ELEC 403

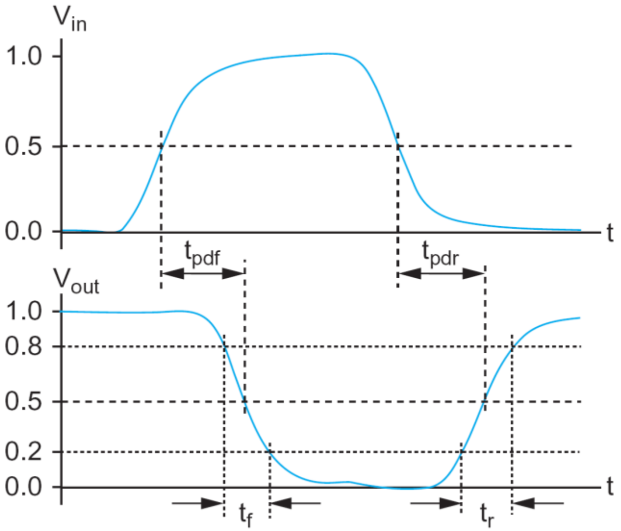
Delay & Static CMOS Logic

Lecture 8 & 9

Reza Molavi
Dept. of ECE
University of British Columbia
reza@ece.ubc.ca

Slides Courtesy : Dr. Res Saleh (UBC), Dr. Sudip Shekhar (UBC), Dr. B. Razavi (UCLA)

Different Delay Definitions



"Delay" → t_{pd}
Max-time → Propagation time
Min-time → Contamination time

Rise Time (t_r, t_{LH}): time taken to rise from 10% to 90% (sometimes 20% to 80%) **Fall Time (t_f, t_{HL})**: time taken to drop from 90% to 10% (sometimes 80% to 20%) **Edge Rate (t_{rf})**: (t_r + t_f)/2.

Rise propagation delay (t_{pdr}, t_{pLH}): maximum time from the input crossing 50% to output crossing 50%, with output rising

Fall propagation delay (t_{pdf}, t_{pHL}): maximum time from the input crossing 50% to output crossing 50%, with output falling

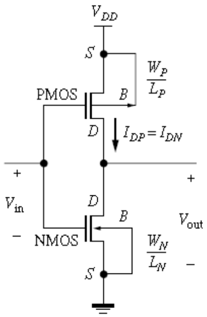
Propagation Delay (t_{pd}): (t_{pHL} + t_{pLH})/2.

Contamination Delay (t_{cd}): minimum time from the input crossing 50% to the output crossing 50%.

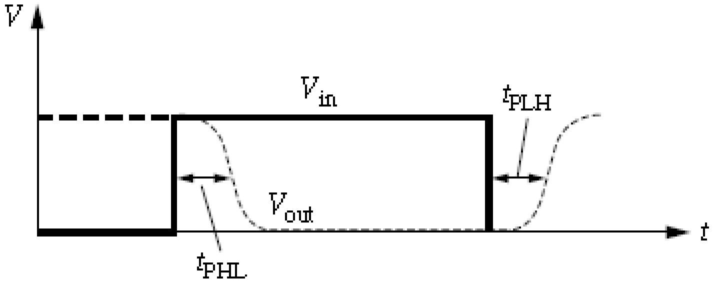
Inverter – Sizing

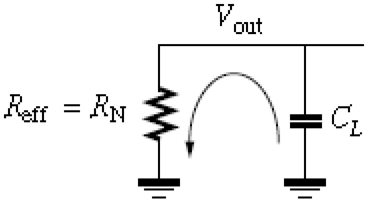
- In ratioed logic families (such as diode-connected load or pseudo NMOS, V_{OL} is a priority so the Size of load is mainly determined by the choice of V_{OL}

- In non-ratioed logic families (such CMOS inverters) propagation delay is important



Speed





Pull-down (output capacitor discharge)

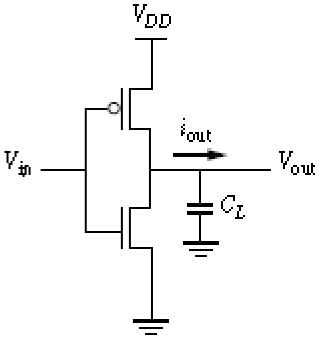
through the NMOS transistor

$$V_{out}(t) = V_{DD}\, e^{-t/R_N C_L}$$

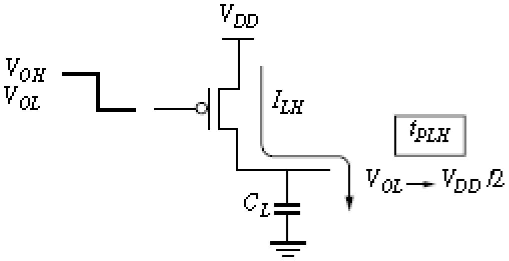
T_{PHL} is the time it take for the capacitor to discharge to 50% of the final value

(Board notes)

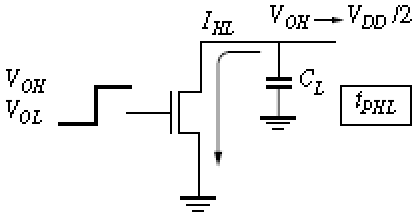
Effective Resistance Calculations



(a)



(b)



(c)

Calculate the delay from the charge/discharge of capacitor

$$I = C_L \frac{dV}{dt} \rightarrow \Delta t \approx \frac{C_L \Delta V}{I_{DS}}$$

$$t_{PHL} = \frac{C_L (V_{DD}/2)}{I_{HL}}$$

To calculate I_{HL}

$$V_{DD} = 1.2 \text{ V}, \ V_T = 0.4 \text{ V, and } E_C L = 0.6 \text{ V}$$

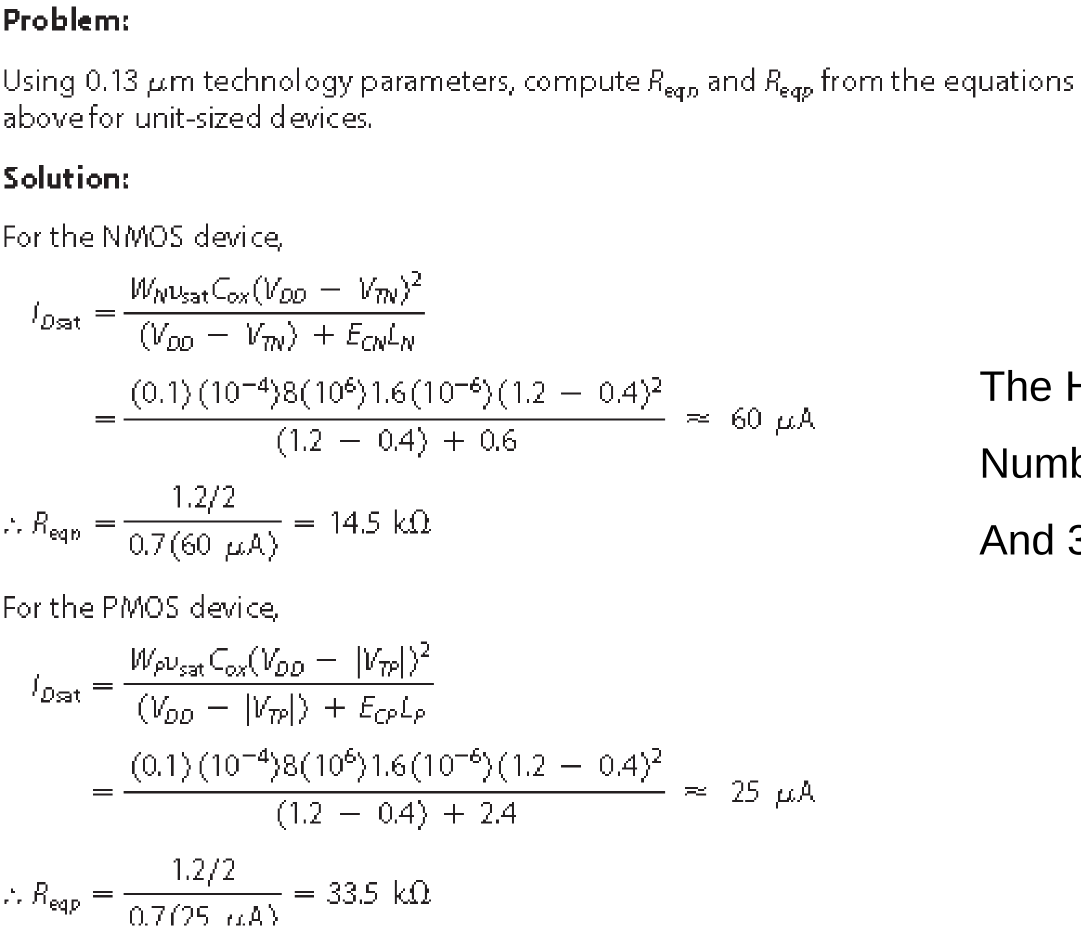
$$V_{Dsat} = \frac{(V_{GS} - V_T) E_C L}{(V_{GS} - V_T) + E_C L} = \frac{(1.2 - 0.4)(0.6)}{(1.2 - 0.4) + (0.6)} \cong 0.34 \text{ V}$$

