EECS 42 Introduction to Electronics for Computer Science

Andrew R. Neureuther

Lecture # 11 Logic Implementation

- Logic Levels and Gate Circuits
- Combination of Logic Functions
- Synthesis from a Truth Table
- NAND Gate Synthesis
- XOR and Introduction to Timing

http://inst.EECS.Berkeley.EDU/~ee42/

Game Plan 03/04/03

Monday 03/04/04

- ❑ Review
- ❑ Logic Implementation: 11.2-11.3 pp. 403-422

Wednesday 03/05/03:

- ❑ Midterm In Class, Closed Book, Closed Notes, Paper Provided, Bring Calculator

Next (8th) Week:

- ❑ Monday: Physical Limits of Logic
- ❑ Wednesday: Dependent Sources

No Problem Set Due 7th week.

Problem set #6 out Monday 3/3 and due at 2:30 3/10 in box in 240 Cory

REMINDER

Midterm March 5th, 3:10-4:03 PM

Closed Book, Closed Notes, Bring Calculator, Paper Provided

Old Exams Are Posted on Web

Review Session 5-7 Mon 3/4/04 in 241 Cory

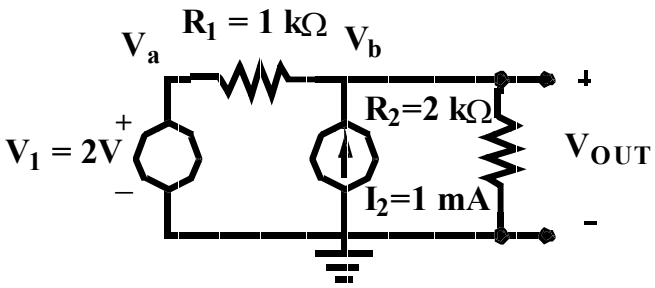
EE 43 Labs Are Not Cancelled but will be light:

First Midterm Exam: Topics

- Basic Circuit Analysis (KVL, KCL)
- Equivalent Circuits and Graphical Solutions for Nonlinear Loads
- Transients in Single Capacitor Circuits
- Node Analysis Technique and Checking Solutions

Exam is in class 3:10-4:03 PM, Closed book, Closed notes, Bring a calculator, Paper provided

Example: Basic Circuit Analysis

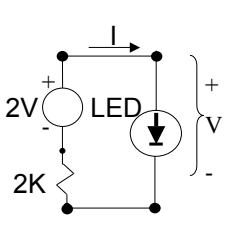


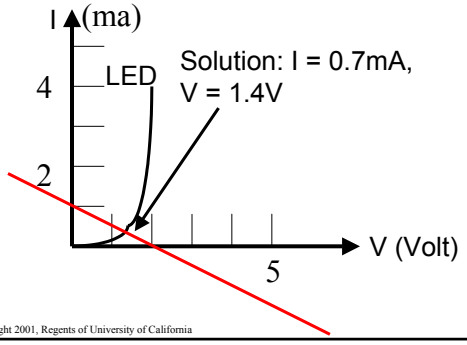
Example: Load-Line Method

Lets hook our 2K resistor + 2V source circuit up to an LED (light-emitting diode), which is a very nonlinear element with the IV graph shown below.

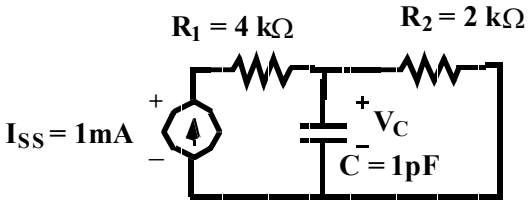
Again we draw the I-V graph of the 2V/2K circuit on the same axes as the graph of the LED. Note that we have to get the sign of the voltage and current correct!! (The sign of the current is reversed from I_{SC})

At the point where the two graphs intersect, the voltages and the currents are equal, in other words we have the solution.

