



CSE140L: Digital Systems Laboratory

Introduction

Instructor: Mohsen Imani

Slides from:

Dr. Pietro Mercati & Prof. Tajana Simunic Rosing

Welcome to CSE 140L!

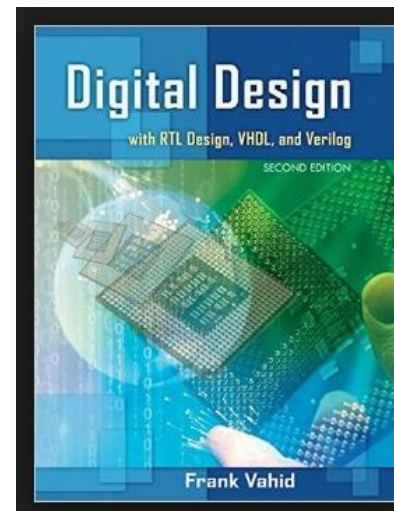
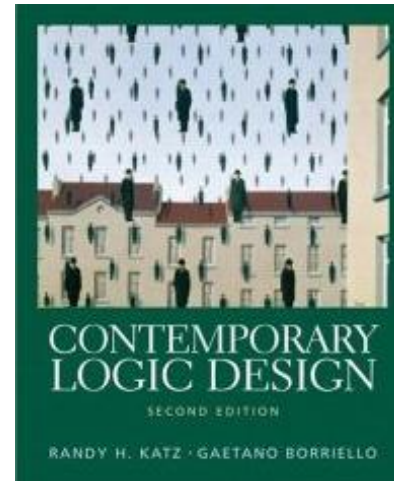
- **Instructor: Mohsen Imani**
 - Email: moimani@ucsd.edu;
 - please put “CSE140L” in the subject line
 - Office Hours:
 - Mon 2-4pm CSE 2217
- **Website:** https://cseweb.ucsd.edu/classes/su17_2/cse140L-a/index.html
 - Homeworks and slides will be here
- **TAs and Tutors**
 - Office hours listed on the class website/Piazza
- **Discussion sessions:** TA will hold both lab and office hour
- **Grades:** <https://tritoned.ucsd.edu/>
- **Announcements and online discussion:** <https://piazza.com>
 - SIGN UP SOON !!!
- **Lab:** TBD

Course Description

- Prerequisites:
 - CSE 20 or Math 15A, and CSE 30.
 - CSE 140 must be taken concurrently
- Objective:
 - Introduce digital components and system design concepts through hands-on experience in a lab
- Grading:
 - Class participation using iClicker: 5%
 - Homeworks (5): #1: 0%, #2,#3: 10% , #4, #5: 15%
 - Final: 45%
- Homeworks:
 - Can be solved individually or in teams of two.
 - Each homework solution should be checked-off with a tutor or a TA before the end of the day on the due date
 - Detailed instructions will be provided on the homework

Textbook and Recommended Readings

- **Recommended textbook:**
 - Contemporary Logic Design by R. Katz & G. Borriello
- Recommended textbook:
 - Digital Design by F. Vahid
- Lecture slides are derived from the slides designed for both books





Demo Overview

Outline

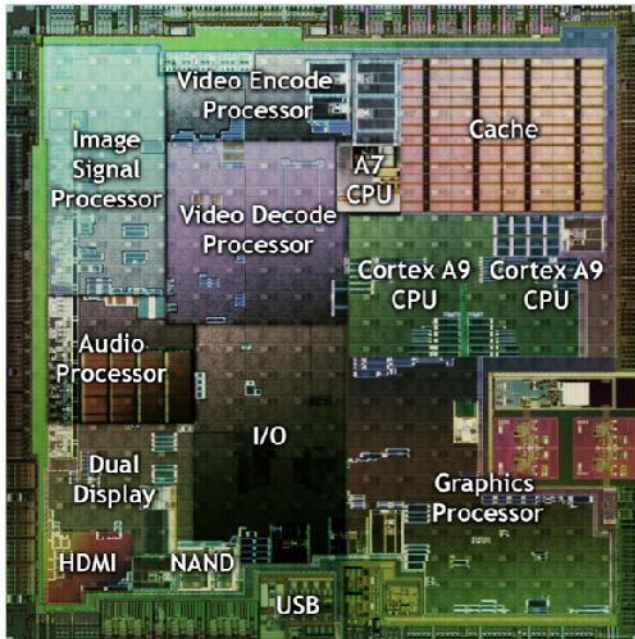
- Transistors
 - How they work
 - How to build basic gates out of transistors
 - How to evaluate delay