NMOS remains in Saturation throughout T_{PHL}

$$t_{PHL} = \frac{C_L (V_{DD}/2)}{(I_{Dsat})_n}$$

$$R_N = \frac{V_{DD}/2}{0.7 (I_{Dsat})_n}$$

Effective Resistance Calculations



Problem:

Using 0.13 μm technology parameters, compute R_{eqn} and R_{eqp} from the equations above for unit-sized devices.

Solution:

For the NMOS device,

$$I_{Dsat} = \frac{W_N \upsilon_{sat} C_{ox} (V_{DD} - V_{TN})^2}{(V_{DD} - V_{TN}) + E_{CN} L_N}$$

$$= \frac{(0.1)(10^{-4})8(10^{6})1.6(10^{-6})(1.2 - 0.4)^2}{(1.2 - 0.4) + 0.6} \approx 60 \ \mu\text{A}$$

$$\therefore R_{eqn} = \frac{1.2/2}{0.7(60 \ \mu\text{A})} = 14.5 \ \text{k}\Omega$$

For the PMOS device,

$$I_{Dsat} = \frac{W_P \upsilon_{sat} C_{ox} (V_{DD} - |V_{TP}|)^2}{(V_{DD} - |V_{TP}|) + E_{CP} L_P}$$

$$= \frac{(0.1)(10^{-4})8(10^{6})1.6(10^{-6})(1.2 - 0.4)^2}{(1.2 - 0.4) + 2.4} \approx 25 \ \mu\text{A}$$

$$\therefore R_{eqp} = \frac{1.2/2}{0.7(25 \ \mu\text{A})} = 33.5 \ \text{k}\Omega$$

The HSPICE

Numbers of 12.5k-ohms

And 30k-ohm

Inverter – Resistance

-The value of T_{PHL} or T_{PLH} determine the value of the resistor (average on-resistance)

- Note that in the pull-down case (NMOS on) device starts in saturation (V_{DD} across its V_{DS} and might enter triode at some point, therefore it makes sense to define an effective resistance

$$R_{eqn} = 12.5\ \text{k}\Omega/\square$$

$$R_{eqp} = 30\ \text{k}\Omega/\square$$

Unit resistance of the devices (per square)

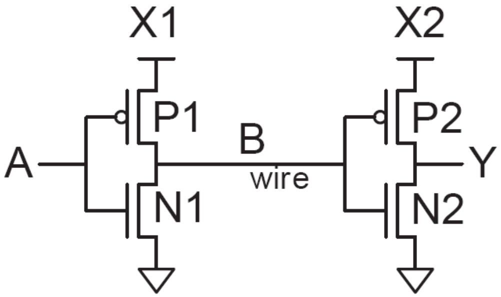
That is for a device with L=W (obtained from HSPICE simulation), note the ration 2.4, does It remind of a number?

$$R_N = R_{eqn} \times \frac{L_N}{W_N}$$

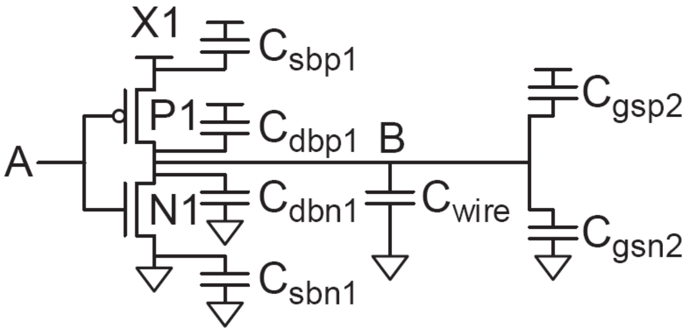
$$R_P = R_{eqp} \times \frac{L_P}{W_P}$$

Total Resistance inversely proportional to W/L (remember that in triode region we could model the transistor simply with a resistor!

Inverter- Capacitances

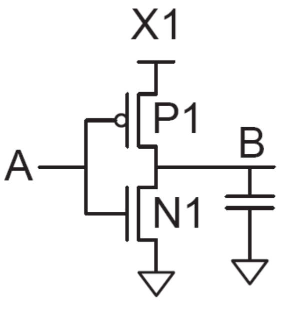


One inverter driving another



Showing all capacitances

Capacitance relevant for delay calculation, i.e.
Sum of all capacitance at the output of first inverter