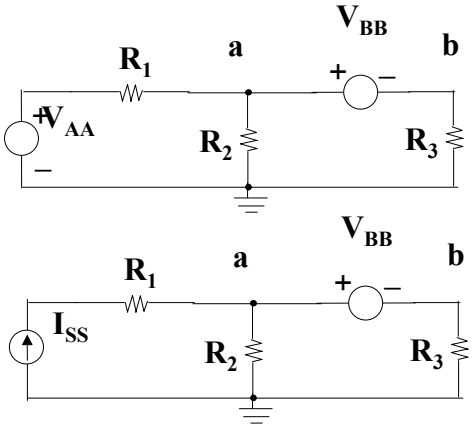




Example: Transient



Example: Node Equation method

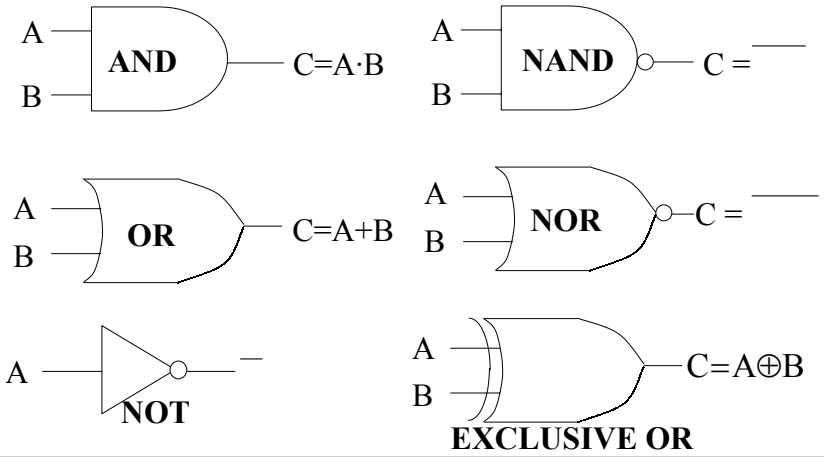


Some Important Logical Functions

- **"AND"** $A \cdot B$ (or $A \cdot B \cdot C$)
- **"OR"** $A + B$ (or $A+B+C+D\ldots$)
- **"INVERT" or "NOT"** not A (or $\overline{A}$)
- **"not AND" = NAND** $\overline{AB}$ (only 0 when A and $B = 1$)
- **"not OR" = NOR** $\overline{A+B}$ (only 1 when $A = B = 0$)
- **exclusive OR = XOR** $A \oplus B$ (only 1 when A, B differ)
  i.e., $A + B$ **except** $A \cdot B$

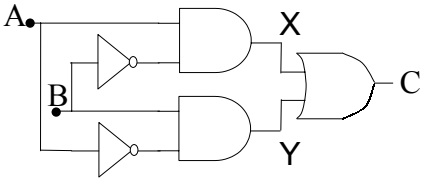
Logic Gates

These are circuits that accomplish a given logic function such as "OR". We will shortly see how such circuits are constructed. Each of the basic logic gates has a unique symbol, and there are several additional logic gates that are regarded as important enough to have their own symbol. The set is: AND, OR, NOT, NAND, NOR, and EXCLUSIVE OR.



Logic Circuits

With a combination of logic gates we can construct any logic function. In these two examples we will find the truth table for the circuit.



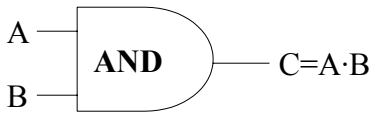
It is helpful to list the intermediate logic values (at the input to the OR gate). Let's call them X and Y.

Now we complete the truth tables for X and Y, and from that for C. (Note that $X = A \bullet \overline{B}$ and $Y = B \bullet \overline{A}$ and finally C = X + Y)

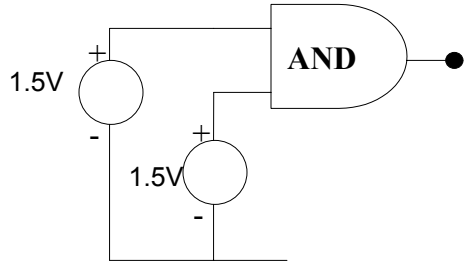
A	B	X	Y	C
0	0	0	0	0
0	1	0	1	1
1	0	1	0	1
1	1	0	0	0

Interestingly, this is the same truth table as the EXCLUSIVE OR

Logic Gates – How are they used in practice?



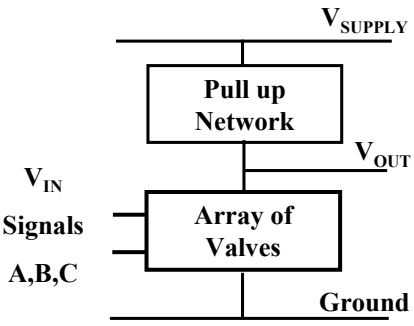
First of all we must agree on what is High (logical **1**) or low (logical **0**). Suppose 1.5 V is **1** and 0V is logical **0**.



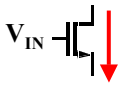
C would have the value of 1.5 V (logical 1).

But it would have the value of 0V logical 0 if either one of the inputs were held at zero V.

Logic Gates – How are they built in practice?