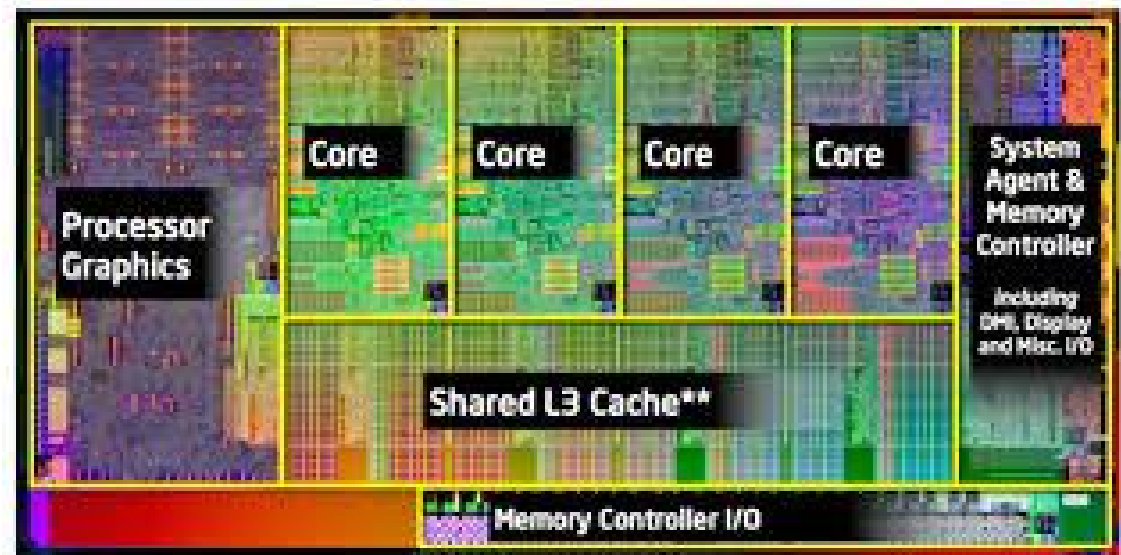
Combinational circuit building blocks: Transistors, gates and timing

The Scope of CSE140L

- We start with Boolean algebra $Y = A \text{ and } B$
- End up with the design of a simple CPU