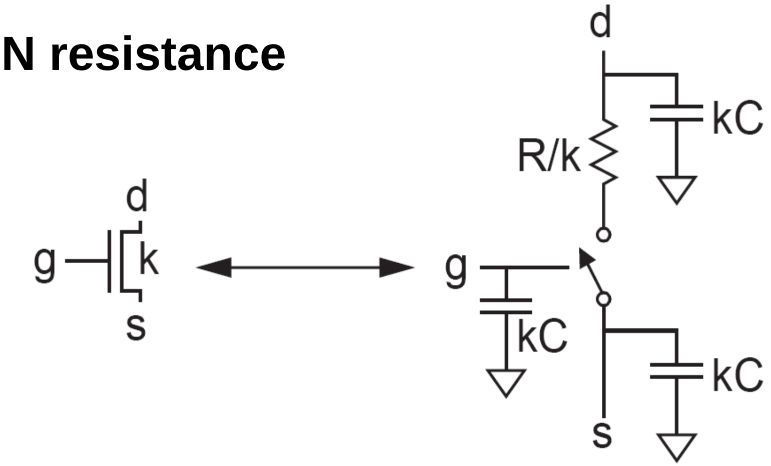


$$C_{out} = C_{dbn1} + C_{dbp1} + C_{wire} + C_{gsn2} + C_{gsp2}$$

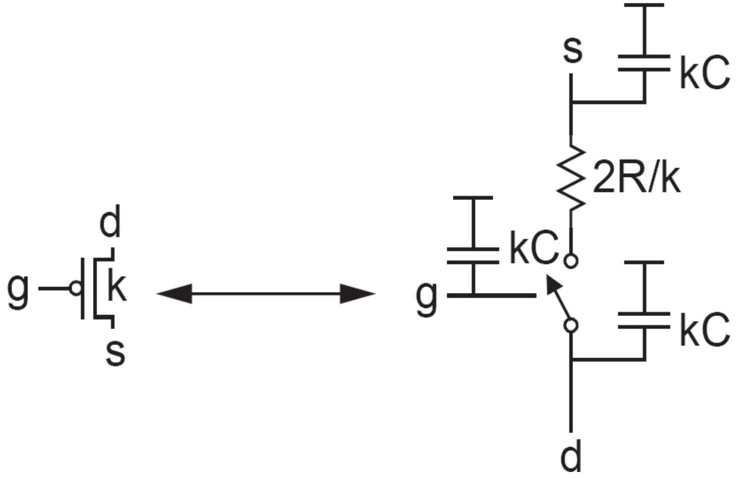
Equivalent Circuits for Transistors

- **MOS transistor == Ideal switch + cap and ON resistance**
- **Unit nMOS has res R, cap C**
- **Unit pMOS has res 2R, cap C**
- **Cap proportional to width**
- **Res inversely proportional to width**

- **Some capacitances may be shorted!**



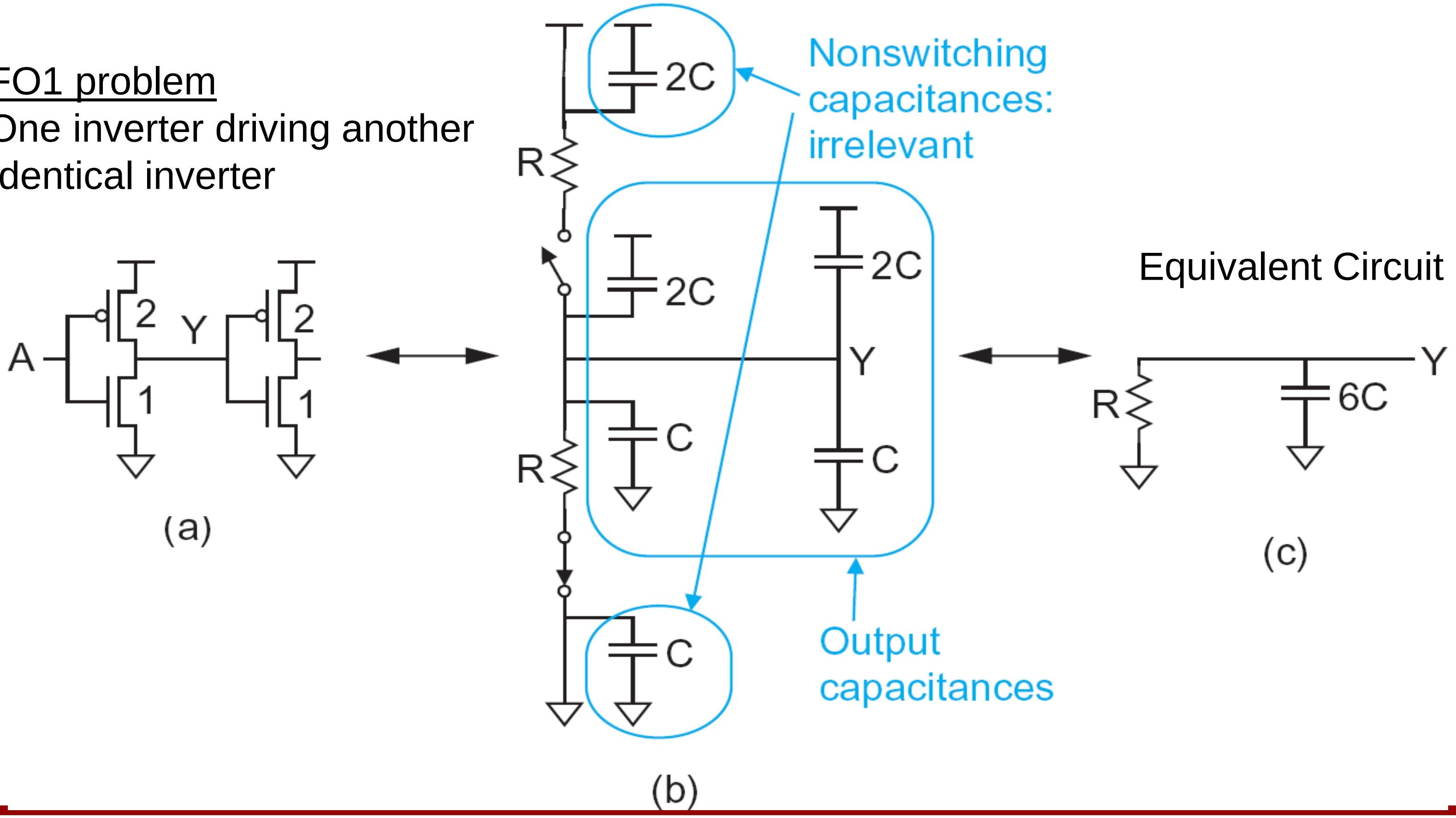
- **Capacitance**
  - **$C = C_g = C_s = C_d = 2$ fF/μm of gate width in 0.6 μm**
  - **Gradually decline to 1 fF/μm in 65 nm**



FO1 Inv

*Assuming C_{eff} = C_g= 1 fF/um (if C_g = 2fF/um we should keep them separate)

