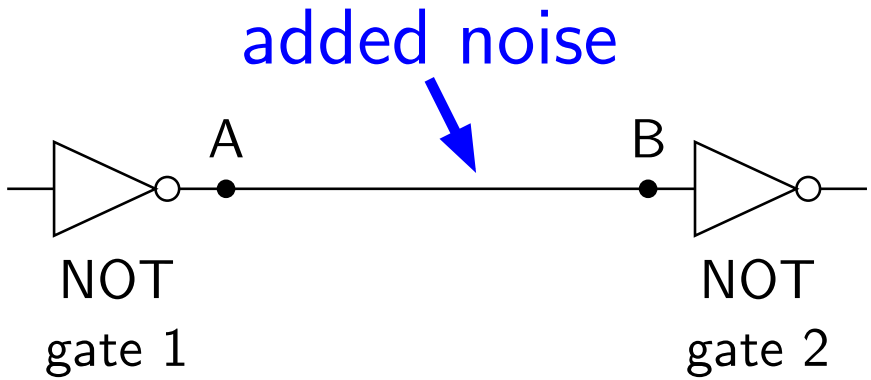
Noise margin example: Low-voltage CMOS

Here again are the numbers for the low-voltage CMOS logic family . . .

item	voltage
V_{DD}	3.3 V
V_{IL}	0.9 V
V_{IH}	1.8 V
V_{OL}	0.36 V
V_{OH}	2.7 V

What are NM_L and NM_H for this logic family?

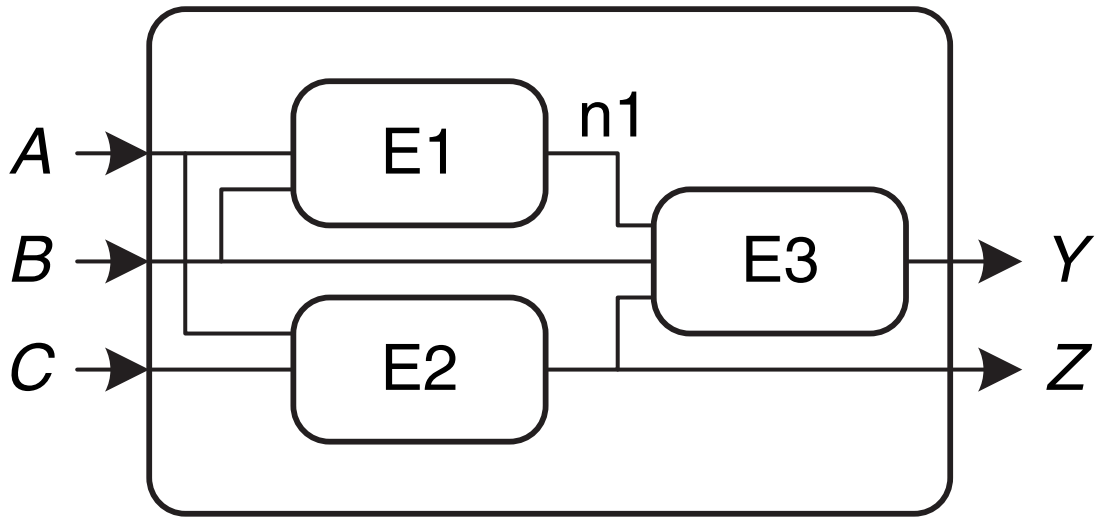
Suppose the input to NOT gate 1 is 3.1 V. How much noise can there be on the wire from A to B before NOT gate 2 might produce wrong output? What if the input to NOT gate 1 is 0.2 V?



Outline of Slide Set 2

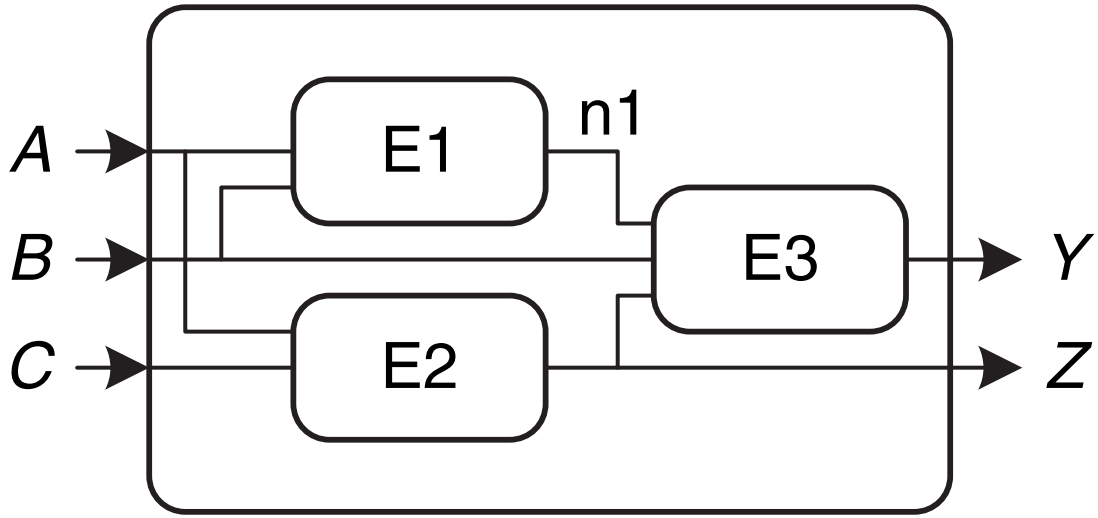
Inputs, outputs, elements and nodes

Here is an example logic circuit:



- The **inputs** are A, B and C.
- The **outputs** are Y and Z.