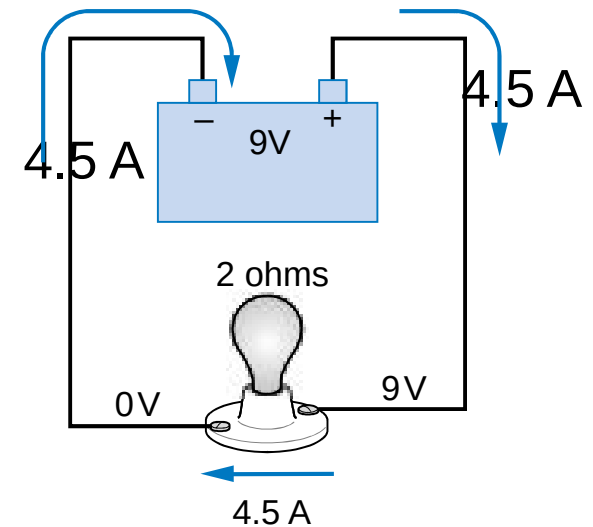


Nvidia Tegra 2 die photo



Switches

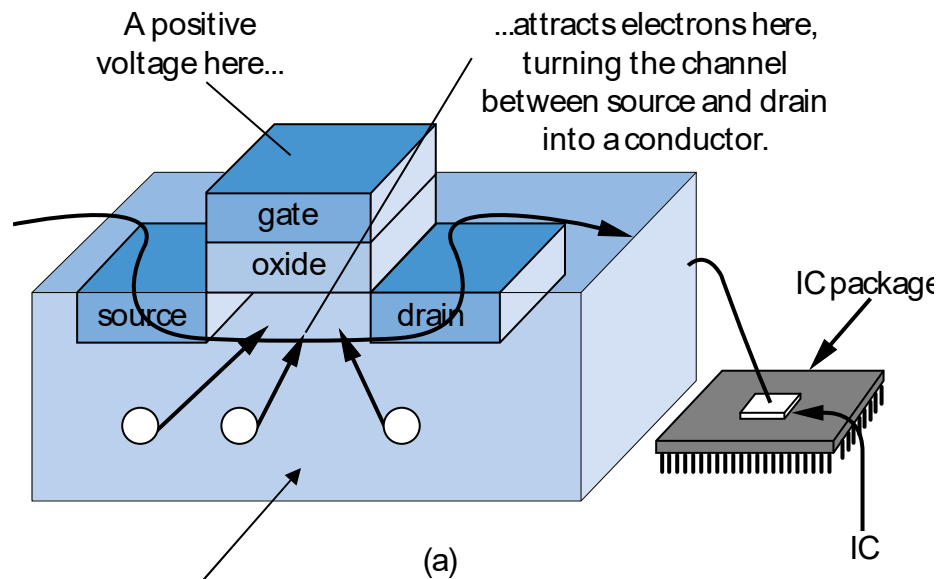
- Electronic switches are the basis of **binary digital circuits**
 - Electrical terminology
 - **Voltage**: Difference in electric potential between two points
 - Analogous to water pressure
 - **Current**: Flow of charged particles
 - Analogous to water flow
 - **Resistance**: Tendency of wire to resist current flow
 - Analogous to water pipe diameter
 - $V = I * R$ (Ohm's Law)



“Binary Digital” = all values are either 0 (low voltage) or 1 (high voltage)

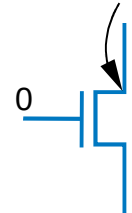
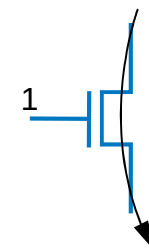
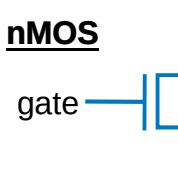
The CMOS Switches

- CMOS circuit (Complementary – MOS)
 - Consists of N and PMOS transistors
 - Both N and PMOS are similar to basic switches
 - $R_p \sim 2 R_n \Rightarrow$ PMOS in series is much slower than NMOS



Silicon -- not quite a conductor or insulator:
Semiconductor

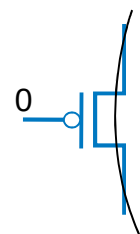
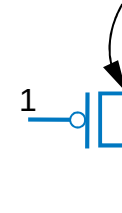
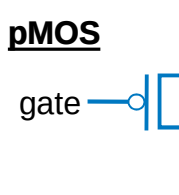
nMOS