FO1 problem
One inverter driving another
Identical inverter

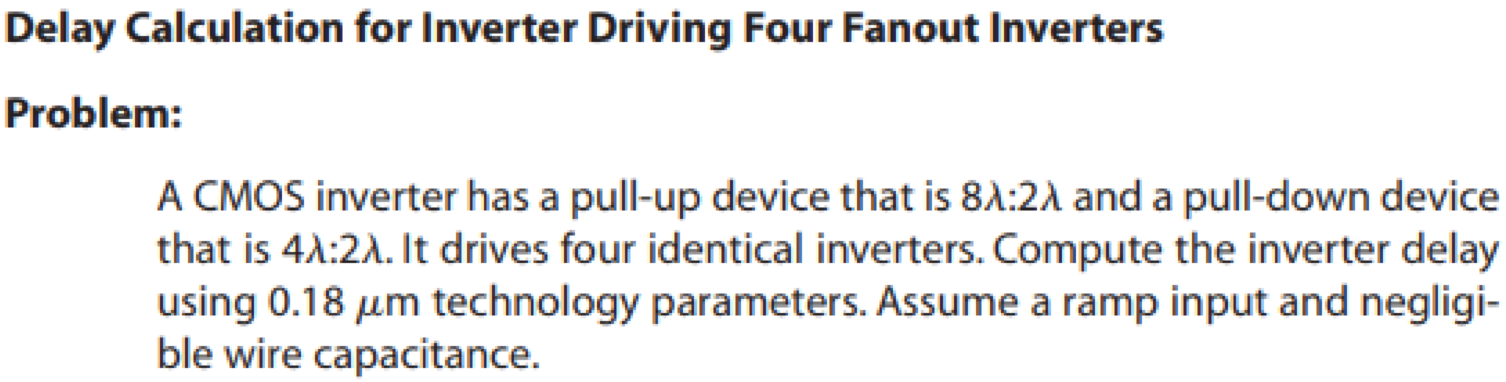


Nonswitching capacitances: irrelevant

Output capacitances

Equivalent Circuit

Delay Calculation - Example



Delay Calculation for Inverter Driving Four Fanout Inverters

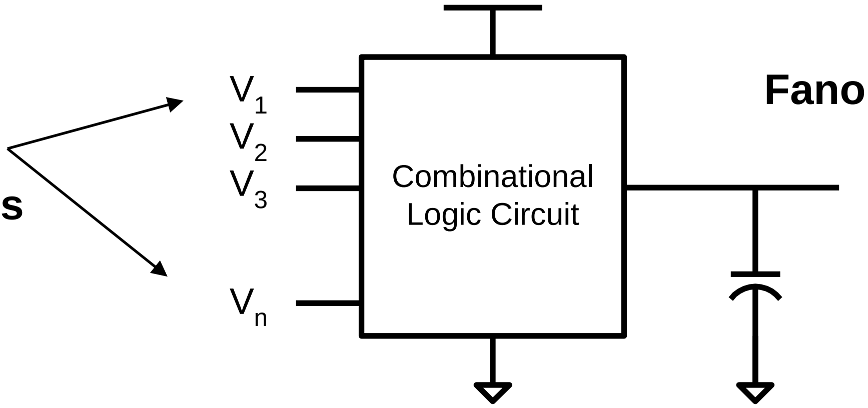
Problem:

A CMOS inverter has a pull-up device that is $8\lambda{:}2\lambda$ and a pull-down device that is $4\lambda{:}2\lambda$. It drives four identical inverters. Compute the inverter delay using 0.18 μm technology parameters. Assume a ramp input and negligible wire capacitance.

Combinational MOS Logic

- Now that we understand the logic abstraction and the properties of valid logic gates, we can consider the issues of design basic building blocks of digital systems
- Typical combinational gate is a multiple input – single output system
- Performs Boolean operations on multiple input variables, drives one or more gates
- Design parameters and considerations:
  - Propagation delay
  - Static and dynamic power
  - Area
  - Noise margins (VTC)

Fanins = no. of inputs



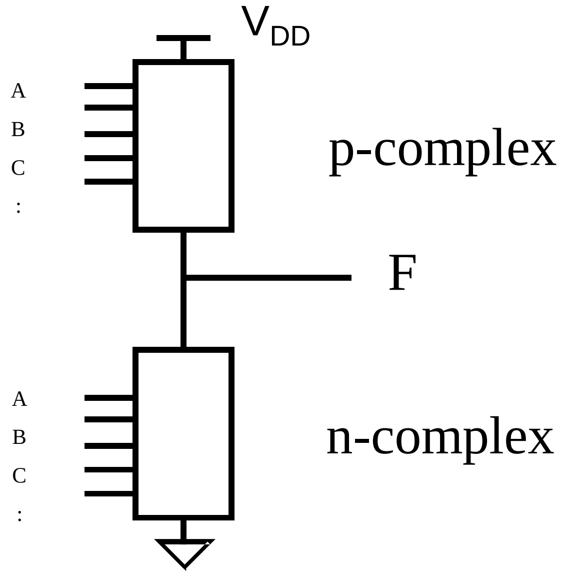
Fanouts = no. of gates driven

$V_{out} = f\ (V_1, V_2, V_3, \ldots V_n)$

Pull-up and Pull-down Networks

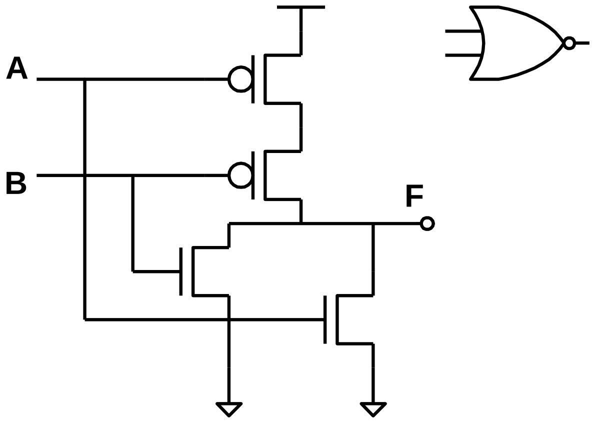
Basic Structure

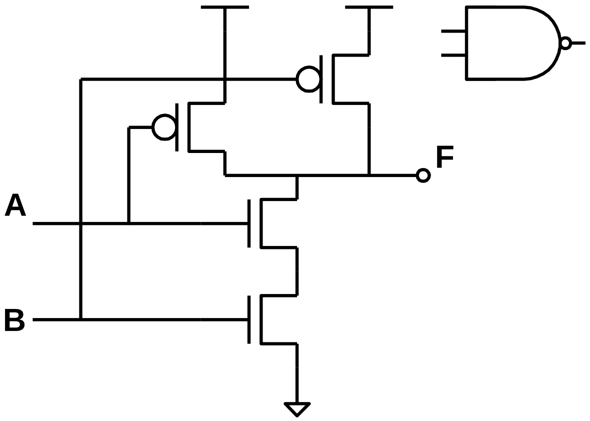
Of all CMOS logic gates



- PMOS pull-up and NMOS pull-down networks are duals of each other
- Configuration of pull-up and pull-down networks create a current connection from the output to *either* Vdd or Gnd, based on the inputs
- PMOS devices have lower drive capability and thus require wider devices to achieve the same on-resistance as its pull-down counterpart

Static CMOS Logic Gates





- These are the most common type of static gates
- Can implement any Boolean expression with these two gates
- Why is static CMOS so popular?
  - It’s very robust!
  - it will eventually produce the right answer
  - Power, shrinking V_{DD}, more circuit noise, process variations, etc. limit use of other design styles

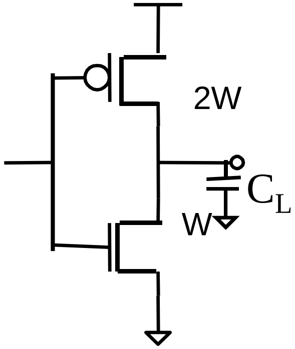
More Properties of Static CMOS Logic

- Fully complementary
- Low static power dissipation!
- Outputs swing full rail – Vdd (V_{OH}) to Gnd (V_{OL})
- Works fine at low Vdd voltages
  - But lower Vdd = less current = slower speed
- Combinational operation
  - Feed it some inputs, wait some delay, result comes out
  - No clocks required for normal operation
- Moderately good performance
  - Drive strength is proportional to transistor size
  - Large loads require large W
- Dual logic networks for N- and P-Channel devices

