TO REVIEW AMPLIFIERS…

- Prove formulas for various amplifier configurations
- Write a linear function and try to create an amplifier to perform the function
- Find the input-output formula for one of the simpler amplifier configurations using the circuit model
- Sketch what would happen to V_0 if you put a large-amplitude sinusoid as the input to an amplifier. Determine when the output would “rail”.

Today, we will look at digital logic gates.

EECS 40 Spring 2003 Lecture 11 Copyright Regents of the University of California S. Ross and W. G. Oldham

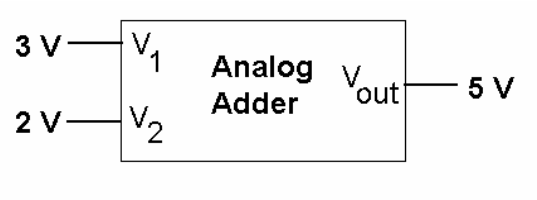
EECS 40 Spring 2003 Lecture 11 Copyright Regents of the University of California S. Ross and W. G. Oldham

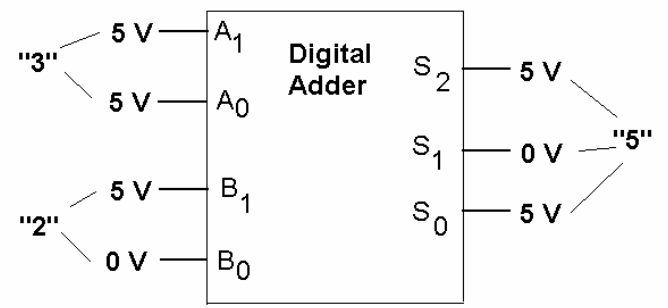
DIGITAL ADVANTAGES

- Digital Communication
  - Digital representation makes encoding and processing easier
  - Reconstruct signal to arbitrary accuracy (need bandwidth)
- Computers
  - Arbitrary computational accuracy, just add more significant digits (big floating-point unit)
  - Can store information with arbitrary accuracy (big memory)

EECS 40 Spring 2003 Lecture 11 Copyright Regents of the University of California S. Ross and W. G. Oldham

ANALOG VS. DIGITAL

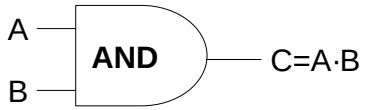




EECS 40 Spring 2003 Lecture 11 Copyright Regents of the University of California S. Ross and W. G. Oldham

EECS 40 Spring 2003 Lecture 11 Copyright Regents of the University of California S. Ross and W. G. Oldham

LOGIC GATES



Here we have an **abstract** symbol for a logic circuit. A, B, and C are **Boolean variables**. Each can take on the value logic 1 or logic 0.

Remember, logic 1 is represented by a potential of, say, 5 V with respect to ground, and logic 0 is usually represented by a potential of 0 V.

Values close to these nominal levels still work as logic 1 or logic 0, e.g., 4 V may work as logic 1.

We will study the detailed electrical workings of logic gates in weeks to come, today we will be abstract.

EECS 40 Spring 2003 Lecture 11 Copyright Regents of the University of California S. Ross and W. G. Oldham

LOGICAL FUNCTIONS

- **"AND"** $A \cdot B$ **alternatively** AB
- **"OR"** $A + B$
- **"INVERT" or "NOT"** $\overline{A}$
- **"not AND" = NAND** $\overline{A \cdot B}$ **alternatively** $\overline{AB}$
- **"not OR" = NOR** $\overline{A + B}$
- **exclusive OR = XOR** $A \oplus B$

EECS 40 Spring 2003 Lecture 11 Copyright Regents of the University of California S. Ross and W. G. Oldham

EECS 40 Spring 2003 Lecture 11 Copyright Regents of the University of California S. Ross and W. G. Oldham