conducts

does not conduct

pMOS



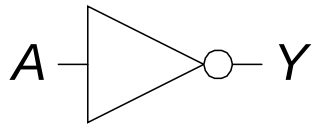
does not conduct

conducts

1 = high voltage
0 = low voltage

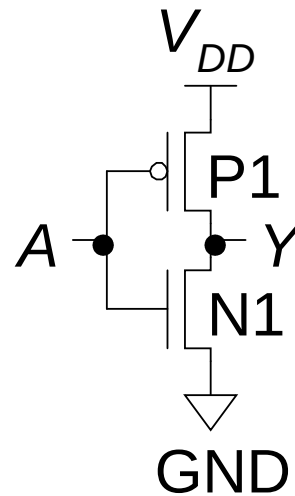
Logic gates: CMOS NOT Gate

NOT



$$Y = \overline{A}$$

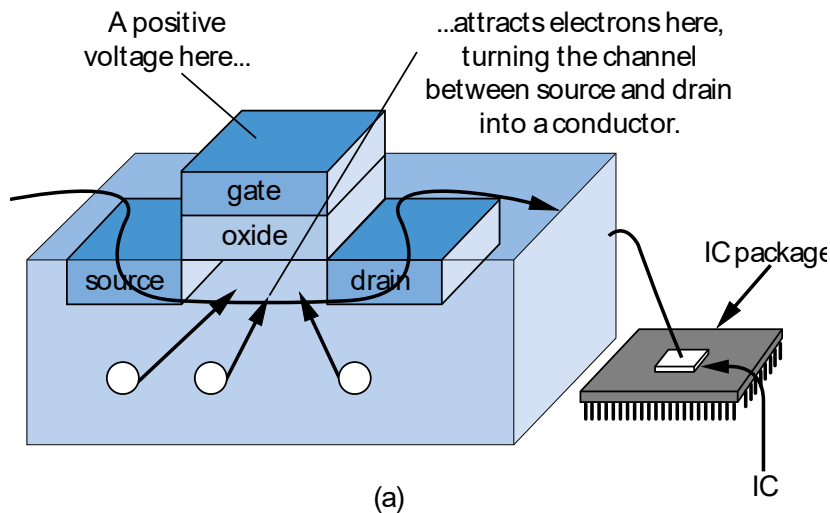
A	Y
0	1
1	0



A	P1	N1	Y
0	ON	OFF	1
1	OFF	ON	0

CMOS delay: resistance

- Resistance:
 - Function of:
 - resistivity r , thickness t : defined by technology
 - Width W , length L : defined by designer
 - Approximate ON transistor with a resistor
 - $R = r' L/W$
 - L is usually minimum; change only W

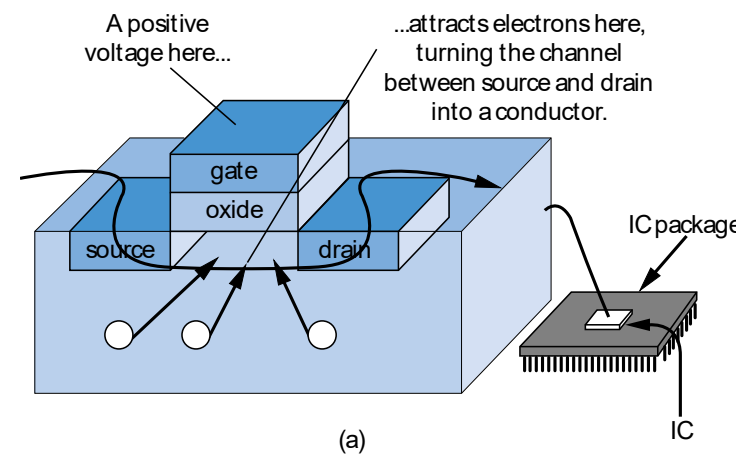
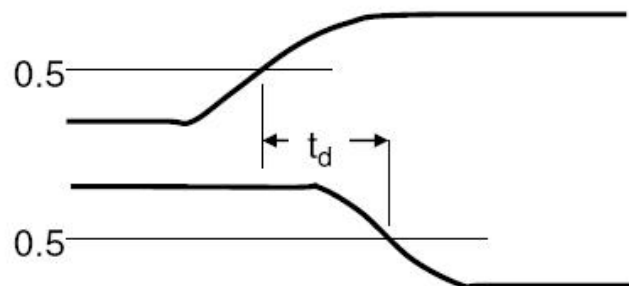