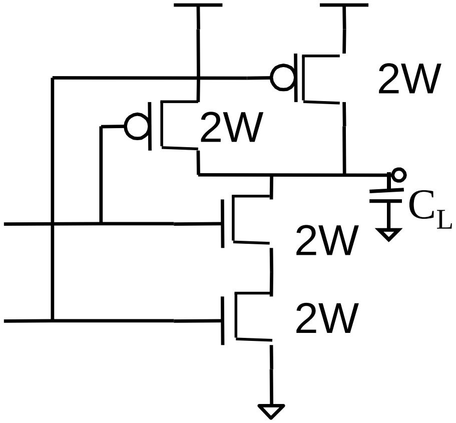
NAND and NOR Sizing

First Design Inverter

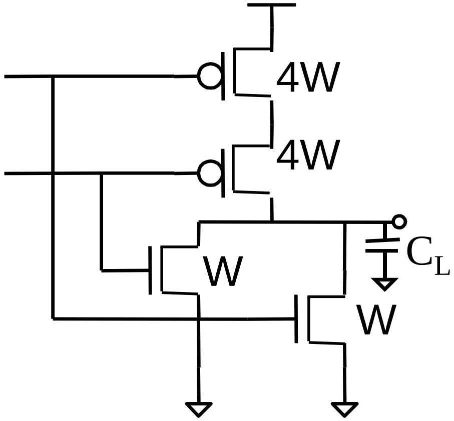
Then Map Results to Gates



(a) Inverter



(b) NAND

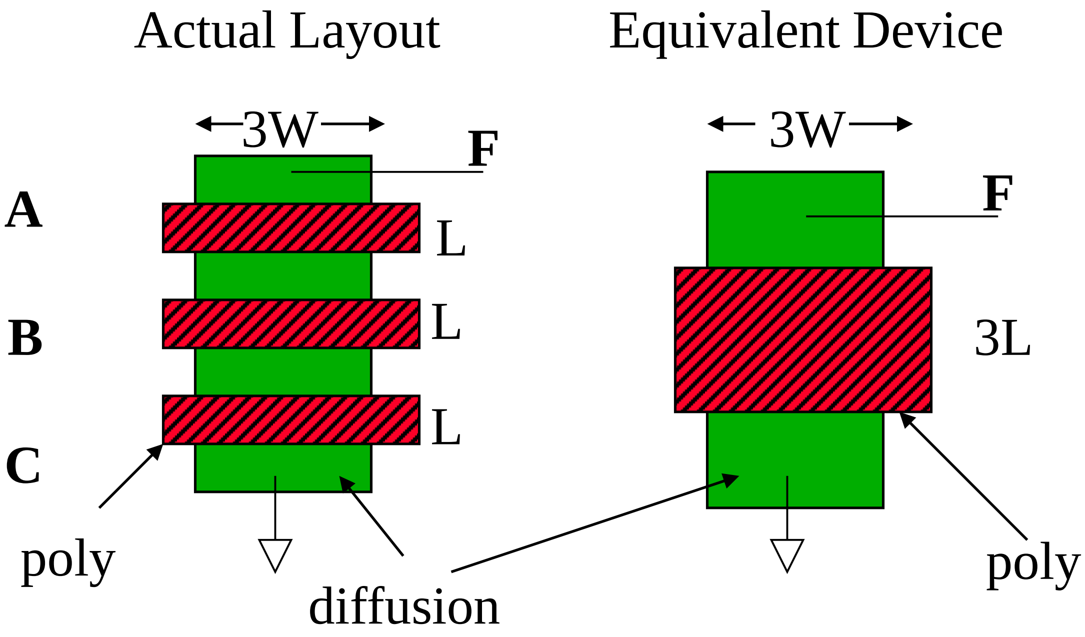


(c) NOR

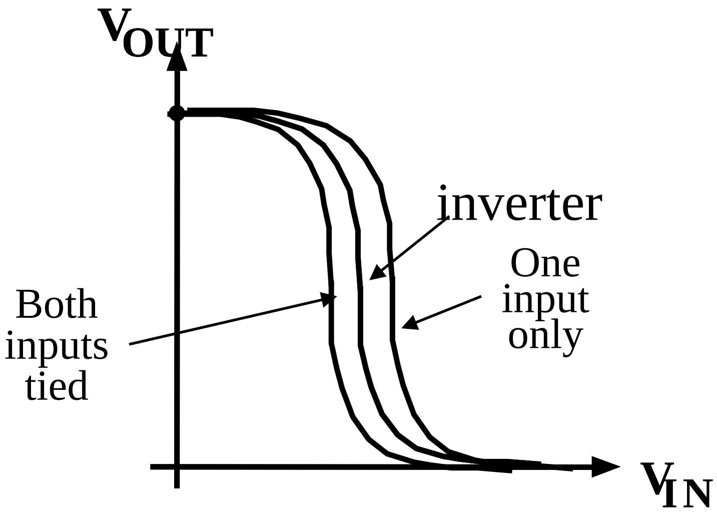
- Drive strength determined by device widths - W (assume L is minimum size)
- For the moment, consider only C_L(we are ignoring the device self-capacitance)
- Pick the right sizes for the basic inverter and then assign values to gates
- What does that mean for parallel and series combinations?
  - For parallel transistors, direct mapping from inverter
  - For series transistors, need to compute equivalent sizes

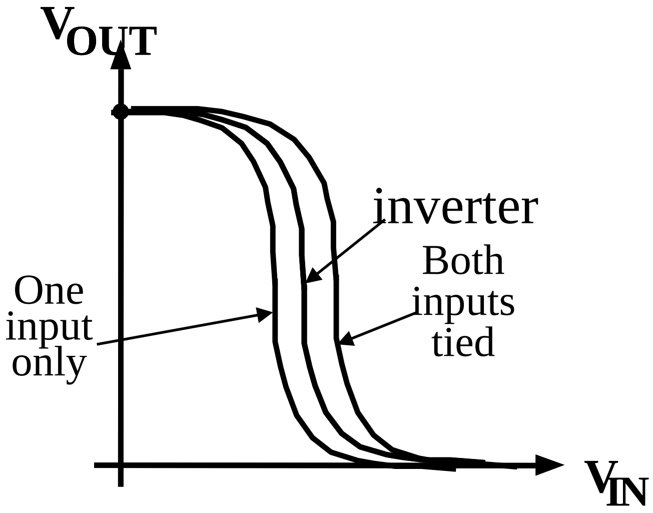
Equivalent Sizes - NAND

Consider a three-input NAND gate (NMOS portion only):

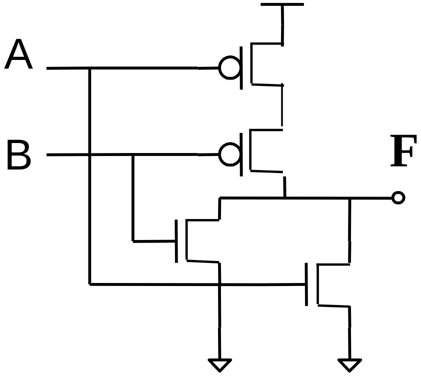


VTC and Noise Margins – NAND and NOR

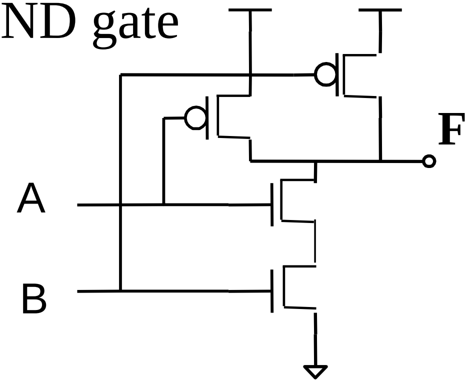




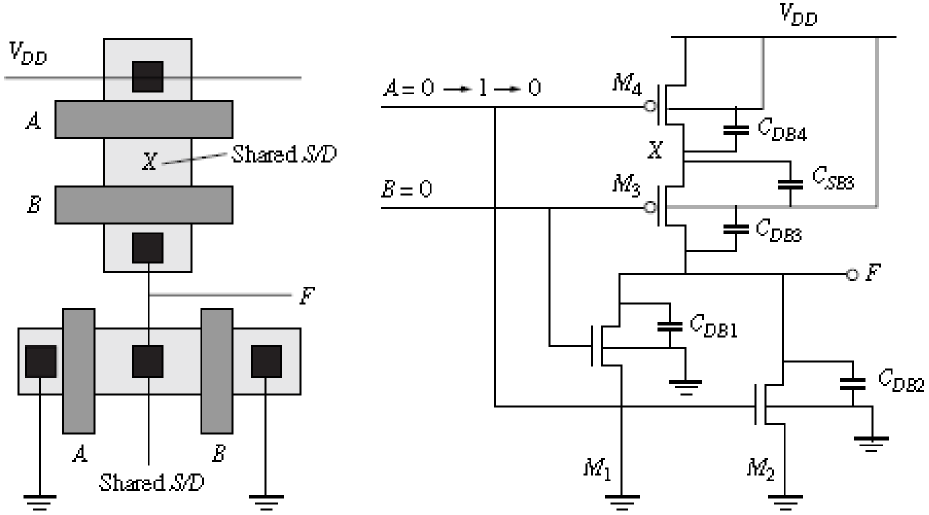
NOR gate



NAND gate



Load Capacitance for NAND and NOR



Consider *sharing*

What is worst-case capacitance calculations, why does it matter for speed calculations?