Image is Figure 2.2 from Harris D. M. and Harris S. L., *Digital Design and Computer Architecture, 2nd ed.*, © 2013, Elsevier, Inc.



- E1, E2 and E3 are called *elements*—each element is **itself a logic circuit**, simpler than the overall circuit.
- A *node* is a **wire** whose voltage communicates a bit value between elements.
- A, B and C are called *input nodes*; Y and Z are called *output nodes*.
- n1 is an example of an *internal node*—a node that is neither an input nor an output of the overall circuit.

Review: *Combinational* versus *Sequential* Logic

The outputs of a *combinational* logic circuit depend only the **current** values of its inputs.

The outputs of a *sequential* logic circuit depend on the **history** of its input values.

A **logic gate** is a **combinational** logic circuit that has a single output, not multiple outputs.

Generic symbol for combinational logic

The symbol Ϛ is often used as a label to indicate that a logic element is combinational, without actually specifying the function of the logic element.

Let's draw a diagram for a generic 3-input, 2-output combinational logic element, then let's draw it another way.

Let's draw a diagram for a generic combinational logic element, with unspecified numbers of inputs and outputs.

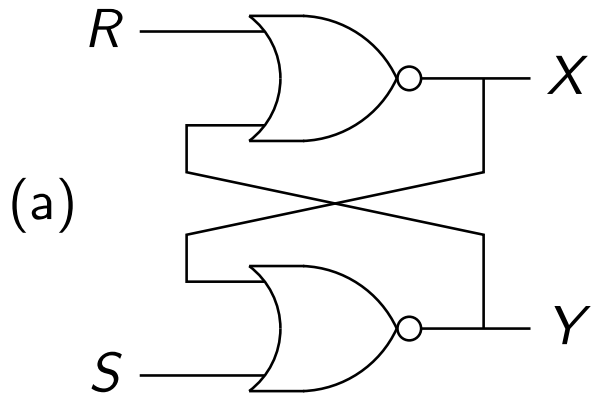
Combinational composition

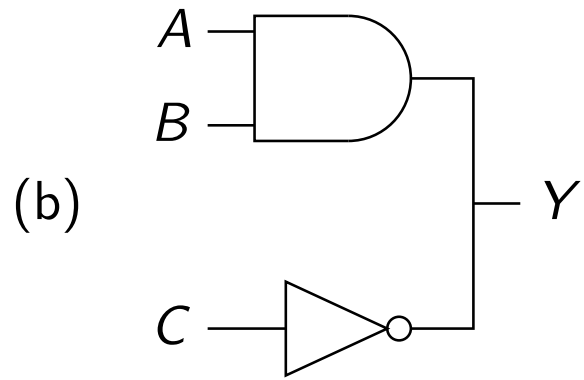
Combinational composition is a term for a method of building complex combinational circuits out of simpler elements. The rules for combinational composition are:

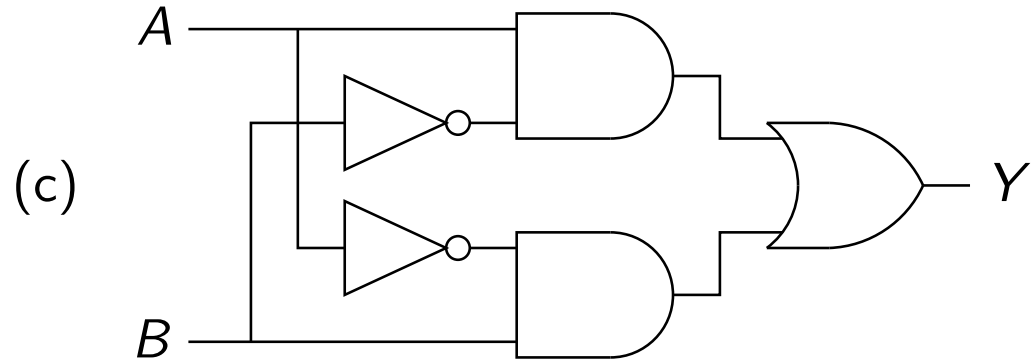
- Each of the elements must itself be combinational.
- Each node in the circuit is either an input node or connects to **exactly one output of an element**.
- There must be **no cyclic paths**—no way to make a path through the circuit that goes more than once through a single element.

If a circuit satifies all three of these rules, it is guaranteed to be combinational.

Examples with logic gates: *Which ones satisfy the combinational composition rules?*







Examples with generic combinational elements

Does either circuit satisfy the combinational composition rules?