A 3D diagram of a rectangular resistor. The width is labeled W , the length is labeled L , and the thickness is labeled t . An arrow labeled I indicates current flowing through the resistor.

$$R = \frac{\rho L}{tW} = \frac{\rho}{t} \frac{L}{W}$$

CMOS delay: capacitance & timing estimates

- Capacitor
 - Stores charge = (capacitance C ; voltage V)
 - Current: $I = \frac{dQ}{dt} = C \frac{dV}{dt}$
- Timing estimate
 - $t_d = \frac{C \Delta V}{I} = \frac{C \Delta V}{\mu_n C_{ox} \frac{W}{L} (V_{DD} - V_{th})}$
- Delay: time to go from 50% to 50% of waveform

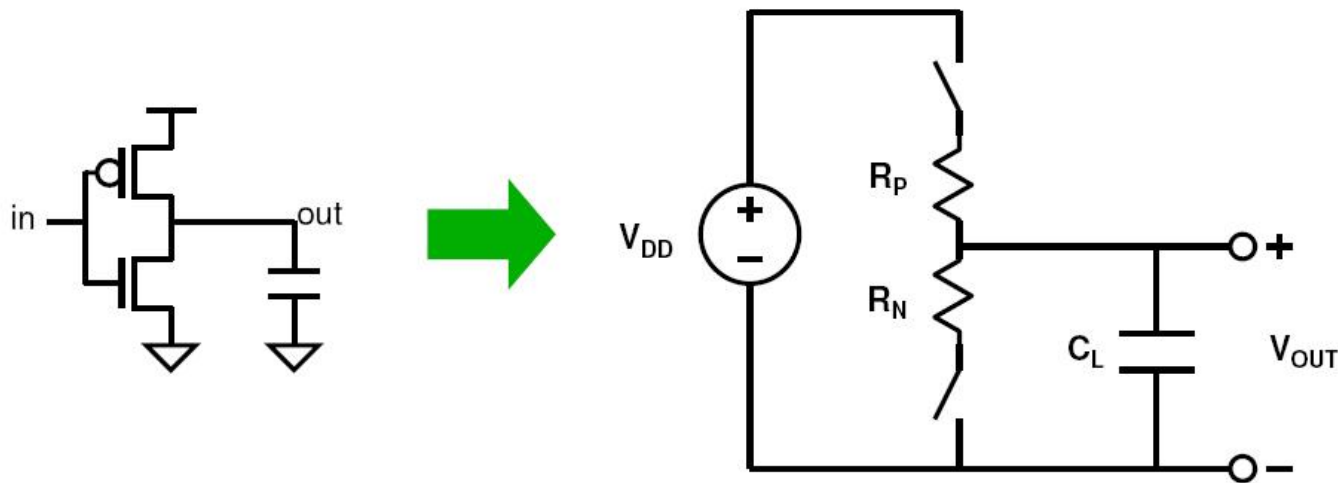


Charge/discharge in CMOS

- Calculate on resistance
- Calculate capacitance of the gates circuit is driving
- Get RC delay & use it as an estimate of circuit delay