$$C_{self} = \underbrace{C_{DB1} + C_{DB2}}_{n^+ \text{shared } S/D} + C_{DB3} + \underbrace{C_{SB3} + C_{DB4}}_{p^+ \text{shared } S/D}$$

$$= C_{DB12} + C_{DB3} + C_{SDB34}$$

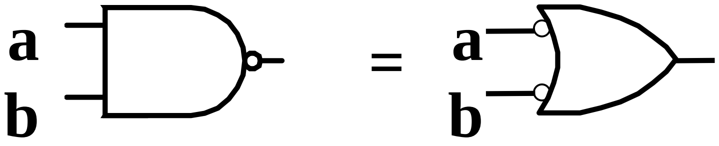
Example: what is the worst case input and output capacitance for a NAND3 CMOS gate

(Board Notes)

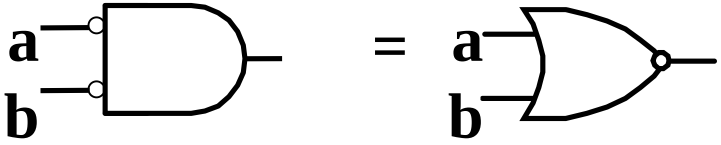
Complex Logic Circuits

- The ability to easily build complex logic gates is one of the most attractive features of MOS logic circuits
- Design principle of the pull-down network:
  - OR operations are performed by parallel connected drivers
  - AND operations are performed by series connected drivers
  - Inversion is provided by the nature of MOS circuit operation
- Don’t get too carried away… Use this knowledge wisely
  - Remember that complex functions don’t have to be implemented with a single gate
  - Can break up very complicated Boolean expressions into a cascade of gate stages
  - Limit series stacks to 3~4
- We will use De Morgan’s Law to build the dual networks

Review of DeMorgan’s Law



$\overline{ab} = \overline{a} + \overline{b}$



$\overline{a}\,\overline{b} = \overline{a + b}$

- De Morgan’s theorem:

  The complement of any logic function is found by complementing all input variables and replacing all AND operations with OR and all OR operations with AND

Circuit Design Rule

- Use De Morgan’s law to find the complement of a function for the pull-down network
- Use Duality to find the pull-up network

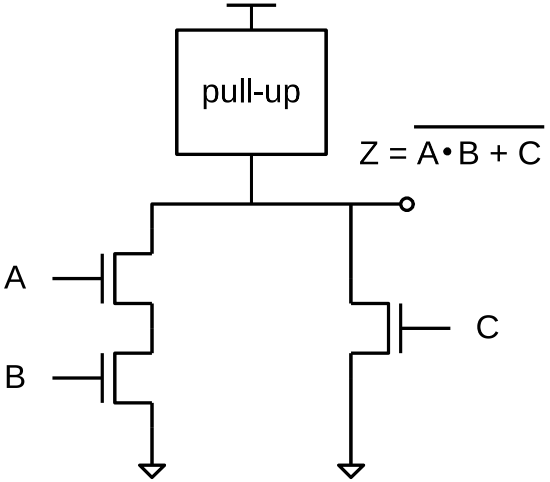
Complex CMOS Gate Design Example

- Implement an AND-OR-INVERT (AOI) function

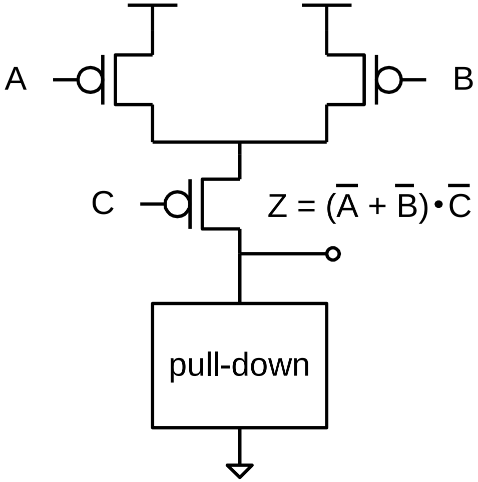
  $Z = \overline{(A \bullet B + C)}$

- Get the expression into forms that enable easy implementation of pull-up and pull-down networks

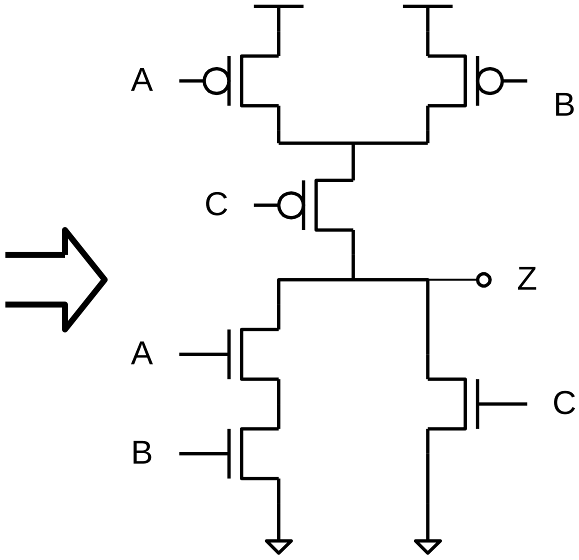
  $\overline{Z} = (A \bullet B + C)$ $\qquad$ $Z = (\overline{A} + \overline{B}) \bullet \overline{C}$