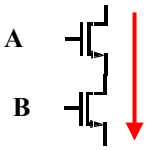


A Valve is a Transistor

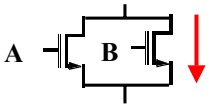


Current flows when V_{IN} is high

Valves in Series => NAND



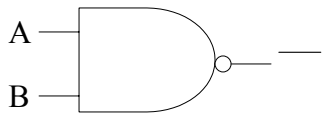
Valves in Parallel => NOR



(You can learn about building gates in EE 141.)

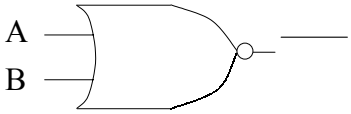
The most common basic gates are NAND and NOR?

Not-AND = NAND



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

Not-OR = NOR

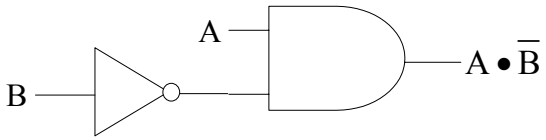


A	B	A+B	$\overline{A+B}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

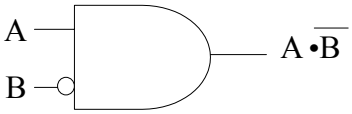
How to Combine Gate to Produce a Desired Logic Function? (More basic Logical Synthesis)

Example

$F = A \bullet \overline{B}$



Again a little shorthand is useful



How to Combine Gate to Produce a Desired Logic Function? (More basic Logical Synthesis)

Suppose we are given a truth table (all logic statements can be represented by a truth table). How can we implement the function?

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.

(Warning this is probably inefficient – we need to minimize, or simplify the expression. You will learn this in CS 150.)

How to Combine Gate to Produce a Desired Logic Function? (More basic Logical Synthesis)

Example:

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

Clearly: F= 1 if

$\overline{A}\,\overline{B}\,C = 1$

or

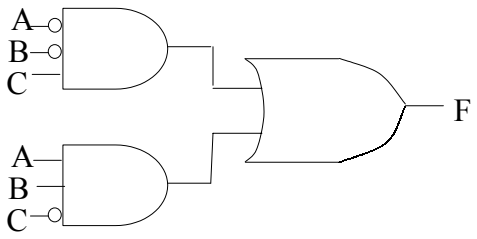
$AB\overline{C} = 1$

i.e. $F=\overline{A}\,\overline{B}C + AB\,\overline{C}$

How to Combine Gate to Produce a Desired Logic Function? (More basic Logical Synthesis)

Example:

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0



$F=\overline{A}\,\overline{B}C + AB\,\overline{C}$

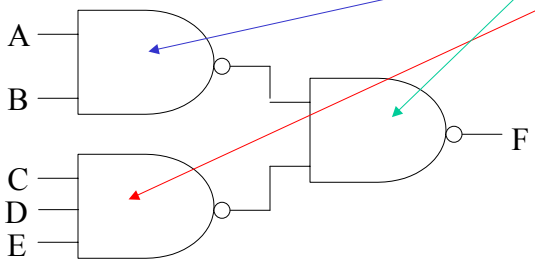
Logical Synthesis Guided by DeMorgan's Theorem

DeMorgan's Theorem :

$$A + B + C = \overline{[\overline{A}\,\overline{B}\,\overline{C}]} \quad \text{or} \quad \overline{A} + \overline{B} + \overline{C} = \overline{[A\,B\,C]}$$

Example of Using DeMorgan's Theorem:

$$F = A \bullet B + C \bullet D \bullet E = \overline{\overline{AB} \bullet \overline{CDE}}$$



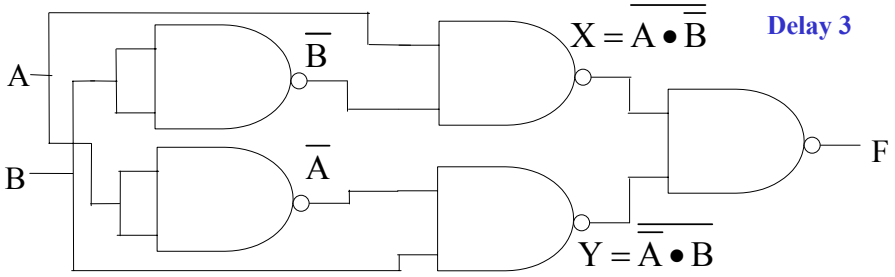
Thus any sum of products expression can be immediately synthesized from NAND gates alone

Logical Synthesis of XOR

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

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

We Need a Timing Diagram!



Timing Diagram for Delays in Logic