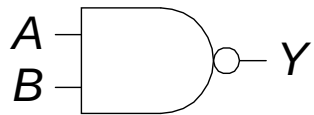
$$= (1 - \frac{V_{OUT}}{V_{DD}})$$

- $R_p \sim 2R_N$



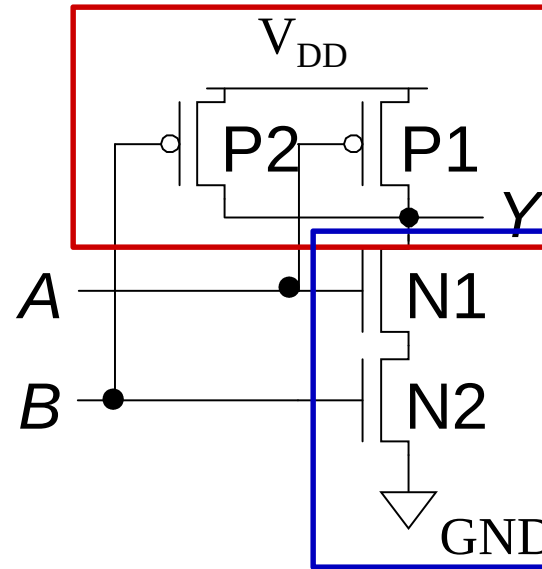
CMOS Two Input NAND Gate

NAND



$$Y = \overline{AB}$$

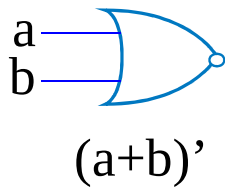
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0