Complement of Z

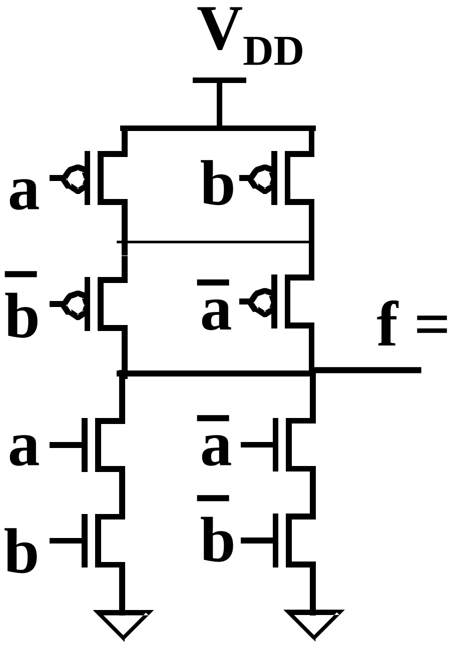


Dual of Complement of Z

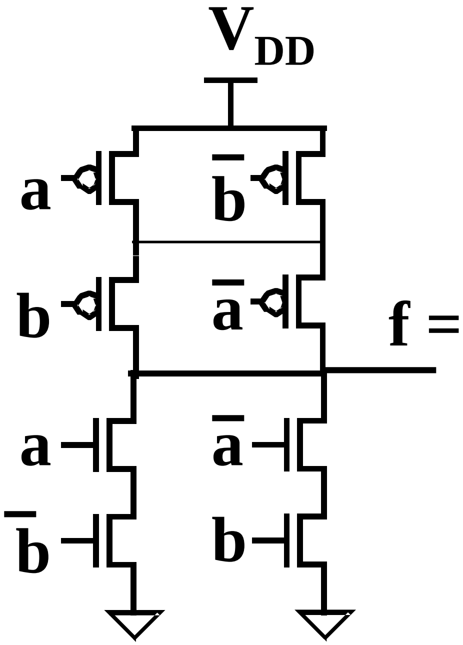


Combine Results

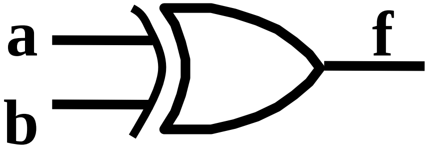
XOR and XNOR Gates

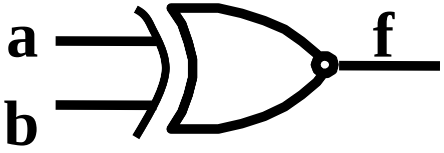


$f = \bar{a}b + a\bar{b}$

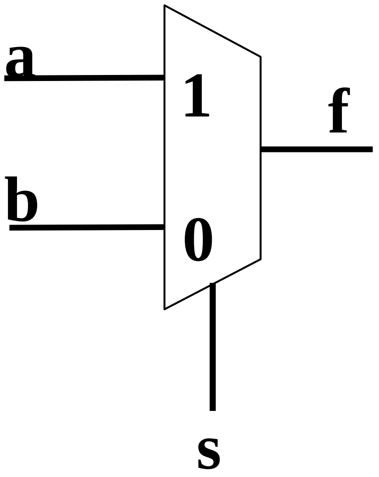


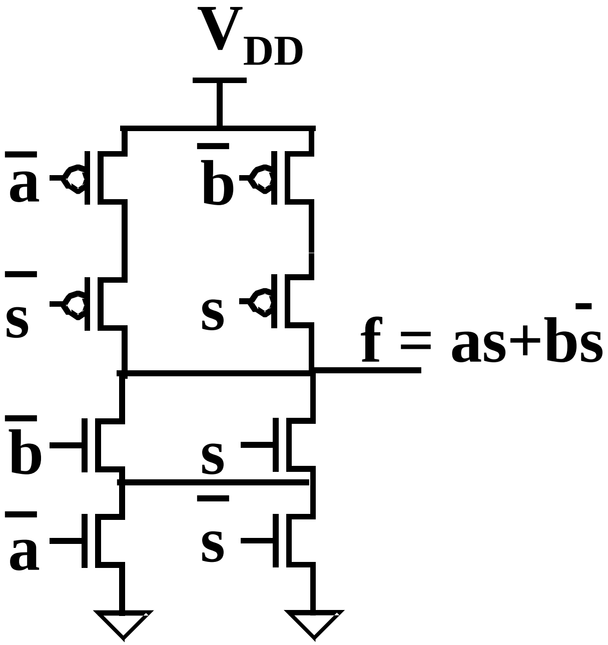
$f = ab + \bar{a}\,\bar{b}$





CMOS Multiplexer

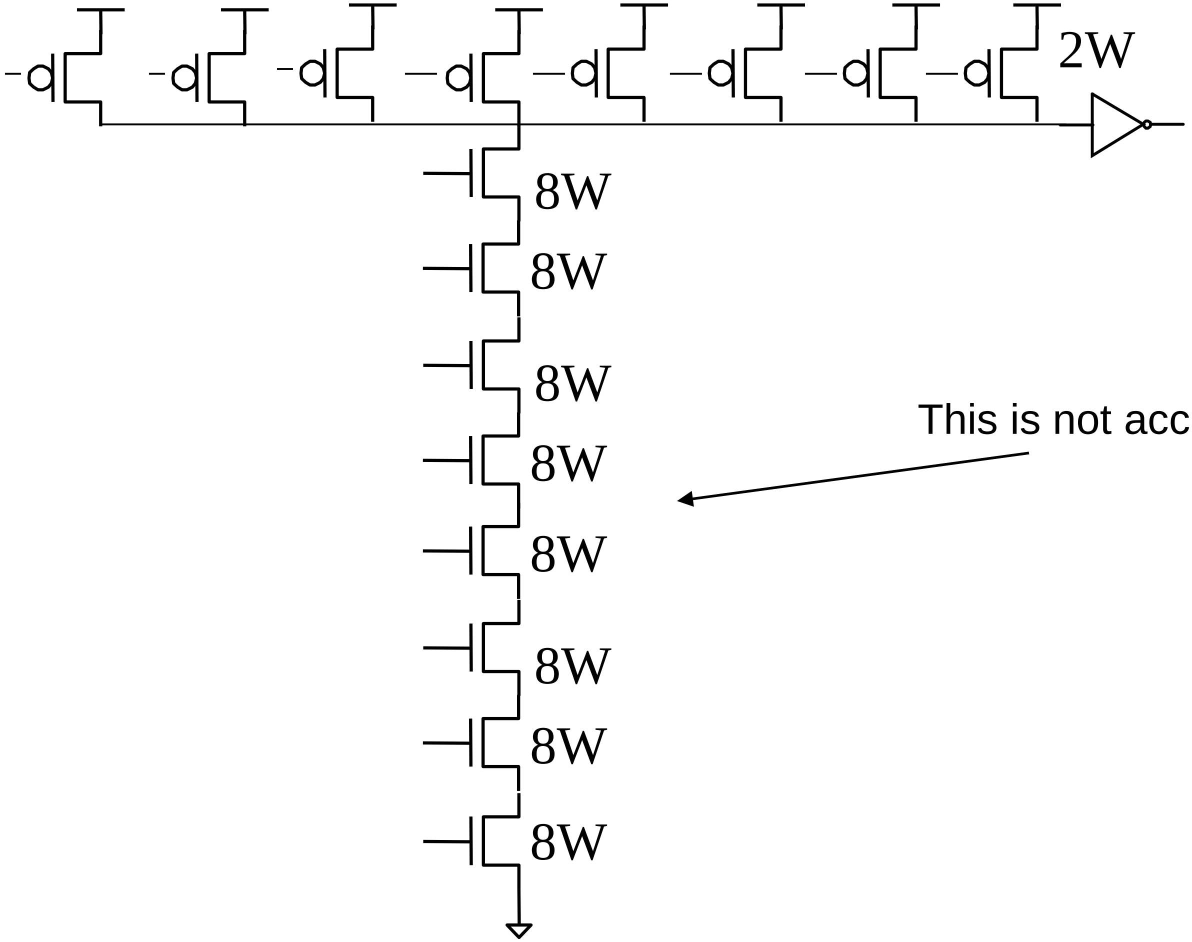




The Bad News

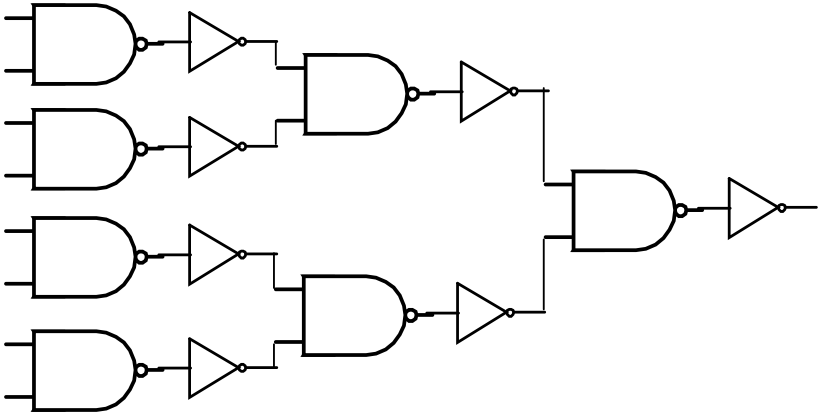
- Slows down dramatically for large fanins due to long series stack of transistors
  - fanin = number of inputs
  - PMOS series stacks worse than nMOS series stacks
- Large number of transistors
  - 2n devices for n-input NAND
  - At least 2 devices per input
- Bigger layout
  - n+ to p+ spacing rule and well spacing rule
  - Large device sizes required to counteract series stack
- Limit the fanin to 3 or 4…or delay and area will be too large

Eight-Input AND gate

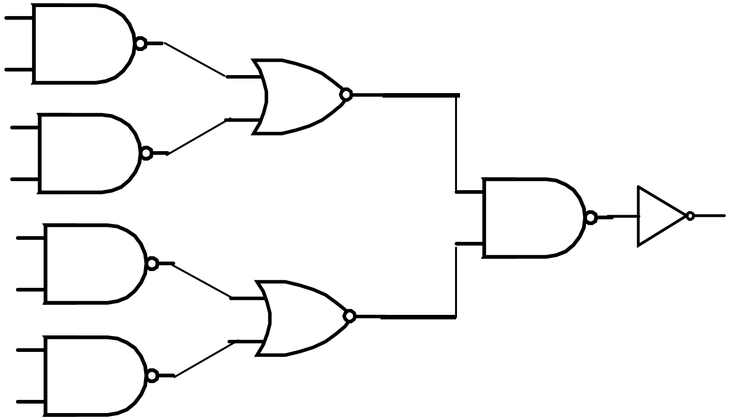


This is not acceptable

Multi-level Logic Implementations



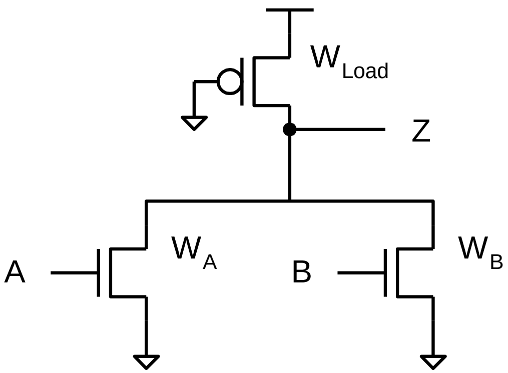
NAND2-INV-NAND2-INV

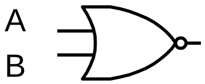


NAND2-NOR2-NAND2-INV

- There are many more options to try
- Better than stacked version but signal has to travel through multiple stages

Pseudo-NMOS Logic – NOR gate