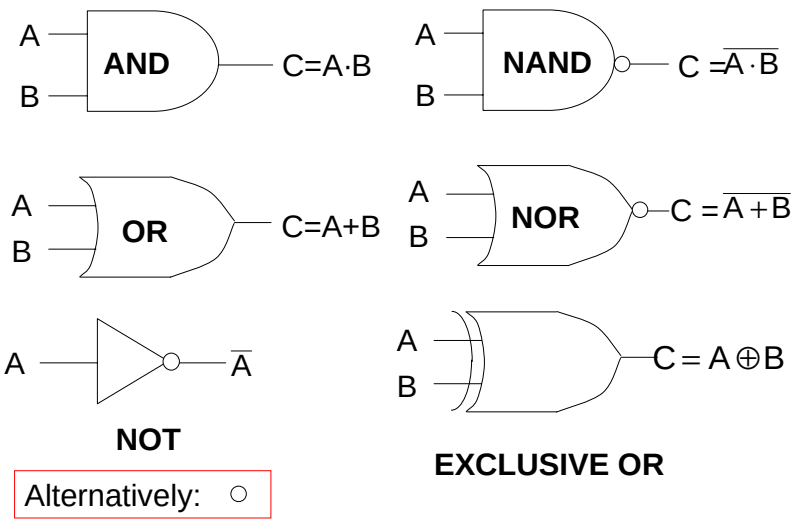
TRUTH TABLE

A truth table gives the logic function output for each possible combination of inputs.

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

EECS 40 Spring 2003 Lecture 11 Copyright Regents of the University of California S. Ross and W. G. Oldham

LOGIC GATE CIRCUIT SYMBOLS



EECS 40 Spring 2003 Lecture 11 Copyright Regents of the University of California S. Ross and W. G. Oldham

EECS 40 Spring 2003 Lecture 11 Copyright Regents of the University of California S. Ross and W. G. Oldham

LOGICAL SYNTHESIS

Suppose we are given a truth table for a logic function we would like to create.

Is there a method to implement the logical function using these basic logic gates?

Answer: There are lots of ways, but one simple way is implementation from “sum of products” formulation.

How to do this: 1) Write sum of products expression from truth table and 2) implement using standard gates.

We may not get the most efficient implementation this way, but we can simplify the circuit afterwards.

EECS 40 Spring 2003 Lecture 11 Copyright Regents of the University of California S. Ross and W. G. Oldham

EXAMPLE: ADDER

A	B	C	S_1	S_0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Input Output

S_1 using sum-of-products:

1) Find where S_1 is "1"

2) Write down product of inputs which create each "1"

$\overline{A}\,B\,C$ $A\,\overline{B}\,C$

$A\,B\,\overline{C}$ $A\,B\,C$

3) Sum all products

$\overline{A}\,B\,C + A\,\overline{B}\,C + A\,B\,\overline{C} + A\,B\,C$

4) Draw circuit

PROPERTIES OF BOOLEAN LOGIC

$A + 0 = A$ $A \cdot 1 = A$

$A + \overline{A} = 1$ $A \cdot \overline{A} = 0$

$A + A = A$ $A \cdot A = A$

$A + B = B + A$ $A \cdot B = B \cdot A$

$A + (B + C) = (A + B) + C$ $(A \cdot B) \cdot C = A \cdot (B \cdot C)$

$A \cdot (B + C) = A \cdot B + A \cdot C$ $A + B \cdot C = (A + B) \cdot (A + C)$

$A + A \cdot B = A$ $A \cdot (A + B) = A$

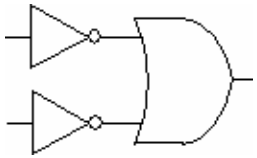
DeMorgan's Law:

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

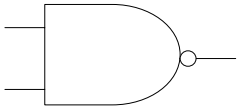
$\overline{A} \cdot \overline{B} = \overline{A + B}$

NAND-NAND IMPLEMENTATIONS

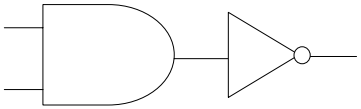
DeMorgan's law tells us this



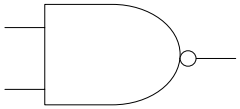
is the same as this



And by definition this



is the same as this



so all sum-of-products expressions can be implemented with one kind of gate: NAND gates. Just replace AND and OR with NAND.

EECS 40 Spring 2003 Lecture 11 Copyright Regents of the University of California S. Ross and W. G. Oldham

EECS 40 Spring 2003 Lecture 11 Copyright Regents of the University of California S. Ross and W. G. Oldham

CREATING A BETTER CIRCUIT

What makes a better digital circuit? Fast and low cost = better.

- Fewer stages
- Fewer total number of individual gates
- Fewer types of gates
- Fewer inputs on each gate (multi-input gates are slower)

Let's try to simplify the sum-of-products expression for S_0 and make a better circuit.

We can use the properties of Boolean logic to do simplification.

EECS 40 Spring 2003 Lecture 11 Copyright Regents of the University of California S. Ross and W. G. Oldham