A	B	P1	P2	N1	N2	Y
0	0	ON	ON	OFF	OFF	1
0	1	ON	OFF	OFF	ON	1
1	0	OFF	ON	ON	OFF	1
1	1	OFF	OFF	ON	ON	0

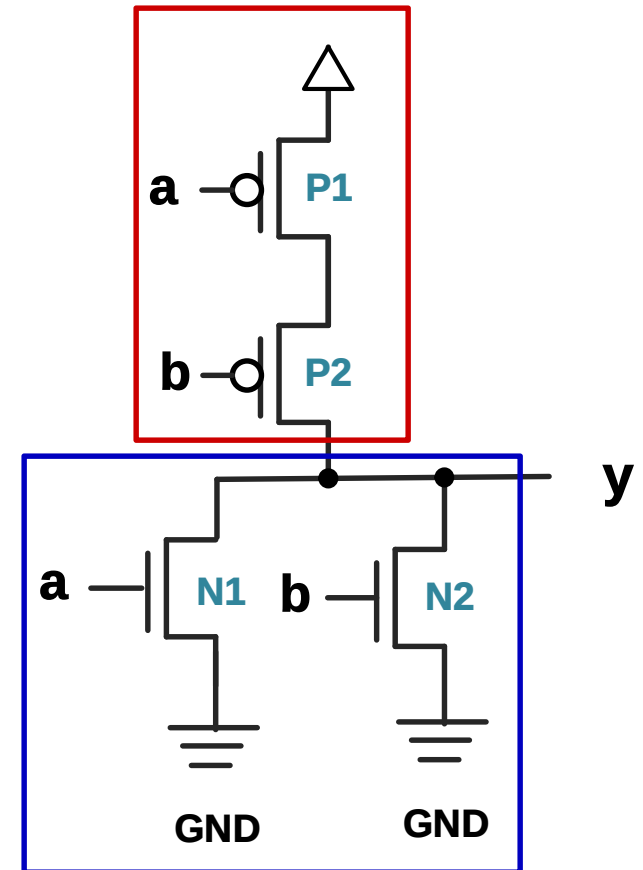
CMOS Two Input NOR Gate

NOR



a	b	NOR
0	0	1
0	1	0
1	0	0
1	1	0

A	B	P1	P2	N1	N2	Y
0	0	ON	ON	OFF	OFF	1
0	1	ON	OFF	OFF	ON	0
1	0	OFF	ON	ON	OFF	0
1	1	OFF	OFF	ON	ON	0



Rules for making gates

The Mathematical Method

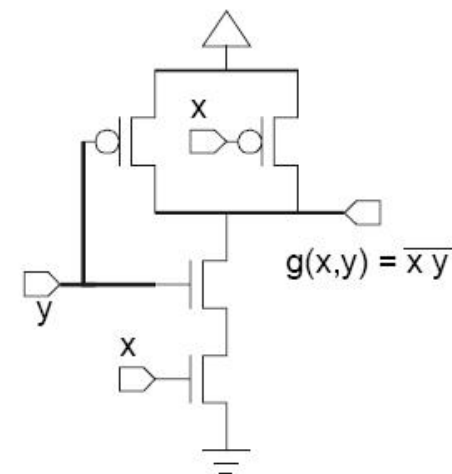
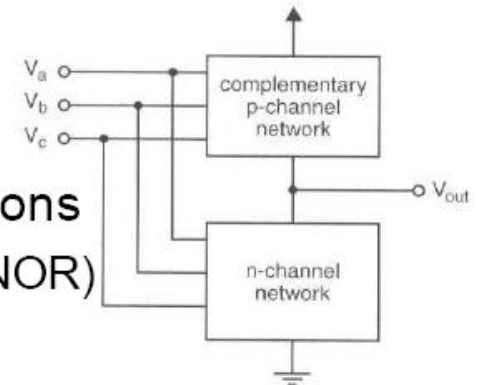
- Given a logic function
 $F = f(a, b, c)$
- Reduce (using DeMorgan) to eliminate inverted operations
 - inverted variables are OK, but not operations (NAND, NOR)
- Form pMOS network by complementing the **inputs**
 $F_p = f(\overline{a}, \overline{b}, \overline{c})$
- Form the nMOS network by complementing the **output**
 $F_n = \overline{f(a, b, c)} = \overline{F}$
- Construct F_n and F_p using AND/OR series/parallel MOSFET structures
 - series = AND, parallel = OR

•EXAMPLE:

$$F = \overline{ab} \Rightarrow$$

$$F_p = \overline{\overline{a} \overline{b}} = a + b; \quad \text{OR/parallel}$$

$$F_n = \overline{\overline{ab}} = ab; \quad \text{AND/series}$$



Another way of making CMOS gates

- Reducing Logic Functions

- fewest operations $\Rightarrow$ fewest txs
- minimized function to eliminate txs
- Example: $x y + x z + x v = x (y + z + v)$

5 operations:
3 AND, 2 OR

txs = 30

3 operations:
1 AND, 2 OR

txs = 18

$$\underline{AND} = \underline{NAND} + \underline{INV}$$

6 txs 4 txs 2 txs

$$\underline{OR} = \underline{NOR} + \underline{INV}$$

6 txs 4 txs 2 txs

- Suggested approach to implement a CMOS logic function

- create nMOS network
 - invert output
 - reduce function, use DeMorgan to eliminate NANDs and NORs
 - implement using **series** for AND and **parallel** for OR
- create pMOS network
 - complement each operation in nMOS network