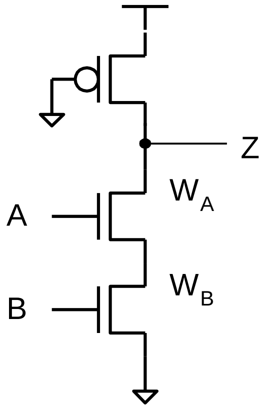


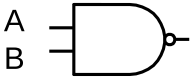
$Z = \overline{A+B}$

A	B	Z
0	0	1
0	1	0
1	0	0
1	1	**0**

- Design issues:
  - Sizing Ratio
    - Ratio pull-up to pull-down (V_{OL} & V_{OH})
    - Propagation delay
  - Subthreshold current can degrade V_{OH} slightly
  - V_{OL} decreases as more devices turning on

Pseudo-nMOS Logic – NAND Gate





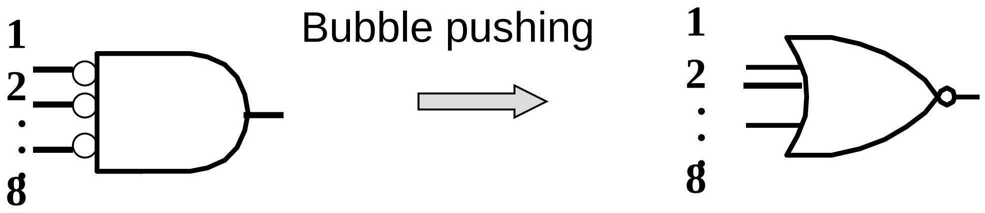
$Z = \overline{A \bullet B}$

A	B	Z
0	0	1
0	1	1
1	0	1
1	1	0

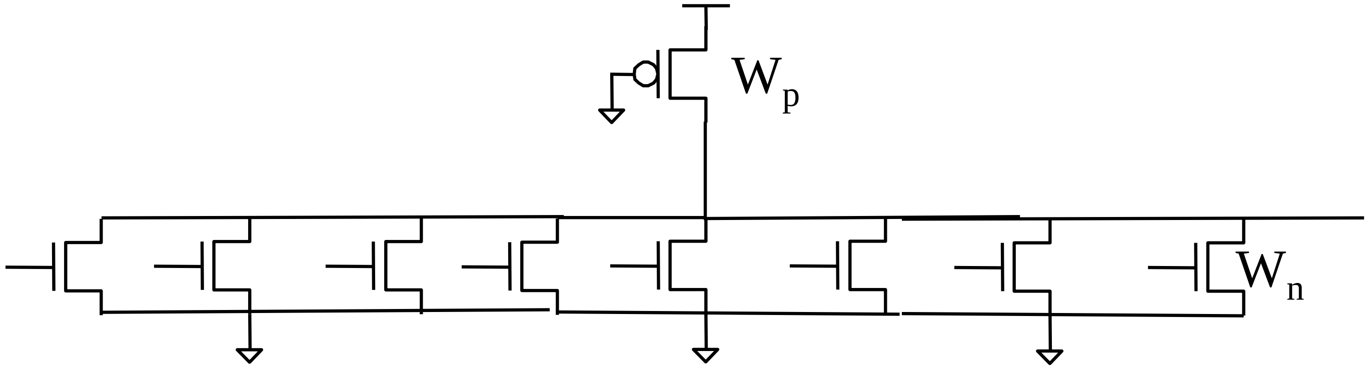
- Issues
  - Sizing Ratio
    - Need to make pull-down devices wider
  - Parasitic cap goes up with bigger devices
  - Lower devices in stack slower compared to upper ones because they see more capacitance

Back to AND8 Option - Use Pseudo-NMOS

Inputs are
Inverted, not
shown here



Bubble pushing



No need for long stack of devices

Properties of Static Pseudo-NMOS Gates

- DC power
  - always conducting current when output is low
- V_{OL} and V_{OH} depend on sizing ratio and input states
- Poor low-to-high transition
- Large fanin NAND gates tend to get big due to ratioing
- As transistor count increases, power consumption is too high
  - Cannot use this approach for all gates on the chip
- But what are its advantages?
  - Good for wide NOR structures
    - Memory decoder
  - Smaller number of transistors (area) / logic function