CMOS Example

- Construct the function below in CMOS

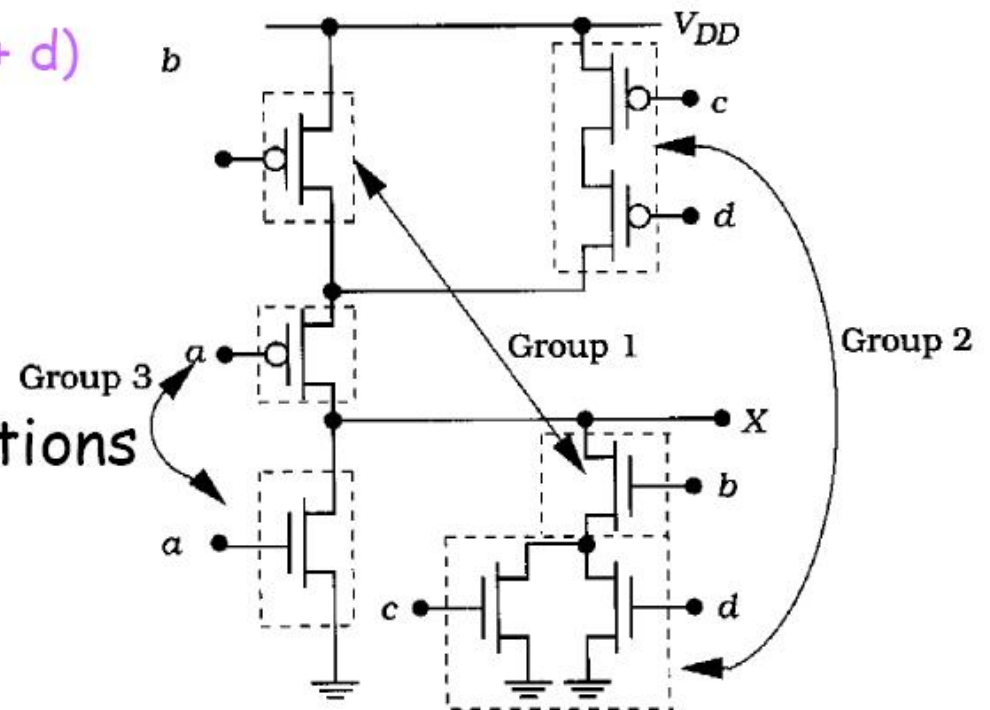
$$F = \overline{a + b \cdot (c + d)}; \text{ remember AND operations occur before OR}$$

- Step 1, invert output and find nMOS

- nMOS; implement $a + b \cdot (c + d)$
- Group 1: c & d in parallel
- Group 2: b in series with G1
- Group 3: a parallel to G2

- Step 2, complement operations

- pMOS
- Group 1: c & d in series
- Group 2: b parallel to G1
- Group 3: a in series with G2



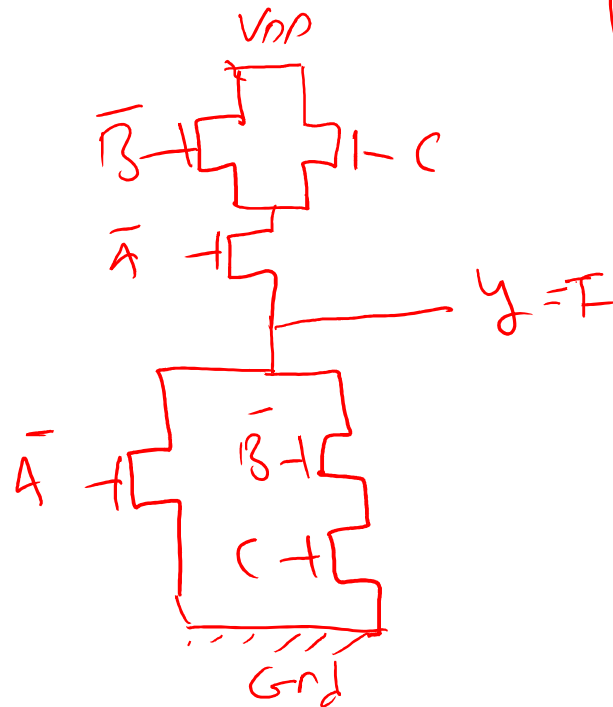
A CMOS design example

- Implement F using CMOS: $F = A * (B + C')$

$$F_n = \bar{F} = \overline{A * (B + C')} = \bar{A} + (\bar{B} + \bar{C}) = \bar{A} + \bar{B} * C$$

$\left\{ \begin{array}{l} \bar{B} \& C \text{ series} \\ \text{then parallel with } \bar{A} \end{array} \right.$

$$F_p = F(\bar{A}, \bar{B}, \bar{C}) = \bar{A} * (\bar{B} + C) \Rightarrow \left\{ \begin{array}{l} \bar{B}, C \text{ parallel} \\ \text{series with } \bar{A} \end{array} \right.$$



What we've covered thus far



- Delay estimates
- Transistor design
- Building basic gates from CMOS